Platypus: A Bio-inspired Amphibious Robot for Seaweed Aquaculture

Nick Yu

Delft University of Technology

TU Delft

Platypus: A Bio-inspired Amphibious Robot for Seaweed Aquaculture

by

Nick Yu

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on August 28, 2024 at 1:30 p.m.

Student Number:	4694740	
Thesis Committee:	Dr. R. Venkatesha Prasad	TU Delft, Supervisor
	Dr. ir. M. Taouil	TU Delft
	Ir. S. Sharma	TU Delft, Daily supervisor

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Preface

This document contains the final report for my master's thesis. This article can therefore be read as a standalone document describing the research that has been conducted. This thesis incorporates work presented in the drafted paper with the same title "Platypus: A Bio-inspired Amphibious Robot for Seaweed Aquaculture", which will be submitted to *Science Robotics* at the end of 2024.

I am deeply grateful to the many individuals who have supported me throughout this challenging and rewarding process. First and foremost, I would like to express my gratitude to my supervisor, Professor Ranga Rao Venkatesha Prasad for supporting my work. I also wish to thank Ir. Suryansh Sharma for his continued guidance and support during the whole project; without your insights and daily advice, this thesis would not have been possible. I would also like to thank Professor Mottaqiallah Taouil for serving on my committee and taking the time to review my work. Next I would like to thank Ir. Jane Ramirez who assisted with many design challenges during this project.

A special thanks goes to my family and friends for their unwavering support and patience. Your belief in me has been a constant source of motivation, and I could not have achieved this milestone without you.

Finally, I would like to thank the Delft University of Technology for providing me with the opportunity to work on hard and interesting problems in a stimulating and supportive environment. The knowledge and skills I have acquired during my time here will undoubtedly shape my future endeavors.

Nick Yu
Delft, August 2024

Summary

Seaweed cultivation has emerged as a vital component of sustainable development, playing a key role in enhancing global food security, carbon sequestration, and marine ecosystem health. The United Nations Sustainable Development Goals highlight the potential of seaweed farming as a sustainable resource, yet the current methods of underwater farming employ labor-intensive and time-consuming interventions necessitating the need for alternative advanced monitoring and sensing systems.

In response to this need, we introduce an innovative amphibious robot, inspired by the unique locomotive abilities of the platypus, designed to navigate both aquatic and terrestrial environments. This robot, named Platypus, incorporates specialized leg structures that allow for stable terrestrial locomotion and efficient aquatic propulsion, mimicking the platypus's natural adaptations. The robot's design includes a modular framework, a custom modular Platynode PCB for distributed control, and advanced sensors for comprehensive environmental monitoring. Additionally, a computer vision pipeline enables semi-autonomous seaweed monitoring. Field trials demonstrate the robot's effectiveness in sensing and monitoring seaweed, highlighting its potential as a versatile tool for optimizing seaweed farm management and contributing to the broader goals of sustainable aquaculture.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
IC	Integrated Circuit
IMU	Inertial Measurement Unit
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
SoC	System on Chip
I2C	Inter-Integrated Circuit
LiPo	Lithium Polymer (battery)
BLE	Bluetooth Low Energy
NIR	Near-Infrared
EC	Electrical Conductivity
ADC	Analog to Digital Converter
UART	Universal Asynchronous Receiver-Transmitter

Symbols

Symbol	Definition	Unit
V	Voltage	[V] (Volts)
I	Current	[A] (Amperes)
f	Frequency	[Hz]
α	Stride angle	[$^\circ$]
F_p	Propulsive Force	[N] (Newtons)
η	Efficiency	[N/W]
P	Power	[W] (Watts)
T	Time period	[S] (seconds)
B	Magnetic field strength	[μT] (microtesla)
b_x, b_y, b_z	Components of the magnetic field vector	[μT] (microtesla)

1

Introduction

The growing importance of seaweed cultivation, particularly in the context of sustainable development, cannot be overstated. As one of the fastest-growing sectors in aquaculture, seaweed farming plays a critical role in addressing global food security, carbon sequestration, and marine ecosystem health [17, 6, 11]. The United Nations Sustainable Development Goals (USDG) emphasize the potential of seaweed as a sustainable resource that can contribute to various environmental and economic benefits [54]. However, the complexity of seaweed farming, which varies across different farm types and depths, necessitates advanced monitoring systems to ensure optimal growth and harvesting conditions [27]. Robots designed to monitor these underwater farms must be capable of navigating through diverse marine environments, often requiring sophisticated adaptations to function efficiently at varying depths and under different conditions [50, 51].

1.1. State of the art amphibian robots

In recent years, the development of amphibious robots has made significant strides, offering promising solutions for environments that demand versatility in both land and aquatic navigation. The current state-of-the-art amphibian robots are typically optimized for terrestrial or aquatic environments. Some robots are designed with interchangeable legs, allowing them to transition between walking on land and swimming. Others, such as the EPFL salamander robot [9], demonstrate proficiency in shallow water swimming but struggle with navigating more complex terrains, such as steps or undulations near the surface. This limitation underscores the need for a more adaptable robotic design to effectively manage the challenges posed by both land and aquatic environments. Nature is often used as a base for design as many creatures possess the ability to walk and swim efficiently, an example of a robot using biomimetic design is the cuttlefish robot [1], by utilizing an undulating fin it propels itself underwater and walks on flat terrains, it however fails when faced with basic obstacles in terrestrial environments.

Figure 1.1: Platypus prototype while walking outdoors in a field trial (top) and a close-up of the front and side view (bottom)(video [43]).

1.2. Taking inspiration from nature

Drawing inspiration from nature, the platypus is an excellent model for designing amphibious robots. The platypus, an extraordinary creature, seamlessly transitions between swimming in water and walking on land. Its unique leg structure, featuring webbed feet that act as paddles, allows it to efficiently maneuver through aquatic environments, while its strong limbs enable it to traverse land easily [16]. Similarly, ducks and other waterfowl utilize flapped paddle techniques that provide both propulsion in water and stability on land. These natural designs offer valuable insights into creating robotic systems that can thrive in both domains.

1.3. Key features of PLATYPUS

Inspired by the platypus, our robotic system incorporates several key features that enhance its amphibious capabilities. The robot's specialized leg design allows it to walk on land with stability while swimming with remarkable efficiency. The robot joints provide an extended range of motion, enabling smooth transitions between land and water. Its modular design, featuring symmetry in the head and tail, ensures balanced movement and adaptability. Additionally, the robot is equipped with advanced sensors capable of recording key environmental parameters and capturing video footage, making it an invaluable tool for monitoring seaweed farms and other marine ecosystems. This combination of bio-inspired design and cutting-edge technology positions our Platypus as a versatile solution for the challenges of modern seaweed farming.

Contributions:. Our key contributions are listed below:

❶ System design of the amphibian robot Platypus including mechanical, electronic and software subsystem design. We also build a custom modular Platynode PCB for robust distributed control (Sec. 3).

❷ Control and design of a novel amphibious leg that allows for motion both inside and outside water (Sec. 4).

❸ Development of a sensor box for sensing different underwater parameters for seaweed aquaculture, including a computer vision pipeline for semi-autonomous seaweed monitoring (Sec. 5).

❹ Performance evaluation of the robot locomotion in field trials and aquaculture monitoring in a real seaweed farm (Sec. 5 and Sec. 4).

2

Related work

Bio-inspired Robots and Mechanics. The mechanics of bird wings inspire our amphibian leg design, where feathers open during the upstroke and close during the downstroke, creating an asymmetry in drag forces [57, 8]. Similar principles of asymmetric drag forces in aquatic mechanisms are discussed in [10]. Our design features a curved leg equipped with a paddle, enabling the robot to swim on the water's surface using circular strokes. During one-half of the stroke cycle, the paddle remains above the water's surface, and during the other half, it submerges, generating an asymmetric drag force. However, this design is limited to surface swimming, as the net drag force is zero when fully submerged, preventing effective underwater propulsion.

In [26, 30], a different approach involves using rigid flaps in the actuator, which open and close during the power and recovery strokes. Another method for achieving asymmetry is through a spring-loaded passive rowing joint, as described in [3], where the joint sweeps back to reduce resistance during the recovery stroke. Conventional fins can also produce asymmetric drag by using skewed actuation signals, which involve a faster power stroke and a slower recovery stroke, as explored in [14, 32, 34, 36]. However, this method has a significant drawback: the need to reduce the power stroke's duty cycle, leading to intermittent and sluggish motion.

In [1], a robot inspired by a cuttlefish is designed, which can be seen in Fig. 2.1. The robot uses flexible undulating fins that move as a shifting wave. Notable is that the robot performs extremely well underwater, but is not optimized for terrestrial terrains other than flat surfaces.

Amphibian Robots. In [10] they developed an amphibian robot by enhancing an existing leg design used in hexapod robots. A combination of a paddle and circular leg allows the robot to swim, walk, and transition. A drawback of this design is that the leg essentially sinks in sand terrains. They combat this by adding a flat sheet to the end of the leg but adds additional drag during swimming. Another design using fin-based legs is presented in [58]. In the field of biomimetics, var-

Figure 2.1: Cuttlefish-inspired amphibian robot

Figure 2.2: A salamander-inspired amphibian robot from EPFL

ious amphibian robots have been inspired by creatures such as snakes, crabs, and turtles [35, 22, 23, 24]. These design all optimize the robot for one of the two terrains often compromising maneuverability in one environment for the other.

Seaweed Aquaculture and Monitoring. In [40] they used an off-the-shelf mini-ROV to capture images of seaweed suspended on buoy lines, applying computer vision techniques to estimate the size of the growing kelp. However, deploying an ROV from shore still requires considerable human labor.

Similarly, in [51] they designed an Autonomous Underwater Vehicle (AUV) for inspecting seaweed farms on the coast of Sweden. Designed to operate fully autonomously, it uses side-scan sonars to navigate between seaweed grow lines. In contrast, [42] introduces a low-cost sensor system specifically for monitoring purposes. This system logs various metrics such as temperature, light intensity, depth, and motion and is compact enough to be attached to seaweed mooring lines at different depths. However, a key limitation is that the sensor system must be retrieved from the farm to offload the logged data.

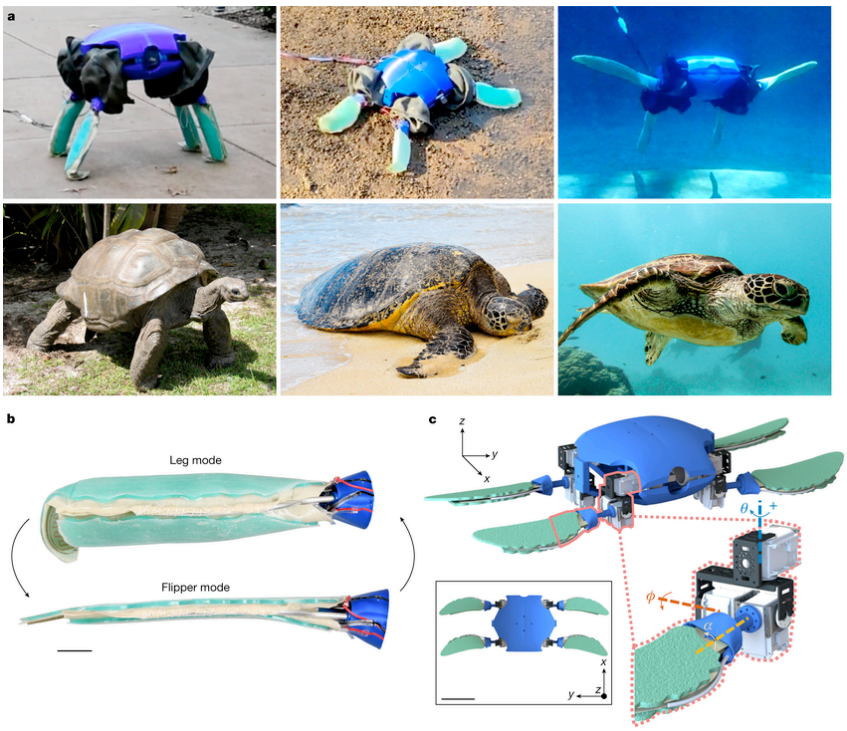


Figure 2.3: ART (Amphibious Robotic Turtle) from [2]

3

Robotic System Design

The goal of Platypus is to offer a platform that can be used to traverse different terrains on land and swim in water while carrying a payload of varied sensors. These locomotion capabilities reinforce its utility for seaweed aquaculture and underwater sensing by allowing it to move through different environments while collecting data. It is dictated by the application scenario that Platypus meets certain design goals.

The design goals of Platypus are summarized as follows:

❶ **Versatile locomotion.** The robot should be able to move through different terrains, including uneven land, inclinations, and steps. This covers the different scenarios it might face when conducting seaweed monitoring activities.

❷ **transition between land and water.** The robot should be able to transition between water and land without external assistance. This means the robot should have the necessary mechanisms to traverse both environments.

❸ **Optimize size/weight with sensor payload.** The robot must carry the sensors required to monitor seaweed. The robot must be small enough to maneuver the lines of planted seaweed and should not exceed 1.5 m in length. Further, the robot's weight should be such that it can be neutrally buoyant when swimming at a depth of 1 m.

❹ **Waterproof body and joints.** The platform must possess a watertight body that protects from water ingress at different joints, body segments, and motors. The robot does not need to dive deeper than a few meters.

❺ **Simple, robust robot design.** The platform should be accessible, easily replicated, and beneficial to the seaweed grower community. This means no complex actuator or mechanical design should be incorporated.

