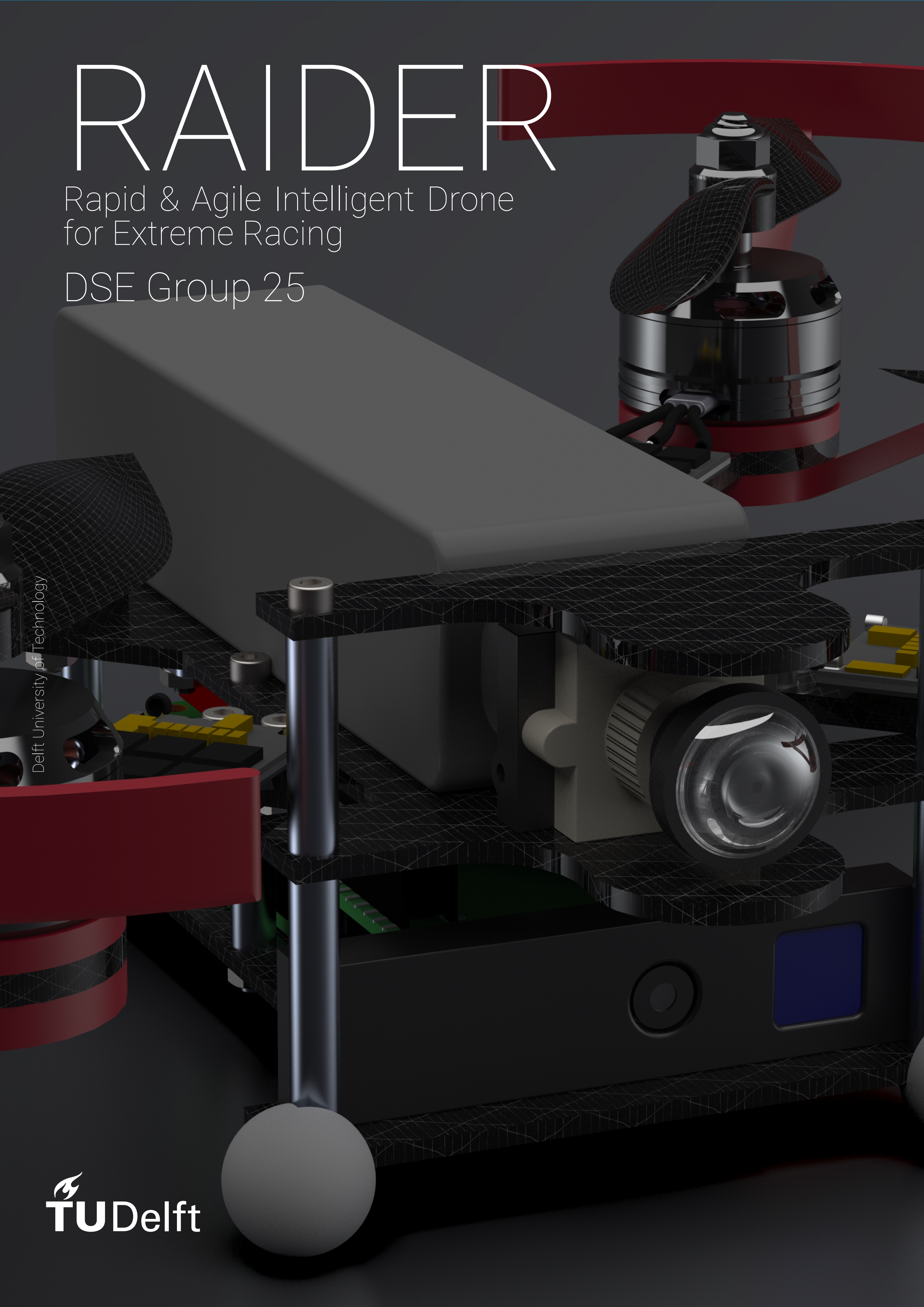


# RAIDER

Rapid & Agile Intelligent Drone
for Extreme Racing

DSE Group 25

Delft University of Technology

TUDelft

*This page is intentionally left blank.*

# RAIDER

Rapid & Agile Intelligent Drone for Extreme Racing

by

DSE Group 25

| | |
|---|---|
| Matteo Barbera | 4438590 |
| Alberto Bonifazi | 4431057 |
| Bart Bouwels | 4434560 |
| Frederic Dupon | 4465040 |
| Daniel Martini Jimenez | 4448650 |
| Rens Oudheusden | 4394593 |
| Paolo Rizzo | 4466101 |
| Thore Roepman | 4367944 |
| Ņikita Širons | 4440986 |

in partial fulfillment of the requirements for the degree of

**Bachelor of Science**
in Aerospace Engineering, AE3200 Design Synthesis Excercise

at the Delft University of Technology,

Version: 1.0
Publication date : July 3, 2018

| | | |
|---|---|---|
| Supervisor: | Dr. ir. C. De Wagter, | TU Delft |
| Coaches: | Dr. ir. R. Fónod, | TU Delft |
| | ir. N. Lavalette, | TU Delft |

# Preface

Although unmanned aerial vehicles date back to the 19th century and are widely used in many industries, full autonomy has yet to be reached. Currently commercially available drones have self flying capabilities and their military counter parts can even track objects for several hours. However, full autonomy can greatly improve the overall safety. Self flying smart robots can patrol the skies around airports, decreasing the amount of bird strikes. They can be used in the search for survivors after a natural disaster to cover large areas quickly and efficiently. In this need for full autonomy our team set out to participate in a competition for fully autonomous drone racing. The goal was not to only win, but also show the level autonomy that can already be reached using current technology.
The smart autonomous racing drone, RAIDER, has been developed by a team of nine aerospace students. The project was carried out in the light of the Design Synthesis Exercise, the final course of their bachelor's degree. The autonomous nature of the drone, together with the high integration needed between software and hardware has been a driving motivation for all members. The project started in April 2018 under the extensive supervision of ir. C. de Wagter.
In light of this project we want to thank ir. C. de Wagter for his constant guidance throughout the project and his continuous feedback on our work. We would also like to thank our two tutors Dr. ir. R. Fónod and ir. N. Lavalette for their comments on our reports and their valuable expertise during the design process. A special thanks is also rewarded to the OSCC committee for the development and execution of the Design Synthesis Exercise. The team would also like to acknowledge the contributions to this project by the members of the TUDelft MAVLab. Especially the tutoring by Shuo Li has been of great help to the control and stability of the drone. As a team we wish him the best of luck during the upcoming IROS2018 event.

*DSE Group 25*
*Delft, June 2018*

# Executive Overview

With every passing year, drones are becoming an increasingly large part of daily life. Drones are utilized by anything from filmmakers to cheaply record aerial footage, to rescue crews in different search and rescue scenarios to save countless lives. However, the technological readiness of the autonomous software has not yet reached a point where the human element can be completely removed from the loop.
One of the best ways to encourage advances in any technical field is to organize competitions. The International Conference on Intelligent Robots (IROS), has, for the last few years, organized a race specifically for autonomous drones, the Autonomous Drone Race(ADR). For the ADR, competitors are required to design, build and program a drone that can autonomously navigate through a series of gates as quickly as possible.
As in the previous iterations of the ADR, each competitor has a 20 minutes time slot to attempt to set the track record. The gates are orange with a white sheet behind the bars, and the final gate once again includes a dynamic obstacle. However, there is an additional gate, the jungle gate (gate 6). Only one of the possible entrances will be marked with the color orange on the day of the race. Last year, the winners of the competition won by completing 70% of the track in 3 minutes and 11 seconds. Group 25 has been tasked with designing a drone that would not only complete the full track, but to do it much faster.

## Project Objectives and Design Requirements

Since the goal of the project was to advance the capabilities of autonomous drones, designing designing a drone that would just win the race was not enough. It has to also be capable of significantly outperforming the competition. The project objective statement was thus defined as:

***"Win the IROS 2018 competition by designing a fast and agile autonomous drone with 9 FTE (full time equivalents) in 10 weeks."***
This objective was reflected using design requirements, with the customer expecting a high performance drone that would be capable of reaching speeds in excess of 100kmh, small turn radius, as low as 3m at 12m/s, while having a flight time of 10 minutes. Additional constrains were imposed on the repair times, cost and size. The requirements aimed to not only capture the need for robust autonomous algorithms, but also the need to be crashworthy.

## Concept Design

During the initial design phase a number of possible hardware configurations and autonomous strategies were considered. During the first literature review, it quickly became clear that most of the concepts were unfeasible. For the conceptual phase, it was decided to focus only on further exploring the tricopter and quadcopter configurations.
In terms of autonomous strategies, the situation was more complicated. The initial level of experience within the group was very low, and therefore three different strategies were analyzed, each with a different level of expected computational intensity. Every autonomous strategy was combined with either a tricopter or a quadcopter configuration, resulting into the three Top Level Concepts (TLCs) shown in Figure 1.

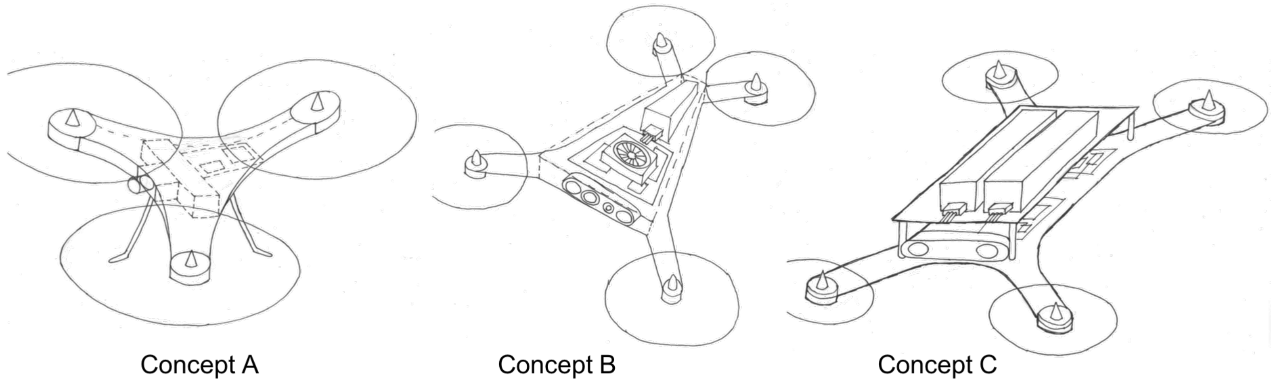


Figure 1: Sketches of the three TLCs

As shown in Figure 1, Concept A was a tricopter. It was the lightest, smallest and fastest of the three concepts. To achieve this, it relied on an lightweight algorithm that made use of the known gate location to pre-plan a path before the race, only using an RGB camera to correct in flight. All of the computations were to be performed on a Pocket Beagle board.

Concept B was a quadcopter that relied on a more tried and tested autonomous strategy: simultaneous localization and mapping (SLAM), using an RGB-D camera for sensing and an Intel i5 CPU board to be able to handle the significantly computationally heavier method.
Concept C sported a much more experimental autonomous strategy, which relied on end-to-end neural networks running on an nVidia Jetson GPU. It carried a stereo-vision camera to be able to capture high resolution images depth images. However, due to its weight, Concept C needed two batteries.

It was important that at the end this stage the most important hardware components were chosen correctly to allow for all of the programs that were needed to be run on them. Therefore, a number of preliminary calculations were carried out for each concept, such as the expected computational cost of each autonomous strategy, total mass, and flight time. Very large safety margins were employed at this stage due to the inaccuracies in the methods that were used.

## Trade-off and final concept

For the trade-off of the Top Level Concepts, a number of criteria were considered that rewarded robust and reliable algorithms, high flight performance, and crashworthiness, while penalizing design complexity. Concept C, while performing very well in terms of algorithm robustness, carried very expensive components, making it susceptible to crashes, and was also deemed very complex. Concept B scored on average very well, but because of SLAM the flight performance suffered greatly, and also carried fragile components. Concept A came out as the winner, scoring especially well in the flight performance category, due to its high top speed and small size, which allowed for a less precise autonomous strategy.
After the trade-off, it became apparent that concept A needed some more improvements. The tricopter configuration was changed to a quadcopter, as this would eliminate the need for a gimbaling rotor, a vulnerable part in concept A. Secondly an extra computer board, the Raspberry Pi Zero W, was added as this provides more camera interfaces and yields more computational power to process images. Lastly a second visual sensor, a Time of Flight (ToF) camera, the Pico Flexx, capable of recording depth information.

## Final Design

With the configuration and overall autonomous strategy finalized, it was time to dive into the detailed design of each subsystem. The team was divided into small subgroups, each responsible for a different subsystem, which focused on structures, power and propulsion, sensor fusion, stability and control, gate detection, and path planning and optimization.

### Structures

The main tasks of the structure is to survive crashes and ensure that none of the sensitive components are damaged. During the design phase four different aspects are assessed: the design of the arm, design of additional crumple zones, selection of the propeller guards, and the damping of the IMU.
In the design of the arms the most demanding load case was the impact at the tip of the arm. To assess the impact, the drone was modeled as rigid body rotating around the gate. A iterative linearized solver was written to find the solution to the complex system which includes the deformation of the arm, stress in the bolts and impact force. This led to an arm design made out of carbon fibre plates being 2.9 mm thick, and with a maximum loading stress of 550 MPa.
As this was not deemed enough to survive a crash additional crumple zones, foam balls with a radius of 7mm, were added to the bottom of the drone. During a crash the foam balls absorb the potential energy from the fall, gradually slowing the drone with a maximum deceleration of 500g, the limit for which consumer electronics are rated.
To reduce the likelihood of crashing two other measures were taken. Firstly the propellers are protected by 3D printed propeller guards surrounding more 50% of the entire drones' perimeter.Secondly the IMU, who's measurements are strongly affected by vibrations with frequencies above the sampling rate, is damped using rubber studs at 400hz, half the sampling rate and above the control frequency.

### Power and propulsion

To assess the power needs of the drone, semi-empirical equations were used, which were corrected with experimental data. By using battery specifications and propeller geometry, these equations were used to find the specifications of the motor needed in order to fulfill the requirements. To check such a motor existed, an iterative process was used where all results were validated with ECalc, an online tool used to estimate the performance of multicopter components.
The resulting configuration consisted of a Turnigy 1550 mAh 6S battery, with 5 x 4.5 inch carbon fiber propellers,

30 A electronic speed controllers, and race grade motors: the XRotor 2207-1750 KV. With these components, the drone was capable of achieving a maximum thrust to weight of 5, a top speed of 160 km/h, nominal flight time of 15 minutes, and full thrust flight time of 1 minute.

## Sensor Fusion

In order to ensure that the drone was able to accurately determine its location, orientation and location of the gates, different sensors are fused, significantly reducing noise. The drone carried 4 sensors with which it localized itself: an inertial measurement unit (IMU), a height sensor, and 2 cameras. In order to fuse all of these measurements together, 3 sensor fusion methods were explored: a 15 and a 31 state extended kalman filter (EKF), and a 15 state unscented kalman filter (UKF). In order to determine which method would fit this project best, a trade-off was done, which evaluated each option in terms of accuracy and computational cost. The 15 state UKF came out as the winner of this trade-off, as the 31 state filter, although slightly more accurate, was too computational expensive. The unscented kalman filter was chosen above the extended filter due to its higher accuracy in non linear regions, in which the drone will mostly be flying when going fast.

## Stability and Control

In order to achieve a stable and controllable flight with a quadcopter, an appropriate control system was necessary. As control solution a linear proportional integral and derivative (PID) controller was selected, as this widely used and has good performance.
A dynamical model of the drone was first constructed, after which the controller was built and tuned around it. The controller designed consisted of four PID feedback loops, all nested within one another. The inner loop controls the pitch, roll, and yaw rate of the drone. The attitude is controlled by another PID loop providing the input to the inner loop. The input to the attitude controller is given by a PID loop controlling the translational velocity, for which input is provided by the top position control.

## Gate Detection

The computer vision subsystem is essential for the drone's autonomy. The drone needs to be able to detect obstacles and recognize gates in the images. After a positive gate detection, the position of the corners was processed to calculate the relative position of the drone with the respect to the gate.
The algorithm that came out as the most promising after a literature study, was a small, 9 layer, convolutional-deconvolutional neural network (DeCNN). This network performs background removal. The DeCNN was trained on 23,000 synthetic images, generated in a CAD environment. In order to decrease the training time, the neural network was trained on the online GPU service offered by Google Cloud.
The output of the network was a grayscale image as visible in Figure 2. This image is further processed by using dilation and binary thresholding. Finally, a lightweight contrast-based algorithms was used to process this image. The gate contour was interpolated with a linear polynomial function, based on the Ramer-Douglas-Peuckert algorithm (RDP).

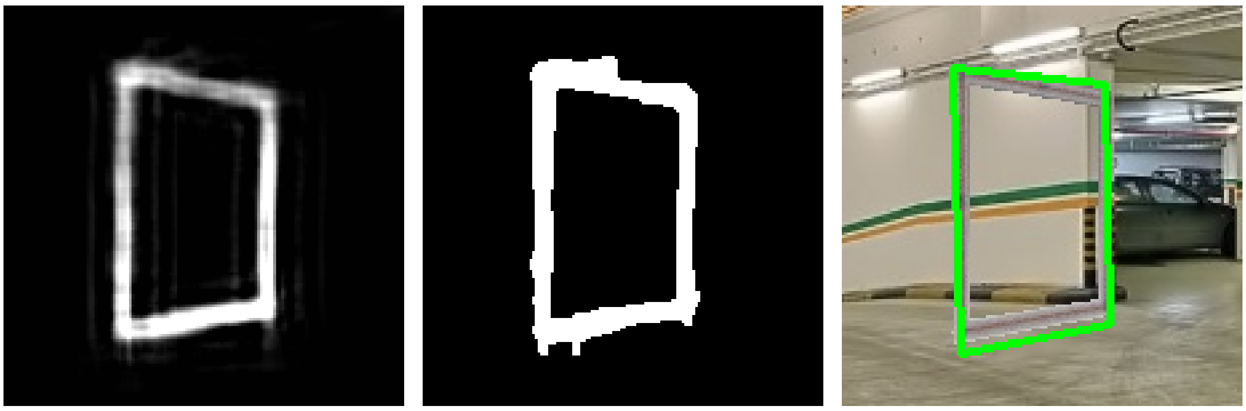


Figure 2: DeCNN-RPD Gate Detection

A benchmark analysis was applied to the DeCNN to evaluate its accuracy and processing time. The results showed that the algorithm was 67% accurate at detecting gates, but the frame rate had to be down-scaled to 3.5 Hz due to the high computational cost. This was where the Time of Flight camera came into play, which was used to detect and track obstacles at high frequencies while racing.

### Path Planning and Optimization

The purpose of path planning was to design a 3D collision free path that took into account the computational limits of the computer boards, and at the same time minimized the control inputs. None of the algorithms that were considered during literature research could quickly deal with this problem however.
Therefore, the developed strategy was based on the work of Mellinger[33], who exploited the differential flatness of quadrotors to produce kino-dynamically feasible trajectories. Differential flatness means that the control inputs could be described as a function of flat output variables (x, y, z and the yaw angle) and their derivatives. By minimizing the second derivative of the acceleration, and the yaw angle, the control inputs could be reduced.
It was also important to generate a path that would maximize the time a gate would be in view, to increase the probability of detecting the gate. The aim was to minimize the second derivative of the yaw angle to achieve smooth transitions of the yaw between the different segments of the race track. This yawing maneuver was initiated before fully passing through a gate, so that the Time of Flight camera could be used to track one of the vertical bars of the gate for localization purposes.
In order to find a nominal racing speed, the probability of completing the track was calculated as a function of the total track time. This was done taking into account IMU drift, gate detection accuracy and motion blur. This resulted in a maximum probability of 86% at 60 seconds. However, this time can be decreased at the cost of this probability.

## Design assessment

After the detailed design was finished, detailed mass, cost and power breakdowns were made which included even the mass of the bolts. The total drone weighs 714 grams, costs €1454,- and uses 164 Watt at hover; the size was 29x29x9cm. Most of the electronics are bought, whilst others needed further detailed design and custom manufacturing. The other hardware components can be cut out of single carbon fibre plate while the software needs to be converted to C or C++ for better performance.
By the means of a compliance matrix all requirements were checked and deemed passed or recognized as more work needed. 16 requirements, of which 7 are stakeholder requirements, also including one about winning the competition yet unfeasible to assess, need more work and their verification procedures are reasoned. As part of the compliance and further assessment of the requirements a sensitivity analysis of the design for nominal velocity, mass, power usage and drag coefficient was done. It was concluded the power usage, drag coefficient and nominal velocity can be increased by as much as 300% before the design starts to fail requirements. The most critical is the mass as the drone can only be made 48 grams heavier, before it can no longer meet the turning radius requirement.
During the operation of the drone, it is expected to crash many times. In order to assess how many spare parts need to be brought to the competition, a failure diagram was made. Each failure mode was then given a probability, based upon which the amount of spare parts that are needed for this failure was calculated. RAIDER's safety was also analyzed. The team found that impact with a head was the most dangerous scenario that could happen. As a result of this safety analysis, some changes were made to the design. This is for instance he reason carbon fiber propellers will be used.
The risks of failure were not the only ones that were analyzed. There are also other risks involved with this project. These were both identified and mitigated. The result of this is for instance the fact that the drone has propeller guards. These allow the drone to stay up in the air after a low-speed collision. The sustainability was also analyzed. The resulting graphs are shown in chapter 19. They show that the drone does not just comply with the requirements, but actually exceeds them.

## Post DSE

With the help of a Gantt chart a detailed production and testing plannings is made that will ensure the drone is at its highest potential during the IROS2018 itself. A more detailed integrated simulation plan including a ROS simulation is proposed to find software related bugs, assess the performance of algorithms and ensure the drone is able to fly the track autonomous.
As is often the case for high-tech products, the developments that are made for racing will eventually make their way onto the consumer market. The most promising market is the bird control market around airports. In order to break into this market, however, a ground station will have to be developed. Although this requires a lot of extra work, RAIDER's autonomy, agility and speed will prove to be invaluable in this market.

# Contents

# Nomenclature

| | |
|---|---|
| $\epsilon$ | Strain[-] |
| $\hat{\imath}$ | Unit vector |
| $\lambda$ | Inflow factor |
| $\Omega$ | Absolute impact angle[rad] |
| $\omega$ | Rotational rate[rad/s] |
| $\phi$ | Inflow angle[rad] |
| $\pi$ | Pi constant |
| $\rho$ | density[kg/m$^3$] |
| $\sigma$ | Solidity factor |
| $\sigma$ | Stress[N/m$^2$] |
| $C_P$ | Power coefficient |
| $C_Q$ | Torque coefficient |
| $C_T$ | Thrust coefficient |
| $C_l$ | Lift coefficient[-] |
| $E(u,v)$ | Change in intensity for the shift [u,v] |
| $I_x$ | Color intensity gradient in the x-direction |
| $I_y$ | Color intensity gradient in the y-direction |
| $K_D$ | Derivative Gain [-] |
| $K_I$ | Integral Gain [-] |
| $K_P$ | Proportional Gain [-] |
| $r_{pitch}$ | Blade pitch |
| A | Area, type noted in subscript[m$^2$] |
| a | Damping constant of a structure[Ns/m] |
| C | Battery capacity[Ah] |
| c | Chord length[m] |
| d | Distance, type noted in subscript[m] |
| E | Youngs modulus[N/m$^2$] |
| F | Force, direction and type noted in subscript[N] |
| G | Shear modulus[N/m$^2$] |
| g | Gravity constant: 9.81m/s$^2$ |
| I | Approximated moment of inertia[kgm$^2$] |
| I | Current[A] |
| k | Spring constant of a structure[N/m] |
| Kv | RPM constant of the motor[$m^-1V^-1$] |
| l | Length, type noted in subscript[m] |
| M | Moment, direction and type noted in subscript[Nm] |
| m | Mass, type noted in subscript[kg] |
| N | Number of propeller blades |
| P | Power needed by the propellers |
| p | Pressure [N/m$^2$] |
| Q | Torque[Nm] |
| R | Blade radius |
| r | Fraction of the blade span |
| r | Moment arm, type noted in subscript[m] |
| T | Thrust produced by propellers |
| t | Thickness, type noted in subscript[m] |
| t | Time[s] |
| U | Energy[J] |
| U | Resultant inflow velocity in forward flight[m/s] |
| V | Voltage[V] |
| v | velocity[m/s] |
| y | Radial position on the blade |

# Acronyms

| | | | |
|---|---|---|---|
| API | Application Programming Interface | MAV | Micro Aerial Vehicle |
| ADR | Autonomous Drone Race | MSE | Mean Squared Error |
| APAC | Asia-Pacific Region | NAV | Navigation Subsystem |
| BEMT | Blade Element Momentum Theory | NLH | New Likelihood of Occurrence after Mitigation |
| BET | Blade Element Theory | NDI | Non-linear Dynamic Inversion |
| CAD | Computer Aided Design | NSV | New Severity of Consequences after Mitigation |
| CAGR | Compounded Annual Growth Rate | OL | Operation and Logistics |
| CNN | Convolutional Neural Network | PCB | Printed Circuit Board |
| CV | Computer Vision | PDB | Power Distribution Board |
| DeCNN | Deconvolutional Neural Network | PID | Proportional Integral-Derivative |
| EKF | Extended Kalman Filter | PP | Power and Propulsion Subsystem |
| EL | Electrical Subsystem | PRU | Programmable Real-time Unit |
| ESC | Electronic Speed Controller | RoI | Return on Investment |
| FEA | Finite Element Analysis | ROS | Robot Operating System |
| FEM | Finite Element Method | RPM | Revolutions |
| FLOPS | Floating point operations | Rx | Receiver |
| FPR | False Positive Rate | SaR | Search and Rescue |
| FPS | Frames per Second | SIFT | Scale Invariant Feature Transform |
| FPV | First-Person View | STR | Structural Subsystem |
| FTE | Full Time Equivalent | SURF | Speeded Up Robust Features |
| IMAV | International Micro Aerial Vehicle | SV | Severity of Consequences |
| IMU | Inertial Measurement Unit | SWOT | Strengths, Weaknesses, Opportunities, and Threats |
| INDI | Incremental Non-linear Dynamic Inversion | SYS | System Integration |
| IROS | International Conference on Intelligent Robots and Systems | ToF | Time of Flight |
| KF | Kalman Filter | TPR | True-Positive Rate |
| LDP | Lightweight Design Philosophy | Tx | Transmitter |
| LiDS | Life cycle Design Strategies | UKF | Unscented Kalman Filter |
| LiPo | Lithium Polymer | VO | Visual Odometry |
| LH | Likelihood of Occurrence | | |

# 1 Introduction

For the last 20 years engineers have developed systems with a higher level of autonomy than ever before. They can be found in numerous situations and applications, like for instance advanced factories and assembly lines. Some help reduce the workload of the employees, while others are so advanced that they completely replace the employees. They monitor the complete manufacturing process autonomously, from start to finish. It is not only possible to find the application of autonomous technology in the manufacturing industry, but also in our daily lives. An example of this is the self driving cars that drive around the streets now. Those cars can drive on highways, change speed and switch lanes. All of this is done while monitoring the environment around it, ensuring that everybody's safety.

Next to autonomous systems, another market that has been booming the last 20 years is the drone market. As microprocessors and computer boards have become widely available, cheap and lightweight, the price of drones has dropped drastically, making them affordable. Quadcopters that are piloted by humans as such are already widely available and being sold by big electronics companies. As quadcopters become more popular, their range of their applications becomes bigger. Currently, quadcopters are being used for recreational flying, areal photography, military applications, search and rescue, drone racing and much more.

Both worlds, the autonomous systems and quadcopter world have been brought together before, resulting in autonomous drones. Although the level of autonomy in such quadcopters is still rather low. They can fly a pre-planned path, automatically hold an altitude or transmit high quality footage to the other side of the world, but they still rely on a human pilot making the decisions. It is a challenge to build a drone which incorporates autonomous systems that exceed human capabilities. This incorporates the long term vision of the project, pursuing the advancement of drone capabilities to a level exceeding that of human piloted drones. The long term vision can be found in the mission need statement below.

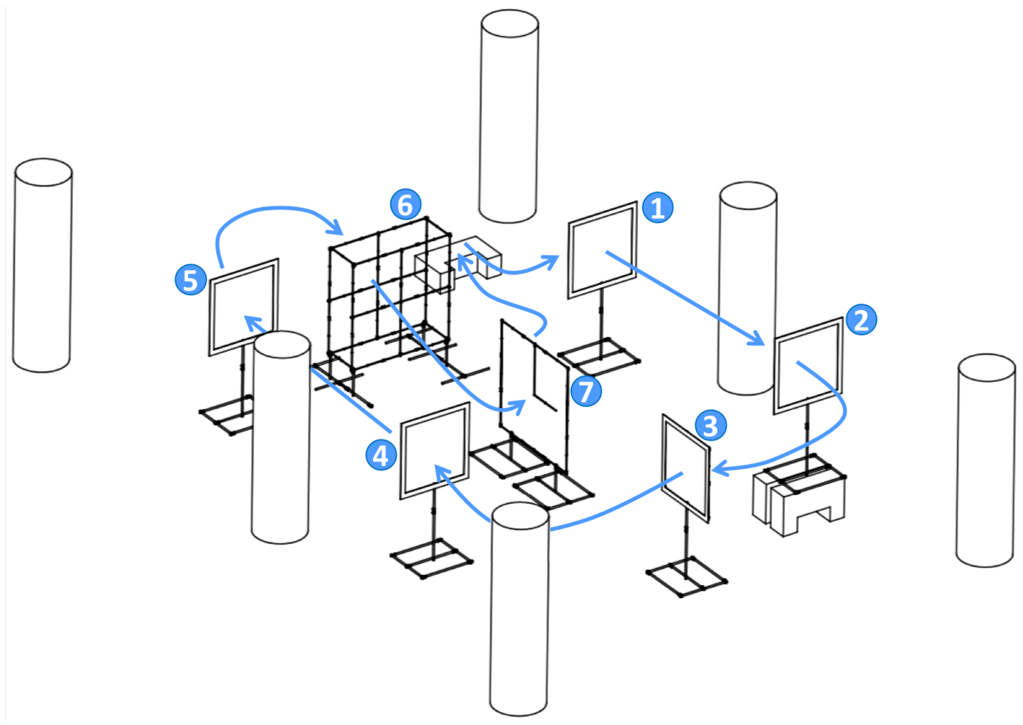
***"Advance the capabilities of autonomous drones to a level exceeding that of human piloted drones"***

As it is practically impossible to directly meet the mission need statement, intermediate steps have to be undertaken. Nowadays, human piloted drone races are held, with drones reaching speeds of over 150 km/h. The top speed reached yet in the IROS 2018 autonomous drone race is 45 km/h, which goes to show that there is a lot of room for improvement in this field. Therefore, as a first step to achieve the mission need statement, competing in the IROS 2018 autonomous drone race is a perfect engineering challenge. More specifically, to not only compete in the IROS 2018 autonomous drone race, but to actually win it. This brings us to the project objective statement, which can be seen below.

***"Win the IROS 2018 competition by designing a fast and agile autonomous drone with 9FTE (full time equivalent) in 10 weeks."***

This report was written to show the process and result of designing a fast and agile autonomous drone with 9FTE (full time equivalent) in 10 weeks. Part 1 includes the process of ending up with a preliminary design, starting from the project mission. This includes a thorough market analysis, performed in Chapter 2. Followed by a mission analysis in Chapter 3 and the preliminary design in Chapter 4. Then in part 2, the detailed design is performed. This consists of Chapters 5 to 11, going over every subsystem design. Finally, part 3 consisting of Chapters 12 to 21, covers the goals, resources and schedule of the project. The reliability, availability, maintainability, and safety of the design have also been addressed, together with a risk analysis and a sustainability excursion. Finally, the post-DSE and long term vision is elaborated upon.

# I
# The Mission



# 2 Market Analysis

Analyzing its market is key to identifying the product's competitive advantage and establishing a strategy to break into attractive market segments. Although our primary objective is that of winning the IROS Autonomous Drone Race, competitions often set the standard for the drone industry and it is worthwhile to seek application areas outside of the IROS competition in which our product may be competitive. Section 2.1 provides an analysis of the most promising market segments identified outside of drone racing, while Section 2.2 focuses on examining and further segmenting the drone racing ecosystem. In Section 2.3 all market insights are compiled into a SWOT matrix, which is used to define a high-level market growth and entry strategy to take advantage of relevant market opportunities while still meeting the project objective.

## 2.1. External Markets

Drones that can autonomously interpret their surroundings and navigate through them are highly sought after in a number of sectors outside of drone racing. Among the most directly related and promising applications are: *search and rescue*, *anti-drone operations* and *bird control*. These are all areas in which lightweight and agile drones have an innate strategic advantage. Other applications such as *delivery* or *surveying and mapping* are deemed uninteresting as our product is designed to be as light as possible for optimal race performance and will therefore not possess the range and payload-carrying capabilities required to be competitive in these sectors.

### 2.1.1. Search and Rescue

The search and rescue (SaR) market is very broad. With drone technology on the rise, interesting opportunities have emerged in this domain that allow for resources to be used more efficiently and that do not require sending people into life-threatening scenarios.

Little market data is available about the use of drones in search and rescue missions specifically. Search and rescue missions are often coordinated by governments and military agencies, so the level of disclosure related to the use of advanced SaR equipment or technology is generally low. As an indication of the market trend, we look at the general search and rescue equipment market, which also includes advanced equipment such as UAVs; the latter is currently valued at $113.6 Billion and is projected to grow at a compounded annual growth rate (CAGR) of 2.03% until 2022[1]. The North American segment of the SaR market is still the largest to date, and the Asia-Pacific (APAC) region is projected to experience the most growth due to fast modernization and increasing demand for high-tech SaR solutions.

Most SaR drones currently in use are focused on combat or other large-scale search and rescue operations, with significant government and military involvement. Key players in this regard are, for instance: General Dynamics Corporation (U.S.), Honeywell International Inc. (U.S.), Leonardo S.p.A. (Italy), Raytheon Company (U.S.) and Textron Systems (U.S.). This kind of application has high barriers to entry and may be difficult to break into for emerging players, given that firms operating in this segment are typically government or defence contractors. Obtaining similar contracts entails working through plenty of red tape which can impose procurement costs that are unsustainable for small businesses.

What has proven successful for small entrepreneurial ventures is to not compete directly with large contractors but rather offer a substantially different and highly-specialized service, or find a niche which falls outside their scope of operations. For instance, Dutch startup Avy managed to break into the search and rescue market by developing drones specifically to detect boats of immigrants stranded in the Mediterranean Sea and deploy inflatable rescue buoys. Given the small size, agility and autonomy of our product, a suitable niche search and rescue application could be the search for survivors in the cramped environments that are the result of an earthquake or a similar natural disaster.

[1] `https://www.marketsandmarkets.com/Market-Reports/search-and-rescue-equipment-market-261129036.html` [Visited at 2 may 2018]

### 2.1.2. Anti-Drone Operations

With drones becoming increasingly popular and accessible to the public, their use for illegal activities has also been on the rise. UAVs have recently held pivotal roles in crimes ranging from drug smuggling to police counter espionage, forcing authorities across the globe to find new ways to counter the use of malicious drones. Among the most innovative solutions to date is that of the Netherlands' national police, which has decided to train bald eagles to snatch rogue drones out of the sky. More standard solutions attempt to use methods such as radio-frequency based systems to communicate and take control of the drones.

The anti-drone market was valued at $ 342.6 Million in 2017 and was projected to grow at a CAGR of 25.9 % between 2017 and 2023[2]. As is the case for most drone-related markets, the largest growth is expected in the APAC market due to the rapid technological development occurring in emerging Asian countries, which are still quite deregulated. Agile and lightweight drones are a natural alternative for intercepting malicious UAVs. In particular, drones with special sensing equipment that can autonomously identify other UAVs are a promising solution to locating malicious drones in cluttered environments. Competitions like the DroneClash [3] give contestants the opportunity to test their drone's ability to intercept and take down other drones, and are beginning to give increasing value to contestants' level of autonomy. While being competitive in this segment will likely involve making significant hardware modifications, the agility and autonomy of our product are attractive characteristics for this application. As discussed in Section 2.3, competitions like the DroneClash can serve as an opportunity to test whether the product is suitable for a similar application and be used as a point of entry into this segment.

### 2.1.3. Bird Control

Birds have been posing serious economic and safety threats in areas such as waste management, agriculture and especially aviation. Most bird control efforts rely on acoustic or laser systems strategically positioned to repel birds. This is a relatively unexplored niche market which has recently seen a number of successful entrepreneurial ventures. The Bird Control Group, for instance, has succeeded in scaling up into a Deloitte Technology Fast50 Rising Star and operating in 76 different countries, by offering tailored laser-based bird control solutions across a number of industries.

The issue with current bird control strategies, including those developed by the Bird Control Group, is that they are static. While their success rate in repelling birds is undeniable, common laser-based solutions involve instruments that need to be mounted and left in the same position. This makes it problematic when large areas need to be covered or when birds do not always appear in the same place; drones are a viable and modular alternative to repel birds in these situations. Bird-chasing drones like those of ClearFlightSolutions (a start-up from the University of Twente)[4] have attracted remarkable attention from airports in Europe and internationally; airports are seeking a more dynamic alternative to the bird-control problem and drones appear to be a very promising and marketable solution. Very little market data is available about the aggregate value brought by these solutions to airports, as the volume of recorded transactions is still very limited. However, several studies that attempt to quantify the value proposition brought by bird control using statistical techniques and case studies have been conducted. In particular, Nitant Shinde developed a Monte-Carlo tool to perform cost-benefit analysis of bird-control solutions at airports for different bird species [43]. He concludes that with state-of-the art technology cost-benefit ratios of over 70 can be achieved, suggesting that high-tech bird control can be an extremely profitable investment for airports and explains their recent interest in such technologies.

The speed, agility, and degree of autonomy that our drone is designed to achieve are very useful properties for the chasing of birds. The technical characteristics of our product give it the potential of being very competitive in this market segment. The biggest challenge will lie in operation and legislation, which can make it difficult to operate drones commercially in European countries and potentially force us to make a lateral move into more deregulated international markets.

## 2.2. Race Drone Market

This section is focused on finding potential opportunities in the race drone market and strategies to be competitive in these markets. To this end, the race drone market is segmented into autonomous and non-autonomous races.

### 2.2.1. Trends and Future Developments

The drone sports ecosystem is expanding rapidly, fueled by disruptive advances in technology and by the growing popularity of drones for recreational purposes. Forecasts by research firm Frost & Sullivan suggests that hobby

[2] `https://www.marketsandmarkets.com/Market-Reports/anti-drone-market-177013645.html` [Visited at 2 May 2018]
[3] `http://www.droneclash.nl/` [Visited at June 24 2018]
[4] `https://clearflightsolutions.com/` [Visited at June 24 2018]

drones are set to grow into a $ 4.4 billion market by the year 2020 [11]; interest in drone technology is booming and, consequently, so is the number of people familiar with drone racing. Drone competition organizers have several revenue generation models, among which four main ones can be distinguished: Ticket sales, sponsoring, participation and merchandising. Growing global interest in fast and intelligent drones has the potential of driving ticket sales, sponsoring and merchandising up significantly. Advances in computing technology, autonomous control and manufacturing have the potential of lowering the barriers to entry for aspiring competitors while also generating upwards pressure on the quality of drones entering competitions; this suggests increases in both the number and quality of participants in competitions, which are key drivers for the 'participation' revenue stream. The drone competition market is well equipped to grow and expected to experience large increases in revenue in the coming years.

### 2.2.2. Autonomous Race Drones vs Piloted Race Drones

At present, the autonomous and piloted segments of the race drone market are still very different, both on a performance and on a market level. Piloted drone races still take up the vast majority of the share in the drone sports market, and overshadow their autonomous counterparts in both revenue and media exposure. While the team's mission is that of advancing *autonomous* drone technology, it is worthwhile to examine whether it is currently feasible to design a product that can be competitive in both segments. Performing well in a piloted drone race with an autonomous vehicle, is, after all, in line with the mission need of bridging the gap between the two segments. Moreover, piloted races offer stronger media exposure and a higher potential profit in the current market. To assess the feasibility of this objective, the fastest commercially available FPV drone, namely the VRX-190, is studied.

Figure 2.1: VRX-190 Race Drone

| | |
|---|---|
| **Speed [km/h]** | 267 |
| **Mass [g]** | 479 |
| **Thrust [g]** | 7400 |
| **Thrust to weight [-]** | 15 |
| **Price [€]** | 615 |

The characteristics of the VRX-190[5] drone exceed by far what has been observed in autonomous drones, as well as the driving customer requirements for this project. NASA JPL recently topped off years of research and development in autonomous vehicle technology with the development of a fully autonomous race drone, that only managed to sustain speeds of approximately 80 km/h in an obstacle course. Being able to sustain speeds comparable to the 267 km/h of the VRX-190 is currently unfeasible for a drone to be operated autonomously, and attempting to do so for the sake of this project would impose killer constraints on the instrumentation, control and navigation subsystems. Moreover, the VRX-190 can be produced for 615 Euro and only weighs 479g; both numbers are far lower than the customer requirements (2500 Euro and 800g), which better reflect the norm in autonomous drone MAVs(Micro Air Vehicles). Attempting to cut weight and cost to compete with drones such as the VRX-190 would inevitably require allocating more scarce resources to structural and aerodynamic design, at the expense of subsystems such as control, navigation and remote sensing, which are crucial to fulfill the project objective.

It was therefore deemed unfeasible to compete with FPV (First Person View) race drones such as the VRX-190 without severely compromising the mission of advancing autonomous race drone technology and the project objective of winning the IROS 2018 competition. Rather, opportunities will be sought in the autonomous race drone market.

### 2.2.3. Opportunities in Autonomous Drone Racing

In the short term, potential opportunities are found in the IROS 2018 Competition and the IMAV(International Micro Air Vehicle) 2018 Competition; the main challenges to be overcome in order to be competitive in these competitions are discussed.

#### IROS Competition Analysis

The IROS ADR 2018 competition will take place in October in Madrid, and winning this competition is the main design objective. Little detail is available about the physical characteristics of previous competitor drones, and standardizing their performance is difficult as the tracks become more demanding every year. Rather, this section will focus on providing a high-level overview of competitors and the main issues they encountered in previous editions.

As described in [37], the main challenge encountered by teams in the competition was gate recognition due to cluttering and the monotonic color. All teams used visual recognition to identify gates, which caused issues in the sharp-turn path region of the circuit; many teams had a small camera-view angle which lead to recognition failure after sharp turns [37]. Rumors state that the IROS competition will move towards less 'obvious' dark grey gates in the

[5] `https://fpvdronereviews.com/guides/fastest-racing-drones/` [Visited at 2 May 2018]

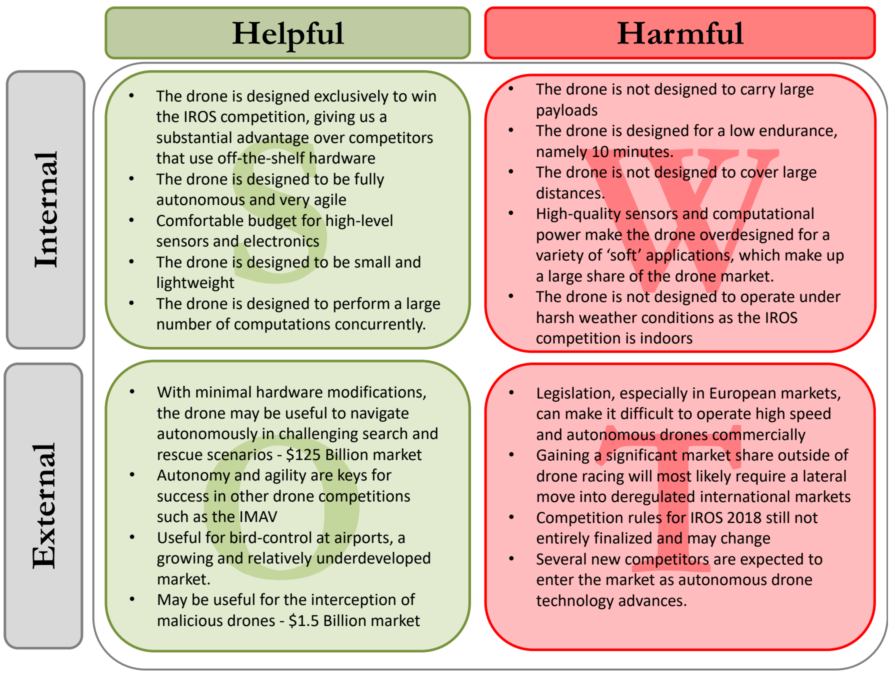
future to better mimic an indoor disaster-struck area. To be competitive in this race, the main challenge is to come up with an effective software and navigation strategy, and in designing hardware that can handle the computational intensiveness and the required measurement accuracy required by the strategy.

**IMAV Competition**

The IMAV 2018 competition will be held this November in Melbourne, Australia. This competition serves as a technology demonstration for state-of-the-art performance of small, lightweight and autonomous drone with emphasis on multi-MAV cooperation. While winning this competition is not a main project objective, it is seen as an effective way to pursue the team's overarching mission of advancing autonomous drone technology. Performing well in the IMAV can lead to strong media exposure, attract investors, and open doors to new projects and ideas for further development. For instance, the competition in 2016 was focused on performance in a search and rescue scenario, which is a key external market opportunity for our product. Moreover, it is feasible to enter the IMAV without making significant changes to the drone hardware as the characteristics that will make the drone perform well in the IROS (autonomy, image processing and design efficiency) are listed as the focus points of the IMAV competition. The MAV-Lab[6], for instance, entered the IMAV competition in 2016 with a modified Bebop drone, which was also used in the 2016 IROS competition. Without adjustments to the hardware other than very simple and modular ones (e.g: plastic chopsticks mounted and used as 'legs' to grip objects), the MAV-Lab managed to rank 2nd in the IROS and gain impressive media exposure at the IMAV [7].

## 2.3. Market Entry and Growth Strategy

In this section, the market opportunities that were identified are summarized in a SWOT (Strengths Weaknesses Opportunities Threats) matrix (Figure 2.2), and a high-level market entry and growth strategy to take advantage of such opportunities while still meeting the project objective is suggested.



| | Helpful | Harmful |
|---|---|---|
| Internal | S<br>• The drone is designed exclusively to win the IROS competition, giving us a substantial advantage over competitors that use off-the-shelf hardware<br>• The drone is designed to be fully autonomous and very agile<br>• Comfortable budget for high-level sensors and electronics<br>• The drone is designed to be small and lightweight<br>• The drone is designed to perform a large number of computations concurrently. | W<br>• The drone is not designed to carry large payloads<br>• The drone is designed for a low endurance, namely 10 minutes.<br>• The drone is not designed to cover large distances.<br>• High-quality sensors and computational power make the drone overdesigned for a variety of 'soft' applications, which make up a large share of the drone market.<br>• The drone is not designed to operate under harsh weather conditions as the IROS competition is indoors |
| External | O<br>• With minimal hardware modifications, the drone may be useful to navigate autonomously in challenging search and rescue scenarios - $125 Billion market<br>• Autonomy and agility are keys for success in other drone competitions such as the IMAV<br>• Useful for bird-control at airports, a growing and relatively underdeveloped market.<br>• May be useful for the interception of malicious drones - $1.5 Billion market | T<br>• Legislation, especially in European markets, can make it difficult to operate high speed and autonomous drones commercially<br>• Gaining a significant market share outside of drone racing will most likely require a lateral move into deregulated international markets<br>• Competition rules for IROS 2018 still not entirely finalized and may change<br>• Several new competitors are expected to enter the market as autonomous drone technology advances. |

Figure 2.2: Technical SWOT matrix

### 2.3.1. Product Development

The drone will be developed with the sole intention of winning the IROS ADR 2018 competition while meeting all user requirements related to budget, operations and sustainability. In the product development phase, all objectives other than winning the IROS competition will be deemed secondary, as will the requirements associated with this. This short-term 'tunnel vision' will give the team a strategic edge over the competition, which consists of commercially available drones with hardware that is not explicitly designed around the navigation, remote sensing and control requirements imposed by the race.

[6]The TU Delft Micro Air Vehicle lab

[7]`http://mavlab.tudelft.nl/imav2016/` [Visited at 2 May 2018]

### 2.3.2. Product Diversification

Upon fulfillment of the project objective (winning the IROS competition), the drone will be adapted to compete in similar competitions. Priority will be given to competitions which require little hardware adjustments, such as the IMAV competition. The aim is to progressively enter other competitions that may require more substantial hardware changes and development costs, such as the DroneClash, and diversify the scope of the product without sacrificing autonomy. This will allow the team to begin exploring possible external application areas while staying in a competitive setting. In the drone market, competitions often set the standard for industry and good performance may open the door to funding and development opportunities.

### 2.3.3. Market Development & Penetration

In this phase, the product will be brought into application areas outside of drone racing to enhance its market potential. The most interesting application areas are considered to be the Anti-bird, Search & Rescue and Anti-drone markets, which are all analyzed in detail in Section 2.1. Breaking into these larger markets is key to fulfill the mission need of advancing the global performance of autonomous drones, as it will allow to develop other aspects of the technology and attract more widespread attention. The key challenges in this phase, especially in the European market, will be related to operation and legislation. Collaborating with airports such as Schiphol and, more generally, operating drones for commercial use in the Netherlands, is extremely difficult and poses serious limitations. These limitations forced innovative companies such as ClearFlightSolutions, an emerging competitor in the anti-bird market, to target other regions like Germany and Canada. Success in this stage will most likely demand expansion into more deregulated international markets.

# 3
# Mission Analysis

In order to make sure the design completes its mission, while complying with the requirements set by the customer, the mission has to be analyzed in detail. This will be done in this chapter. Starting with a clear description of the mission and project objective in Section 3.1, followed by an overview of the race regulations in Section 3.2. Next, the stakeholder requirements will be discussed in Section 3.3, followed by the functional flow block diagram in Section 3.4 and the functional breakdown structure in Section 3.5.

## 3.1. Project Mission and Objective

In order to have a clear-cut definition of what to achieve in this project, proper mission and project objective statement formulation is of paramount importance. As has been explained in Chapter 2, autonomous drones can be very useful in many situations, such as search and rescue, and bird control. The industry, however, needs to change its mindset on the application of autonomous vehicles before they can be widely adopted. They need to create a mindset in which autonomy is more centrally placed. Also, referring back to this project, drone autonomy is still on a low level and has not reached, or exceeded human capabilities. There are drones that are capable of flying a pre-planned path, that can follow a certain trajectory or that recognize features, but none of them can do so at the speed a human piloted drone pilot. Therefore, following this philosophy, the mission need statement of this project is formulated as follows:

***"Advance the capabilities of autonomous drones to a level exceeding that of human piloted drones"***

Now that a proper mission need statement has been formulated, the project objective statement can be defined as well. Since it is impossible to achieve the mission at once, intermediate steps have to be undertaken. Steps that trigger innovation in drone autonomy, or in other words, steps that bring the capabilities of autonomous drones closer to that of human piloted ones. A good way to show off this advance in autonomous drone technology, is by means of drone racing. In human piloted drone races, drones race through gates at speeds exceeding 150km/h. It is an engineering challenge to design an autonomous drone able to exceed this performance. At the moment, the maximum speed achieved in an autonomous drone race, such as the IROS race, is 43km/h. Therefore, it was decided to participate and win the IROS 2018 autonomous drone race. This is reflected in the project objective statement of this project, that can be seen below.

***"Win the IROS 2018 competition by designing a fast and agile autonomous drone with 9fte (full time equivalent) in 10 weeks."***

## 3.2. IROS 2018 Regulations

The IROS 2018 autonomous drone race is a drone race in which, just like in human piloted drone races, the drones have to fly through a set of gates on a race track. In the IROS 2018 race, the drones get 20 minutes to complete the track as quickly as possible or to get as far as possible in the track.
The drone that completes the track in the least amount of time, wins the race. If no drone completed the track, the drone that got furthest in the track wins.
As in any race, the IROS 2018 race is subject to regulations. They are published on the IROS 2018 website and they will be listed below[1]. Not complying with the regulations can lead to disqualification.

- Each team will get 20 minutes for as many attempts they need.
- Each attempt must start at the starting position.
- The team's official score will be selected from the best attempt.
- Each attempt will be recorded in time.

[1] `http://rise.skku.edu/iros2018racing/index.php/rules-regulations/` [Visited at June 22 2018]

- The group time does not include the time that is needed for turning on ground computers or ground support equipment.
- Power of the drone can be applied before each attempt. (booting time)
- The attempt time is measured from the take off until the finishing of the attempt.
- If twenty minutes passes without finishing the course, the flight time and the last gate the drone passed is recorded.
- Each attempt is considered finished by successful passing the final gate in the correct order.
- The attempt is considered not finished if the drone deviates from the course, does not pass each gate in order, malfunctions, is not able to sustain safe and controlled flight, or if the team operator declares ending the attempt.
- If the drone does not finish track, the ID number of gate and ending time is recorded.
- The winning team shall be the team with shortest flight time if the team finishes the whole course.
- For grading teams who did not finish the whole course will be compared with the ID number of the gate it reached. If two or more teams reached the same gate, the shorter the flight time the higher the ranking.
- The judging committee reserves the right to stop any team's attempt if considered dangerous or not following the guidelines.
- The judging committee reserves the right to rule out any attempt's record if any unfair activity is found.

The IROS 2018 ADR is organized annually. Each year, the layout of the track or gates is changed slightly. Changes such as displacing the gates or changing their color, also changing their shape and functionality is a possibility. This to stimulate innovation every year.
At the start of this design synthesis exercise, the 2018 race track was not yet published. Therefore, the IROS 2017 race track was taken as a reference, while taking into account possible track changes. Although the track incorporates some significant changes this year, the team is confident the drone will perform according to standards.
This year, the track consists of 7 gates. The first 5 gates are normal gates for which the position is known. Gate 6 will be a jungle gym gate, consisting of a double 2x2 matrix, of which only one entrance and exit will be open. Gate 7 will be a dynamic gate, having a moving element. The gate color will be orange, exactly like it was in 2017. An overview of the complete track can be seen in Figure 3.1, and a detail of the jungle gate and the dynamic gate can be seen in fig. 3.2.

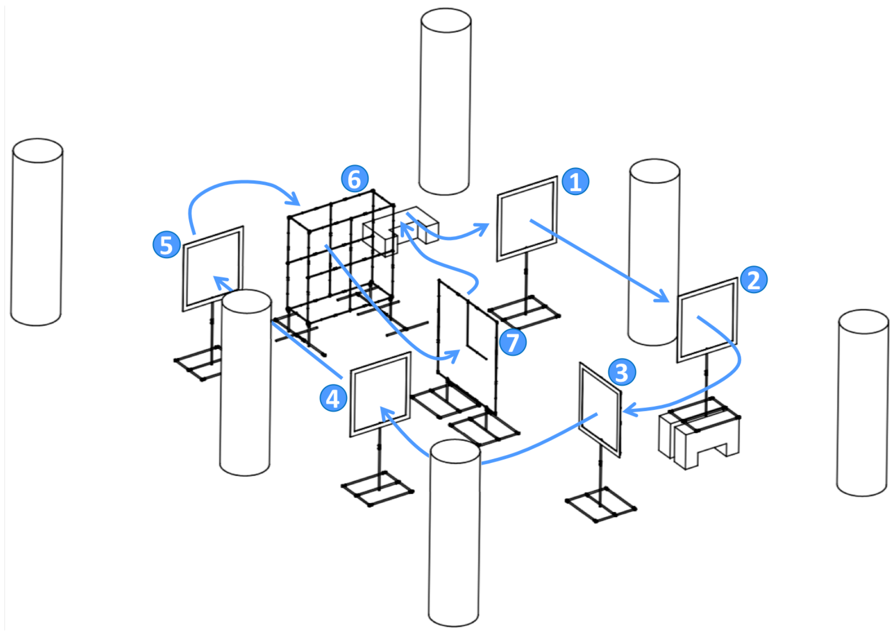


Figure 3.1: Overview of racing track with gates indicated by number

## 3.3. Stakeholder Requirements

When receiving the design task, several requirements have been set by the customer. These requirements have to be met in order to fulfill the mission objective. In this case winning the IROS 2018 autonomous drone race competition. In this section those requirements will be revisited and elaborated upon.
The requirements set by the customer can be divided in several categories. Those categories include performance,

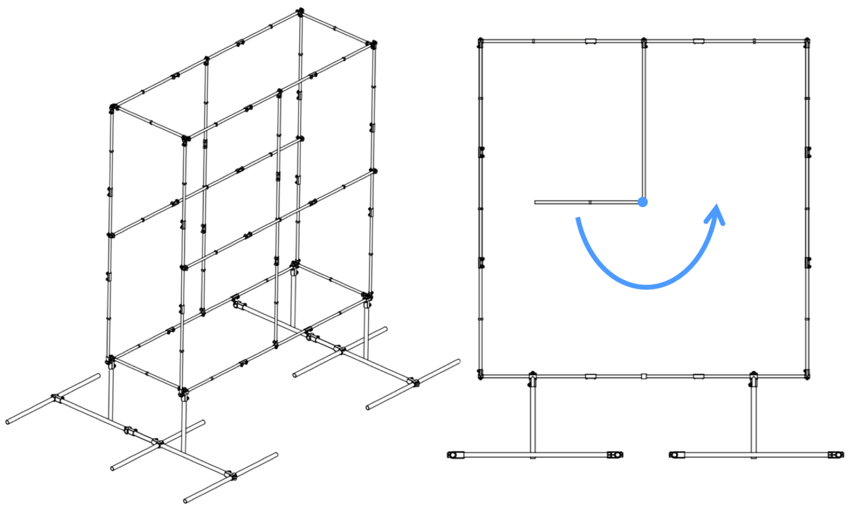


Figure 3.2: Overview of the jungle gate [left] and the dynamic gate [right]

safety & repairability, sustainability, engineering budgets, cost and a miscellaneous group. The category the requirement belongs to is indicated in the requirement identifier with its respective abbreviation.Although requirements have to be met in order to win the IROS 2018 autonomous drone race competition, not all requirements are of the same importance. Some requirements might be critical to win the race, while others have only a small influence on the drone's capabilities when not met.

Therefore, the requirements are divided in 3 tiers. Tier 1 containing the most demanding stakeholder requirements, to which the most resources are allocated. They are requirements that have to be met in order to win the race. Next, tier 2 requirements are requirements that can be beneficial to fulfill the mission, but are not vital to win the race. Finally, tier 3 requirements are requirements imposed by the customer, but which can be compromised, in order to have better performance. The requirements are sorted by tier number in Tables 3.1 to 3.3.

Some requirements that were initially set by the customer have been revised in consultation with the customer. This because they were poorly defined or technically unfeasible. In Tables 3.1 to 3.3 the revised requirements are labeled with an 'r' behind the requirement identifier. Finally, the driving and key requirements are indicated with a key or a steer symbol in the requirement table.

Table 3.1: List of Tier 1 Customer Requirements

| | | **Tier 1** |
|---|---|---|
| **SH-P-2r** | 🔑☸ | The drone shall be able to autonomously fly a full lap of the IROS 2018 track, without crashing and in at most 10 minutes. |
| **SH-SR-8r** | 🔑 | Only parts that cost less than 10% of the unit cost and can be replaced in under 1 minute shall require replacement upon impact with concrete when free-falling from a height of 3m with zero initial velocity. |
| **SH-SR-9r** | 🔑☸ | Only parts that cost less than 10% of the unit cost and can be replaced in under 1 minute shall require replacement upon crashing into a gate at 100 km/h. |
| **SH-EB-13** | 🔑 | The drone shall comply with all requirements given by the 2018 autonomous drone race website if available. |
| **SH-O-16** | 🔑☸ | The drone shall use onboard computation only. |
| **SH-O-18** | 🔑 | The drone shall be able to detect IROS 2018 autonomous drone race gates. |

Table 3.2: List of Tier 2 Customer Requirements

| Tier 2 | | |
|---|---|---|
| **SH-P-1** | 🔑☸ | The drone shall be able to automatically take-off and land. |
| **SH-P-6** | 🔑 | The drone shall be able to fly for at least 10 minutes. |
| **SH-P-7** | 🔑☸ | The onboard computer vision shall be able to track gates at 30 frames per second. |
| **SH-S-11** | 🔑☸ | The motors, frame, electronics and cameras shall be replaceable. |
| **SH-EB-12r** | 🔑 | No dimension of the drone shall exceed 60 cm. |
| **SH-O-17r** | 🔑 | The drone shall use, but is not restricted to, at least one camera for indoor navigation. |
| **SH-O-19r** | 🔑 | The aggregate time required to replace all replaceable components of the drone shall not exceed 7 min. |

Table 3.3: List of Tier 3 Customer Requirements

| Tier 3 | | |
|---|---|---|
| **SH-P-3r** | 🔑 | The drone shall possess the hardware characteristics to fly at a maximum speed of 100 km/h in horizontal straight flight. |
| **SH-P-4** | | The drone shall be able to linearly accelerate 3g. |
| **SH-P-5** | 🔑 | The drone shall be able to make 3m radius arcs at 12 m/s. |
| **SH-S-10r** | 🔑 | The drone shall be made for 100% of recyclable materials. |
| **SH-EB-14** | 🔑 | The total mass shall be less than 800 gram. |
| **SH-C-15r** | 🔑 | The aggregate hardware cost of the drone shall not exceed 2500 Euro, excluding maintenance, repair and operational costs. |

## 3.4. Functional Flow Block Diagram

The functional flow diagram illustrates the operations of the quadcopter trough out its operational lifetime. It is constructed in a sequential manner, starting from the manufacturing of the drone until it has finished its mission. In order to maintain clarity and readability, the functional flow diagram is constructed in several levels. The top level shows the top functions of the quadcopter in sequential manner. The top level functions are elaborated upon in the second level, with the sub-functions grouped by their respective top level function in a dashed line box. The same accounts for the third level that elaborates on second level functions and the fourth level that elaborates on third level functions. Also, when multiple output possibilities are present, a logic gate is used. Their output is true or false, depicted by the complete functional flow diagram set can be seen inFigures 3.3 and 3.4.

## 3.5. Functional Breakdown Structure

Another way of representing the quadcopter functions is by means of a functional breakdown structure. This diagram illustrates the same functions as the functional flow diagram, but instead of showing them in chronological order, it orders them by category. For consistency, the labeling of both the functional breakdown diagram and the functional flow diagram functions is the same. As a labeling scheme, abbreviations are preferred above numbers. The using abbreviations in labels instead of numbers allows for painless adaptation or restructuring of the functional breakdown structure or functional flow diagram. The functional breakdown structure can be seen in Figures 3.5 and 3.6.