We consider the fundamental design choices and constraints mentioned above while designing Platypus. It consists of a 3 segment design, with each segment containing two amphibian legs and high-torque servos connecting the different

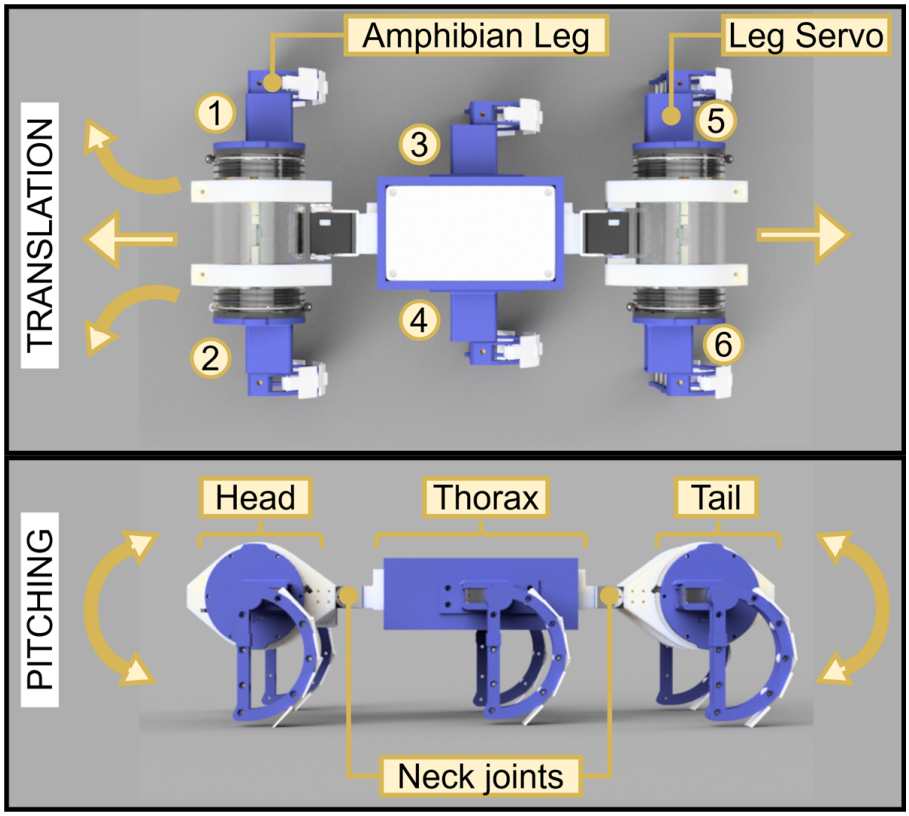


Figure 3.1: Platypus design with all components and degrees of freedom possible with its 3 segmented body and articulated neck joints. The neck joints allow for head/tail pitching. This range of motion assists in walking on land and swimming underwater using its special amphibian D-shaped legs.

segments together to form neck joints. This results in a segmented hexapod design with 6 specially designed D-shaped legs with flexible flaps that allow the robot to walk like other terrestrial hexapods like RHex [47] while also swimming using the bio-inspired flapped paddle design [49]. This design can be seen in Fig. 3.1. In the following sections, we highlight the novel design choices made to design Platypus.

3.1. Nature-inspired 3 Segment Body Design

Animals that swim often have a body design that allows them to swim and maneuver inside water. For example, animals like seals and sea lions have flexible bodies with articulated joints that help them twist and turn in the water. These marine animals rely on flexible joints in their spines and necks for agility and motion [4]. Another example is that of otters who use a combination of paddles and undulating movements of their bodies and tails to swim. They use their flexible spine and joints to maneuver inside water [31]. This non-monolithic design gives us a clue for a useful feature when designing Platypus.

Platypus takes inspiration from nature and is built using three separate segments: a head, a thorax, and a tail interconnected by servo-actuated neck joints. The head and tail segments are identical and comprise two legs on either side and a

hollow, transparent cylindrical body that houses the robot control PCB - Platynode and 2 Waveshare 360° ST3020 servo motors with 25 Kg-cm torque [55] for the amphibian legs.

The head and tail segments are built around the cast acrylic 75 mm diameter watertight locking cylinders from blue robotics [45], which use specialized aluminum end caps with integrated rubber o rings to ensure water tightness up to 120 mm depth. The end caps are coated with Molykote 111 silicon grease [13] to ensure the water tightness of the seal. The motors and the legs are held together at the outer cap using a custom 3D-printed mount with only the wires from the servos penetrating the cylinder. These are epoxied to ensure a watertight seal, while the servos are waterproofed using marine grease and conformal coating. The head/-tail segment is self-contained and houses its own sensing and control electronics and independent battery as a power source. The cylinders are transparent, allowing the head/tail segments to contain any cameras and LED lights needed for underwater operation.

The head/tail segments are attached to the middle-thorax segment using 2 Waveshare ST3025 high torque servo motors with 40 Kg-cm torque [56]. The neck joint uses two 3D-printed mounts that attach the head/tail segments to the servo's rotary axis. The distance between the head/tail segments and the motor drive axis is minimized to reduce the moment arm the motor needs to drive. The higher torque requirements stem from the need for the neck servos to lift the weight of the head/tail segments with all electronics and any added ballast weights that maintain its buoyancy. This is important as the robot needs to maintain its orientation, especially when pitching the head or tail. These linkages allow the head/tail segments to be actuated in the pitch axis, as shown in Fig. 3.1. If the motors are mounted vertically, the head moves in the horizontal plane akin to a yawing motion. However, we mount the servos horizontally, leading to the head moving in the vertical plane akin to a pitching motion.

The thorax is the middle segment, which contains the two legs on either side and houses the neck servos that link to the head/tail segments. The middle segment is built around an 160 mm x 90 mm IP68 polycarbonate watertight box that houses the Platynode control PCB and the sensor payload for seaweed monitoring. It powers and controls the high-torque neck servos and contains an independent battery like the head/tail segments. The description of the Platynode PCB that controls each segment is presented in Sec. 3.2, and the sensor payload description, including the choice and design of different sensors, is given in Sec. 5.

3.2. Design of a Modular Platynode

The design of Platypus is based on individually controlled and powered segments: a head, thorax, and tail. These are all controlled by a self-contained controller and sensing PCB called Platynode. Each Platynode consists of basic system blocks: sensors, memory and computation units, power management, and actuators, as shown in Fig. 3.2. The platform uses an Inertial Measurement Unit (IMU) and a magnetometer to infer its acceleration and orientation. We use BMI088, a 6-axis

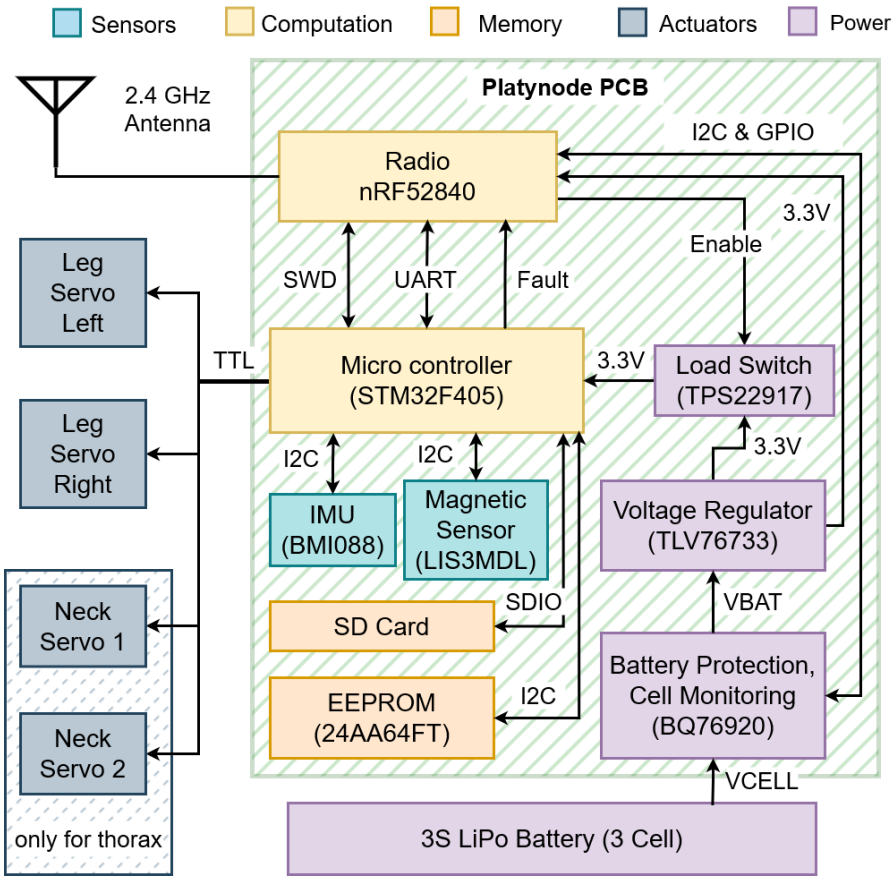


Figure 3.2: System components of Platypus's modular robot control PCB - Platynode.

inertial sensor with a highly accurate 16-bit digital triaxial accelerometer and a 16-bit digital triaxial gyroscope. We use the LIS3MDL 3-axis digital output magnetic sensor to measure magnetic fields and derive the segment's orientation. The STM32F405 microcontroller is the main processing unit for controlling the robot by giving the desired actuation commands to the two leg servos and, in the case of the thorax, additionally to the two neck servos as well.

The main actuators for achieving all the motion of Platypus are $360°$ DC powered servo motors that control the amphibian legs and two high torque servos for the neck joints. The servo motors are rated for up to 14 V and feature a high-precision magnetic encoder with 12-bit angular position feedback. The servos use TTL logic and can be driven using a shared bus. These servos can also be programmed to provide information about the current draw and the estimated load they drive.

Based on the high-level control algorithm, the microcontroller generates the control signals to drive these servos. Using the position feedback, it achieves closed-loop control of each leg and neck joint. Since each segment contains its own Platynode and is independently powered, the system adds robustness by adopting a distributed control strategy to drive its 3 pairs of legs.

As each pair of legs is individually actuated, there is a need to synchronize them

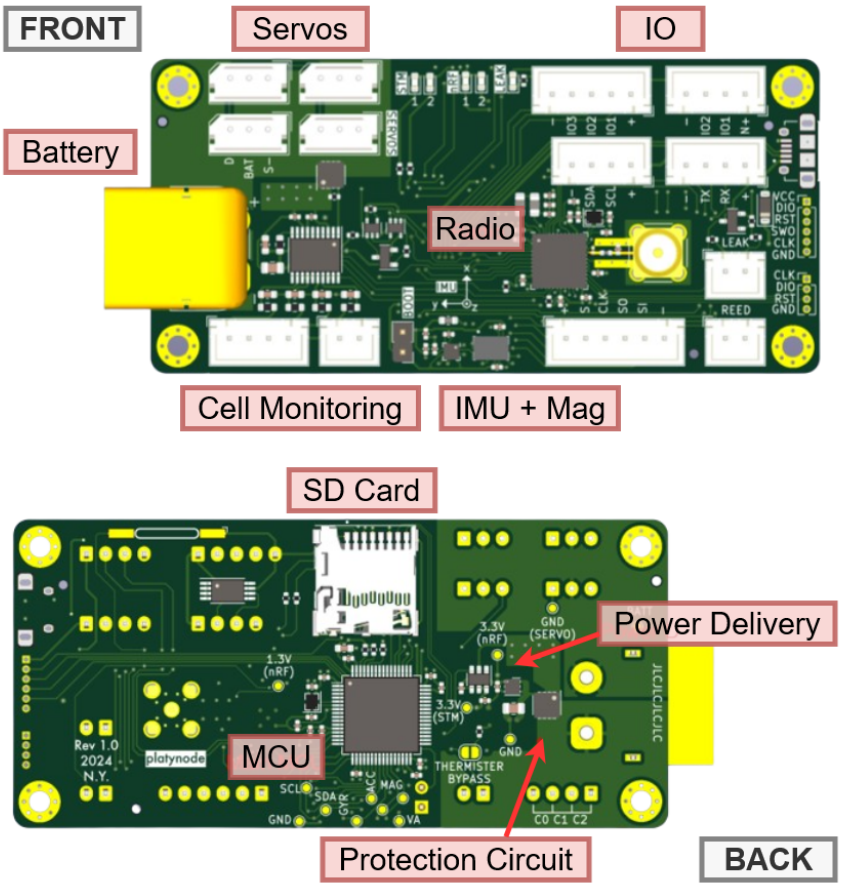


Figure 3.3: Custom-developed Platynode PCB with different components.

and share data wirelessly. This is enabled using a low-power, long-range radio that can communicate with other nodes to exchange control parameters and flash over-the-air firmware when the robot is assembled. We used a 2.4 GHz nRF52840 SoC for this purpose. The physical layer (PHY) communication channel design and clock synchronization method developed to enable message exchange between the different segments are described in Sec. 3.3 and Sec. 3.4, respectively.

A 2200 mAh 12.6 V 3S LiPo battery powers the servos and the entire electronic system. Power management and regulator ICs ensure the voltage is converted to 3.3 V for powering the different ICs. The individual battery cell voltages are also monitored using a BQ76920 IC that can communicate the state of these cells over I2C. The nRF52 radio is also responsible for shutting down the entire subsystem if a fault is detected or the robot operator commands it to do so.

3.2.1. Magnetometer Characterization

We use a magnetometer in Platypus to find the orientations of the whole robot and each individual segment. This information is useful when the robot needs to move over obstacles or climb up (or down) stairs. Another use-case would

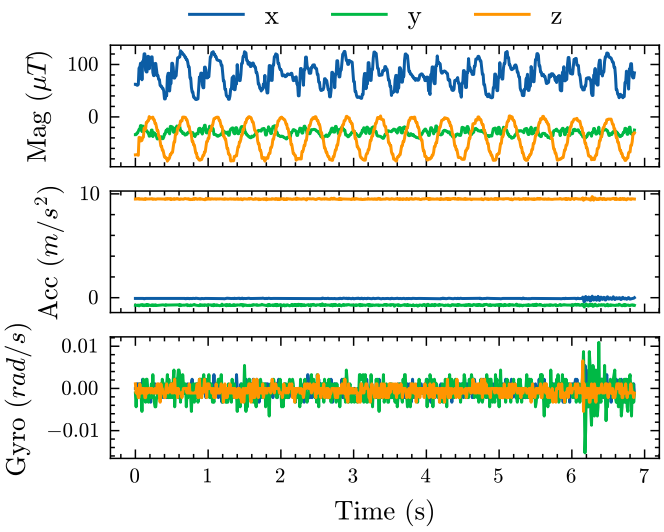


Figure 3.4: Variation of magnetometer, accelerometer, gyroscope readings of a Platynode in the presence of servos with constant velocity $\omega = 4\pi\ rad/s$.

be underwater navigation because traditional localization methods like GPS do not work inside water. Typically, divers can navigate inside water using a compass and visual landmarks. Platypus can utilize a similar strategy. However, it requires a reliable magnetic sensor reading. Magnetometers are notorious for being affected by stray magnetic fields and producing erroneous values. This leads us to investigate the use of a magnetometer on Platynode.

We characterize how accelerometer, gyroscope, and magnetometer readings are affected by the presence of an active and idle servo motor. In our experiment, we mounted a Platynode on a flat surface with the IMU and magnetometer's z-axis facing upward. Three servos were positioned 5 cm away from the IMU, aligned along the negative y, positive y, and positive x axes of the IMU's coordinate frame. Sensor data were sampled at 100 Hz under four different conditions: with no servos nearby, with all servos idle, with servos moving at a constant velocity, and with servos operating at a velocity modulated by a sine wave.

Across all conditions, the accelerometer and gyroscope readings remained unaffected. However, the presence of servos introduced a constant offset in the measured magnetic field, attributed to the servos' internal permanent magnets. Notably, the magnetometer readings were significantly affected by the movement of the servos, while the accelerometer and gyroscope were immune to the electromagnetic interference (EMI) caused by the servos. This effect can be seen in Fig. 3.4, which shows the recorded values from the accelerometer, gyroscope, and magnetometer.

Further analysis of the magnetometer reveals that it is affected by two types of effects: soft iron and hard iron effects. ① Soft iron refers to materials that can be easily magnetized and demagnetized. When a magnetometer is exposed to a soft iron material, it can become temporarily magnetized in response to the ex-

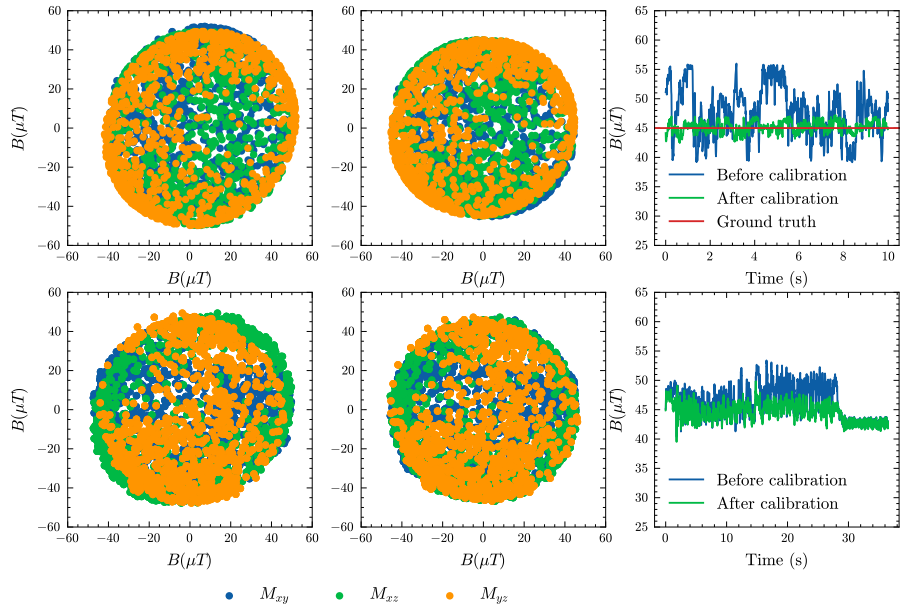


Figure 3.5: Magnetometer readings with and without our calibration method on simulated data (top) and real magnetometer reading data from a Platynode (bottom). From left to right: the measurement locus of the XY, YZ and XZ before calibration, after calibration, and magnetometer measurements over time. After calibration the measurements should converge to a single value, which is the earth's magnetic field strength.

ternal magnetic field. This results in a non-linear distortion of the magnetic field measurements. The soft iron material acts like a variable-strength magnet that modifies the original magnetic field. The effect is often characterized by changes in both the amplitude and orientation of the magnetic field vector. ② Hard iron, on the other hand, refers to materials that are magnetized and retain their magnetization over time. When a magnetometer is exposed to a hard iron material, it experiences a constant offset in its readings. The hard iron effect shifts the entire magnetic field measurement, introducing a fixed bias to the sensor readings. Unlike the soft iron effect, the hard iron effect does not involve changes in the shape or linearity of the magnetic field.

Soft iron effects can be corrected using calibration techniques that involve characterizing and compensating for the non-linear distortions introduced by the soft iron material. Hard iron effects can be corrected by subtracting a constant offset from the magnetometer readings. This offset is determined during calibration, where the magnetometer is exposed to known magnetic fields in different orientations.

A simplified model of the measured magnetic field can be described as

$$\mathbf{h_m} = \mathbf{A_{si}}\mathbf{h} + \mathbf{b_{hi}} \tag{3.1}$$

where $\mathbf{h}$ is the true magnetic field, $\mathbf{A_{si}}$ is a 3x3 matrix representing the soft iron effects and $\mathbf{b_{hi}}$ is the hard iron offset. From this it is clear that we can extract the true magnetic field if we have $\mathbf{b_{hi}}$ and $\mathbf{A_{si}^{-1}}$

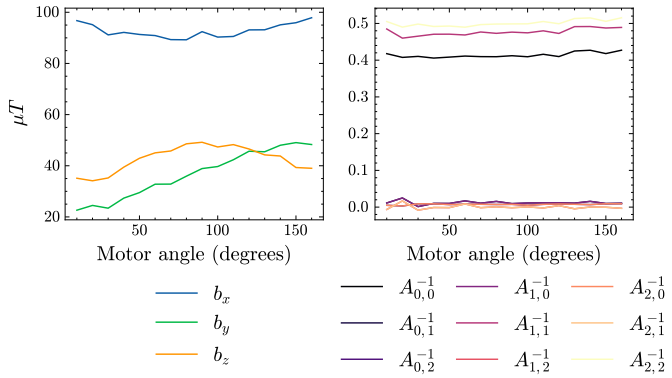


Figure 3.6: Hard iron offsets (left) and soft iron matrix parameters (right) for the magnetometer readings with varying servo position in a Platynode.

$$\mathbf{h} = \mathbf{A_{si}^{-1}}(\mathbf{h_m} - \mathbf{b_{hi}}) \tag{3.2}$$

Note that this model does not take into account stochastic noise or sensor non-linearities.

To characterize the two types of effects, we employ the Ellipsoid fitting method for magnetometer calibration, as described in [33]. With enough samples $\mathbf{h_m}$, we can estimate $\mathbf{b_{hi}}$ and $\mathbf{A_{si}^{-1}}$ using this algorithm. This method assumes that the Earth's magnetic field remains constant at any location, meaning its magnitude should remain unchanged regardless of the magnetometer's orientation. The effectiveness of the calibration method can be seen in Fig. 3.5, where it is applied on simulated data (using a magnetometer measurement model [41]) and real-world data.

Next, we estimate the soft iron and hard iron effects of a servo at different positions. We attach a Platynode to a servo and perform a figure-eight motion at every 10-degree increment of the servo's position. The calibration technique is then applied to each set of measurements. Fig. 3.6 shows the relationship between motor angle and $\mathbf{b_{hi}}$, $\mathbf{A_{si}^{-1}}$. The soft iron parameters remain constant, while the hard iron effects vary with motor angle. This suggests that after calibrating the system for known motor positions, we can approximate $\mathbf{b_{hi}}$ for all motor positions and effectively filter out the hard-iron offsets caused by the servo in the Platynode.

3.3. Wireless Communication Link Design

The different segments of the Platypus do not have any wiring between them to facilitate easy waterproofing and replacement in the field. However, since each segment only controls a pair of legs from the 6 total, the controllers must be synchronized to generate different walking or swimming gaits. This is facilitated by building a wireless communication link using the 2.4 GHz nRF52840 SoC. The radio is capable of using Bluetooth Low Energy (BLE) as well as custom PHY layer protocols. The choice of radio is based on the ease of connection and the perfor-

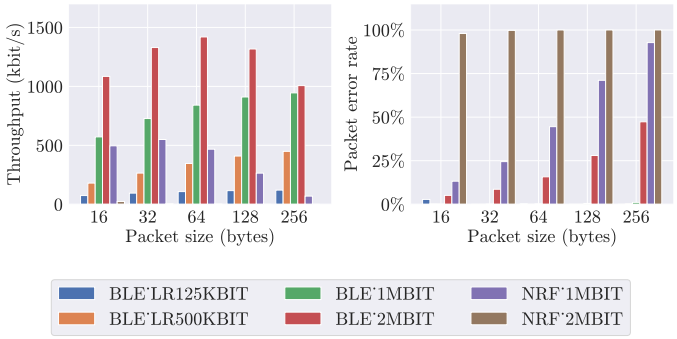


Figure 3.7: Variation of throughput and packet error with different packet sizes for different PHY protocols on the nRF52 underwater.

mance of the wireless technology both inside and outside water. Since the robot is deployed in the field, using a smartphone or laptop to connect to the internal control PCB is valuable.

However, the more important consideration is that the different Platypus segments will be immersed completely in water. This means that the communication link must operate both inside and outside water. The maximum distance the link must operate is the longitudinal length of the Platypus robot, which refers to the maximum separation between the head and tail segments. This corresponds to a 20 cm distance.

We test the link performance using the communication metrics of throughput and packet error rate for each PHY protocol available on the nRF52840 SoC. These include Long Range version of BLE with data rates of 125 Kbit/s, 500 Kbit/s, regular BLE with data rates 1 Mbit/s and 2 Mbit/s. We also included Nordic's proprietary 2.4GHz radio mode with data rates of 1 Mbit/s and 2 Mbit/s. Two nRF chip were placed 20ċm apart in a box filled with water at a depth of 30 cm and used to send packets using 2.4 GHz channels from 0 to 80 and with a transmit power from 0 to 8 dBm. The results from this characterization are shown in Fig. 3.7. From the graphs, we can see that the best performance out of all of the PHY protocols is for BLE_LR500KBIT in the 2.45GHz channel with 448KB/s throughput and 0.2% worst-case packet error rate for a packet size of 255 bytes.

We also characterize this PHY protocol's performance with varying transmit power frequency channels. The Received Signal Strength Indicator (RSSI), datarate, and packet error rate obtained are shown in Fig. 3.8. We can observe that we have best packet reception in the middle of the 2.4GHz band and a transmission power above 4dBm. This leads us to select BLE_LR500KBIT as the PHY protocol for the message exchange in Platypus with a transmission power of 4dBm and channel 40.

3.4. nRF Clock Synchronization

The high-level control algorithm that controls the gaits of the Platypus requires the different legs to be actuated simultaneously. This is trivial in traditional robots,

$\overline{RSSI}$ (*dBm*)

	0 dBm	2 dBm	4 dBm	6 dBm	8 dBm
ch 0	-93.55	-91.84	-90.10	-88.44	-86.86
ch 20	-92.48	-90.76	-88.74	-87.06	-85.33
ch 40	-91.17	-89.72	-87.63	-85.84	-84.08
ch 60	-89.49	-88.33	-86.24	-84.28	-82.55
ch 80	-88.82	-88.00	-85.90	-83.85	-82.09

$\overline{Datarate}$ (*KB/s*)

	0 dBm	2 dBm	4 dBm	6 dBm	8 dBm
ch 0	0.00	0.00	6.69	299.22	438.15
ch 20	0.00	83.01	422.40	447.70	448.84
ch 40	66.03	370.57	448.19	449.50	449.17
ch 60	382.24	437.82	448.52	449.41	449.33
ch 80	397.34	414.31	419.30	425.50	425.74

% Received ($\overline{crcok}$/sent)

	0 dBm	2 dBm	4 dBm	6 dBm	8 dBm
ch 0	0.0	0.0	0.015	0.67	0.97
ch 20	0.0	0.18	0.94	1.0	1.0
ch 40	0.15	0.82	1.0	1.0	1.0
ch 60	0.85	0.97	1.0	1.0	1.0
ch 80	0.88	0.92	0.93	0.95	0.95

Figure 3.8: RSSI, Datarate and Packet error rate for BLE Long Range 500 Kbit/s with varying transmit powers underwater.

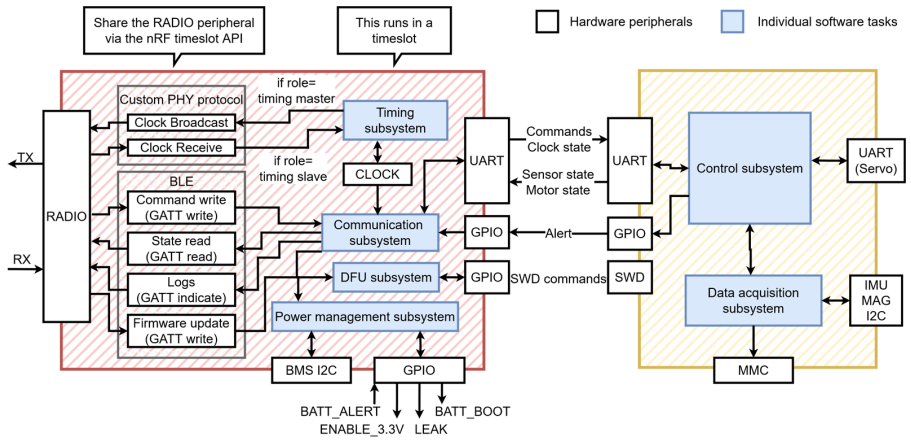


Figure 3.9: Software architecture running on a single Platynode. On the left is the nRF52 SoC, and on the right is the main MCU. We show the key software tasks that run on each Platynode and the hardware peripherals they interact with to make Platypus work.

but since Platypus uses a three-segment body design with wireless links interconnecting the different Platynode controllers, we require a method of synchronizing them to operate simultaneously.

We made an algorithm for syncing TIMER peripherals of the nRF52 radios in each

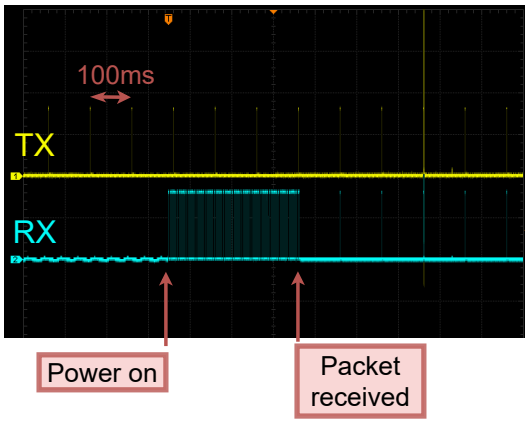


Figure 3.10: Oscilloscope capture of clock syncing, in this setup one Platynode is set as the timing master TX (yellow) and another as a node RX (cyan). When the RADIO starts transmitting or listening a high signal is generated.