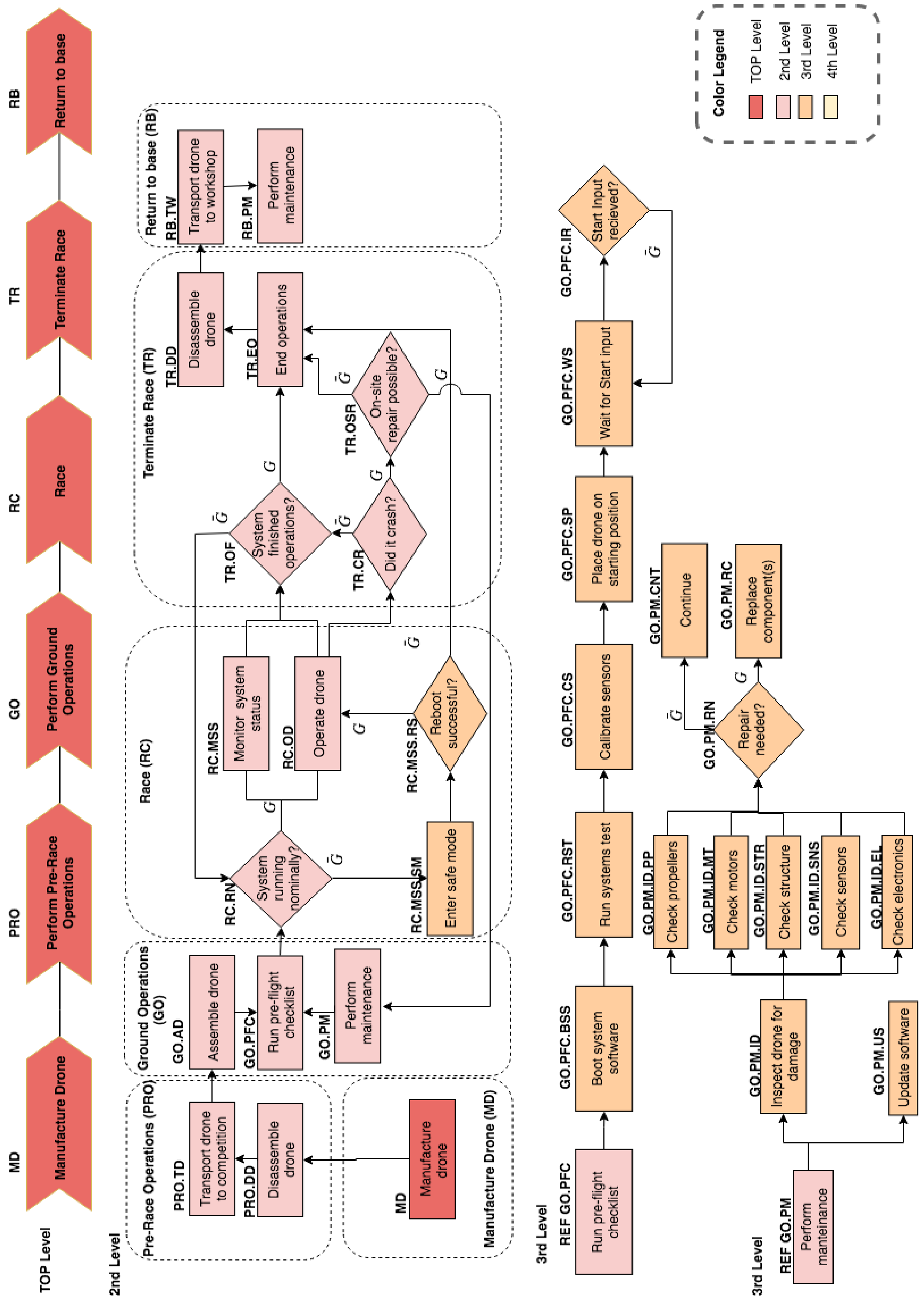

TOP Level
MD Manufacture Drone
PRO Perform Pre-Race Operations
GO Perform Ground Operations
RC Race
TR Terminate Race
RB Return to base
2nd Level
Pre-Race Operations (PRO)
PRO.TD Transport drone to competition
PRO.DD Disassemble drone
Manufacture Drone (MD)
MD Manufacture drone
Ground Operations (GO)
GO.AD Assemble drone
GO.PFC Run pre-flight checklist
GO.PM Perform maintenance
Race (RC)
RC.RN System running nominally?
RC.MSS System running nominally?
RC.MSS.SM Enter safe mode
RC.MSS.RS Reboot successful?
RC.MSS Monitor system status
RC.OD Operate drone
Terminate Race (TR)
TR.OF System finished operations?
TR.CR Did it crash?
TR.OSR On-site repair possible?
TR.DD Disassemble drone
TR.EO End operations
Return to base (RB)
RB.TW Transport drone to workshop
RB.PM Perform maintenance
G
$\bar{G}$
3rd Level
REF GO.PFC Run pre-flight checklist
GO.PFC.BSS Boot system software
GO.PFC.RST Run systems test
GO.PFC.CS Calibrate sensors
GO.PFC.SP Place drone on starting position
GO.PFC.WS Wait for Start input
GO.PFC.IR Start Input recieved?
3rd Level
REF GO.PM Perform manteinance
GO.PM.ID Inspect drone for damage
GO.PM.US Update software
GO.PM.ID.PP Check propellers
GO.PM.ID.MT Check motors
GO.PM.ID.STR Check structure
GO.PM.ID.SNS Check sensors
GO.PM.ID.EL Check electronics
GO.PM.RN Repair needed?
GO.PM.CNT Continue
GO.PM.RC Replace component(s)
Color Legend
TOP Level
2nd Level
3rd Level
4th Level

Figure 3.3: Functional Flow diagram Part 1

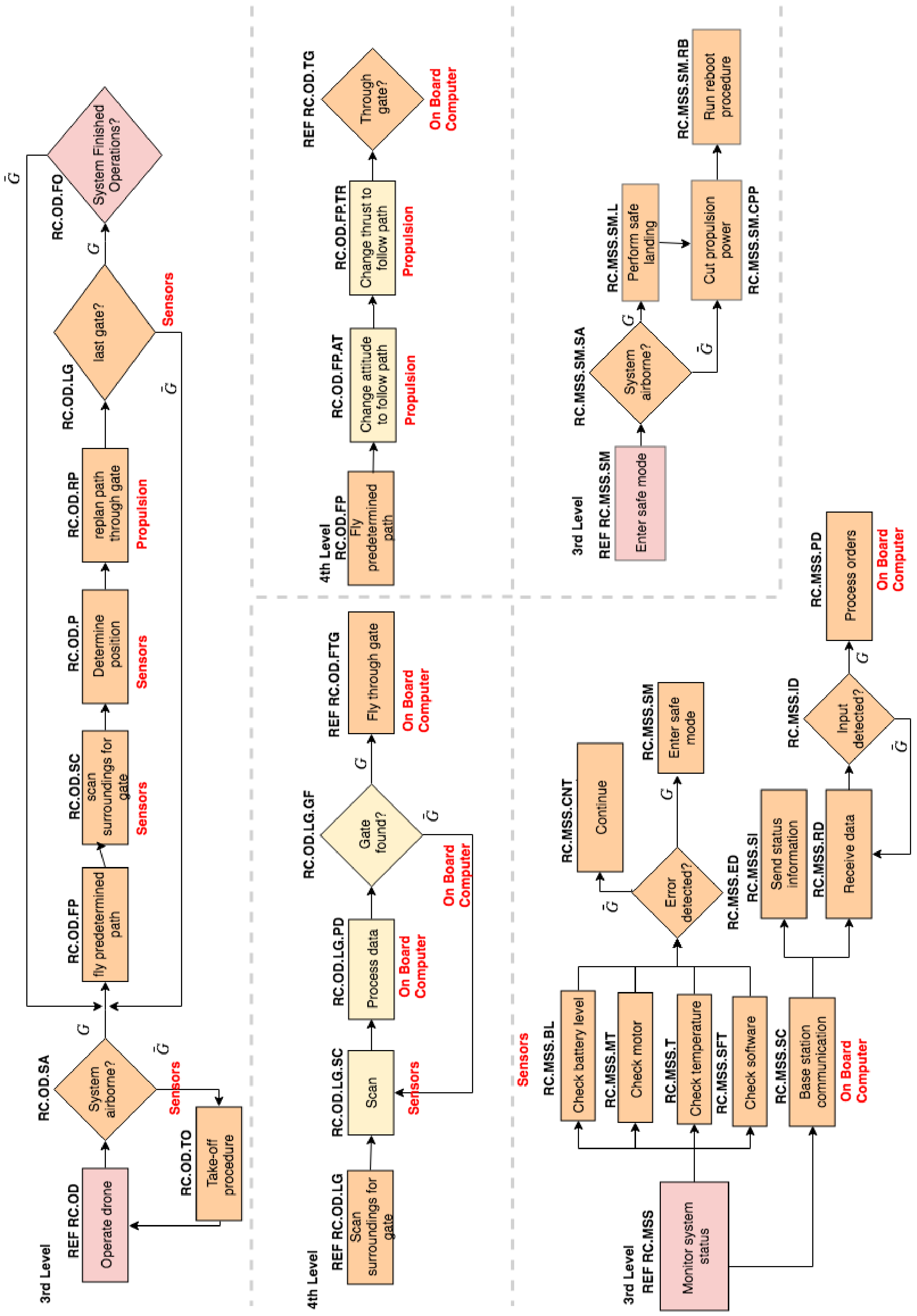


Figure 3.4: Functional Flow diagram Part 2

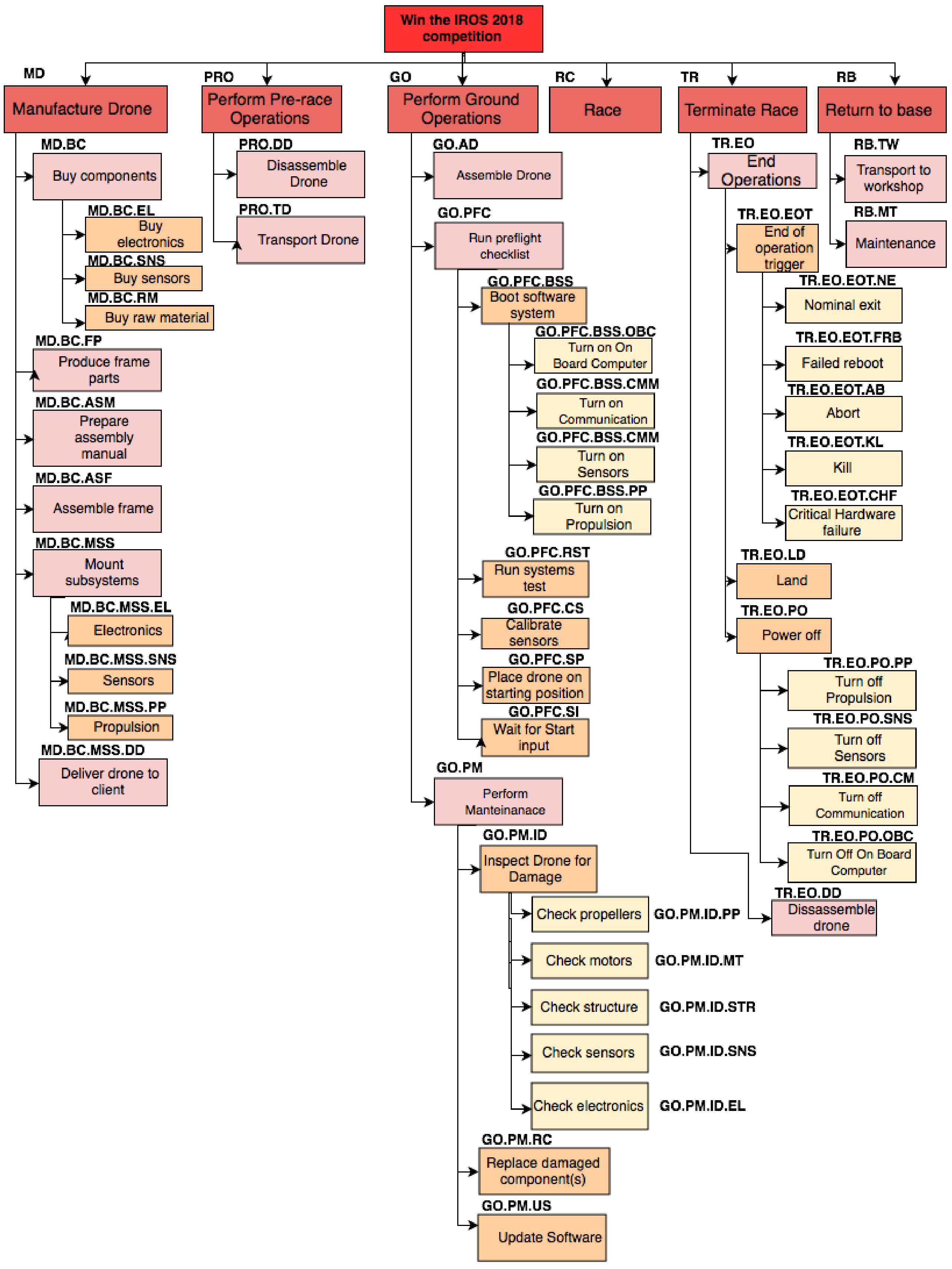


Figure 3.5: Functional Breakdown Structure Part 1

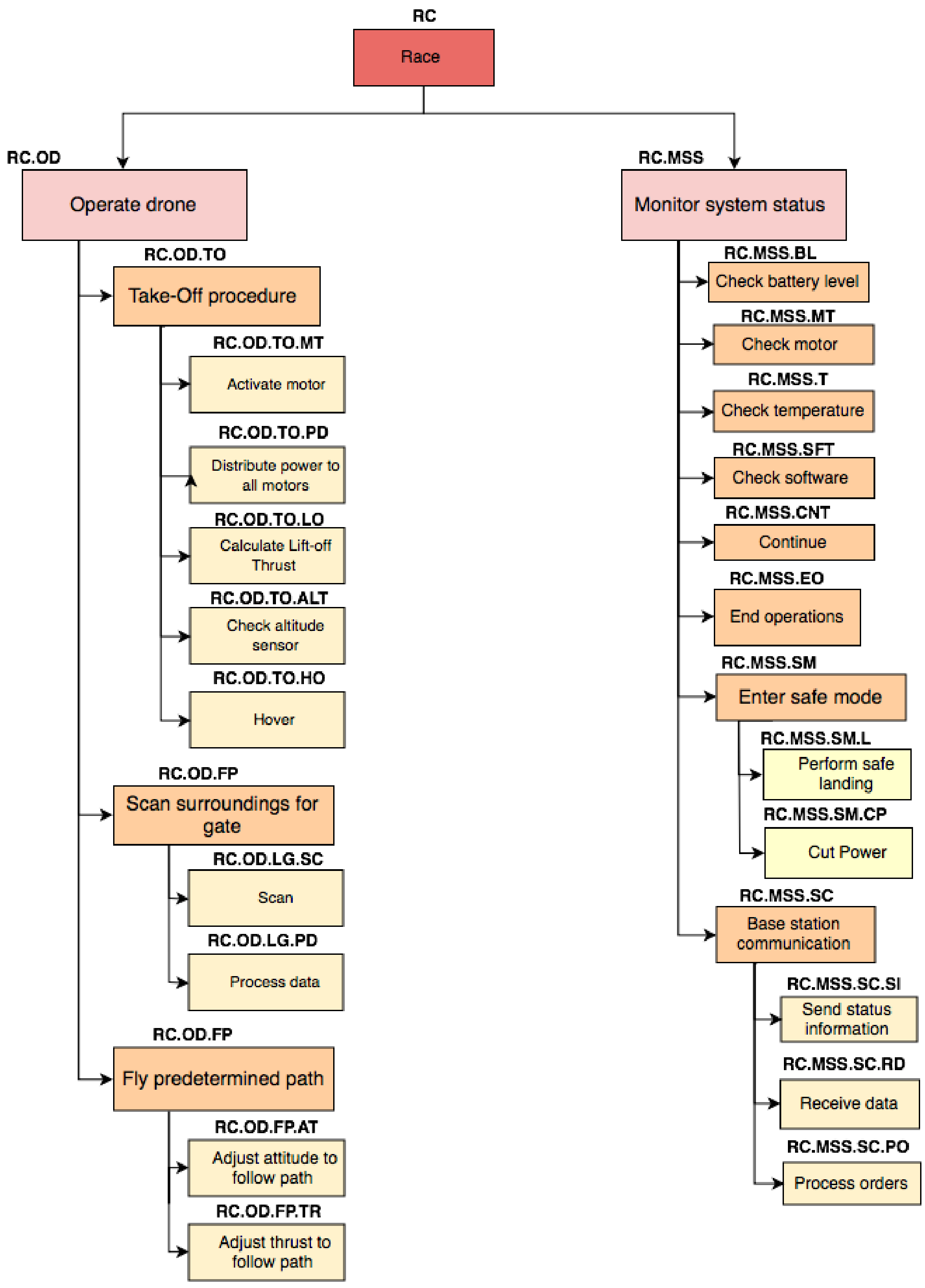


Figure 3.6: Functional Breakdown Structure Part 2

# 4
# Preliminary Design

The purpose of this chapter is to give an introduction of the design of RAIDER developed up to the midterm before the detailed design phase started. In this way, what the design philosophy that was employed was and why certain choices were made.
First the design philosophy is presented in Section 4.1. Afterwards the communication flow diagram identified for the design are reported in Section 4.2. The hardware selected for the drone is detailed in Section 4.3, and the resource used until the preliminary design in Section 4.3.3.

## 4.1. Design Philosophy

The idea behind the philosophy is to build a sturdy agile drone with the smallest size possible. The main consequence of this approach is that the drone has limited payload carrying capabilities, which negatively effect the amount of computational resources on board. This strategy also has many advantages, however. The biggest of these is the small size which allows for larger margins of error when passing through gates. The costs are drastically reduced as well, the material used is lower, the onboard computers and sensors are cheaper, meaning that it is possible to allow for many replacement parts. Furthermore considering the limited development time available, it is beneficial to have a lighter software strategy, which allows for more time spent testing it.

### 4.1.1. Structural Configuration

The structural configuration selected for the design is the quadcopter. Quadcopters are the most widely used multicopter, they have been extensively tested in both piloted and autonomous drone racing. The benefits of quadcopters are the possibility of hovering and the high reliability. As the name implies a quadcopter has four upwards-facing rotors providing the lift. Controlling the thrust setting of each motor individually allows the drone to be flown. The yaw motion is decoupled from the other motions, which means that it can be adjusted independently. For our application, quadcopters were found to fall in a sweet spot because they provide better payload capabilities of tricopter, and less structural weight then drones with more than 4 motors.

### 4.1.2. Algorithms and Payload Interaction

The drone needs to be able to win the IROS 2018 competition by passing through all gates in the shortest amount of time possible. As a result, a system of sensors and algorithms to outperform the competitors was designed. The software package includes a path planning algorithm, a gate and obstacle detection algorithm, and a positioning algorithm.

#### Path Planning

The main idea is to have an algorithm that is computationally light and able to run online. A prepossessed path is already loaded offline, and the drone will only have to update it during the course with the inputs from the gate and obstacle detection and positioning algorithms. This algorithm is to run on the Pocket Beagle [1] a very tiny 1GHz computational unit.

#### Gate and Obstacle Detection

This algorithm is able to detect the gate, and assess the relative position of the drone to the gate. It employs two cameras: a small RGB camera and a Time of Flight (ToF) depth camera. The former is used for most of the time because of the higher range of these cameras. The latter is used from approximately 4 meters from the gate until the gate is passed because higher frame rates are possible with this at the expense of range. The ToF camera is also used for detecting obstacles, this is a separate routine that runs continuously to assess the presence of unknown objects. If any is detected, it estimates its relative position to it and the obstacle's dimensions. The gate detection algorithm works on a different processor, which incorporates dedicated connectors for cameras, the Raspberry Pi Zero W[2]. The two processor are connected and the relative position estimate is passed to the Pocket Beagle. In case an obstacle is detected the spatial dimensions of the obstacle together with the relative position are sent to the Pocket Beagle.

[1] `http://beagleboard.org/pocket` [Visited at ,]
[2] `https://www.raspberrypi.org/products/raspberry-pi-zero-w/` [Visited at 25 June 2018]

**Positioning Algorithm**
The drone needs to be able to track is position, velocity and acceleration at any point in time. In order to do this, an IMU is used to estimate these at an high frequency. In order to limit the integration error caused by these sensors, sensor fusion and the absolute position update of the drone are used. The absolute position is obtained from the relative position of the drone of the gate detection method, given that the exact position of all gates is known beforehand, and the altimeter. This algorithm runs on the Pocket Beagle. The position estimates from the sensor fusion are constantly passed to the path planning and control algorithms.

## 4.2. Subsystem Interface Definition

In this section the interfaces between different subsystems are explained and shown using the communication flow diagram, the hardware block diagram and the software block diagram.

### 4.2.1. Communication Flow Diagram

The communication flow diagram is depicted in Figure 4.1. It provides an overview of the different interactions between software and hardware. In particular, it illustrates the flow of data through the system, and in and out of the environment. Each block represents an important element for the understanding of how the information flows. The arrows between the different boxes represent the flow of information between blocks with a brief explanation in Table 4.1. In the next subsection Section 4.2.2, a more detailed block diagram for the hardware and the software are shown.

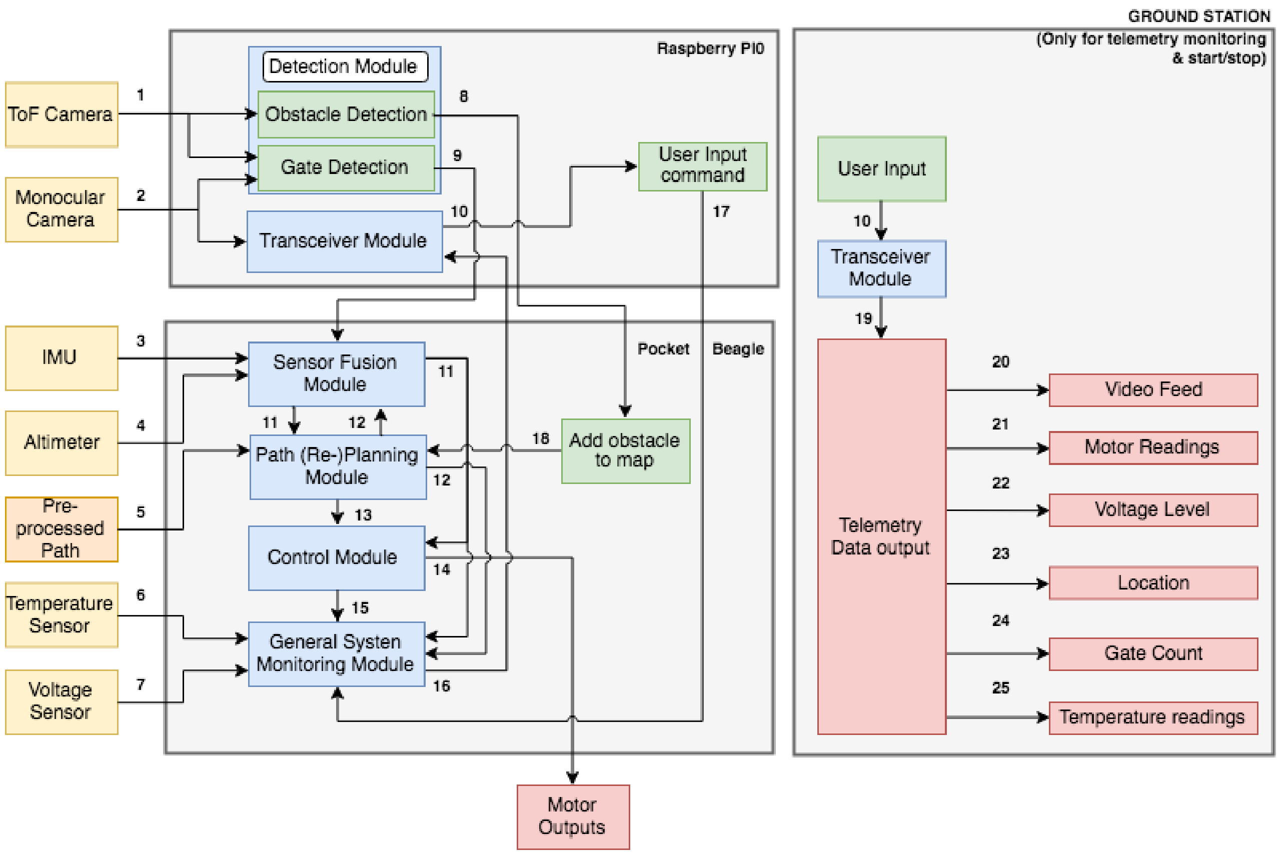


Figure 4.1: Communication Flow Diagram

### 4.2.2. Hardware & Software Block Diagram

The hardware and software flow diagrams are used to give an overview of the interactions and mutual relations between the different parts on both the hardware and software side of the design.
The hardware block diagram is shown in Figure 4.2. Each block represents a component. A link indicates a connection between 2 components. A double arrow represents a mutual relation, while a single arrow means an interaction from one component to the other. An explanation for each link is given in Table 4.2. The interactions between the components will be further refined in the detailed design phase, in particular in Chapter 5, Chapter 15, and Chap-

Table 4.1: Explanation of Links in Figure 4.1.

| Link | Explanation | Link | Explanation |
|---|---|---|---|
| 1 | ToF camera feed | 14 | Required thrust from motors |
| 2 | Monocular camera video feed | 15 | Required motor performance |
| 3 | IMU measurements | 16 | Complete set of sensor and status data |
| 4 | Altimeter measurements | 17 | Shut-down command from user |
| 5 | Pre-processed trajectory data | 18 | Updated way points for path update |
| 6 | Temperature sensor measurements | 19 | Complete set of sensor and status data and video feed |
| 7 | Battery voltage measurements | 20 | Monocular camera video feed |
| 8 | Obstacle dimensions and relative position of drone to obstacle | 21 | Required motor performance readings |
| 9 | Relative position of drone to gate and centre of gate | 22 | Battery voltage measurements for telemetry |
| 10 | Received command from user | 23 | Location of drone estimated by sensor fusion |
| 11 | Absolute position, velocity, acceleration and heading | 24 | Number of passed gates |
| 12 | Number of passed gates | 25 | Readings of the temperature sensor |
| 13 | Required trajectory data (yaw, position, velocity, acceleration) | | |

ter 11, and Chapter 6, where also the electrical block diagram is located.
The software block diagram is shown in Figure 4.3. In this diagram the triangular yellow box represent the inputs and the triangular red box the output and the blue square box the main piece of software in RAIDER. The arrows between the different boxes represent the flow of information between blocks with a brief explanation. In the detailed phase Chapter 7, Chapter 8, Chapter 9 have their own software diagram explaining in detail how the software works. Section 11.1 focuses on how the data is managed in the system and present the data handling block diagram.

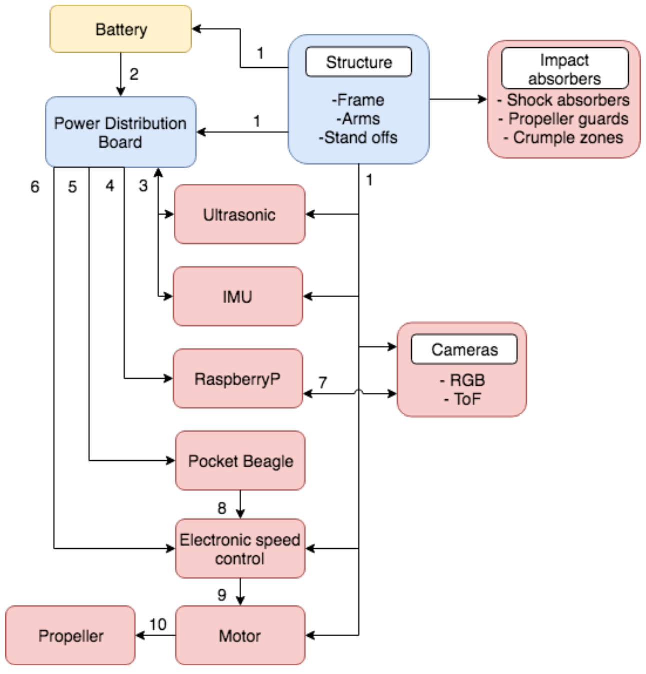


Figure 4.2: Hardware Diagram

Table 4.2: Explanation Links of in Figure 4.2

| Link | Explanation |
|---|---|
| 1 | Provide structural support |
| 2 | Provide power |
| 3 | Distribute power to sensors and connect sensor to Pocket Beagle |
| 4 | Distribute power and structural support |
| 5 | Distribute power, structural support and send data from sensors |
| 6 | Distribute power |
| 7 | Raspberry Pi distributes power to cameras and cameras provide data to board |
| 8 | Provide control input to ESC |
| 9 | Required thrust per motors |
| 10 | Provide attachment for propellers |

## 4.3. Hardware Preliminary Design

This section provides the hardware preliminary design, in particular it focuses on the choice of sensors. Finally, initial estimates of the mass and power budgets are given.

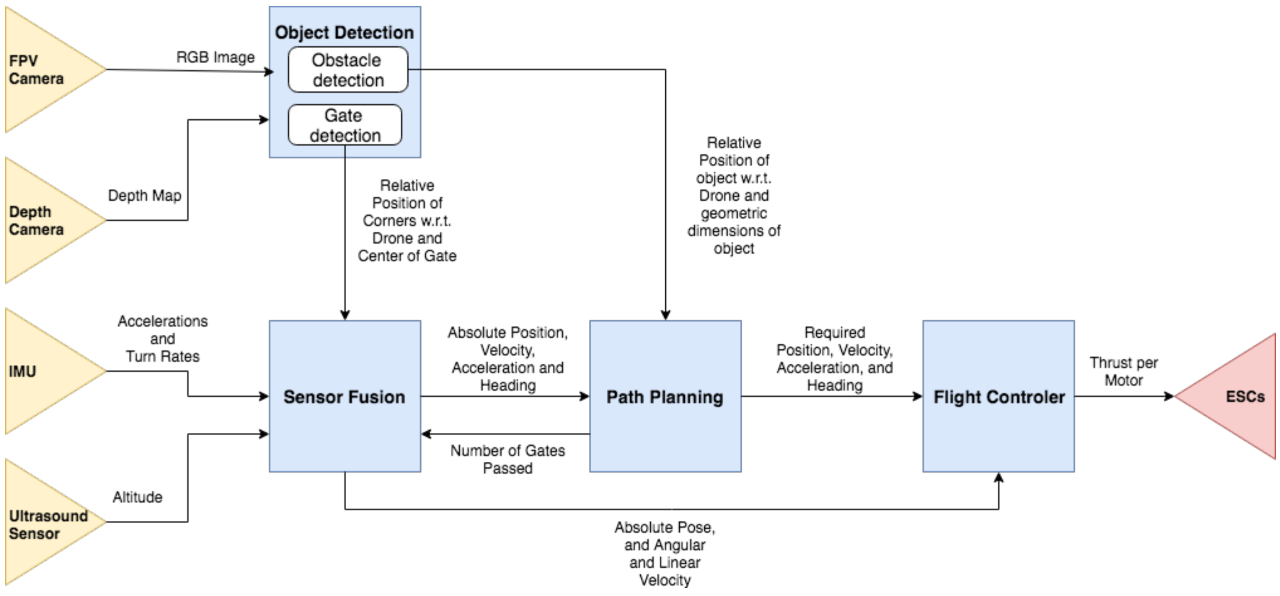

FPV Camera
RGB Image
Object Detection
Obstacle detection
Gate detection
Depth Camera
Depth Map
Relative Position of Corners w.r.t. Drone and Center of Gate
Relative Position of object w.r.t. Drone and geometric dimensions of object
IMU
Accelerations and Turn Rates
Ultrasound Sensor
Altitude
Sensor Fusion
Absolute Position, Velocity, Acceleration and Heading
Number of Gates Passed
Path Planning
Required Position, Velocity, Acceleration, and Heading
Flight Controler
Thrust per Motor
ESCs
Absolute Pose, and Angular and Linear Velocity

Figure 4.3: The figure shows the software block diagram.

### 4.3.1. Payload

In general, the sensors should have a small size and a low power consumption. First the IMU will be discussed. After that the camera selection will be explained. Third, the team's choice of altimeter is elaborated upon and finally the choice of onboard computers is explained.

#### IMU

The IMU is the most critical component of the drone, as its localization system strongly relies on integrated measurements of its accelerometers and gyroscopes. The drawback of using IMU measurements is that its estimate becomes too imprecise after few seconds. It has multiple sources of error such as the initial bias and the vibrations exerted by the propellers. These errors are introduce in the integration constants, and for the position they propagate quadratically. This is further elaborated upon in Section 7.1.1.

The VN-100 is the only sensor in its class to offer a quaternion based drift compensated Kalman filter. It operates with 32-bit floating point precision at update rates as high as 800 Hz[3]. Additionally, it can be calibrated for operational conditions, which minimize structural errors. This IMU has been selected among the best quality off-the shelf components, this is not only reflected on the sampling rate which is more than eight times higher than a standard IMU but also in the price.

#### Cameras

The cameras have the role of detecting gates and obstacles, in particular it should be lightweight and power efficient. RGB race drone cameras were considered for detecting gates because they are very light and cheap. In order to increase the probability of detecting gates, the RGB camera should have a high resolution and an high fps (frames per second). The CONNEX ProSight HX[4] was chosen for its low latency and high resolution with an average power consumption of 1 W and a mass of 13 grams. More specifically, it has a resolution of 1280x720, a vertical field of view of 59.06 degrees and a horizontal field of view of 105 degrees.

In order to increases the robustness of the gate detecting approach, the RGB camera is paired with a ToF camera. Depth is in fact an extra useful parameter, which is independent of object color. These type of technology is currently under development and multiple cameras exist, that match the requirements. The Pico Flexx[5] ToF camera weighs only 8 grams with a length of 68 mm. It is connected to the onboard computer using a USB2.0/3.0 interface. The only drawback is that the maximum range of this camera is 4 meters. it has a reolution of 224 * 171, a horizontal resolution of 62 degrees and a vertical resolution of 45 degrees. This is enough to be useful for detecting the gates and obstacles, but not enough to allow it to fully replace the RGB camera.

[3] `https://www.vectornav.com/products/vn-100` [Visited at May 24 2018]
[4] `https://www.amimon.com/fpv-market/prosight-product-page-2/` [Visited at May 24 2018]
[5] `https://pmdtec.com/picofamily/assets/datasheet/Data-sheet-PMD_RD_Brief_CB_pico_flexx_V0201.pdf` [Visited at May 24 2018]

#### Altimeter
An altimeter is used to provide an accurate measurement of the drone altitude. An ultrasound senor is preferred over a barometer because at low altitude, of approximately 1-5 meters, ultrasound sensors are more accurate. The MB1240 XL-MaxSonar-EZ4 was chosen. The sensor is designed and calibrated to provide reliable information even in environments with strong acoustic or electrical noise sources[6], which helps reduce measurement noise due to the sound coming from the propellers and motors.

#### On-board Computers
The main purpose of having two cameras is to increase the gate detection capability, and thus the autonomy of our drone. As more sensors are added more computational power is needed to process all of the information. Two computers are taken into consideration: the Raspberry Pi Zero W and the Pocket Beagle. The specifications of both can be found in Table 4.3[7].

Table 4.3: Specifications for processors

| | **Pocket beagle** | **Raspberry Pi Zero W** |
|---|---|---|
| **Size** | 56x35x5mm | 65x30x5mm |
| **Processing Power** | ARM Cortex A-8 1GHz, ARM Cortex M-3, 2 PRUs at 200MHz | single Broadcom BCM2835 1GHz |
| **Memory** | 512MB DDR3 RAM, 64KB Dedicated Memory | 512MB DDR2 RAM |
| **Connectivity** | -USB2.0<br>-SD slot | -USB2.0<br>-SD slot<br>-CSI camera interface<br>-Mini HDMI, WiFi, Bluetooth |
| **Price** | €25,- | €10,- |

Since the Pocket Beagle does not have a dedicated camera interface a lot of precious processing power is lost in the decoding of the signal. [8] to connect the camera to the microcontroller. Although this interface can be simulated, using the PRU inside a dedicated peripheral is preferred, which makes the Raspberry Pi Zero W perfectly suited for this task.
As a single processor is deemed to be incapable of managing all operations alone, path planning, gate detection and positioning, with the computational power available.[9] The depth camera chosen uses a usb connection and therefore there is no need for an CSI interface. Since the Raspberry Pi has all the interfaces needed for processing the image of the depth and RGB camera, it will be used for image processing, gate detection and gate dewarping. Given that the Cortex A-8 has better computing capabilities according to benchmarks [10], the Pocket Beagle will be used as main processing unit.

### 4.3.2. Planform Design
Special care was taken to make sure the smallest possible propellers were used, this resulted in 4 x 4.5 inch propellers being chosen. The reason for the team choosing the smallest possible propellers is the design philosophy. Since the propellers take up most of the horizontal space on the drone, they have to be as small as possible in order to allow the drone to be as small as possible.
These propellers then had to be combined with motors. It is deemed that the motor should have the following minimal configuration KV-rating of 2500, thrust of 5 N, and a max peak current of 14 A. The F30 KV2800 from T-Motor for instance, complies with these requirements, and in the specifications they were tested with the propeller size selected [11]. Once the motor is designed, the battery can be selected considering the hoover requirement of 10 minutes. In this preliminary phase the Turnigy 1300mAh 4S[12] was selected.

After sizing the propellers, the rest of the components could be distributed over the frame. Some initial consideration are reported as follows. Both of the cameras need to be in the front. Then, the computer boards and Power

[6] *https://www.maxbotix.com/Ultrasonic_Sensors/MB1240.htm* [Visited at May 25 2018]
[7] *https://www.arrow.com/en/research-and-events/articles/comparing-pocketbeagle-specs-and-rpi-zero-specs* [Visited at May 25 2018]
[8] *https://www.mipi.org/specifications/csi-2* [Visited at May 25 2018]
[9] *https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/* [Visited at may 25 2018]
[10] *https://www.teachmemicro.com/raspberry-pi-zero-vs-pocketbeagle/* [Visited at May 25 2018]
[11] *http://store-en.tmotor.com/goods.php?id=303* [Visited at May 25 2018]
[12] *https://hobbyking.com/nl_nl/turnigy-1300mah-4s-45c-lipoly-pack.html* [Visited at ]

Distribution Board (PDB) should be put behind the camera on the top of the frame. Here, they will be cooled by the airflow produced by the propellers, which allows them to be stacked on top of each other. The IMU was placed behind the other components, as far away from large currents as possible to lower the amount of interference it will encounter. Lastly, for the top side of the frame, the ESC's were placed near each of the motors to save space near the middle of the drone. After this, the battery was placed in the middle in order to keep the center of gravity as close to the center of the drone as possible, allowing for the best possible angular accelerations. Finally, the ultrasound sensor is placed on the bottom.

### 4.3.3. Preliminary Resource Allocation

The list of components developed in Section 4.3.1 and Section 4.3.2 is used to make an estimate of the preliminary mass, power and cost for RAIDER, shown in Table 4.4. The final resource allocation of the design is shown in Chapter 12.

Table 4.4: Top level mass, power, and cost budget

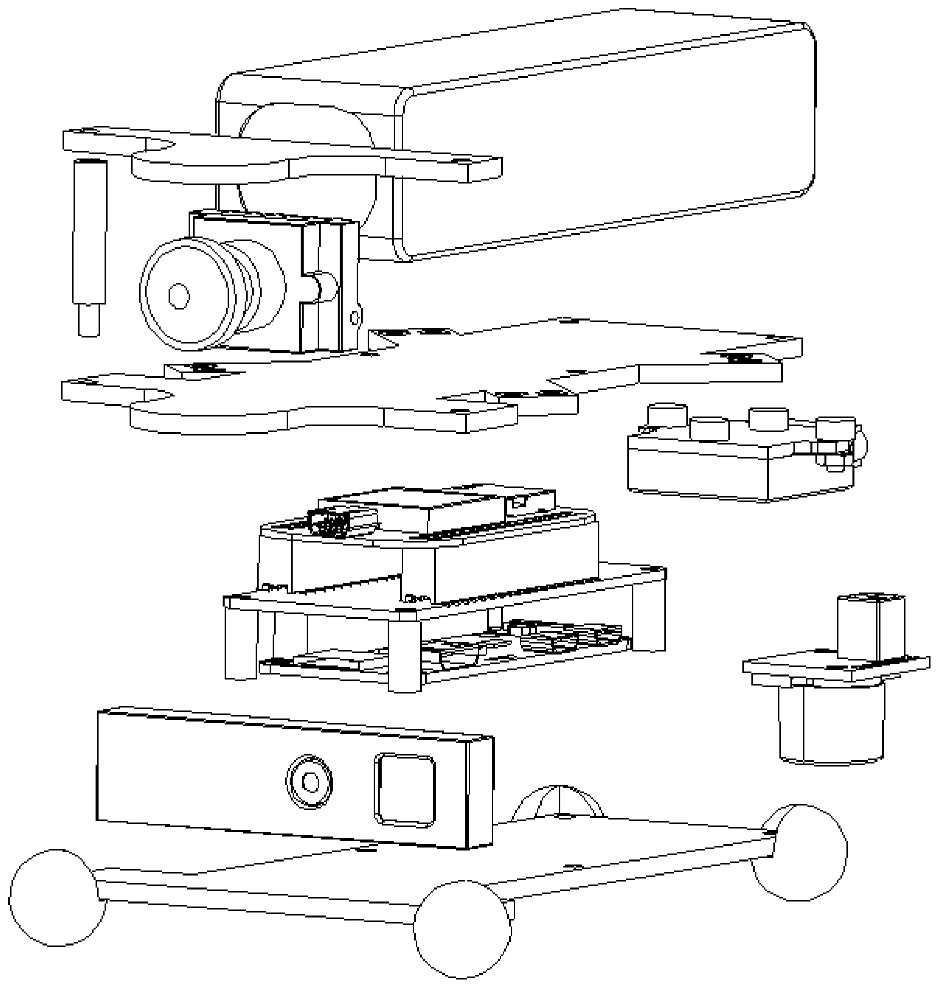
| | # | **Piece mass[g]** | **Piece power[W]** | **Piece cost[€]** |
|---|---|---|---|---|
| Computer | 1 | 15.0 | 6.0 | € 25.00 |
| Graphics computer | 1 | 9.0 | 5.0 | € 25.00 |
| Depth camera | 1 | 8.0 | 0.3 | € 300.00 |
| RGB camera | 1 | 14.0 | 1.0 | € 50.00 |
| Ultrasonic altimeter | 1 | 6.1 | 0.5 | € 35.00 |
| Power distribution board | 1 | 10.0 | 1.0 | € 100.00 |
| Electronic speed controller | 4 | 10.0 | 0.5 | € 15.00 |
| Motor | 4 | 80.0 | 23.0 | € 15.00 |
| Propeller | 4 | 2.0 | 0.0 | € 2.00 |
| Frame | 1 | 65.0 | 0.0 | € 100.00 |
| Battery | 1 | 154.0 | 0.0 | € 30.00 |
| Cables | 1 | 10.0 | 0.0 | € 10.00 |
| IMU | 1 | 15.0 | 0.5 | € 450.00 |
| **Total** | | 398.1 | 108.3 | € 1.253.00 |

A final mass budget is compiled using the mass of the components selected. The motor, ESC, propeller, battery, sensors and computers weights are taken from component chosen in the previous sections. The frame and PDB, however, will be designed for the drone, in the table the weights reported come from commercially available alternatives with similar requirements.
Several elements still need to be considered in detail, such as the mounting and the wire length for each sensor. The cable weight is assumed to be 10 grams for the moment.

Power management is important to ensure the battery is capable of delivering the required power for the required time. As opposed to a normal race drone an autonomous race drone has a lot of components that use power. A preliminary power budget is used to manage the amount of power each component can use and to size the battery. Even if most components have been selected, the actual power might still vary. For instance, the power used by sensors depend greatly on the update frequency required. Computing units also have a degree of uncertainty. It is not known which interfaces will be used. In general for all components, but in particular for the motors, the usage profile is not yet tested before and therefore precise results are unknown. Also the loss of power in cables or imperfect soldering is not accounted for, but this is deemed very small.

To keep repair cost down parts should be easily replaceable and cheap. For each component a financial budget has been allocated. Although re-budgeting might be necessary later on,this budget is to ensure that the components can cheaply be replaced by new ones.

# II
# Detailed Design



# 5
# Structural design

To complete the mission the drone should stay in one piece. The two parts that were identified to be subjected to the highest loads are the arms and the shock absorbers. The critical load cases in different direction were identified for both. They were taken directly from the requirements. Furthermore the IMU damping was identified to be of paramount importance as RAIDER will depend highly on the IMU. Lastly using the risk map from Chapter 18 the team decided the drone needs propeller guards to protect the propellers and decrease the chance of crashing.

## 5.1. Arms Design

The arms protrude from the second deck outwards and attach the motors to the frame. They transfer the motor loads to the center of gravity. The arms are attached to the frame in the center of the body.

### 5.1.1. Loading

The two load cases that were identified are: Thrust of the motor at the tip and impact with a gate at 100km/h.

#### Case 1: Thrust of the Motor

As stated above, the arm attaches the motor to the frame. The maximum thrust of the motor needs to be transferred to the frame where the center of gravity is located. This is a quasi constant load as the arms should be capable of handling these loads for a long time.

| **Load case summary** | |
|---|---|
| Type | Constant force |
| State | When performing high g manoeuvres |
| Location | At the tip of the arm |
| Orientation | In negative z direction of the body reference frame |
| Magnitude | 6.1N |

#### Case 2: Impact of Arm with Gate

Impact is modeled at the tip of the arm. As a consequence of the impact the drone will rotate around the gate.

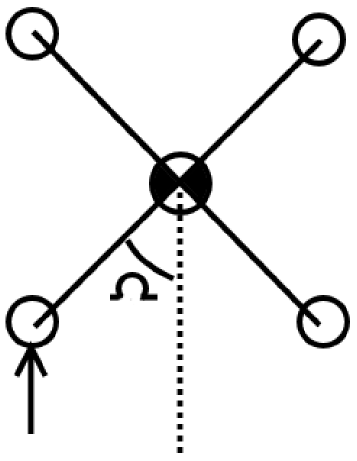


Figure 5.1: Impact with a gate at the end of a the arm

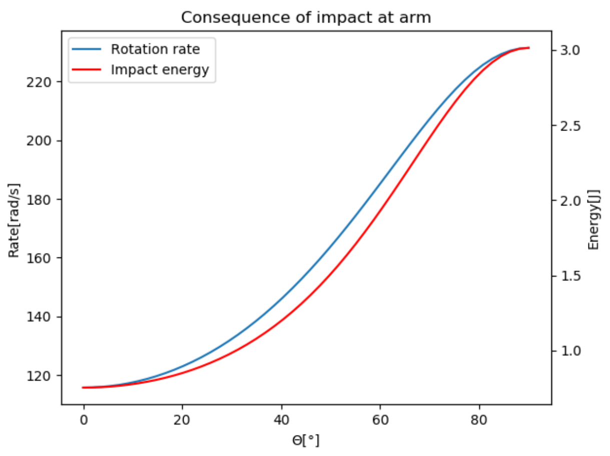


Figure 5.2: Impact energy at the tip of the arm

Due to the drone's inertia and high forward speed it is assumed the centre of gravity of the drone will stay at its initial trajectory and the drone will need to rotate since the arm can not go through the solid boundary of the gate. The angle needed for the rotation is derived using Figure 5.1. The absolute angle($\Omega$) and its derivative($\omega$) are given by Equation (5.1) and Equation (5.2) respectively. $\Omega$ is defined as shown in Figure 5.1.

$$\Omega = arctan\Big(\frac{0.147}{x}\Big) \tag{5.1}$$

$$\omega = \frac{0.147}{x^2 + 0.147^2} \cdot v \tag{5.2}$$

The velocity v is assumed to stay constant, but distance x depends on the pitch angle ($\theta$) of the drone. As the drone will be pitching down more than 50 degrees when flying at high speeds, this needs to be taken into account. The distance x is then given as a function of $\theta$ by Equation (5.3).

$$x = cos(\theta) \cdot 0.147 \tag{5.3}$$

The energy that is needed to obtain the near instantaneous rotation can be calculated by multiplying the impact force with its work. The latter of which is compiled into a energy conservation equation. The former is calculated using Equation (5.4).

$$U_{work} = \frac{1}{2} I \omega^2 \tag{5.4}$$

The inertia (I) was estimated using a radius of gyration that puts all of the mass in a sphere with a 1.5cm radius. This covers the entire width of the frame and should therefore be representative for the entire drone. The impact energy is highly dependent on the pitch. The ratio between the two is shown in Figure 5.2. In this graph, it can be seen that the impact energy increases with the pitch angle. The maximum impact energy occurs when the drone has a pitch angle of 90°. However, according to analysis, the drone will be at a pitch angle of approximately 50°at 100km/h.

| **Load case summary** | |
|---|---|
| Type | Impact force |
| State | When flying at 100km/h at a pitch angle of 50° |
| Location | At the tip of the arm |
| Orientation | Opposing the velocity vector |
| Magnitude | 1.2J |

### 5.1.2. Design

Meeting the impact load case is very demanding. Therefore, a trade-off was made about how this load case should be approached. Two option are identified: The drone should stay fully impact or the arm is allowed to break off at a predetermined place. The latter is to ensure the arm is not completely lost and can easily be replaced.

Although both options are feasible, they handle the impact very differently. They will not result in very different arm designs, but this choice will have great consequences on the overall design. If the structure is not to fail in any way the arms are directly attached to and part of the middle frame. However the high spin rate this would induce would result in a hard crash. For this reason the following approach was chosen; The arms are not allowed to break off, but should protect the drone sufficiently. Furthermore the overall structural design should protect all vital components.

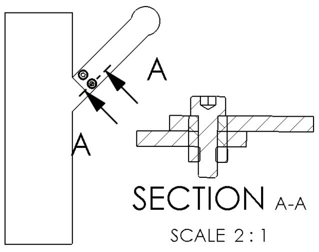


Figure 5.3: Arm Design

The design will consist of four arms, all of which are connected to the center frame plate. The connection mechanism is designed such that it will partially deform on impact, but will not to break during such a crash. The same holds for the carbon fibre arm.

The arm will be made out of a solid carbon fibre laminate. The thickness needed will be determined using the structural analysis described in Section 5.1.3. The arm is mounted on top of the center frame plate. The amount of overlap, bolt and insert size will follow from the first structural analysis in which the thrust load case is considered. An overview of the arm is given in Figure 5.3.

#### Material

Different materials were considered for the design of the arms. All materials in consideration are shown in Table 5.1.

Table 5.1: Arm material trade off

| **Material** | **Yield stress** $[MPa]$ | **Youngs modulus** $[GPa]$ | **Density** $[g/cm^3]$ | **Raw price** $[€/kg]$ |
|---|---|---|---|---|
| Steel | 350.0 | 200.0 | 7.80[1] | 0.72[2] |
| AL-7075 | 503.0 | 71.7 | 2.81[3] | 11.31[4] |
| Carbon composite with resin[5] | | | 1.45 | 298.50 |
| Carbon laminate(0°tension) | 803.0 | 57.1 | | |
| Carbon laminate(0°compression) | 750.0 | 54.8 | | |
| Carbon laminate(90°tension) | 722.0 | 56.4 | | |
| Carbon laminate(90°compression) | 741.0 | 48.7 | [6] | |
| Resin | 71.0 | 3.2 | | |
| Titanium | 590.0 | 103.0 | 4.51[7] | 56.97[8] |

The carbon fiber plates with a resin core are by far the lightest. These are also the most popular option in the market for FPV drones and have therefore proven to be up to the job. Downsides of using carbon fibres plates are their cost and difficulty to manufacture. Due to the fact that they are anisotropic and consist of two materials they are harder to model.
For the design four different thicknesses were taken into consideration: 0.7mm, 1.5mm, 2.2mm and 2.9mm. The thicknesses depend on the fibre laminate which has single ply thickness of approximately 0.7mm.

### 5.1.3. Model

Modeling composites can be difficult and time consuming, due to the fact that composite structures do not behave isotropically[9]. However the carbon plates that will be used for the drone are much thicker than a single sheet of carbon and consist of multiple layers of carbon oriented in different directions.

#### Assumptions

During modeling the following assumptions are made:

- **The carbon layers act as isotropic materials**
  Although carbon fibres are not isotropic by themselves when a proper layup is used(0°-90°mixed with 45°) the laminate can be assumed to be quasi isotropic[58]. Furthermore quasi isotropic properties can be obtained for carbon composites from the manufacturer's datasheets. These show that the transverse properties are within 5% of the longitudinal properties[10].

- **Small deformations**
  All deformations resulting from forces are small, this ensures that all deformations are independent of each other.

- **Rigid cross section**
  It is assumed that the cross section will not change shape due to the loading. Since all deformations will be small, forces will only undergo very small orientation changes due to their different places in the structure. Furthermore all angles are small so integration can be done by multiplication.

- **Rigid frame**
  During this design phase the focus is on the arm and the bolts that keep it secure. Due to size of the interference with the arm, it can be assumed to be rigid and replaceable by a single force.

- **Safety factor 1.5**
  Although carbon fibre layups can be unpredictable, most properties are known for the proposed materials.

[1] `http://www.matweb.com/search/datasheet.aspx?bassnum=MS0001&ckck=1` [Visited at June 5 2018]
[2] `http://www.meps.co.uk/World%20Carbon%20Price.htm` [Visited at June 5 2018]
[3] `http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA7075T6` [Visited at June 4 2018]
[4] `https://www.mcb.eu/en/metal/aluminum/aluminum-sheet/flat/miscellaneous/aluminium-plate-strip-en-aw-7075-t651/p/2800-0056` [Visited at June 4 2018]
[5] `https://www.rockwestcomposites.com/plates-panels-angles/carbon-fiber-plate/carbon-fiber-fabric-plate/403-410-group` [Visited at June 5 2018]
[6] `https://www.toraycma.com/files/library/166e096be76c7eb7.pdf` [Visited at June 8 2018]
[7] `http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MTU041` [Visited at June 5 2018]
[8] `http://www.steelforge.com/raw-materials/titanium-forgings/` [Visited at June 5 2018]
[9] `https://appliedcax.com/docs/presentations/FEMAP-Symposium-2016-Basics-of-FEA-Composite-Modeling.pdf` [Visited at 26 June 2018]
[10] `https://www.toraycma.com/files/library/166e096be76c7eb7.pdf` [Visited at June 8 2018]

Therefore, as the structure is rather simple, a safety factor of 1.5 as used in the aerospace industry was deemed enough.

- **Infinitesimal small bolts**
  The bolts are modelled as point forces instead of around the entire hole as will be done in the FEM analysis. The stress around the bolts will be higher in reality than modeled, but this assumption simplifies the model and allows the use of single constant cross section.

### External Loads in Case 1

For the first load case the free body diagram in Figure 5.4 was made. By taking moments around the y-axis at the location of the $F_{frame}$ the force on the bolts can be calculated. Both bolts will be taking the same amount of force as they are symmetrically placed and the motor force only creates a moment around y. The frame force is the normal force of the center frame on the arm that ensures that the arm does not intersect with the frame. Together with the bolt force they counteract the moment created by the thrust. The full static equations of motion can be found in Equation (5.5).

$$\begin{aligned}\sum F_x &: 0\\ \sum F_y &: 0\\ \sum F_z &: 0 = & -F_{thrust} + F_{bolt1} + F_{bolt2} - F_{frame}\\ \sum M_x &: 0 = & r_{y,bolt1} \cdot F_{bolt1} - r_{y,bolt2} \cdot F_{bolt2}\\ \sum M_y &: 0 = & r_{x,thrust} \cdot F_{thrust} - r_{x,bolt1} \cdot F_{bolt1} - r_{x,bolt2} \cdot F_{bolt2}\\ \sum M_z &: 0\end{aligned} \tag{5.5}$$

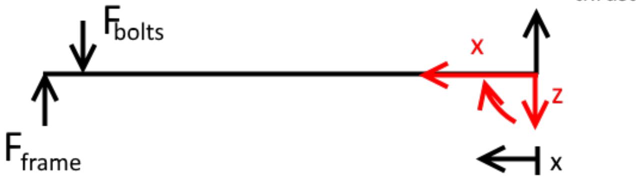


Figure 5.4: Free body diagram of load case 1

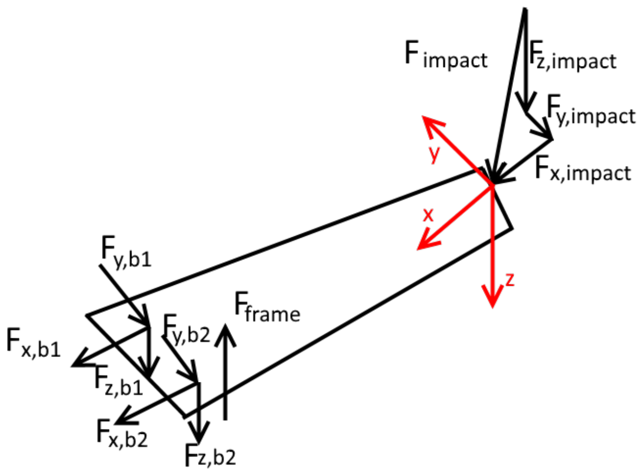


Figure 5.5: Free body diagram of load case 2

### External Loads in Case 2

The second load case is a bit more complex as the impact force is under an angle. Furthermore the precise force is unknown until the deformations are solved and the force at the point of impact are solved for. This yields a total of 8 unknowns: The six reaction forces at the bolts, a reaction forces by the frame and an unknown force due to the impact at tip of the arm.

The reaction of the frame is placed on the x-axis at the end of the overlapping part of the frame. In reality the frame will bend slightly with the arm, however since the frame is assumed to be rigid the arm will solely rotate around the end of the frame.

Six equations are given by the equilibrium equations in Equation (5.6).

$$\begin{aligned}\sum F_x &: 0\\ \sum F_x &: 0 = & F_{x,bolt1} + F_{x,bolt2} + F_{x,impact}\\ \sum F_y &: 0 = & F_{y,bolt1} + F_{y,bolt2} + F_{y,impact}\\ \sum F_Z &: 0 = & F_{z,bolt1} + F_{z,bolt2} + F_{frame} + F_{z,impact}\\ \sum M_x^A &: 0 = & F_{z,bolt1} \cdot r_{y,bolt1} + F_{z,bolt2} \cdot r_{y,bolt2}\\ \sum M_y^A &: 0 = & F_{z,bolt1} \cdot r_{x,bolt1} + F_{z,bolt2} \cdot r_{x,bolt2} + F_{frame} \cdot r_{x,frame}\\ \sum M_z^B &: 0 = & -F_{x,bolt1} \cdot r_{y,bolt1} + F_{x,bolt2} \cdot r_{y,bolt2} - F_{x,impact} \cdot r_{y,impact}\end{aligned} \tag{5.6}$$

The displacement of the tip is caused by the deflection of the carbon plates and the deformation of the bolts. Both should be taken into account as the force due to the load rapidly decreases with a higher impact deformation, as given by Equation (5.7).

$$\begin{aligned} F_{impact} &= \frac{U_{impact}}{d_m} \\ F_{x,impact} &= \hat{i_x} \cdot F_{impact} \\ F_{y,impact} &= \hat{i_y} \cdot F_{impact} \\ F_{z,impact} &= \hat{i_z} \cdot F_{impact} \end{aligned} \tag{5.7}$$

The impact force can be related to the displacement. Although this gives 10 equations in total, another unknown, the displacement, was introduced. To this extend there are 12 unknowns and 10 equations; not enough for a solution.
The displacement at the tip of the arm is a combination of two effects: The bolts' deformation and the structure's deformation. The former is rather difficult to calculate as of yet. The latter, however, can easily be taken into account.
The bolts will be made of aluminum to be as light as possible. Their deformations can be related to their respective forces using Equation (5.8).

$$\begin{aligned} d_{z,bolt} &= l_{bolt}\frac{F_{z,bolt}}{E_{bolt}A_{bolt}} \\ d_{y,bolt} &= l_{bolt}\frac{F_{y,bolt}}{G_{bolt}A_{bolt}} \\ d_{x,bolt1} &= l_{bolt}\frac{F_{x,bolt1}}{G_{bolt}A_{bolt}} \\ d_{x,bolt2} &= l_{bolt}\frac{F_{x,bolt2}}{G_{bolt}A_{bolt}} \end{aligned} \tag{5.8}$$

These are combined with geometrical constrains to produce the displacements at the tip, which can be calculated according to Equation (5.9).

$$\begin{aligned} d_{z,m} &= d_{z,bolt}\frac{r_{CA}}{r_C B} \\ d_{x,m} &= \frac{1}{2}d_{x,bolt1} + \frac{1}{2}d_{x,bolt2} \\ d_{x,m} &= d_{y,bolt} + d_{x,bolt1}\frac{r_{AB}}{r_{y,bolt1}} - d_{x,bolt2}\frac{r_{AB}}{r_{y,bolt2}} \\ d_m &= d_{x,m}\hat{i_x} + d_{y,m}\hat{i_y} + d_{z,m}\hat{i_z} \end{aligned} \tag{5.9}$$

The displacement of the arms can now be calculated. With a total of 19 unknowns and 18 equations, the system is still not fully determined. The last equation results from the fact that both bolts need to stay at the same position. They are not allowed to move differently in the y direction and therefore the last constraint is given by Equation (5.10).

$$F_{y,bolt1} = F_{y,bolt2} \tag{5.10}$$

By linearizing and iterating, the entire system consisting of Equation (5.6)-Equation (5.10) can be solved. This results in all external forces acting on the arm.
At first the effects of the deformation of the arm was not taken into account when calculating the deformation. This, however, resulted in very high stress concentrations in the arm and deflection that were the same size as the bolt deformation. This was solved using an iterative process in which a certain deformation was added to the external solving methods. The program stop iterating when the error between the added displacement and the actual deformation of the structure dropped below a set threshold.

### Internal loads

Using the external loads calculated in Section 5.1.3 and Section 5.1.3, the internal loads were calculated by taking the moments around Y and Z. Since none of the load cases introduced a torque, it was not taken into account.
Since the carbon plates were assumed to be isotropic, Equation (5.11) was used to model the bending stress.

$$\sigma = \frac{F_{normal}}{A} + \frac{M_z y}{I_{zz}} + \frac{M_y z}{I_{yy}} \tag{5.11}$$

The stress was calculated for the top and bottom corners of a cross section as these will be the places were the highest stress will occur. The deformation of the plate was calculated by integrating the deformation of a single section over the entire beam. This method allows for extra forces and ensures that the same model can be used for both cases. The deformation of a single cross section is given by Equation (5.12).

$$\begin{aligned} \frac{dw}{dx} &= \frac{M \cdot x}{EI} \\ w = &= \frac{M \cdot x^2}{2EI} \end{aligned} \tag{5.12}$$

### 5.1.4. Results

The models have been run for four thickness of the arm: 0.7mm, 1.5mm, 2.2mm, and 2.9mm. The maximum stress for each was calculated in the second load case using the methods described before. The 2.9mm carbon was chosen as it is the only plate that can withstand the maximum stress caused by an impact at 100 km/h. All results in this section are all calculated for a plate thickness of 2.9mm.

#### Case 1: Results

The thrust of the motor is relatively low due to the low weight of the drone. However, due to the small overlap of the frame and arm, the force on the arm and frame are relatively high compared to thrust force. The exact numbers are shown in Table 5.2.

Table 5.2: External force load case 1

| External force | Point of application $[m]$ | Magnitude $[N]$ |
|---|---|---|
| thrust | (0.000 0.000 0.000) | (0.0 0.0 -6.1) |
| Bolt 1 | (0.068 -0.003 0.000) | (0.0 0.0 44.7) |
| Bolt 2 | (0.068 0.003 0.000) | (0.0 0.0 44.7) |
| Frame | (0.073 0.000 0.000) | (0.0 0.0 -83.3) |

Table 5.3: Internal loads case 1

| | Magnitude | |
|---|---|---|
| Plate maximum stress | 23.7 | MPa |
| Plate minimum stress | -23.7 | MPa |
| Plate bending in z direction | 0.5 | mm |
| Bolt normal stress | 11.8 | MPa |
| Bolt shear stress | 0.0 | MPa |
| Bolt Von Mises stress | 11.8 | MPa |

As can be seen in Table 5.2, the forces on the bolts are very small as is the force that the frame exerts on the arm. This load case does not induce much bending in the structure.
In Table 5.3 the stresses in the structure are shown. None of them exceed the critical stress of 750MPa for the carbon plate. Therefore, the structure will not have any issue dealing with the stresses induced by the thrust.

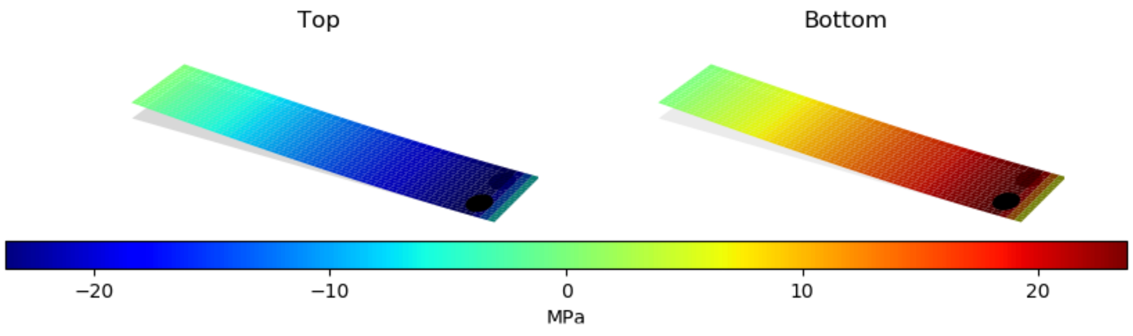


Figure 5.6: Case1: Modelled stress with a deformation scale of 20

The stress distribution in the frame is shown in Figure 5.6. It can clearly be seen that the stress is highest near the bolts and gradually decreases towards both ends of the arm.
The stress in the arm due to the thrust loading bends the arm upwards, introducing compressive stress in the top part of arm and tensile stress in the bottom part, as shown in Figure 5.6.

#### Case 2: Results

The second case was a lot more challenging to solve, especially due to the complex external loading. This was done as described in Section 5.1.3 using an iterative linear solver. The external forces are summarized in Table 5.4.

Table 5.4: External force load case 2

| External force | Point of application $[m]$ | Magnitude $[N]$ |
|---|---|---|
| Bolt 1 | (0.068; 0.003; 0.000) | (409.7; 19.7; 965.9) |
| Bolt 2 | (0.068; -0.003; 0.000) | (-449.2; 19.7; 965.9) |
| Frame | (0.063; 0.000; 0.000) | (0.0; 0.0; -2085.2) |
| Impact | (0.000; 0.000; 0.000) | (39.5; -39.5; 153.4) |

Table 5.5: Internal loads case 2

| | Magnitude | |
|---|---|---|
| Plate maximum stress | 584.5 | MPa |
| Plate minimum stress | -582.3 | MPa |
| Plate z bending | 10.1 | mm |
| Bolt normal stress | 254.1 | MPa |
| Bolt shear stress | 118.2 | MPa |
| Bolt Von Mises stress | 280.2 | MPa |

The external forces introduce a final deformation of approximately 6mm at the tip due to the elongation and deformation of the bolts. Coupled with a deformation due to plate bending of 10mm, the magnitude of the impact force is calculated to be 180N at deformation of 20mm. The stress distribution is shown in Figure 5.7.

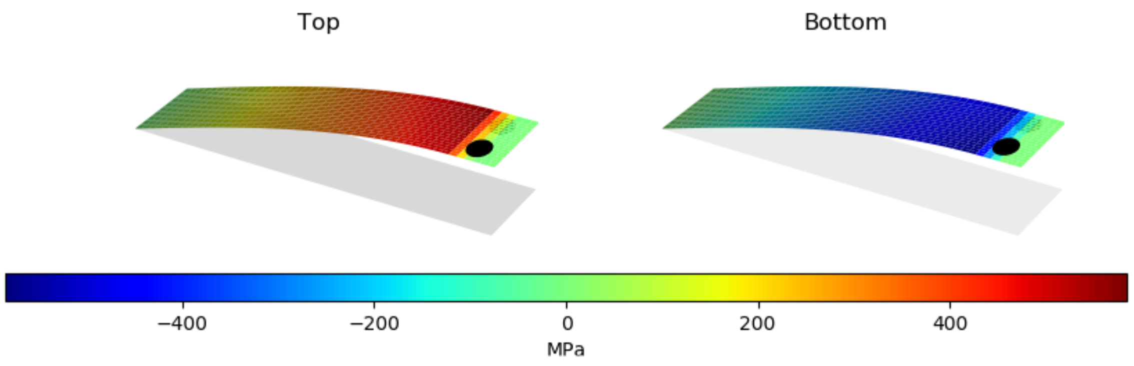


Figure 5.7: Case2: Modelled stress with a deformations scale of 4

As can been seen in Figure 5.7 the internal stress reaches zero at the locations of the bolts. The difference is explained by the downward bending instead of up and the fact that the frame force is now applied at the start of the overlap. The deformation in the carbon plate is far more excessive than it was in the first load cases. Note that the deformation scale used is 5 times as small as for the first load case. The large bending is beneficial as it reduces the load of the impact.

## 5.1.5. Verification

The model for internal stresses was verified using the following test cases:

- The bending of the arm was validated using simple formulas for beam bending. The arm is roughly clamped at the side of the frame as the bolts and frame prevent it from bending. This yields the bending of the arm should be comparable to the bending of a clamped beam, for which Section 5.1.5 holds.

$$\delta = -\frac{FL^3}{3EI}$$

  In the first case of the applied thrust the total bending of the beam was modeled to be 1.1mm; the formula results in a 1.2mm deflection. This is caused by the bolt locations; The arm can not start bending before the bolts, whereas the clamped beam can.

- When no load is applied to the structure, the structure does not deform. Nor are any reaction forces present. Furthermore for each load case the static equilibrium was checked before the internal stresses were calculated.

## 5.1.6. Validation

To make sure the models are correct validation is performed using software that implements Finite Element Anaylsis (FEA). The first load case of the arms was validated by fixing the location of the bolts and applying the thrust force at the tip of the arm. The entire structure is drawn using Solidworks, which has a built in FEM workspace that was used for the analysis. The stress distribution is shown in Figure 5.8 and Figure 5.9.
A big difference can be seen around the holes of the bolt. These were not modelled in the program, but high stress concentration can be found around them. The main reason is the modeling constrained used, as fixed geometry can

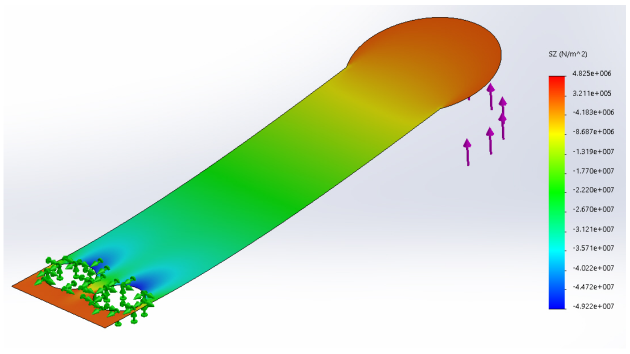


Figure 5.8: Case1: Stress in the top calculated using Solidworks

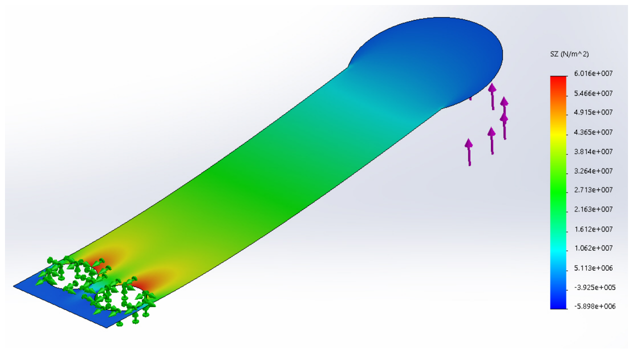


Figure 5.9: Case1: Stress in the bottom calculated using Solidworks

introduce a lot of stresses that are different than for a bolt. A bolted connection always has some play, whilst a fixed geometry constrain does not.
When we the holes are ignored and the bottom stress is probed just before the holes the FEA predicts a stress of 24MPa. This number is very similar as the numbers found in the model, which were 23.7MPa. Furthermore when the same is done for top part of the structure -27MPa in the FEA and -23.7MPa in the model. Both results are close together, yielding that the model predict the stress rather accurately.

## 5.2. Crumple Zones

The drone will crash many times while it is being tested and optimized. In order for fast development to be possible, the drone has to be able to keep going after a crash. In order to achieve this, a sturdy structure with quickly replaceable crumble zones are needed. This lowers the forces on the electronics and sensors. Although some components may be able to handle loads up to 1000g[53], the g-load for the drone is limited to the lowest rating of 500g[34].

### 5.2.1. Loading

The drone has a top speed of 100km/h whilst having a mass of about 700 grams. This yields a total impact energy of 270J, which should be completely absorbed by the drone. This is a lot higher than the impact at the tip of the arm as the drone is not assumed to tumble after the impact. However unless the drone is flying into a wall at full speed, this will not happen. In case the drone loses all of its lift when flying at 100km/h it will skid over the ground gradually slowing down. And when it hits a gate at 100km/h it will tumble over it, without losing too much speed as can be seen in the previous section.
The drone should be able to stay intact when it suddenly loses lift and falls on the ground. The stakeholder requirement states that this could happen from up to a maximum height of 3m. Combined with the mass of 700 grams, this is a potential energy of 20.5J. The shock absorber has to be able decrease the g-load below g00g in the case this happens.

### 5.2.2. Design

The top of the internal structure is protected by propeller guards which will be designed later. They stick out above the frame and will therefore most of the impacts from above. However the lower side of the frame is still unprotected and in case the drone falls to the ground the impact energy is too high to survive this without protection. In order to alleviate some of the impact energy, foam balls will be placed around each corner of the frame to soften the impact from each angle.
The radius of the ball should be higher than the distance needed to accomplish a deceleration lower than 500g. This yields a radius of at least 6mm. Since the foam only compresses to about 20% of its initial size, the balls' radius was set at 7mm[12].