Platynode. Our method is based on an existing technique from Nordic Semiconductor [29]. This technique uses an old Nordic SDK, which is not active anymore and thus requires rewriting and implementing on the newer nRF Connect [38] software stack.

Synchronizing nodes in a BLE network presents challenges. For example, broadcasting a clock value in a star topology with a central peripheral and multiple other nodes is difficult due to latency. When using BLE's connection-oriented unicasting, the delay between devices receiving a message we measured can be around 24 ms, and up to 75 ms with three nodes. While a consistent delay could be accounted for in clock synchronization, the latency is not deterministic, making synchronization unreliable. Alternatively, using BLE's advertising and scanning features to broadcast a message from a peripheral to central nodes also has issues. Advertising and scanning can occur at different intervals, with scanning done in a "scan window" and advertising on three channels in the 2.4 GHz band. This variability means that there's no guarantee all central nodes will receive the broadcast simultaneously. A possible solution is to continuously advertise the clock timing and proceed with other operations only after all peripherals confirm receipt of a synchronization packet.

Our method operates as follows: a single Platynode is designated as the "timing master," which maintains the global clock for the entire network. This timing master periodically captures and broadcasts its clock at 100 ms intervals. The remaining nodes listen for these broadcasts within a brief reception window. Due to the short window and the long interval between broadcasts, packet reception is infrequent. To address this, non-master nodes alternate between two states: SYNCED and DESYNCED. In the DESYNCED state, nodes listen more frequently until they receive a packet, at which point they transition to the SYNCED state. Once in the SYNCED state, nodes align their reception windows with the known broadcast interval. In Figure 3.10 we show how the interaction works.

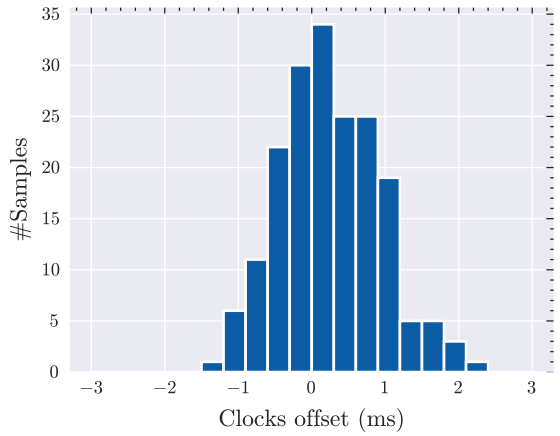


Figure 3.11: Distribution of average PWM phase difference measured in a window of 30 s between two Platynodes using our synchronization method over a period of 30 minutes.

Our method can synchronize two or more Platynodes while not monopolizing the RADIO peripheral, i.e. one can run other radio protocols simultaneously as this sync method is running in the background. It is also optimized for power consumption as the receive window of the nodes is synchronized with the send window of the master, allowing you to use a small reception window.

We characterize how well this method works by having 2 Platynodes output a PWM signal generated from synchronized TIMERs. We estimate the clock difference as the phase offset of the PWM signals. A PWM is created using the nRF's 1 KHz timer, with a duty cycle of 100 ticks / 10000. The clock drift can be estimated as the phase shift of the PWM signals from 2 boards. We use a Kingst LA1010 logic analyzer to measure the two signals simultaneously. The logic analyzer is set at a sampling rate of 10 MHz and 100 MSa.

We measure the average difference between the TIMERs of the two Platynodes. Each sample in Fig. 3.11 is the average PWM phase difference measured in a window of 30 s. Over a period of 30 minutes, the maximum difference observed is 2 ms. The average clock difference between the nodes over 30 minutes is 0.92 ms. We also characterize how long it takes for this algorithm to sync from startup. A node synchronized on average in 0.17 seconds after power cycling.

3.5. Software Architecture Design

Platypus consists of three Platynodes, one in each segment: head, thorax, and tail, linked together by wireless BLE links. These nodes all receive commands from an external commander, which can be either a single-board computer like a Raspberry Pi that processes input from the user or the autonomous robot controller. This main controller converts the high-level gait into required servo actuation signals for each pair of legs and/or the neck joint servos. It then transmits the signal to three Platynodes in each robot segment. This is done using a connection-oriented BLE for each Platynode. The high-level gait controller can either generate the trajectory of the robot using a pre-programmed route or us-

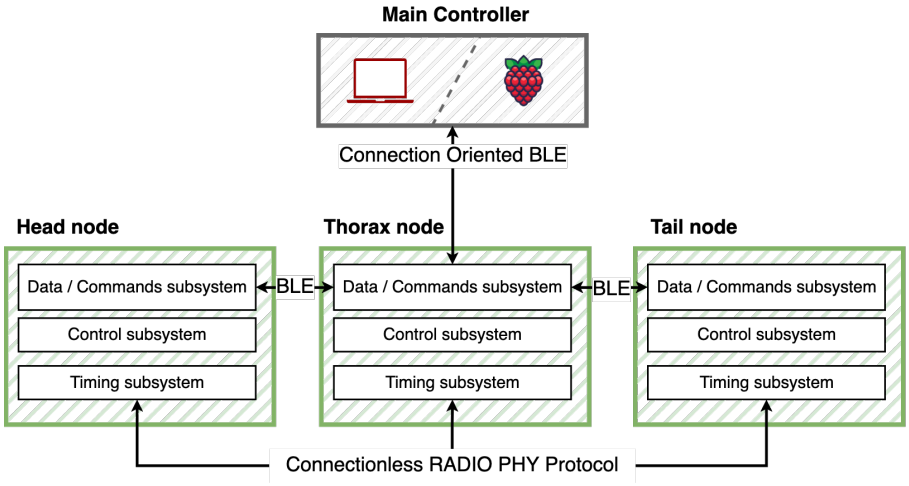


Figure 3.12: High-level software architecture of the distributed software blocks running on the different Platynodes.

ing the feedback from the sensor box. Suppose the robot relies on the sensor data to process the trajectory. In that case, the Raspberry Pi of the sensor box shall act as the main controller for generating low-level actuation signals for each Platynode.

Fig. 3.12 shows the high-level software architecture of Platypus. The main controller communicates with the thorax node, high-level commands can be given, such as changing the system state from **idle** to **homing** (moving the motors to their start position), **walking**, or **turning**. The controller can also change all control parameters on the nodes. Any message the thorax node receives is relayed to the other nodes.

Fig. 3.9 shows the software architecture running on a single Platynode. The firmware on the Platynode is built on top of the Zephyr RTOS [1], where subsystems are organized as their own task. We opt to minimize the number of periodic tasks by grouping functionality where possible. The nRF52 SoC on the Platynode is responsible for all communication and safety-critical systems, while the STM32 handles all control logic. By having this separation, the control loop of the Platynode is preempted as little as possible, allowing for a faster loop. Any commands received from the main controller, and also the nRF's clock are communicated to the STM32 over UART.

[1] https://zephyrproject.org/

4

Locomotion and Control

The high-level control system has two main objectives: (a) to manage the robot's gait during walking or swimming and (b) to maintain a desired yaw (heading) during turns. Our leg design includes flaps that remain inactive on land but transform the leg into an efficient paddle when submerged. These flaps simplify the control system required to actuate the robot. On land, we utilize the control laws developed for the well-known hexapod, RHex [46, 48], while in water, we employ simple oscillatory gaits.

We explore integrating a flapped-paddle design with RHex's curved C-shaped leg. Our approach effectively extends the functionality of the C-shaped leg, enabling smooth locomotion both in and out of water using straightforward rotary motion. Fig. 4.1 illustrates our design. This section details the amphibian leg's design and its locomotion application in aquatic and terrestrial environments.

4.1. Amphibian Leg Design and Characterization

The flapped paddle leg design is the key element of Platypus, enabling amphibian locomotion. We devise a novel design for amphibious locomotion that is:

❶ **Simple**: It does not require complex actuation mechanisms for transitioning across terrestrial and aquatic environments,

❷ **Efficient**: It completely utilizes the dominant drag forces for locomotion,

❸ **Versatile and Robust**: It enables the robot to traverse undulatory terrain, steep obstacles, staircases, and aquatic environments without complex control strategies or expensive sensing and perception.

Our design employs the flat, flapped paddle studied in [49] for underwater environments for a turtle-styled robot. They demonstrated that the propulsive efficiency in a conventional flipper is diminished because of its symmetric structure. To effectively move in aquatic media, the lateral drag force, F_y, which is significantly greater than the longitudinal force, F_x, must be completely utilized for propulsion, i.e., the drag force must not average to zero.

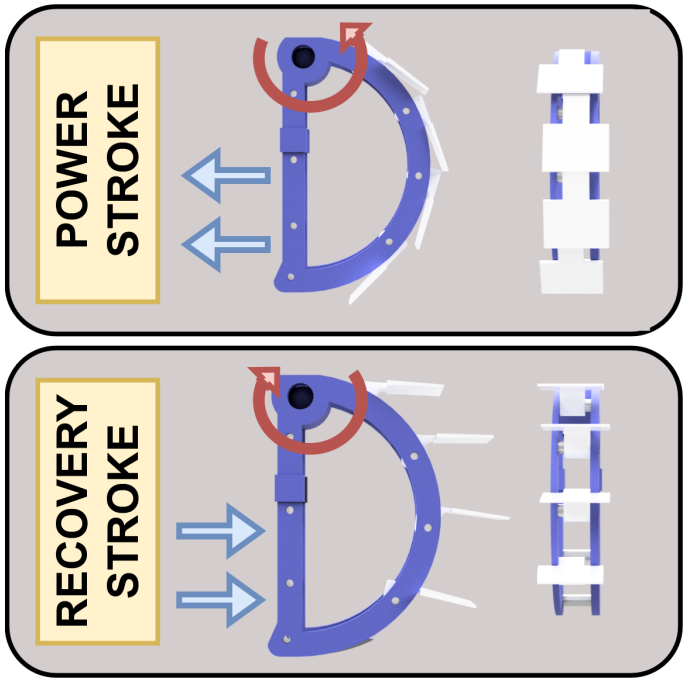


Figure 4.1: Our amphibian leg design with bio-inspired flaps allows swimming.

When a conventional flipper is used in aquatic environments, the drag force averages out to zero, and thus, even when the lateral force generated is larger than the longitudinal force, its utilization for propulsion cannot be achieved.

We offer a simple, passive, flapped-paddle mechanism that introduces asymmetry in the lateral force by design to address this challenge. The asymmetry in the flapped-paddle structure ensures that the drag force does not average to zero during the oscillation cycle, thereby enabling effective utilization of the generated forces for locomotion.

In many underwater robots, lateral drag forces—though not contributing to net thrust—are often much stronger than the longitudinal forces. This phenomenon is observed in the RHex-Aqua robot [19, 21]. Fig. 4.2 shows the forces acting on a conventional flipper and on a straight flapped paddle from [49]. Specifically, F_y represents the lateral drag force, and F_x represents the longitudinal force, which includes both drag and wake forces and is noticeably weaker. Our design addresses the limitations of conventional flippers. This design features flaps arranged in an overlapping, louver-like structure that opens during the recovery stroke and closes during the power stroke. This mechanism nearly fully asymmetrizes the previously dominant lateral drag force (depicted in Fig. 4.3), allowing it to be harnessed for underwater locomotion rather than relying on the weaker longitudinal forces. As a result, the flapped paddle-fin significantly improves propulsive efficiency and net thrust compared to conventional flippers.

Fig. 4.5 shows the underwater test rig we used. We measure the generated force and torque of the oscillatory gait of different leg designs using a torque bench.

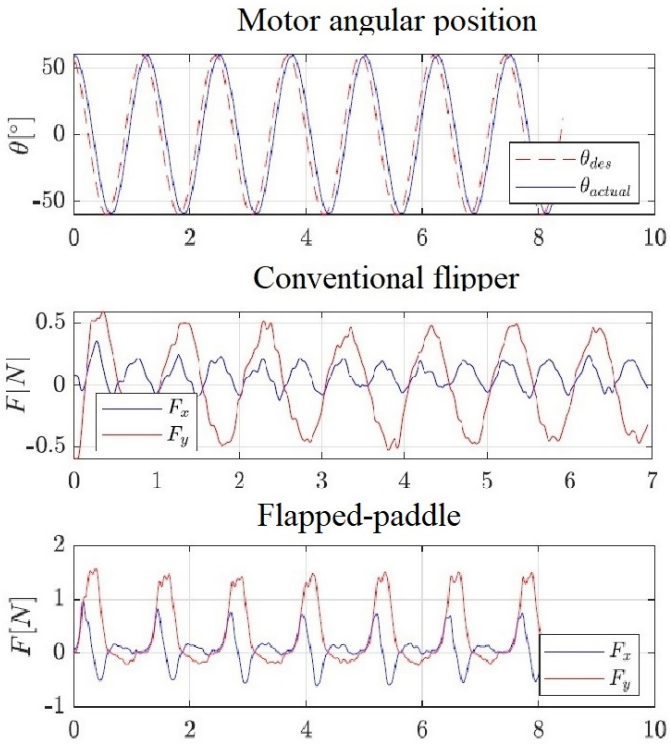


Figure 4.2: From [49], the symmetry in the lateral drag force (F_y) for the conventional flipper (top) results in zero average force in an aquatic environment. Notice that the lateral drag force for the straight flapped-paddle is suppressed in the recovery stroke, thus maximizing propulsive efficiency because $F_x < F_y$.

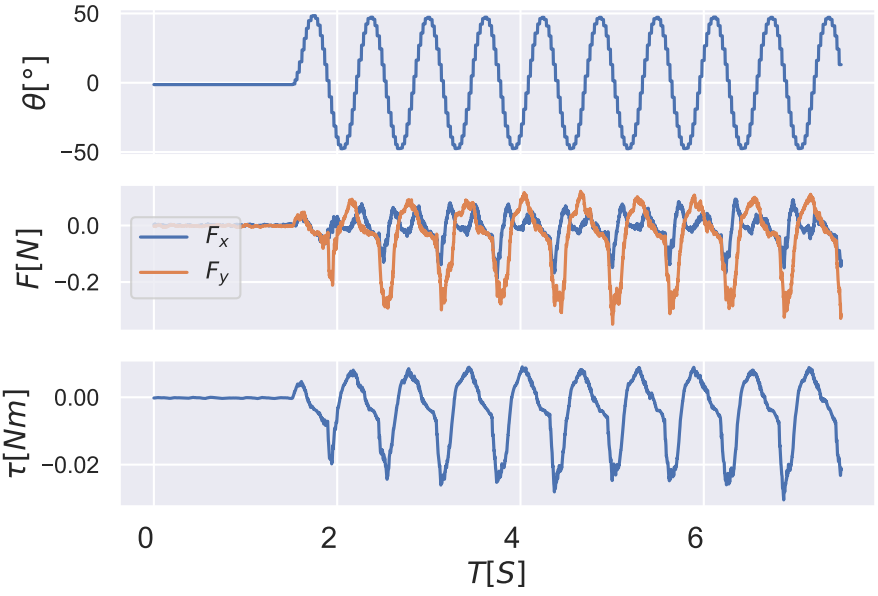


Figure 4.3: Measured forces from our leg design, it can be seen that there is an asymmetry in the lateral force.

The leg is attached to a rod connected to a 6-axis force-torque sensor [1] and is

[1] ATI mini45

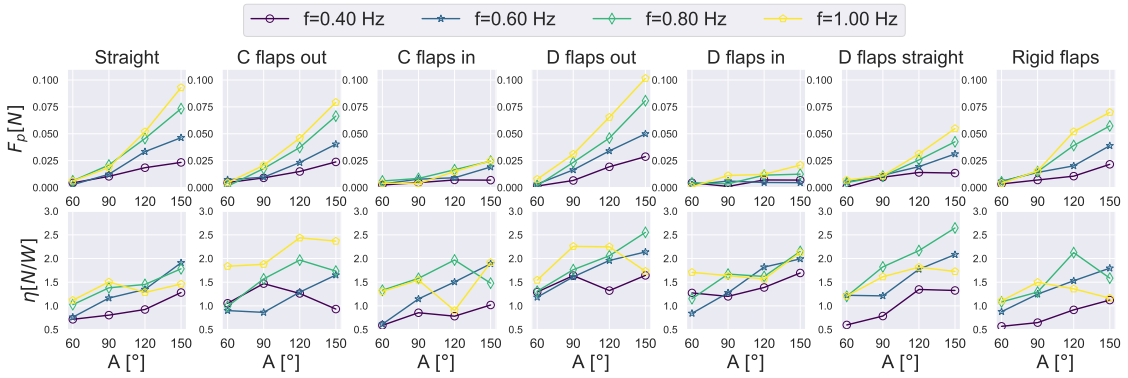


Figure 4.4: The measured propulsive force and efficiency of all leg configurations.

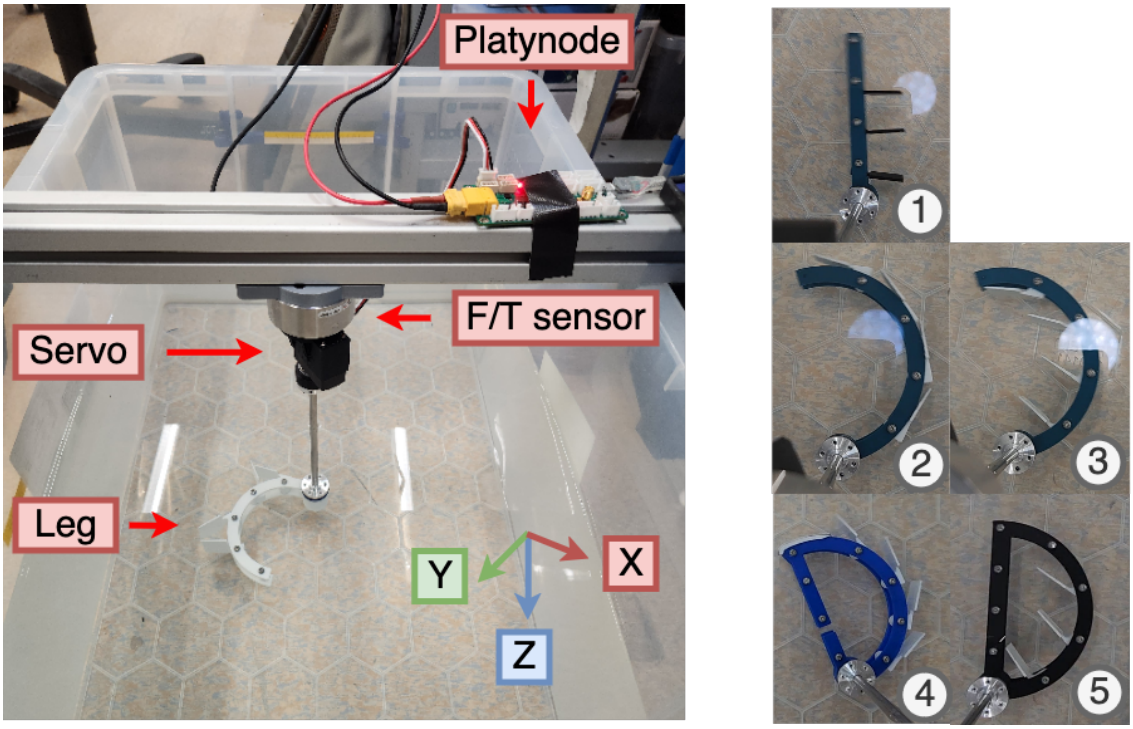


Figure 4.5: Underwater force and torque testbed used to study and characterize the flapped paddle leg design (left). Different flapped paddle leg designs, from top to bottom: straight leg, C-leg, D-leg (right)

actuated by the leg servo described in Sec. 3.

Taking inspiration from the RHex leg design [37], we developed three leg prototypes: a simple straight leg, a curved leg (termed the “C-leg”) and a hybrid design combining the previous two, which we call the “D-leg”. These designs are illustrated in Fig. 4.5. The addition of the straight segment to the C-leg allows for additional variable stiffness. Moreover, the D-leg’s straight segment provides an additional option to put flaps. From these three basic leg designs, we derived six configurations: ① Straight leg, which excels in swimming but is less effective fro walking; ② C-leg with flaps mounted on the outside of the arc; ③ a C-leg with flaps inside of the arc; D-leg with flaps ④ inside of the arc, ⑤ outside and ⑥

flaps only on the straight segment.

The flaps used in these designs are modeled as 22x30 mm rectangles made from 3D-printed thermoplastic polyurethane (TPU). All leg configurations were fitted with these TPU-based flaps. We also explored the use of fully rigid flaps with the straight leg (⑦).

The legs were created using a parametric model, the model is based on a disk where you vary set the chord (length of the straight part), radius, and arc length.

Using the oscillatory gait with reference

$$\theta_r(t) = \frac{A}{2} sin(2\pi f t) \tag{4.1}$$

we measured the generated propulsive force (F_p) and consumed power (P). The ratio of the propulsive force F_p over the consumed power is used as a metric for the efficiency of the leg movement, which we define as

$$\eta = \frac{1}{nT} \int_0^{nT} \frac{F_p(t)}{P(t)} dt \tag{4.2}$$

Fig. 4.4 shows the performance of all leg configurations across different gait parameters. It can be seen that the straight leg and D-leg generate the highest thrust, followed by the C-leg with outer-mounted flaps. Leg designs with flaps on the outside generally outperform those with the flaps on the inside. Designs with flaps on the outer side generally outperform those with flaps on the inner side, as the outer flaps have a greater range of motion. This range allows the flaps to naturally adjust to optimal angles for drag during both the power and recovery strokes.

Based on these experiment results, the D-leg with outer-mounted flaps is chosen as the final design, due to its superior performance in underwater locomotion while still maintaining good walking capability.

4.2. Walking characterization and optimization

The equations of motion governing Platypus on land are similar to those for RHex-styled hexapods. For locomotion on flat ground, alternate legs numbered 1, 4, 5 (part of set A) and legs numbered 2, 3, 6 (part of set B) as indicated in Fig 4.1, are respectively synchronized, and alternatively actuated such that when set A is in contact with the ground (i.e. *contact gait*) while set B has lifted off (i.e. *recovery gait*). The gait during one time-period is designed as follows. The contact gait is designed as

$$\theta_r(t) = -\alpha + \frac{4\alpha t}{T}, \ \forall t \in [0, T/2), \tag{4.3}$$

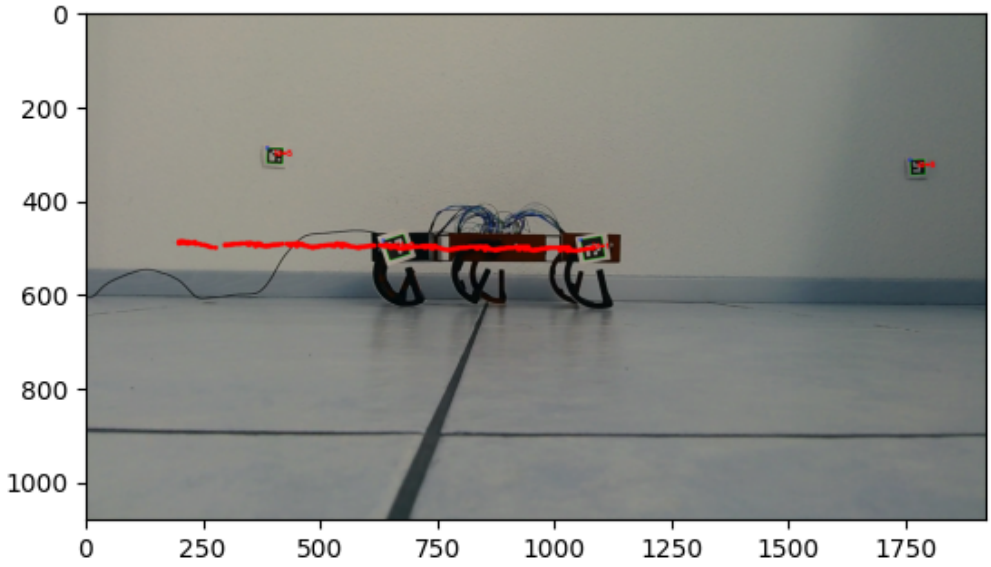


Figure 4.6: Walking gait parameter optimization by characterizing the speed of speed of Platypus measured using reference AR markers.

and the recovery gait as

$$\theta_r(t) = \alpha + \frac{4(\pi - \alpha)(t - T/2)}{T}, \ \forall t \in [T/2, T), \tag{4.4}$$

where α is the stride angle and T is the time period of the entire gait. The idea behind this simple gait is that when one set of legs completes its contact stride and is about to lift off, the other set comes in contact with the ground. This ensures the robot is supported by at least 3 legs at any given time, within a contact angle of α. We need to set the value of parameters α and T. We do so by evaluating their effect on how fast Platypus covers a straight line distance and the energy consumed to do so.

We measure the time it takes for Platypus to walk a set distance at different frequencies (inverse of T) and α. Stride angles of 30 to 130 degrees spaced at intervals of 10 degrees and frequencies from 0.1 to 0.31 Hz are used in the study. We use AR markers to visually detect the position of Platypus as it moves. These are used to determine the speed of the robot while the power consumption is logged by the Platynode using the servo current consumption. Fig. 4.6 shows an example of how the distance traveled is measured, the red line is the position of the marker at each frame from the captured video and Fig. 4.7 shows an example of the motor and sensor signals captured from one of many trials of the study.

Fig. 4.8 shows the results of this characterization. We find that higher stride angles and higher frequency increase the distance walked in a given time(speed) but also increase the power consumption. Based on this, we set the values of these parameters at α = 90 degrees and f = 0.2 Hz. Which offers a good balance

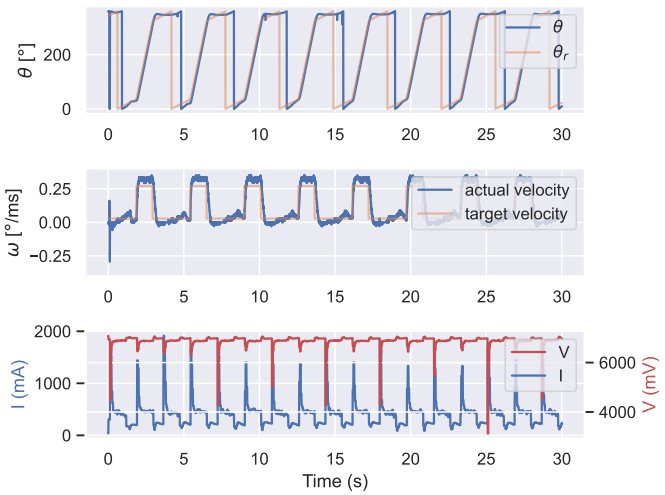


Figure 4.7: An example of sensor and motor data from one of the walking trials as part of the gait parameter optimization study of Platypus. This is from a single servo with gait $f = 0.28Hz$ and stride=$70°$

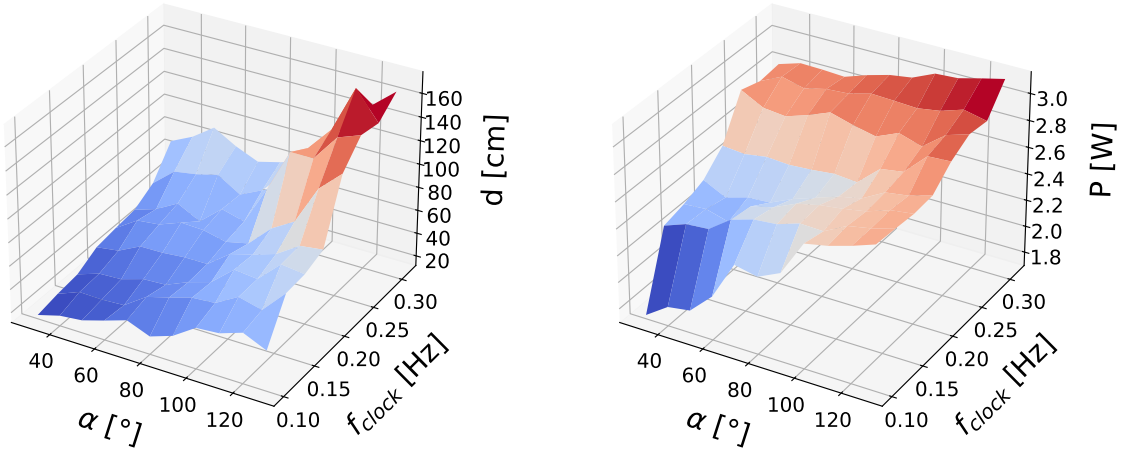


Figure 4.8: Study of walking gait parameters on the speed and power consumption for Platypus.

between speed and power consumption without making the robot unstable during walking.