#### Material

Foam consists of bubbles of gasses trapped in cells. Usually air is used, although other gasses are also seen. The cell structure is made using polymers which are of most interest in designing the damping structure. They define the plateau region that can be seen in Figure 5.10. Therefore, the different materials shown in the Table 5.6[11,12,13] were considered for the crumple zones.

[11] `https://plastics.ulprospector.com/generics/45/c/t/polyurethane-pur-properties-processing` [Visited at June 11 2018]
[12] `https://www.makeitfrom.com/material-properties/Polystyrene-PS` [Visited at June 11 2018]
[13] `http://www.vinidex.com.au/technical/material-properties/polyethylene-properties/` [Visited at June 11 2018]

Table 5.6: Materials for foam curmple zones

| Material | Density $[g/cm^3]$ | Youns modulus [GPa] |
|---|---|---|
| Polyurethane | 1.2 | 2.2 |
| Polystyrene | 1.0 | 2.4 |
| Polyethylene | 0.95 | 0.8 |

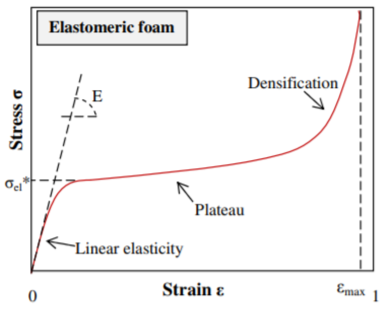


Figure 5.10: Stress strain relation for foam

### 5.2.3. Model

The main task is to act as a crumple zone of the drone. This will ensure that the speed is slowly bled off, reducing the deceleration. The maximum allowable deceleration is 500g, as discussed in the section intro. The amount of crumple zone as a function of the drone's speed is given by Equation (5.13).

$$d = \frac{0.5v^2}{500g} \tag{5.13}$$

Here, g is the acceleration due to gravity, v is the impact speed and d is the needed amount of crumble zone needed. This equation assumes the drone is brought to a complete still after crash and al velocity is lost. It can be related to energy using Equation (5.14).

$$d = \frac{U}{500g * m} \tag{5.14}$$

Both d and g represent the same variables as in Equation (5.13). Now, m is the mass of the drone and U is the impact energy.
Next to structure of the foam should be calculated. The stress strain curve of foam, see Figure 5.10[12], can be divided in three sections: The elastic phase, the plateau phase and densification phase.
The plateau phase is of most interest as in this region most of the energy can be absorbed. The elastic phase will be ignored for the design of the crumple zones since very little energy is absorbed here.
From the stress strain curve it can be assumed that the stress is linear while in the plateau phase and can be calculated using Equation (5.15).

$$\sigma = \sigma_{el} + \epsilon \cdot E_{pl} \tag{5.15}$$

The total non dimensional energy absorbed by the foam can be calculated by integrating Equation (5.15) over the strain, the result of which can be seen in Equation (5.16).

$$e = \sigma_{el}\epsilon + 0.5\epsilon E_{pl} \tag{5.16}$$

Adding the dimensions results in Equation (5.17).

$$U = A\sigma_{el}\frac{d}{l} + A\frac{d^2}{l^2}0.5E_{pl} \tag{5.17}$$

This can be solved to find the deformation distance from a certain energy, the result of which can be seen in Equation (5.18).

$$d = \frac{l\sqrt{A^2\sigma_{el}^2 + 2E_{pl}AU} - Al\sigma_{el}}{AE_{pl}} \tag{5.18}$$

In this equation d is used to denote the compression distance, l is the total height of the foam and A is the cross section area, $\sigma_{el}$ is the stress at which the plateau region begins and $E_{pl}$ is the stress strain constant in the plateau region.
The precise value for $E_{pl}$ is very hard to compute and is near zero as can be seen in Figure 5.10. Since the purpose of the model is to give an estimate anyway, it will be assumed to be zero. This greatly simplifies Equation (5.18) into Equation (5.19).

$$d = \frac{U}{A\sigma_{el}}l \tag{5.19}$$

The properties of the foam are expressed by $\sigma_{el}$, which describes the stress at which the foam starts failing. The failing of the foam is due to the cell walls buckling and can be modelled using a thin wall buckling. The buckling load is given by Equation (5.20) [12].

$$F_{crit} = \frac{\pi^2 E_s I}{l_c^2} \tag{5.20}$$

Here, $F_{crit}$ is the critical load at which the cell wall buckles, $E_s$ the Youngs modulus of the material used for the foam, I is the moment of inertia of the wall, and $l_c$ the cell wall height. This formula holds for square cells which in reality will not exists but it gives a good approximation. If cubic cells are assumed(width, length and height the same), the formula simplifies even further into Equation (5.21).

$$\sigma_{el} = \frac{\pi^2 E_s t^2}{12 l^2} \tag{5.21}$$

Using Equation (5.21) and Equation (5.19) the cell size in the foam can be accurately designed to give enough damping while still being lightweight. The weight of certain cell size can be deduced using the density of the polymer and Equation (5.22).

$$\rho_{foam} = 3\rho_s \frac{t}{l} \tag{5.22}$$

In Equation (5.22), $\rho_s$ is the density of the material used for the foam. Combining this, the foams can be optimized for their weight.

### 5.2.4. Result

The case in which a impact of 100km/h was taken into account was deemed unfeasible to reach as it would imply a crumble zone of over 8cm. As this is twice the width of the frame it was deemed much to big. Therefore only the second case, a fall from 3m was taken into account. To stay below the g limit a crumple zone of 6mm is needed.
For each material the foam cell structure was calculated, the results are shown in Table 5.7.

| **Material** | **Cell size** [mm] | **Cell wall thickness** [$\mu m$] | **Foam density** [$g/cm^3$] |
|---|---|---|---|
| Polyurethane | 0.5 | 2.01 | 0.0145 |
| Polystyrene | 0.5 | 1.93 | 0.0116 |
| Polyethylene | 0.5 | 3.35 | 0.0190 |

Table 5.7: Foam design for crash absorbers

As shown in the table polystyrene foam will be the lightest and adequate for the drop requirement. The protective ball crumple balls will have a radius of 7mm to allow for some extra crumble zone.

### 5.2.5. Validation

The same formulas were used in by de Vries[12] and he validated them to be within 20% of the actual properties of the test specimen. As no further derivation or other theories were used these formulas are still valid.

## 5.3. IMU Damper

The drone is very depended on the measurements of the IMU, therefore they should be very precise and have a high frequency. However the rotors of the drone introduce high amplitude vibrations which cause excessive IMU drift and imprecise measurements.
To overcome this issue it was decided the IMU needs to be mechanically filtered from noise by putting a damper between the IMU and the chassis. According to literature the minimum control frequency is 200Hz. Therefore all vibrations with frequencies above this threshold should be damped.

### 5.3.1. Loading

Vibrations in the frame of a quadcopter are mostly caused by instabilities in the motors. This means that the frequency of the vibrations will be similar to the rotational velocity of the motors[56].
Although the motors are located at the end of the arms, the other parts of the structure are considered stiff enough to transport all vibrations to the mounting plate of the IMU. The noise of the motors is present in all six degrees of freedom.
Due to the complicated structure it is very hard to predict the precise magnitude of the vibrations that will reach the the position of the IMU. Since the IMU is gathering data at 800Hz, as explained in Section 7.1.1, any vibrations of up

to 400Hz can be measured. However, in order to ensure that all vibrations of over 400Hz are completely damped out, the damping threshold was set at 200Hz.

### 5.3.2. Design

To dampen the motion of the IMU it will be bolted to the frame with rubber feet. The rubbers will guide the loads through and dampen all high frequencies. The IMU will be mounted on the rubber feet using the two bolting holes located 30.48mm apart.

The rubber feet are small circular pieces of rubber with a bolt on top of it, which are commercially available. The height and diameter will be determined using the models described below.

The commercially available parts use rubber as damping element. In order to keep cost down the commercially available units will be used. Therefore the main damping material will be rubber. The rubber has a youngs modulus of 0.02GPa and damping ratio of 0.4[14].

### 5.3.3. Model

The IMU has six degrees of freedom (DoF) on the dampers, namely the x,y,z position and the three rotations, see Figure 5.11. The two supporting damped bolts are modeled as a spring and a damper collection. The equation of

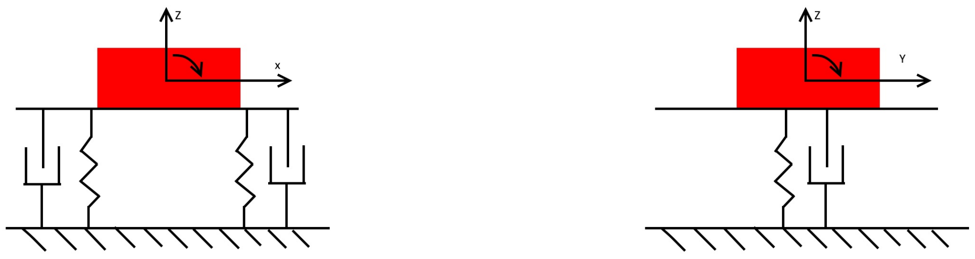


Figure 5.11: Free body diagrams of the IMU damper constellation

motion are noted down for each of the six DoF. For these equations it is assumed that there is no coupling between the rotations. Furthermore all angles are deemed small so velocities can all be calculated by multiplying the rotational velocity by the radius. Furthermore the distance (d) between the location of the damper and center of mass of the IMU is 15mm[15]. Note that the IMU is approximately a square so $I_{xx} = I_{yy}$. This results in Equation (5.23).

$$\begin{aligned} \sum F_x &= & m\ddot{x} &= & -2\dot{x}a_{xy} - 2xk_{xy} + D_x(t) \\ \sum F_y &= & m\ddot{y} &= & -2\dot{y}a_{xy} - 2zk_{xy} + D_y(t) \\ \sum F_z &= & m\ddot{z} &= & -2\dot{z}a_z - 2zk_z + D_z(t) \\ \sum M_x &= & I_{xx}\ddot{q} &= & -2\dot{q}a_q - 2qk_q + D_q(t) \\ \sum M_y &= & I_{xx}\ddot{p} &= & -2\dot{p}d^2a_z - 2pd^2k_z + D_p(t) \\ \sum M_z &= & I_{zz}\ddot{r} &= & -2\dot{r}d^2a_{xy} - 2rd^2k_{xy} + D_r(t) \end{aligned} \tag{5.23}$$

The a and k terms in these equations are the damping and spring coefficient of the rubber feet respectively. These could be calculated by assuming the feet act as clamped mass less beam supporting a mass. For this, Equation (5.24) hold true.

$$\begin{aligned} a_{xy} &= \zeta\sqrt{mk_{xy}} & k_{xy} &= \frac{3EI_{xx,damper}}{l^3} \\ a_z &= \zeta\sqrt{mk_z} & k_z &= \frac{AE}{l} \\ a_q &= \zeta\sqrt{Ixxk_q} & k_q &= \frac{EI_{xx,damper}}{l} \end{aligned} \tag{5.24}$$

In these formulas $I_{xx,damper}$ is the area moment of inertia of the damper, E is the Youngs modulus of the damper material, $\zeta$ is the damping coefficient of the material and A is the cross sectional area. All the D functions in Equation (5.23) are disturbance forces acting on the IMU.

The ordinary differential equations were then transferred to the frequency domain and transfer functions were compiled. Using these transfer functions bode plots were made for all different vibrations.

[14] `https://www.azom.com/properties.aspx?ArticleID=920` [Visited at June 13 2018]

[15] `https://www.vectornav.com/docs/default-source/documentation/vn-100-documentation/vn-100-user-manual-(um001).pdf?sfvrsn=b49fe6b9_18` [Visited at June 13 2018]

### 5.3.4. Result

Although different combination of height and radius of dampers are possible a suitable was found. The IMU will be damped by two rubber dampers with a height of 3mm and a radius of 3mm. The damping characteristics of these dampers can be found in Figure 5.12.

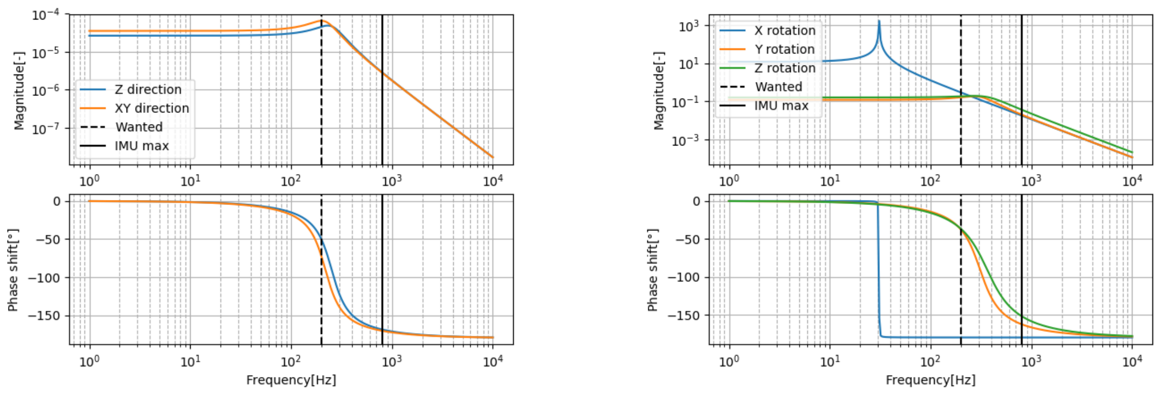


Figure 5.12: Frequency response of the damper system. The black striped vertical line points out the wanted cutoff frequency. The solid line corresponds to the maximum frequency of oscillations that the IMU can measure.

The size of the rubber parts dampens the other parts of the drone very well. The graphs at the left show the acceleration response to disturbances in the X, Y and Z direction. The X and Y direction are combined as the damper bends in the same direction and will yield a similar curve. The 3x3mm damper has a slight peak at the wanted frequency of 200Hz, but also dampens very steeply at higher frequency. This is a wanted effect and lowers all vibrations above the 200Hz.
Secondly the results for the rotational vibrations are given in the right plot. The damping is different in these cases as it mostly depends on the arms to the dampers. The green and orange curves, showing the Z and Y rotation respectively, are damped at the same frequency. Furthermore the Y and Z rotational damping frequency is slightly higher than for the accelerations.
However a problem arises in the rotation vibration around the X-axis. The only damping that can be found around that direction comes from the bending forces in the rubber. The rubber is however not stiff in bending resistance and therefore a very sharp peek arises at 30Hz. The high peak is 100 times higher than the normal response and is therefore unwanted. Furthermore directly after the peak the response is damped with a phase difference of 180°.

The problem arose from the fact that the IMU has only two mounting location that could be used for the damping bolts. To overcome the sharp peek at 30Hz two rubber pieces will be placed along the Y-axis, effectively damping the IMU at two more locations. This leads to the kinematic equation in Equation (5.25) and the response as shown in Figure 5.13.

$$\sum M_x = I_{xx}\ddot{q} = -2\dot{q}a_q - \dot{q}d^2a_z - qd^2k_z - 2qk_q + D_q(t) \tag{5.25}$$

This configuration removes the high frequency and dampens the X rotation together with the other rotations.

## 5.4. Propeller Guards

The propellers of the drone will be turning at more than 30000RPM. In the case a propeller collides with its surrounding, the drone will rapidly lose control and crash. During most of its life, the drone will either be hovering or flying slowly. During this time, a propeller guard would save a propeller from breaking. At high speeds, however, the propeller guards are allowed to fail.
An added benefit of the propeller guards is the fact that it will also protected the surround and each livings beings from the rotating blades. The propellers, although small and lightweight, have a lot of kinetic energy due to their high rotational rate, as explained in Section 17.4.1.

### 5.4.1. Design

In general four different concepts of propeller guards can be identified: quarter, half, full and side quards. All four types are shown in Figure 5.15.
The fully circular propeller quards were immediately deemed unfeasible as they would severely obstruct the view of the camera. A quick trade off based on the most important parameters was made to find the type that best suites

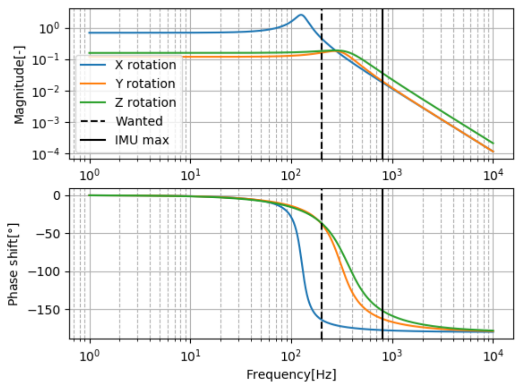


Figure 5.13: Frequency response with added rubbers

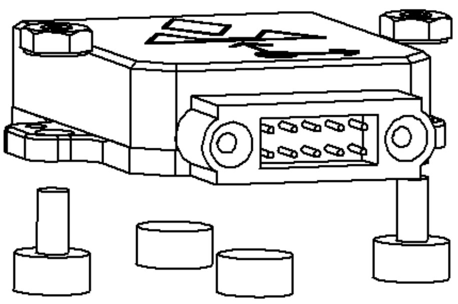
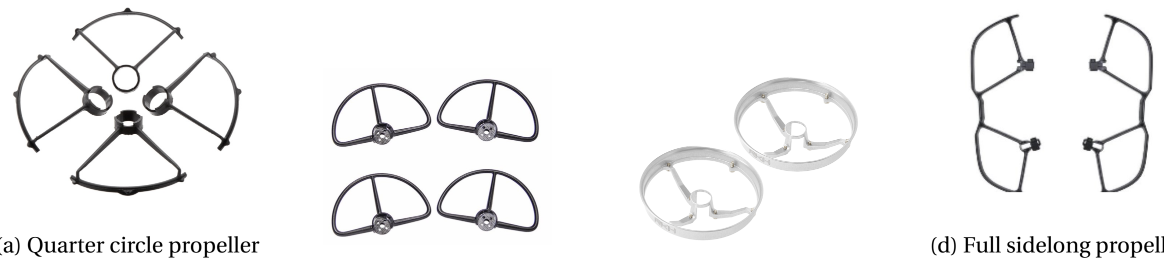


Figure 5.14: Rubber feet setup



(a) Quarter circle propeller guards (b) Half circle propeller guards (c) Full circle propeller guards (d) Full sidelong propeller guards

Figure 5.15: Propeller guard types

the needs, this can be seen in Table 5.8. All values were found using reference part and the perimeter coverage was determined as a ratio considering a square perimeter. All criteria have the same weight. The score was determined using Equation (5.26)

$$Score = \frac{Perimetercoverage}{Weight \cdot numberofbolts} \tag{5.26}$$

This way the guard typ with the highest protection for the best repairability and weight can be chosen.

Table 5.8: Propeller guard trade off

| **Criteria** | **Quarter** | **Half** | **Full side** |
|---|---|---|---|
| Weight per rotor[g] | 9.0 | 12.0 | 14.0 |
| Number of bolts per guard[#] | 4 | 4 | 8 |
| Perimeter coverage in total[%] | 45.5 | 68.2 | 72.3 |
| **Score** | 0.0126 | 0.0142 | 0.0065 |

It is clear that the full side propeller guards are penalized very heavily for having some many bolts. Having more bolts extends the time needed to replace a broken one and it also adds to the mounting complexity. The half and quarter types are at a near tie, however as the half give a lot more protection they were chosen.

The propeller guards are designed such that they can easily be 3d printed when damaged. The files necessary to do so will be delivered with the drone.

### 5.4.2. Validation

As no real calculation are done to ensure the propeller guards are able to prevent the propellers from colliding, other methods are employed to validate the working of the propeller guards.
A lot of parts are 3d printed for small home build drones with great success. This also includes propeller guards as they break often during hard crashes and are usually easy to print. They are commonly available on thingiverse, an online platform for 3d parts[16]. The same methods has also been used for other educationally build drones[52].

[16] `https://www.youtube.com/watch?v=ppAhPB4rgmk` [Visited at June 7 2018]

# 6 Propulsion and Power

A number of requirements are dedicated to the Propulsion and Power subsystem, driving the design to be built around performance and endurance. As the POS states, the aim of the project is to design not only an autonomous drone, but one that is also fast and agile. A lot of work was already done during the initial sizing of RAIDER during the Mid Term phase, and the work that was done now, during the detailed design phase, began by going over the previously conducted research. An overview of the method that was developed during the detailed design phase is given in Section 6.1, which summarizes the Mid Term and the new approach. Section 6.2 goes over the main analytical concepts behind rotor aerodynamics to provide the reader with insight in the topic. This is used to justify certain choices that were taken during the development of the Propulsion and Power sizing method, described in Section 6.3. A breakdown of the final configuration of the drone is given in Section 6.4.

## 6.1. Overview of Approach

The approach used in the Mid Term consisted of a semi-empirical method, based on the work of Staples[1] and Coco [10]. Staples developed a simple formula to estimate the thrust generated by a drone propeller using as input the propeller geometry and RPM of the motor, based on simple momentum theory formulas empirically corrected using experimental data. Coco developed a method of estimating the power required by the propeller to generate the calculated thrust in a similar manner, and both their work was combined with additional elementary equations for battery sizing to conduct an initial sizing of the Propulsion and Power subsystem.

One of the problems encountered during the development of this approach was that the team lacked the knowledge to fully grasp the assumptions and formulas used by Staples and Coco, and due to this, the first week of the detailed design phase was dedicated on researching the physics behind rotor aerodynamics. The initial idea was to develop a simple numerical approach to calculate the thrust based on Blade Element Momentum Theory, but as soon as the difficulty of writing such a script in the time frame available became apparent, the sizing method reverted back to using semi-empirical formulas, this time with the additional insight provided by the knowledge on helicopter-like propeller aerodynamics. A program was developed that estimated the thrust and power generated by over 2000 combinations of real Lithium Polymer (LiPo) batteries and propellers, and a suitable configuration was chosen and validated with the use of external programs such as ECalc[2]. ECalc, if given a certain multicopter configuration, is capable of approximating the feasibility of the design, based on statistical data provided by the components' manufacturers and external users. If the configuration was validated, the process was repeated, this time inputting more accurate estimates of the configuration until the developed script and ECalc values converged. In this way, a final design was reached, and the breakdown of the final components chosen is given at the end of this Chapter.

## 6.2. Rotor Aerodynamics

Due to the relatively low complexity and technological novelty of drones, empirical methods such as a Class I or II weight estimation methods for aircraft do not exist for quadcopters. For the same reason, sources that deal with the basics of drone aerodynamics are also hard to come by, and in general it is more effective to have a very rough estimation of the drone's parameters, build it, and optimize it through trial and error. The financial consequences of changing the hardware after prototyping a drone are many orders of magnitudes lower than for an aircraft or spacecraft. Nonetheless, understanding the physics of rotor aerodynamics, even in the broadest terms, can only help the design process. As such, the basics of helicopter rotor aerodynamics (similar to a drone's in all but size) are explained in this section. The theory reported and the insights provided are all based on the work of Seddon [48], unless stated otherwise.

[1] *https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html* [Visited at May 30 2018]

[2] *https://www.ecalc.ch/xcoptercalc.php* [Visited at June 12 2018]

### 6.2.1. Momentum Theory

The simplest depiction of the rotor and its aerodynamics is given by Momentum Theory. In this theory, the rotor is considered an actuator disc, which does not impart any rotation to the flow. This simplification makes this theory only suitable for roughly describing hover and vertical flight, as the wake generated by the rotor plays too big a role in forward flight. However, Momentum Theory can still provide some insight on the impact of certain rotor design choices.

Assuming the flow to be incompressible, the pressure at the actuator disc can be calculated using Bernoulli's equation. For hover, the velocity of the flow far enough away from the disc is zero, therefore at the inflow:

$$p_\infty = p_i + \frac{1}{2}\rho v_i^2 \tag{6.1}$$

While at the outflow, using $\Delta p$ to describe the pressure imparted by the disc to the flow:

$$p_i + \Delta p + \frac{1}{2}\rho v_i^2 = p_\infty + \frac{1}{2}\rho v_\infty^2 \tag{6.2}$$

With $p_\infty$ in $\mathrm{kg/m^3}$ and $v_\infty$ in m/s representing the pressure and velocity far enough away from the actuator disc and $p_i$ and $v_i$ the induced pressure and velocity respectively. In general, the subscript $i$ is used to indicate the properties of the flow at the actuator disc. Combining Equation (6.1) and Equation (6.2):

$$\Delta p = \frac{1}{2}\rho v_\infty^2 \tag{6.3}$$

$\Delta p$ is the thrust per unit area of the disc, which, due to conservation of momentum, can be expressed as:

$$T = \rho A v_i v_\infty \tag{6.4}$$

Where $T$, in N, is the thrust and $A$ in $\mathrm{m^2}$ the actuator disc area. It then follows from Equation (6.3) and Equation (6.4) that $v_i = 2v_\infty$. Using this fact and rearranging Equation (6.4) results in:

$$v_i = \sqrt{\frac{T}{2\rho A}} \tag{6.5}$$

The work done on the air can be expressed by $Tv_i$, therefore the power of the rotor can be written as:

$$P = Tv_i = \sqrt{\frac{T^3}{2\rho A}} \tag{6.6}$$

Where $P$ is the power. The ratio $T/A$ is called the disc loading. From these simple derivations, it can already be seen that a low disc loading is beneficial, as at the same thrust the rotor will induce a greater acceleration in the flow and use less power. More in-depth relationships between thrust, power, and other rotor parameters cannot be found with just Momentum Theory, and it is thus necessary to introduce a greater level of detail in the description of the rotor.

### 6.2.2. Blade Element Momentum Theory

Blade Element Theory (BET) applies the airfoil theory of aircraft to the rotating blades of rotorcraft. When the relationships derived from Momentum Theory are combined with BET, the overall method is described as Blade Element Momentum Theory (BEMT). The blades are assumed to be rigid, and the major complication of BEMT with respect to simple Momentum Theory is the need to integrate over the radius of the blade, as the elementary forces of each blade segment needs to be summed together.

#### Vertical Flight

Figure 6.1 shows a top view of the rotor, and a blade segment of width $\mathrm{d}y$ at radius $y$ with the relevant angles and forces. $\theta$ is the pitch of the blade measured from the horizontal plane of the rotor, which for helicopters is controlled by the pilot through the collective. It is the sum of the angle of attack between the blade chord and the perceived flow, $\alpha$, and the inflow angle $\Phi$. $\Phi$ can be described by the tangent of the velocity component perpendicular to the disc $V_c + v_i$ and the in-plane velocity component $\Omega y$. The resultant of these two velocity components is defined as $U$.

Throughout this section almost all of the equations make use of non-dimensional parameters. The value they represent and how they are made non-dimensional is described in Table 6.1. The symbols used in the table are $\rho$ for the air density, $A$ for the rotor disc area, $\Omega$ for the angular velocity of the rotor in rad/s, and $R$ for the rotor radius in m.

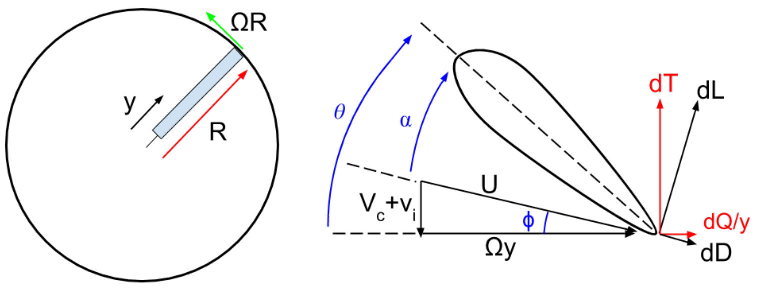


Figure 6.1: Illustration of rotor and blade element

| Symbol | Non-dimensional Symbol | Factor | Explanation |
|---|---|---|---|
| $y$ | $r$ | $R$ | Fraction of blade span |
| $V_c + v_i$ | $\lambda = r\phi$ | $\Omega R$ | Inflow factor |
| $\mathrm{d}T$ | $\mathrm{d}C_t$ | $\rho A(\Omega R)^2$ | Thrust coefficient |
| $\mathrm{d}Q$ | $\mathrm{d}C_q$ | $\rho AR(\Omega R)^2$ | Torque coefficient |

Table 6.1: Non-dimensional symbols common in BEMT equations

The lift and drag of a blade segment can be expressed using the lift and drag formulae:

$$\mathrm{d}L = C_l \frac{1}{2}\rho U^2 c\mathrm{d}y \tag{6.7}$$

$$\mathrm{d}D = C_d \frac{1}{2}\rho U^2 c\mathrm{d}y \tag{6.8}$$

Where $c$ is the chord length in m of the blade segment. The thrust $T$ and torque $Q$ of a blade segment can be expressed as a function of the lift, drag, and inflow angle. If the inflow angle is small enough, the expressions become $\mathrm{d}T \approx \mathrm{d}L$ and $\mathrm{d}Q \approx (\Phi\mathrm{d}L + \mathrm{d}D)y$. A useful expression for the rotor geometry is the solidity factor:

$$\sigma = \frac{Nc}{\pi R} \tag{6.9}$$

Where $N$ is the number of blades of the rotor. For $N$ blades, the thrust and torque expressed in non-dimensional coefficients become:

$$\mathrm{d}T = \frac{1}{2}\sigma C_l r^2 \mathrm{d}r \tag{6.10}$$

$$\mathrm{d}Q = \frac{1}{2}\sigma(\Phi C_l + C_d)r^3 \mathrm{d}r \tag{6.11}$$

The rotor power $P$ is equal to the rotor torque. A further distinction is made between the component of the torque containing the lift term, and the one containing the drag term. These are defined as the induced power and the profile drag power. In non-dimensional coefficients, and substituting $\Phi = \lambda / r$:

$$\mathrm{d}C_P = \mathrm{d}C_Q = \mathrm{d}C_{P_i} + \mathrm{d}C_{P_o} = \frac{1}{2}\sigma C_l \lambda r^2 \mathrm{d}r + \frac{1}{2}\sigma C_d r^3 \mathrm{d}r \tag{6.12}$$

Noting that the induced power can be expressed as $\lambda \mathrm{d}C_T$, assuming uniform flow over the blade and constant profile drag, after integration the power becomes:

$$\mathrm{C}_P = \lambda C_T + \frac{1}{8}\sigma C_{d_0} \tag{6.13}$$

From Equation (6.13) it can be seen that a low solidity factor is beneficial, in that it results in a lower profile drag and therefore less power required. This is achieved by a larger rotor radius, and a lower number of blades. However, while it may reduce the power required, a lower solidity factor reduces the thrust produced (Equation (6.10)), which would have to be compensated by a higher blade pitch, which impacts the stall behaviour of the rotor and the profile drag. Therefore a balance must be found to ensure an efficient design, which cannot be found by analytical equations alone.

#### Forward Flight

The equations for forward flight are derived in much the same manner as for vertical flight, but there are complications introduced by the advancing rotor, which now also has an angle of attack with respect to the flight path. As a result, the velocity is made of three different components; the first two are the tangential and radial components to the rotor plane, which are described by similar relationships as those used for the two velocity components in vertical flight. The third velocity component is a function of the flow normal to the blade, the inflow factor due to the angle between the rotor and the flight path vector, and an additional velocity caused by the flapping motion of the blade. This flapping motion is caused by the higher perceived flow velocity as the rotor rotates into the flow, due to the forward speed.

The additional dependencies of the induced velocity at the rotor adds a degree of complexity that made it prohibitive to build a program that would be more accurate than simpler formulas in the time frame available. Terms such as blade flapping could not be ignored, as they have a non-negligible impact even for the much shorter blades of small quadcopters [21]. Moreover, even if simple approximations of all these terms were found (which would regardless make the equations barely more accurate than semi-empirical relationships, if at all), many of these parameters depend on each other, and a numerical approach would require many iterative loops.

Additional complications stem from the application of helicopter equations to drones. For example, many of the derived formulas for helicopter aerodynamics include non-dimensional parameters that are made non-dimensional, by dividing by the angular velocity of the rotor. For helicopters, the RPM during its operation remains relatively constant, the thrust being generated by a change in the collective setting. For quadcopters, however, a change in thrust is achieved by changing the RPM, making it more difficult to obtain useful results without re-deriving the equations. Also, this specific project requires the drone to reach very high speeds, which can only be achieved by a large inclination of the drone with respect to the incoming flow, making the small angle approximation invalid and greatly complicating the equations that need to be solved.

## 6.3. Sizing Method

In this section, the sizing method developed is discussed. The approach is an amalgamation of formulas developed by Staples[3], Coco [10], concepts covered during courses of the Aerospace Bachelor of the TU Delft, and insights gained while researching rotor aerodynamics. Section 6.3.1 describes the equations used to calculate the required motor specifications, and Section 6.3.2 discusses the equations utilized for the flight time estimation. Section 6.3.3 explains the overall logic and flow of the approach that was used to size the drone, and Section 6.3.4 discusses the tests conducted to verify and validate the method.

### 6.3.1. Sizing of the Motor

In order to obtain the thrust generated by a rotor, it is necessary to know the distribution of the airflow over the rotor blades. This is not an easy undertaking, and there have been a number of models derived from analytical derivation and experimental work that attempts to accurately describe the distribution of the induced velocity, such as the Drees inflow model or the Mangler and Squire model [27]. However, the more accurate the model, the more complicated it is and the harder it is to implement. Both the simpler and intricate models require a number of iterations before reaching an acceptable result, as they require knowledge of parameters that the inflow will then be used to estimate. Even the Momentum Theory description of the inflow requires thrust to be known or initially guessed, and the model does not even take into account the wake generated by the rotor, rotor geometry, or the non-uniformity of the flow. Empirical correction factors can be added to account for phenomena such as the non-uniformity of the flow and tip losses, but they cannot get around the iterative nature of the process.

The approach utilized in this report to estimate the thrust of the thrust of a drone propeller is based on the equation derived by Staples[4]. The equation is based on momentum theory, discussed in Section 6.2.1, and assumes that the induced velocity of the air is equal to the pitch speed of the propeller, which in turn is solely dependant on the RPM and the pitch of the rotor. The implied assumptions of this model is that the induced velocity profile is constant along the blade span and that it is not dependent on the number of blades of a propeller or any other geometrical parameter but the pitch. As discussed in Section 6.2, this is a very poor assumption. However, Staples added an empirical correction based on thrust data of drone propellers of many different sizes, provided by Matthew McCrink of the Ohio State University. The empirical correction is shown in Equation (6.14), where $d$ and $r_{pitch}$ are the diameter and pitch of the propeller respectively, both in meters.

[3] *https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html* [Visited at May 30 2018]

[4] *https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html* [Visited at May 30 2018]

$$\text{Correction Factor} = \left(\frac{1}{3.295} \cdot \frac{d}{r_{pitch}}\right)^{1.5} \tag{6.14}$$

The formula produces results which consistently estimate a value that is between 15-30% lower than the test data. This is still not an exceptionally accurate estimate, but as it is a very simple relationship requiring only a single iteration, implementing it allowed for a more efficient allocation of the team's resources over the rest of the subsystems. Moreover, the equation is based on geometrical propeller parameters that are extremely easy to obtain, as opposed to chord and twist distribution, which are almost never provided by drone propeller manufacturers. The thrust equation is given in full in Equation (6.15).

$$T = \rho \cdot \left(\frac{\pi \cdot d^2}{4}\right) \cdot \left(\frac{RPM}{60} \cdot r_{pitch}\right) \cdot \left(\frac{RPM}{60} \cdot r_{pitch} - V_0\right) \cdot \left(3.295 \cdot \frac{d}{r_{pitch}}\right)^{1.5} \tag{6.15}$$

As was introduced in Section 6.1, the sizing approach developed needs real battery and propeller parameters to calculate which combinations produce a feasible configuration. One of the outputs of the program was the specifications of the motor that were needed, and to generate these specifications Equation (6.15) was rearranged to find the maximum RPM the motor would need to deliver as a function of the thrust. The resulting quadratic equation is shown in Equation (6.16).

$$RPM = 60 \cdot \rho r_{pitch} \frac{\pi d^2}{4} \cdot \frac{V_0 \pm \sqrt{V_0^2 - 4\left(\frac{3.29546 r_{pitch}}{d}\right)^{1.5} \frac{4T}{\rho \pi d^2}}}{2T\left(\frac{3.29546 r_{pitch}}{d}\right)^{1.5}} \tag{6.16}$$

This approach required the thrust to be known beforehand. To estimate the critical thrust required, two critical cases were considered, namely the maximum forward speed requirement and the turning speed at 3m turn radius requirement. The free body diagrams of these two cases are shown in Figure 6.2.

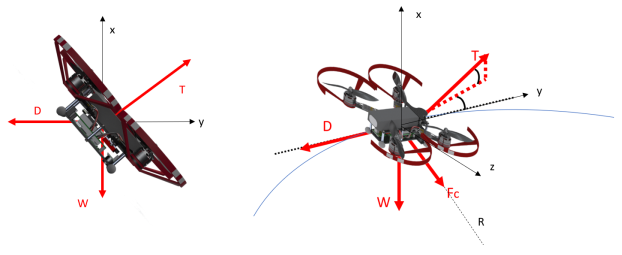


Figure 6.2: Free body diagrams of the critical thrust scenarios

Only the profile drag of the drone was considered, as other drag types such as induced drag were assumed to have a significantly lower impact. The reasoning behind this choice was that the drone, during the critical thrust scenarios, would have to be pitched to extremely high angles, causing it to behave like a flat plate with its surface perpendicular to the incoming flow. Hoerner [20] collected the $C_{d_0}$ of flat plates and many other shapes from many different experiments, and the drone's $C_{d_0}$ was taken to be the same as a flat plate with the shape of a cross perpendicular to the flow, which Hoerner states to be 1.2.

Considering the drone to behave like a flat plate is a very rough assumption, and very little data, other than the derived formulas in Section 6.2, was available to formally justify neglecting other drag types, or at least estimate the uncertainty caused by this decision. After the sensitivity analysis (discussed in Chapter 13) was conducted, however, it was found that the outcome of the program was very insensitive to this parameter, and therefore no further resources were spent attempting to improve the uncertainty in the $C_{d_0}$ value.

The remaining forces shown in Figure 6.2 could be obtained from the drone parameters and requirements. The only other approximation was for the mass of the motors, as all other components were known a priori from the sizing of the structure, and, having already been chosen, from the information given by the manufacturer. For each configuration, since a real battery was considered, its correct mass value could also be taken into account. The magnitude of the critical required thrust was then obtained by finding the resultant of these forces, and the RPM from Equation (6.16).

A second semi-empirical equation was used to calculate the power needed by the motor, as a function of thrust and RPM. This equation followed from the work of Staples and Coco [10], who followed a similar procedure as explained previously to correct the equation with experimental data. The power equation is given in Equation (6.17).

$$P_{in} = \rho \cdot \left(\frac{\pi}{4}\right) \cdot \left(\left(\frac{RPM}{60}\right)^3 \cdot r_{pitch}^{1.5} \cdot d^{3.5}\right) \cdot \left(\frac{1}{3.29546}\right)^{1.5} \cdot \left(0.79122 \cdot \frac{d}{r_{pitch}}\right)^{0.5} \tag{6.17}$$

Equation (6.17) was validated in a previous report [4] using data obtained from drone motor manufacturers, and showed that the difference between the predicted values and experiments was 16%. The propeller geometry and the calculated RPM needed for the critical case was inputted in Equation (6.17) to obtain $P_{\text{crit}}$.

The final steps in the process of obtaining the motor specifications involved using elementary equations learned during the Aerospace BSc of TU Delft regarding current and voltages. First, the KV of the motor was calculated. The KV is an indication of the RPM generated per volt, but is a value calculated by manufacturers often for unloaded motors, and therefore a 25% safety factor was used in calculations. This equation is given in Equation (6.18).

$$KV = 1.25 \cdot \frac{RPM_{\text{max}}}{V_{\text{bat}}} \tag{6.18}$$

Finally, the critical motor current was calculated with Equation (6.19). This equation was used both to check that the power demanded by the motor did not exceed the motor's physical capabilities, and to size the ESC.

$$A_{\text{crit}} = \frac{P_{in}}{V_{\text{bat}}} \tag{6.19}$$

### 6.3.2. Flight Time Estimation

Section 6.3.1 discusses the different equations that were used to find a suitable motor given a certain configuration. To size the motor, however, the critical case must be considered, as the motor must be capable of delivering peak performance when required. To find a suitable configuration that can meet the 10 minute flying requirement, sizing with the values of the critical case leads to a significantly overdesigned solution. It was thus necessary to find an approach that would approximate the average power needed during nominal operations. This was achieved by scaling the power required in the critical scenarios by a correction factor that depended on the drone velocity. Intuitively, it was known that the power depended on $V^3$, and thus a third degree polynomial was derived.

$$P(V^3) = c_1 V^3 + c_2 V^2 + c_3 V + c_4 \tag{6.20}$$

To solve for the coefficients, the following four conditions were assumed:

- $f(V_{\text{crit}}) = P_{\text{crit}}$
- $f(0) = P_{\text{hover}}$
- $f'(0) = 0$
- $f''(0) = 0$

Where $V_{\text{crit}}$ and $P_{\text{crit}}$ are the average velocity and power at the critical condition, and $P_{\text{hover}}$ the power required when hovering, which was obtained with the same approach as $P_{\text{crit}}$, but by initially considering that the thrust required would only be equal to the weight. The remaining two conditions were assumed to prevent the polynomial to correct the power required to values that were below $P_{\text{hover}}$. The value of $V_{\text{crit}}$ and $P_{\text{crit}}$ were calculated by taking the average of the velocity and power for the two critical cases of maximum forward speed and maximum turn speed. From inspecting the IROS 2018 track layout shown in Section 3.2, approximately half of the track is composed of straight stretches and the other half in turns, and thus a simple average was taken. While it is likely that the distribution of power will not be split in such a manner, this percentage of maximum forward and turn speed was tested in the sensitivity analysis and showed no impact on the final outcome of the method (Chapter 13). The coefficients of the correction polynomial were thus found, and the function is given in Equation (6.21).

$$P_{\text{out}_{\text{nom}}} = (P_{\text{crit}} - P_{\text{hover}}) \cdot (\frac{V_{\text{nom}}}{V_{\text{crit}}})^3 + P_{\text{hover}} \tag{6.21}$$

Where $P_{\text{out}_{\text{nom}}}$ [W] is the average power the motor requires during nominal conditions, and $V_{\text{nom}}$ [m/s] the average speed of the drone during nominal conditions. This speed was found to be 1 m/s, as discussed in Section 9.5.1. The average power that the battery needs to deliver to the motor is found by dividing $P_{\text{out}_{\text{nom}}}$ by the motor efficiency, taken to be 80%.

$$P_{\text{in}_{\text{nom}}} = \frac{P_{\text{out}_{\text{nom}}}}{\eta_{\text{mot}}} \tag{6.22}$$

The total average power the batter needs to deliver during nominal operations was computed by adding the power of all the other electrical components, $P_{\text{other}}$, to $P_{\text{in}_{\text{nom}}}$. As the computer boards and other electrical components had already been chosen [4], this value could be calculated exactly and was computed to be 16 W. To test whether a change in this value could cause a significant change in the model outcome, this parameter was tested in the sensitivity analysis in Chapter 13, but the model turned out to not be sensitive to changes in this parameter. By dividing $P_{\text{in}_{\text{nom}}}$ by the battery voltage, $I_{\text{actual}}$, the actual current the batter needs to deliver was computed. This value was used in Peukert's formula (Equation (6.23)) to compute the actual battery capacity, taking the Peukert constant for Lithium Polymer (LiPo) batteries as 1.05 [54].

$$C_{\text{actual}} = C_{\text{bat}} \cdot \left( \frac{C_{\text{bat}}}{I_{\text{actual}}\, t_D} \right)^k \tag{6.23}$$

In Equation (6.23), $C_{\text{actual}}$ [Ah] is the actual capacity of the battery, $C_{\text{bat}}$ [Ah] the capacity of the battery as stated by the manufacturer, $I_{\text{actual}}$ [A] the current the battery needs to deliver during nominal operations, $t_D$ [hrs] the nominal discharge time, calculated by finding the reciprocal of the battery C-rating as given by the manufacturer, and $k$ the Peukert constant. Finally, the flying time under nominal operating conditions was calculated as shown in Equation (6.24), $t_{D_{\text{nom}}}$ [mins] is the flying time.

$$t_{D_{\text{nom}}} = \frac{C_{\text{actual}}}{I_{\text{actual}}} \cdot 60 \tag{6.24}$$

### 6.3.3. Sizing Approach Logic

The equations discussed in Section 6.3.1 and Section 6.3.2 were applied to over 2000 combinations of batteries and propeller considered, and the program's logic is illustrated in Figure 6.3. The batteries considered were obtained from HobbyKing[5], with capacities ranging from 1000 mAh to 3000 mAh with 3, 4, 5, and 6 cells. The propeller geometry considered ranged from 4 to 5 inches, with many varying pitches up to 0.5 inches less than the diameter. The propeller geometries were generated within the program with steps of 0.5 inches for both diameter and pitch. Different thresholds were placed throughout the program, checking that the calculated values did not exceed physically reasonable values, or that the flying time was above 10 minutes. If either of these cases were found to be true, the configuration was regarded as unfeasible and was discarded.

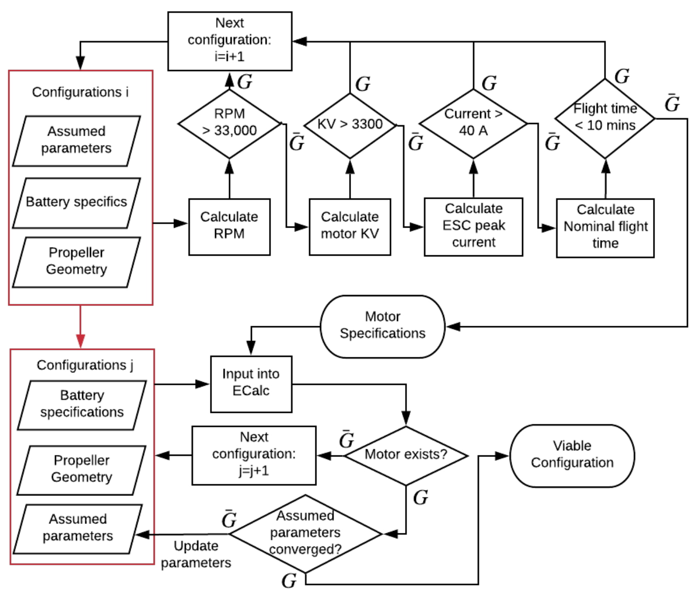


Figure 6.3: Propulsion and power process flowchart

The output of the program was a list of potentially viable configurations that specified the total mass of the drone, battery, motor and ESC specifications, the propeller geometry, and various performance parameters. These configurations (j in Figure 6.3) were a subset of all the configurations analyzed at the beginning. The program was not capable of knowing if a motor with the calculated specifications existed, and therefore the viable configurations were

[5] *https://hobbyking.com/nl_nl/?___store=nl_nl* [Visited at June 8 2018]

inputted into ECalc[6], an online tool that lets users check the expected performance of a certain drone configuration. ECalc offers the possibility to choose real motor, batteries, ESCs, and other components, calculating the performance of the inputted configuration and warning users if the configuration is expected to be feasible or not. The problem of this tool is that all the components need to be more or less known beforehand, while the written program would be able to output suitable motor specification from just the battery and propeller geometry. ECalc was thus used to test the output of the program, and, after choosing a suitable motor, the values estimated by ECalc were similar to those outputted by the written script, the program was rerun using more accurate inputs until the script's and ECalc's values converged, as it was then possible to more accurately estimate the drone's total mass after choosing the motors and knowing the diameter of the propeller required. This process is shown in Figure 6.3.

The sizing method discussed in this section was conducted as explained, and it was found that the 4 inch propellers that had been used for the final concept was not used in any of the viable configurations. In fact, to be able to meet all the stakeholder requirements for the power and propulsion subsystem, only 5 inch propellers could be used. 12 configurations were found to be viable, and were all given to ECalc, to check if a motor existed with the needed specifications and to ensure that the configurations were indeed feasible. This process further reduced the number of viable configurations, and once the remaining configurations were improved through the iterative process, the final components were chosen based on the configuration with the lowest mass. The design was optimized for lowest mass, as lighter drones experience lower impact energy in case of a crash and are in general safer, especially for high performance drones such as this one. The final output of the program, in other words the final specifications of the hardware, is reported in Table 6.2.

| **General** | | | **Motor and Battery** | | |
|---|---|---|---|---|---|
| **Parameter** | **Value** | **Unit** | **Parameter** | **Value** | **Unit** |
| Total mass | 711 | g | Max RPM | 31470 | - |
| T/W | 5 | - | KV | 1668 | - |
| ESC max current | 29.7 | A | Max power | 549 | - |
| Nominal $t_D$ | 15.6 | mins | Battery capacity | 1550 | mAh |
| Full thrust $t_D$ | 0.7 | mins | Cells | 6 | - |
| Propeller diameter | 5 | inch | C-Rate | 65 | C |
| Propeller pitch | 4.5 | inch | | | |

Table 6.2: Output of sizing approach

### 6.3.4. Verification and Validation

The implementation of the different equations in the program was, in general, verified by considering a specific combination of battery and propeller geometry, and performing the calculations of the program by hand, or by visually inspecting the output of the program.

In terms of validation, the majority of the formulas are basic equations, validated extensively by various industries. Equation (6.16) and Equation (6.17), however, were not as straightforward. As stated previously, the thrust equations had already been validated by Staples[7], and demonstrated an underestimation of the thrust by approximately 15%. The equation is conservative, which is advantageous for this type of sizing. The power equation was validated by comparing the results with the experimental data of five different FPV motors, each tested with two different propeller geometries. The results of these comparison yielded an RMS error of 12.2%, and a total error of 16.6%. For a more detailed look at the validation process conducted for these two equations, the interested reader is referred to [4].

As seen in Figure 6.3, a step in the sizing approach involved inputting the results of the program into ECalc, and the output of ECalc were used to define more accurate inputs in the program, until the values converged. The validation of the program was thus integrated within the sizing approach itself, and was present at each step of every iteration of the configuration. ECalc claims to produce values that are within 15% of real results. These values are hard to validate, as there are no sources that provide a drone configuration to that level of detail together with experimental data measuring flight time, maximum speed, and other values. However, ECalc has been used in different academic projects [6] [3], and is, in general, regarded as one of the best online tools for the sizing of radio controlled projects. Table 6.3 shows the performance estimated by the written program and the predictions made be ECalc.

[6]`https://www.ecalc.ch/xcoptercalc.php` [Visited at June 10 2018]
[7]`https://www.electricrcaircraftguy.com` [Visited at May 30 2018]

| Parameter | Script | ECalc |
|---|---|---|
| Max Speed [km/h] | 188 | 162 |
| T/W | 5.0 | 5.3 |
| Max RPM | 31470 | 30307 |
| Max motor current [A] | 29.7 | 20.0 |
| Nominal $t_D$ [mins] | 15.7 | 14.3 |
| Full thrust $t_D$ [mins] | 0.7 | 1.0 |

Table 6.3: Sizing approach and ECalc outputs

Table 6.3 shows the performance estimated by the written program and the predictions made be ECalc. The top speed was calculated by first using Equation (6.15), using as inputs the manufacturer's motor RPM, diameter and pitch of the final propeller, and an initial forward speed of zero. The static case shown in Figure 6.2 was then solved iteratively, taking the drag to be initially zero, until the drag grew higher than the thrust. The script outputs a maximum speed 16% higher than stated by ECalc, just outside ECalc's accuracy. This difference is rather pronounced, could be caused by the fact that at these speeds just considering the profile drag is no longer sufficient, or because speed effects on the RPM of the motor was not taken into account. Regardless, even taking $-15\%$ of the value outputted by ECalc, the maximum forward speed is 137.7 km/h, still significantly higher than the requirement of 100 km/h. This follows from the fact that the configuration was sized for the critical case, the cornering requirement, and has thus higher capabilities in forward flight than required. Until tests can be conducted, RAIDER's top speed is taken to 160 km/h, the more conservative estimate.

For the other results, both outputs are very close to each other. In general, almost all the values are within 15% of the values calculated by ECalc, except for the maximum motor current and the flight time at full thrust. The large difference in the motor current likely suggests a mistake in the approach. The motor can, however, withstand currents of up to 39.9 A, and while in the future work can be conducted to obtain a better current estimate, for the final configuration the maximum motor current should not pose any problems. The difference in flight time at full thrust is also higher than 15%, but as this flight time was eventually derived from the maximum motor current, a lower full thrust flight time is expected, as the error in the equations carried forward. One thing to note for the other outputs is that the nominal flight time calculated by ECalc is the hover time, while the script's is at a speed of 1 m/s; while not hover, the drone is very close to being stationary, and the two values were deemed close enough for comparison.

Overall, many approximations were made, but the model was capable of producing the same accuracy as a program such as ECalc, with the added benefit of being able to run without knowing the configuration a priori. A number of estimates were found to likely be significantly off the real value, an uncertainty that cannot unfortunately be quantified until real tests are conducted. A more accurate approach, perhaps a numerical BEMT approach, could have resulted in values closer to reality, but increasing the model complexity may have result additional parameters, augmenting the potential sources of errors [28], and would have taken a significantly higher amount of time to run.

## 6.4. Components Breakdown and Remarks

The sizing approach discussed in Section 6.3 would output the specifications of motor and ESC, and select a specific battery and propeller geometry, but many of these components still had to be chosen from the multitude of products offered by multicopter component manufacturers. Due to the configuration sporting a 6S battery, a contraint was placed on many of these components, as not every motor and ESC is designed to function at the high voltage provided by a battery with this many cells. The battery that allows this configuration to function is the Turnigy Nanotech 1550 mAh 6S with a 65C continuous maximum C-rating, and burst C-rating of 130C. The rest of the components was chosen from components for FPV drones, as these components are optimized to be lightweight and at the same time be capable of handling the high currents needed to reach the desired performance.

Due to the discrete nature of motors and other items, a motor with the exact specifications needed did not exist, and a motor with the closest performance was picked instead, always rounding up. This had the added benefit of including a small margin in the final design and slightly overshoot the performance requirements. The chosen motor is the HobbyWing XRotor Race PRO 2207 1750 KV, specifically designed for the high performance needed by FPV drones, capable of being able to support a 6S battery, a maximum power of 950 W, and a maximum RPM of over 31000. The ESC is the Turnigy Multistar 30A Rev16 v3. This is ESC is significantly overdesigned, as it can handle 30 A of continuous current, and the configuration's peak is slightly below 30 A. This specific ESC was chosen as it was unknown if the duration of the peak current would be longer than what other ESCs could handle, and for an increase of only a few grams, a significant safety margin could be added. Operating at significantly lower currents also has no negative effect on the ESC, as it prevents the temperature of the component from increasing to levels that would decrease its efficiency [19]. Finally, the propeller chosen was the MasterAirscrew RS 5045 5x4.5. This propeller is made of carbon fiber, as it is lightweight and was found to be a better material than plastic at reducing the safety risk, as discussed in the reliability, availability, maintainability and safety (RAMS) analysis in Chapter 17. A summary of the final power and propulsion components is given in Table 6.4, with the associated electrical block diagram shown in Figure 6.4, illustrating the connection between the different hardware components.

| **Component** | **Product** |
|---|---|
| Motor | HobbyWing XRotor Race PRO 2207 - 1750KV |
| ESC | Turnigy Multistar 30A Rev16 v3ESC |
| Propeller | Master Airscrew 5045 |
| Battery | Turnigy 1550mAh 6s 65-130C |

Table 6.4: List of final power and propulsion components

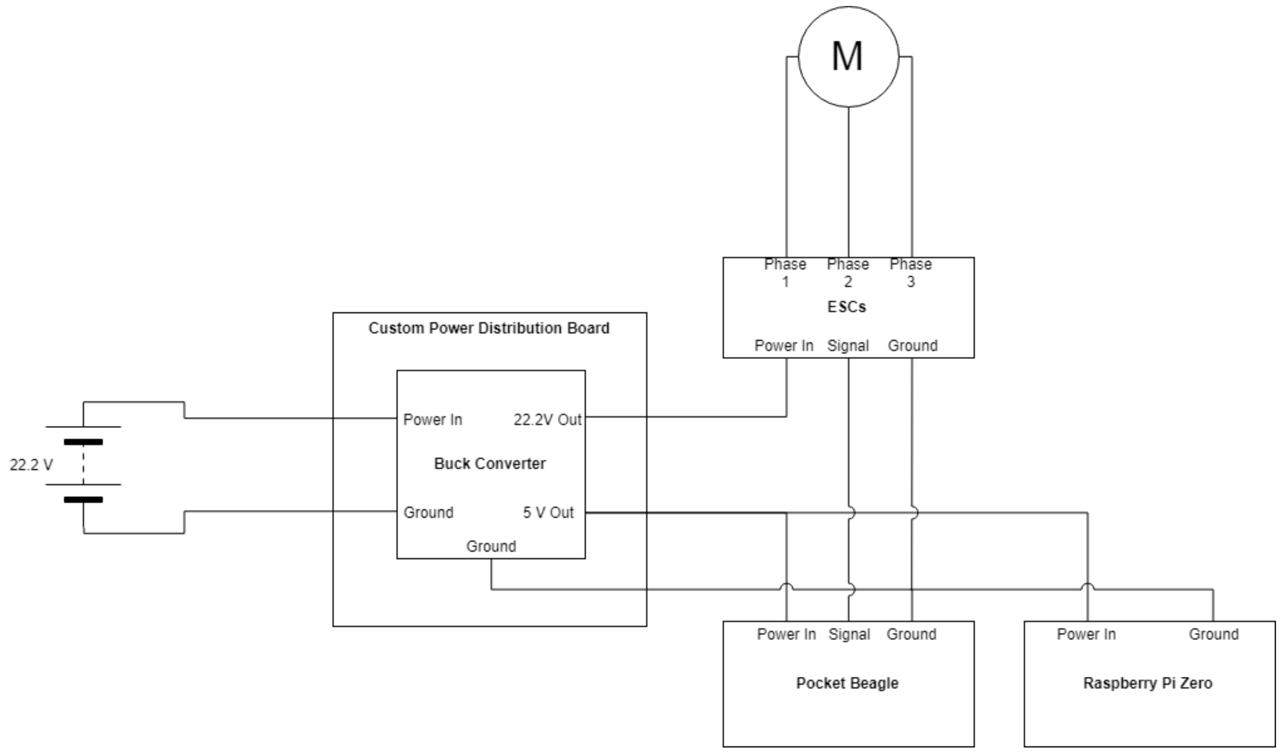


Figure 6.4: Electric block diagram of the final design

While the current configuration should already be sufficient to meet all the requirements, most of the FPV drones of this size sport 3-blade propeller. 3-blade propellers were not considered during the sizing of the propulsion and power subsystem, as the equations used to estimate the thrust and the power had been empirically corrected for drones carrying 2 blade propellers. As was seen in Section 6.2, thrust and power are related to the number of blades and geometry of propellers with a non-linear relationship, therefore using Equation (6.15) and Equation (6.17) to estimate the thrust and power generated by propellers with more than 2 blades, or extrapolating new empirical correction factors, would have introduced larger error than the equations already had. Large sets of experimental thrust data collected with drone motors with propellers with any number of blades is unfortunately not easily available. Thus, for a future improvement of the sizing approach, it is suggested to carry out experiments to be able to both improve the reliability of the current thrust and power equations used, as well find an accurate equation that allows for 3-blade and 4-blade propellers to also be considered. For RAIDER, using a 3-blade propeller should allow the diameter to be reduced, as it more blades increases the raw thrust produced, which might be beneficial as the size reduction would allow the drone to fly with a less frequent positive gate detection.

# 7 Electronics and Sensor Fusion

In this chapter, the electronic components that were not completely worked out in Chapter 4 are further expanded upon. After this, in Section 7.2 through Section 7.2.2, a sensor fusion algorithm will be chosen. Finally, a short conclusion with a diagram showing the detailed workings of the subsystem will be provided in Section 7.3.

## 7.1. Electronics

The goal of the electronic subsystem is to provide accurate measurements to the software for localization and housekeeping.

### 7.1.1. IMU Sensor

An Inertial Measurement Unit (IMU) combines 3-axis accelerometers, 3-axis gyroscopes and 3-axis magnetometers into one package. The IMU chosen by the team is the VectorNav VN-100[1]. This allows the drone to calculate its pose (location and orientation in the world), linear and angular velocities and its linear accelerations. However, an IMU has the disadvantage that the pose is not measured directly, rather, it's calculated through the integration of the accelerations. Due to this integration, drift is introduced into the pose calculation, quickly making IMU pose measurements useless. A solution is to use a different localization method to eliminate this drift.

The rate at which this drift increases is quadratic for the position and linear for the linear velocity and angular orientation. This makes the position uncertainty the driving case in terms of drift. This position drift depends on a few parameters of each IMU: the accelerometer bias, the accelerometer misalignment, and the Gyroscope Angle Random Walk (Gyroscope ARW). This can be seen in Equation (7.1) [2], where *Drift* is the position drift, $E_{Bias}$ is the error due to accelerometer bias, $E_{AM}$ the error due to accelerometer misalignment, and $E_{ARW}$ the error due to gyroscope angle random walk.

$$Drift = \frac{1}{2} g_0 T^2 \left(E_{Bias} + E_{AM} + E_{ARW}\right) \tag{7.1}$$

The different errors in Equation (7.1) are expanded upon in Equations (7.2) to (7.4). These equations were taken from VectorNav's website. Since RAIDER will be using one of their IMUs, it is assumed that they have verified and validated these equations.

$$E_{Bias} = b_a \tag{7.2}$$

$$E_{AM} = sin(\theta) \tag{7.3}$$

$$E_{ARW} = sin\left(ARW\sqrt{\frac{T}{3600}}\right) \tag{7.4}$$

Provided that the IMU is properly calibrated before flight, the parameter $b_a$ in Equation (7.2) is the In-Run Bias Stability provided by the manufacturer. $\theta$ used in Equation (7.3) is the maximum accelerometer misalignment, and the $ARW$ in Equation (7.4) is the gyroscope angle random walk. This can be approximated by the value of the gyroscope's van Allen Deviation after 1 second[3]. For the VectorNav VN-100, these values can be found in Table 7.1.

Table 7.1: Drift influencing values for the VectorNav VN-100

| Parameter | Value | Unit |
|---|---|---|
| $b_a$ | 0.04 | $mg$ |
| $\theta$ | 0.05 | $Deg$ |
| $ARW$ | 0.175 | $\frac{Deg}{\sqrt{Hour}}$ |

[1] `https://www.vectornav.com/products/vn-100` [Visited at 12 June 2018]
[2] `https://www.vectornav.com/support/library/imu-and-ins#unaided-position-estimate` [Visited at June 5 2018]
[3] `https://ez.analog.com/docs/DOC-2163` [Visited at 5 June 2018]

The graphs that were used to calculate the ARW can be found on the manufacturer's website[4].

Since the gates of IROS2018 will be 1.4 x 1.4 meters and RAIDER will be approximately 28 centimeters wide, the drone is allowed to deviate almost 55 centimeters from the center line. In order to account for an initial position that is not on the center line and other disturbances, RAIDER is allowed to deviate by 45 centimeters from its initial path. For the VectorNav VN-100 to drift so much, it takes approximately 9.3 seconds. This means that RAIDER has to pass the gate in less than 9.3 seconds after the last gate detection and successful location correction. Ideally, within these 9.3 seconds, RAIDER will have already detected the next gate and localized itself. This way it can be assured the virtual world stays closely aligned to the real world.

### 7.1.2. Temperature Sensors

In order to prevent the overheating of the computer boards, temperature sensors are placed on them. These sensors are normally integrated into the boards themselves. However, since RAIDER carries 2 computers, the data of the Raspberry pi's sensors will be send to the Pocket Beagle for centralized temperature monitoring. The same will happen with the temperature sensors aboard the custom Power Distribution Board (PDB) that will connect the 2 computers.

### 7.1.3. Voltage Sensor

The output voltage of the battery is measured in order to assess its state of charge. When the state of charge gets too low, the drone will have to perform an autonomous landing in order to ensure the safety of the surroundings and drone. This voltage will be measured on the custom PDB and will be monitored by the system monitoring segment aboard the Pocket Beagle (as can be seen in fig. 11.2).

## 7.2. Sensor Fusion

All sensors are inherently affected by (random) noise and bias, making their measurements inaccurate [24]. The degree to which this happens can be reduced by comparing measurements from several sensors and minimizing the mean squared error. This is exactly the principle on which all Kalman filters work.

They also allow the combining of low frequency, accurate position measurements with high-frequency, inaccurate position measurements. Which is crucial for the functioning of autonomous drones. Aboard RAIDER, the IMU will be used to provide high frequency (800 Hz) pose information. However, as stated in Section 7.1.1, IMU's are subject to drift. In order to limit this drift, the drone's pose and velocities have to be corrected for every few seconds. This will be done using Visual Odometry (VO). The drone will use its cameras to find and track the gates and its own relative position from them. Combining this with the knowledge of the position of each gate results in a full localization of the drone. Although this will be relatively low frequency (3Hz-30 Hz), it allows for the IMU to be 'reset', eliminating the accumulated drift. The effect of this can be seen in Figure 7.1 [38], where every 120 seconds (at every green circle), the IMU drift is reduced due to GPS measurements localizing the drone.

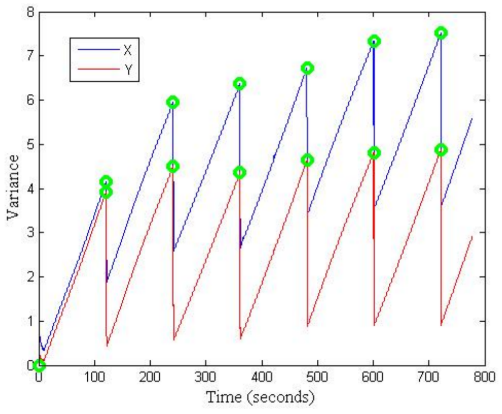


Figure 7.1: IMU Drift Reduction due to Low Frequency GPS Measurements [38]

### 7.2.1. Software Packages

In order to implement sensor fusion in the provided time, preexisting software packages have to be used. Three such packages were identified: ETH Zurich ASL Sensor Fusion[5], Robot Pose EKF[6], and Robot Localization[7].

ETH Zurich's Autonomous Systems Lab has developed its own sensor fusion package[60]. It is already optimized for Micro Aerial Vehicles (MAVs) and automatically calibrates itself. The latter allows the drone to quickly re-calibrate itself after a crash, lowering the expected downtime in between runs. With 31 dimension this sensor fusion package uses the largest state vector. It tracks not just the pose, but also all velocities, linear accelerations, and sensor biases

[4] `https://www.vectornav.com/support/faq/acceleromter-gyro-questions` [Visited at 5 June 2018]
[5] `http://wiki.ros.org/ethzasl_sensor_fusion` [Visited at 4 June 2018]
[6] `http://wiki.ros.org/robot_pose_ekf` [Visited at 4 June 2018]
[7] `http://wiki.ros.org/robot_localization` [Visited at 4 June 2018]

among other things. There are two versions of this algorithms, one allows for the IMU with 1 other sensor. The other allows for any number of other sensors to be used [60].

Robot Pose EKF is the lightest of the sensor fusion algorithms proposed here. It only tracks the drone's position and orientation. It, however, cannot estimate the IMU's orientation. Another filter is needed to implement this [42].

The Robot Localization package was designed by Thomas Moore and Daniel Stouch from the Sensor Processing and Networking Division of Charles River Analytics, Massachusetts[38]. It includes both an Extended and an Unscented Kalman Filter (EKF and UKF respectively). Although it tracks more state variables than Robot Pose EKF, it does not track as many as ETH Zurich's algorithm. Robot Localization has a 15 dimensional state vector, consisting of the drone's pose, linear and rotational velocities, and linear accelerations [8]. It was designed to be as non-restrictive as possible, allowing for any combination of sensors to be used with it[38].

### 7.2.2. Sensor Fusion Comparison

Since RAIDER needs accurate positioning, Robot Pose EKF is quickly eliminated. Although it is the least computational intensive solution, the accuracy suffers, which is unacceptable for this application. This leaves 3 options: Robot Localization's EKF or UKF, or ETH Zurich's EKF.

The main difference between an EKF and an UKF is that the EKF linearizes the system around its estimation and assumes Additive Gaussian White Noise. UKF works differently, making it much more robust in highly nonlinear situations. Because the drone is to fly at extreme angles, the drone will most likely often be in the highly nonlinear region.