4.3. Outdoor Field Trials

The benefits of having 3 segments connected with actuated joints over a single monolithic body extend to walking on land, especially when undulations are present. Fig. 4.9 shows the results of our outdoor field trials. It can be seen that the robot can easily maneuver different terrains with ease. The trials included terrain that was uneven and grassy, had fallen leaves and other natural debris and had inclines in the form of steps. Platypus used its 3 body design to handle all of these challenges and can successfully navigate such environments. A video demonstration of selected trials is also available to watch through this link [43].

Figure 4.9: Outdoor trials conducted with Platypus for walking on different terrains [43].

5

Seaweed Monitoring and Aquaculture

Seaweed holds great promise as an alternative food source and an agricultural product with wide utility [28, 17, 54]. Seaweed farming involves cultivating and harvesting seaweed grown in shallow and deep waters. Different types of seaweeds are grown in different parts of the world with a prominent farming practice in Asia with increasing production in Europe [7, 27]. One of the most cultivated seaweed varieties, Saccharina, is now also grown in the Netherlands.

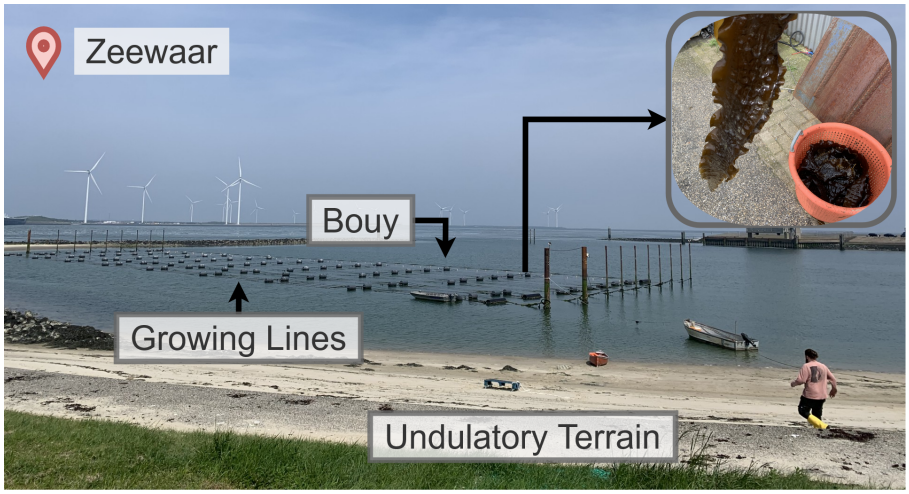


Figure 5.1: Field trials were conducted at the Zeewaar seaweed farm, which grows ribbon-like green and brown-colored seaweed called Saccharina.

We focus on this crop as it is cultivated in a 3-5 m deep bay in the province of Zeeland as part of the first seaweed farm in The Netherlands. The seaweed farm, Zeewaar, is located in the Oosterschelde National Park and is operated by The Seaweed Company. The saline water here is affected by minor currents but lacks major sea currents. The seaweed here is harvested for various reasons, including as food, feedstock, and industrial applications. The Saccharina plants grow as

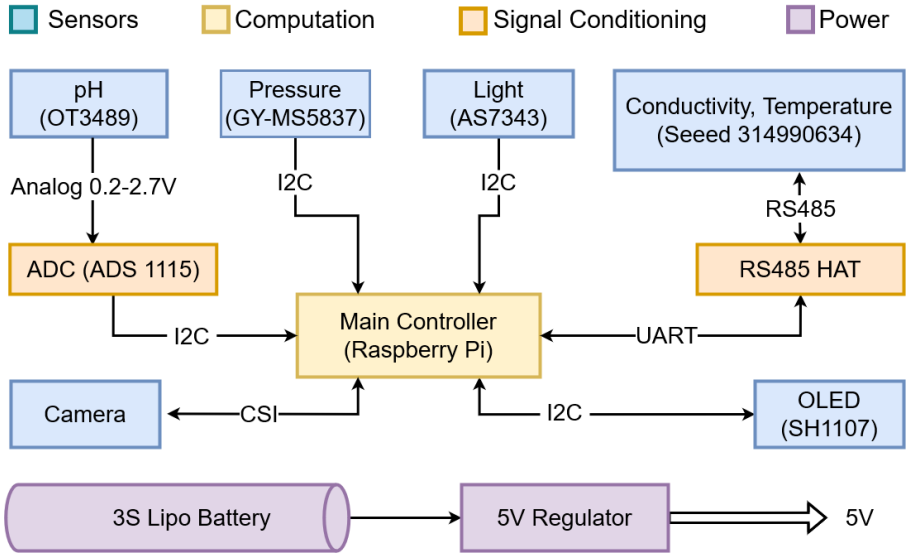


Figure 5.2: System components of Platypus's sensor box with different sensors used for data collection.

elongated ribbon-like green and brown-colored blades that reach a length of several meters. Fig. 5.1 shows the seaweed farm as well as a harvested seaweed. The seaweed here is grown on rope lines between buoys. Multiple such lines are made with a separation of 2 m between two consecutive lines. The weight of the plants causes the rope lines to buckle and curve underwater to a depth of one to two meters. The growing season for the seaweed is typically between December and May. We visited the seaweed farm twice, two weeks apart in May, to collect data and conduct our sensor system field trials.

The seaweed grows vertically downwards from just a few centimeters to up to 3 m. The seaweed grows as densely packed quasi-continuous bundles along the seeded line. The seaweed growth is primarily limited to a 2-D plane formed by the suspended rope lines. The current inspection and monitoring of crops is done manually by an employee and caretaker of the farm. This dangerous, expensive, and time-consuming task requires a boat and tremendous human intervention. The seaweed farmer needs to row each seaweed line from the shore wearing an oil skin and lift each line to inspect it visually. This process is repeated weekly from planting up to harvesting. Thus, There is great scope for Platypus to aid in remote and (semi)automated monitoring that can significantly reduce labor costs and human effort.

5.1. Sensor Box Design

What all sensors do we use and what they can sense and how that relates to seaweed aquaculture.

The sensor box records five different physical parameters underwater to record and monitor seaweed growth, as shown in Fig. 5.2. These include a pH sensor, pressure sensor, multi-spectral ambient light sensor, water conductivity sensor,

Figure 5.3: Sensor box prototype used for seaweed monitoring field trials.

and temperature sensor. In addition to this, we also record underwater footage using a waterproof camera to capture high-quality images and videos of the growing seaweed. In particular, we use an OT3489 module to sense pH, giving an analog output between 0-2.7 V, and require an external ADC module to translate the signal into a digital message over the I2C bus. The GY-MS5837 pressure sensor is a highly compact barometer giving a water depth resolution of 2 mm. To sense the light intensity inside water, we use an AS7343 spectral sensor. It has 14 channels with individual channels covering approximately 380 nm to 1000 nm. It has 12 channels in the visible spectrum (VIS) to near-infrared (NIR) range, a clear and flicker channel [39]. The SEEED studio 314990634 electrical conductivity sensor measures the EC and temperature. It has a long waterproof probe that can be dipped in water and gives these values using the RS485 interface. An RS485 to UART hardware shield makes the signal acquisition possible using a Raspberry Pi 4, which acts as the system's main controller, computation, and logging unit. The pressure and light sensors are waterproofed using a 3d printed enclosure and epoxy. Care is taken to ensure their sensing element is not occluded in this process. The pH and conductivity sensor includes a waterproof probe that can be used as is. The entire system is placed in a waterproof box and adjusted in buoyancy to allow the box to sink into the water. The sensor box can be carried by Platypus as a stand-alone add-on or be integrated as part of the thorax segment. A 3S Lipo battery and a 5V regulator power the system. A small OLED screen is also placed on the transparent box lid for in-field debugging. Fig. 5.3 shows the stand-alone sensor box. All sensor measurements were logged at a rate of 1Hz.

Field trials were conducted to determine which sensors were suitable for use. Each trial involved moving to a random location between growing lines and lowering the sensor box from the water's surface to the seabed, and then pulling it back up at a constant speed. The sensor box remained powered on throughout

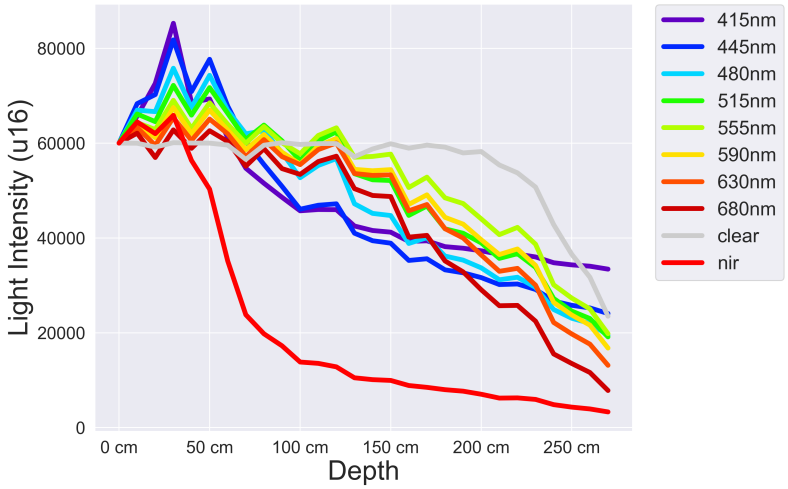


Figure 5.4: Varying ambient light with depth for different spectral frequencies at the seaweed farm.

Figure 5.5: Visual seaweed monitoring pipeline.

the entire process. These trials were carried out at 21 locations across the farm. Excluding the data collected when the sensor box was above water, 1500 samples were collected, with an average measured depth of 1.5m, and a maximum depth of 2.72m. All trials were conducted on a single day, over approximately 45 minutes.

The results showed no significant variation in conductivity, pH, or temperature with depth or across different farm locations. This suggests that in shallow water, these parameters remain consistent, allowing them to be measured reliably at a single point on the farm. Only light intensity did vary with depth. Fig. 5.4 illustrates how different light wavelengths are absorbed at different depths, with longer wavelengths (such as red light) attenuating faster than blue and green wavelengths, due to the higher absorption of longer wavelengths by water.

5.2. Underwater Image Processing

One of the key methods of monitoring seaweed growth underwater is using visual data. One can track their progress with time by capturing the size of the different seaweed plants. This can be done either with a human operator in the loop or autonomously by the robot. The seaweed plant size and biomass estimation from captured visual data is described in Sec. 5.3.

In any visual plant estimation or monitoring method, a computer vision data acquisition and processing pipeline must be set up so that we can capture high-quality

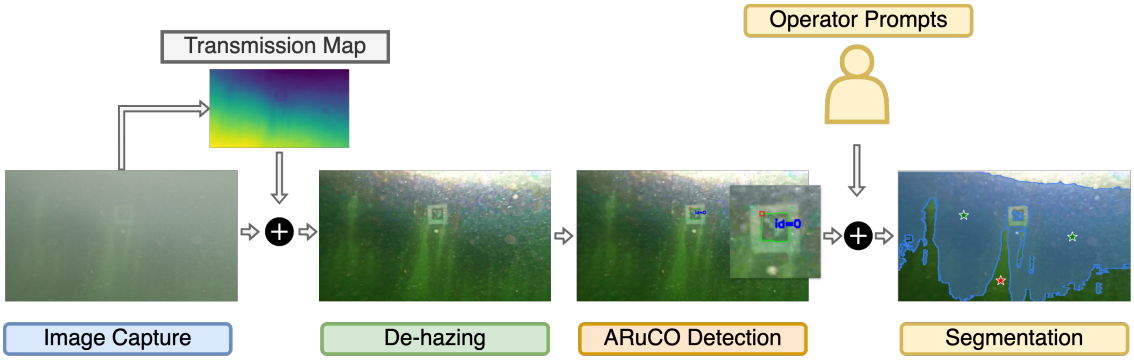


Figure 5.6: Running a captured seaweed image through our pipeline.

images. The captured images are used to detect the biomass. Since the seaweed grows vertically downwards as wide 2D planar sheets, Platypus can monitor them from the side, perpendicular to their growth direction. This is possible by swimming between the rope lines where seaweed grows and capturing the image by pointing the camera directly at the seaweed. We use ARuCo markers attached to each seaweed plant to identify each plant and estimate depth. The information from the marker detection is used along with the seaweed photos to first segment the part of the image with the seaweed and then estimate the seaweed biomass.

However, there is an issue with the captured images. As seaweed grows in shallow open water, water turbidity can be high. This results in hazy images caused by light scattering off floating particles in the water column. This effect can be countered by decreasing the distance between the camera and the seaweed plant. However, taking close-up images of the seaweed is not valid for visual monitoring as we require the entire plant to be visible and in the frame to determine its size. We, thus, need to de-haze the captured images. This pre-processing step is crucial for seaweed detection and ARuCo marker detection. The entire pipeline is shown in Fig. 5.5. The pipeline applied to an example from our database of images is shown in Fig. 5.6.

Many dehazing methods rely on a simplified physics model of how haze works proposed in [12], which is sometimes called the Duntley model

$$I(x) = J(x)t(x) + A(1 - t(x)) \tag{5.1}$$

where I is the observed pixel intensity,J is the haze-free scene radiance, A is the global atmospheric light, and t is the portion of light that is transmitted by the medium without scattering it. The goal of haze removal is to recover J, A, and t from I.

We try three methods of dehazing based on this model: Dark Channel Prior [25], Bright Channel Prior [18], and Fast Visibility Restoration [53]. We use these methods to restore the non-hazy image and apply marker detection on that image. We apply these processing steps to a database of 7650 frames of a marker taken

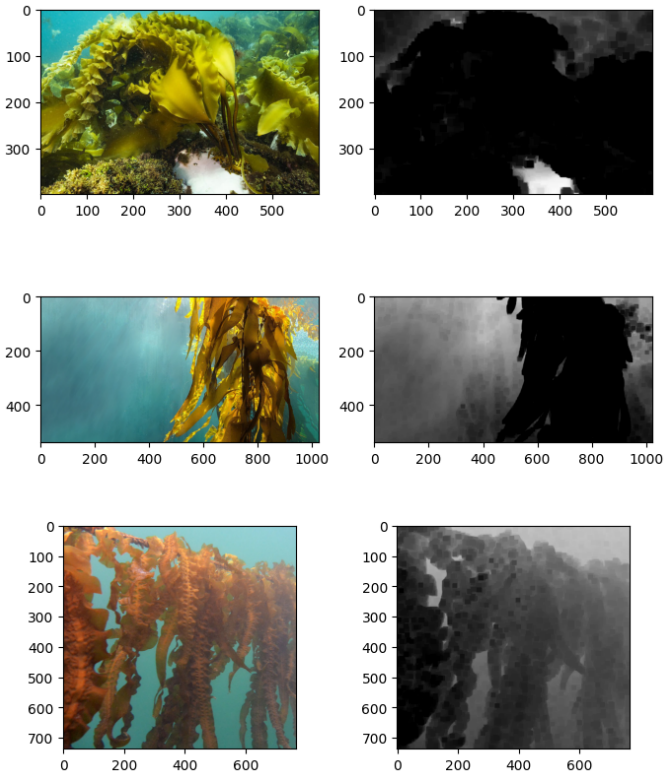


Figure 5.7: Examples of shallow seaweed images and their corresponding dark channels.