The amount of floating point operations (FLOPS) that are needed for each iteration of the filter varies per application. The result is that a very exact estimate of the computational cost cannot be given at this stage. A rough estimate, however, can be made based on the findings of [39] and [55]. The average value of the amount of FLOPS that was calculated by the two methods is used. The values for $N$, $M$, and $P$ are 15 or 31, 9, and 4 respectively. The final 2 were chosen to reflect that linear accelerations, and linear and angular velocities are measured and that only the speed of each of the 4 rotors can be controlled. The first 2 reflect the size of the state vector of the Robot Localization and ETH Zurich algorithms respectively. One should note that these values for the Extended Kalman Filters are only valid if the Jacobians of both the F and H matrices are analytically calculated beforehand. The results can be seen in Table 7.2.

Table 7.2: Computational Cost of each Sensor Fusion Package

| | **FLOPS [-]** | **Percentage of Pocket Beagle Computational Power [%]** |
|---|---:|---:|
| ETH Zurich EKF | 195 000 | 40.7 |
| Robot Localization EKF | 43 000 | 9.0 |
| Robot Localization UKF | 61 000 | 12.7 |

The Percentage of Pocket Beagle Computational Power is calculated by assuming the filter runs at 800 Hz, the frequency at which the IMU outputs its data. The total amount of floating-point operations per second were compared to the computational power of the Pocket Beagle that was calculated in Section 11.3. In order to fairly compare the three, a small trade-off was done. The result of this can be seen in Table 7.3. The trade-off criteria are:

- **Computational Cost:** Since RAIDER is being designed to be as light as possible, there is not a lot of mass for computer boards. This makes the lighter algorithms better. Since RAIDER's design philosophy is based on it being a light weight drone, this has gotten a weight of 2.
- **Robustness to Non Linearity:** The drone will be flying at large attitudes, meaning that it will often be in the very non linear part of its model. This means that linearization does not result in a good approximation. Accurate localization is important for the drone and since it will often fly at high attitudes, it is important to have a sensor fusion method that is still accurate, even in the non linear part of the drone's flight envelope. Therefore the weight of this criterion is 2.
- **Internal Model Quality:** During the interval where there is no accurate localization coming from the VO, the internal model has to predict the current state of the drone. The closer this model is to reality, the better this prediction is going to be. Although it is an advantage to have a model that was designed specifically for MAVs, the difference in accuracy will not be very large. Thus resulting in a weight of 1.

[8] `http://docs.ros.org/melodic/api/robot_localization/html/` [Visited at 4 June 2018]

- **State Vector Size:** A bigger state vector results in more accurate prediction. It does come at the cost of computational speed, however. This increase accuracy is clearly important, it is not as big as the increase due to the use of an algorithm that does not linearize when the drone is in the very non linear domain. Therefore, the weight of this criterion is 1.

Table 7.3: Trade-off of Sensor Fusion Packages

| | | **ETH Zurich EKF** | **Robot Localization EKF** | **Robot Localization UKF** |
|---|---|---|---|---|
| **Computational Cost** | 2 | 195000 FLOPS Red:-2 | 43000 FLOPS Green:+1 | 61000 FLOPS Yellow:+0 |
| **Robustness to Non Linearity** | 2 | Low since a EKF locally linearizes system. Orange:-1 | Low since a EKF locally linearizes system. Orange:-1 | High since a UKF does not linearize. Green:+1 |
| **Internal Model Quality** | 1 | High since it was specifically made for MAVs. Green:+1 | Low, since it is very universal. Orange:-1 | Low since it is very universal.1 Orange:-1 |
| **State Vector Size** | 1 | 31 dimensional Blue:+2 | 15 dimensional Yellow:+0 | 15 dimensional Yellow:+0 |
| **Results** | | **-3** | **-1** | **1** |

Each of the packages was scored on a scale of -2 to 2 (represented by red, orange, yellow, green and blue from low to high) on each of the trade-off criteria. Then, to conclude, these scores were multiplied by the weight of each criterion and summed. This shows that Robot Localization's Unscented Kalman Filter is the best option for RAIDER.

## 7.3. Sensor Fusion Conclusion

The detailed working of the Sensor Fusion subsystem is shown in Figure 7.2. It clearly shows how the sensor fusion subsystem not only does the sensor fusion itself, but it also calculates the absolute position from the data it is given by both the gate detection and the path planning subsystems.

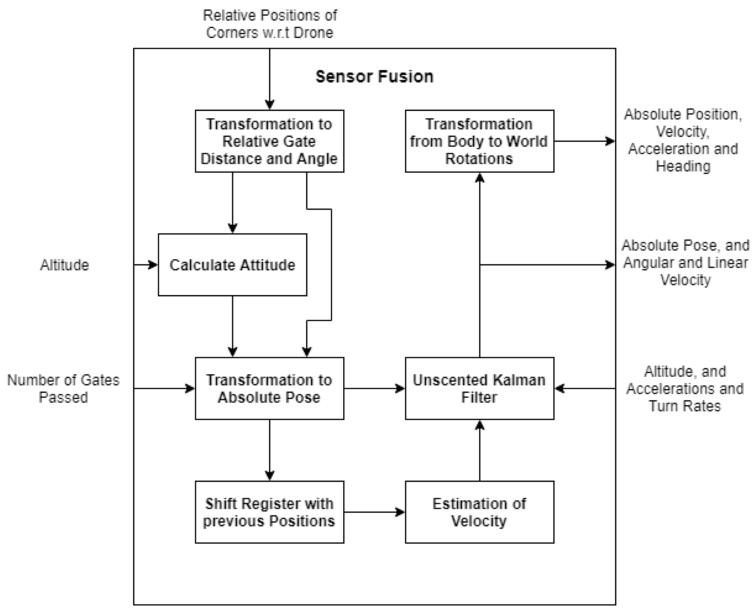

Relative Positions of Corners w.r.t Drone
Sensor Fusion
Transformation to Relative Gate Distance and Angle
Transformation from Body to World Rotations
Absolute Position, Velocity, Acceleration and Heading
Altitude
Calculate Attitude
Absolute Pose, and Angular and Linear Velocity
Number of Gates Passed
Transformation to Absolute Pose
Unscented Kalman Filter
Altitude, and Accelerations and Turn Rates
Shift Register with previous Positions
Estimation of Velocity

Figure 7.2: Flowchart for the Sensor Fusion Subsystem

## 7.4. Verification and Validation

The models need to be verified in order for the results to be of any value.

### 7.4.1. IMU Drift Model

The model that was used in Section 7.1.1 was taken from the IMU's manufacturer's website. It is therefore assumed to have been validated. The calculation for solving the equation was validated by a colleague.

### 7.4.2. Software Package Computational Cost

As a unit test for verification, the excel sheet that was used to calculate everything in Table 7.2 was checked for errors by a colleague. Since the formulae used are from literature, they are assumed to have been validated.

# 8 Gate Detection

Detecting and recognizing objects is one of the most important uses of vision systems in intelligent agents. The human brain can distinguish between more than 30 000 visual categories, and can detect objects in the span of a few hundred milliseconds. Although technology is nowhere near human performance in this task, there has been a considerable amount of breakthroughs in the field during the past few years. The work presented here is only a small part of what research has achieved nowadays, but is an essential element of the drone's software.

## 8.1. Approach Overview

This chapter elaborates on the computer vision algorithms that have been considered, their development, their accuracy and throughput time that lead to the final decision for an optimal algorithm to be implemented in the drone's software. The three main algorithms that have been considered for further development range from classic color based algorithm to the state-of-art computer vision techniques. As previously mentioned, the strong suit of the design is the compromise of selecting the algorithms and the hardware together to meet a specific set of requirements. Thus, accuracy of detection is not the only parameter to take into account, but also the processing speed.

The development of the gate-detection subsystem has been divided into four different stages. The first part concerns the image pre-processing where all of the synthetic images that have been generated for testing, verification and validation are labelled with the corner position and some distortions are applied in order to fill the gap with reality. Secondly, a general study on the algorithms to use and adapt has been performed followed by the choice of three main algorithms to consider in more detail. These algorithms have very different working principles: one is color-based, one mainly relies on template matching, which is similar to cross-correlation, and one is a small convolutional neural network. After this, a benchmark analysis has been applied to designate the algorithm to integrate in the on-board computer. Finally, the gate localization, also called gate dewarping, was implemented using the corner position estimated from the gate images.

## 8.2. Image Pre-processing

In order to challenge the participants of the ADR and to simulate a real-life environment, new gates have been designed with new dimensions and features. These features are crucial for the gate detection algorithm logic. The fact that the gates have been changed from a bright thick orange to a thinner gate border with a white contour, reduces the accuracy of any algorithms based on color significantly. Moreover, since the orange present in the gate is not really visible from medium to long range, the white contour is the main feature to be observed which is a color commonly seen in the background making the sampling for gate detection to be totally independent of the specific color of the gate.

Besides the challenging track and the new gate features, developing an algorithm of this type without any data is unfeasible. Due to the change in gate, all of the images provided from previous years became unrepresentative. For this reason, a data set of synthetic images has been created, using the information provided by the organizers about the gate dimensions and color. In these images, the background and gate position have been changed.. The first query that this approach gives rise to, is the reliability of artificial images[1]. In real life condition images are sensitive to motion, light and other characteristics that might not give a perfect visualization on the gate such as the CAD renders. To fill the quality gap with reality all the background used have been taken from real images and only the gate itself has been positioned in this environments.

Images have been processed implementing different distortions in order to make the simulation as realistic as possible. The quality distortions that have been implemented in the images are noise, blur, compression (JPEG), contrast, erosion, dilation, closing, opening and color gradient modification[2]. These quality distortions have been introduced

[1] *https://machinelearning.apple.com/2017/07/07/GAN.html* [Visited at 26 June 2018]

[2] *https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html* [Visited at 5 June 2016]

in different levels within low thresholds, even visible for the human eye, without combining them. At the same time the available data has been significantly augmented by introducing all of these variables.

The images fed to the algorithms have been compressed to 512x288 pixels keeping the same aspect ratio of the original image (16:9) in order to lower the computational cost. This also implies a lower depth visibility range to avoid having too many details in the background that will lead to an overall better performance.

## 8.3. Gate Detection Algorithms

The IROS race environment is a pre-designed track. Consequently the a-priori knowledge about the geometry of the gate and their distribution can be used as the basis for a higher-order gate feature detector that will be developed in this section. In domains such as this one, the track predominantly consists of rectilinear structures. With rectangles making up much of the key geometry to be recognized, it is prudent to use this as the basis of a feature detector in order to be able to track the gate while it is in sight. The main feature to be identified is not the gate center but the corners, which is the main purpose of the algorithms developed in this section. The corner position will be further passed to the gate dewarping algorithm which is in charge for determining the gate inclination and localizing the gate with respect to the drone position by using standard mathematical transformations.

The approaches discussed in this section include the algorithms considered during the literature study and the models that have been developed and benchmarked to choose the best software solution.

### 8.3.1. Considered Algorithms

#### Feature Based Matching

Feature based tracking element algorithms are able to recognize specific key points of objects in the image and identify a probable match in another image containing a similar object. The initial idea was to implement this algorithm in order to state whether there was a gate or not by tracking the number of key-points encountered in a specific frame. Furthermore, if accuracy of matching keypoints on both images is high, the homography matrix could be found, thus the exact location of the gate on image. To remove possible keypoints outliers, that were matched incorrectly, RANSAC [15] and least median of squares [30]. The main algorithms that have been considered for this approach are mentioned here below.

**SIFT** (Scale-Invariant Feature Transform), is a scale invariant method, which might find corners at different scales on the same image. SIFT is a four steps process that extracts distinctive features from the images [29]. The first step is to find potential points of interest at different scale levels with the difference-of-Gaussian function. Those keypoints are then processed further to select only valid ones, by passing them through the stability test. Afterwards for each selected keypoint their orientation is found. And lastly keypoint descriptors are found by measuring local image gradients, so that those keypoints can be identified on a rotated and scaled image.
**SURF** (Speeded Up Robust Features) is a robust feature detector that works on a similar principle to SIFT. The main difference is that it is faster. Both algorithms output keypoints of the input image that are handled in the same manner.
**ORB** (Oriented FAST and rotated BRIEF) uses FAST (Features from Accelerated Segment Test) to find keypoints in the image and then uses a Harris corner detector to select the best match. The point descriptor, direction of intensity and orientation information, is tackled using Rotated BRIEF [46].

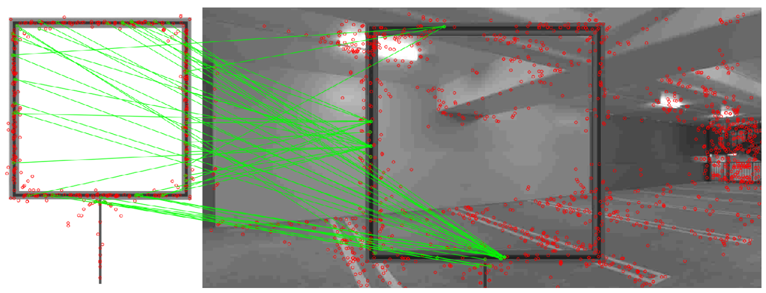


Figure 8.1: Feature based tracking algorithm (SIFT) example. The green line and points indicate the feature that are present in both images while the red circles are features or key points of the individual images which are not being matched.

Unfortunately, the IROS gates have a very simple design. This complicates the task of identifying specific patterns. The gate is made out of thin carbon fibre tubes and it has a homogeneous structure, therefore there are very few unique characteristics. The few key-points there were identified were matched to the incorrect sides of the gate as the four bars are exactly the same on all sides and sometimes these key-points were identified in the background leading to incorrect gate detections as can be seen in Figure 8.1.

### Snake Gate

Snake-gate detection is a simple classifier that consists of a cascade of filters[47], this algorithm relies on color gradient recognition and it is tuned in order to identify the shape of the gate based on color and dimensions. This algorithm starts analyzing an image by detecting a continuous line of points. When this is found, it looks for the next edge of the gate in the opposite orientation, horizontally or vertically depending on how the first one was located. If the recognized door edges meet the minimum length requirement, a sequence of filters is applied to the gate in order to return the number of identified corners and edges. This is used as an input for another filter which returns the shape of the gate in order to retrieve the relative position of the drone with respect to the gate.

The search for the gate is performed by implementing the filter on random sample images and the search is efficient because before diving into computational expensive filters it applies the ones which are easier to verify. For instance the color gradient of the first pixel is a rather fast scan, and the neighboring pixels are scanned based again on the color and also the shape, taking into account a possible side view, for instance when the drone is not directly facing the gate.

Although this algorithm presents a relatively computationally inexpensive alternative for gate recognition, it has some drawbacks. First of all the snake algorithm heavily relies on the color contrast between the gate borders and background. Moreover it is very specific which makes it applicable only for determined shapes with the given aspect ratio and color, instead of looking at contrast in a more general sense. This is however not an aspect that concerns the team's autonomous strategy as the main goal is winning the 2018 IROS competition.

## 8.3.2. Dense Lines Harris Corner Detection

Although corners are easily recognizable for humans, it is very difficult for computers to translate this feature into a point. These regions in the image are characterized by large variation in intensity in all the directions. The principle of this algorithms consist on checking the intensity gradient of the image color for a displacement in the vertical and horizontal direction in order to detect a corner. When the eigenvalues of the Jacobian matrix of the shifted intensity function are large enough such that these values exceed a certain normalized threshold, the kernel or window that is being analyzed is considered a corner. The shifted intensity function is shown in Equation (8.1), where $I_x$ and $I_y$ are the intensity gradients in the horizontal and vertical direction respectively that are calculated using a Taylor series expansion with a second order truncation error, $w(x, y)$ is the window matrix that moves along the pixels of the image during sampling and outputs weights for the pixels in a certain region between one and zero. The mathematical implementation first calculates the Sobel operator which outputs the gradient of the color intensity at each pixel. After this the corner detection function will calculate the probability of having a corner in the window given the fact that the intensity difference needs to be maximized for this case [8].

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \cdot \sum_{(x,y)} w(x,y) \cdot \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} \tag{8.1}$$

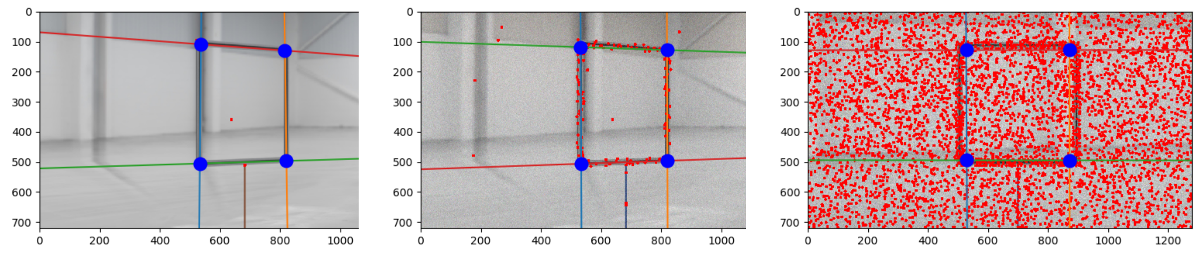
Moreover, the algorithm input is reduced to gray-scale image such that the intensity gradient calculation process is computationally minimized as there is only one single information (intensity) channel instead of three.

Using this already existing algorithm it was possible to make some fine adjustment to the parameters such as the size of the kernel, the size of the tested area, in order to get the best estimate of a region with a lot of points, and some image blurring. The high sensitivity of the parameters returned a series of points that were localized mainly around the edges of the gate. It is possible to notice the density of points throughout the images changes in function of the image distortion that was applied. Normally really noisy images returned a remarkably higher number of points than compressed and blurred images.

Furthermore, two different algorithms were applied to detect the gate. The first one is the *gradient method*, which considers points at a certain radius from each other and connects them, while the ones that do not meet the radius requirement are considered outliers and removed. This algorithm detected a corner as soon as the gradient between the previous and the next connection is above 60 degrees. The fact that the point concentration varies constantly due to the distortion applied to the figures, made this algorithm highly dependent on the number of points that was constantly varying and the parameters to be changed were being over-fit for a certain type of distortion.

The algorithm which ended up being more successful in dealing with this sequence of points is the *Dense line approach*. In this case the points detected are interpolated by making a vertical linear regression on points lying within a certain horizontal range (20 pixels). Out of all the lines found, only the two most dense were considered. The following step consist on drawing the horizontal lines but this time there was a constraint in the space to analyze thanks to the vertical lines. This allows the algorithm to find the two horizontal lines with highest number of points within a previously determined vertical range (50 pixels). This range is larger than the previous one because of perspective, the horizontal distance to a gate will normally be much larger than the vertical distance.

The final step is to find the corner positions. This can be done by solving a simple algebraic equation to calculate the four intersections between the lines. A direct visualization of this process is in Figure 8.2



(a) Blurred Image (b) Noisy Image (c) Very Noisy Image

Figure 8.2: Gate corners detection using the Dense lines Harris corner algorithm. The parameters have not been changed from one test to another in order to develop a software as robust as possible.

The main drawback of this algorithm is that it can find a gate when there is none present, as it can happen that a high concentration of points is processed somewhere else and the lines drawn will intersect in order to detect the four corner with a shape of a rectangle. In order to avoid this a relatively weak but effective condition has been implemented: the aspect ratio is checked before determining if the shape detected is an actual gate. This is done by estimating the gate rotation and dividing the average length of the horizontal bars with the one of the vertical sides of the gate. Apart from the aspect ratio, the point distribution is considered as most of the points have to be concentrated in the edges to avoid recognizing a gate just from random points in a background which may form a squared shape.

As expected, this method is very sensitive to changes in the environment, specially light conditions and high-contrast objects. The test images have been processed with different distortions but the fact that the background is relatively smooth and hence does not interfere with the gate contour makes the algorithm work almost perfectly. However, robustness is one of the main aspects to take into account for the computer vision algorithms. Therefore, this algorithm is not recommended for normal RGB images but it can be very accurate if the background is not visible which, for instance, is the case the depth images the ToF camera provides.

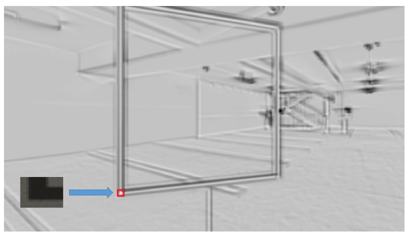
### 8.3.3. Template Matching

Template matching is a powerful method for finding the location of a template image (patch) on a source image. This approach could be used to identify the location of the corners of a gate. The main idea behind this technique is to find a best-fit location of the template image (T) on the source image (I) by sliding a patch around. At each new patch location on the source image, the "distance" is calculated, which represents how well patch fits at the current spot. The result matrix (R) stores the calculated distance, and later is analyzed for the location with the maximum match. There are multiple methods of calculating metrics, however, the CCORR NORMED Equation (8.2) was selected, as it performs best in the benchmark.

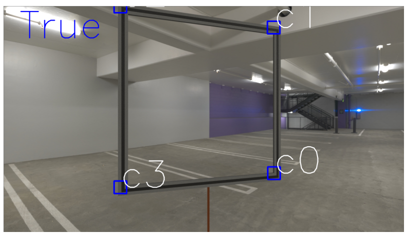
$$R(x,y) = \frac{\sum_{x'y'}\left(T(x',y')\cdot I(x+x',y+y')\right)}{\sqrt{\sum_{x'y'} T(x',y')^2 \cdot \sum_{x'y'} I(x+x',y+y')^2}} \tag{8.2}$$

In Figure 8.3b this process is clearly shown. From the clean gate synthetic image, the corner patch was cut (20x20 px). The next step would be to slide patch on entire image domain and calculate metric for each coordinate. Lastly, the global maximum in the result matrix is found, and thus the coordinates of the corner as well.

Unfortunately, with the simplicity of this method, a lot of drawbacks arise. The most obvious one is that false corners could be detected in the background. In order to distinguish true corners from false detections, some conditional



(a) Example of R matrix for bottom left corner.



(b) Example of template matching method detecting all four corners.

functions are used. Those function use insights of the gate geometry, such as an order of the corners and the relative position between them.

The main limitation of this method is to find gate corners when the camera is at angles above 20 degrees. Then gate warps drastically, so each corner is unique, and does not match any template. To solve this problem either deformed template of corners should be taken before the race, or current nominal patch could be transformed by Affine Transform[3] to generate corners at different perspective angles, an example is shown in 8.4. However, practice shows that the total accuracy increase would be small, compared to the increase in computational time. Another problem is that template must be scaled, in order to find both gates that are close by and far away. This could be resolved by re-sizing patch, which, again, adds a lot of computational time, especially if the gate is not present in the image, since all possible scaled patches have to be checked, before stating that the gate is not present in the source image.

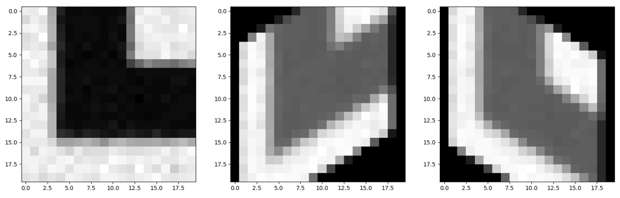


Figure 8.4: Example of corner transformation, from left to right 0/30/-30 degrees.

## 8.3.4. Deconvolutional Neural Network (DeCNN)

Image analysis is one of the most prominent fields in deep learning since, at the moment, the best object detectors rely on convolutional neural networks. Even though, the cognitive process of the human brain a convolutional neural network is different, there are some similarities in the layered structure and in the fact that this kind of system captures specific characteristics of the environment that it is being taught about[13]. The approach that is proposed here is the most accurate but also the most computationally expensive. The main idea is to remove the background, outputting the image with the gate in black. After the gate image with no background is produced, the task of detecting the corner can be done with a lightweight algorithm called *Ramer-Douglas-Peucker* to detect the contour of the gate (RDP-Contour), also known as the iterative end-point fit algorithm. It takes a curve composed of line segments and finds a similar curve with fewer points to fit it [62]. In this case only the contour approximation with four points is selected. The implementation of this algorithm works with more than 90% accuracy in the target images, which should be similar to the output of the neural network.

The input of the neural network is the RGB image, while the output is only black and white. Figure 8.6 shows what the system should receive as input and what image it would produce to be processed further.

The training was performed with 11,450 synthetic images of the gate at different distances, positions, rotations in all the axis, and with different backgrounds. This variety of images allowed for a robust detector that performed efficiently in any situation. This will solve most of the problems encountered with other algorithms that are sensitive to changes in the environment like the lighting conditions [59]. Each input image has a similar twin where the background has been removed in order to "teach" the network the desired output and minimize the error such that the image coming out from the last deconvolutional layer is exactly the same as the target one. Therefore, the total number of images used during the training increases to 22,900.

[3] `https://www.mathworks.com/discovery/affine-transformation.html` [Visited at 24 June 2018]

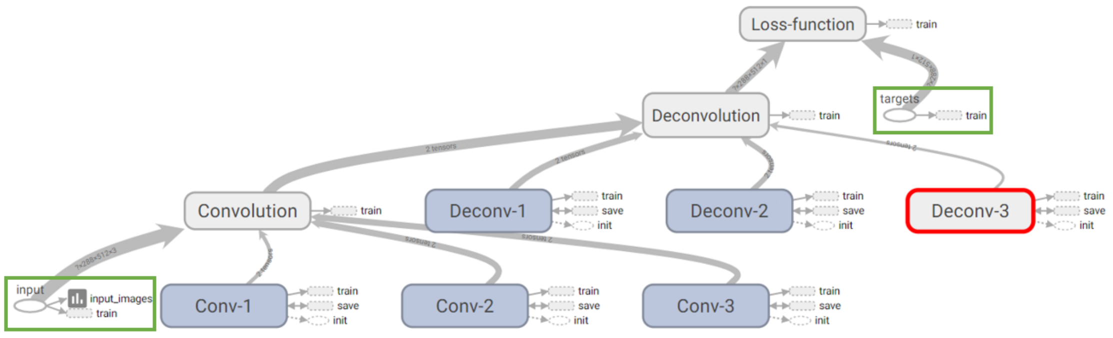


Figure 8.5: Main structure of the Convolutional-Deconvolutional Neural Network. Apart from the main layers it is possible to visualize the input image placeholder at the beginning of each training session in the first convolutional layer and the target images placeholder at the loss function to minimize the error of the output.

The DeConvolutional Neural Network (DeCNN) structure was be visualized in Figure 8.5, there are two placeholders in order to indicate the data provided to the system. The first one is the input image fed to the neural network. The second one is introduced in the structure after the last deconvolutional layer, which is a black and white image. At this point the loss function tries to minimize the error which is the difference in the output and the target image given in the second placeholder. The "Deconv-3" layer has been marked in red in order to indicate that once the model is ready to be used the output used for further image processing comes from this layer.

The structure of the DeCNN is minimized in order to save computational power, there are only three convolutional layers with multiple filters, three pooling layer which progressively reduce the image size as shown in Figure 8.6, and three deconvolutional layers, also known as transpose convolution. It is important to take into account that in this DeCNN there are no fully connected neurons as the image does not need to be classified but just compressed and re-enlarged to produce a background removal. This configuration is optimal because the system needs to detect a gate

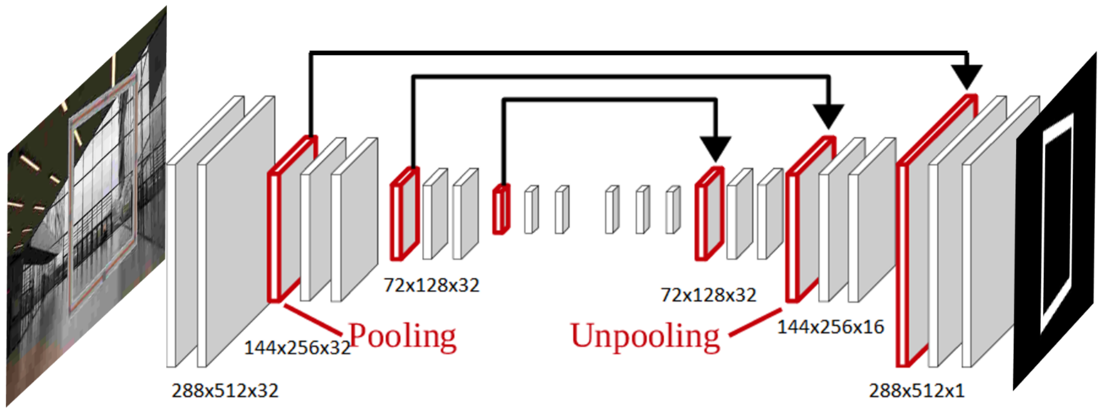


Figure 8.6: Convolutional-Deconvolutional Neural Network pipeline. The image size gets reduced during the first half of the pipeline and after the information has been processed the neural network enlarges the output until it obtains another image based on the back error propagation and loss function. The size of the tensors that comes out of each layer is indicated after each set of (de)convolutional-(un)pooling layers

which has very simple features. The choice of three convolutional layers comes from the fact that each one extract basic features from the image, and based on research the features extracted after the third layer become very abstract [64]. For instance the first layer could detect lines, the second one borders and the last one corners and contours. The choice of three pooling and deconvolution layers is made simply because of symmetry as the output image must have the same size of the input one. Moreover, the number of elements included in the structure is a key factor for the software development as the limited computational resources represent a major restriction, especially in these kinds of algorithms were the number of layers escalates very quickly to increase the accuracy of the classifier. In fact the number of operations to be performed increases very quickly with the number of layers. The convolutional layers need to "swipe" over all of the pixels of the image, it is not only about the number of layers but the specific parameters of the filters used, like the kernel size.

While training, one of the most important parameters to look at in order to evaluate whether the neural networks is actually working is the loss function. The loss, which is the difference between the target image and the output image, decreases exponentially with the number of iterations. However, this is not a clear indicator of how well the neural network would perform in real life because the parameters might be overfit for the images used during training. In simple terms the neural network would be memorizing instead of learning. In order to avoid overfitting, the number of images needs to be as big as possible and the list of images needs to be shuffled to have different cases in the batches fed to the model at each iteration.

The result of the model is visible in Figure 8.7. The image is not as perfect as the targets because there is some noise and the system is relatively small. This can be easily filtered using erosion, dilation, blur and binary threshold in order to highlight the gate which is clearly evident to the human eye. After the post-processing of the image the RDP contour approximation detection is executed to output the same as the other algorithms.

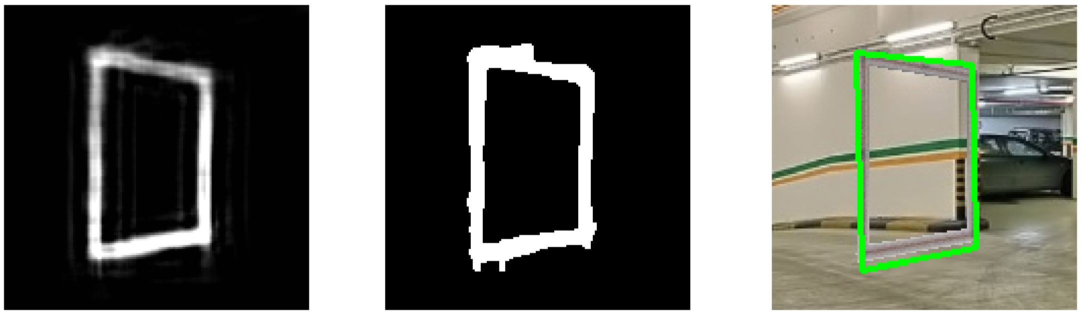

(a) Output image retrieved from third Deconvolutional layer (b) DeCNN output image after thresholding (c) RDP-Contour result plotted on DeCNN input image

Figure 8.7: DeCNN-RPD gate detection

## 8.4. Results: Benchmark

The software integration concerning the computer vision department is finalized in this section. Out of the three proposed algorithms, one has been selected. Each of the algorithms was tested and tuned with different images. However, the images for this analysis were not used before in order to simulate the performance of the algorithm in a real-life condition. For all the images the validation process records whether the algorithm has found the gate or not. In this way it is possible to report a performance estimation as shown in Table 8.1 in order to have some statistical data about the binary predictor (gate or no-gate) of each method to quantify and benchmark them. If the gate is detected it calculates accuracy based on the Mean Squared Error (MSE) of the position of each corner in order to verify that the detected corners are not randomly allocated in the image. Moreover, the accuracy of each frame is considered to check whether the points are correctly identified for the True Positive Rate (TPR) and False Positive Rate (FPR) estimation.

The throughput time has been measured on an Intel i5-4210 H processor equipped with 2 cores running at a base frequency of 2.9 GHz. Even if this measurement do not reflect the real testing conditions, it was deemed to be enough for relative comparison, further information about the compatibility between the software and the chosen hardware (Raspberry Pi Zero W) will be elaborated after the algorithm selection in the design system integration chapter.

Then Receiver Operating Characteristic (ROC) space is used to have a reference frame for the performance of a binary classifier, which, in this case, refers to determining if there is a gate or not. The space is defined by the TPR which indicates the *probability of correct detection*, and the FPR or the *probability of a false alarm*, also called a *Type I error* in statistical analysis. The benchmark evaluation outputs a value for these two probabilities at each iteration where a different batch of images will be tested for the three tested algorithms. The ratio between this two probabilities should be higher than 50% detection reliability, which refers to perform better than a random guess generator. The image batch categories used for the benchmark are displayed in Table 8.1, this classification has been chosen based on the video feed recorded by the drone during the optimal trajectory planning. The first category has images where the drone is approaching the gate, from 5 to 3 m distance. The following batch contains images where the drone rotates around the gate up to an angle of 60 degrees from a distance of 3 meters. In the third image set, the drone flies away from the gate, from 3 to 5 meters. On the fourth benchmark iteration, the images used consider a rotation up to 60 degrees of the drone with respect to the gate from a 5 meters radius. Finally the last set of images contains random backgrounds, to test the robustness of the algorithm when no gates are on sight. The benchmark parameters used to quantify the algorithms are the True-Positive ratio (TP), the True-Positive ratio with random backgrounds that have been added to the gate renders(TP-B) in order to demonstrate the robustness of the algorithm. The other parameters are the mean squared error of the corner position (MSE) measured in pixels squared, and the throughput time measured in milliseconds.

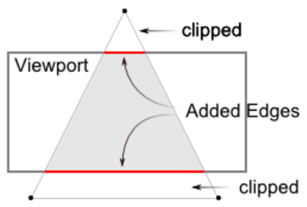
Table 8.1: Algorithms Benchmarking

| **Image Batch [Batch size]** | **Benchmark Parameters** | **Name of method** | | |
|---|---|---|---|---|
| | | **Dense Lines-Harris** | **Template Matching** | **DeCNN** |
| Approaching Gate (5m to 3m) [15] | TP [%] | 0 | 0 | 66.7 |
| | TP-B[%] | 0 | 0 | 53.3 |
| | MSE [$px^2$] | [-] | [-] | 133.3 |
| | Time [ms] | 12.2 | 35.4 | 167.1 |
| Rotating w.r.t. Gate ($\theta$=60, R=3m) [15] | TP [%] | 26.7 | 26.7 | 26.7 |
| | TP-B[%] | 0 | 13.4 | 26.7 |
| | MSE [$px^2$] | 18,884 | 5.5 | 33.6 |
| | Time [ms] | 16.1 | 35.4 | 123.5 |
| Fly away from Gate (3m to 5m) [15] | TP [%] | 6.7 | 60.0 | 86.7 |
| | TP-B[%] | 6.7 | 0 | 100 |
| | MSE [$px^2$] | 51 | 105.8 | 137.7 |
| | Time [ms] | 15.3 | 37.4 | 119.9 |
| Rotating w.r.t. Gate ($\theta$=60, R=5m) [15] | TP [%] | 60.0 | 46.7 | 100 |
| | TP-B[%] | 0 | 0 | 86.7 |
| | MSE [$px^2$] | 688 | 213 | 186.6 |
| | Time [ms] | 13.5 | 37.6 | 119.2 |
| Background images (No Gate) [60] | TN [%] | 3 | 100 | 100 |
| | Time [ms] | 25.2 | 41.3 | 119.7 |

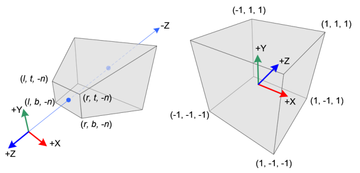
From the analysis conducted above, it is possible to deduce that the algorithms that outperforms the other two is the DeCNN as it is the most robust to changes in background and it is not highly affected by the gate position with respect to the drone. In fact, the machine learning approach stands out because it is able to recognize and learn specific features of the gate, rather than looking for the best point combination or match in the figure. This aspect is extremely important specially because the gate will not be positioned in front of a white background and the gate-detector needs to be able to identify the gate at different distances and angles. Even in some cases such as in the first batch image test, when the drone is approaching the next waypoint and not all the corners are visible. Therefore, the final computer vision software for the RAIDER concept has been deemed to be the nine layer DeCNN-RDP-Contour algorithm, with an average gate accuracy of 78.9%; this is a weighted average based on the accuracies retrieved from Table 8.1 and the batch size of each benchmarked case.

## 8.5. Gate Dewarping

Gate dewarping is an essential technique, which allows RAIDER to estimate the relative position of the gate with respect to the camera, and vice versa, to calculate the location of the corners on the camera image. This technique would be used in both ways. To generate location of the corners on the image of a known gate. That would be used for the benchmarking Section 8.4 to calculate the accuracy of the algorithms. And to find the location of the gate w.r.t the RAIDER, based on detected corners.



(a) A triangle clip frustum



(b) Perspective Frustum and Normalized Device Coordinates (NDC)

In order to calculate the position of the corners in 2D space, the 3D scene must be transformed to clip coordinates, then, these clip coordinates are then transformed to the normalized device coordinates (NDC) by dividing by a component of the clip coordinates. In other words, the real coordinates of the viewing frustum would be transformed into the cube with vertices [-1,1], as shown in Figure 8.8b. The pyramid frustum is set by the 6 parameters, near

(n) and far plane (f), left (l) and right (r) x-coordinates and top (t) and bottom (b) y-coordinates in the eye (camera) coordinate system. If the object is present in the viewing volume it will be seen by the camera. The distance to the near plane could be approximated by the camera focal length, which is typically around 23mm, the far plane 20m. Right, left, top, bottom, coordinates might be calculated by using camera FOV and the Pythagorean theorem. From this information, the transformation matrix from 3D to clipping coordinate system could be derived Equation (8.3), it further simplifies due to the fact that the right and left, and the top and bottom coordinates are symmetric, thus resulting in the terms with $r + l$ and $t + b$ being zero. Furthermore, the $(x_c, y_c, z_c)$ are divided by $w_c$ to get points in NDC. Lastly, the x and y coordinates are scaled to pixel size and used to find corners in the image, as can be seen in Figure 8.9.
Gate dewarping can be done by doing the opposite of what explained above and by using an inverse transformation matrix Equation (8.3).

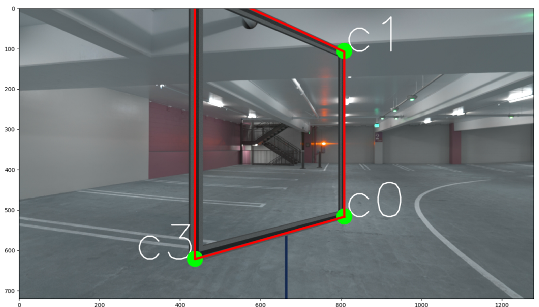


Figure 8.9: Example of estimated location of the gate corners

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix} \tag{8.3}$$

## 8.6. Software Verification & Validation

### 8.6.1. Verification

In order to assess the suitability of the simulation model for gate detection, the programming methods have been verified using unit test throughout the process. The verification tests have been applied at each stage of the development to check that each module was correctly coded. The image pre-processing unit was verified by controlling the level of distortion introduced in the images and by visually controlling the labelling of the corners. Moreover the corner detector, mainly composed of mathematical functions and array manipulation was verified by testing the method with input variables of which the result was already known. An example of this test is the intersection function where the line that intersects are the input and the point where this happens is the output. Another example is the corner detector that was verified to make sure it wasn't biased by checking that it would not return any corners in monochromatic images without any feature to be detected. Finally a system test was performed for each gate detection algorithm making sure that the TPR and FPR was not below the minimum threshold of 50% accuracy.

The DeCNN algorithm-hardware compatibility has been checked based on literature. Many projects that use convolutional neural networks, such as YOLO, have been run on a Raspberry Pi 3[4,5]. Even though the Raspberry Pi 3 has a quad-core processor it is comparable to the single ARM processor of the Raspberry Pi Zero W because most of these projects have been implemented in Python which cannot run on multiple cores at the same time. Although, in general, these projects run at a low frequency of less than 1 fps[6], it is important to remember that the YOLO CNN performs localization and classification of objects with 20 convolutional layers [45] , while the DeCNN developed for the racing drone has only 3 of these. More information about the performance of de DeCNN on the Raspberry Pi Zero W can be found in Chapter 11, where the frequency at which it can be run is approximated.

### 8.6.2. Validation

For validation, the software has to be tested with real images of gates, in order to ensure that the training set is a reliable source of images. The aim of the validation is not only to ensure that the model response is correct but also to demonstrate that synthetically generated images yield significantly better performance than using a limited set of

[4] *https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi-225f29635c74* [Visited at 21 June 2018]

[5] *https://github.com/DT42/BerryNet/blob/master/README.md* [Visited at 21 June 2018]

[6] *https://nl.mathworks.com/help/gpucoder/examples/pedestrian-detection.html?s_tid=gn_loc_drop&s_eid=PSM_15028* [Visited at 21 June 2018]

real images and apply different types of quality distortions on this ones [14]. Thus, in order to conduct a validation it is needed to run another benchmark simulation, on a small set of images as this ones have to be manually labelled by indicating the four corner position at every frame.

Furthermore, the complete validation process should be conducted on the real hardware,and if possible in later validation stages, using a real-time input of the camera in order to test the throughput time and the accuracy in standard operational conditions.

## 8.7. Time of Flight Camera: Algorithm Implementation

The software strategy for the ToF camera was not tested in an artificial racing environment due to the unavailability of gates depth images during the early subsystem design development. However, even if the implementation was only partially developed, the integration strategy has been established as this camera is an essential component for the success of RAIDER.

The primary mission of the ToF camera is to detect objects in front of drone and to classify these as either a gate or an obstacle. This distinction should be easy to make since obstacles are not normally hollow. The depth camera ensures that the drone does not hit a column or a gate side, making the drone more reliable. The secondary mission of the ToF camera is updating the gate waypoints and correcting drift values of IMU measurements.

The object detection with the ToF camera is achieved by threshold a depth map, in order to get a binary image, such as the output of the DeCNN, where the color intensity represent how close is an object with respect to the camera. Objects are detected with a resolution from 2 to 4 centimeters [7] and objects further than 4 meters cannot be detected therefore will have a black background color instead. To distinguish gate from the obstacle, the average distance or colour of the enclosed contour area is taken and compared with the background distance or colour respectively. If the average value is close to the background one, then the object is hollow, thus it is a gate, otherwise an obstacle. The same RDP-Contour algorithm executed on the DeCNN images is used to find the gate's corners. Since the ToF camera outputs the distance to the gate, the velocity of the drone can be calculated by numerically differentiating the distance. Additionally, because the ToF camera can perform at a high frame rate, its images are expected to be more reliable.

As for the implementation on the hardware, the ToF software will be executed on a thread with lower priority. Moreover, the algorithm to process the depth images is very light computationally as the resolution of the images (224 x 171px) is almost half of the RGB ones and they have only one information channel (depth) to be processed.

Finally the ToF algorithm will have two main modes, the first one outputs the relative position in all the axis, this method consist on finding obstacles by applying a simple contrast based algorithm and extracting the position of keypoints by looking at the depth values. The second one, returns the absolute position and is used once the gate is detected. This method tracks the gate when is fully visible but also after 1.6 meters from it when only the gate bars are in the field of view of the camera. Therefore, also when only a part of the gate will be visible, the update is done without having to process the corner position with the gate dewarping. This second algorithm is executed when only one corner is detected and it estimates the direction to be followed by examining the corner orientation. q

## 8.8. Strategy Overview and Conclusion

Figure 8.10 gives a complete overview of the main functions that are handled on the Raspberry Pi Zero W. All of the different steps and their logic is explained in it. The in and output and their dimensions, are clearly indicated outside of the main computer and marked in red. There are two different branches coming into the computer from the two cameras that feed the input images. As explained for the ToF camera, the running priority of each camera's output on the processor is determined based on the frequency of each algorithm. Therefore a higher running priority is assigned to the RGB image branch than to the depth image algorithm, since the DeCNN was selected as the main gate detection algorithm. However, if the gate is in sight of the ToF camera, then priorities would be switched because ToF will provide higher update rate of states than DeCNN. The throughput time and the amount of FLOPS that are needed to generate the gray-scale image and to perform the gate detection have been calculated. Subsequently this parameter has been translated into a frame rate based on the specific operations handled on board of the Raspberry Pi Zero W. In order to calculate the FLOPS it is required to have information of each CNN layer, such as the filters that were used, the size of the input, the kernel size, the strides, and the padding. With this parameters it is possible to estimate the number of FLOPS per layer[8]. The current neural network configuration has a performance of 80 MFLOPS. In order to highlight the gate contour found by the DeCNN, the RPD-Contour algorithm is used to localize

[7] *https://pmdtec.com/picofamily/assets/datasheet/Data-sheet-PMD_RD_Brief_CB_pico_flexx_V0201.pdf* [Visited at 22 June 2018]

[8] *https://github.com/sagartesla/flops-cnn* [Visited at 26 June 2018]

the point coordinates in the image. The complexity of contour is approximated to $O(N \log N)$[9], where N is the total number of pixels (288x512), this leads to another 2.9 MFLOPS, together with the image processing, for a total of 82.9 MFLOPS that are needed to calculate the gate position.

Finally, the *DeCNN-RDP-Contour* algorithm was estimated to have a running frequency of 3 FPS while the depth images processing is really fast. Therefore, the frequency depends on the video-feed itself which varies from 5 to 45 FPS depending on the distance from an object. This has been explained in more detail in Chapter 11.

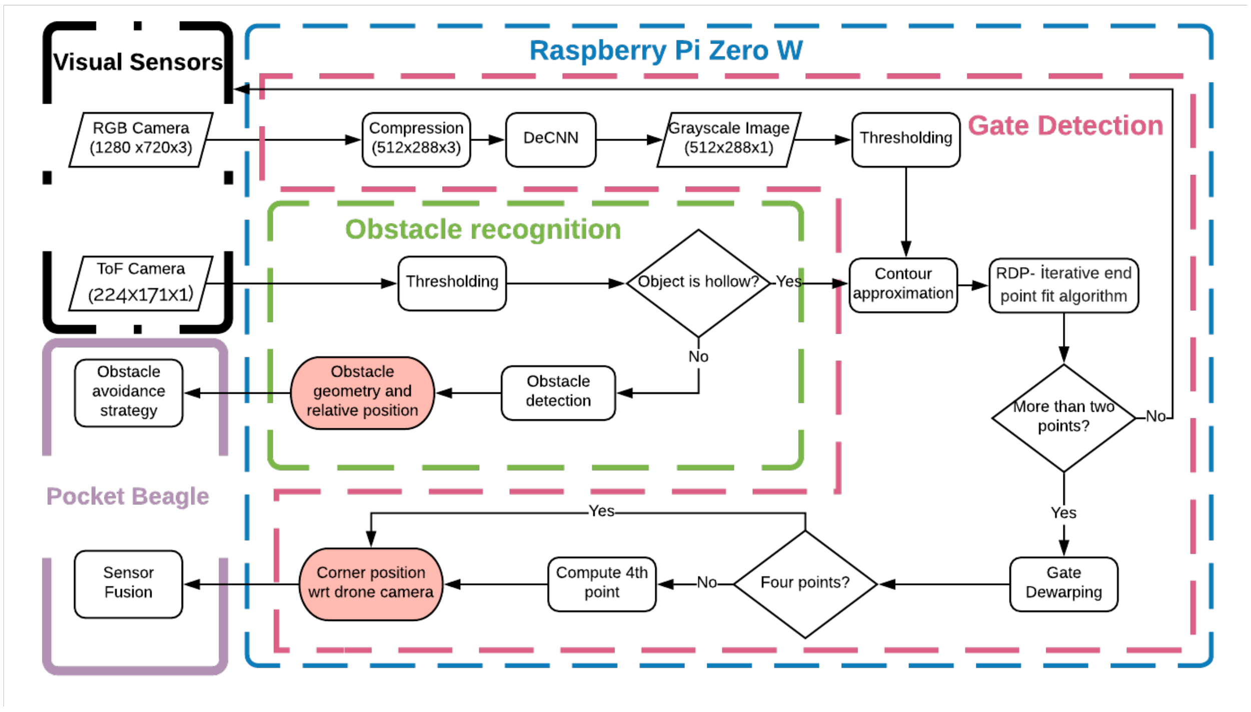


Figure 8.10: Gate detection strategy.

## 8.9. Future Development

### Programming Language

The programming language that has been used to simulate this algorithms is Python because it's widely supported for machine learning, complex data analysis and visualization. The implementation is aided by multiple open source application programming interfaces (APIs), in particular openCV[10] which was used for all the standard computer vision algorithms. Python is an interpreted language and is not compiled like C or C++ which significantly increases the throughput time. The main difference is the fact that Python is a dynamic programming language, where the type of variables do not need to be declared beforehand such as in static programming. Therefore, the variables that are created are not optimized for memory usage. In future implementations the software should be rewritten in a compilable language and optimized for efficient memory access. Finally, the programming language should be changed, to allow for multi-threading in order to concurrently execute the algorithm in charge of handling the RGB images and the one that uses the depth camera. In this way the overall accuracy will increase as both cameras will be used to extract different information about the environment.

[9] *https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html* [Visited at 26 June 2018]

[10] *https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html* [Visited at 30 May 2018]

# 9
# Path Planning & Optimization

Path planning is crucial to RAIDER's success in the IROS 2018 ADR and one of the most researched problems in the field of robotics. The primary objective of the path planning and optimization system is to determine an optimal and collision-free 3D path through a set of target nodes (gates) that respects the drone's kinodynamic limits. In this section, RAIDER's path planning and optimization approach designed for the IROS 2018 competition is discussed. Section 9.1 presents a brief overview of the related literature and most promising algorithms, motivating our choice to modify the joint polynomial optimization first proposed by Mellinger and Kumar [32] to achieve a robust and dynamically feasible path at minimal computational cost. Section 9.2 elaborates on our optimization approach and the way in which we intend to apply it in the ADR is discussed in Section 9.3. In Section 9.4, the path planning routine that was built will be stress tested to verify its robustness to problems posed under demanding constraints. Finally, we consider the limits of the Control and Gate Detection subsystems to plan a nominal path for participating in the race and validate it by means of an integrated simulation in Section 9.5.

## 9.1. Background & Present Work

Several general-purpose algorithms capable of generating trajectories for dynamical systems in space are available and well documented in literature.

Generally, roboticists tend to focus on discrete or randomized search approaches for motion planning. Common discrete approaches include using standard algorithms such as A* and Dijkstra's to search over graphs of feasible actions in the state-space. While these algorithms are promising for systems of four to five states, such as automobiles, their complexity increases dramatically with the dimensionality of the state-space [41]. Randomized approaches such as RRT (Rapidly-exploring Random Trees) are resistant to the 'curse of dimensionality' because the number of nodes to be explored does not scale as poorly with the size of the state-space, rendering them suitable, in principle, to search the 12 dimensional state-space of a quadcopter for an optimal sequence of actions. However, while these algorithms have been shown to return 'reasonable' trajectories, they are not designed to compute optimal trajectories and are likely to lead to overly costly, lengthy or dynamically unfeasible paths. In a race environment, this is highly impractical as the challenge does not lie in obtaining a 'reasonable' trajectory. This is a relatively simple problem given that the environment and most waypoints (gate positions) are known beforehand. Rather, the challenge lies in adaptively determining optimal trajectories that minimize some combination of the required control effort and time. Recently, RRT has been modified to guarantee global optimality for dynamical systems in the limit of infinite samples [22]. However, Bry showed that RRT* is very expensive for multiple DOF dynamical systems; he reported a runtime of 120s to optimize a path around 5 obstacles using a polynomial steering function on a 2GHz processor [5].

Instead, the control community has focused on treating path planning as a constrained non-linear optimization problem. The issue with this formulation is that the limits imposed by the dynamics of the system, control inputs, and obstacles all enter the problem as mathematical constraints. Respecting all constraints explicitly may be computationally intensive and can result in slow and numerically unstable optimization routines, particularly so when a large number of obstacles is present and the domain is represented as an occupancy-grid map with high resolution (which is a natural and tractable way to represent environments and has long been the standard in robotics). For instance, Barry and Majumdar report several minutes (3 to 5) of computational time to optimize a 4.5m trajectory for a 12-state, 5-input airplane maneuvering between cylindrical obstacles by solving the problem in the optimal control formulation using direct collocation[5][2]. In order to be competitive in the IROS competition, the path planning routine is required to run online and continuously re-plan the path in a fraction of a second, so standard optimization approaches are deemed unfeasible.

Mellinger and Kumar [32] bypass this problem by exploiting the property of differential flatness to ensure the dynamic feasibility of their trajectories. In their formulation, the dynamics of the drone did not enter the problem as formal constraints but rather, the control effort required to follow the path was limited by minimizing snap (the second derivative of the acceleration). From the algebraic mapping that exists between the differentially flat trajectory parameters and the required control inputs it is then trivial to verify the dynamic feasibility of the path. Bry et al.

successfully applied this method to a planning problem involving the flight of a quadcopter in a cluttered indoor environment [5]; due to their smoothness and tractability they choose polynomials as their basis functions and report a runtime of merely 0.2ms in jointly optimizing a three-segment path on a 2GHz Intel Atom processor. This method is extremely promising and the only one in literature that has been proven efficient enough to consider running it online for high-frequency re-planning.

In this chapter, we propose a modified version of the joint polynomial optimization approach described above geared to succeed in the IROS competition.

## 9.2. Method

The path planning approach described in this section is capable of efficiently determining the optimal path, which is subject to constraints on both the path parameters and their derivatives, through a series of waypoints. We describe a routine based largely on the work of Mellinger and Bry [32] [5] that allows for high-frequency path re-planning and is suitable to generating dynamically feasible paths through all gates and to avoid unexpected obstacles detected at close range. Section 9.2.1 discusses the path parameters, whereas Section 9.2 and Section 9.2.3 elaborate on the details of the optimization approach, focusing on the trajectory optimization and time segment allocation problems respectively. Finally, Section 9.3 discusses how the optimization routines will be implemented dynamically in the race.

### 9.2.1. Differential Flatness and Dynamic Feasibility

Given the multiplicity of reference frames involved in the path planning problem of a quadrotor, it is imperative to define the coordinate systems that will be used in this work, an overview of which is shown in Figure 9.1. Firstly, the track coordinate system $x, y, z$ is fixed and centered in the origin $O$ of the race track. The drone coordinate system, instead, is centered in the drone's center of mass and made up of the drone body axes $x_b, y_b, z_b$ and is continuously translating and rotating relative to $x, y, z$.

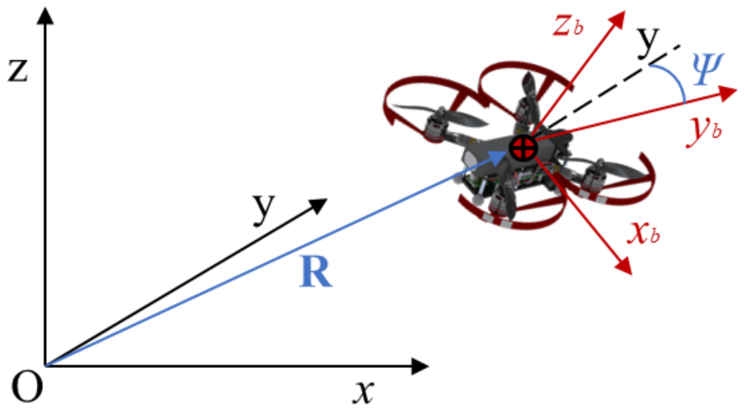


Figure 9.1: An overview of the coordinate systems used in this section. $x, y, z$ (black) is the track coordinate system and $x_b, y_b, z_b$ is the drone coordinate system (red).

Mellinger shows that the problem of quadrotor dynamics with four control inputs, namely the net thrust from all propellers and the three control moments around the drone body axes, is differentially flat. This means that there exists at least one set of parameters $\mathbf{\Theta}$ for which the state of the system can be written in terms of $\mathbf{\Theta}$ and its derivatives at any point in time. In this work, we select the following flat outputs:

$$\mathbf{\Theta} = [x, y, z, \psi]^{\mathrm{T}} \tag{9.1}$$

Where the triplet $\mathbf{R} = [x, y, z]^{\mathrm{T}}$ represents the position of the drone's center of mass in space relative to the track coordinate system and the angle $\psi$ is the yaw relative to the track system as shown in figure Figure 9.1. A trajectory $\mathbf{\Theta}(t)$ is a smooth curve defined parametrically in the flat output space as follows:

$$\mathbf{\Theta}(t) : [t_0, t_m] \rightarrow \mathbb{R}^3 \times SO(2) \tag{9.2}$$

For this particular set of flat outputs, Mellinger [33] shows that there exists some smooth map $M$ such that Section 9.2.1 holds.

$$(\mathbf{x}, \mathbf{u}) = M(\mathbf{\Theta}, \dot{\mathbf{\Theta}}, \ddot{\mathbf{\Theta}}, \dddot{\mathbf{\Theta}}, \ddddot{\mathbf{\Theta}}) \tag{9.3}$$

Where $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$ is the 12-dimensional state vector of the drone, including the position and velocity of the center of mass and the orientation (Euler) angles and angular velocities. The vector $\mathbf{u}$ contains the four control inputs $\mathbf{u} = [u_1, u_2, u_3, u_4]^\mathrm{T}$, which represent the net thrust from the propellers and the control moments about the x,y, and z axes respectively. A full demonstration that some $M$ exists for which Section 9.2.1 holds is not reported here but can be found in [33], it is sufficient for the reader to know that such a mapping exists and is smooth.

The magnitude of the net thrust $u_1$ is quite straightforward to express in terms of the differentially flat trajectory parameters $\mathbf{\Theta}$:

$$u_1 = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2} \tag{9.4}$$

Given that the lifting capabilities of the drone hardware desired by the customer are significantly higher than those at which we expect to operate in the race, the control input $u_1$ is not deemed critical from a path planning perspective. The control moments $u_2$, $u_3$ and $u_4$ are much less predictable and will effectively define the dynamic feasibility of the trajectory. In $M$, the inputs $u_2$ and $u_3$ appear as functions of the fourth derivative of the position $\ddddot{\mathbf{R}}$ (known as *snap*), whereas $u_4$ is related to the second derivative of the yaw angle $\ddot{\psi}$. Therefore, we focus on ensuring minimal control effort by minimizing $\ddddot{\mathbf{R}}$ and $\ddot{\psi}$ (as is commonly done in literature), and later verify that the net thrust profile $u_1$(t) throughout the trajectory is suitable.

## 9.2.2. Trajectory Optimization

### Position

Given that the states and inputs are dependent not only on the flat position variables $\mathbf{R} = [x, y, z]^\mathrm{T}$ but also on their derivatives, it is convenient to choose a basis function for $\mathbf{R}$ that has well-defined and tractable derivatives at any point in time. For this reason, we choose polynomials as our basis functions. Now consider the progression of a flat position variable, for instance $x$, throughout the racetrack. The trajectory will need to meet a series of discrete position constraints throughout the track, which we call *waypoints*; physically, these waypoints represent the spatial coordinates of the center of the gate, through which the drone must pass in order to minimize the probability of collision. The flat output variable will also need to meet constraints on its derivatives. For instance, we impose that the velocity vector should be perpendicular to all gates to minimize the likelihood of drifting sideways while passing through a gate.

A natural way to enforce all this is to define the position along the trajectory $\mathbf{R}_T(t) = [x_T(t), y_T(t), z_T(t)]^\mathrm{T}$ as a series of polynomial segments of order $N$ that satisfy several continuity constraints at the waypoints.

$$\mathbf{R}_T(t) = \begin{cases} \sum_{i=0}^{N} \mathbf{r}_{T1,i} t^i : & t_0 \leq x \leq t_1 \\ \sum_{i=0}^{N} \mathbf{r}_{T2,i} t^i : & t_1 < x \leq t_2 \\ \vdots & \\ \sum_{i=0}^{N} \mathbf{r}_{Tm,i} t^i : & t_{m-1} < x \leq t_m \end{cases} \tag{9.5}$$

As explained in Section 9.2.1 we are looking for a trajectory that minimizes the snap as it is mapped into the control inputs $u_2$ and $u_3$. Therefore, we minimize the square of the fourth derivative of the position. The resulting optimization problem is the following:

$$\begin{aligned} &\min \int_{t_0}^{t_m} \mu_r \left\| \frac{\mathrm{d}^4 \mathbf{R}_T}{\mathrm{dt}^4} \right\|^2 dt \\ &\text{s.t.} \quad \mathbf{R}_T(t_i) = \mathbf{R}_i \qquad\qquad i = 0, ..., m \\ &\qquad \left. \frac{\mathrm{d}^k x}{\mathrm{dt}^k} \right|_{t=t_j} = 0 \text{ or free} \quad j = 0, \dots, m; \quad k = 1, \dots, 4 \\ &\qquad \left. \frac{\mathrm{d}^k y}{\mathrm{dt}^k} \right|_{t=t_j} = 0 \text{ or free} \quad j = 0, \dots m; \quad k = 1, \dots, 4 \\ &\qquad \left. \frac{\mathrm{d}^k z}{\mathrm{dt}^k} \right|_{t=t_j} = 0 \text{ or free} \quad j = 0, \dots m; \quad k = 1, \dots, 4 \end{aligned} \tag{9.6}$$

Here, $\mu_r$ is a constant to make the integral non-dimensional, and $\mathbf{R}_i = [x_i, y_i, z_i]$ are explicit position constraints at the waypoints (gates). It is possible to formulate the problem in Equation (9.6) as a standard quadratic program

(QP) by grouping the coefficients of **R** into a vector **r** and the constraints into a vector **b**. As observed by Bry [5], solving the problem for all polynomial segments at once yields lower cost trajectories than optimizing each segment individually, so it was chosen to frame the problem in the joint optimization formulation and solve a single QP for all time rather than multiple QPs for every polynomial segment. Since the components of **R** are decoupled in their cost function (they are positions along orthogonal axes), it is possible to solve for $x, y$ and $z$ separately. 3 QPs were therefore solved over the entire time span of the trajectory (one for each component of **R**); notwithstanding what is done by Bry [5], we solve the problem directly in the constrained formulation:

$$\begin{aligned} \min_{\mathbf{r}} \quad & \mathbf{r}^{\mathbf{T}}\mathbf{Q}\mathbf{r} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{r} - \mathbf{b} = 0 \end{aligned} \tag{9.7}$$

Where the matrix **A** maps the coefficients **r** to the appropriate polynomial derivatives to be constrained, and **Q** is a Hessian cost matrix that maps each polynomial derivative to the appropriate penalty.

The constraint matrix **A** is relatively straightforward to construct given the analytical traceability of polynomial derivatives. The $k$th derivative of **R** (or any of its components) at some time $t$ can be written as:

$$\mathbf{R}^k(t) = \sum_{n=k}^{N} \left( \prod_{m=0}^{k-1} (n-m) \right) \mathbf{r}_n t^{n-k} \tag{9.8}$$

A single polynomial segment starting from time 0 at some waypoint and ending at time $\tau$ at the next waypoint must satisfy constraints on both its ends, which can be constructed as:

$$\mathbf{A} = \begin{bmatrix} A_0 \\ A_\tau \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_\tau \end{bmatrix} \tag{9.9}$$

$$\mathbf{A}_{0,kn} = \begin{cases} \prod_{m=0}^{k-1}(k-m): & k=n \\ 0: & \text{otherwise} \end{cases} \qquad \mathbf{b}_{0,k} = \mathbf{R}^{(\mathbf{k})}(0) \tag{9.10}$$

$$\mathbf{A}_{\tau,kn} = \begin{cases} \left(\prod_{m=0}^{k-1}(n-m)\right)\tau^{n-k}: & k \geq n \\ 0: & \text{otherwise} \end{cases} \qquad \mathbf{b}_{\tau,k} = \mathbf{R}^{(\mathbf{k})}(\tau) \tag{9.11}$$

Where $\mathbf{A}_{0,kn}$ and $\mathbf{A}_{\tau,kn}$ are the components of matrices $A_0$ and $A_\tau$ respectively, which both have dimensions $5 \times N$ (one row per derivative $k$ including the 0th derivative and one column per polynomial coefficient). Denoting $\mathbf{R}_j^{(k)}$ as the value of the $k$th derivative of the trajectory at waypoint $j$, continuity of the $k$ derivative follows from: $-\mathbf{R}_{j+1}^{(k)} + \mathbf{R}_j^{(k)} = 0$. Let $A_0^j$ be the constraint matrix for a trajectory starting at waypoint $j$ and let $A_\tau^j$ be the constraint matrix for a trajectory ending at waypoint $j$. Then we can write matrix **A** in such a way as to enforce specific values on the derivatives $\mathbf{R}_j^{(k)}$ while also imposing their continuity:

$$\mathbf{A} = \begin{bmatrix} A_0^0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -A_\tau^0 & A_0^1 & 0 & 0 & \dots & 0 & 0 \\ 0 & -A_\tau^1 & A_0^2 & 0 & \dots & 0 & 0 \\ 0 & 0 & -A_\tau^2 & A_0^3 & \dots & 0 & 0 \\ & & & & \ddots & & \\ \dots & \dots & \dots & \dots & & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & -A_\tau^{m-1} & A_0^m \\ 0 & 0 & 0 & 0 & 0 & 0 & A_\tau^m \end{bmatrix} \tag{9.12}$$

The Hessian matrix Q is obtained from the cost function in Equation (9.6), using standard mathematical formulations. The square of a polynomial, $\mathbf{R}^2$, can be written using the convolution sum:

$$(\mathbf{R}^2)_n = \sum_{j=0}^{N} r_j r_{n-j} \tag{9.13}$$

Here, $(\mathbf{R}^2)_n$ is the $n$th coefficient of a polynomial of degree $N$. Bry shows that it is possible to rewrite the cost function using Equation (9.13) and Equation (9.8), where the 4th derivative of **R** is computed. In order to construct the Hessian matrix, the rewritten cost function is derived two times with respect to each of the polynomial's coefficient. Equation (9.14) shows how to obtain all components for a particular segment, where $\tau$ represents its time duration. The full matrix **Q** is constructed assembling the different matrix segments as a block diagonal matrix. A complete derivation can be found in [5].

$$\mathbf{Q}_{il} = 2\left( \prod_{m=0}^{3} (i-m)(l-m) \right) \frac{\tau^{i+l-8+1}}{(i+l-8+1)} \tag{9.14}$$

### Yaw

Our approach to optimizing the yaw angle $\psi$ differs from anything encountered in literature and is geared specifically towards navigation through sparse waypoints with characteristics that are useful to track. In order to increase the chances of RAIDER winning the IROS 2018 ADR, the priority of the path planning should not only be to minimize the control inputs required, but more importantly to increase the chances of the drone to orient itself in the environment. For this reason, the planned path needs to maximize the probability of detecting a gate with one of the techniques presented in Chapter 8. The yaw angle needs to be optimized such that the drone is always pointing to the next gate. This formulation of the yaw angle is applicable to the full trajectory, except for the parts of the path where the drone needs to adjust its orientation after passing a gate to start pointing to the next one. This situation is clearly displayed in Figure 9.2, where the red dots represent the gates and the vertical bars represent the transition zones. These changes in orientation constitute small discontinuities along the path, which are impossible to actuate in reality. Equation (9.15) and Equation (9.16) show how the yaw function is defined.

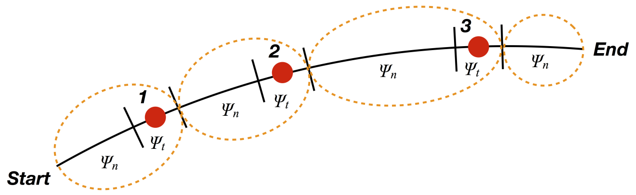


Figure 9.2: Schematic representation of the variation of the yaw function along the trajectory.

$$\psi_T(t) = \begin{cases} \psi_{T1}(t) & t_0 \le t \le t_1 + \frac{t_{join}}{2} \\ \psi_{T2}(t) & t_1 + \frac{t_{join}}{2} < t \le t_2 + \frac{t_{join}}{2} \\ \vdots & \\ \psi_{Tm}(t): & t_{m-1} + \frac{t_{join}}{2} < t \le t_m \end{cases} \tag{9.15}$$

Here, $t_{join}$ is the time required to pass a transition zone, and $[t_0, t_1, t_2 \ldots t_{m-1}, t_m]^T$ represents the time of the different **m+1** waypoints, which include the initial, the final positions and the time at which a gate is crossed.

$$\psi_{Ti}(t) = \begin{cases} \psi_{ni}(t) = \arctan\left(\frac{x(t)-x(t_i)}{y(t)-y(t_i)}\right) & t_{start} \le t \le t_i - \frac{t_{join}}{2} \\ \psi_{ti}(t) & t_i - \frac{t_{join}}{2} < t \le t_i + \frac{t_{join}}{2} \end{cases} \tag{9.16}$$

Equation (9.16) is applicable from i = 1 to m-1, the last segment ($\psi_{Tm}(t)$) has only $\psi_n(t)$ part, from the beginning ($t_{m-1} + \frac{t_{join}}{2}$) until the end ($t_m$). x(t), y(t) have been obtained as explained in Section 9.2.2.

$\psi_n(t)$ represents the intra-gate part, the yaw angle in the track reference system described in Section 9.2.1, can be calculated from the $[x, y]^{\mathrm{T}}$ trajectory and the position of the waypoints by means of a simple geometric relationship. Due to the sparsity of the gates this portion of the yaw is easily controllable as the drone is typically required to adjust the yaw slightly at every new position to keep looking at the same gate. The arctan has the characteristic that are needed for the polynomial function described in section 9.2.2, it is easily differentiable and continuous at every point in time.
$\psi_t(t)$ is the transition part, where a relevant change in yaw angle is required. The controllability of the drone in this part depends on the duration of the maneuver, and as shown by Mellinger [33] the second derivative of the yaw. A cost function is defined as shown in Equation (9.17) to optimize these parameters.

$$\min \int_{t_{0i}}^{t_{0i}+t_{join}} \left\| \frac{\mathrm{d}^2\psi_{ti}}{\mathrm{d}t^2} \right\|^2 dt \qquad i = 0, \ldots, m-1 \tag{9.17}$$

Where $t_{0i}$ refers to the initial time of the transition segment, and $t_{0i} + t_{join}$ refers to the final time of the transition segment. The function used to fit the initial and final state is the cubic smoothing spline, which fits a sample of points between each $\psi_{ni}$ and $\psi_{ti}$,ensuring a smooth transition that minimizes the control inputs.

### 9.2.3. Time Segment Allocation

In the IROS competition, the total time allowed for the trajectory is important but the arrival time at different waypoints is not (except perhaps for the dynamic obstacle but our solution to this particular challenge is explained in Section 9.3). Intuitively, a good solution is one that also distributes the total allowed time optimally between each trajectory segment. This can be done a-posteriori by first assuming a time distribution (for instance by assuming the quadrotor travels at constant velocity and spends the same amount of time in each segment), computing the minimum-snap trajectory and then solving a second optimization problem to re-scale the coefficients in such a way as to optimize the time allocated per segment [32]. Let $T_i$ be the time spent in the $i$th segment, then the optimization problem to be solved is:

$$\begin{aligned} \min \quad & f(\mathbf{T}) & \\ \text{s.t.} \quad & \sum T_i = t_m & i = 1,\dots,m \\ & T_i \geq 0 & i = 1,\dots,m \end{aligned} \tag{9.18}$$

where $f(\mathbf{T})$ is the solution to the optimization problem in Equation (9.6) with segment times $\mathbf{T} = [T_1, T_2, \dots, T_m]$. As done by Mellinger, we solve this via a constrained gradient descent method. We compute the directional derivatives of $f(\mathbf{T})$ along $m$ directions $\mathbf{g}_i$:

$$\nabla_{\mathbf{g}_i} f = \frac{f(\mathbf{T} + h\mathbf{g}_i) - f(\mathbf{T})}{h} \tag{9.19}$$

where $h$ is small and the vectors $\mathbf{g}_i$ are constructed such that the $i$th element has a value of 1 and all other elements have value $\frac{-1}{m-1}$. This way, $\sum \mathbf{g}_i = 0$ and different time distributions can be obtained by adding a small multiple of $\mathbf{g}_i$ to the vector of time segments $\mathbf{T}$ without changing the total time $t_m = \sum T_i$. For instance, in the case where m=3 (3-segment optimization problem), we have: $\mathbf{g}_1 = [1, \frac{-1}{2}, \frac{-1}{2}]^{\mathrm{T}}$, $\mathbf{g}_2 = [\frac{-1}{2}, 1, \frac{-1}{2}]^{\mathrm{T}}$ and $\mathbf{g}_3 = [\frac{-1}{2}, \frac{-1}{2}, 1]^{\mathrm{T}}$ and the final gradient is:

$$\nabla f = \begin{bmatrix} \nabla_{\mathbf{g}_1} f \\ \nabla_{\mathbf{g}_2} f \\ \nabla_{\mathbf{g}_3} f \end{bmatrix} = \begin{bmatrix} \frac{f(\mathbf{T}+h\mathbf{g}_1)-f(\mathbf{T})}{h} \\ \frac{f(\mathbf{T}+h\mathbf{g}_2)-f(\mathbf{T})}{h} \\ \frac{f(\mathbf{T}+h\mathbf{g}_3)-f(\mathbf{T})}{h} \end{bmatrix} \tag{9.20}$$

## 9.3. Application

Figure 9.3 shows an overview of how this optimization approach will be applied in the IROS competition, which is further explained below.

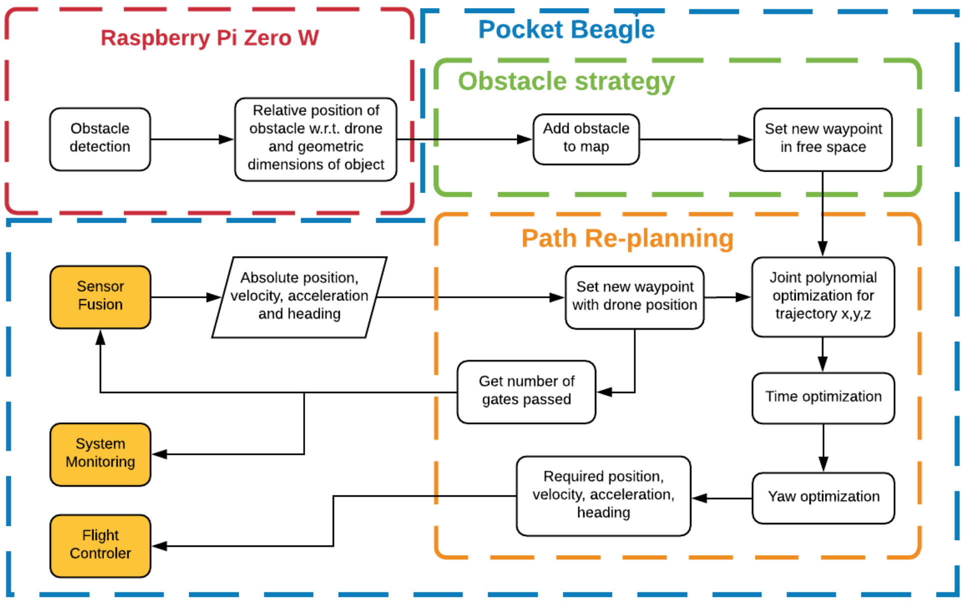

Raspberry Pi Zero W
Pocket Beagle
Obstacle strategy
Obstacle detection
Relative position of obstacle w.r.t. drone and geometric dimensions of object
Add obstacle to map
Set new waypoint in free space
Path Re-planning
Sensor Fusion
Absolute position, velocity, acceleration and heading
Set new waypoint with drone position
Joint polynomial optimization for trajectory x,y,z
Get number of gates passed
Time optimization
System Monitoring
Required position, velocity, acceleration, heading
Yaw optimization
Flight Controler

Figure 9.3: The figure shows the flow chart for the path planning.

### Path Planning and Replanning

Given the gate positions known a-priori, RAIDER will first plan an optimal path $\mathbf{\Theta}$ through all gates from start to finish of the track and begin following it. Every time a new pose estimate is received and the path planning routine is not running, RAIDER will re-plan a path through the remaining portion of the track taking its current position, velocity and acceleration as the first waypoint constraints to correct for drift. The constraints are imposed in such a way that the drone flies through the center of every gate and with its velocity vector perpendicular to the face of the gate, in order to minimize the probability of colliding with the gate or drifting towards its edges. We impose zero velocity, acceleration, jerk and snap both at the start (takeoff) and at the end (landing) of the track. We use this application, which is the most intensive, to verify our path planning routine in Section 9.4 and validate its compatibility with the other subsystems in the context of the IROS competition in Section 9.5.

#### Obstacle Avoidance

In the event that an unexpected obstacle is detected by the depth camera, RAIDER will receive its relative position to the obstacle and the width of the obstacle in the plane in which it is being viewed. RAIDER will then place waypoints clear of the obstacle, at a minimum in-plane distance equivalent to twice the width of the drone. The new waypoints will be used to plan a safe correction path from the last known position, velocity and acceleration to the next gate. This application uses the routine described in Section 9.2 and verified in Section 9.4. We validate the feasibility of this approach in relation to the other subsystems and in the context of the IROS competition in Section 9.5, by simulating the avoidance of an unexpected obstacle detected at close range.

#### Dynamic Gate Avoidance

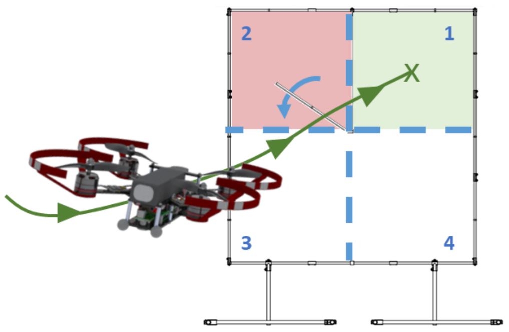


Figure 9.4: A schematic of RAIDER selecting the safest quadrant of the dynamic gate to pass through

It is also worthwhile to mention how we intend to pass through the dynamic gate, which contains a bar that rotates about the center of the gate. We assume that the direction of rotation of the bar is known as it can be observed before the race and otherwise tracked by the depth camera. The idea is to take advantage of RAIDER's agility to treat the dynamic gate as a static obstacle from a path planning perspective. We divide the dynamic gate into four quadrants of equal dimensions, and impose that RAIDER shall pass through the center of the quadrant that is furthest from the one in which the bar was last located. This approach will be validated in Section 9.5 by showing that under the suggested nominal path RAIDER has more than enough time to 'beat' the rotating bar to the center of the chosen quadrant under the range of rotational speeds expected.

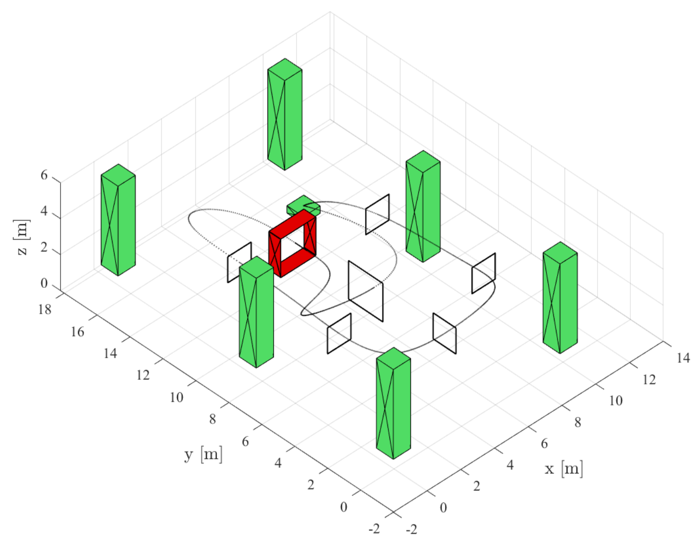
## 9.4. Prototype Testing and Verification

In order to verify the performance of the path planning approach described in Section 9.2, we built a working prototype in MATLAB and used it to plan optimal paths through different environments including the IROS racetrack. In this section, we discuss the unit tests and system tests performed to ensure the robustness of our routine. We also perform a full-scale system test on the most critical case expected in race operations, namely the task of planning an extremely aggressive path from start to finish of the track. The quadratic programming problem described in Section 9.2.2 is solved using the interior-point-convex version of MATLAB's `quadprog` solver whereas the time segment allocation problem described in Section 9.2.3 is solved using the `fmincon` nonlinear solver. The optimization problem described in Section 9.2.2 is solved using a cubic smoothing spline implemented through MATLAB's `csaps` solver.

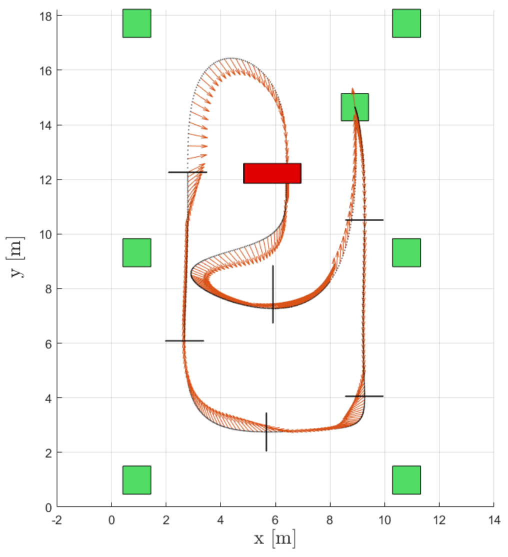
### 9.4.1. Comparison with Analytical Solutions

The individual parts of code that make up the path planning routine were verified by means of several unit tests that consisted of solving optimization problems with known analytical solutions or properties. The joint polynomial optimization routine described in Section 9.2.2 was tested by determining the optimal path for analytically simple waypoint problems. For instance, we verify that the minimum-snap solution to the problem with $m = 5$ and $n = 4$ is the only fourth degree polynomial that passes through all 5 waypoints. The yaw optimization routine described in Section 9.2.2 was also verified by solving problems with a known analytical solution. We optimize the yaw angle for a series of $m = 4$ waypoints all aligned along the $y$-axis and verify that the optimal solution is indeed the trivial case in which the drone maintains its body axis $y_b$ aligned with the $y$-axis ($\psi = 0$) for the entire path. We also solve a problem with $m = 4$ in which the waypoints are equally spaced around a circle and the drone is taken to be travelling at constant velocity; we verify that the resulting yaw distribution varies equally across all four segments. The same circular waypoint setup is used to verify the time segment allocation unit described in Section 9.2.3 as well. We impose that all waypoints should be traversed with the same velocity and with the velocity vector aligned with the tangent to the circle at the location of each waypoint. After initializing the problem with varying time segments per quarter of the circle, we verify that the final solution has allocated the same amount of time to every segment.

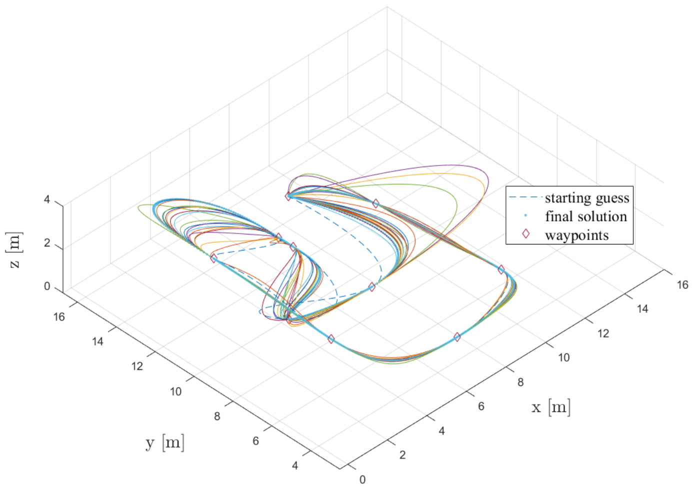
### 9.4.2. Full-Scale System Test: Critical Path through Entire Racetrack



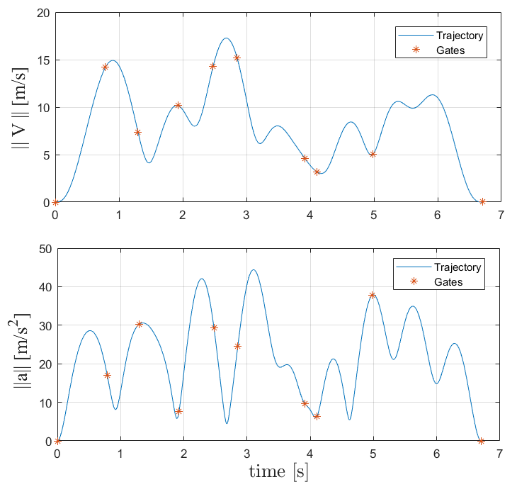
(a) 3d view of the optimal path through the entire racetrack for $t_m$ = 6.7s



(b) Top view of the global path shown in (a). The arrows are aligned with the direction of the drone body axis $y_b$ throughout the trajectory.



(c) Solution converging from an initial guess to the optimal path that satisfies all waypoint constraints shown in (a).



(d) Velocity (top) and acceleration (bottom) norms demanded by the path

Figure 9.5: Results of full-scale system test

The task of determining an optimal path through the entire racetrack is, intuitively, the most intensive as it involves jointly optimizing the largest amount of polynomial segments. It is a natural case on which to stress-test our path planning routine. In its current formulation, our path planning algorithm is designed to be flexible and takes in a user-specified total time allowed for the entire path without imposing it. This way, the user can easily control the degree of 'safety' desired and use the parameter $t_m$ to choose whether to generate safer (easily controllable) or more aggressive trajectories. This flexibility is desirable in the context of high-frequency re-planning on-board

RAIDER because, in practice, the total time allowed for the trajectory will depend on a large number of factors, such as battery life, gate detection performance and the time left in the competition. For instance, on the final attempt in the competition, with little time left on the clock, it may be required to choose a small $t_m$ to generate a very fast and aggressive path, while in the first attempt it may be preferable to prioritize controllability and choose a larger $t_m$.

We test our algorithm on the most numerically demanding case that may, in principle, be encountered, namely the fastest possible path that RAIDER can fly given its dynamic constraints. For the purpose of this system test, we make a conservative estimate for such a path iteratively by comparing the thrust-to-weight ratio demanded by the trajectory at every point in time to the maximum available thrust associated to the demanded airspeed at that point in time, using the methods described in Section 6.3.1 (as explained in Section 6.3.1, the thrust available is a function of airspeed since the propellers lose efficiency with increasing airspeed). Whenever the required thrust exceeds the available thrust, a constraint is activated and the total time $t_m$ is decreased. The fastest possible path that does not activate any of RAIDER's dynamic constraints takes $t_m$ = 6.7s to complete the entire track. It must be noted that this path is not representative of the true dynamic capabilities of RAIDER, but it is determined by using the simplified drag model described in Section 6.3.1, and assuming that the drone is navigating the path as a point mass and neglects the changes in attitude required to accelerate along the trajectory. A nominal path through the IROS racetrack that fully accounts for the rigid body dynamics of the drone, the performance of the flight controller and the gate detection subsystem is also determined using the approach described in Section 9.2 and is presented in Section 9.5.1. The intent here, instead, is to validate that our routine does not encounter numerical stability issues under an extremely demanding total allowed time $t_m$ that purposely overestimates RAIDER's dynamic capabilities. Failed convergence of the path planning routine is a critical risk associated to this subsystem and it is imperative to ensure that the algorithm is robust under conditions that are more demanding than the ones expected under nominal race operations. For instance, Bry reports numerical stability issues in his implementation[5] for more than 5 segments under tight time constraints.

For the smallest possible $t_m$ = 6.7 s, we find that the polynomial order of $\mathbf{R}$ for which the minimum cost is achieved is $n$=8. As can be seen in Figure 9.5a, the resulting trajectory smoothly meets all constraints and appears to be natural, which serves as qualitative validation for the cost function used.Figure 9.5d shows the net velocity and acceleration required along the path. Remarkably, our routine encountered none of the numerical issues described by Bry while generating paths with polynomial segments of orders up to $n$=15 for $t_m$=6.7s; Figure 9.5c shows the convergence of the polynomial trajectory solution $\mathbf{R_T}$ to the final one shown in Figure 9.5a. The optimal yaw attitude as determined via the method described in Section 9.2.2 is also plotted in Figure 9.5b and shows smooth intra-gate transitions.

## 9.5. Integrated Race Strategy and Validation

In this section, we delve more into detail regarding the application and integration of the path planning routine in the IROS race as described in Section 9.3. In Section 9.5.1 we determine a nominal path through the entire racetrack that takes into account the limits of other subsystems and validate its feasibility by means of an integrated race simulation; in doing this, it was also validated that RAIDER will be able to safely avoid the dynamic gate. In Section 9.5.2 the proof of concept was provided for the obstacle avoidance routine described in Section 9.3 by simulating the case in which the drone detects an unexpected obstacle at close range and must plan a dynamically feasible maneuver to avoid it.

### 9.5.1. Nominal Path through Entire Racetrack

In section 9.4.2, it was showed that in principle the planning algorithm is able to cope with planning a path with $t_m$ = 6.7 s. This is obviously far from the nominally optimal path for completing the track and winning the race. Such a path needs to take into account the constraints of the computer vision, the thrust capabilities of RAIDER, the maximum allowable drift and the flight controller.

In Chapter 8, a strategy based on an RGB and ToF cameras running simultaneously was defined. The ToF only works if the gate is at a distance of 4 m, since it is required to have a detection before this moment the RGB camera has the constraining performance. It runs a detection algorithm at 3.1 fps, and in a frame in which a gate is in view, it has a detection accuracy of 65 % with the camera at an angle between 45 and -45 degrees with respect to the gate. In order to propose a nominal path to be flown during the race, the probability of finishing the track has to plotted as a function of the total time. This was done by assuming that every segment is an independent Bernoulli trial. Using the path shown in Table 9.3, an approximation of the percentage of time that is spent in each segment can then be calculated by normalizing these results. Multiplying this with the total track time, the amount of time spent in each segment, together with the time the gate is in sight during the segment is calculated.
The probability of passing through the gate at the end of a segment is then calculated using the equation in Equation (9.21). Here, $\bar{G}$ is the even that no gates have been detected, $N$ is the event that a gate was last detected $n$ frames

ago. $P(Pass|n)$ is the probability that the drone flies through the gate on its IMU, given that the last detection was $n$ frames ago.

$$P(Pass) = P(\bar{G}) \cdot P(\text{Pass}\,|\,) + \sum_{n=0}^{n_{max}-1} P(N=n) \cdot P(Pass|N) \tag{9.21}$$

Here, equations in Equation (9.22) were used to calculate each of the probabilities used in Equation (9.21).

$$\begin{aligned} P(\bar{G}) &= (1 - c_{\text{MB}} \cdot \eta_{CNN})^{n_{max}} \\ P(N=n) &= (1 - c_{\text{MB}}\eta_{CNN})^{n} \cdot c_{\text{MB}} \cdot \eta_{CNN} \\ P(Pass|\bar{G}) &= 1 - 2 \cdot \text{CNF}(\mu = 0, \sigma = \frac{x_{Drift}}{3}, x = -0.5) \end{aligned} \tag{9.22}$$

In Equation (9.22), $\eta_{CNN}$ is the accuracy of the CNN, as described in Chapter 8, $\mu$ is the average drift value, which is 0 due to the random nature of drift. $x_{Drift}$ is the amount of drift that is calculated according to Section 7.1.1. This value for $x_{Drift}$ is then divided by 3 to account for the fact that the manufacturer's measurements can be slightly off. They are however, expected to be very accurate, which is why the value of 3 was chosen. $x$ sets the maximum amount of allowable drift, which is 50cm. CNF, in Equation (9.22), represents the cumulative normal distribution. The amount of drift is modelled like a normal distribution because of its origins in noise, which can often be modelled by a normal distribution. Finally, $c_{\text{MB}}$, is used to model the effects of motion blur on the neural nets detection accuracy. In literature, motion blur is often modeled as can be seen in Equation (9.23) [36], where g(x) is the image with motion blur applied and f(x) is the original image. The L in Equation (9.23) is the distance that the object in the picture moves during the exposure time. This distance can therefore be calculated by multiplying the drone's speed by the shutter speed. In order to be able use Equation (9.23), it had to be normalized, which was done by adding 1 to L, resulting in $\frac{1}{L+1}$.

$$g(x) = \frac{1}{L} f(x) \tag{9.23}$$

The result of this calculation can be seen in Figure 9.6. It should be noted, however, that the program has not been validated. As can be seen here, finishing the track in 60 seconds has the highest probability of success. This is why a total time of 60 seconds was chosen to be the nominal path. In the event of a competitor having finished the course in a smaller amount of time or RAIDER having finished the track in 60 seconds already, the team recommends a total track time of a little over 40 seconds. After this track times of 30 seconds and less would even be possible, although the probability of finishing the track drops very quickly.

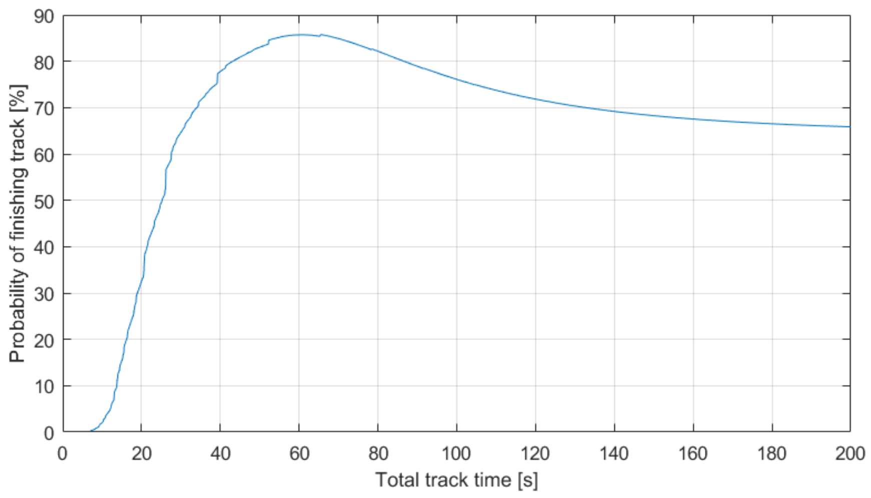
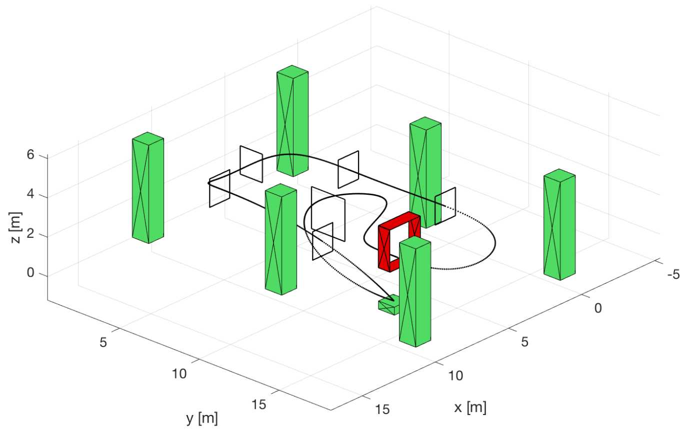
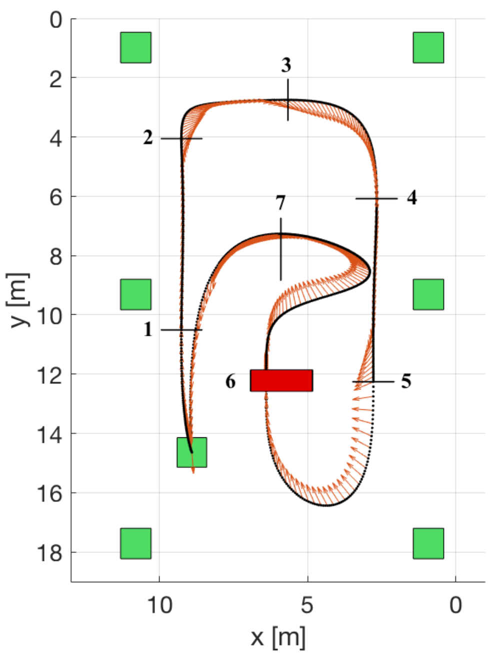


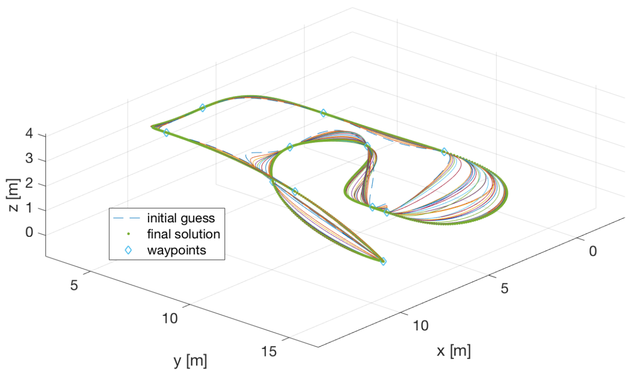
Figure 9.6: The probability of finishing the track as a function of the total track time.



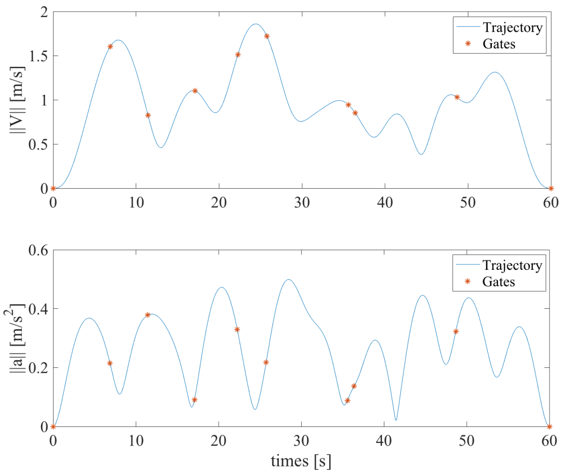
(a) The figure shows the 3D trajectory for the nominal path.



(b) The figure shows the yaw angle in each point of the race.



(c) The figure shows the iterative convergence of the planned path.



(d) The figure shows the norm of the velocity on the top and the acceleration on the bottom deemed by the path.

Figure 9.7: The figures show the nominal path for t = 60 s. From left to right: the 3D trajectory, the top view, the convergence of the planned path and total velocity, acceleration and T/W.

The minimum distance at which the drone will have a clear view of the gate can be easily calculated via the geometric relation shown in Equation (9.25), where $\omega$ is the smallest field of view of the camera and $h$ the biggest geometric dimension of a gate. Table 9.2 shows the results for all different gates and cameras. The maximum allowable time before needing a positive gate detection computed in chapter 7 is 9.3 s. Furthermore, the nominal path needs to be within the thrust limits of the propulsion systems, and it needs to be controllable, which means that it doesn't experience abrupt and excessive accelerations. The dynamic thrust limit for the given point in time is calculated using the dynamic thrust equation of Chapter 6.

$$\begin{aligned} &\text{g : number of gates detected in n frames} \\ &\text{X : successful detection of a gate in a frame} \\ &P(g \geq 1) = \sum_{i=1}^{n} \binom{n}{i} \cdot P(X)^i \cdot (1 - P(X))^{n-i} \end{aligned} \tag{9.24}$$

$$\text{range} = \frac{h/2}{\tan(\omega/2)} \tag{9.25}$$

Table 9.1: This table shows the probability of detecting at least a gate in a certain amount of frames.

| Frames | Probability of detection |
|---|---|
| 2 | 0.878 |
| 3 | 0.957 |
| 4 | 0.985 |
| 5 | 0.995 |

Table 9.2: This table shows the minimum distance at which the cameras are able to have a clear view on the gate.

| Range | RGB camera [m] | ToF camera [m] |
|---|---|---|
| Normal gate | 1.26 | 1.72 |
| Jungle gate | 1.88 | 2.57 |
| Dynamic gate | 1.88 | 2.57 |

In order to meet all these requirements an iterative approach was used with the intent of minimizing the total time required to finish the race. An optimum is found at 60 s of race, and the results are shown in fig. 9.7, comparing fig. 9.7c and fig. 9.5c, the convergence for 60 s is much smoother than for the fastest path. In Table 9.3, it is possible to view the time of each polynomial segment shown in Figure 9.7b and the amount of time each gate is in view. In this way, it is possible to validate that all segments comply with the before mentioned requirement of 1.15 s. T6 and T7 are the segments corresponding to the dynamic gate and the jungle gate, even though the quadcopter spends a relatively high amount of time in these segments, the amount the actual gate is in view is relatively low. The reason is that these gates are much bigger than the normal ones and in order to reach these gates large maneuvers are required, as shown in Figure 9.7a.

Table 9.3: Optimal time allocation for total time $t_m$=60s.

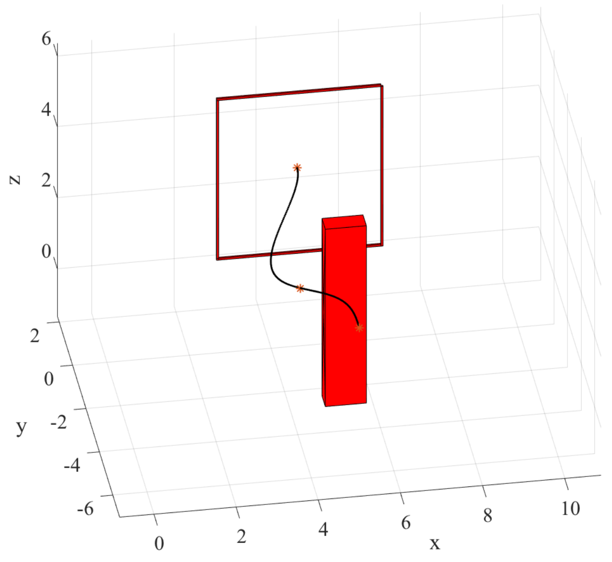
| **Segment** | **Time segment [s]** | **Gate in view [s]** |
|---|---|---|
| T1 | 6.88 | 3.24 |
| T2 | 4.55 | 1.71 |
| T3 | 5.66 | 2.69 |
| T4 | 5.17 | 3.63 |
| T5 | 3.49 | 2.43 |
| T6 | 10.55 | 2.87 |
| T7 | 12.39 | 2.02 |
| T8 (landing) | 11.31 | 2.57 |

From Table 9.3, it is also possible to validate that the expected drift won't be above the allowed one, except for gate 7, which requires some extra verification steps. From Figure 9.7d and the range calculated in Table 9.2, it is possible to calculate that the time it takes to reach the gate from the minimum distance at which is possible to view the entire gate is 2 s. From Table 9.3, the time, the gate is in view, is 2.02 s, which means that the drone doesn't have an update on its current position for 9.24, 8.37s (gate not in view) added to 0.87s, the time it takes for 3 frames to have a minimum reliability displayed in Table 9.1. Note that the time between the moment when the dynamic gate is last located and RAIDER goes through it is 2 s, meaning that RAIDER will always beat the dynamic gate to the safest point in the quadrant of Figure 9.4 for rotational speeds up to 3.93 rad/s. It is important to keep in mind that all computations are performed on the worst case scenario.

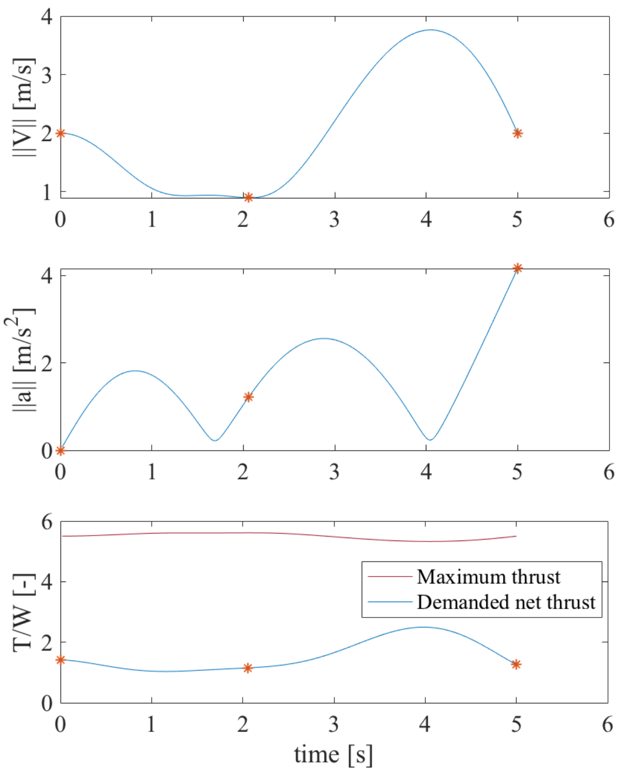
In Chapter 6, it is presented the equation to compute the dynamic thrust, we used this equation as the upper limit for the maximum T/W available, which is 5.5. The required thrust is well below the limit, since the maximum required T/W is 2. In this way it is also verified that the path doesn't require excessive accelerations. A full validation of the controllability of this path follows in Chapter 10.

### 9.5.2. Close-Range Obstacle Avoidance

This test solves the path re-planning problem in which an unexpected column-shaped obstacle is encountered at close range and shows proof of concept of the obstacle avoidance strategy suggested in Section 9.3. It is assumed that the obstacle is detected at a distance of 2 meters, which corresponds to 50% of the range of our depth camera. At the moment in which the obstacle is detected, the drone is taken to be travelling at 2 m/s in the direction of the obstacle and with 0 acceleration along the same axis. A new waypoint is placed to the left of the obstacle at two times the width of the drone and the path through the original waypoint is updated. After successfully avoiding the obstacle, the simulation ends with the drone returning through the original initial velocity. The path planning routine succeeds in generating a smooth and natural path around the obstacle. As shown in Figure 9.8b, the maneuvre demands reasonable accelerations that are well below RAIDER's capabilities as discussed in Chapter 6.



(a) The figure shows a 3D trajectory for obstacle avoidance.



(b) The figure shows from top to bottom:the norm of velocity, norm of acceleration and T/W for the obstacle avoidance example.

Figure 9.8: The figures show the results of the nominal path. The 3D trajectory is displayed on the top left, and the total velocity and acceleration on the right.

## 9.6. Runtime estimation

It is important to verify that the strategy can run on the processor available. In order to estimate the run-time on the Pocket Beagle, we considered the value given by BRY in PAPER by applying a conversion for the number of segments. He reported that his algorithm takes 0.18 ms on a 2GHz dual core Intel Atom processor for a 3 segment polynomial optimization.

## 9.7. Future Development

The programming language that has been used to simulate this algorithms is Matlab because it is convenient to make proof of concept in this language. For further development, we recommend to use C or C++ because they are compiled languages, which means that the throughput time would be significantly lower. One of the main reasons is that the variables created are not optimized for memory usage, which does not happen in compiled languages.

# 10 Stability and control

In this chapter, RAIDER's stability is determined and its control subsystem is designed. This allows it to attain stable flight that accurately follows the planned path.
As quadcopter dynamics are highly non-linear, solutions have to be found that tackle this problem. This will be done in Section 10.1, where non-linear and linear approaches are discussed in Section 10.1.1 and Section 10.1.2 respectively. Finally, the design of the control system itself will be discussed in Section 10.2. This includes modeling the dynamics of the quadrotor in Section 10.2.1, designing the controller in Section 10.2.3, tuning the controller in Section 10.2.4 and discussing the results in Section 10.2.5.

## 10.1. Control Solutions

As mentioned above, there are several techniques to create the control system for the quadcopter. These will be discussed below. They are divided into non-linear and linear control solutions.

### 10.1.1. Non-Linear Control

#### Non-linear Dynamic Inversion

The first possible method is Non-linear Dynamic Inversion (NDI). It is a control method that does not linearize around a fixed point, instead, it cancels plant dynamics non-linearities by inverting the model and predicting the state. This method can be very accurate, but only if an accurate model of the quadrotor dynamics is available. [50]
As the model can never fully represent real life, inaccuracies are inevitable and robustness can be questionable. Then there is the fact that developing an accurate model can be expensive or not possible at all.[1,2][50]

#### Incremental Non-linear Dynamic Inversion

Another method of tackling the non-linearity problem is by means of Incremental Non-linear Dynamic Inversion (INDI). This variation of NDI focuses on reducing the dependency on an accurate model. This is done by using sensor measurements instead of a model for the largest part of state predicting[50] .
The challenges that are asociated with INDI are the fact that the sensor measurements are never 100% accurate and the fact that an inverted actuator model is still necessary. As this method is not highly dependent on the dynamic model, this control system is more robust and can take un-modeled dynamical disturbances, such as gusts, into account[50] .

#### Backstepping

Backstepping control is a recursive stabilizing control solution working from inside out.[44] It breaks a non-linear system up into smaller subsystems. To those systems the Lyapunov function is applied to obtain the control law.[63]

#### Sliding Mode

Sliding mode control is a way to design a control system that tackles the non-linearity by implementing a discontinuous control function. The function features a specific feedback controller for the given input. [51] The discontinuous function is time independent and consists of continuous feedback functions. Furthermore, a sliding function is used, such that the appropriate control feedback function can be selected.

### 10.1.2. Linear Control

#### Proportional Integral Derivative Controller

As linear approach to the control problem, the proportional integral derivative (PID) controller is a possible solution. As a PID controller is a linear controller, it will only perform properly when the drone operates close to an equilibrium point where the dynamics of the drone remain approximately linear. The PID controller together with the plant form

[1] `https://www.tudelft.nl/technology-transfer/development-innovation/research-exhibition-projects/dynamic-inversion-control/` [Visited at June 22 2018]
[2] `http://www.diva-portal.org/smash/get/diva2:18843/FULLTEXT01.pdfDec` [Visited at June 22 2018]

a negative feedback loop. The input state is compared with the measured state and the error is fed into the PID controller. The PID controller then tries to bring this error to zero by outputting proper control inputs to the plant. [3] The PID controller transforms the error to the control input by applying Equation (10.1). This equation consists of three terms, the proportional term, the integration term and the derivative term. The controller can be tuned to obtain a desired response by changing the $K_P$, $K_I$, or $K_D$ values. $K_P$ being the proportional gain, $K_I$ being the integration gain and $K_D$ being the derivative gain.

$$u = K_p e(t) + K_I \int_{t_0}^{t} e(t)\,dt + K_D \frac{d}{dt} e(t) \tag{10.1}$$

The proportional term corrects the input for the error at the current time, the integral for the past time and the derivative for the rate of change of the error.[4] An example layout of a basic PID controller can be seen in Figure 10.2, it is important to remind that the controller is positioned in front of the System which models all the dynamics of the quadcopter.[31]

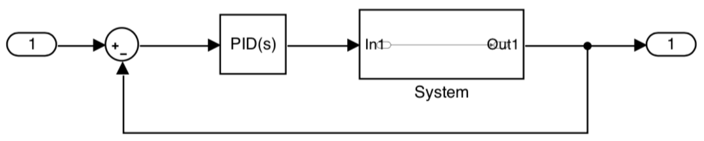


Figure 10.1: Basic PID Controller System Layout

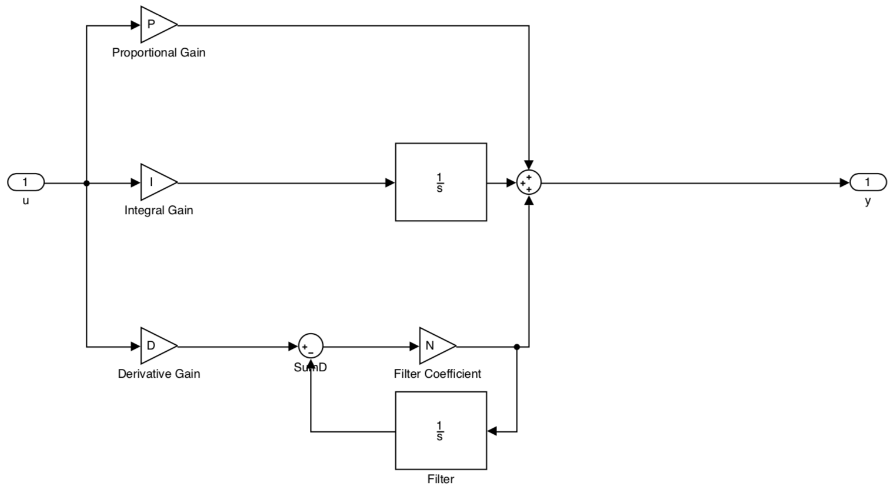


Figure 10.2: Basic PID Controller Block Layout

The advantage of a PID controller, is the fact that it is widely used, and can be implemented with basic mathematical knowledge. As long as there is understanding of the proportional, integral and derivative terms. Also, the controller can be tuned (changing the K values) with limited knowledge and engineering intuition.

#### Proportional Integral Derivative Gain Scheduling

As an extension to the PID controller, one can apply PID gain scheduling. For a PID controller there is only a certain range in which it performs well. In order to have a wide working range, one can design several PID controllers. Those controllers are tuned, each for their own consecutive range. Then, according to the current state, a switching controller switches to the corresponding PID controller. This transition can be made smooth by for example interpolating the gain values of the consecutive PID controllers.

## 10.2. Controller Design

After researching the different control solutions, it was decided to implement a PID controller. This was chosen for its popularity in drone systems and its high performance. Additionally, since there is only a limited amount

[3] `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID` [Visited at June 22 2018]
[4] `http://wiki.theuavguide.com/wiki/PID_Controller` [Visited at June 22 2018]

of time and knowledge in this project, a PID controller was deemed to be the most feasible to design within the constraints. In order to increase the versatility and full flight envelope performance of the controller, the possibility of gain scheduling in further design will be discussed.

### 10.2.1. Flight Dynamics

In order to create a representative control system, it is useful to model the quadcopter's flight dynamics first. The reference frame used is the right-handed body reference frame, as can be seen in Figure 10.3. It is assumed that the origin coincides with the center of gravity of the quadcopter, the x-axis goes through the front of the drone and the z-axis points downwards, perpendicular to the drone. The rotors are labeled from 1 through 4 and given a rotating direction. Furthermore, a and b are the location of each motor with respect to the center of gravity. As the path is given in earth reference frame coordinates, the earth coordinate system can also be seen in Figure 10.3[50].

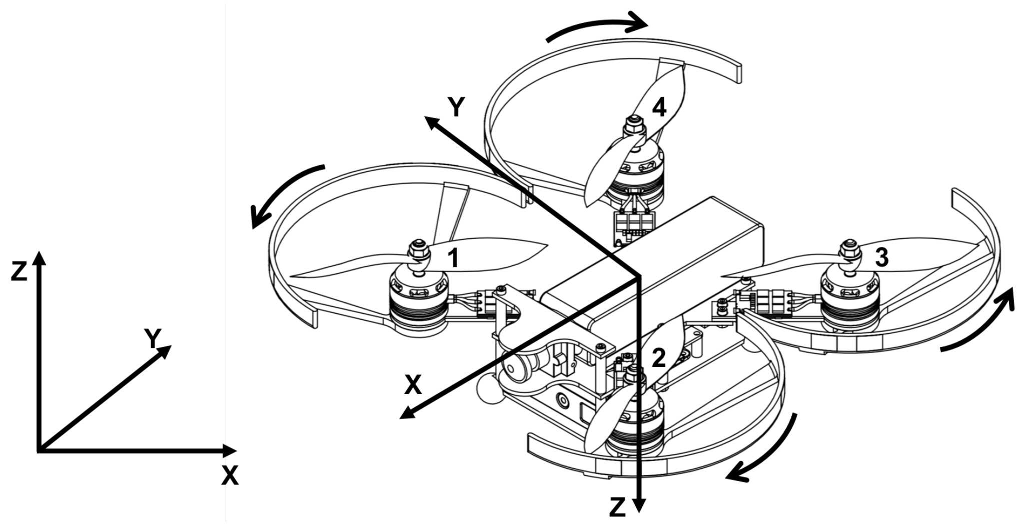


Figure 10.3: Earth Reference Frame (left), Body Reference Frame on Drone (right)

Now by applying basic dynamics, it is possible to find the total moment on the drone. This is illustrated in Equation (10.2)[50] , where $I_v$ is the moment of inertia, $\dot{\Omega}$ is the angular acceleration, and $\Omega$ is the angular velocity of the quadrotor[50].

$$M = I_v \cdot \dot{\Omega} + \Omega \times I_v \Omega \tag{10.2}$$

It is assumed that there are three moments acting on the quadrotor: The control moment $M_C$, which is the moment generated by the uneven lift of the rotors, the aerodynamic moment $M_A$, and the moment induced on the drone by the rotors due to the gyroscopic effect and accelerations of the rotors $M_R$. Substituting these moments for $M$ in Equation (10.2) results in Equation (10.3)[50].

$$M_C - M_R + M_A = I_v \cdot \dot{\Omega} + \Omega \times I_v \Omega \tag{10.3}$$

In order to simplify our model, $M_A$ is considered negligible. Furthermore, $M_C$ is given by Equation (10.4)[50] . In this equation, the torque, $\tau$, is generated by a rotor due to its aerodynamic drag is modeled to be $\tau = k_2 \cdot \omega^2$. Similarly, the thrust, $T$, generated by the rotors is modeled to be $T = k_1 \cdot \omega^2$.

$$M_C = \begin{bmatrix} a \cdot k_1 \cdot \left(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2\right) \\ b \cdot k_1 \cdot \left(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2\right) \\ k_2 \cdot \left(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2\right) \end{bmatrix} \tag{10.4}$$

Furthermore, $M_R$ is given by Equation (10.5), which can be expanded into Equation (10.6). In order to simplify the dynamic model, $M_R$ is also neglected. This is a reasonable assumption to make in this stage of the design.

$$M_R = \sum_{i=1}^{4} M_{Ri} = \sum_{i=1}^{4} I_r \cdot \dot{\omega} + \omega \times I_r \omega [50] \tag{10.5}$$

$$M_R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ I_{R_{zz}} & -I_{R_{zz}} & I_{R_{zz}} & -I_{R_{zz}} \end{bmatrix} \cdot \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \\ \dot{\omega}_4 \end{bmatrix} + \begin{bmatrix} I_{R_{zz}}\Omega_y & -I_{R_{zz}}\Omega_y & I_{R_{zz}}\Omega_y & -I_{R_{zz}}\Omega_y \\ -I_{R_{zz}}\Omega_x & I_{R_{zz}}\Omega_x & -I_{R_{zz}}\Omega_x & I_{R_{zz}}\Omega_x \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} [50] \tag{10.6}$$

### 10.2.2. Actuator Model

In order to take the effects of the non-zero spin-up time of the rotors into account, a first order transfer function given in Equation (10.7) is used to model the spin-up time of the rotors. In this equation, $\tau$ is the time constant and can be set to a value that approximates the response of a general motor applied in drones, in this control system iteration $\tau$ was set as 0.05.

$$H(s) = \frac{1}{\tau s + 1} \tag{10.7}$$

### 10.2.3. Controller

Now that the dynamics of the quad copter are defined and the controller type is chosen, the controller, or plant, model can be made and tuned to simulate the response of the system. The control system consists of four feedback loops. The inner most loop is a loop that controls the angular velocity of the drone. The loop around the inner loop controls the attitude of the drone. The loop around that controls the translational velocity of the drone, and the outer most loop controls the position of the drone in space. These different loops have been identified from most outer to inner loop by Level 1, Level 2, Level 3, and Top Level respectively, which can be seen in Figures 10.4 to 10.7. Every level will be discussed in detail below.

Level 3 consists of three main parts. Firstly, the PID controller transforms the error between the current angular velocity and the reference angular velocity value to a desired angular acceleration output. This desired angular acceleration output is then transformed to desired rotor speeds by a series of matrix multiplications. For determining the rotor speeds the desired total thrust should also be determined. This desired thrust is determined in the outer loops. The desired rotor speeds are then fed into the actuator model which models the response of the rotors spinning up. This response is then fed into the dynamics of the drone which outputs the angular accelerations about each axis. This is then integrated to obtain the angular velocity and fed back into the PID controller.
After tuning the PID controller in Level 3, the output will be kept close to the input reference angular velocity by the negative feedback of the PID controller. Level 3's block diagram can be seen in Figure 10.4.

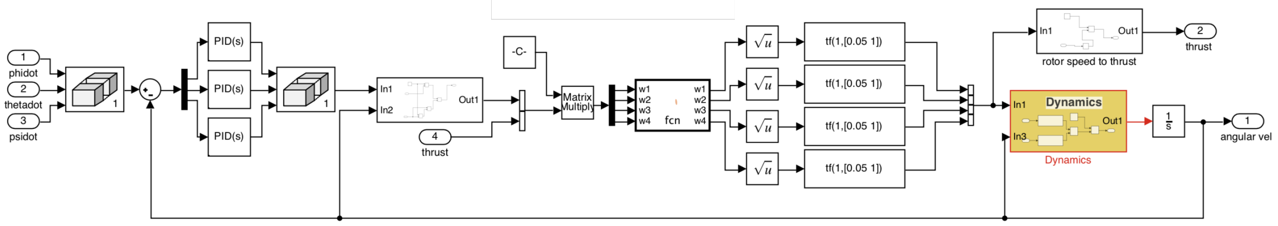


Figure 10.4: Level 3 of the PID controller

As Level 3 inputs and outputs the desired and simulated angular velocity, Level 3 can be seen as the angular velocity controller. By placing a PID control loop around the angular velocity controller, these two loops together control the angular position. This is the Level 2 control loop. It takes in the desired angular position and outputs the simulated angular position response. The block diagram of Level 2 is depicted in Figure 10.5.

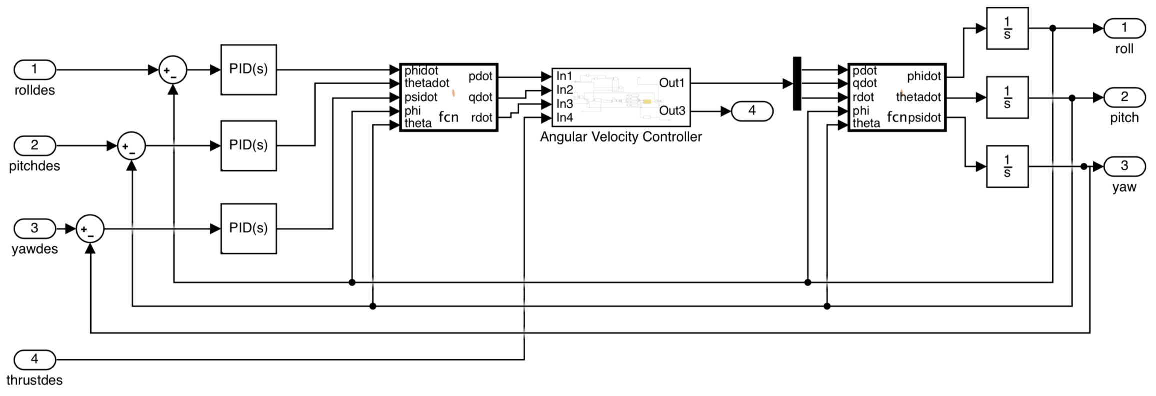


Figure 10.5: Level 2 of the PID Controller

Level 1 is the translational velocity controller. Requires the desired translational velocity to be calculated. It calculates the error between the translational and desired velocity and feeds it to the PID controllers. The PID controllers

transform this error in translational velocity to a desired attitude that is fed into the attitude controller (Level 2). This transformation of the PID controller is based on the assumption that the pitch and roll angle are proportional to the accelerations in the x and y direction in the earth frame respectively. This assumption is the result of linearizing the drone dynamics about the hovering state. After the desired attitude is fed through the attitude controller and the attitude response is simulated, the attitude of the drone together with the thrust of the drone can be converted to accelerations in the x, y, z directions of the earth frame. These are then integrated and the velocities in x, y, and z directions in the earth frame are obtained. These are then fed back into the PID controller. The acceleration in the z direction is not regulated by the attitude of the drone but solely the thrust. The PID controller for the velocity in the z direction thus directly outputs the desired thrust which is directly fed into Level 3.
Level 1, can be seen in fig. 10.6.

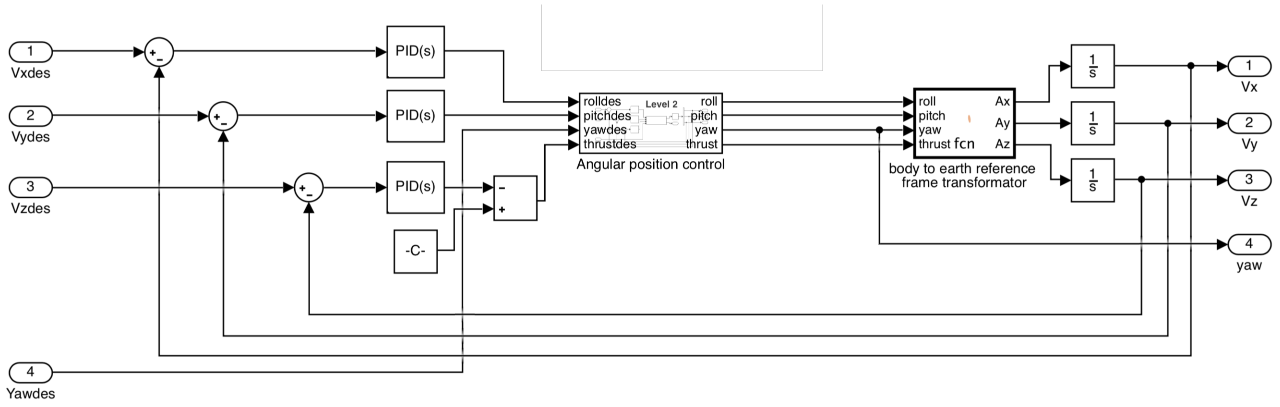


Figure 10.6: Level 1 of the PID Controller

The Top Level requires the desired x, y, and z position and the desired yaw angle. The PID controllers need the error in position as an input and transform this to a desired translational velocity, which is their output. This reference velocity is then fed into the translational velocity controller (Level 2) which outputs the translational velocity response. This is then integrated to obtain the position response in earth reference frame. The desired yaw angle is fed directly into the attitude controller (Level 2). The top level can be seen in Figure 10.7.

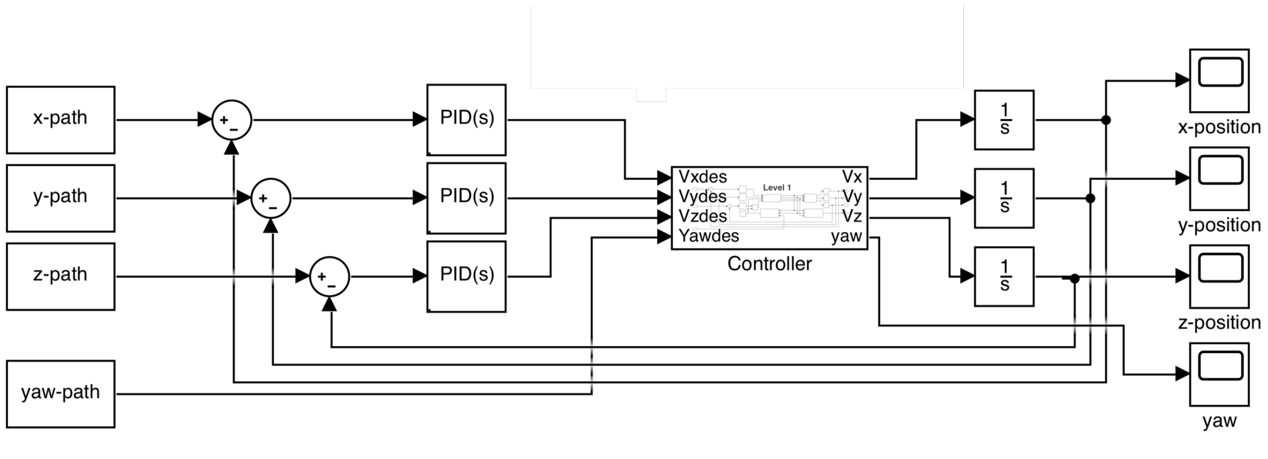


Figure 10.7: Top Level of the PID Controller

### 10.2.4. PID Tuning

After designing the PID controller, the PID loops have to be tuned. Tuning the PID loop means changing the K values until a desired response is achieved. As no well defined approach to PID tuning exists, PID tuning highly relies on experience and engineering intuition. Although some tricks exist in case the designer has neither of them, their effectiveness is highly dependent on the system.
As a general approach in this design, the PID controllers were tuned from inside out. Meaning that the inner loop PID was tuned first. If an appropriate response for the to be tuned loop is achieved, one advances to the next outer loop. Continuing like this until all the PID modules are tuned and the response of the final system is as desired. The result of the tuned PID values for this system can be seen in Table 10.1.

Table 10.1: PID values for the PID controllers used.

| **x** | **Top Level** | **Level 1** | **Level 2** | **Level 3** |
|---|---|---|---|---|
| $K_P$ | 1.71 | 0.7 | 15 | 30 |
| $K_I$ | 0 | 0.001 | 0.1 | 0 |
| $K_D$ | 0 | 0.09 | 0.4 | 1.4 |
| **y** | **Top Level** | **Level 1** | **Level 2** | **Level 3** |
| $K_P$ | 1.71 | 0.7 | 20 | 30 |
| $K_I$ | 0 | 0.001 | 0.1 | 0 |
| $K_D$ | 0 | 0.025 | 0.2 | 1.2 |
| **z** | **Top Level** | **Level 1** | **Level 2** | **Level 3** |
| $K_P$ | 2.1 | 10 | 15 | 20 |
| $K_I$ | 0 | 0.2 | 0.1 | 0 |
| $K_D$ | 0 | 3.5 | 0.2 | 1.2 |

### 10.2.5. Results

After tuning the controller, it is possible to plot the response of the controller due to a given input. In order to demonstrate the effectiveness of the controller, the controller was given a step input of 1m, to which the response can be seen in fig. 10.8. Note that the spike in the beginning of the response in z-direction is due to gravity acting on the quadcopter starting from time 0.

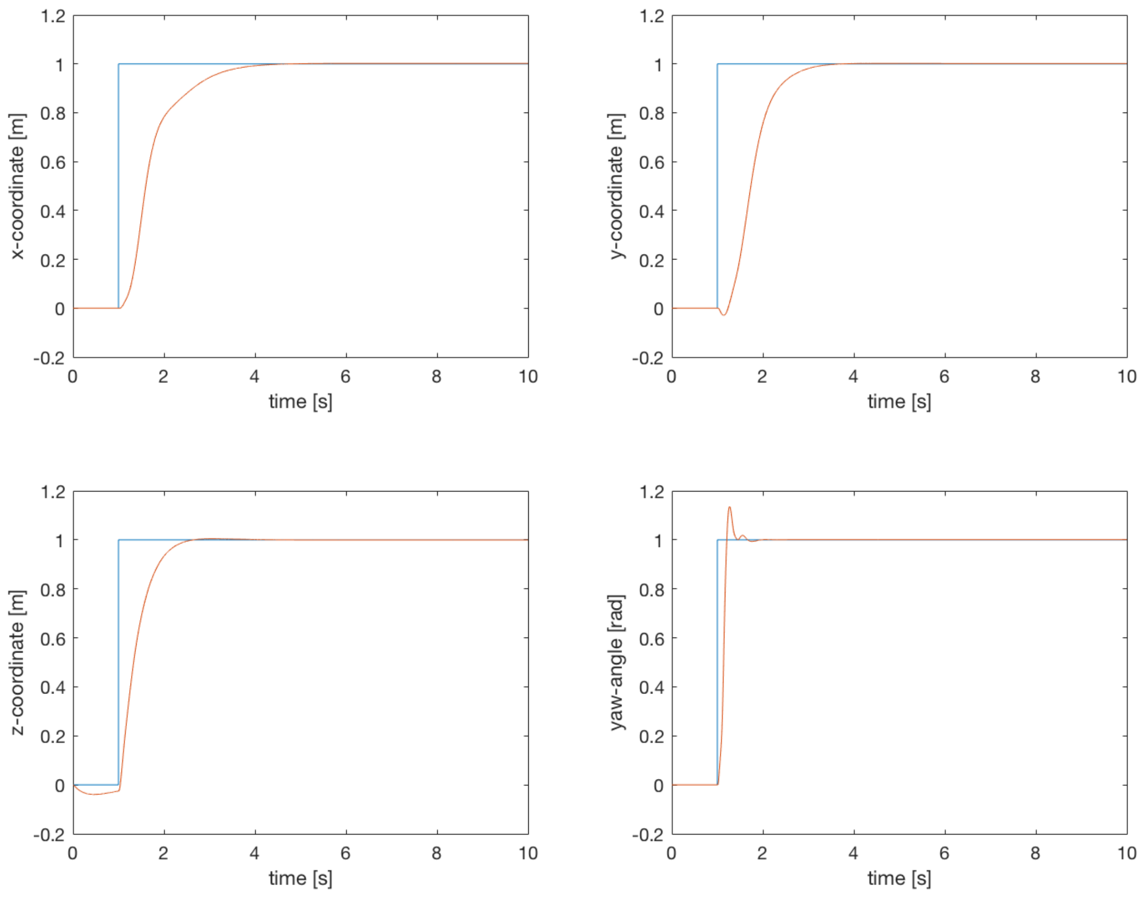


Figure 10.8: Response to a step input of 1 in yaw angle [rad] and x, y, and z direction displacement [m]

After checking the controller with step inputs on every level, it is possible to feed the controller a path and check its response. This has been done in Figure 10.9. It consists of a pre-planned path delivered by the path planning subsystem, assuming the location of every gate, thus the complete path is known. As can be seen in the graph, both the path and the flown path are very close to each other. This is due to the fact that the simulation of the drone dynamics is very simplified and that the sampling rate of the values obtained from the polynomial of the given path was about 0.001, corresponding to about 1000Hz. The controller has been tested to give as accurate results up to a frequency of 300Hz. When the controller is implemented in the quadcopter, the sampling rate will depend on the controller board, sensors and power system. If a lower frequency is desired, the controller might have to be re-tuned, or even PID gain scheduling might be needed in order to not exceed the bounds of the controller.

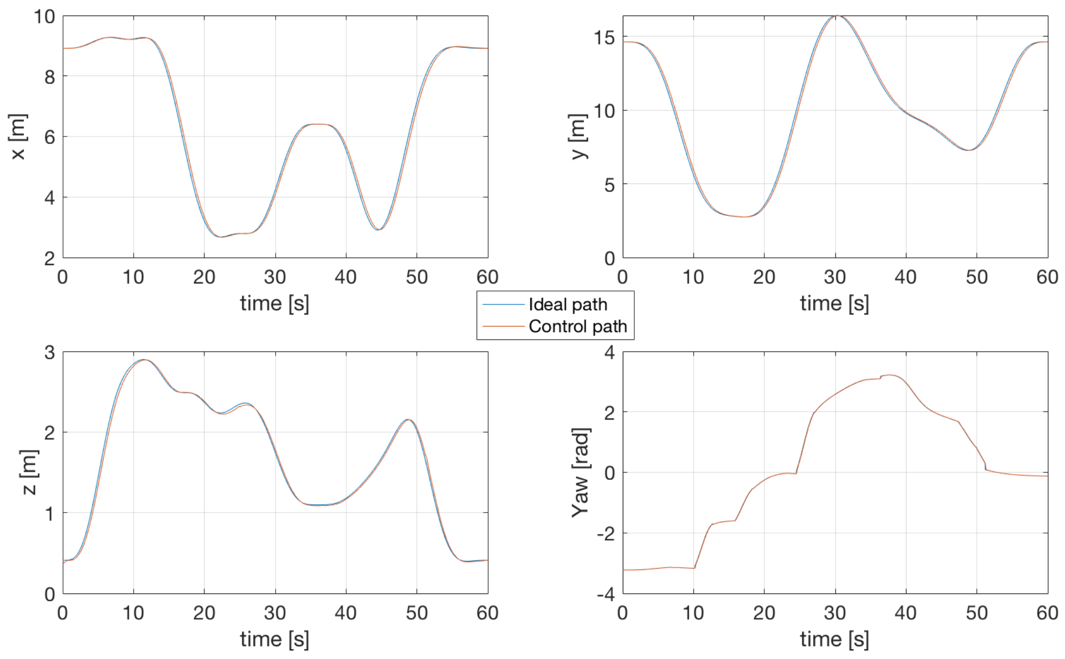


Figure 10.9: Response of planned path in yaw angle [rad] and x,y and z direction displacement [m]

## Verification

As a verification measure, each level of the controller has been subjected to unit tests, in order to check if the desired output is given for a certain input. Also, each level independently has been given step inputs in order to verify its effectiveness. If the response was unstable or not close enough to the input variable, re-tuning was performed.