Method	# Frames w. detected markers
No pre-processing	249
Dark Channel Prior	833
Bright Channel Prior	12
Fast Visibility	3

Table 5.1: Comparison of Dark Channel Prior, Bright Channel Prior and Fast Visibility Restoration on seaweed images with a marker collected at Zeeland.

from 5 videos we collected as part of our field visit to the seaweed farm. We evaluate the effect of the dehazing method by calculating the successful detection of ARuCo markers. The results from the analysis can be seen in Table. 5.1. Out of the three methods, DCP works best. With DCP, the marker is detected in 2x the number of frames with markers versus if no DCP is applied.

DCP works by assuming a statistical prior, in [25] they introduce the notion of a *dark channel* J^{dark}, which is given by

$$J^{dark}(x) = \min_{y \in \Omega(x)} (\min_{c \in \{r,g,b\}} J^c(y)) \tag{5.2}$$

where $J^c(x)$ is a color channel of $J(x)$ and $\Omega(x)$ is a local patch centered at x. In [25] they observe that for non-hazy images, the *dark channel* is low and usually near zero. This property can then be used to derive the transmission map $t(x)$ and atmospheric light A, allowing the non-hazy image J to be extracted.

Implementation	#Frames	Runtime (s)	Speedup	Precision
Base Implementation	47	0.4221	100.00%	0.94
RGB DownUpSample	44	0.1605	262.95%	0.88
GreenBlue DownUpSample	44	0.1481	284.94%	0.88
GreenBlue	47	0.3657	115.44%	0.94
Green	45	0.3349	126.05%	0.90
Green DownUpSample	43	0.1397	302.18%	0.86
Transmission Gaussian	46	0.3137	134.57%	0.92

Table 5.2: Different DCP optimizations and their benchmarking results when run on a Raspberry Pi 4. Per implementation we measure the number of frames with detected markers and the average runtime per frame, from this we calculate the relative speedup compared to the base implementation.

While DCP is a known method for dehazing images above water, its application to underwater images is limited. When the weather is foggy, particles in the air are larger than the wavelengths of visible light. This causes light to scatter equally across all colors, making the image appear white or gray. In water, particles are also larger than visible light wavelengths, but absorption is stronger than scattering. This means that different light colors are absorbed differently in water, increasing absorption for longer wavelengths. As a result, underwater images often look blue or green, with very low intensity in the red color channel. Because of this, the dark channel in underwater images doesn't change much with distance or transmission, unlike in regular images. i.e. $J^{dark} \rightarrow 0$ even for hazy images. This makes the "dark channel prior" technique ineffective for improving hazy underwater images since there will always be one color channel (usually red) with low intensity, whether the image is hazy or not [18]. However, we find that the Dark Channel Prior works well for our data. Fig. 5.7 shows images of seaweed in shallow water along with their estimated dark channel as the dark channel is near zero for the clear parts of the seaweed. It shows that this assumption holds for shallow water, but for images in deep water, this assumption may not hold.

One major limitation of using the DCP is that it is not computationally efficient and requires long processing times. For example, on a Raspberry Pi 3, the algorithm takes an average of 3.941 seconds, while on a Raspberry Pi 4 with NEON optimizations, it still requires an average of 0.422 seconds. This limits the robot's applicability as the ARuCo markers (if not the seaweed) must be detected in real-time. We, thus, need to apply certain optimizations to the DCP algorithm. Note that the base implementation of DCP already has some proposed optimizations, like guided filter on the transmission map over soft-matting. All of the optimizations we try are applied over the base DCP implementation.

The optimizations we explored include the following: refining the transmission map using a Gaussian or mean filter, using the rough transmission map and skipping the refinement step, and calculating the transmission map from a downsampled image. Since ArUco markers are black and white, another approach we considered was calculating the dark channel on only one or two color channels, reasoning that the marker's color channels would be approximately the same. We tested using just the green channel or the green and blue channels for estimating

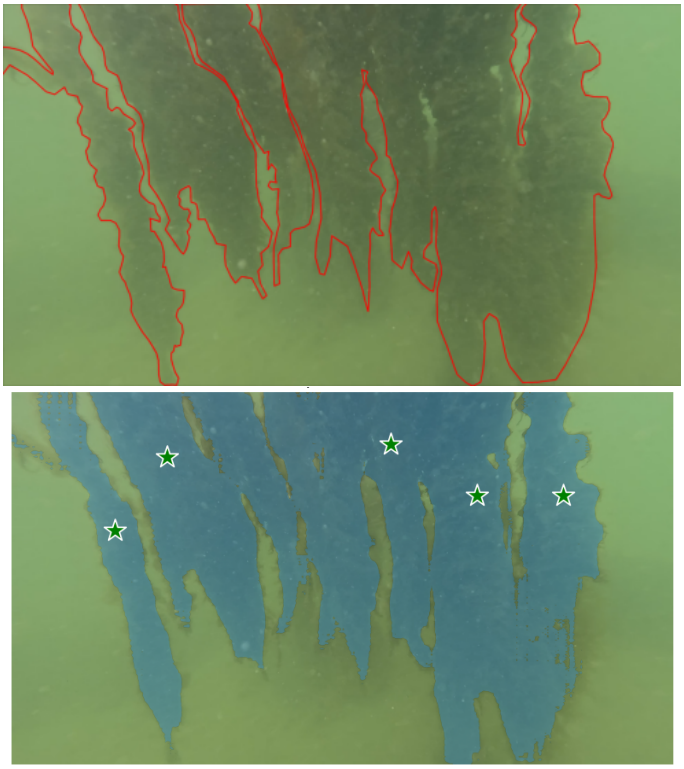


Figure 5.8: Example segmentation performance when using the SAM2 model on a sample from our database of seaweed images. Top shows the manually labeled image with red lines. Bottom shows the input prompts to the model and the resulting segmentation map.

the transmission map.

We benchmark the performance of the different optimizations on a Raspberry Pi model 4. We accomplished this by randomly selecting 50 images containing a visible marker and running the algorithm on each frame. We then counted the number of frames where a marker was successfully detected and calculated the average time taken to process an image using the entire algorithm. The results for these are shown in Table. 5.2. The implementation that was the best performing was when downsampling and upsampling the transmission map and combining that with estimating the transmission map from a single color channel. This results in $0.14/0.422 \times 100\% = 33\%$ of the run time of the base implementation. On one of the videos from our dataset with 250 marker images, the base implementation detected 232 frames, while our faster-optimized version could detect 223 frames.

5.3. Seaweed Biomass Estimation

Following the dehazing and ARuCo marker detection, the next step in the processing pipeline is to segment the image to identify the seaweed region. To this end, we perform the semantic segmentation of seaweed from the RGB images. The idea is that one can estimate the growth of the seaweed based on the pixel size of the segmented seaweed over time. While there is some prior work about automatic segmentation of seaweed images using neural networks [20], this ap-

proach does not work with our collected dataset.

As a result, we had to manually label the seaweed images by using a human in the loop. Since this process is very laborious and time-intensive, we only segmented 20 seaweed images. After obtaining the segmented images, tracking the growth of the seaweed plant over time becomes trivial. It is vital that we can automatically segment the images. To this end, we tried using mean-shift clustering on the transmission map. However, this approach does not work, as automatically identifying which clusters were seaweed was not straightforward. Some learning-based approach we considered were HR-Net [52] and Deeplabv3+ [5]. HR-Net was not suitable due to its high data requirements and computational demands, making it impractical for edge devices. Deeplabv3+ was not chosen as it also required high-quality labeled images to train and produced segmentation maps with insufficient boundary detail between seaweed and background, likely due to its encoder-decoder architecture, as upscaling is used to generate the final segmentation map.

Finally, we can segment seaweed images from our dataset using the latest released Meta Segment Anything Model 2 (SAM2) [44]. SAM2 is a model for image and video semantic segmentation aimed at promptable visual segmentation, i.e. based on user given prompts in form of points, bounding boxes or image masks, the network generates a segmentation map. The model is designed to work on any video or image data without any prior training on similar data. Fig. 5.8 shows the results when using the SAM2 for segmentation. Given proper prompts, the model performs well on seaweed images. This significantly speeds up the segmentation process since the operator would only need to click the seaweed segment. However, this process still requires human intervention. Nevertheless, the de-hazed images themselves are still useful for a human operator to visually inspect each plant remotely and make Platypus a valuable part of the remote monitoring process.

We also attempt automatic prompt generation, we explored using the already generated transmission $t(x)$ from the DCP step to create candidate points fed to the SAM2 model.

5.3.1. Candidate prompt generation attempts

During the Dark Channel Prior (DCP) process, one vital step is estimating the distance of each pixel to the camera, which results in the transmission map $t(x)$, this is equivalent to the idea of a **depth map**. A depth map in computer vision is an image or a data representation that contains information about the distance of surfaces in a scene from a given viewpoint.

Since seaweed is grown in open water, photos of the plants are typically free of obstructions other than other seaweed and grow lines. As a result, a typical frame consists mainly of the plants and the background. Given that the background is empty and significantly farther away, an accurate depth map would allow us to distinguish the seaweed by identifying pixels that are closer to the camera.

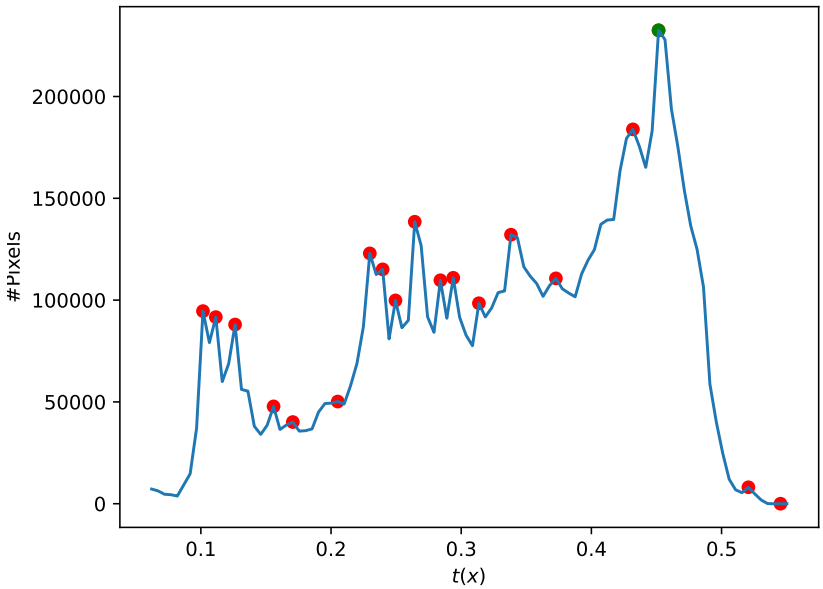


Figure 5.9: Histogram of the transmission map.

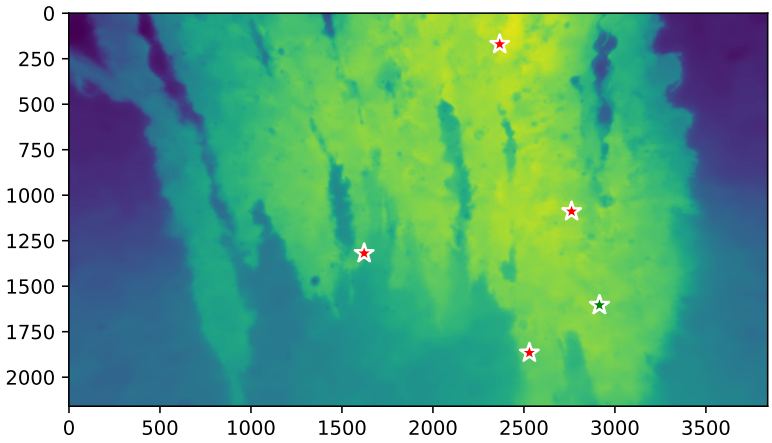


Figure 5.10: Example of a seaweed image with SAM2 candidate prompts, overlayed on transmission map.

Fig. 5.9 shows a histogram of the estimated transmission map ($t(x)$), from the histogram we can see that there is a strong peak at higher values of $t(x)$, which corresponds to “closer” pixels. We then devise the following algorithm:

Using this histogram we first find n peaks in the transmission map with the highest

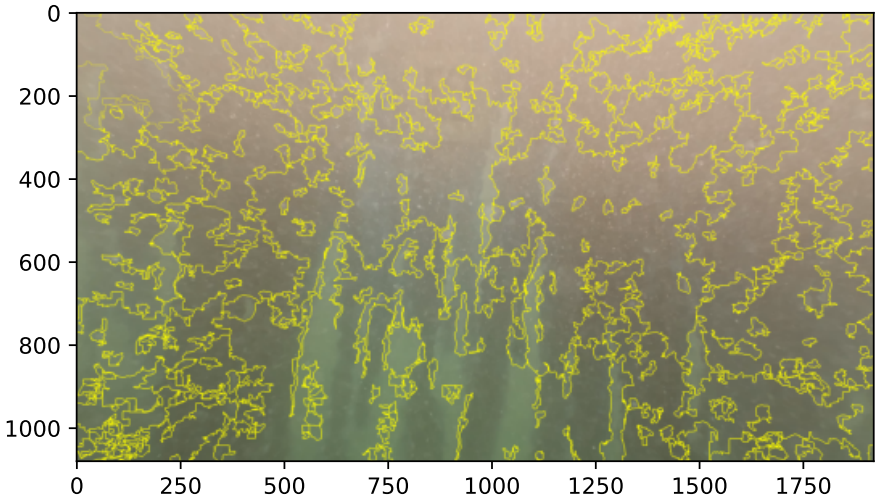


Figure 5.11: Example of seaweed image with the method from [15] applied.