## Validation

The best way to validate this subsystem would be to implement it on a physical quadcopter and check whether the response is as expected. Another possibility would be to feed it to a validated quadcopter model, and check whether the response is as desired.

## Gain Scheduling

The path that is planned by the path planning subsystem takes approximately 60 seconds. While flying this path, the speeds and angles are kept relatively low. This way the inputs still stay within bounds of the controller, without rendering its performance unacceptable. Therefore, gain scheduling is not deemed necessary in this iteration. However, if the quadcopter has to attain higher speeds and perform more extreme manoeuvres, or fly paths at a frequency, this controller's performance will no longer be sufficient. The solution for this will be gain scheduling. This implies having the controller linearized and tuned at several working points. Then, according to the current state of the drone, switch between the gains the controller is tuned for, as was explained in Section 10.1.2.

# 11 Data handling and Integration

In order for RAIDER to work correctly, all data has to be delivered to the right component at the right time. How this will be done is explained in this chapter. First, in Section 11.1, the data flow will be explained. Then, in Section 11.2, each hardware connection will be determined and explained. After which, in Section 11.3, the team will determine whether or not processing boards can run the algorithms at sufficient speed. After that, in Section 11.4, the different flight modes in which the drone can operate will be discussed. And finally, in Section 11.5, the Verification and Validation procedures for all of the models that were used will be explained.

## 11.1. Data Handling

The general data flow for RAIDER can be seen in Figure 11.1, this is a combination of the data handling and the communicational flow diagrams. Here the in- and outputs of each computer board can be seen. The Raspberry Pi Zero W does all of the image processing and it uses its WiFi connection to send telemetry data to the ground station. While the Raspberry Pi Zero W is calculating the drone's position with respect to the gate, the Pocket Beagle uses IMU and Ultrasound measurements to approximate it. Both the path planning and the fight controller algorithms are run on the Pocket Beagle. This outputs the thrust commands to each of the ESCs. The Custom Power Distribution Board (Custom PDB) that can be seen in Figure 11.1 is merely a way to increase the connectivity of the Pocket Beagle and the Raspberry Pi Zero W. They will be connected with the GPIO headers and all the PDB does is pass the information on to the relevant computer board.

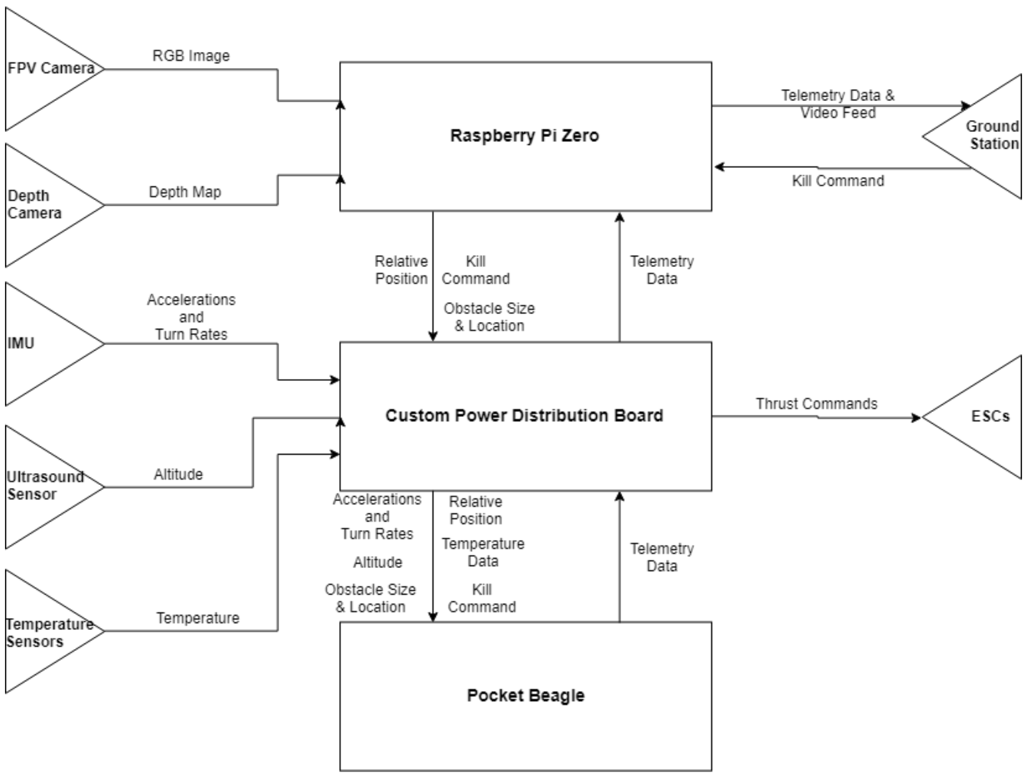


Figure 11.1: General Data Handling Diagram

As stated above, Figure 11.1 is very general. Figure 11.2 gives a more detailed overview of communication inside of the drone. It shows which subsystems will be run on which processing units on top of what exactly is sent by and to each.The incoming images go to the Raspberry Pi Zero W's main processor, the ARM 11, which runs the gate detection. This then calculates the the relative position of RAIDER with respect to the gate at a relatively low frequency. On the Pocket Beagle's main processor, the ARM Cortex-A8, this will be fused with altitude and high-frequency IMU measurements, using an Unscented Kalman Filter, as explained in Section 11.1. The result of this is a high-frequency estimate of the pose, velocity, and acceleration of the drone, which is then sent to the WiFi transceiver on the Raspberry Pi Zero W, and the path planning and flight controller on the Pocket Beagle. In order to make the most of the processing power on the Pocket Beagle, the flight controller was moved to the ARM Cortex-M3. This was shown to be possible by [25], and [40]. As shown in , the sensor fusion algorithm only uses a small percentage of the processing power of the Pocket Beagle. The rest of this will be used for path planning. There are 2 reasons for putting the sensor fusion algorithm on the Pocket Beagle, together with the path planning: Firstly, it reduces the needed capacity of the connection between the 2 computer boards. Secondly, the Pocket Beagle has more computational power so it can handle the load of running both sensor fusion and path planning on the same processor much better.

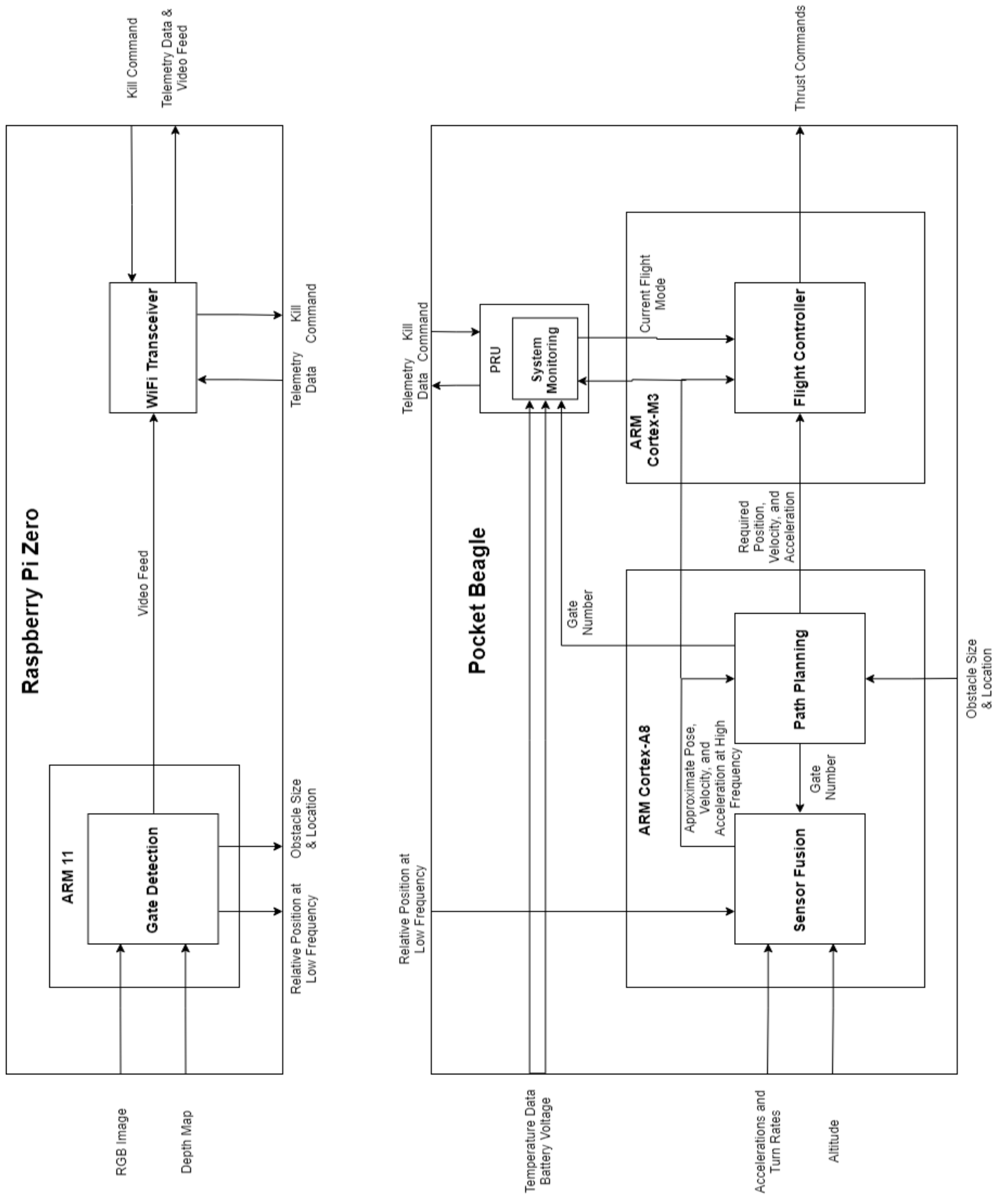


Figure 11.2: Detailed Data Handling Diagram

## 11.2. Hardware Connections

The reason a the team decided to use a Raspberry Pi Zero W instead of a second Pocket Beagle are the interfaces. As can be seen in Figure 11.3, the RGB camera needs a CSI camera interface. This is not present on the Pocket Beagle. The depth camera only needs a micro USB cable, which can also be directly connected to the Raspberry Pi Zero W. The Pocket Beagle, only has a micro USB port and 2 2x18 pin GPIO headers for connectivity. These GPIO headers are what will be used to connect it to everything. As stated above, the power distribution board will be connected to both computer boards using this.

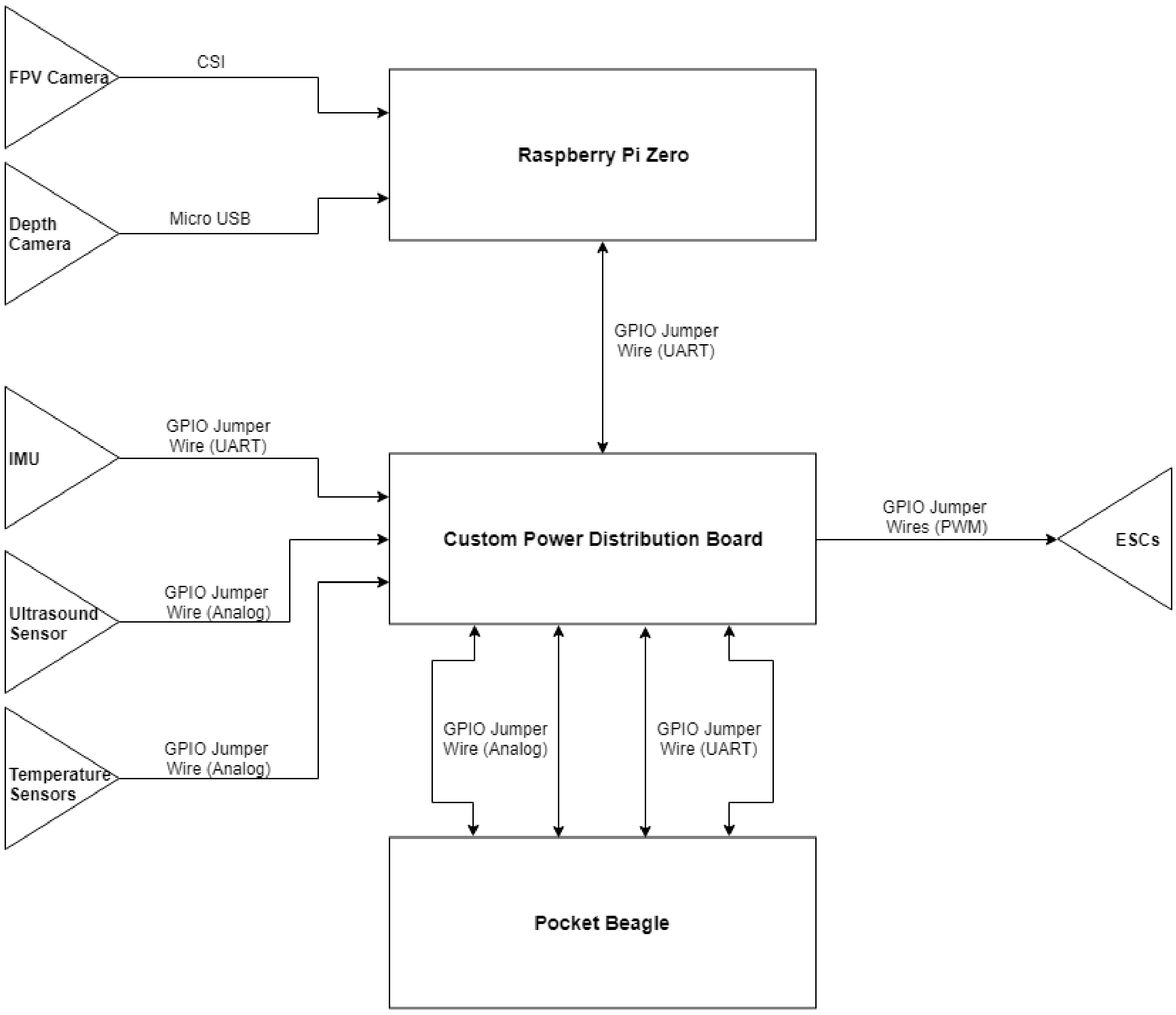


Figure 11.3: Hardware Diagram

### 11.2.1. Connection Bandwidths

As shown in Figure 11.3, there are 4 connections between the PDB and the Pocket Beagle. These connections are there because the PDB acts as an improved input and output interface for the Pocket Beagle. The 2 analog connections provide a direct connection between the Ultrasound and Temperature Sensors, and the Pocket Beagle. The first UART connection connects the IMU and the Pocket Beagle and the second one connects the 2 computer boards. Since the computer boards are directly connected to their respective sensors and actuators (with only a slight extension through the PDB), no bandwidth issues are expected in these connections.

The UART connections between the two computer boards each have a maximum baud rate of 115.2 kilobits per second in both up and download[1,2]. The messages that are sent that are used for UAV communication use the ASCII

[1] *https://github.com/beagleboard/pocketbeagle/wiki/System-Reference-Manual* [Visited at 11 June 2018]

[2] *https://www.argon40.com/resources/a-comprehensive-guide-on-raspberry-pi-gpio-pinout/* [Visited at 11 June 2018]

format[3]. An example message format for each type of message can be found in Appendix A. The resulting baud rate for each of the message types can be found Table 11.1.

Table 11.1: Baud Rate Needed for each Message Type

| | Message Size [Characters] | Message Size [bits] | Frequency [Hz] | Baud Rate [Kbps] |
|---|---|---|---|---|
| Exact Pose Message | 232 | 1856 | 10 | 18.6 |
| Kill Message | 124 | 992 | 10 | 9.9 |
| Obstacle Avoidance | 179 | 45 | 45 | 64.4 |
| Telemetry Message | 628 | 5024 | 10 | 50.2 |

The data in Table 11.1 results in a total needed capacity of 50.2 Kbps from the Pocket Beagle to the Raspberry Pi Zero W. The needed bandwidth from the Raspberry Pi Zero W to the Pocket Beagle is 92.9 Kbps. Both of these are significantly lower than the 115.2 Kbps limit. A such, no bandwidth issues are expected.

The Telemetry and Kill Messages also have to be sent to and from the ground station, respectively. This requires a data rate of 60.1Kbps. On top of this, the video feed from the RGB camera will be be sent to the ground station. At 30 Hz, 720p video streaming takes up to 4.5 Mbps, adding up to a total needed bandwidth of 4.5Mbps. This is, however, only a fraction of the data rate the WiFi module aboard the Raspberry Pi can handle, which is 72 Mbps[4].

### 11.2.2. Cable Weight

Now that all of the interfaces are defined, the total cable weight can be calculated. The result of this can be seen in Table 11.2. Note that the connection between the Pocket Beagle and the ESCs is not in here because the cables for this are included in the ESCs' weights. Also note that there are no cables between the Pocket Beagle and the Raspberry Pi Zero W since all communication runs directly through the PDB.

Table 11.2: Cable Weights

| **From** | **To** | **Type** | **Length [cm]** | **Weight [g]** |
|---|---|---|---|---|
| FPV Camera | Raspberry Pi Zero W | MIPI CSI-2 Cable | 10 | 0.8 |
| Depth Camera | Raspberry Pi Zero W | Micro USB Cable | 20 | 7 |
| IMU | PDB | GPIO Jumper Wires (UART) | 10 | 0.8 |
| Ultrasound Sensor | PDB | GPIO Jumper Wires (Analog) | 10 | 0.4 |
| Temperature Sensors | PDB | GPIO Jumper Wires (Analog) | 10 | 0.4 |

The weight values for the GPIO Jumper Wires were calculated using the average weight per cm of a group of jumper wires ($0.019\frac{g}{cm}$[5]) and the fact that a UART connection needs 4 cables and an analog one only needs 2. The weight of the MIPI CSI-2 Cable was calculated using an average weight of $0.075\frac{g}{cm}$[6]. The resulting total cable weight is 9.4 grams.

## 11.3. Hardware Capabilities

In order to make sure that all of the programs that were designed run at acceptable frame rates on the available hardware, the computational power of each board was compared to that needed by the algorithms. First the Power of the Raspberry Pi Zero W will be checked, after that the same will be done for the Pocket Beagle.

#### Raspberry Pi Zero W

According to [9], the Raspberry Pi Zero W has 0.319 $GFLOP \cdot s^{-1}$ of computational power. Most of this computational power will be taken up by the DeCNN and its RDP based corner detector, which need approximately 82.9 $MFLOP$ per iteration. For the RDP algorithm that will run to detect corners in the images from the ToF camera, this number is only 732.5 $KFLOP$, because the number of pixels is much smaller. Since the maximum frame rate of the Pico Flexx is 45 Hz, the RDP algorithm will also run at this speed. This results in a total of 56.25 $MFLOP \cdot s^{-1}$, leaving 286 $MFLOP \cdot s^{-1}$ for the CNN, which will then run at 3.4Hz. Although the 3.4Hz is constraining for the maximum speed at

[3] `http://www.barnardmicrosystems.com/media/presentations/IET_UAV_C2_Barnard_DEC_2007.pdf` [Visited at 11 June 2018]

[4] `https://www.pocketables.com/2017/03/raspberry-pi-zero-w-wifi-performance.html` [Visited at 11 June 2018]

[5] `https://www.amazon.co.uk/Assorted-Multicolored-Flexible-Solderless-Breadboard/dp/B0087ZRVES/ref=pd_bxgy_ce_img_y/277-4324215-8292916` [Visited at 11 June 2018]

[6] `https://www.adafruit.com/product/1647` [Visited at 11 June 2018]

which the track can be traversed, it does not completely stop this method from being used, since the RDP algorithm becomes more important when it gets within 4 meters. The number of $FLOPS$ that is used for each algorithm is explained in Chapter 8.

#### Pocket Beagle

The total computational power of the Pocket Beagle is approximately 20% higher than that of the Raspberry Pi Zero W[7] , resulting in a total power of 0.383 $GFLOP \cdot s^{-1}$. As stated in Section 11.1, the chosen Sensor Fusion algorithm is expected to take 12.7% of the Pocket Beagle.

This leaves 87.3% of the Pocket Beagle's power. As demonstrated in Section 9.6, the expected run time of the path planning algorithm is 0.18 ms on a 2GHz dual core Intel Atom processor for a 3 segment polynomial optimization. We converted this number to 11 segments by $0.18 \cdot \frac{11}{3}$, so the run-time for 11 segments is 0.66 ms. Since this processor has 4 times as many threads and runs at twice the frequency as the Pocket Beagle's Cortex-A8, this value is multiplied by 8 in order to roughly scale for the difference between the processors. This results in a computational time of 5.28ms if the full power of the board was available. Since only 87% is left, the 5.28ms has to be divided by it. Resulting in a total computational time of 6.07ms, or an update rate of 165Hz. Since the minimal frequency at which the path planning has to run is determined by the obstacle avoidance, and the Pico Flexx only updates at 45Hz, this is more than enough. The Pocket Beagle can thus run everything that is needed.

## 11.4. Flight Modes

In order to make sure that RAIDER is safe, 3 flight modes were implemented: Normal Flight, Safe Landing and Emergency Shutdown. These are elaborated upon in Section 11.4.1, Section 11.4.2, and Section 11.4.3 respectively.

### 11.4.1. Normal Flight

Normal flight is, as is to be expected, the flight mode the drone will be in most of the time. The drone will be flying through the race track, finding gates and determining its path between them.

### 11.4.2. Safe Landing

Safe landing is the mode that the drone will enter when an error, low battery voltage, a position outside of the track, an obstacle is close to drone or an overheating computer board are detected. RAIDER will then autonomously land in order to make sure that nothing is damaged in case the situations deteriorates. The drone will autonomously decide to enter this mode.

### 11.4.3. Emergency Shutdown

Emergency Shutdown is only to be used in the event of an imminent collision. The drone can only enter this mode when the command is given by the ground station. It shuts down the drone as quickly as possible while it is still in flight. This makes the drone fall out of the air, possibly damaging it or its surroundings. This is accepted as a risk, however, since it can no longer behave erratically and it immediately starts to slow down when this mode is activated. Due to the shutdown, the imminent collision will occur at a lower speed, lowering the risk of injury.

## 11.5. Model Verification and Validation

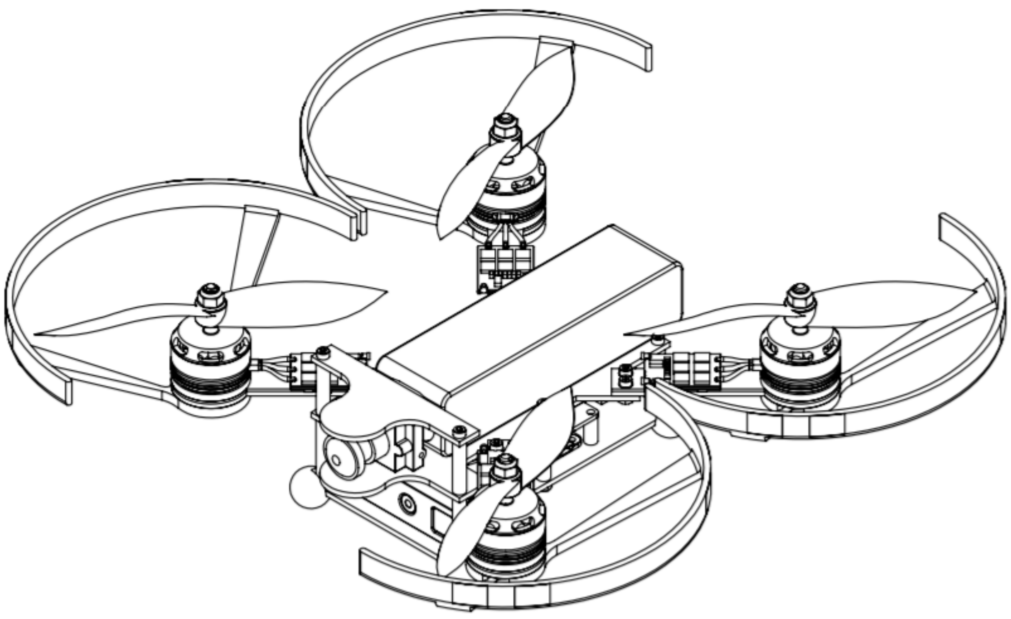
#### Connection Capabilities

The best way to validate the model in Section 11.2.1 is by testing it out. After buying the 2 computer boards and the PDB, the performance of the connection can be determined by seeing how much data can be moved through them. The same can be done for the WiFi connection between the Raspberry Pi Zero W and the ground station.As a unit

test for verification, the excel sheet that was used to calculate everything in Section 11.2.1 was checked by manually counting all of the characters in the Kill Message and comparing that to the result of the sheet. This also gave the amount of characters in the header, allowing for part of the other messages to be verified using this.

[7] `https://www.teachmemicro.com/raspberry-pi-zero-vs-pocketbeagle/` [Visited at 7 June 2018]

# III
# Further Development



12

# Final Design Overview

During the detailed design phase, the complete design has also been drawn in CAD-software. A benefit of a complete CAD assembly is the amount of information it can provide.
Using accurate CAD sketches, the mass of custom components can be very precisely estimated. Furthermore the mass of subcomponents such as bolts can better be taken into account, further improving the overall assessment of the design. A detailed breakdown of the cost and mass budgets is done in Section 12.1.
After that, exploded views of the arm and PCB assembly will be presented. These will be incorporated in the final exploded view of the total drone. All of these views are accompanied by a bill of materials and are shown in Section 12.2.
A short summary of the most important parameters of the drone can be found in Figure 12.1.

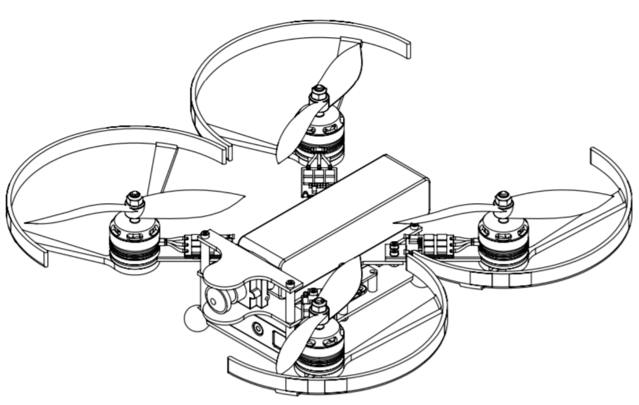


Figure 12.1: Isometric view of RAIDER

| | |
|---|---|
| **Name** | RAIDER |
| **Size(W x L x H)** | (291.70 x 289.20 x 82.91) |
| **Number of Motors** | 4 |
| **Battery Capacity** | 1550mAh |
| **Battery Configuration** | 6S |
| **Hover Flight Time** | 15.6min |
| **Full Thrust Flight Time** | 0.7min |
| **Maximum Motor Power** | 549W |
| **T/W** | 5.0 |
| **Safety Features** | - Propeller guards |
| | - IMU Damper |
| | - Carbon Frame |
| | - crumple zones |
| **Cameras** | - Connex ProSight RGB |
| | - Pico Flexx Depth |
| **Sensors** | - VectorNav VN-100 IMU |
| | - Ultrasonic height sensor |

## 12.1. Final resources

The driving resources are: the mass, power, computational resources, cost, and the size. The mass, the power and the cost are common parameters used for aerospace vehicles. The computational resources are particularly important for this project because the objective is to have an autonomous vehicle. The size is an important resource because it is directly linked to the amount of localization error the drone can sustain, while being able to make it through the race.

### 12.1.1. Mass budget

The cost and mass budgets are split in different sub categories to make them more readable. The structures will contain all the carbon components, their connections and all components needed for mitigating structural risks. The second category contains all the components related to the sensor fusion, gate detection and other computational components. The category also includes the dampers needed for the IMU. The last category contains all components related to the power and propulsion of the drone, including their mounting necessities.
The final mass breakdown is shown in Table 12.1. For custom designed components the masses shown were taken either directly from the CAD software. For the off the shelf components these were taken from data sheets obtained from manufactures.

Table 12.1: Detailed mass breakdown

| Category | Component | Material | Mass[gram] | Qt |
|---|---|---|---|---|
| **Structures** | Center Frame | T7000g | 23.69 | 1 |
| | Arm Frame | T7000g | 5.56 | 4 |
| | Arm Insert | 7075-T6 (SN) | 0.12 | 8 |

| | | | | |
|---|---|---|---|---|
| | DIN 912 M2.5 x 8 | Steel12.9 | 0.54 | 14 |
| | DIN 934 - M2.5 | Steel12.9 | 0.26 | 14 |
| | Proppeller Guard | ABS | 11.07 | 4 |
| | Motor Spacer | ABS | 1.77 | 4 |
| | Frame Insert | 7075-T6 (SN) | 0.09 | 8 |
| | Bottom Stand-Off | 7075-T6 (SN) | 1.54 | 6 |
| | Bottom Frame | T7000g | 19.27 | 1 |
| | Crumple Zone | Polyurethane | 0.01 | 4 |
| | Top Stand-Off | 7075-T6 (SN) | 1.21 | 2 |
| | Top Frame | T7000g | 4.66 | 1 |
| | **Total** | | **145.8** | |
| **Sensors and Computers** | VectorNav VN-100 | | 15.00 | 1 |
| | IMU Damper | Rubber | 0.15 | 2 |
| | IMU Damper2 | Rubber | 0.08 | 2 |
| | Motherboard | C-Glass Fiber | 15.12 | 1 |
| | Raspberry Pi Zero W | | 9.00 | 1 |
| | Pocket Beagle | | 15.00 | 1 |
| | PCB Stand-Off | ABS | 0.17 | 4 |
| | Pico Flexx | | 8.00 | 1 |
| | DIN 934 - M2.5 | Steel12.9 | 0.26 | 2 |
| | Additional Cables | | 10.00 | 1 |
| | XL-MaxSonar-EZ | | 6.10 | 1 |
| | RGB Camera | | 14.00 | 1 |
| | **Total** | | **93.88** | |
| **Power and Propulsion** | DIN 912 M3 x 8 | Steel12.9 | 0.87 | 16 |
| | XRotor Motor | | 31.00 | 4 |
| | Turnigy Multistar 30A Rev16 v3 ESC | | 15.80 | 4 |
| | Master Airscrew 5045 Propeller | | 5.90 | 4 |
| | Turnigy 1550mAh 6s | | 247.00 | 1 |
| | **Total** | | **474.92** | |
| **Total** | | | **714.80** | |

As stated before, all masses are taken from the CAD software or manufacturers' data sheets. All components that will be used are included in the table, even bolts and wires, in order to make the overall mass as accurate as possible.
All masses are final and will no longer change with the design. Small contingencies of 5% are still in place as it is not always possible to precisely recreate a part in CAD software. This is due to limitations on manufacturing equipment and is more thoroughly explained in Chapter 15.
The drone will weigh approximately 656 grams, which is roughly 145 grams lighter than the limit of 800 gram set by the stakeholder. At this mass the drone will be able to fulfill the needs of the stakeholder and have excellence endurance.
66% of the mass is taken up by the battery, motors and other propulsion elements. This is to be expected as these are vital for the performance of the drone. The powerful motors together with the battery are needed to obtain the thrust to weight ratio of 5 and the endurance of 10 minutes during normal flight.
A mere 15% of the mass is reserved the artificial intelligence aboard the drone. This allows for two cameras, a very precise IMU and two computers to be carried.
The last 20% of the drone's weight is taken up by the structural components that will keep the drone together while in flight and while crashing. This category of components also includes some of the protective parts, such as propeller guards and crumple zones.
The mass of 711 gram is almost the final mass and except for minor changes will not change much.

### Mass development

For all projects in the field of aerospace engineering it is of utmost importance to keep track of the total mass at all times. Design specific elements such as motors, batteries and propellers depend directly on the mass. If the frame becomes heavier they will have to change too. When the mass management is not done properly this can cause a negative spiral in which the drone becomes heavier and heavier.
To help in the design of the drone a detailed mass budget was made at the start of the detailed design phase. Furthermore, as the design became more detailed, the contingency margins became smaller. The mass development is drawn in Figure 12.2.
At the start of the detailed design phase the mass was estimated to be roughly 400 gram, Chapter 4. The upper design margin was expected to be 480 gram.

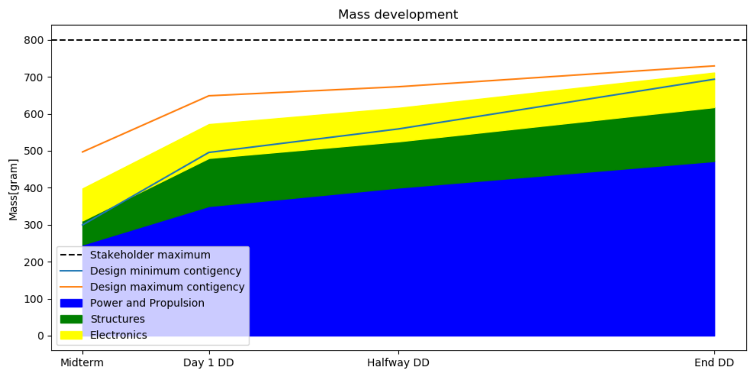


Figure 12.2: Mass development during the detailed design phase

The team quickly realized that some of the risk mitigation strategies had to be implemented to lower risks in the final design. These strategies include addition of propeller guards and crumple zones. Furthermore, after detailed analysis the stress due to an impact was found to be higher than expected, this resulted in additional protections being needed on the components. This is why, after the first day of the detailed design phase, the total mass had grown by 150 grams, as can be seen in Figure 12.2.
During iterations of the design the outline of components became more precisely known and therefore their design margins reduced. This resulted in a final design mass of 656 grams, with only very small design margins left.

### 12.1.2. Power budget

The amount power that is used by each subsystem is shown in Table 12.2. Compared to the power estimate in Chapter 4, which was 110W, the power needed grew because it was found that more power is consumed by the motors. A more accurate analysis for the different usage profiles was done. This is explained in Chapter 6.

Table 12.2: Power breakdown

| **Component** | **Power Consumption [W]** | **Qt** |
|---|---|---|
| Pocket Beagle | 6.0 | 1 |
| Raspberry Pi Zero W | 5.0 | 1 |
| Pico Flexx | 0.3 | 1 |
| RGB Camera | 1.0 | 1 |
| XL-MaxSonar-EZ | 0.5 | 1 |
| Power Distribution Board | 0.96 | 1 |
| Turnigy Multistar 30A Rev16 v3 ESC | 0.5 | 4 |
| XRotor Motor | 37 | 4 |
| VectorNav VN-100 | 0.5 | 1 |
| **Total** | **164.3** | |

One thing to note is that the power that is used by the motors was calculated assuming a speed of 1 m/s. During the track, the drone will be accelerating and decelerating, but, on average, the throttle of the drone is expected to remain very low, and thus a power draw of 37W per motor is likely to be a reasonable value. At full throttle, however, the needed power of each motor was calculated to be around 550W in Section 6.3.3, an order of magnitude larger than what is reported in Table 12.2. This has significant implications on the flight time. To meet the flying time requirement, the battery capacity was sized according to the velocities expected during the race, which are much lower than the capabilities of the hardware.

### 12.1.3. Computational Resources

RAIDER carries 2 computer boards. As shown in Chapter 11, the gate recognition runs on the Raspberry Pi Zero W, whereas the path planning, sensor fusion, flight controller, and system monitoring are all integrated into the Pocket Beagle.

Since the Raspberry Pi Zero W's BCM2835 only has to do computations for the gate detection, this is allowed to take up 100% of its power. On the Pocket Beagle, however, this is a little more complicated. As seen in Figure 11.2, only the sensor fusion and path planning are run on the Pocket Beagle's main processor, the ARM Cortex A8. Here, the sensor fusion uses approximately 13% of the computational power. The path planning takes up the rest. On top of the A8, the Pocket Beagle also has an ARM Cortex M3 and 2 Programmable Realtime Units (PRUs). The flight controller will be run on the M3, allowing it take up all of its power. Finally, the system monitoring module will run on one of the PRUs since it only needs a very small amount of power.

### 12.1.4. Cost

Using the overall parts list a detailed cost breakdown can be made. This cost breakdown only includes all the components and does not take into account any spare parts, which are listen in Table 17.8. The table containing all parts can be found in Table 12.3.

Table 12.3: Cost breakdown

| Category | Component | Cost[€] | Qt |
|---|---|---|---|
| Structures | Center Frame | 15.00 | 1 |
| | Arm Frame | 5.00 | 4 |
| | Additional Hardware | 50.00 | |
| | Propeller Guard | 0.28 | 4 |
| | Motor Spacer | 0.04 | 4 |
| | Bottom Frame | 10.00 | 1 |
| | Crumple Zone | 5.00 | 4 |
| | Top Frame | 5.00 | 1 |
| | **Total** | **71.27** | |
| Sensors and Computers | VectorNav VN-100 | 450.00 | 1 |
| | IMU Damper | 5.00 | 4 |
| | Power Distribution Board | 50.00 | 1 |
| | Raspberry Pi Zero W | 25.00 | 1 |
| | Pocket Beagle | 25.00 | 1 |
| | PCB Stand-Off | 0.04 | 4 |
| | Pico Flexx | 300.00 | 1 |
| | Additional Cables | 50.00 | 1 |
| | XL-MaxSonar-EZ | 35.00 | 1 |
| | RGB Camera | 76.50 | 1 |
| | **Total** | **1031.67** | |
| Power and Propulsion | Additional Hardware | 0.20 | 16 |
| | XRotor Motor | 21.65 | 4 |
| | Turnigy Multistar 30A Rev16 v3 ESC | 15.2 | 4 |
| | 5045 Propeller | 4.00 | 4 |
| | Turnigy 1550mAh 6s | 28.46 | 1 |
| | **Total** | **195.06** | |
| Production | Hours of skilled labour | 31.10 | 5 |
| | **Total** | **155.5** | |
| **Total** | | **1454.-** | |

The cost for the power and propulsion, and electronics subsystems is mostly set, the cost for the frame on the other hand had to be estimated. The cost estimation for the carbon parts was made based on the surface cost of a single carbon plate[1]. The costs for water cutting, the manufacturing technique as described in Chapter 15, are not included in this. However, all costs were rounded up to account for this, since the cost of water cutting for such small order is very hard to estimate.

For all parts considered to be 3D printed only the raw material cost were taken into consideration[2]. It is expected the end user will have a 3D printer available to make spare parts. For all other parts, the cost was taken from HobbyKing[3]. The labour costs were estimated using the average labour cost in the Netherlands[4]. This amount was deemed reasonable because the assembly and testing of the drone are simple enough for a relatively unskilled worker to perform them.

The final cost of all components in the drone sums up to €1454,-. This is only slightly more than half of the budget set by the stakeholder requirement and leaves enough for spare parts to be bought. A remark should be made to the result of the preliminary design in Chapter 4. In fact, the total cost estimated was €1253, showing a very good estimation of contingency in the early design stage and that the detail of the design was already quite high.

[1] `https://www.rockwestcomposites.com/plates-panels-angles/carbon-fiber-plate/carbon-fiber-fabric-plate/404-410-group` [Visited at June 20 2018]

[2] `http://3dinsider.com/3d-printing-filament-cost/` [Visited at June 20 2018]

[3] `https://hobbyking.com/` [Visited at 2 July 2018]

[4] `https://www.cbs.nl/en-gb/news/2012/17/increase-labour-costs-in-the-netherlands-equals-eu-average` [Visited at 2 July 2018]

### 12.1.5. Size

The final drone will have a width and length of 291.7mm and 289.2mm respectively. The center of the propellers are placed 147mm apart resulting in a gap of 20mm between the tips of the propellers at all times. This was done to ensure that they will not collide and to optimize the performance.
The total height of drone is 82.9mm. A front and top of the drone can be seen in Figure 12.4 and Figure 12.3 respectively. At the beginning of the design phase the dimensions were 220 mm, 220 mm and 85 mm, respectively for width, length and height. While the height stayed the same the other dimensions increased. After the reiteration of the design the power and propulsion subsystem required an increased propeller in order to perform adequately. The main reason is that less assumptions are made for the detailed design, and more accurate margins are considered as explained in Chapter 6.

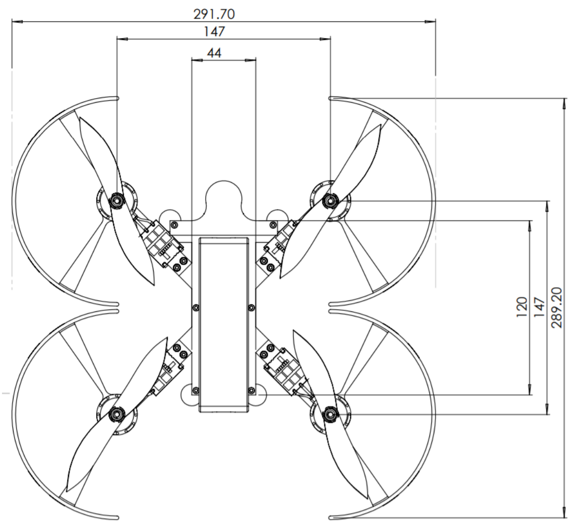


Figure 12.3: RAIDER top view

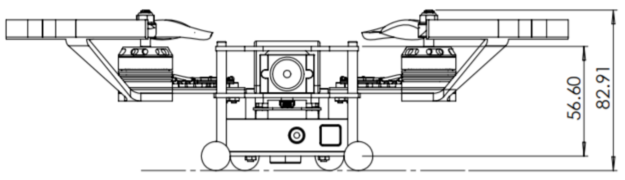


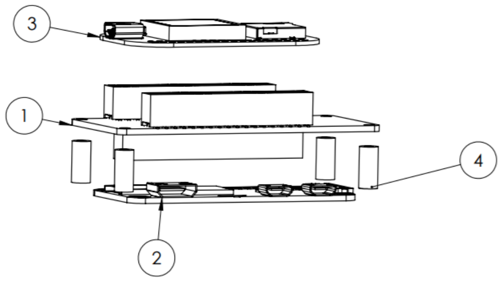
Figure 12.4: RAIDER front view

## 12.2. Assembly overview

Using the CAD software exploded views can be made to further show how all components are mounted together. Furthermore exploded views can be used to check on forehand for assembly errors.

### 12.2.1. PCB stack

In the center of the drone, in between two carbon frame plates, the main computer boards are located. They are separated by the custom designed power distribution board, which acts as a motherboard. Both computers are mounted to the motherboard using their GPIO pins and other mounting hardware. The motherboard is then attached to the carbon plates using PCB stands lifting all boards off the plate. An exploded view of all components can bee seen in Figure 12.5.



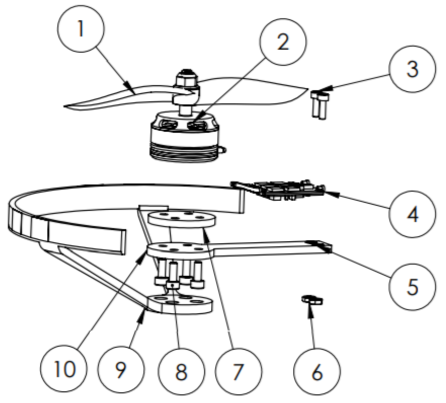
| # | Name | Qt |
|---|---|---|
| 1 | Motherboard | 1 |
| 2 | Raspberry Pi Zero W | 1 |
| 3 | Pocket Beagle | 1 |
| 4 | PCB Stand-Off | 4 |

Figure 12.5: Exploded view of the pcb assembly

Mounting the computers on a separate motherboard takes more effort since the motherboard has to be designed just for this purpose. It has the massive advantage, however, of provided all of the connectivity that is needed in a well organized manner. It also reduces the amount of cables that are needed, making debugging much easier and faster. The team decided to integrate this functionality into the custom PDB since this was needed anyway to convert the 22.2V of the battery to 5V for the computers.

### 12.2.2. Arm

The most complex sub assembly of the drone is the arm. The drone will have four arms in total. They are all essentially the same, apart from the fact that 2 are mirrored. They can, however, be built using the same components and only the propeller guards will have to be rotated in order to fit them on the opposite positions. An exploded view of just the arm and its components is shown in Figure 12.6.



| # | Name | Qt |
|---|---|---|
| 1 | 5045 Propeller | 1 |
| 2 | XRotor Motor 1750KV | 1 |
| 3 | DIN912 M2.5x8 | 2 |
| 4 | Kiss ESC | 1 |
| 5 | Arm Inserts | 2 |
| 6 | DIN934 M2.5 | 2 |
| 7 | Motor Spacer | 1 |
| 8 | DIN912 M3x8 | 4 |
| 9 | Propeller Guard | 1 |
| 10 | Arm Frame | 1 |

Figure 12.6: Exploded arm view

The propeller is bolted to the rotor with a single bolt, to allow for easy maintenance. Since, during propeller impacts, the motor shafts can also bend, the motors themselves are mounted to the arm frame using only four bolts. The motor is lifted by a 3D printed spacer. The arm attaches the motor to the center frame and consists of a 2.9mm thick carbon plate comprised of 4 layers of carbon. The bolt holes are reinforced using aluminum inserts. Although this starts a galvanic reaction, the lifespan of the drone is short and therefore the low weight is deemed more important. To optimize performance, the ESC was placed as close to the motor as possible directly on the arm. The cables to the motor can be soldered and the placement should ensure good cooling. Lastly the propeller guards were implement to ensure nothing gets caught in the propeller by accident. The propeller guards are designed to be printable and have to be press fit over the bolts.

### 12.2.3. Complete Drone

The exploded view of the final frame can be seen and is explained in Figure 12.7.

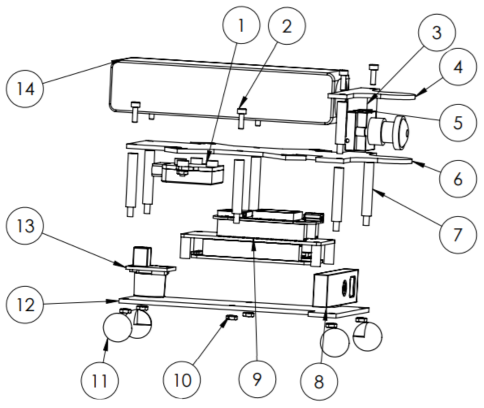


Figure 12.7: Exploded view of the total assembly

| # | Name | Qt |
|---|---|---|
| 1 | IMU with Dampers | 1 |
| 2 | DIN912 m2.5x8 | 6 |
| 3 | Top Stand-Offs | 2 |
| 4 | Top Frame | 1 |
| 5 | Connex Camera | 1 |
| 6 | Center Frame | 1 |
| 7 | Bottom Stand-Offs | 6 |
| 8 | Pico Flexx | 1 |
| 9 | PCB Assembly | 1 |
| 10 | DIN934 m2.5 | 6 |
| 11 | Crumple Zones | 4 |
| 12 | Bottom Frame | 1 |
| 13 | XL-MaxSonar-EZ | 1 |
| 14 | Turnigy 1550mAh 6s | 1 |

As can be seen in the exploded view a lot of components are placed between the two frame plates, for which there are several reasons.Firstly, the small, compact size is beneficial during the race. Secondly, the frames protect vital and expensive electronics from all sides, increasing the crashworthiness of the drone.The bottom plate houses the main PCB assembly, the Pico Flexx depth camera and the ultrasound sensor. The PCB assembly is lifted off the bottom plate using standoffs. These standoffs can be made of foam so they act as a shock absorber, which will further increase the crashworthiness of the drone. The battery, RGB camera and damped IMU are mounted to the top plate. The IMU is damped using four dampers, detailed in Section 5.3. Two of the dampers have thread ends for securing the IMU, while the others are just rubber feet. An extra frame is mounted above the RGB camera to protect it from an impact from above. The frame is designed such that it covers the camera and prevent the battery from sliding forward in the camera. The frame sticks out to the front, just like the center frame, to protect the fragile lens of the camera.

# 13 Sensitivity Analysis

As described by Loucks and van Beek [28], the goal of a sensitivity analysis is to quantitatively estimate the impact that certain model inputs have on the outcome of the model. This data, together with an analysis of the uncertainties, can support the decision process as well as determine how much weight should be placed in the results of a certain model based on the reliability of the output. It can also help establish if it is worth allocating resources to reduce the uncertainties: if the result of the model is not very sensitive to the magnitude of these uncertainties, said resources can be more effectively allocated to other issues.

With regards to this project's design, it can be useful to generate the information needed to state, with evidence, that the configuration chosen is a feasible solution even when parameters change from their assumed value. It also demonstrates that the hardware is not over-designed beyond reasonable margins pertinent to this stage of the design. The sensitivity analysis was conducted on the approach used to define the final hardware configuration of the drone, described in Section 6.3.3. This model outputs all viable configuration that can be achieved with a certain combination of existing batteries and propellers. As such, a standard sensitivity analysis cannot be conducted due to the discrete nature of this output. Instead, certain input parameters which had to be assumed were varied, until the chosen configuration was no longer able to fulfill the driving requirements, or the performance demanded by the configuration exceeded the physical limits of the hardware.

The first step was to identify the uncertain inputs and observe the impact caused by individually varying them. The inputs considered were the assumed profile drag coefficients $C_{d_0}$, the percentage of the race spent turning as opposed to flying level $c_{\text{turn}}$, the nominal average speed during the race $V_{\text{nom}}$, the mass of the cables and structure $m_{\text{other}}$, and the power required by components other than the motors $P_{\text{other}}$. None of these parameters caused the configuration to become unfeasible when decreased, as was to be expected, and as such no lower bound for the different parameters was found. $c_{\text{turn}}$ did not cause the configuration to be no longer feasible when varied within its physical range (0 to 1), and $C_{d_0}$ for both turn drag and drag at 100 km/h required an increase of over 800%. The model was thus found to not be sensitive to these two parameters, and they will therefore not be discussed further. The upper limit of the increase allowed for the remaining parameters is reported in Table 13.1.

| **Parameter** | **Value Increase** | **Percentage Increase** | **Comment** |
|---|---:|---:|---|
| $V_{\text{nom}}$ | 5.6 m/s | 560% | Beyond this speed flight time was below 10 minutes |
| $m_{\text{other}}$ | 48 g | 27% | Beyond this mass the RPM required exceeded motor's capabilities |
| $P_{\text{other}}$ | 47 W | 293% | Beyond this power flight time was below 10 minutes |

Table 13.1: Upper bound of parameters tested for the sensitivity analysis

Table 13.1 shows that the configuration is not sensitive to both the power of additional components and the average track speed. In general, the power required by the four motors is much higher than that of the lightweight Raspberry Pi, PocketBeagle and the other electronics on board, therefore it is reasonable to see a large increase in $P_{\text{other}}$ before it becomes a significant problem. In terms of flying speed, racing at an average velocity of 6.6 m/s would result in completing the track in 9.1 seconds, a value that does not take into account the physical constraints of the hardware. In general, however, a 560% increase in the nominal racing speed is not unexpected, as the hardware is designed to achieve a thrust to weight of 5. Also, the reason the configuration was not deemed feasible anymore at 6.6 m/s was that the flight time fell below 10 minutes. Originally, this requirement was imposed such that the drone would have the chance to complete many attempts before having to spend the precious allotted time to switch the battery, and at this speed, the drone would be able to complete 9-10 attempts. Flying even faster would increase this number even further, and thus the configuration could be still considered feasible even though the flying time would be below the required amount.

The configuration, however, showed to be more sensitive to changes in the mass of the drone. This follows from the fact that the configuration was designed to achieve the driving requirements of the tight turn radius and high top speed. Due to the discrete nature of the components, the motors and batteries were chosen to be slightly higher than what was required, and thus there is still some room for the structure and cable mass to increase before the required thrust to weight ratio can no longer be reached. The overall mass of the drone is, at this point in the design, known to with a low margin of error. This is due to the fact that the majority of the components are bought, rather than designed, and have a known weight. The mass of the structure was obtained from the CAD model, as explained in Chapter 12, and is thus also not expected to increase by a percentage as high as 27% once manufactured. Of course, another cause of a mass increase is if it is discovered during validation tests that the drone is incapable of withstanding the ultimate load cases, which would result in a sturdier structure. If the carbon plates thickness was increased by 1 mm everywhere, the resulting mass increase would be of around 21 g, which is still below the upper increase allowed. Even then, most of the mass increase would likely occur only in the arms or as a result of additional shock absorbers, and would not happen everywhere on the drone.

Of course, the sensitivity analysis can show that certain parameters are allowed to increase by a certain value, but without understanding the uncertainty in the model output, this value is meaningless. An uncertainty analysis of the model used is unfortunately hard to accurately conduct within the project's resources. As stated in Section 6.3.4, the model uncertainty was estimated to be close to what was claimed by ECalc (±15%), as the output of both programs was similar for a number of configurations. Assuming a direct relationship between a change in the inputs and the model output, the mass increase allowed in the worst case is 27% – 15%, or around 21 g, still leaving an acceptable margin to the mass budget of the structure.

# 14 Compliance and Feasibility

Although the requirement compliance is checked in every subsystem's chapter, it is summarized in this chapter. First, in Section 14.1, the total table of technical requirements is shown. This also shows whether or not they have been complied with. Then, in Section 14.2, the final compliance with the stakeholder requirements is shown. And finally, in Section 14.3, the reasons for the non-compliance are explained.

## 14.1. Technical Requirement Compliance

Each technical requirement is shown in Table 14.1. The first 2 columns explain the requirement. The third shows whether or not the requirement has been Verified and Validated. And finally, the fourth column shows where the compliance was verified and validated. When the word Children is used in Table 14.1, it means that this requirement is verified because all of its child requirements have been verified. If a requirement is verified but not validated, ~is used. If a requirement is not verified or validate the × symbol is used. These reasons for this will be explained in Section 14.3.

Table 14.1: Requirement Compliance Table

| | | | |
|---|---|---|---|
| **Resources** | | | |
| Sys-Co-1 | The project shall stay within available resources. | ✓ | Children |
| Sys-Co-1-1 | The project shall be completed within the set time. | ✓ | Children |
| Sys-Co-1-1-1 | The project shall be completed within 50 working days. | ✓ | Inspection |
| Sys-Co-1-1-2 | The project shall be completed with 9 full time members. | ✓ | Inspection |
| Sys-Co-1-2 | A single unit shall cost no more than 2500 euros, excluding maintenance, operations and repair costs. | ✓ | Table 17.6 |
| Sys-Co-1-3 | The drone shall be transportable to the race location. | ✓ | Chapter 16 |
| Sys-Co-1-4 | The drone shall be producible using facilities of the TU Delft. | ✓ | Chapter 15 |
| **Safety** | | | |
| Sys-Co-2 | The drone shall not pose any safety hazards at any phase of its life cycle. | ✓ | Children |
| Sys-Co-2-1 | The drone shall not pose any safety hazards during production. | ✓ | Chapter 15 |
| Sys-Co-2-2 | The drone shall not pose any safety hazards during operation. | ✓ | Section 17.4 |
| Sys-Co-2-2-2 | Thrust unit debris shall not have an energy higher than 20J. | ✓ | Section 17.4 |
| Sys-Co-2-2-3 | The energy storage system shall not pose safety hazards during operation. | ✓ | Section 17.4 |
| Sys-Co-2-3 | The drone shall not pose any safety hazards at end of life. | ✓ | Section 19.3 |
| **Sustainable** | | | |
| Sys-Co-3 | The drone shall adhere to the principles of sustainability to a level that is at least equal to the one specified by the stakeholder. | ✓ | Children |
| Sys-Co-3-1 | The drone shall consist of sustainable materials. | ✓ | Section 19.3 |
| Sys-Co-3-1-1 | All materials used shall be non hazardous. | ✓ | Section 19.3 |
| Sys-Co-3-1-2 | 100% of materials shall be recyclable. | ✓ | Section 19.3 |
| Sys-Co-3-2 | The drone shall be manufactured in a sustainable manner. | ✓ | Section 19.3 |
| Sys-Co-3-2-1 | The manufacturing technique shall use no hazardous consumables. | ✓ | Section 19.3 |
| Sys-Co-3-2-2 | The manufacturing techniques shall use maximum power of 2kWh of energy. | ✓ | Section 19.3 |

| | | | |
|---|---|---|---|
| Sys-Co-3-2-3 | The manufacturing techniques shall produce a maximum 1kg of waste. | ✓ | Section 19.3 |
| Sys-Co-3-3 | The drone shall have a modular design at least on a subsystems level. | ✓ | Chapter 5 |
| Sys-Co-3-4 | The drone shall be operated in a sustainable manner. | ✓ | Children |
| Sys-Co-3-4-1 | The drone shall require at most 200W of power, in nominal conditions. | ✓ | Chapter 16 |
| Sys-Co-3-5 | The drone shall have a sustainable end of life. | ✓ | Section 19.3 |
| Sys-Co-3-5-1 | Components of the drone shall be reusable after winning the race. | ✓ | Chapter 15 |
| **Legal** | | | |
| Sys-Co-4 | The drone shall comply with all relevant regulations at every stage of its operational lifetime. | ✓ | Children |
| Sys-Co-4-1 | The drone shall comply with all regulations imposed by the governmental entities of the countries in which it is operated. | ✓ | Children |
| Sys-Co-4-1-1 | The drone shall adhere to freely available bandwidths for communications. | ✓ | Section 11.2.1 |
| Sys-Co-4-2 | The drone shall comply with all regulations imposed by the IROS organization. | ✓ | Section 3.2 |
| Sys-Co-4-2-1 | Enrollment for the IROS 2018 ADR shall be done in time. | × | Section 14.3 |
| Sys-Co-4-2-2 | The team shall be present at the IROS 2018 ADR. | ~ | Section 20.3 |
| Sys-Co-4-2-3 | The drone shall be present at the IROS 2018 ADR. | ~ | Chapter 16 |
| **Structures** | | | |
| Sys-Te-5 | The drone shall have structural integrity. | ✓ | Chapter 5 |
| Sys-Te-5-1 | The structure shall be able to handle loads imposed whilst flying. | ✓ | Section 5.1 |
| Sys-Te-5-1-1 | The structure shall transfer a load in positive Z-direction of 20N from a thrust unit to the other parts of the frame. | ✓ | Section 5.1 |
| Sys-Te-5-2 | The structure shall be able to handle vibrations introduced whilst flying. | ✓ | Section 5.3 |
| Sys-Te-5-2-1 | The structure shall have eigenfrequencies higher than 5kHz. | ✓ | Section 5.3 |
| Sys-Te-5-3 | The structure shall ensure structural integrity during a crash. | ✓ | Section 5.1,5.2 |
| Sys-Te-5-3-1 | The structure shall not be permanently deformed by a fall at 100km/h on a solid floor from a height of 3m. | ✓ | Section 5.2 |
| Sys-Te-5-3-2 | The structure shall not be permanently deformed upon an impact in X or Y direction at a speed of 100km/h. | ✓ | Section 5.1,5.2 |
| Sys-Te-5-4 | The structure shall have a drag coefficient of 0.7 or lower. | | |
| Sys-Te-5-5 | The structure shall by smaller than 600mm in any direction. | ✓ | Section 5.1 |
| **Communications** | | | |
| Sys-Te-6 | The drone shall be able to communicate with the ground segment. | ✓ | Children |
| Sys-Te-6-1 | The communications subsystem shall handle the internal communication of the drone. | ✓ | Section 11.2.1 |
| Sys-Te-6-2 | The communications subsystem shall provide the capability to communicate with the drone externally. | ✓ | Section 11.2.1 |
| Sys-Te-6-2-1 | The communication subsystem shall be able to process input from the ground segment. | ✓ | Section 11.2.1 |
| Sys-Te-6-2-2 | The communication subsystem shall be able to send a feed of the drone's sensors to the ground segment. | ✓ | Section 11.2.1 |
| Sys-Te-6-2-3 | The communication subsystem shall provide information on the drone's system status. | ✓ | Appendix A |
| Sys-Te-6-2-3-1 | The communication subsystem shall provide information on the drone's battery state of charge. | ✓ | Appendix A |

| | | | |
|---|---|---|---|
| Sys-Te-6-2-3-2 | The communication subsystem shall provide information on the drone's software errors. | ✓ | Appendix A |
| Sys-Te-6-2-3-3 | The communication subsystem shall provide information on the drone's power usage. | ✓ | Appendix A |
| Sys-Te-6-2-3-4 | The communication subsystem shall provide information on the drone's temperatures. | ✓ | Appendix A |
| **Power** | | | |
| Sys-Te-7 | The power subsystem shall supply enough power to the drone for a flight of 10 minutes. | ~ | Children |
| Sys-Te-7-1 | The power subsystem shall store energy. | ~ | Children |
| Sys-Te-7-1-1 | The power subsystem shall store at least 60kJ of energy. | ✓ | Section 6.4 |
| Sys-Te-7-1-2 | The power subsystem shall be rechargeable. | ✓ | Section 6.4 |
| Sys-Te-7-1-3 | The power storage unit shall be replaceable in 15s. | ~ | Section 14.3 |
| Sys-Te-7-1-4 | The power storage unit shall have a minimal C-rating of 35. | ✓ | Section 6.4 |
| Sys-Te-7-2 | The power subsystem shall provide each subsystem with power. | ✓ | Children |
| Sys-Te-7-2-1 | The power subsystem shall provide a maximum power of at least 250W to the ESCs. | ✓ | Section 6.4 |
| Sys-Te-7-2-2 | The power subsystem shall provide 6W to the flight computer. | ✓ | Chapter 12 |
| **Guidance** | | | |
| Sys-Te-8 | The drone shall plan the flight on board. | ✓ | Children |
| Sys-Te-8-1 | The guidance subsystem shall plan the path through the gates. | ✓ | Children |
| Sys-Te-8-1-1 | The path shall be updated at at least 10Hz. | ✓ | Section 11.3 |
| Sys-Te-8-1-2 | The path shall be calculated in 3 dimensional space. | ✓ | Section 9.2 |
| Sys-Te-8-1-3 | The path shall be calculated at least 1 gate in advance. | ✓ | Section 9.2 |
| Sys-Te-8-1-4 | The path shall not violate any manoeuvrability constrains. | ✓ | Section 10.2.3 |
| Sys-Te-8-2 | The guidance subsystem shall only use on board computation whilst flying. | ✓ | Section 11.1 |
| Sys-Te-8-3 | The guidance subsystem shall command the flight computer. | ✓ | Children |
| Sys-Te-8-3-1 | The guidance subsystem shall send the position command at 10Hz. | ✓ | Section 11.1 |
| Sys-Te-8-3-2 | The guidance subsystem shall send the heading command at 10Hz. | ✓ | Section 11.1 |
| Sys-Te-8-4 | The guidance subsystem shall have a safe mode. | ✓ | Children |
| Sys-Te-8-4-1 | The guidance safe mode shall land the drone on the ground. | ✓ | Section 11.4 |
| Sys-Te-8-5 | The guidance computer shall land the drone at the end of the track. | ✓ | Section 9.5.1 |
| **Localization** | | | |
| Sys-Te-9 | The localization subsystem shall determine the drone's location on-board. | ✓ | Section 11.1 |
| Sys-Te-9-1 | The localization subsystem shall determine the drone's location in space. | ✓ | Children |
| Sys-Te-9-1-1 | The localization subsystem shall update the location at at least 500Hz. | ✓ | Section 7.2.2 |
| Sys-Te-9-1-2 | The localization subsystem shall calculate its accuracy. | ✓ | Section 7.1.1 |
| Sys-Te-9-2 | The localization subsystem shall detect gates. | ✓ | Section 8.4 |
| Sys-Te-9-2-1 | The localization subsystem shall determine the gate's position with an accuracy of 10cm. | ✓ | Section 8.7 |
| Sys-Te-9-2-2 | The localization subsystem shall determine the gate's orientation with an accuracy of 0.15 rad. | ✓ | Section 8.5 |
| Sys-Te-9-3 | The gate localization can be performed at a rate of 30 FPS | ✓ | Section 8.8 |
| **Control** | | | |

| | | | |
|---|---|---|---|
| Sys-Te-10 | The drone shall be controllable. | ✓ | Children |
| Sys-Te-10-1 | The control subsystem shall determine the attitude of the drone. | ✓ | Children |
| Sys-Te-10-1-1 | The control subsystem shall determine the attitude around the 3 axes. | ✓ | Section 7.1.1 |
| Sys-Te-10-1-2 | The control subsystem shall determine the attitude with an accuracy of 0.1rad. | ✓ | Section 7.1.1 |
| Sys-Te-10-1-3 | The control subsystem shall update the attitude measurement at at least 500Hz. | ✓ | Section 7.1.1 |
| Sys-Te-10-1-4 | The control subsystem shall determine the rotational rates with an accuracy of 0.03rad/s. | ✓ | Section 7.1.1 |
| Sys-Te-10-1-5 | The control subsystem shall acquire the rotational rate measurement at at least 500Hz. | ✓ | Section 7.1.1 |
| Sys-Te-10-2 | The control subsystem shall control the attitude of the drone. | ✓ | Section 10.2.5 |
| Sys-Te-10-4 | The control subsystem shall control the velocity of the drone. | ✓ | Section 10.2.3 |
| Sys-Te-10-4-1 | The control subsystem shall provide a velocity increment of 10m/s ± 5% within 1s. | | |
| Sys-Te-10-5 | The control subsystem shall communicate with the propulsion subsystem. | ✓ | Section 11.1 |
| **Maneuverability** | | | |
| Sys-Te-11 | The drone shall meet the maneuverability demands of the customer. | ~ | Children |
| Sys-Te-11-1 | The drone shall accelerate with 3g in any direction. | ~ | Section 14.3 |
| Sys-Te-11-2 | The drone shall reach a speed of 28m/s in the XY-plane[1]. | ~ | Section 14.3 |
| Sys-Te-11-3 | The drone shall be able to make a turn with a radius of 3m at 12m/s. | ~ | Section 14.3 |
| **Propulsion** | | | |
| Sys-Te-12 | The drone shall be able to fly. | | |
| Sys-Te-12-1 | The propulsion subsystem shall provide thrust. | ✓ | Children |
| Sys-Te-12-1-1 | Each thrust unit shall provide a thrust of 4N continuously. | ✓ | Section 6.3.3 |
| Sys-Te-12-1-2 | Each thrust unit shall provide a thrust of 5N for at least 2s. | ✓ | Section 6.3.4 |
| Sys-Te-12-2 | The propulsion subsystem shall control the thrust. | | |
| Sys-Te-12-2-1 | The propulsion subsystem shall be able to control the thrust level with an accuracy of 0.5N. | | |
| Sys-Te-12-2-2 | The propulsion subsystem shall take control commands at a frequency of 400Hz. | | |
| Sys-Te-12-4 | A single entire thrust unit shall be replaceable in 60s. | ~ | Section 14.3 |
| Sys-Te-12-4-1 | The propeller shall be replaceable in 15s. | ~ | Section 14.3 |
| Sys-Te-12-4-2 | The motor and ESC shall be replaceable in 60s. | ~ | Section 14.3 |

## 14.2. Stakeholder Requirements

Similarly to Table 14.1, the first two columns of Tables 14.2 to 14.4 have each requirement's identifier and explanation in them. The third column shows which requirements have been complied with and the final column links to where this compliance has be validated. The symbols that were explained in Section 14.1 are used here as well.

[1]The XY-plane is orthogonal to axis of rotation of the propellers

Table 14.2: List of Tier 1 Customer Requirements

| Tier 1 | | | |
|---|---|---|---|
| **SH-P-2r** | The drone shall be able to autonomously fly a full lap of the IROS 2018 track, without crashing and in at most 10 minutes. | ~ | Section 14.3 |
| **SH-SR-8r** | Only parts that cost less than 10% of the unit cost and can be replaced in under 1 minute shall require replacement upon impact with concrete when free-falling from a height of 3m with zero initial velocity. | ~ | Section 14.3 |
| **SH-SR-9r** | Only parts that cost less than 10% of the unit cost and can be replaced in under 1 minute shall require replacement upon crashing into a gate at 100 km/h. | ~ | Section 14.3 |
| **SH-EB-13** | The drone shall comply with all requirements given by the 2018 autonomous drone race website if available. | ✓ | Section 3.2 |
| **SH-O-16** | The drone shall use onboard computation only. | ✓ | Chapter 11 |
| **SH-O-18** | The drone shall be able to detect IROS 2018 autonomous drone race gates. | ✓ | Chapter 8 |

Table 14.3: List of Tier 2 Customer Requirements

| Tier 2 | | | |
|---|---|---|---|
| **SH-P-1** | The drone shall be able to automatically take off and land. | ✓ | Section 9.5.1 |
| **SH-P-6** | The drone shall be able to fly for at least 10 minutes. | ~ | Section 14.3 |
| **SH-P-7** | The onboard computer vision shall be able to track gates at 30 frames per second. | ✓ | Section 11.3 |
| **SH-S-11** | The motors, frame, electronics and cameras shall be replaceable. | ✓ | Chapter 5 |
| **SH-EB-12r** | No dimension of the drone shall exceed 60 cm. | ✓ | Chapter 12 |
| **SH-O-17r** | The drone shall use, but is not restricted to, at least one camera for indoor navigation. | ✓ | Chapter 12 |
| **SH-O-19r** | The aggregate time required to replace all replaceable components of the drone shall not exceed 7 min. | ✓ | Appendix B |

Table 14.4: List of Tier 3 Customer Requirements

| Tier 3 | | | |
|---|---|---|---|
| **SH-P-3r** | The drone shall possess the hardware characteristics to fly at a maximum speed of 100 km/h in horizontal straight flight. | ~ | Section 14.3 |
| **SH-P-4** | The drone shall be able to linearly accelerate 3g. | ~ | Section 14.3 |
| **SH-P-5** | The drone shall be able to make 3m radius arcs at 12 m/s. | ~ | Section 14.3 |
| **SH-S-10r** | The drone shall be made for 100% of recyclable materials. | ✓ | Section 19.3 |
| **SH-EB-14** | The total mass shall be less than 800 gram. | ✓ | Chapter 12 |
| **SH-C-15r** | The aggregate hardware cost of the drone shall not exceed 2500 Euro, excluding maintenance, repair and operational costs. | ✓ | Section 17.2 |

## 14.3. Incomplete Requirement Explanations

All of the requirements that have not been verified or validated yet are listed in Table 14.5.

| Requirement ID | Reason of Non-Compliance |
|---|---|
| Sys-Co-4-2-1 | Since RAIDER was designed as an exercise, the intent was never to actually compete in the competition. Therefore the team did not register with the organizers. |
| Sys-Co-4-2-2 | Since the IROS is, as of writing this, still 3 months away, it is impossible to validate that all of the team and the drone will be present. This has been planned in Chapter 16. |
| Sys-Co-4-2-3 | See Sys-Co-4-2-2 |
| Sys-Te-7-1-3 | Without building the drone, it is impossible to get exact times for the replacement of different parts. However, the drone was designed to be quickly repairable, as explained in Chapter 5. |
| Sys-Te-11-1 | Without building the drone and actually testing it, it is impossible to know the exact value of the thrust. However, the expected thrust-to-weight ratio is over 5. An acceleration of 3g should not pose any problem with this. |
| Sys-Te-11-2 | Without building the drone and actually testing it, it is impossible to know the exact value of the thrust. However, the top speed of the drone is expected to be 160km/h, 44m/s. Since this is much higher than the 28m/s the requirement states, it is expected to be complied with. |
| Sys-Te-11-3 | Without building the drone and actually testing it, it is impossible to know the exact value of the thrust. However, since the drone was designed for this requirement, it is expected to be complied with. |
| Sys-Te-12-4-1 | See Sys-Te-7-1-3 |
| Sys-Te-12-4-2 | See Sys-Te-7-1-3 |
| SH-P-2r | Without having the drone physically fly through the track, this cannot be fully validated. However, the simulations that is explained in Chapter 9, shows that it should be possible. |
| SH-SR-8r | Without crash testing the drone, it is impossible to calculate the exact probabilities of each component breaking. However, as explained in Chapter 5, RAIDER was designed to withstand crashes and expensive components were placed in shielded locations in order to maximize the chance of complying with this requirement. |
| SH-SR-9r | See SH-SR-9r |
| SH-P-6 | Although the battery was sized to comply with this requirement in Section 6.3.2, it cannot be fully validated without physically trying it. |
| SH-P-3r | See Sys-Te-11-2 |
| SH-P-4 | See Sys-Te-11-1 |
| SH-P-5 | See Sys-Te-11-3 |

Table 14.5: Requirements that have not been complied with and the reasons for this.

# 15
# Production Plan

Before the drone can be tested it needs to be built. During the final phases of the design the team had to take the manufacturability of all components into account. In this chapter a detailed overview is given for the production of the frame as can be seen in Section 15.1), and the electronics in Section 15.2).

## 15.1. Frame

The carbon frame has been designed for it to easily be manufactured and produced. The carbon plates are off the shelf components and can be bought cheaply. All components for a single drone can be cut from one plate[1] and cutting template has been made. A scaled drawing of template is shown in Figure 15.1.

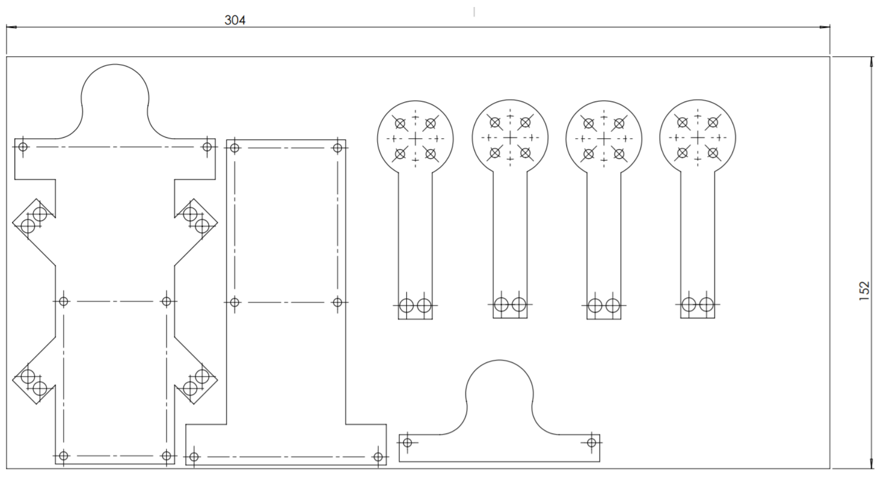
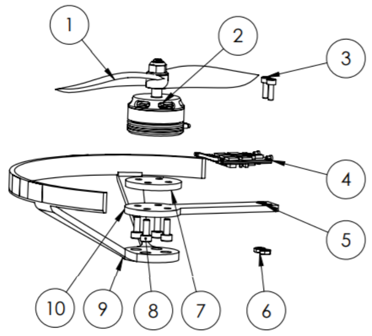


Figure 15.1: Cutting templates for a plate of 152x304mm

Detailed manufacturing instruction are separated between the main assembly and the arm assembly. The arm assembly will come together with the final assembly last.

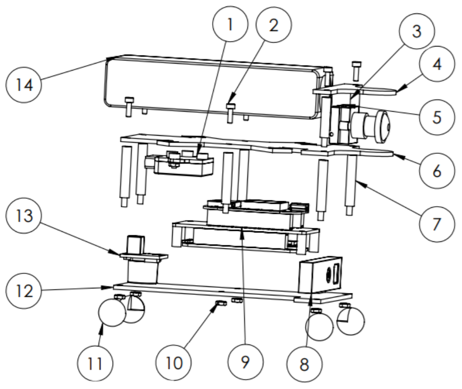
### Arm



1. Have the carbon arm frame water cut. Water cutting is preferred over milling as during the milling process the fibres can easily be damaged or the plate can delaminate.
2. Glue the arm inserts into the pre-cut holes.
3. 3D print the motor spacer.
4. Mount the motor and motor spacer to the end of the arm frame using 4 DIN912 M3 bolts.
5. Mount the ESC using zip ties on the top of the arm.
6. Cut and strip the wires of the motor to the appropriate length.
7. Solder the wires from the motor to the ESC.
8. 3D print the propeller guard.
9. Mount the propeller guard using a press fit over the M3 bolt heads.

[1] `https://www.rockwestcomposites.com/plates-panels-angles/carbon-fiber-plate/carbon-fiber-fabric-plate/404-410-group` [Visited at June 20 2018]

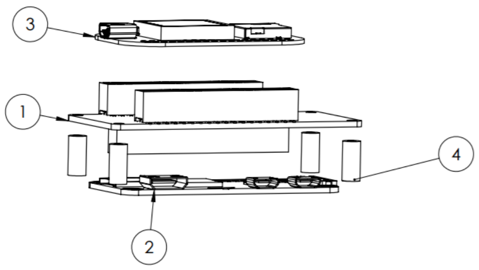
**Center Frame**



1. Have the center, bottom, and top frames water cut.
2. Glue the frame inserts at the locations of the arm bolts. These holes are pre-cut bigger than the holes for the frame stand-offs.
3. Start with the bottom frame and mount the Pico Flexx camera.
4. Directly behind the Pico Flexx camera, the PCB stand-offs can be glued to the carbon plate. The placement should be done with care to avoid hole alignment issues.
5. Mount the PCD assembly onto the PCD stand-offs after the glue has fully cured.
6. At the rear of the bottom frame the ultra sound depth sensor can be mounted through the hole.
7. Mount the bottom stand-offs through the pre-cut holes in the bottom plate and secure them using the six nuts.
8. Mount the IMU dampers to the lower side of the center frame. Careful alignment is needed to ensure the holes match the bolts.
9. Mount the IMU on top of the dampers and secure it using the nuts.
10. Now place the center frame on top of the bottom standoffs. The rear four stand-offs can be secured using M2.5 bolts. Leave the front standoffs unsecured for now for ease of assembly.
11. Mount the Connex ProSight camera to the front of the frame and ensure it does not stick in front of the protective covers.
12. Mount the top stand-offs to the front to holes. By doing so the bottom standoffs are secured.
13. Mount the top frame on top off the standoffs and secure it using the remaining bolts.
14. Slide on the legs and crumble zones.

## 15.2. Electronics

Two types of electronics should be distinguished: off the shelf components and the home built motherboard. The former should be bought well in advance of the race to make sure delivery time does not become a problem. The latter should be designed in more detail. As described in Part II, the board will convert the voltages to the right level for the cameras and other electronics. Furthermore it will provide the Pocket Beagle and Raspberry Pi Zero W with a data bus. The board first needs to be designed in Eagle or Altium, after which the PCB can be developed. To do so an electrical engineer should be acquired.
After the PCB has been developed is should be populated with all the components. A specific order is suggested to minimize the chance of failures and to allow for debugging to take place.



1. Solder together the power stage.
2. Test the power stage for voltage levels, first at low currents, then at higher currents.
3. While testing ensure the board does not get too warm and there is no excessive power loss.
4. Solder the headers for the Pocket Beagle, IMU, and ultrasound sensor together.
5. Test communication between the Pocket Beagle, IMU, and ultrasound sensor.
6. Solder ESC wires to the PDU.
7. Connect the ESC and check whether the control algorithms work together with the ESC and the drone can be kept stable.
8. Solder the headers for the Raspberry Pi Zero W.
9. Check the communication between Raspberry Pi Zero W and the Pocket Beagle.

Using this stepwise approach with intermediate testing, the risk on excessive damaged due to errors is reduced. The testing allows for critical errors to be found, such as voltage spikes and bad communication before the drone is fully assembled. If such errors occur during flight, both the drone and its surroundings could be damaged.

# 16 Operation and Logistics

In this chapter the operations and logistics of the drone will be discussed. It will provide guidance for how to handle the drone in all the different phases of the lifetime of the drone. This will include the user interfaces of the drone, maintenance instructions, transport instructions and set-up instructions. This section will be divided into the three parts that can be seen in Figure 16.1.

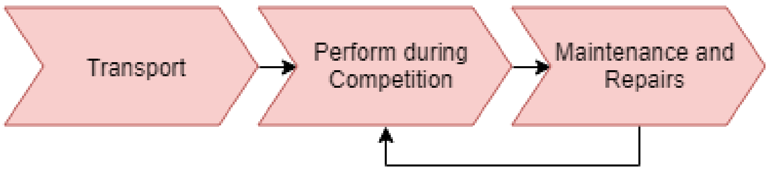


Figure 16.1: Operation & Logistics Sections

Section 16.1 will address the transport phase. After that, the competition phase will be explained.

## 16.1. Transport

To participate in the IROS 2018 Autonomous Drone Race, the drone will need to be transported to the race, how this will be organized can be found in Figure 16.2.

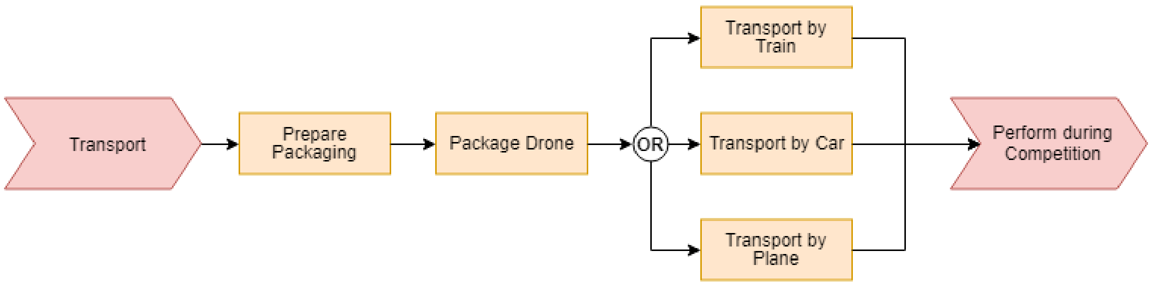


Figure 16.2: Transport Flow Diagram

The drone consists of fragile and expensive components, which should be taken into account for during transport. The race will be held in Madrid, so there are three viable options of transportation, either by plane, by car, or by train. Which of these methods of transportation is used doesn't matter as long as cautious handling of the drone is ensured. During transport, the drone should be fixed and unable to move. A special box with a foam or shock absorbing material will be provided for transporting the drone, in order to damp any vibrations and shocks induced during transportation.

## 16.2. Perform during Competition

When the drone arrives at the competition a number of different actions need to be performed for successful operation. These are outlined in Figure 16.3.

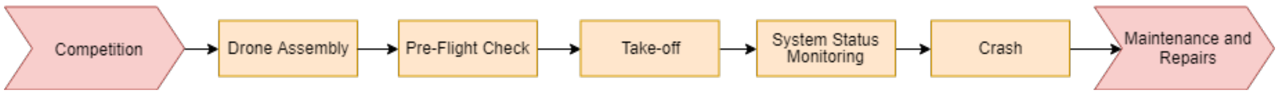


Figure 16.3: Competition Flow Diagram

First, the drone has to be set up for the race. This consists of assembling the drone and going through the pre-flight checklist. During the race itself, the drone's performance has to be monitored and maintenance and repairs have to be conducted in case the drone gets damaged. The assembly, pre-flight checks, and drone monitoring will be discussed in detail in this section. The maintenance and repairs are discussed in the next section.

### 16.2.1. Drone Assembly

The assembly of the drone was made as easy as possible. The connections between the separate parts will be made of simple connectors such as nuts and bolts. This is explained in detail in Chapter 15.

### 16.2.2. Pre-Flight Checks

After the drone is assembled, the system needs to be set-up and the pre-flight checks have to be performed. For the set-up of the system, the drone has to establish a wireless connection with the base station computer used for performance monitoring. After that the pre-flight checks shall be performed. The pre-flight checks include: booting up the software, monitoring system status, calibrating sensors, and placing the drone at the starting position. After these checks, the drone will wait on the starting position until the start input is given by the user. The software boot consists of four steps: turning on the on board computers, turning on the communications system, turning on the sensors, and turning on the propulsion system. The sensor calibration should guarantee that the sensors are well aligned and provide a correct output for a given input. The system status monitoring should make sure that all of the systems are running nominally. The monitoring of the system status should not only be done at the pre-flight checks, but continuously throughout the race. This system monitoring will be discussed in more detail in Section 16.2.3.

### 16.2.3. System Status Monitoring

The system status monitoring consists of checking the status of the individual subsystems and transmitting this to the base station. These two functions are executed continuously throughout the race. The subsystem monitoring is divided into checking the battery voltage, checking the motors, checking the temperatures, and checking the software. Battery monitoring is important so that the drone doesn't crash when the battery dies. Motor monitoring is important for identifying motor malfunction. Temperature monitoring is important in order to ensure the temperature of the critical subsystems stay within the operating temperature and to prevent overheating or fire. Lastly, checking the software is important for managing any unexpected bugs that might come up during the race and minimizing their impact on the performance. When an error in one of the subsystems is found, the drone ends its operations. Using the base station computer, the status of the individual subsystems can be monitored by the user. In an emergency, it is possible for the user to take control of the drone by means of the base station computer.

## 16.3. Maintenance and Repairs

As can be seen in Figure 16.4, the first thing to do when maintenance is to be performed, is to inspect the drone.

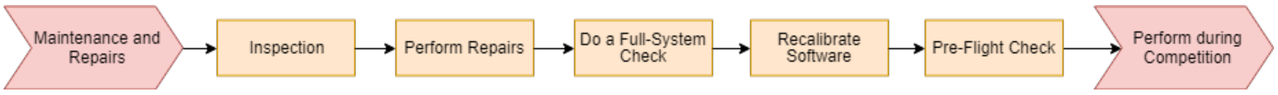


Figure 16.4: Maintenance and Repairs Flow Diagram

During inspection the propellers, motors, sensors, structure and electronics have to be thoroughly checked for damage or any other anomalies. This is followed by possibly disassembling parts in order to increase accessibility. After inspecting the drone rigorously, the eventual broken or worn components have to be replaced. When the repair is finished, a system check is performed, in order to make sure that all of the replaced components perform according to specifications. Finally, before taking the drone back into operation, the drone will be updated with the latest software version and the pre-flight checklist will be performed.

# 17
# Reliability, Availability, Maintainability, and Safety Analysis

Reliability, Availability, Maintainability, and Safety Analysis (RAMS Analysis) is an integral part of any design. The combination of the four aspects of RAMS determine for a large part the success of the project. First Reliability will be calculated in Section 17.1. Then, the drone's availability will be discussed in Section 17.2, after which the maintainability will be discussed. Finally, the safety of the drone will be discussed in Section 17.4.

## 17.1. Reliability

The first part of RAMS is reliability. It is defined as the probability that the system performs as specified by the customer. This probability can be analyzed by finding all of the different failure modes of the drone and using probabilistic theory to sum them. Finding these failure modes can be done using a failure diagram, which is depicted in Figure 17.1. This failure diagram is an OR tree, except for the lowest level. Here, there is a clearly indicated AND connection, where the failure of any 1 sensor does not immediately cause a full system failure.
Reliability for RAIDER can be subdivided into 2 parts: Reliability during Flight and reliability due to crashing, which will be discussed in Section 17.1.1 and Section 17.1.2 respectively. For most projects, reliability during flight is the main component of this. For RAIDER, however, this is different, the reason being the above normal high probability of crashing.

### 17.1.1. Reliability during Flight

First the reliability in flight will be assessed.

#### Electronics Failure

The probability of sensor and computer board failure is very difficult to determine since failures normally do not occur for years. Since the drone only has to function for about 20 minutes, these failure rates are assumed to be negligible. All components that are used in RAIDER are off the shelf products, so on top of the fact that the odds of them failing during flight are negligible, they are also relatively easy to replace.

#### Power & Propulsion Failure

The probability of battery failure is much higher than that of PDB or ESC, provided that the propellers have guards around them. The average number of charge-discharge cycles for failure is approximately 732 [61]. Assuming that a charge lasts 10 minutes, this results in a failure rate of 0.0082 failures per hour.

$$\lambda_p = \left(\frac{\lambda_1}{1.9 \cdot 80000} + \frac{\lambda_2}{1.1 \cdot 5 \cdot 10^5}\right) \tag{17.1}$$

Equation (17.1) shows a relationship between motor failure rate and the ratio between the expected lifecycle and its actual lifecycle. A temperature of 40 degrees was chosen because the IROS2018 competition will be organized at the end of the summer in Madrid. An actual lifecycle of 100 hours was chosen. This accommodates for both the testing of the drone and flying the race. The result of this assumption is $\lambda_1 = \lambda_2 = 0.13$, resulting in a motor failure rate of $1.1 \cdot 10^{-6}$ failures per hour.
Finally, the Mean Time Between Failures was calculated to 500 hours for propellers [35]. This corresponds to a failure rate of 0.002 failures per hour.

The total failure rate for Power & Propulsion Failures is then 0.0102 failures per hour.

#### Structural Failure

The probability of a structural failure during nominal flight is assumed to be negligible. The arms, for instance, are design to withstand around 150N that they will be subjected to during a crash, but during normal operating conditions, they only have to withstand 20N.

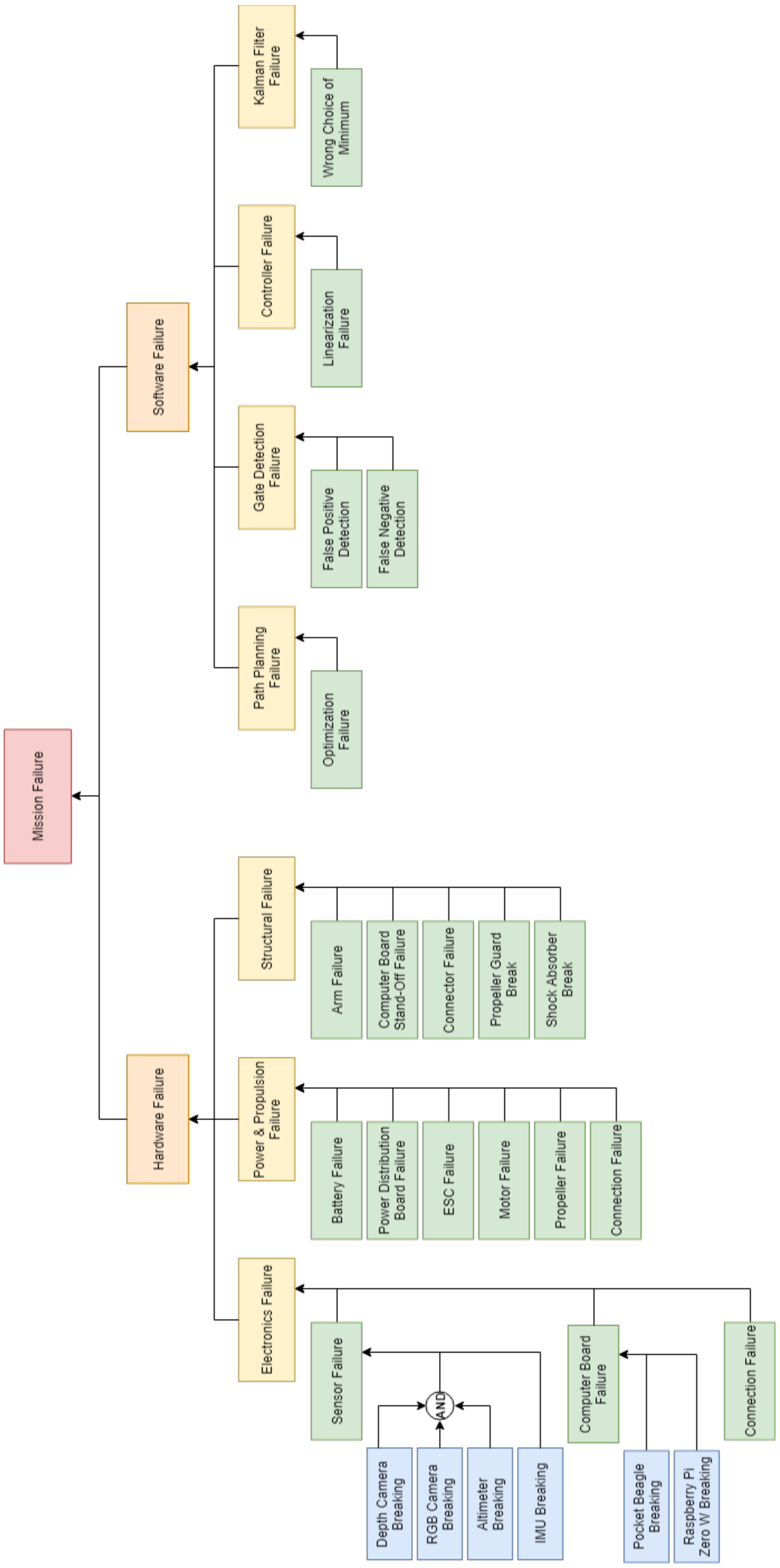


Figure 17.1: Failure Diagram

#### Software Failure

It is almost impossible to put a failure rate on the software. Although at every update, the reliability instantly decreases due to the possibility of unknown bugs being introduced, the overall trend is towards a higher reliability since any bug that is found and dealt with will not show up again. This, together with the amount of situations that the software can get into, is what makes it so difficult to estimate its reliability without programming it and testing in the real world.

#### In-Flight Reliability Summary

Although software reliability is near impossible to determine, the hardware failure rate is approximately 0.01 failures per hour. When assuming a negative exponential reliability distribution and a service life of 20 minutes, this results in a hardware reliability of 66.7%. This can be mitigated by putting a battery that has only been tested a few times on the drone before the race, since according to [61]'s data, battery failures are normally distributed. It also warrants to at least 1 extra battery being brought to the race.

### 17.1.2. Reliability due to Crashing

The drone's reliability due to crashing is impossible to quantify without actually testing it. Therefore, it will be assessed very similarly to the general risk in Chapter 18. Each identified risk will be graded on both its probability of occurrence and its severity. Severity will be rated on a scale of 1 to 5 where the numbers represent Negligible, Moderate, Significant, Critical and Catastrophic respectively. Probability of occurrence will similarly be rated from very low to very high on a scale of 1 to 5. It should be noted, however, that the scale relating the probability of occurrence score and values can be seen in Table 17.1.

Table 17.1: Probability Score to Approximate Probability Conversion

| Score | Probability |
|---|---|
| 5 | 90% < P < 100% |
| 4 | 50% < P < 90% |
| 3 | 20% < P < 50% |
| 2 | 5% < P < 20% |
| 1 | 0% < P < 5% |

#### Electronics Failure

During crashes, the main risk related to electronics failure is that of the sensors or PCBs breaking due to shock loads generated by the impact. The result of analyzing these risks with regards to the electronics can be found in Table 17.2.

Table 17.2: Electronics Risks

| | Severity | Probability |
|---|---|---|
| Pocket Beagle Breaking | 5 | 1 |
| Raspberry Pi Zero Breaking | 4 | 1 |
| Depth Camera Breaking | 3 | 2 |
| RGB Camera Breaking | 3 | 3 |
| Altimeter Breaking | 3 | 2 |
| IMU Breaking | 5 | 1 |
| Connection Failure | 3 | 4 |

Since the Pocket Beagle runs all of the algorithms that keep the drone up in the air, breaking this would be catastrophic. Although the Raspberry Pi Zero is a critical component for the functioning of the drone, it can still work without it as RAIDER would just be flying blind. Therefore breaking the Raspberry Pi Zero has been given a severity of 4. Breaking the IMU, on the other hand, would be catastrophic because the rate at which the gate detection updates the drone's pose is much too low to keep a drone in the air. All of the other risks have been given a severity of 3 because they are redundant. If 1 of them occurs, the others can take over the functionality. The reliability of the drone would just go down if this were to happen. The computer boards and IMU are well protected by the structure

and damped by their connections, resulting in a probability of 1. Although the Altimeter is mostly well protected, a part of it sticks out of the structure, resulting in a probability of 2. The cables that connect the different sensors and computer boards aren't very prone to breaking, but they can be pulled out of their sockets. Therefore, during assembly and maintenance, the team has to make sure that all of these cables are fully inserted into their respective sockets and that no cables are pulling on each other. Last, but not least, is the RGB camera, it is quite well protected by the motors from the top and the structure from the bottom, top and sides. Because the lens is relatively fragile, this resulted in a probability of 3.

### Power & Propulsion Failure

A synonym for the Power and Propulsion system is the power train. It encompasses all elements of the drone that provide and distribute the electrical energy and those convert it into thrust. The results of the reliability analysis of the power train can be seen in Table 17.3

Table 17.3: Power and Propulsion Risks

| | Severity | Probability |
|---|---|---|
| Battery Failure | 5 | 3 |
| PDB Breaking | 5 | 1 |
| ESC Failure | 5 | 3 |
| Motor Failure | 5 | 2 |
| Propeller Breaking | 5 | 4 |
| Connection Failure | 4 | 2 |

Losing any part of the power train is catastrophic for the drone, resulting in 5's all across the board.

The probability of a propeller breaking when it hits anything is very high. However, due to the fact that propeller guards have been implemented into the design, the drone can take small collisions, reducing the probability to 4. Often, when a propeller hits something, the ESC burns through. This happens a little bit less often than a propeller hitting something, resulting in a 3. The PDB is just a circuit board that will be sandwiched in between the 2 computer boards, this results in it having the same probability of breaking as the 2 computer boards. The odds of any of the connections failing is rather low. The reason for this is that all connectors in the power train have a locking mechanism in them, making it impossible to unplug them by pulling on the wires.

### Structural Failure

Some parts of the structure were designed to break. They are designed to take up as much impact energy as possible and then break, without it having any impact on the functioning of the drone. These parts are the propeller guards and the crumple zones. The risks associated with the structure are in Table 17.4.

Table 17.4: Structural Risks

| | Severity | Probability |
|---|---|---|
| Arm Breaking | 5 | 3 |
| Connection Breaking | 5 | 3 |
| Stand-Off Screw Breaking | 3 | 2 |
| Propeller Guard Breaking | 1 | 5 |
| Shock Absorber Breaking | 1 | 4 |

There are 2 ways in which an arm can break off: Firstly, the arm itself can break, and secondly, the arm's connection to the rest of the frame can break. These are the first 2 risks and they were given a severity of 5 since a quadcopter cannot fly properly with just 3 of its rotors functional. The different platforms and computer boards are connected using stand-off screws. Since 1 will be placed at each corner, losing 1 will not have a catastrophic or critical effect, resulting in a score of 3. As stated above, the propeller guards and shock absorbers are made to break. Therefore the severity of this happening was minimized, resulting in a 1.

The probability of breaking one of the propeller guards is very high since they encompass the entire outside of the drone. If it bumps into anything while in flight, these will be hit it first. The team therefore decided this would have a probability of 5. The odds of the crumple zones breaking are slightly lower because they are only on the bottom of the drone, resulting in a score of 4. If the drone were to fly into anything at top speed, most of the impact would be absorbed by the arms, even after the propeller guards break. For this reason, the probability of the arm breaking was rated to be a 3. Although the connections were designed to be stronger than the arms themselves, the difference is not high enough to warrant a full point of lower probability, resulting in a score of 3. Finally, the stand-off screws have a low probability of breaking because they are designed for this, seeing as they are quite difficult to replace.

### Software Failure

As long as all of the hardware stays intact, the software is not at risk of breaking. Therefore no further analysis was done on the reliability of the software during a crash.

### Crash Reliability Summary

In summary, the largest risks are listed in order of probability in Table 17.5. Especially these should be taken into account when planning for availability, maintenance, safety, and logistics. The risks that were detailed here will be expanded upon in Chapter 18.

Table 17.5: Most Prominent Crash Reliability Risks

| | Severity | Probability |
|---|---|---|
| Propeller Breaking | 5 | 4 |
| Arm Breaking | 5 | 4 |
| Pocket Beagle Breaking | 5 | 3 |
| Battery Failure | 5 | 3 |
| PDB Breaking | 5 | 3 |
| ESC Failure | 5 | 3 |
| Structural Connection Breaking | 5 | 3 |

## 17.2. Availability

Although RAIDER only has to be ready to fly for 20 minutes during the IROS2018 competition, it should still have an availability as high as possible. There will, however, only be 1 production unit of the drone. The result being that anything that might break has to be ready to be placed during the race. These replacements have to be done as quickly as possible in order to allow RAIDER to continue trying to finish the course. The components from which the drone is made are all off the shelf products, allowing the team to easily get replacement parts whenever they are needed.

In order to assess the availability and maintainability of the drone, a time per run of 1 minute was chosen. This means that if no maintenance or repairs have to be performed, the time between subsequent take offs is 1 minute. At first, the total flight time of 20 minutes is assumed, after which the total flight time is divided by he run time to calculate the number of tries the drone has to finish the track.
Then, taking into account the probabilities of each failure happening and which repairs have to be performed after each attempt, the total maintenance time is calculated. However, this maintenance time has to be added on top of the the total flight time, since the clock is not stopped during repairs. Which results in a total time of over 20 minutes. Therefore, the process has to iterate, taking the allotted time of 20 minutes and subtracting the previous iteration's maintenance time for the total flight time. This can be seen in Equation (17.2) for iteration k.

$$(\text{Total Flight Time})_k = \text{Total Allotted Time} - (\text{Maintenance Time})_{k-1} \tag{17.2}$$

This flight time is then again divided into runs, making the process iterative. The final iteration of this script can be seen in Appendix B. The resulting values can be found in Table 17.6. And the probabilities of each failure, together with the total number of replacement parts will be explained in Section 17.3. Note that the total flight time is not a multiple of 1. This is the case because an optimum was found at 12 runs, or 12 minutes. The extra time allows for inaccuracies in the estimation methods for both flight and repair time.

Table 17.6: Results of Availability and Maintenance Analysis

| Variable | Value | Unit |
|---|---|---|
| Total Time | 19:36 | [Minutes : Seconds] |
| Total Maintenance Time | 6:54 | [Minutes : Seconds] |
| Number of Runs | 12 | [-] |
| Availability | 64.8 | [%] |
| Unit Cost | 1220 | [€] |
| Replacement Parts Cost | 775 | [€] |
| Total Cost | 1995 | [€] |

## 17.3. Maintainability

Because RAIDER only has 20 minutes to win the race, it has to be as maintainable as possible. Since making sure that all components can be replaced as quickly as possible allows the drone to be in the air quickly after a crash, this was one of the main priorities during the structures subsystem design. Since the drone will not function for a long time, it does not have to be maintained.

The estimated probabilities for each failure in Section 17.1 are shown in Table 17.7. These were used to calculated the amount of replacement parts that have to be brought to the IROS2018 competition.

Table 17.7: Assumed Probability of each Failure

| **Failure Type** | **Probability Rating** | **Probability** |
|---|---:|---:|
| Propeller Guard Breaking | 5 | 95% |
| Electronic Connection Failure | 4 | 80% |
| Propeller Breaking | 4 | 80% |
| Shock Absorber Breaking | 4 | 80% |
| Arm Breaking | 3 | 40% |
| ESC Breaking | 3 | 40% |
| RGB Camera Breaking | 3 | 20% |
| Battery Failure | 3 | 20% |
| Structural Connection Breaking | 3 | 20% |
| Depth Camera Breaking | 2 | 5% |
| Altimeter Breaking | 2 | 5% |
| Power Connection Breaking | 2 | 5% |
| Stand-Off Screw Breaking | 2 | 5% |
| Motor Failure | 2 | 5% |
| Pocket Beagle Breaking | 1 | 1% |
| Raspberry Pi Zero Breaking | 1 | 1% |
| PDB Breaking | 1 | 1% |
| IMU Breaking | 1 | 1% |

The amount of replacement parts was calculated using the script that was described in Section 17.2 and the results can be found in Table 17.8. These parts sum to a total cost of €774.31. In order to get to these numbers, a few things were changed from the raw probabilities.
Firstly, due to the fact that there are 4 propellers on the drone, the amount of replacements was multiplied by 2. The value of 2 was chosen because the drone would hit 2 propellers if it were to fly into a wall. Secondly, it is assumed that in the event of an arm breaking off, half of the motors and ESCs can be dismounted and reused. Finally, the parts that will not have any replacements are not listed in Table 17.8.

Table 17.8: Total Amount of Replacement Parts

| **Part** | **Amount** | **Approximate Unit Price [€]** | **Total Parts Price [€]** |
|---|---:|---:|---:|
| Propeller Guard | 22 | 0.18 | 3.96 |
| Propeller | 20 | 0.25 | 5.00 |
| Shock Absorber | 10 | 0.10 | 1.00 |
| Arm | 7 | 3.00 | 21.00 |
| Motor | 4 | 15.00 | 60.00 |
| ESC | 8 | 15.00 | 120.00 |
| RGB Camera | 2 | 76.50 | 153.00 |
| Battery | 3 | 30.00 | 90.00 |
| Nuts & Bolts | 2 | 0.10 | 0.20 |
| Depth Camera | 1 | 300.00 | 300.00 |
| Altimeter | 1 | 35.00 | 35.00 |
| Stand-Off Screw | 1 | 0.15 | 0.15 |

## 17.4. Safety

Safety analysis can be done by looking at the Failure Diagram, Figure 17.1, and determining which failures could compromise RAIDER's safety. After this is done, countermeasures can be taken to reduce the probability and impact of the hazards.

The failure modes that are grouped into electronics failure could all cause the drone to become uncontrollable and crash. While the drone is uncontrollable, it could easily fly into spectators, competitors or any of the surroundings, possibly injuring or damaging them on impact. These injuries and damage can be caused by both the impact itself or by the propellers.
Although many of the power and propulsion failure modes would also cause the drone to become uncontrollable or fall out of the air, a short circuit is also possible. Furthermore, LiPo batteries are know to start a fire when they are punctured. Both battery punctures and short circuits can cause fires, resulting in the second group of safety hazards.
The structural failure modes would all make the drone uncontrollable, so these will be discussed in the Impact Hazard section.
Due to a bug in the software, the drone could start to behave erratically, adding to the impact hazard.

## 17.4.1. Impact Hazard

There are a few ways in which the impact of the drone can cause injuries or damage. Firstly, the impact itself is dangerous. Secondly, a propeller can cut through many things, which causes injuries. Thirdly, upon impact, it is possible for parts of the drone to break off and cause harm by impacting something that way.

### Direct Impact

How dangerous an object is upon impact can be determined in multiple ways. One of the main ways in which this can be done is by calculating the Head Injury Criterion (HIC). The $\text{HIC}_{36}$ can be calculated using Equation (17.3) [7], where $V_{Head}$ is calculated using the maximum linear momentum of the drone.

$$\text{HIC}_{36} = 0.0036 \left\{ \frac{V_{Head}}{0.0036} \right\}^{2.5} \tag{17.3}$$

RAIDER's maximum linear momentum is 30 $\frac{kg \cdot m}{s}$ at its top speed of 160 km/h. Assuming that a head weighs 5kg and that all of the linear momentum is transferred to the head, results in a value for the $\text{HIC}_{36}$ of 2295. Converting this to the probability of AIS 2, 3, and 4 injuries was done using [26]. During the IROS2018 competition, however, the expected maximum speed will not exceed 2 m/s. Taking a safety margin of 100% of this, results in a speed of 4 m/s or 14.4 km/h. At this speed, the same calculations can be done, with a resulting $\text{HIC}_{36}$ of 1. The probabilities of each type of injury for both 165 km/h and the 14.4 km/h flight can be found in Table 17.9.

Table 17.9: Probability of Injury Type

| AIS Rating | Type of Injury | Probability at 165 km/h | Probability at 14.4 km/h |
|---|---|---|---|
| 2 | Recoverable | 82.0% | 0% |
| 3 | Possibly Recoverable | 65.1% | 0% |
| 4 | Not Fully Recoverable without Care | 55.4% | 0% |

The team chose $\text{HIC}_{36}$ over the more strict $\text{HIC}_{15}$ because the first point of a collision will most likely be one of the drone's arms, which are designed to soften the impact. This would reduce the drone's linear momentum significantly.

### Propeller Impact

Propeller tips are often very sharp. Combining this with the fact that they can reach speeds in the order of mach 0.6, shows just how dangerous propellers can be. The IAS rating of the lacerations that are caused by drone propellers is a function of for instance how sharp the tip is and the tip velocity. Such a relation can be seen in Figure 17.2 [1].
Looking at Figure 17.2, it quickly becomes evident that Carbon Fiber Flat Tip propellers consistently cause the lowest amount of damage. Plastic Sharp Tip propellers, on the other hand, on average do the highest amount of damage.RAIDER's 5 inch propellers can reach speeds of up to 33000 RPM. At this speed, the propeller tips reach velocities

of 220 m/s or 720 ft/s. This, according to Figure 17.2, results in injuries of about IAS 2 for carbon fiber reinforced plastic flat tipped propellers. These are very similar to the Glass Fiber Reinforced Plastic (GFRP) propellers that will be on RAIDER, which is why this material was chosen in Chapter 6.

### Drone Debris

Although the drone itself can cause significant harm when it directly impacts anything, if a propeller were to shatter, the debris would have a much higher velocity and could therefore cause a lot of damage as well. At the moment a propeller shatters, the blade that is moving forward can have a speed of up to the sum of its tip speed and the drone's

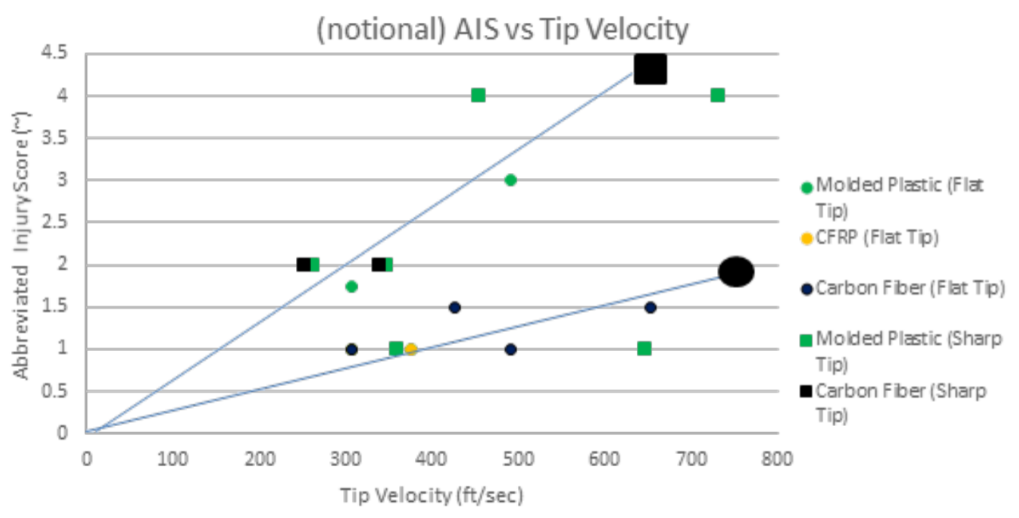


Figure 17.2: IAS Rating as a Function of Tip Velocity [1]

speed. This results in a total velocity of approximately 265 m/s. However, due to the large drag at such a high velocity, it will slow down very quickly. Especially the smallest pieces will decelerate very quickly since they have a tiny mass.

The Puncture Fracture Toughness of human skin is approximately 24 kJ/m$^2$ [23]. If a piece of the propeller containing more energy per area hits the skin, it will pierce it. The safe range from the drone is calculated using this. It is assumed that the propeller breaks in chord wise direction. The break can be at any distance from the center hub. The propeller was modelled as a flat plate with a thickness and chord length of 1 and 10 mm respectively. Each piece has a length from the fracture to the propeller tip and starts with the average velocity between both of its ends. The simulation then takes a drag constant of 1.2 and runs until the specific kinetic energy drops below 24 kJ/m$^2$. The distance at which each piece becomes safe was recorded and can be seen in Figure 17.3.

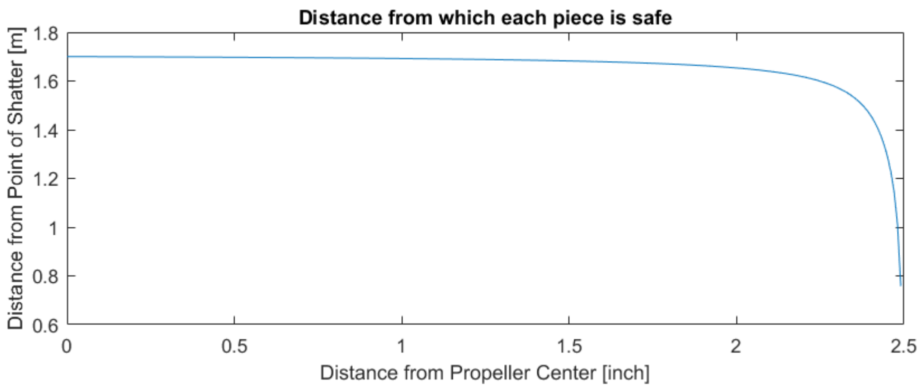


Figure 17.3: Range at which Propeller Pieces can pierce Human Skin

The mass of each piece was approximated using the density of a 50 volume percent glass fiber, 50 volume percent polypropylene GFRP. The resulting density is 1.7265 g/cm$^3$ [16]. The end result of this model is almost exactly 1.7 meters. This means that up to 1.7 meters from the place where the propeller shattered, the pieces could pierce skin and cause lacerations.

### 17.4.2. Fire Hazard

There are 2 potential sources of fire aboard RAIDER: The battery and the electronics. Lithium Polymer batteries are known to burst into flames when they are discharged too quickly or pierced. The electronics can cause fires when too high a current is ran through them, this could be due to a short circuit, but also during normal operation. The fact that it is commonplace to have solder on FPV race drones melt slightly during races clearly shows this fact.

### 17.4.3. Countermeasures

### Impact Hazard

According to FAA regulations, the highest allowable chance of an AIS 3 rated injury occurring during the operation of a drone over people is 30% [1]. This shows just how dangerous fast flying drones are and that clear safety precautions have to be taken. These precautions are the kill switch that one of the operators will be in charge of controlling. If this switch is flicked, the drone will immediately power down and drop out of the air. Because this is not ideal since it can still cause damage to the surroundings, the drone will also have a safe mode. This safe mode will be activated when an error in the software, low battery voltage or an overheated component are detected. Safe mode will immediately land the drone autonomously.

In order to reduce the impact energy upon collision, propeller guards were implemented on the drone. These, together with the arms that were designed to soften the impact in case of a collision should spread out impact as much as possible, making the drone much safer. The propeller guards also reduce the risk of lacerations due to the propeller blades hitting a person. Although they do not help much at high speeds, at the expected typical race speed of 14 km/h they will hold up and make the drone bounce harmlessly off any obstacles.

The propeller guards also increase the drone's safety when a propeller shatters. If the drone crashes into anything at a high enough speed to break off the propeller guard, very shortly after that, the propeller will break as well. This means that the propeller guard is still almost in the same place, allowing it to slow down any debris that hits it. The kill switch also increases the safety with respect to this hazard because the 1.7 meters of maximum dispersion is at a motor speed of 33000 RPM. By hitting the switch, the propellers will slow down quickly, quickly reducing the distance at which the propellers could be dangerous.

### Fire Hazard

The probability of the battery getting punctured can be massively decreased by putting it in a place where it is well protected by the structure. The top of the drone, where the battery on RAIDER is, is one of these places. From the bottom, the battery is protected by the structure and from the top , the motors keep it off the ground. The drone's safe mode protects it from getting too hot. If the drone measures too high a temperature near the battery, it will land as quickly as possible, giving it time to cool down again.

In the event of the drone starting to smoke and the safe mode not working, the operator can also kill the drone. This stops the drone from becoming an uncontrollable fireball. When the structure was designed, the team also took into account that the electronics should have enough airflow over them to cool them. This, together with over current protection on the ESCs, massively reduces the probability of the electronics catching fire due to over currents.

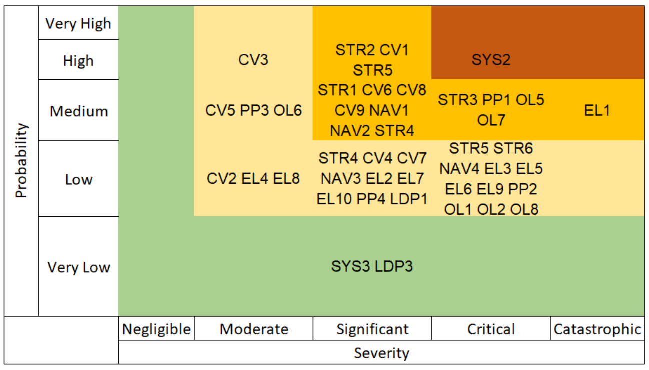
# 18 Risk Assessment and Mitigation

While the application of experimental technology offers the potential of significantly enhanced capabilities, it can also lead to excessive delays and cost blow-outs. This chapter mainly focuses on the potential hazards of each specific subsystem in order to identify the Achilles' heel of the design. A risk analysis of the overall design will follow, mainly considering the integration of each subsystem. Finally all the risks will be analyzed in terms of their cause and effect followed by a mitigation strategy.The risks have been evaluated with regard to their probability of occurrence and the severity of their consequences. To do so, a standardized method has been used where both categories have been divided in five levels [57]. After the evaluation, the risks have been depicted in a risk map to effectively visualize the different hazards as risk is the combination of the likelihood of occurrence and the severity of the consequences. The closer a risk is to the top right corner of Table 18.1, the more resources have to be put into mitigating this risk.

Table 18.1: Risk quantification parameters

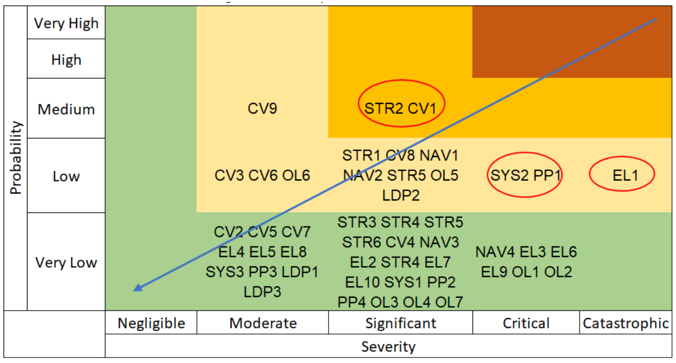
| Level | Probability | Level | Severity | Description |
|---|---|---|---|---|
| 5 | Very High | 5 | Catastrophic | Mission failure |
| 4 | High | 4 | Critical | Mission success questionable |
| 3 | Medium | 3 | Significant | Decrease in performance |
| 2 | Low | 2 | Marginal | Secondary mission compromised |
| 1 | Very Low | 1 | Negligible | Small reduction in performance |

The severity of the risk has been selected based on the consequences of this risk on the Project Objective Statement. The secondary mission is intended as racing as fast as possible, exceed the expectations regarding autonomous races. Although this is still an important aspects of the design, it does not need to be met in order to win the race. The probability has been discretized in five different levels which do not have specific quantifiable ranges due to the uncertainties that are difficult to model or predict [65]. Due to the lack of data the probability levels give an indication of the plausibility of an event occurring, based on the engineering knowledge of the experts and stakeholders, and the insight that the designers have developed. These estimations are aided by specific analysis conducted in Chapter 17 and Chapter 3, which give a top-down approach to analyze the mission in general, and the subsystems designs in Part II to investigate the details of each component that were sized for specific extreme cases.
The different risks have been grouped by subsystem, in order to identify the most critical design aspects and divide the mitigation approaches that have been undertaken by each department, all of the risks concerning the design have been implemented after mitigation while the ones that cannot be integrated due to their natural cause represent an inherent risk of the drone which are mitigated with specific suggestion to undertake when the drone is operated. The abbreviation STR is used for the structural subsystems, EL for the electrical subsystems including the different sensors, NAV indicates the navigation subsystem which comprises the path planning and the control subsystem. Moreover, CV stands for computer vision, PP for power and propulsion, SYS for system integration and OL for operations and logistics. The risks identified for each subsystem can be found in Table 18.2.
A quantitative evaluation of the potential weaknesses of the design has been reported in the same tables and was be visualized in Figure 18.1, based on the probability of occurrence and the impact it may have on the performance, schedule and cost [18]. These three parameters have been weighted equally when quantifying the risks. The risks listed on the table were derived mainly from the functional analysis and breakdown of the different tasks the drone was to perform. By analyzing the different stages of the operational life of the drone it was possible to get an overview of the drone's functions to identify which of these were more vulnerable. Moreover, the subsystem interface analysis was used to display the interconnections which highlighted the most important elements in the software and the payload. The quantification of the risk, has been done based on experimental data from similar projects, and interviews with experts and stakeholders. Additionally the judgment of the designers of each specific subsystem provided valuable insight into different areas for the evaluation of risks. Risk analysis is an iterative process where the risk mitigation or abatement, if possible, is continuously performed, to achieve the minimum risk level. Due to the application of risk mitigation strategies, elements of the risk map shift towards the bottom left corner, as indicated by the arrow in Figure 18.2. Most of these risks are inherent to their process and cannot be completely removed,



| Probability | | | | | |
|---|---|---|---|---|---|
| Very High | | | | | |
| High | | CV3 | STR2 CV1 STR5 | SYS2 | |
| Medium | | CV5 PP3 OL6 | STR1 CV6 CV8 CV9 NAV1 NAV2 STR4 | STR3 PP1 OL5 OL7 | EL1 |
| Low | | CV2 EL4 EL8 | STR4 CV4 CV7 NAV3 EL2 EL7 EL10 PP4 LDP1 | STR5 STR6 NAV4 EL3 EL5 EL6 EL9 PP2 OL1 OL2 OL8 | |
| Very Low | | | SYS3 LDP3 | | |
| Severity | Negligible | Moderate | Significant | Critical | Catastrophic |

Figure 18.1: Risk Map before Mitigation



| Probability | | | | | |
|---|---|---|---|---|---|
| Very High | | | | | |
| High | | | | | |
| Medium | | CV9 | STR2 CV1 | | |
| Low | | CV3 CV6 OL6 | STR1 CV8 NAV1 NAV2 STR5 OL5 LDP2 | SYS2 PP1 | EL1 |
| Very Low | | CV2 CV5 CV7 EL4 EL5 EL8 SYS3 PP3 LDP1 LDP3 | STR3 STR4 STR5 STR6 CV4 NAV3 EL2 STR4 EL7 EL10 SYS1 PP2 PP4 OL3 OL4 OL7 | NAV4 EL3 EL6 EL9 OL1 OL2 | |
| Severity | Negligible | Moderate | Significant | Critical | Catastrophic |

Figure 18.2: Mitigated Risk Map

but their consequences can be alleviated by proper planning and communication between the design departments. Most of the mitigation strategies were formulated by evaluating other options within the same branch of the design tree that the risk event belonged to, and combining feasible solutions. The analysis performed in this chapter delineates the risks that represent a major weakness in the design. These risks can be identified in the risk map posterior to the mitigation. The risks are STR2, CV1, SYS2, PP1, and EL1 as highlighted with the red circles in Figure 18.2. A detailed description of all the other risks and their mitigation strategies can be found in the tables at the end of the chapter.

The risk STR2 states that propeller failure is possible during the use of the drone. The propellers can break under the centripetal forces it experiences during flight and, in case of a crash, it is likely that the propeller will break due to the impact. This can be regarded as favourable and unfavourable at the same time. If the propeller breaks, most of the impact energy is taken up by the breaking of the propeller and is not transferred to the frame or the motors, which prevents structural or motor damage. On the other hand, it is not favourable if the propellers need to be replaced whenever the drone experiences a minor crash. To lower the likelihood of this risk, propeller guards have been added. Moreover, the risk identified as PP1 highlights the possibility of breaking a motor due to excessive use at maximum power which would lead to burning the motor itself and a complete loss of stability, which results in the drone losing the capability of flying. To mitigate this risks popular FPV motors have been used, these allow the drone go much faster than the maximum speed as required by the customer, reducing the average throttle that is used. The software risks concerning the computer vision (CV1) was determined to be in a critical region. If the gate is not detected, the path cannot be corrected, and the drone will constantly be looking for a gate or will move without having a correct pose estimate. To be more specific, this risk is about the chance of not identifying the gate in an image taken by the cameras and estimating the distance at which the gate is located from the drone. This risk can be mitigated by adding an additional sensor to provide the drone with depth information and introduce an extra degree of freedom in favour of the drone that can rely on different strategies for the same purpose of detecting a gate. The IMU drift integration error (SYS2) is very important, as has been mentioned previously in Chapter 7, because the consequence in the navigation strategy would be fatal. Therefore, to mitigate this risk the best IMU was chosen from the market which is portrayed in the price of the sensor that exceeds the cost of normal IMU by over 100 times. Apart from the kalman filter that the IMU has integrated on-board, an unscented kalman filter has been included in the data-handling section for sensor and software integration to increase propagation accuracy of the pose.

Finally, the EL1 risk states that an electronic speed controller (ESC) might fail. This can be the case when the drone is flying at high performance. Breakdown can be prevented by appropriately sizing the ESCs with the proper rating. Also, by ordering high quality ESCs from reliable manufacturers, and by having the ESCs properly positioned for maximum cooling, the risk of ESC failure is reduced. This risk has been identified thanks to the contribution of multiple drone amateur's that reported it in different occasions.

Table 18.2: Risk analysis per subsystem

## Computer vision subsystem

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| CV1 | Gate is not detected | Lighting condition or too much noise in the background | Path cannot be corrected | 4 | 3 | Use two cameras with two different algorithms during the race: CNN and Depth images analysis | 3 | 3 |
| CV2 | Gate detection processing takes too long | Too much information to process | Detection is not useful | 2 | 2 | Set maximum processing time and fly slower | 1 | 2 |
| CV3 | Not all corners are identified | View on the gate is not optimal | True negative detections is observed | 4 | 2 | If three corners are detected in the shape of a gate keep searching in same region | 2 | 2 |
| CV4 | Object width is not estimated correctly | Time of flight camera calibration | The drone might fly too close to the object and collide | 2 | 3 | If object is not hollow (possible collision) choose a random point with a significance radius from obstacle | 1 | 3 |
| CV5 | Gate contour is only partially detected by the algorithm | Background has too many features | Algorithm detects corners where there are none | 3 | 2 | Optimal layered structures with sufficient filters and features extraction for gate recognition | 1 | 2 |
| CV6 | Drone images are too blurry | Camera FPS is too low or drone is flying too fast | Impossible to extract correct features from image | 3 | 3 | Fly slower and use both ToF and RGB camera | 2 | 2 |
| CV7 | Synthetic images do not reflect reality | Background is smooth and gate is not properly rendered | Algorithm is not tuned for real race environment | 2 | 3 | Distort images, use real, detail rich background images | 1 | 2 |
| CV8 | False positive detection | Algorithm identifies shape of a gate using background waypoints | Drone flies in wrong direction | 3 | 3 | Use virtual ToF to ensure gate is close and gate waypoints to verify direction | 2 | 3 |
| CV9 | Gate color is changed by organizers | Late agreements to facilitate participants | Algorithm need to be retuned and re-trained | 3 | 3 | Develop easily tunable, modular algorithms | 3 | 2 |

## Structure subsystem

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| STR1 | Arm damage due to impact | Hitting an obstacle or gate side | Destabilization and impact with the ground | 3 | 3 | Detachable arms that snap off upon impact | 2 | 3 |
| STR2 | Propeller damage | Impact due to collision | Impossible to generate lift | 4 | 3 | Add a propeller casing | 3 | 3 |
| STR3 | Deformation of the frame | Impact due to collision | Load transmission to fragile components | 3 | 4 | Insert shock absorbers as the legs | 1 | 3 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| STR4 | Bolt shear-out | Bending moment created by the motors | Significant deformation in carbon fibre arm and replacement is needed | 2 | 3 | Add carbon inserts at critical bolt connections | 1 | 3 |
| STR5 | Structural damage | Eigenfrequency of structure is similar to frequency induced by motors | Sensor readings are inaccurate and destabilization | 2 | 4 | Structure was made more rigid and carbon fibres lay-up chosen to be as isotropic as possible | 1 | 3 |
| STR6 | Assembly failure | Component loss and compromised stability | Frame does not provide sufficient support to all subsystems | 2 | 4 | Safety factor of 1.5 was used for a number of connections during sizing | 1 | 3 |

**Navigation subsystem(incl. Path planning and Control)**

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| NAV1 | Stabilization is too hard to achieve | Assumptions for drone dynamics don't reproduce real model | Drone is unstable | 3 | 3 | Validate results with real simulators (eg. Ecalc) | 2 | 3 |
| NAV2 | Path planned is not dynamically feasible | Too much thrust required | Path cannot be followed | 3 | 3 | Dynamic constraints of the drone (maximum thrust as a function of V) incorporated | 2 | 3 |
| NAV3 | Path is not controllable | Required maneuvers too aggressive | Path cannot be followed | 2 | 3 | Minimum snap optimization for controllability - generated trajectory minimizes control effort | 1 | 3 |
| NAV4 | Path planning optimization routine does not converge | Numerical issues encountered in optimization routine (near-singular matrices, etc.) | No path planned | 2 | 4 | Optimization method was validated to be numerically stable for 15 path segments and small trajectory times | 1 | 4 |

**Electronic subsystem**

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| EL1 | ESC failure | Overheating due to overpowering | Motor shuts off, not enough power followed by crash | 3 | 5 | Rated current of the ESC is chosen based on critical magnitude of current | 2 | 5 |
| EL2 | Flight time is less than 10 minutes | Battery discharges quicker than expected or is not fully charged | Significant delays on the race or not finishing the track | 2 | 3 | Large contingency for battery selection and monitor battery charge | 1 | 3 |
| EL3 | IMU failure | Sensor damage or Timing or calibration error | Impossibility to orient drone | 2 | 4 | Use IMU protective casing | 1 | 4 |
| EL4 | Ultrasound sensor failure | Damaged sensor | Impossibility to retrieve Drone Altitude | 2 | 2 | Protect sensor and ensure sensor quality | 1 | 2 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| EL5 | Failure of the camera | Impact due to collision | Camera should be replaced | 3 | 3 | Encase camera in protective structure | 1 | 3 |
| EL6 | Camera failure | Connection/Component failure | Failure to recognize gate | 2 | 4 | Protect camera and use second visual sensor | 1 | 2 |
| EL7 | Battery catches fire | Overheating of the system or short circuit | Safety of the drone and spectators is compromised | 2 | 4 | Voltage and temperature monitoring | 1 | 4 |
| EL8 | Computer overheats | Excessive computer use without sufficient cooling | Failure to control the drone and possible impact | 2 | 3 | Use temperature sensors to enter safe mode | 1 | 3 |
| EL9 | Receiver and transmitter failure | Defective connection or signal interference | Impossible to monitor drone status | 2 | 2 | Test connections and position Tx/Rx in zones with sufficient coverage and enter in safe landing mode at error | 1 | 2 |
| EL10 | Camera signal is not received | Connection failure or incorrect format | No view of gate and thus no path | 2 | 4 | Use a tried-and-tested camera and use second one | 1 | 4 |

**Power and Propulsion subsystem**

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| PP1 | One motor fails | Motor burns out | Drone loses stability and suffers major damage | 3 | 4 | Limit period of time at which motor max power is used and testing before competition | 2 | 4 |
| PP2 | Propeller debris hits a person | Impact due to collision | Safety of spectator is compromised | 2 | 4 | Add propeller guard and sent kill command to reduce motor rpm | 1 | 3 |
| PP3 | Propeller guard interfere with blade motion | Guard is bent or detached | Propeller damage | 3 | 2 | Easily detachable propeller guards | 1 | 2 |
| PP4 | Enter in vortex-ring state in vertical flight | High propeller pitch | Stall of the blade | 2 | 3 | Test severity lift loss and use three blade prop if severity is extreme | 1 | 3 |

**System integration**

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| SYS1 | On board computer is not able to process all the data for the three main programs | Clock speed is too low and processor overflows | Throughput time is too long, drone is not able to detect a gate effectively. Plan the path and control at the same time | 3 | 3 | Implement two computers one for gate detection and one for path planning and control | 1 | 3 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| SYS2 | Excessive IMU noise | Vibrations of the structure | Imprecise measurements. | 4 | 3 | Mechanically filter noise by using a damper between the IMU and the chassis | 2 | 3 |
| SYS3 | Excessive IMU drift | Integration error due to lack of gate detection for absolute pose | Drone pose estimates are incorrect | 4 | 4 | Choose accurate IMU and implement UKF to increase propagation accuracy of pose | 2 | 4 |
| SYS4 | Motor performs suboptimally | Data provided by the manufacturer is not accurate | Maximum speed is not reached | 1 | 3 | Use redundancy when choosing a motor to meet performance requirement | 1 | 2 |

**Operations and Logistics**

| ID | Risk | Cause | Effect | LH | SV | Mitigation | NLH | NSV |
|---|---|---|---|---|---|---|---|---|
| OL1 | Damage during transport | Insufficient transport protection | Drone damaged may not participate in the race | 2 | 4 | Include measures of safe transportation the design phase of the drone | 1 | 4 |
| OL2 | Drone gets stolen during transport | Thief takes the drone | Impossible to participate in the race | 2 | 4 | Ensure safe transportation, with the drone secured against theft | 1 | 4 |
| OL3 | Drone batteries are not swapped in time | Bad maintenance during race | Batteries discharged too much and quality degrades | 3 | 3 | Install a low voltage monitor/alarm in the drone | 1 | 3 |
| OL4 | Drone replacement parts are not enough | More parts of the same type break multiple times in the race | Expensive to repair the drone and stop racing | 3 | 3 | Taking into account cheap replacement parts during design phase | 1 | 3 |
| OL5 | Drone is damaged during testing | Collision while experimenting novel software techniques | Drone is damaged | 3 | 4 | Add crumple zones in the drone and test in low damage environment (foam covered ground) | 2 | 3 |
| OL6 | The test environment does not simulate the race closely | Drone design and training is done with different gates and different configurations | Software is not optimal during competition | 3 | 2 | Use CAD of the IROS circuit to reproduce race environment | 2 | 2 |
| OL7 | Drone crashes into a person | Bad safety measures in drone design | Person is injured and drone is damaged | 3 | 4 | Enter safe mode in case of emergency | 1 | 3 |
| OL8 | Propeller debris hits a person close-by | Improper safety measures integrated in the design | Person is injured | 2 | 4 | Implement safety measures in the drone (propeller guards) and to the environment in which the drone is racing (nets) | 1 | 3 |

# 19 Sustainable Development Strategy

In this chapter the strategy to sustainable development is defined. A sustainability objective statement is introduced to make the group's intention on this matter explicit: *Win the IROS 2018 competition by designing an autonomous race drone with 9FTE, who strive to adhere with the ideals of sustainability.* In Section 19.1 the approach to sustainable development is analyzed. Section 19.2 describes how to assess the sustainability of a design using the Life cycle Design Strategies (LiDS) wheel and how it was applied to the final design.

## 19.1. Approach to Sustainable Development

An analysis of what sustainability development entails is proposed. In this way the strategy taken to assess the sustainability of a product can be understood better. In this section different aspects of sustainability will be discussed further.

- **Holistic Approach:** Holistic approach means balancing social, environmental, and economic aspects into the project. It is clearly summarized in Figure 19.1. An active participation of stakeholders is fundamental to balance the different parts.

- **Institutional Scale:** Many institutions provide guidelines such as ISO 26000, ISO 14001 and the "Sustainability Reporting Guidelines" for organizations. In this way they can assess their level of sustainability. For this project the Eco-Management and Audit Scheme (EMAS) is used, because it focuses more on the environmental aspects, which are considered more critical. These guidelines were released in August 2017 by the European Union and are based on the ISO 14001.

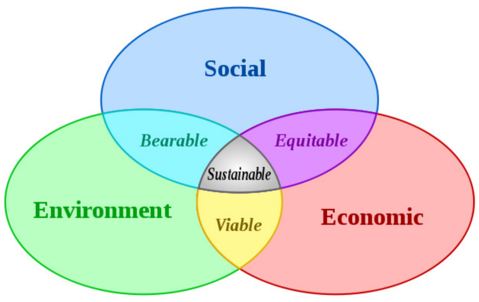


Figure 19.1: The interrelations of social, environmental, and economic aspects[49].

- **Long-Term Orientation:** It means that the decisions that are made need to take into account the needs of future generations. The evolution of the product over time needs to be considered. This translates to a higher complexity in the design process because of the increased number of uncertainties [17].

- **Risk Reduction:** A precautionary principle needs to be examined because of the complexity, uncertainties, irreversibility and non-linearity of the project [17]. Concerning the impact of the project on the environment and society, prevention is more efficient than correction. For instance, the use of natural resources needs to be sustainable, in the sense that it should not exceed the rate at which they are renewed. Another example is that the waste production should not exceed the absorptive capacity of the environment [49].

- **Ethical Values:** Sustainable development also translates to ethical values to which stakeholders and engineers should adhere. These key values are equity, respect for the people's life that are changed by the use of the product, and environmental sustainability.

- **Participation:** In order to advance in sustainable development, all members of the project, from stakeholders to engineers, should be involved. They need to define the problems, design possible solutions, collaborate to implement them, and monitor and evaluate the outcome [17].

## 19.2. Sustainable Design

Sustainability is an aspect of the design that should be considered from the beginning of the process. In fact, it is proven that considering this aspect only at a later stage causes a steep increase in cost. One of the reasons might be the need to comply with a certain unforeseen regulation, which causes the design to change considerably[1]. This

[1] `http://www.solidworks.com/sustainability/` [Visited at May 5 2018]

section will be used during the design process as a guideline to assess the sustainability. For this purpose a LiDS wheel is used. This tool was developed as a part of the United Nations Environment Programme by Hans Brezet and Carolien van Hemel Brezet[2]. Considerations on sustainability regard the full lifetime of the drone , which is divided into: selection of materials, manufacturing, optimization of lifetime, operational phase, and end-of-life. Each element is given a level of sustainability from 0 to 4. The meaning of each level is described in this section and the sustainability of all top-level concepts is analyzed.

### Selection of Materials

Material selection has a high relevance in the sustainability strategy.
**Level 0:** Materials used are hazardous, exhaustible or non recyclable.
**Level 1:** Materials used are non-hazardous, exhaustible and non recyclable, which means that the materials are safe to work with and their use does not harm the environment.
**Level 2:** The drone is made of non-hazardous, exhaustible and recyclable materials.
**Level 3:** The drone is made of non-hazardous, mainly non-exhaustible and recyclable materials. In this case the material is renewable, which means that it will be available for future generations. The production of the material does not require an elevated amount of energy.
**Level 4:** The drone is made of recycled, non-hazardous and non-exhaustible materials. The production of the material does not require an elevated amount of energy.

Given that one of the stakeholder requirements focuses on the recyclability of the drone, it means that the design should aim to be at least of **Level 2**.

### Manufacturing

The way in which the drone is manufactured is a fundamental aspect that needs to be taken into account in the life-cycle of the drone.
**Level 0:** The manufacturing technique used is highly harmful to the environment. It uses a high amount of energy and hazardous consumables. In addition, it produces an elevated quantity of waste.
**Level 1:** The manufacturing technique used is harmful to the environment. It uses a high amount of energy and few hazardous consumables. In addition, it produces an elevated quantity of waste.
**Level 2:** The manufacturing technique used does little harm to the environment. It uses a moderate amount of energy and no hazardous consumables. In addition, it generates a limited amount of waste.
**Level 3:** The manufacturing technique used is not fully optimized, and it still uses many production steps. It uses a minimal amount of energy, possibly coming from a renewable power source, and it uses no hazardous consumables. In addition, it generates a limited amount of waste.
**Level 4:** The manufacturing technique used is optimized to minimize the production steps. It uses a minimal amount of energy coming from a renewable power source, and it uses no hazardous consumables. In addition, it generates a limited amount of waste, which can be recycled and reused.

There are no specific requirements imposed by the stakeholder for this lifecycle phase. For this reason the acceptable level is set to **Level 2**. In this case the manufacturing technique is not optimal, but it does not lead to excessive harm for the environment and it is safe for the workers.

### Optimization of lifetime

The optimization of lifetime refers to the fact that the lifespan of the object should be maximized. Therefore repairability, durability and reliability should be maximized.
**Level 0:** The drone is not able to perform the task for which it is designed.
**Level 1:** The drone is able to perform its task, with high success rate. Reliability of the race drone plays a crucial role, as it should not fail during the race. In addition, the lower the need for repairs, the lower the use of materials and consumption of resources.
**Level 2:** The drone is able to perform its task, with high success rate, and it is repairable. Repairability leads to lower operational costs. In fact, if the drone was not repairable, it would mean that if it breaks, another one needs to be built with obvious consequences on the sustainability.
**Level 3:** The drone is able to perform its task, with high success rate. In addition, it is repairable, built with a modular design and built with standard components. The main consequence of modular design is an increased ease in the repairability of the drone. This characteristic is very useful for a race drone because in case of a malfunctioning part, it can be quickly replaced. In addition a modular design also leads to the possibility of upgrading the drone. It is

[2]`https://www.matbase.com/design-for-sustainability-guidelines/` [Visited at May 5 2018]

also easier to transport the drone because it is possible to divide it in smaller pieces. This has a high impact on the logistics costs.
**Level 4:** The drone is able to perform its task, with high success rate, and it has a long operational lifetime, which means that money can be saved. In fact, it will take longer before a new drone needs to be built. In addition, it is repairable, built with a modular design and built with standard components.

The stakeholder clearly states "The motors, frame, electronics and cameras shall be replaceable.", which means that a modular structure is required. For this reason the design should be at least of **Level 3**.

### Operational phase

One of the main goals of sustainability is to reduce the amount of emissions and pollution during the operational phase.
**Level 0:** The drone requires a high amount of energy and a high number of consumables in order to be used.
**Level 1:** The drone requires a low amount of energy and a moderate quantity of consumables.
**Level 2:** The drone requires a low amount of energy and few consumables. Lowering the energy consumption is beneficial because a large portion of the worldwide supply of electrical energy originates from power plants that create unwanted emission gases.
**Level 3:** The drone requires a low amount of energy and few or no consumables. Usage of the drone does not disturb people in its proximity. In this case the noise is considered the most relevant aspect because it could have a harmful impact on living beings, as it might damage psychological and physiological health. Each country has slightly different noise regulations, however, all of them state that during a full-working-day a person should not be exposed to noise levels exceeding 80-85 dB. Therefore, the noise level should be below 80 dB. The propulsion system will be the main source of noise, and it needs to be studied in detail to minimize it.
**Level 4:** The drone is powered by clean energy and it uses green consumables. The cleaner the energy source is, the less impact drone has on the environment. The noise produced by the drone during usage is below 80 dB.

There are no specific stakeholder requirements for the operational phase. For this reason the minimum requirement for this life cycle phase is **Level 2**.

### End-of-life

The behaviour of the product at the end-of-life needs to be considered as well. The term end-of-life refers to the moment at which it can not fulfill its requirements anymore.
**Level 0:** At the end of its life the product becomes waste, which cannot be reused, recycled or refurbished. Furthermore, incineration of this waste will create high harmful emissions.
**Level 1:** The drone has a second life after winning the race, it can be reused, or refurbished, for other purposes, which require lower performance. Incineration might create some harmful emissions.
**Level 2:** Incineration of the drone will create low emissions, or even no emission.
**Level 3:** The drone has a second life after winning the race, it can be reused, or refurbished, for other purposes, which require lower performance. The drone is partially recyclable, and incineration of the other parts might create low emissions, or even no emissions.
**Level 4:** The drone is fully recyclable.

One of the the stakeholder requirements focuses on the recyclability of the drone, it means that the End-of-life should be at least of **Level 4**.

## 19.3. Sustainability of the Final Design

The sustainability of the final design is assessed via the LiDS wheel. It is carefully checked that the design complies with the minimum requirements described above.

#### Selection of Materials

All materials in chapter 5 have been carefully selected in order to meet the sustainability requirements. Carbon-fiber is selected as the main material for the frame design since there are two frames they would be connected by the aluminum rods, ABS plastic for the propeller guards and foam for protecting frame from the impact. Besides material for the drone frame, there are electrical components, such as ESCs, microcontrollers, motors, batteries, and sensors.
All materials for the structure are non-hazardous, non-exhaustible and they are recyclable. Nowadays, recyclability of the carbon fiber is in further development, but multiple solutions are already available[3]. Due to the fact that the

[3] *https://www.compositesworld.com/columns/recycled-carbon-fiber-its-time-has-come-* [Visited at May 20 2018]

ABS is thermoplastic, it is 100% recyclable[4] and later might be used to re-create propeller guards or any other 3D printed part.
Recycling any electronic part is a more complicated process, as they usually contain heavy metals such as gold, silver, platinum and base metal like aluminum, copper and iron which needs to be extracted first. Commonly shredding or dismantling methods are used to get valuable components or pieces of metals, however, the efficiency is rather low. Therefore it is advised to reuse electronics components for further projects. It has been shown that it is possible to recycle batteries. Although this process is still quite expensive, but there are many researcher working on it.
Selected: **Level 3**.

**Manufacturing**
The analysis of the sustainability of the manufacturing phase follows from the strategy developed in chapter 15. Only the structure of the drone needs to be manufactured. Working with carbon fibers is critical because carbon fibres easily break if stretched, which release small fibres particles into the surrounding atmosphere[5]. Extra caution must be taken, such as personal protective equipment to protect from the inhaling of fibers, and the contact with the skin or eyes. Additionally, machining of carbon fibre must be conducted in special facilities.
For 3D printing, it is only required to have access to a 3D printer, for instance the ones in the D:DREAM Hall or in some of the faculties. The assembling of the drone can be done at any place. Mounting and testing electronic components and batteries is deemed to be not dangerous, as they all operate at low current. A more detailed analysis follow in Table 19.1.
Selected: **Level 3**.

**Optimization of Lifetime**
At the current stage, it is assumed that the failure rate of the final design is low, because the ToF camera should provide a feedback to the drone about nearby obstacles, thus the drone has a high chance of avoiding direct impacts. However, this drone operates at high speeds, which can lead to many crashes, especially during the testing phase. Although the drone should be able to withstand hard impacts, as specified by the customer requirement, some small parts, such as legs, propellers, and motors are expected to be damaged more than any other components. All these parts are cheap, accessible and easily replaceable.
Selected: **Level 3**.

**Operational Phase**
The final design consumes little power. In fact, the main idea behind the drone is to minimize its size, which leads to a low mass and low power consumption. LiPo batteries are used as power source. Batteries have a certain life cycle after which they need to be replaced with a new one. Although batteries are not a green source of energy, they could be recharged up to 100 times, thus it is not expected to replace them often. Moreover, broken or degraded batteries will be collected and recycled.
Furthermore, the final propeller size is similar to that of commercially available drones. The measured noise level that is produced by different drones of the same category is 70 dB[6], which is below the requirement of 80 dB.
From the perspective of social sustainability, the drone does not break any laws and regulations [7], as it would only fly in specified areas, like an IROS 2018 race track, TU Delft Cyber Zoo and never outdoors. The drone would not violate any privacy law, since none of the photos that are made during the operations are shared with the public.
Selected: **Level 3**.

**End-of-Life**
At the end of the project, all the software would be publicly available and might be used by other developers of the drones, which is very important from a social sustainability prospective.
Structural components could be reused or recycled. All electronics parts, such as sensors, ESCs and microcontrollers might be used for other drones or any project which requires them, or recycled. These electrical components are generic and not limited to aerospace-related applications.
Selected: **Level 4**.

**Conclusion**
Based on this analysis, the following LiDS wheel can be constructed. Figure 19.2 clearly shows that the final design meets well the requirements, and the design can be considered sustainable. However, this analysis should be redone every month.

[4] `https://web.archive.org/web/20140306033349/http://www.heathland.nl/abs-recycling.html` [Visited at June 19 2018]
[5] `https://www.monash.edu/ohs/info-docs/safety-topics/chemical-management/carbon-fibre-composites-ohs-information-` [Visited at May 24 2018]
[6] `https://www.bu.edu/ufmal/files/2016/07/aiaa-2016-2873.pdf` [Visited at 10 June 2018]
[7] `https://www.tomstechtime.com/netherlands` [Visited at June 19 2018]

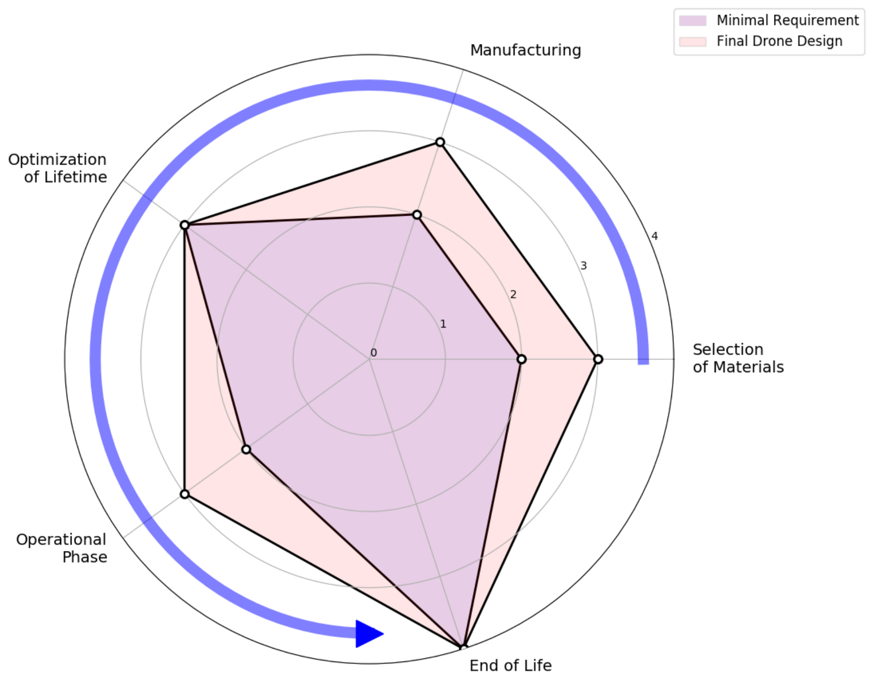


Figure 19.2: Wheel diagram of the final design

### 19.3.1. Advanced Sustainable Design

After the LiDS wheel analysis, a software is used to estimate how much natural resource the drone consumes in his whole lifecycle. SolidWorks released a plug-in for its famous CAD software that enables designers to estimate the sustainability, "SolidWorks Sustainability" [8], based on the data collected from GaBi [9].
The results of the analysis are shown in Table 19.1, to give a feeling of what these values mean it is possible to explore [10]. For instance, the carbon footprint equals the one of a car driving for 75 km, and the overall energy used equals the amount used for watching TV for 5 hours.

Table 19.1: This table shows the result of the sustainability analysis using Solidworks Sustainability.

| | Carbon Footprint (kg CO2) | Energy Consumption (MJ) | Air Acidification (mol $H^+$ equivalent) | Water Eutrophication (kg Nitrogen equivalent) |
|---|---|---|---|---|
| Material | 16.4055 | 239.526 | 6.3135 | 0.00372749 |
| Manufacturing | 0.216275 | 3.14191 | 0.0684549 | 2.2367e-005 |
| End Of Life | 1.68854 | 1.25654 | 0.0668368 | 0.000429328 |
| Transportation | 0.165795 | 2.43577 | 0.0682267 | 6.43294e-005 |
| Total | 18.4761 | 246.36 | 6.51702 | 0.00424352 |

[8] *http://www.solidworks.com/sustainability/* [Visited at 5 may 2018]
[9] *http://www.gabi-software.com/* [Visited at 5 may 2018]
[10] *https://www.solidworks.com/sustainability/products/calculator/index.htm??LANG=en&BSLca=0.000&BSLai=0.000&BSLwa=0.000&BSLen=0.000&CURca=18.476&CURai=18.476&CURwa=18.476&CURen=18.476&BSLname=&CURname=Final&CML=yes&Month=Jun&Day=22&Year=2018&Time=14%3A00&VID=PR* [Visited at 20 June 2018]

# 20 Post-DSE to IROS

If this was a real project, the next step would be to produce the drone and to test it. The purpose of this chapter is to plan for the next steps that needs to be taken. First, a way to create an integrated simulation is proposed in section 20.1. In Section 20.2, a work flow diagram is created. More in depth information are provided in the Gantt Chart in Section 20.3. Finally, a cost breakdown of all costs up until the IROS 2018 competition is provided in Section 20.4.

## 20.1. Integrated Simulation

In the chapters below, only parts of the system were integrated and simulated together to demonstrate the functionality of the design. In order to achieve full integration, ROS (Robot Operating System) might be used as a tool, which combines software packages with the hardware implementation. The drone would be put to a virtual environment, where performance and reliability of the drone could be studied on multiple sample tests cases or full track.

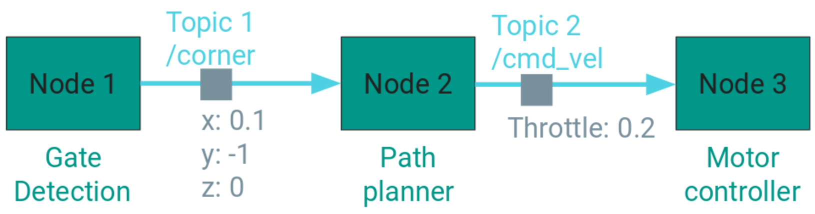


Figure 20.1: Example of ROS Node map

ROS is the robotics software platform that works as publisher subscribed network. It means that a hardware abstraction and software solutions are linked to each other. Sensors, devices or motors could be completely described in packages. Another benefit of the package, that the sensors that are not available to developer might be still used in simulation as far as they are correctly described. For instance, it allows using ToF camera, even though if it is unavailable. Moreover, Nodes are used to control the robot, like motor controller, path planned, gate detection. The communication between nodes is achieved through topics (command or state), an example of node map is shown in Figure 20.1. The environment could be built in Gazebo. A Gazebo is a real-world physics simulator tool, which defines the map, obstacles and realistic interaction between drone and space. The race track could be loaded to Gazebo, thus the entire model could be validated, and actual lap time could be estimated as well. The end result would have all information about the drone such as state variables, camera view, discovered gates and updated planned path.

Due to ROS modularity, subsystems might be optimized and validated. It is recommended to build ROS simulation model before actual drone, in order to prove design integrity. Afterwards, ROS could be used as a visualization and drone control tool, so that during testing and developing it would be easy for the user to see the processes and decisions done by the drone.

## 20.2. Work Flow Diagram

In order to start planning, a work flow diagram was made. This can be seen in Figure 20.2 and Figure 20.3. This shows the tasks that have to be performed in a sequential order. An additional benefit of a work flow diagram is that it clearly shows which tasks depend on each other. The letters that were used to label the entries allow for a more intuitive way of labeling each of the tasks. They also allow for easy restructuring. This is the first part of the Project Design and Development Logic diagram. The continuation of this part is included in Section 21.1.

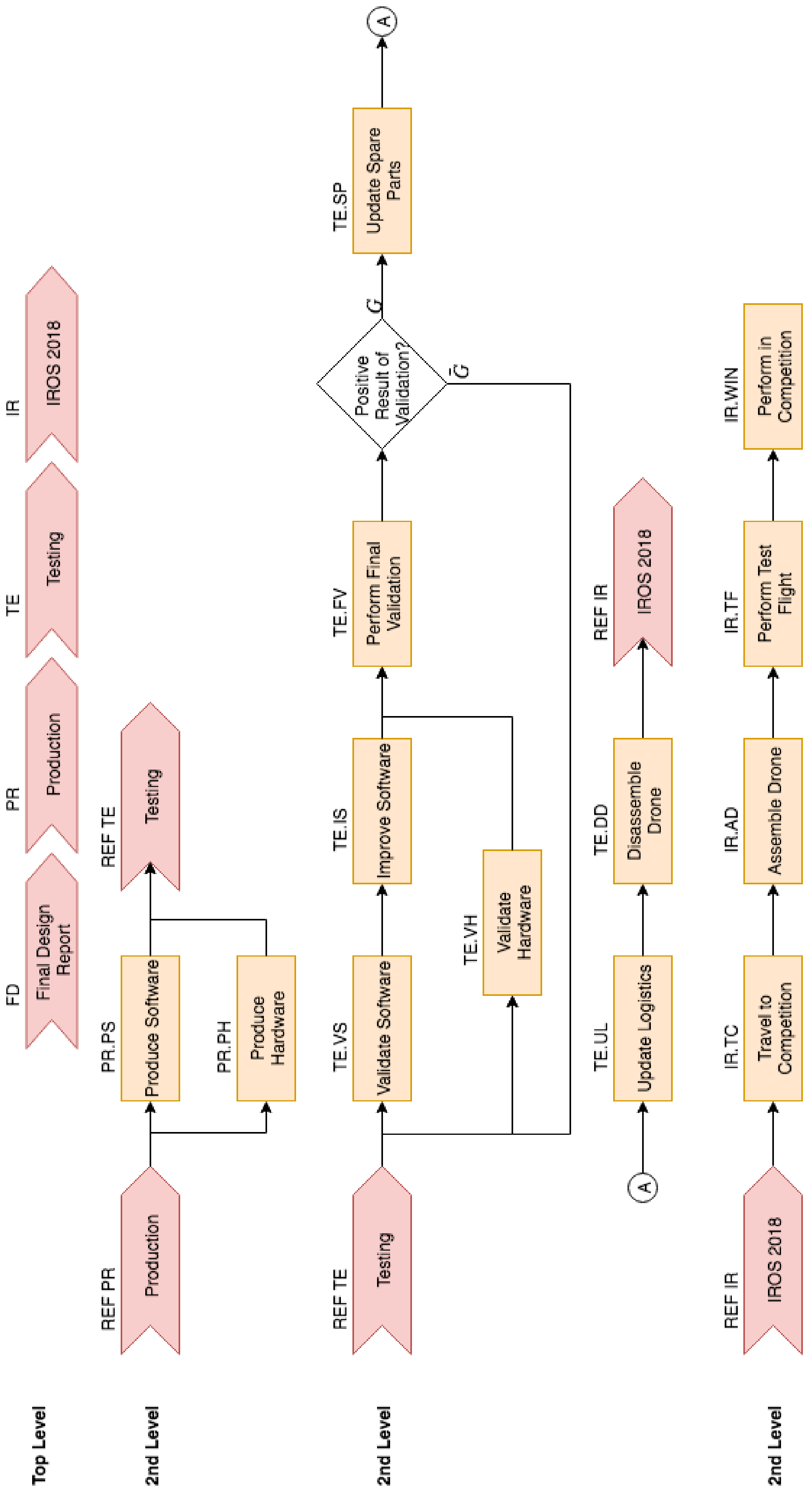


Figure 20.2: Work Flow Diagram part 1

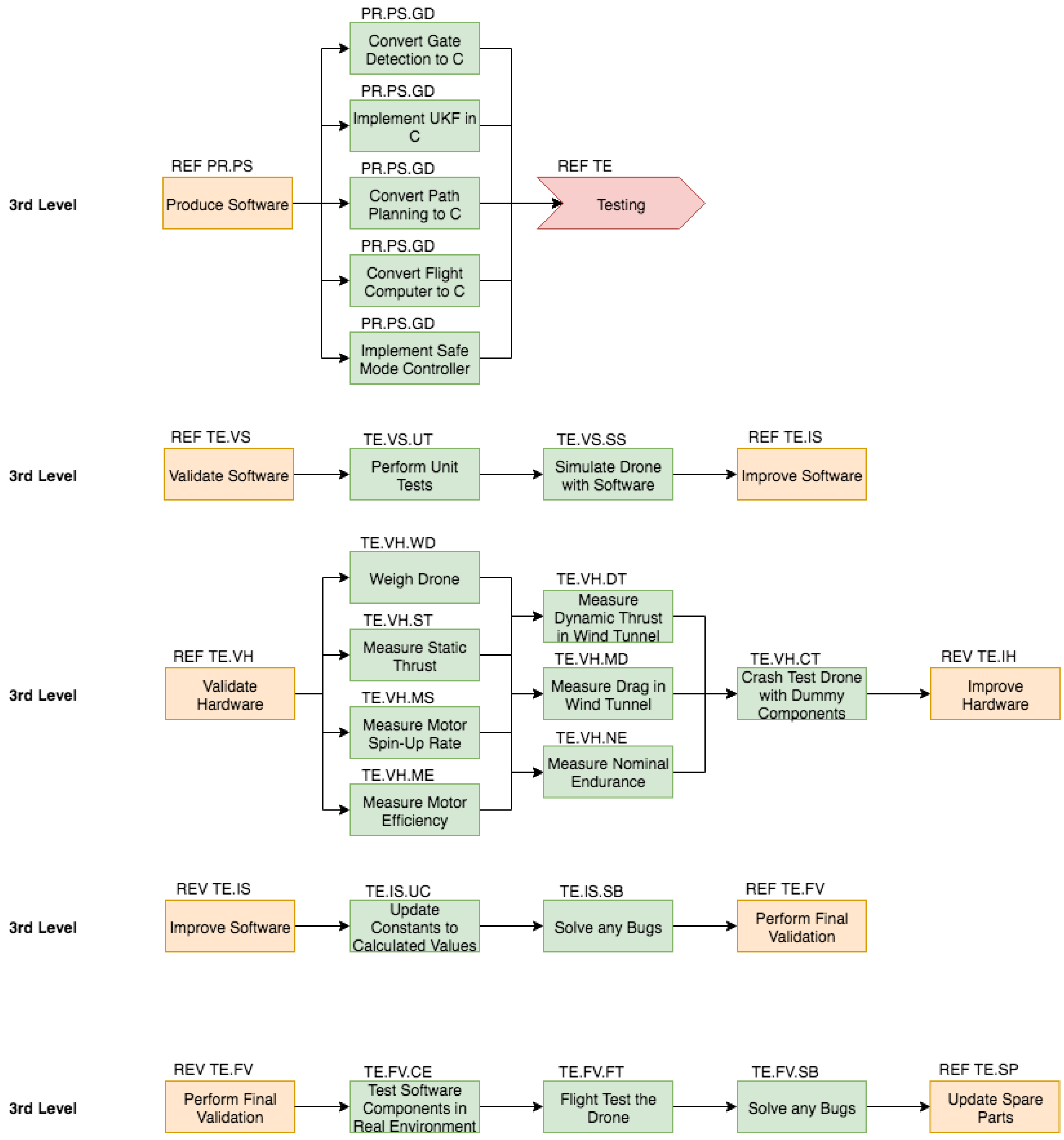
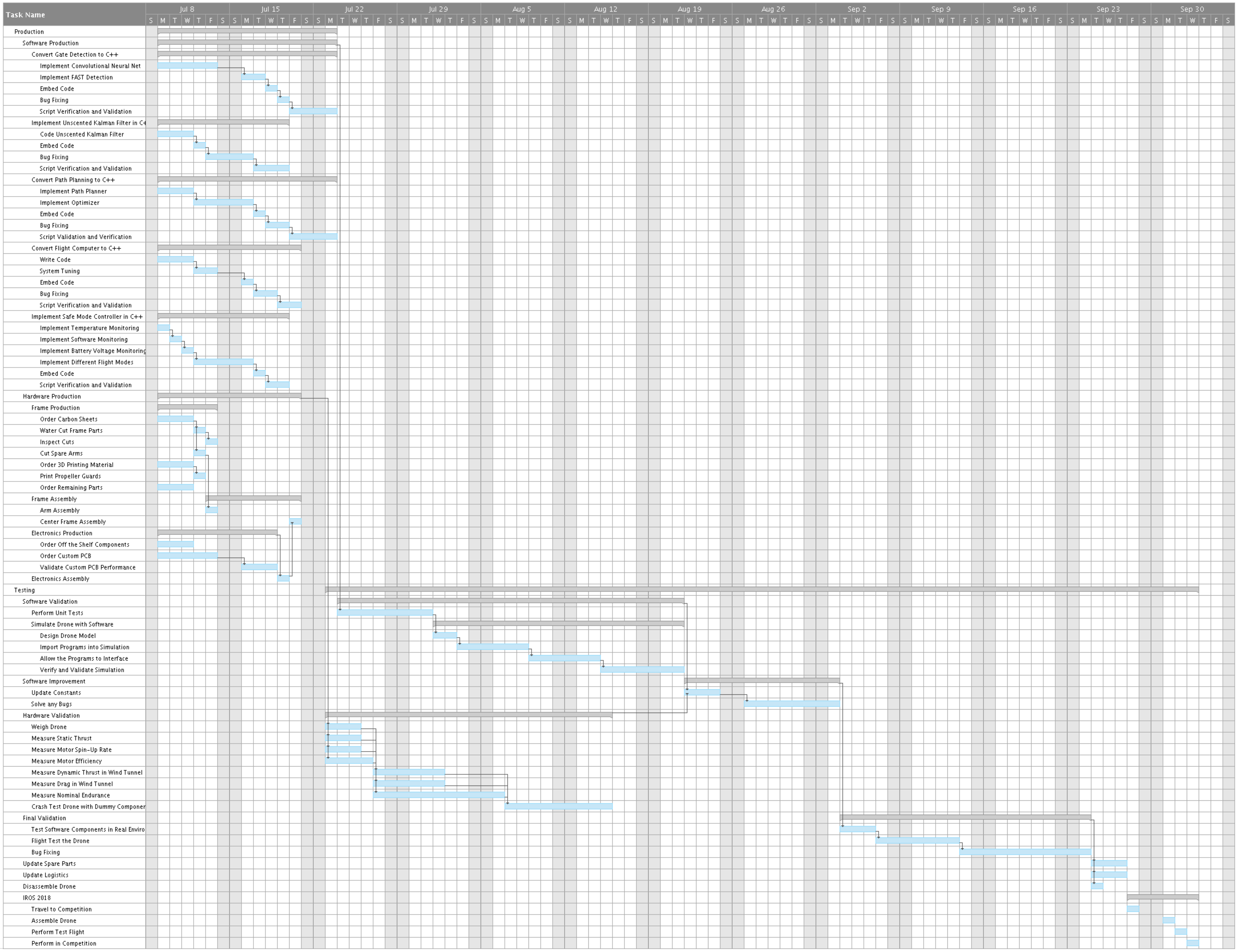


Figure 20.3: Work Flow Diagram part 2

## 20.3. Gantt Chart

The work flow diagram that is shown in Figure 20.2 and Figure 20.3 was then processed into a Gantt chart by adding estimates of the time needed to do each of the tasks. This, together with the sequential representation of the tasks in the work flow diagram, allows the Gantt chart to be used for the planning of the project. The Gantt chart for the time between the end of the DSE and the IROS 2018 competition is shown in Figure 20.4. A part of software validation is Simulating the drone in software, this could be done using ROS, which has been explained in Section 20.1.

The software verification that will be done before the testing phase consists of using computer generated data and images. All validation will be done as can be seen in Figure 20.4.



| | Task Name |
|---|---|
| 1 | Production |
| 2 | Software Production |
| 3 | Convert Gate Detection to C++ |
| 4 | Implement Convolutional Neural Net |
| 5 | Implement FAST Detection |
| 6 | Embed Code |
| 7 | Bug Fixing |
| 8 | Script Verification and Validation |
| 9 | Implement Unscented Kalman Filter in C+ |
| 10 | Code Unscented Kalman Filter |
| 11 | Embed Code |
| 12 | Bug Fixing |
| 13 | Script Verification and Validation |
| 14 | Convert Path Planning to C++ |
| 15 | Implement Path Planner |
| 16 | Implement Optimizer |
| 17 | Embed Code |
| 18 | Bug Fixing |
| 19 | Script Validation and Verification |
| 20 | Convert Flight Computer to C++ |
| 21 | Write Code |
| 22 | System Tuning |
| 23 | Embed Code |
| 24 | Bug Fixing |
| 25 | Script Verification and Validation |
| 26 | Implement Safe Mode Controller in C++ |
| 27 | Implement Temperature Monitoring |
| 28 | Implement Software Monitoring |
| 29 | Implement Battery Voltage Monitoring |
| 30 | Implement Different Flight Modes |
| 31 | Embed Code |
| 32 | Script Verification and Validation |
| 33 | Hardware Production |
| 34 | Frame Production |
| 35 | Order Carbon Sheets |
| 36 | Water Cut Frame Parts |
| 37 | Inspect Cuts |
| 38 | Cut Spare Arms |
| 39 | Order 3D Printing Material |
| 40 | Print Propeller Guards |
| 41 | Order Remaining Parts |
| 42 | Frame Assembly |
| 43 | Arm Assembly |
| 44 | Center Frame Assembly |
| 45 | Electronics Production |
| 46 | Order Off the Shelf Components |
| 47 | Order Custom PCB |
| 48 | Validate Custom PCB Performance |
| 49 | Electronics Assembly |
| 50 | Testing |
| 51 | Software Validation |
| 52 | Perform Unit Tests |
| 53 | Simulate Drone with Software |
| 54 | Design Drone Model |
| 55 | Import Programs into Simulation |
| 56 | Allow the Programs to Interface |
| 57 | Verify and Validate Simulation |
| 58 | Software Improvement |
| 59 | Update Constants |
| 60 | Solve any Bugs |
| 61 | Hardware Validation |
| 62 | Weigh Drone |
| 63 | Measure Static Thrust |
| 64 | Measure Motor Spin-Up Rate |
| 65 | Measure Motor Efficiency |
| 66 | Measure Dynamic Thrust in Wind Tunnel |
| 67 | Measure Drag in Wind Tunnel |
| 68 | Measure Nominal Endurance |
| 69 | Crash Test Drone with Dummy Componer |
| 70 | Final Validation |
| 71 | Test Software Components in Real Enviro |
| 72 | Flight Test the Drone |
| 73 | Bug Fixing |
| 74 | Update Spare Parts |
| 75 | Update Logistics |
| 76 | Disassemble Drone |
| 77 | IROS 2018 |
| 78 | Travel to Competition |
| 79 | Assemble Drone |
| 80 | Perform Test Flight |
| 81 | Perform in Competition |

## 20.4. Cost Breakdown Structure

Between the end of the DSE and the IROS 2018 competition, many costs will be incurred. These can be split up into 2 main categories: Organizational and Production costs, described in Section 20.4.1 and Section 20.4.2 respectively. The complete breakdown can be found in Figure 20.5.

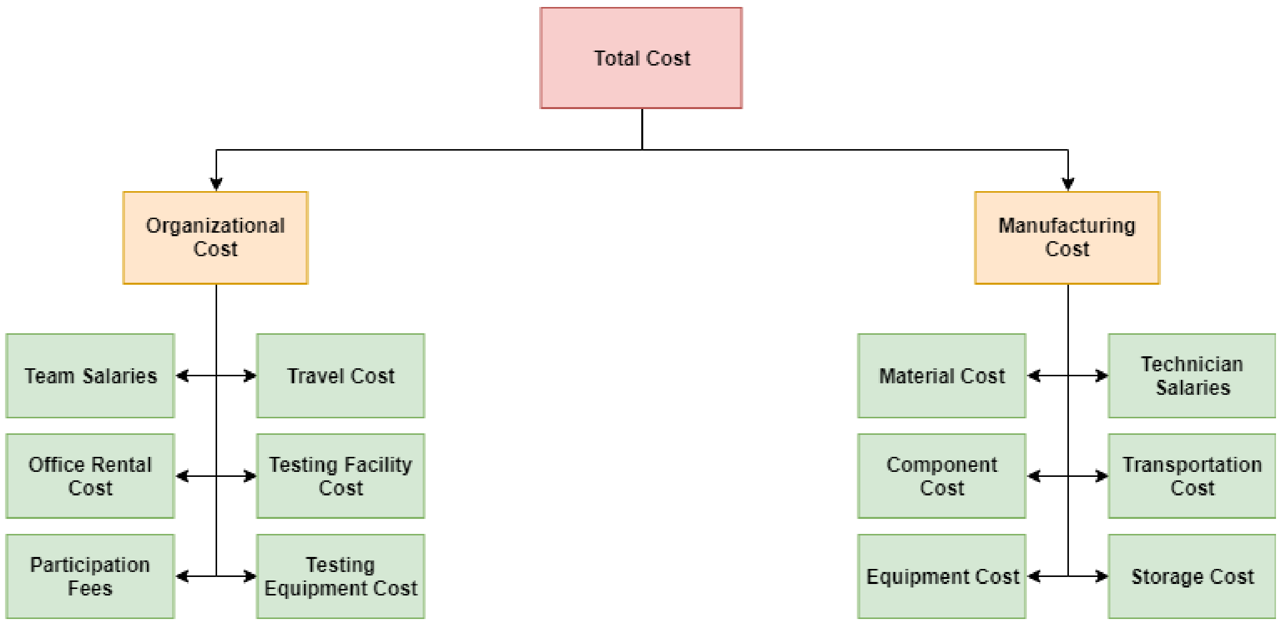


Figure 20.5: Cost Breakdown Structure from the end of the DSE to the IROS 2018 competition

### 20.4.1. Organizational Cost

This part deals with all of the costs that are associated with the design team. The design team is the team that will be doing all of the validation and improvement of the drone. They will also be the operating crew who will go to and participate in the IROS competition. All of these things add to the total cost of the project. They are further explained below.

**Team Salaries**
After the DSE it is assumed that the team does not consist out of students. Since any employee needs a salary to live, these will have to be payed for by the project.
**Office Rental Cost**
The team will also need a place to work, which has to be paid for as well. Although it is possible that the TU will provide this, it is not certain at this point in time.
**Participation Fees**
No information on a participation fee was found. There should, however, be some budget left over for unforeseen circumstances, which include a participation fee.
**Travel Cost**
The team has to travel to Madrid to participate, which will cost money.
**Testing Facility Cost**
Both the model and for instance the controller can be improved by updating the physical constants of the drone to be closer to their actual value. In order to calculate these improved values, the drone has to be tested in certain facilities. It is possible that the TU will provide these, but it is by no means ensured.
**Testing Equipment Cost**
Any consumable that is being us during the tests has to be bought by the team. This includes things like fillings for smoke generators in wind tunnels, etc.

### 20.4.2. Manufacturing Cost

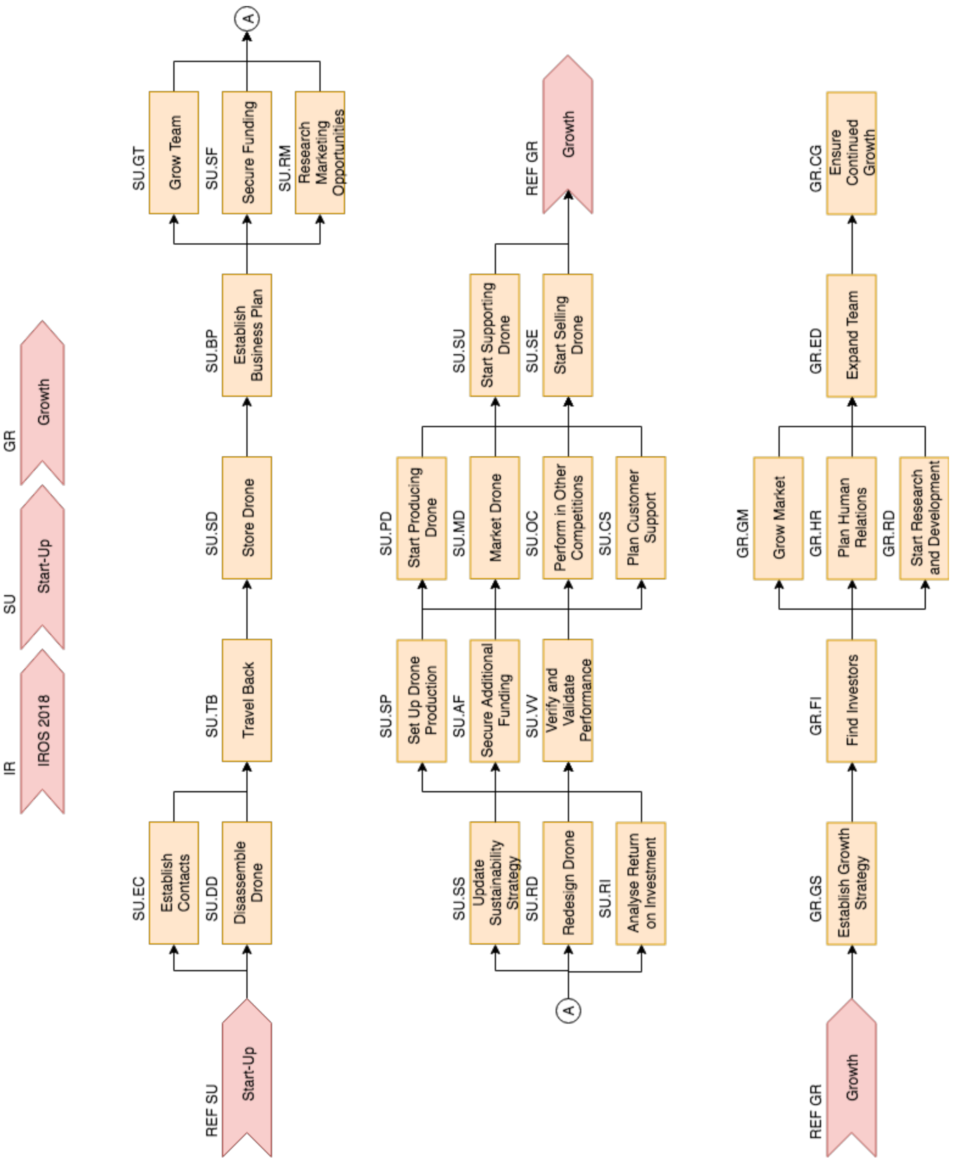
The manufacturing cost was divided up into 6 categories: material cost, technician salaries, component cost, transportation cost, equipment cost, and storage cost. The production methods are worked out in more detail in Chapter 15.

# 21 Long Term Vision

In this section, the long term vision for the project is explained. The short vision until the IROS 2018 competition is explained in Chapter 20, so this chapter deals with the vision after this race. Firstly, a the continued project design and development logic is shown in Section 21.1. Secondly, a more detailed inspection of the markets that were identified in Chapter 2 is done. Finally, in Section 21.3, the Return on Investment (RoI) is investigated for one of the markets in Section 21.2.

## 21.1. Project Design and Development Logic

The first part of the Project Design and Development Logic diagram is Figure 20.2 and Figure 20.3. These show the development until the IROS 2018 competition. The continuation of this, which talks about the development after the race, can be seen in Figure 21.1.



IR IROS 2018
SU Start-Up
GR Growth
REF SU Start-Up
SU.EC Establish Contacts
SU.DD Disassemble Drone
SU.TB Travel Back
SU.SD Store Drone
SU.BP Establish Business Plan
SU.GT Grow Team
SU.SF Secure Funding
SU.RM Research Marketing Opportunities
A
SU.SS Update Sustainability Strategy
SU.RD Redesign Drone
SU.RI Analyse Return on Investment
SU.SP Set Up Drone Production
SU.AF Secure Additional Funding
SU.VV Verify and Validate Performance
SU.PD Start Producing Drone
SU.MD Market Drone
SU.OC Perform in Other Competitions
SU.CS Plan Customer Support
SU.SU Start Supporting Drone
SU.SE Start Selling Drone
REF GR Growth
REF GR Growth
GR.GS Establish Growth Strategy
GR.FI Find Investors
GR.GM Grow Market
GR.HR Plan Human Relations
GR.RD Start Research and Development
GR.ED Expand Team
GR.CG Ensure Continued Growth

Figure 21.1: Continuation of Project Design and Development Logic Diagram

## 21.2. Future Market

Competitions like the IROS and IMAV are a great way for a drone company to show off its product. These will obviously be used for exactly this reason, even if the drone that is shown has to be tweaked slightly to ensure maximum performance in the controlled environment of a race. The market that the team will strive to get into is the bird control market. This was deemed the most profitable for an adaptation of RAIDER because it values the performance specifications as races do. In order to effectively follow and scare off birds, a drone has to be fast and agile. Adding the ability to autonomously follow and guide birds away from e.g. airports would allow one operator to manage a swarm of drones, making airports a lot safer. As explained in Section 2.1.3, a cost to benefit ratio of over 70 can be achieved, clearly showing the profitability.

A few changes can to be made to RAIDER to improve its performance in the bird control market: Firstly, the ToF camera will not be of much use. With its limited range of only 4 meters, it will not help finding or chasing birds. Since places like airports and farm fields are normally very open, there will be little risk of running into trees or buildings, it also isn't needed for collision avoidance. Secondly, the IMU can be replaced with a cheaper alternative that uses GPS. Although the IROS competition is indoor, with little GPS coverage, the large open spaces in which the drone will function should have no such problems. Using GPS to localize the drone allows for a cheaper IMU to be used since it will get an updated location once every second, where there are no large periods of time without updates. The ultrasound altimeter can also be removed, further decreasing the price of the drone.

An additional advantage of entering the bird control market is the fact that a bird control drone can be relatively easily converted into a drone that counters malicious drones invading an operator's airspace. In order to become a stand-alone system for both bird and drone control, a ground station that monitors the airspace has to be designed. This ground station would then send a drone to investigate the area where a bird or malicious drone was detected. The drone can then scare them off or bring a drone down when it finds it.

## 21.3. Return on Investment

Since RAIDER has no marketable value as it is, it will have to be modified to break into any market. The market chosen for this is the bird control market, which was explained in Section 21.2. On top of modifying the drone, a base station would also have to be designed to send the drones to where they are needed. Since no calculations have been done on this base station, it is impossible to calculate its upfront or maintenance cost.
As explained in Section 21.2, the most expensive parts of the drone can be removed when it is being redesigned to fit into the bird control market. This would drastically reduce the price since the IMU and the ToF camera together make up approximately 60% of the price of the drone. Although the IMU price cannot completely be eliminated, a normal IMU and GPS combination costs less than €100, reducing the total unit price to approximately €600. This can be reduced even further by doing away with the ultrasound altimeter, which lowers the cost by another €35. In order to allow the drone to scare away birds, a speaker should be placed on it, which would add a little bit of cost and weight. However, the increase to cost due to this modification are not expected to exceed €100. This results in a total unit cost of the drone of €700. This cost reduction would present a significant increase in the RoI of this project.

Calculating the actual RoI for the new drone is a problem because the operational costs have to be considered. The current cost of bird control on Eindhoven Airport is approximately €800 000 [43]. This allows for a maintenance cost of over €2100 per day, which is easily achievable by a fully autonomous drone based system.
RoI is a very important aspect of any commercial product and should therefore be done as early as possible, which is why it was represented in Figure 21.1. It is to be done concurrently with the redesign of the drone. After both the redesign and the RoI analysis are done, more investors can be attracted.

# 22
# Conclusion

The aim of this project is to design an autonomous racing drone that will win the IROS 2018 Autonomous Drone Race. The result of the past 10 weeks is a drone where every single component has been chosen and analyzed in order to match a specific strategy that has been developed in order to maximize the drone's performance. The Micro Air Vehicle (MAV) is called RAIDER which is an acronym that suits the design and describes best its main features: Rapid and Agile Intelligent Drone for Extreme Racing.

The structural configuration of the drone is a quadcopter which has been designed around the payload and electronics that are required for the system. The structure has been designed based on the most critical load case: an impact with a gate at a speed of 100 km/h. The propulsion system has been sized with the objective of maximizing performance and endurance. Each of the four arms carries an XRotor Motor which gives the drone a maximum thrust-to-weight ratio of 5.4 and a maximum power consumption of 950 W, which allows the drone to a have a nominal flight time of 10 minutes. The price of one unit is €2047. This includes all replacement parts such as extra batteries, propellers and cameras.

The most relevant sensors that have been selected are the inertial measurement unit, an RGB camera which commonly seen on FPV racing drones, and a ToF camera which is used to create a depth map of the environment. This last element has been introduced in order to greatly increase the reliability of the drone when estimating the distance from an object and to avoid obstacles. The other camera which outputs normal images is used to detect gates based on a 9 layer Convolutional-Deconvolutional Neural Network that performs background removal. On top of the processed images a simple contrast based algorithms is executed to find the four corners of the gate. These are then used to estimate the relative position of the drone using simple mathematical principles and perspective.

The positioning is mainly driven by the IMU which outputs sufficiently accurate measurements during 9.3 seconds before the integration error starts to become too big. This problem has been tackled by using a sensor fusion algorithm which combines the IMU measurements with the relative position estimates from the gate detection. This can be used for absolute positioning using the a-priori knowledge of the gates location. The measurements, which are sampled at a different frequency, 800Hz and 30Hz respectively, are fed to an Unscented Kalman Filter (UKF) which outputs the best pose estimation. The drone is capable of high-frequency online path planning, due to a joint polynomial optimization routine designed to generate minimum snap trajectories through the gates. The initial optimal path is generated before the race according to a desired completion time for the entire track, which allows for either safe or aggressive initial trajectories to be generated based on the amount of time left in the competition. The output of the path planning are the required dynamic state variables. A proportional integral-derivative (PID) controller has been designed for RAIDER since this type of controller is widely used for MAVs for its high performance. The controller takes the output variables from the path planning together with the pose estimate from the sensor fusion algorithm to calculate the required amount of thrust for the desired trajectory.

The software is executed on two different computer boards, the gate detection algorithm is run by the Raspberry Pi Zero W, while the path planning, flight controller, sensor fusion algorithm and the system monitoring are placed on the Pocket Beagle. The integrated navigation strategy is able to finish the race track in 60 seconds which is an astonishing result considering that no one ever finish the track and that the winners of IROS 2017 were timed 3 minutes at least passing only nine out of the ten gates.

The overall design has been developed concurrently, taking a unique focus shaped by the goals, resources, and schedule into account. To this extent the reliability, availability, maintainability, and safety of the design have been analyzed. The risks of every subsystem have been identified and with the purpose of mitigating them, specific strategies have been adopted. These mitigation strategies include components such as an IMU damper, propeller guards and shock absorbs. Moreover, the sustainability of the design has been considered from environmental to social aspects. For instance, the carbon footprint is equivalent to the one of a car driving for 75 km. Finally, the team has conceived a plan for future steps to be taken in order to achieve more ambitious objectives after winning the IROS 2018 competition, to drive the technology that will allow drones to autonomously complete a wide variety of tasks.

# Reference List

[1] D. Arterburn, M. Ewing, R. Prabhu, F. Zhu, and D. Francis. FAA UAS Center of Excellence task A4: UAS ground collision severity evaluation, revision 2. 2017.

[2] Andrew J. Barry, Tim Jenks, Anirudha Majumdar, Huai Ti Lin, Ivo G. Ros, Andrew A. Biewener, and Russ Tedrake. Flying between obstacles with an autonomous knife-edge maneuver. *Proceedings - IEEE International Conference on Robotics and Automation*, January 2014. ISSN 1050-4729. doi: 10.1109/ICRA.2014.6907217.

[3] Dmitry Bershadsky, Stephen Haviland, and Eric N. Johnson. Electric Multirotor Propulsion System Sizing for Performance Prediction and Design Optimization. In *57th Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2016. doi: 10.2514/6.2016-0581.

[4] A. Bonifazi, T. Roepman, R. Oudheusden, F. Dupon, D.M. Jimenez, B. Bouwels, and ... Autonomous race drone: Midterm report. Unpublished manuscript, 2018.

[5] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002, 2015. doi: 10.1177/0278364914558129.

[6] Paweł Burdziakowski. Low Cost Hexacopter Autonomous Platform for Testing and Developing Photogrammetry Technologies and Intelligent Navigation Systems. In *Environmental Engineering 10th International Conference*, 2017. doi: 10.3846/enviro.2017.172.

[7] W. Burgard, O. Brock, and C. Stachniss. *Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing*, pages 352–359. MIT Press, 2008. ISBN 9780262255868.

[8] Y. Cao. Multimedia Computing: Algorithms, Systems, and Applications: Feature Extraction. Technical report, Department of Computer Science, The University of Massachusetts Lowell, 2018.

[9] M. F. Cloutier, C. Paradis, and V. M. Weaver. A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement. *Electronics*, 5(4), September 2016. ISSN 2079-9292. doi: 10.3390/electronics5040061.

[10] Paul J. Coco. Mae 586 Project Work in Mechanical Engineering: Tricopter Design. Technical report, NC State University, 2012.

[11] Khalid Dadah. The global Drone Racing League incorporates virtual reality into the experience., 2017.

[12] D.V.W.M de Vries. Characterization of polymeric foams. Mt09.22, Eindhoven University of Technology, July 2009.

[13] Michael Eickenberg, Alexandre Gramfort, Gaël Varoquaux, and Bertrand Thirion. *Seeing it all: Convolutional network layers map the function of the human visual system*, volume 152, pages 184–194. Elsevier, 2017.

[14] Artem Rozantsev et. al. On Rendering Synthetic Images for Training an Object Detector. Technical report, Swiss Federal Institute of Technology, Lausanne (EPFL), 2014.

[15] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *SRI International*, pages 381–395, June 1981. doi: doi:10.1145/358669.358692.

[16] S. Y. Fu, B. Lauke, E. Mäder, C. Y. Yue, and X. Hu. Tensile properties of short-glass-fiber- and short-carbon-fiber-reinforced polypropylene composites. *Composites Part A: Applied Science and Manufacturing*, 31(10):1117 – 1125, 2000. ISSN 1359-835X. doi: https://doi.org/10.1016/S1359-835X(00)00068-3.

[17] R. Gareis, M. Huemann, and A. Martinuzzi. Relating sustainable development and project management. In *PMI Research Conference: Defining the Future of Project Management*, 2010. retrieved from https://www.pmi.org.

[18] Australian Government Department of defense. *Techninical Risk Assessment Handbook*. DSTO Science and Technology for a Secure World, Canberra ACT 2600, 2010.

[19] Clayton Green. Modeling and test of the efficiency of electronic speed controllers for brushless DC motors modeling and test of the efficiency of electronic speed controllers for brushless DC motors. Msc thesis, California Polytechnic State University, September 2015.

[20] S.F. Hoerner. *Fluid dynamic drag*. Published by the author, 1965. ISBN 978-9993623939.

[21] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire J Tomlin. Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007. doi: 10.2514/6.2007-6461.

[22] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *CoRR*, abs/1105.1186, 2011. URL `http://arxiv.org/abs/1105.1186`.

[23] P. Khanna, B. R. Flam, B. Osborn, J. A. Strom, and S. Bhansali. Skin penetration and fracture strength testing of silicon dioxide microneedles. *Sensors and Actuators A: Physical*, 170(1):180 – 186, 2011. ISSN 0924-4247. doi: https://doi.org/10.1016/j.sna.2010.09.024.

[24] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. Using Inertial Sensors for Position and Orientation Estimation. *CoRR*, abs/1704.06053, 2017. URL `http://arxiv.org/abs/1704.06053`.

[25] T. Gopala Krishna and S. Om Kumar. Design of UAV based on ARM Flight Controller. *International Journal of Industrial Electronics and Electrical Engineering*, 5(4), April 2017. ISSN 2347-6982. doi: IJIEEE-IRAJ-DOI-7770.

[26] S. Kuppa. Injury Criteria for Side Impact Dummies, 2004.

[27] J.G. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge Aerospace Series, 2000. ISBN 0-521-66060-2.

[28] D.P. Loucks and E. van Beek. *Water resources systems planning and management*. UNESCO Publishing, 2005. ISBN 92-3-103998-9.

[29] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

[30] Desire L. Massart and Leonard Kaufman. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, 187:171–179, January 1986. doi: doi.org/10.1016/S0003-2670(00)82910-4.

[31] MATLAB. *9.4.0.813654 (R2018a)*. The MathWorks Inc., Natick, Massachusetts, 2018.

[32] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011. doi: 10.1109/ICRA.2011.5980409.

[33] Daniel Mellinger. *Trajectory Generation and Control for Quadrotors*. Phd dissertation, University of Pennsylvania, January 2012.

[34] MIL-STD-883E. Test method for standard electronics. Standard, Department of Defense, Columbus, USA, August 1977.

[35] H. Mirzahosseinian and R. Piplani. A study of repairable parts inventory system operating under performance-based contract. *European Journal of Operational Research*, 214(2):256 – 261, 2011. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2011.04.035.

[36] M. Ebrahimi Moghaddam and M. Jamzad. Motion blur identification in noisy images using mathematical models and statistical measures. *Pattern Recognition*, 40(7):1946 – 1957, 2007. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2006.11.022.

[37] H. Moon, Y. Sun, J. Baltes, and S. J. Kim. The IROS 2016 Competitions [competitions]. *IEEE Robotics Automation Magazine*, 24(1):20–29, March 2017. ISSN 1070-9932. doi: 10.1109/MRA.2016.2646090.

[38] T. Moore and D. Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[39] A. Papazoglou, S. Longo, D. Auger, and F. Assadian. Nonlinear Filtering Techniques Comparison for Battery State Estimation. *Journal of Sustainable Development of Energy, Water and Environment Systems*, 2(3):259–269, 2014. doi: 10.13044/j.sdewes.2014.02.0021.

[40] C. Park, N. Cho, K. Lee, and Kim Y. Formation Flight of Multiple UAVs via Onboard Sensor Information sharing. *Sensors*, 15(7), July 2015. ISSN 1424-8220. doi: 10.3390/s150717397.

[41] Mihail Pivtoraiko and Alonzo Kelly. Kinodynamic motion planning with state lattice motion primitives. pages 2172–2179, September 2011.

[42] T. H. Post. Precise localization of a UAV using visual odometry. MSc Report 037RaM2015, University of Twente, December 2015.

[43] T. H. Post. Value analysis of Integral bird control at an airport: a decision making support tool. Msc report, Delft University of Technology, July 2017.

[44] S. H. Rajani and U. Nair. Design and analysis of a nonlinear controller for a hypersonic wind tunnel. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–4, December 2013. doi: 10.1109/ICCIC.2013.6724138.

[45] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016. doi: 10.1109/CVPR.2016.91.

[46] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011. doi: 10.1109/ICCV.2011.6126544.

[47] K.Y.W. Scheper, M. Karásek, C. De Wagter, B.D.W Remes, and G.C.H.E de Croon. First autonomous multi-room exploration with an insect-inspired flapping wing vehicle. Article in preparation, 2018.

[48] J. Seddon. *Basic Helicopter Aerodynamics*. BSP Professional Books, 1990. ISBN 0-632-02032-6.

[49] A.J.G. Silvius and R. Schipper. A maturity model for integrating sustainability in projects and project management. *Journal of Modern Project Management*, 3, 2015. retrieved from http:://www.researchgate.net.

[50] Ewoud Smeur, Q Chu, and Guido Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. 39:1–12, December 2015.

[51] S. Spurgeon. Sliding mode control: a tutorial. In *2014 European Control Conference (ECC)*, pages 2272–2277, June 2014. doi: 10.1109/ECC.2014.6862622.

[52] C.H. Tan, J.T.H. Goh, W.J. Ang, J. Le Lee, E.S. Lin, G.S. Soh, and S. Foong. Design and development of micro-aerial vehicle for tree inspections. In *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 593–598, November 2017. doi: 10.1109/ICCIS.2017.8274844.

[53] E. Tempelman, M.M.S. Dwaikat, and C. Spitás. Experimental and Analytical Study of Free-Fall Drop Impact Testing of Portable Products. *Experimental Mechanics*, 52(9):1385–1395, November 2012. ISSN 1741-2765. doi: 10.1007/s11340-011-9584-y.

[54] L.W. Traub. Calculation of Constant Power Lithium Battery Discharge Curves. *Batteries*, 2:1–17, 2016. doi: 10.3390/batteries2020017.

[55] A. Valade, P. Acco, Pi. Grabolosa, and J. Fourniols. A Study about Kalman Filters Applied to Embedded Sensors. *Sensors*, 17(12), December 2017. ISSN 1424-8220. doi: 10.3390/s17122810.

[56] J. Verbeke and S Debruyne. Vibration analysis of a UAV multirotor frame. September 2016.

[57] W. Verhagen. AE3221-i Systems Engineering and Aerospace Design. Technical report, Delft University of Technology, 2018.

[58] J Wang, P.J Callus, and M.K Bannister. Experimental and numerical investigation of the tension and compression strength of un-notched and notched quasi-isotropic laminates. *Composite Structures*, 64(3):297 – 306, 2004. ISSN 0263-8223. doi: 10.1016/j.compstruct.2003.08.012.

[59] R. Wang. *Edge detection using convolutional neural network*, pages 12–20. 2016.

[60] Stephan M. Weiss. Vision based navigation for micro helicopters. 2012. doi: 10.3929/ethz-a-007344020.

[61] N. Williard, W. He, M. Osterman, and M. Pecht. Reliability and failure analysis of Lithium Ion batteries for electronic systems. In *2012 13th International Conference on Electronic Packaging Technology High Density Packaging*, pages 1051–1055, August 2012. doi: 10.1109/ICEPT-HDP.2012.6474788.

[62] S. T. Wu and M. R. G. Marquez. A non-self-intersection Douglas-Peucker algorithm. In *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pages 60–66, October 2003. doi: 10.1109/SIBGRA.2003.1240992.

[63] Sheng Zhang and Wei-qi Qian. Dynamic backstepping control for pure-feedback nonlinear systems. June 2017.

[64] W. Zhao and S. Du. Spectral x2013;Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8): 4544–4554, August 2016. ISSN 0196-2892. doi: 10.1109/TGRS.2016.2543748.

[65] E. Zio and N. Pedroni. *Uncertainty characterization in risk analysis for decision making practice*. Foundation for an Industrial Safety Culture. ISBN ISSN 2100-3874.

# A
# Message Formats

Example messages for inter computer board communication[1].

Listing A.1: Exact Pose Message

```
BEGIN HEADER
    time 99:99:99.999
    date 99999999
    ua_id ZZ−999_ZZ
    msg_num 999
    msg_version 9.99
END HEADER

BEGIN POSE
    c1_x 9.999
    c1_y 9.999
    c1_z 9.999
    c2_x 9.999
    c2_y 9.999
    c2_z 9.999
    c3_x 9.999
    c3_y 9.999
    c3_z 9.999
    c4_x 9.999
    c4_y 9.999
    c4_z 9.999
END POSE
```

Listing A.2: Kill Message

```
BEGIN HEADER
    time 99:99:99.999
    date 99999999
    ua_id ZZ−999_ZZ
    msg_num 999
    msg_version 9.99
END HEADER

BEGIN MESSAGE
    kill 9
END MESSAGE
```

Listing A.3: Obstacle Avoidance Message

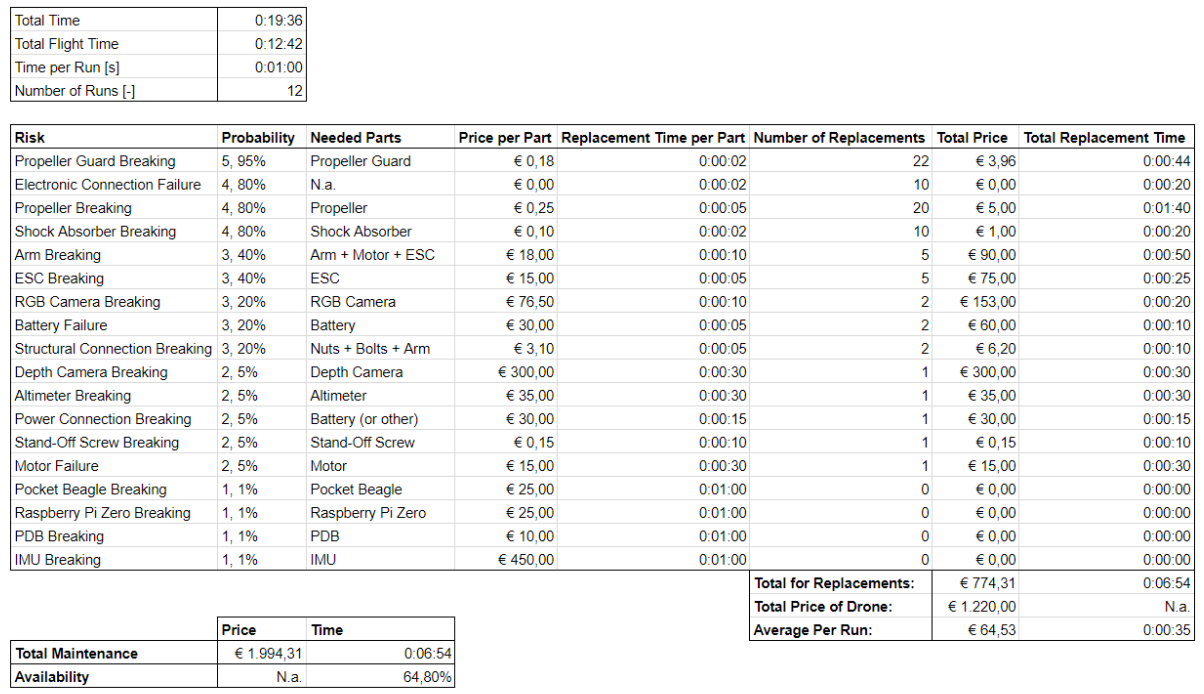
```
BEGIN HEADER
    time 99:99:99.999
    date 99999999
    ua_id ZZ−999_ZZ
    msg_num 999
    msg_version 9.99
END HEADER

BEGIN OBSTACLE
    obs 9
    obs_x 9.999
    obs_y 9.999
    obs_z 9.999
    width 9.99
    height 9.99
END OBSTACLE
```

[1] *http://www.barnardmicrosystems.com/media/presentations/IET_UAV_C2_Barnard_DEC_2007.pdf* [Visited at 11 June 2018]

Listing A.4: Telemetry Message

```
BEGIN HEADER
    time 99:99:99.999
    date 99999999
    ua_id ZZ-999_ZZ
    msg_num 999
    msg_version 9.99
END HEADER

BEGIN POSE
    x_pos 99.999
    y_pos 99.999
    z_pos 99.999
    pitch 999,99
    roll 999,99
    yaw 999,99
    x_vel 99.999
    y_vel 99.999
    z_vel 99.999
END POSE

BEGIN MOTOR
    id 1
    set_speed 99999
    actual_speed 99999
END MOTOR

BEGIN MOTOR
id 2
set_speed 99999
actual_speed 99999
END MOTOR

BEGIN MOTOR
    id 3
    set_speed 99999
    actual_speed 99999
END MOTOR

BEGIN MOTOR
    id 4
    set_speed 99999
    actual_speed 99999
END MOTOR

BEGIN BATTERY
    voltage 99.999
    charge 99.9
    power_use 999.9
    temp 999.9
END BATTERY

BEGIN STATUS
    mode 9
    connection 99.9
    sw_error 999
    beagle_temp 999.9
    pi_temp 999.9
    pdb_temp 999.9
    gate_num 9
END STATUS
```

# B
# Result of Availability Anaysis



| | |
|---|---|
| Total Time | 0:19:36 |
| Total Flight Time | 0:12:42 |
| Time per Run [s] | 0:01:00 |
| Number of Runs [-] | 12 |

| Risk | Probability | Needed Parts | Price per Part | Replacement Time per Part | Number of Replacements | Total Price | Total Replacement Time |
|---|---|---|---|---|---|---|---|
| Propeller Guard Breaking | 5, 95% | Propeller Guard | € 0,18 | 0:00:02 | 22 | € 3,96 | 0:00:44 |
| Electronic Connection Failure | 4, 80% | N.a. | € 0,00 | 0:00:02 | 10 | € 0,00 | 0:00:20 |
| Propeller Breaking | 4, 80% | Propeller | € 0,25 | 0:00:05 | 20 | € 5,00 | 0:01:40 |
| Shock Absorber Breaking | 4, 80% | Shock Absorber | € 0,10 | 0:00:02 | 10 | € 1,00 | 0:00:20 |
| Arm Breaking | 3, 40% | Arm + Motor + ESC | € 18,00 | 0:00:10 | 5 | € 90,00 | 0:00:50 |
| ESC Breaking | 3, 40% | ESC | € 15,00 | 0:00:05 | 5 | € 75,00 | 0:00:25 |
| RGB Camera Breaking | 3, 20% | RGB Camera | € 76,50 | 0:00:10 | 2 | € 153,00 | 0:00:20 |
| Battery Failure | 3, 20% | Battery | € 30,00 | 0:00:05 | 2 | € 60,00 | 0:00:10 |
| Structural Connection Breaking | 3, 20% | Nuts + Bolts + Arm | € 3,10 | 0:00:05 | 2 | € 6,20 | 0:00:10 |
| Depth Camera Breaking | 2, 5% | Depth Camera | € 300,00 | 0:00:30 | 1 | € 300,00 | 0:00:30 |
| Altimeter Breaking | 2, 5% | Altimeter | € 35,00 | 0:00:30 | 1 | € 35,00 | 0:00:30 |
| Power Connection Breaking | 2, 5% | Battery (or other) | € 30,00 | 0:00:15 | 1 | € 30,00 | 0:00:15 |
| Stand-Off Screw Breaking | 2, 5% | Stand-Off Screw | € 0,15 | 0:00:10 | 1 | € 0,15 | 0:00:10 |
| Motor Failure | 2, 5% | Motor | € 15,00 | 0:00:30 | 1 | € 15,00 | 0:00:30 |
| Pocket Beagle Breaking | 1, 1% | Pocket Beagle | € 25,00 | 0:01:00 | 0 | € 0,00 | 0:00:00 |
| Raspberry Pi Zero Breaking | 1, 1% | Raspberry Pi Zero | € 25,00 | 0:01:00 | 0 | € 0,00 | 0:00:00 |
| PDB Breaking | 1, 1% | PDB | € 10,00 | 0:01:00 | 0 | € 0,00 | 0:00:00 |
| IMU Breaking | 1, 1% | IMU | € 450,00 | 0:01:00 | 0 | € 0,00 | 0:00:00 |
| | | | | | **Total for Replacements:** | € 774,31 | 0:06:54 |
| | | | | | **Total Price of Drone:** | € 1.220,00 | N.a. |
| | | | | | **Average Per Run:** | € 64,53 | 0:00:35 |

| | Price | Time |
|---|---|---|
| **Total Maintenance** | € 1.994,31 | 0:06:54 |
| **Availability** | N.a. | 64,80% |

Figure B.1: Final Iteration of the Availability and Maintainability Script