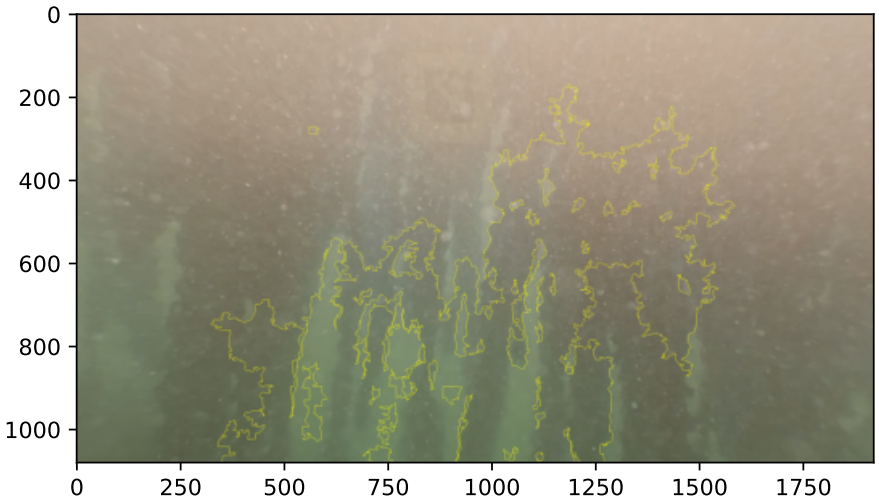


Figure 5.12: Seaweed image with three “superpixels” filtered with the largest pixel size.

values (i.e. closest to the camera). For each peak, we find the equivalent pixel coordinate of that depth value and use that as a candidate prompt. Fig. 5.10 shows an example of generated candidate prompts with a transmission map.

However, this method is found to be inadequate for most images, as the transmission map often does not accurately reflect true depth. Additionally, our findings suggest that SAM2 performs best when given a diverse set of prompts, including not just the closest seaweed but also those that are further away from the camera.

Another attempt at candidate prompt generation was using **superpixels** [15]. Felzenszwalb and Huttenlocher’s method utilizes a graph-based technique where an image is represented as a graph with pixels as nodes and edges weighted by the similarity between connected pixels. The algorithm iteratively merges nodes based on a measure of internal and external differences, creating segments that

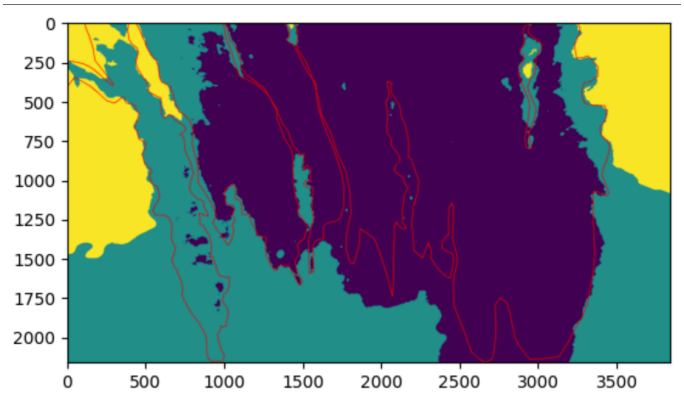


Figure 5.13: Seaweed image with mean-shift clustering applied to the transmission map. A bandwidth of 0.044 is used, which is estimated using `scikit-learn`'s `estimate_bandwidth` function. The red lines illustrate the ground truth outline of the seaweed.

preserve image boundaries and perceptual coherence. This process results in compact, contiguous regions that capture the essential structure of the image, effectively functioning as superpixels. Groups of pixels that represent regions of the image.

The idea was that image segments of seaweed often have similar texture and color and that these can be grouped. Fig. 5.11 shows a seaweed image after the superpixel algorithm is applied. Fig. 5.12 shows the same image with only the largest segments shows. In theory, any points inside these segments would then be fed into SAM2 as prompts, as they would only contain seaweed parts. However, this method does not work. As the largest superpixels do not always correspond to only seaweed segments. We find that often, some background image segments are larger than all seaweed segments, resulting in large background superpixels being generated and chosen as candidate prompts.

5.4. Non-learning approaches

Other than learning-based approaches and approaches built on top of SAM2, we explored non-learning clustering algorithms. Similar to [40], we tried using mean-shift clustering on the RGB image, and also on the transmission map $t(x)$. In [40] they capture images using an ROV of images of seaweed in an open-water farm, which is further away from the coast. Unlike in our dataset, the images they used were during clear conditions with very little turbidity, in which there is a clear separation between the background and plants. Since the farms we are interested in are near coastal areas, the images are often not clear.

We find that simple clustering on the image does not separate the background. Moreover, an added challenge would be to classify the generated clusters. In Fig. 5.13 we see an example seaweed image with mean-shift clustering applied.

Thus we find that SAM2 with manual operator prompts performs the best for biomass estimation.

6

Discussions

Applicability to other seaweed farms. Based on our discussions with the seaweed farm operators, we find that having an amphibian robotic platform that can facilitate remote seaweed monitoring is highly beneficial to all seaweed farms. It can aid in conducting visual inspection without a human being physically being present at the farm and this holds great value. The human being involved in the process is not necessarily a deal breaker. Having the robot collect valuable sensor and image data along with a method of uniquely identifying different plants gives the farmer the ability to monitor the growth of the seaweed. This data can be used to further optimize the growth and monitor if any intervention is required. Different farms can have different water conditions. For instance, farms in Asia are often shallower than the one in The Netherlands. This means that it is possible to encounter regions where the water is too shallow for an underwater vehicle or robot. In such regions having an amphibious robot like Platypus is much more beneficial.

Other applications. The Platypus platform is presented as being designed for a singular application of seaweed aquaculture. However, the applicability of the robot extends beyond just this use case. As an amphibian robot that can transition between land and water, as well as successfully maneuver challenging terrain, it can find employability for any port or harbor monitoring scenario. The ability to jump into the water from the shore without requiring any external intervention is a great benefit. Further, the platform can be fitted with different sensors like thermal cameras or sonars to map or monitor the underwater environment. This can be useful for ship or hull monitoring, marine life monitoring, and even for search and rescue missions. It's also relatively easy to add additional payload without compromising on performance. The design of the platform uses relatively simple components, making it accessible.

Limitations. The robot is not easily waterproof. Despite the design, there is a need for period maintenance of the different components to ensure waterproofness is maintained. This can be attributed to the silicone grease that is applied which can dissipate over time. Further, seawater is particularly harsh, which

makes the construction of the robot extremely challenging. Long exposure to this sea water can have catastrophic impacts on the working of the robot. Additionally, the monitoring of seaweed is still not completely autonomous and does require some human intervention. Future work could be to, for example, use SAM2 as a bootstrapping tool to train something akin to a Region Proposal Network to generate prompts, which eliminates the need for an operator. However, with this platform the process does move towards a higher degree of autonomy. Despite some limitations, we believe Platypus holds great potential for the seaweed farming community.

7

Conclusion

We developed Platypus, a platform for aquaculture monitoring by reducing the effort required through human intervention. By integrating the adaptability and versatility of a bio-inspired design, our amphibious robot offers a robust solution capable of efficient navigation in both terrestrial and aquatic environments. The innovative design features, including a modular three-segment body, specialized amphibious legs, and advanced sensing capabilities, enable the Platypus to operate effectively under various environmental conditions and support sustainable seaweed farming practices.

The successful field trials highlight the robot's potential for real-world applications, providing a foundation for future advancements in autonomous monitoring technologies for aquaculture and other marine ecosystems.

We present a complete system design, which includes all hardware, mechanical, and software needed to build the robot. We optimize different design and control parameters to make the robot suitable for amphibian locomotion. We also develop a sensing and computer vision pipeline to enable effective seaweed monitoring benchmarking its performance to run on our robot hardware.

As the demand for sustainable practices continues to grow, Platypus platform presents a significant step forward in leveraging robotic technology to address the challenges of modern aquaculture, aligning with global sustainability goals and contributing to the health of marine environments.

References

[1] Erdem Arslan and Kadir Akça. “A design methodology for cuttlefish shaped amphibious robot”. In: *Avrupa Bilim ve Teknoloji Dergisi* (2019), pp. 214–224.

[2] Robert Baines et al. “Multi-environment robotic transitions through adaptive morphogenesis”. In: *Nature* 610.7931 (2022), pp. 283–289.

[3] Sanaz Bazaz Behbahani and Xiaobo Tan. “Design and modeling of flexible passive rowing joint for robotic fish pectoral fins”. In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1119–1132.

[4] Annalisa Berta, James L Sumich, and Kit M Kovacs. *Marine mammals: evolutionary biology*. Elsevier, 2005.

[5] Liang-Chieh Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[6] Ik Kyo Chung, Cynthia F. Sondak, and John Beardall. “Using marine macroalgae for carbon sequestration: a critical appraisal”. In: *Journal of Applied Phycology* 23.5 (2011), pp. 877–886.

[7] Cordis. europa. eu Cordis. “The benefits of seaweed cultivation”. In: *CORDIS | European Commission* (July 2021). url: `https://cordis.europa.eu/article/id/430375-the-benefits-of-seaweed-cultivation`.

[8] Kristen E Crandell and Bret W Tobalske. “Kinematics and aerodynamics of avian upstrokes during slow flight”. In: *Journal of Experimental Biology* 218.16 (2015), pp. 2518–2527.

[9] Alessandro Crespi et al. “Salamandra robotica II: an amphibious robot to study salamander-like swimming and walking gaits”. In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 308–320.

[10] Bir Bikram Dey, Sandeep Manjanna, and Gregory Dudek. “Ninja legs: Amphibious one degree of freedom robotic legs”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 5622–5628.

[11] Carlos M. Duarte et al. “Can seaweed farming play a role in climate change mitigation and adaptation?” In: *Frontiers in Marine Science* 4 (2017). doi: `10.3389/fmars.2017.00100`.

[12] Seibert Q Duntley, Almerian R Boileau, and Rudolph W Preisendorfer. “Image transmission by the troposphere I”. In: *JOSA* 47.6 (1957), pp. 499–506.

[13] DuPont™. *MOLYKOTE® 111 Compound*. Form No. 22-1087O-01 (01/24) AGP17374. DuPont™. Jan. 2024. url: https://www.dupont.com/content/dam/dupont/amer/us/en/Molykote/public/documents/en/MOLYKOTE_111_Compound_22-1087O-01.pdf.

[14] Christopher J Esposito et al. “A robotic fish caudal fin: effects of stiffness and motor program on locomotor performance”. In: *Journal of Experimental Biology* 215.1 (2012), pp. 56–67.

[15] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Efficient graph-based image segmentation”. In: *International journal of computer vision* 59 (2004), pp. 167–181.

[16] Frank E. Fish. “Limits of Nature and Advances of Technology in Marine Propulsion”. In: *Integrative and Comparative Biology* 46.6 (2006), pp. 820–829. doi: 10.1093/icb/icl036. url: https://doi.org/10.1093/icb/icl036.

[17] Food and Agriculture Organization of the United Nations. *The global status of seaweed production, trade and utilization*. https://www.fao.org/3/I8695EN/i8695en.pdf. 2018. url: https://www.fao.org/3/I8695EN/i8695en.pdf.

[18] Yakun Gao, Haibin Li, and Shuhuan Wen. “Restoration and enhancement of underwater images based on bright channel prior”. In: *Mathematical problems in engineering* 2016.1 (2016), p. 3141478.

[19] Christina Georgiades, Meyer Nahon, and Martin Buehler. “Simulation of an underwater hexapod robot”. In: *Ocean Engineering* 36.1 (2009), pp. 39–47.

[20] Jeroen Gerlo et al. “Seaweed growth monitoring with a low-cost vision-based system”. In: *Sensors* 23.22 (2023), p. 9197.

[21] Roza Gkliva, Michael Sfakiotakis, and Maarja Kruusmaa. “Development and experimental assessment of a flexible robot fin”. In: *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE. 2018, pp. 208–213.

[22] Nicole M Graf, Alexander M Behr, and Kathryn A Daltorio. “Crab-like hexapod feet for amphibious walking in sand and waves”. In: *Biomimetic and Biohybrid Systems: 8th International Conference, Living Machines 2019, Nara, Japan, July 9–12, 2019, Proceedings 8*. Springer. 2019, pp. 158–170.

[23] Helen Greiner et al. “Autonomous legged underwater vehicles for near land warfare”. In: *Proceedings of Symposium on Autonomous Underwater Vehicle Technology*. Ieee. 1996, pp. 41–48.

[24] Bin Han et al. “Mechanism design and gait experiment of an amphibian robotic turtle”. In: *Advanced Robotics* 25.16 (2011), pp. 2083–2097.

[25] Kaiming He, Jian Sun, and Xiaoou Tang. “Single image haze removal using dark channel prior”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.12 (2010), pp. 2341–2353.

[26] Saad Bin Abul Kashem et al. “Design and Implementation of a Quadruped Amphibious Robot Using Duck Feet”. In: *Robotics* 8.3 (2019), p. 77.

[27] Jang K Kim et al. “Seaweed aquaculture: cultivation technologies, challenges and its ecosystem services”. In: *Algae* 32.1 (2017), pp. 1–13.

[28] Jl Koch et al. *The role of seaweed in the future food system: The potential of Dutch parties in this young sector*. Tech. rep. Wageningen Economic Research, 2021.

[29] Audun Korneliussen. *nRF5-ble-timesync-demo*. [Online; accessed 1. Aug. 2024]. Aug. 2024. url: https://github.com/nordic-auko/nRF5-ble-timesync-demo/tree/master.

[30] Bokeon Kwak and Joonbum Bae. “Design of hair-like appendages and comparative analysis on their coordination toward steady and efficient swimming”. In: *Bioinspiration & Biomimetics* 12.3 (2017), p. 036014.

[31] Serge Larivière. “Lontra longicaudis”. In: *Mammalian species* 609 (1999), pp. 1–5.

[32] George V Lauder. “Function of the caudal fin during locomotion in fishes: kinematics, flow visualization, and evolutionary patterns”. In: *American Zoologist* 40.1 (2000), pp. 101–122.

[33] Qingde Li and John G Griffiths. “Least squares ellipsoid specific fitting”. In: *Geometric modeling and processing, 2004. proceedings*. IEEE. 2004, pp. 335–340.

[34] Stephen Carl Licht et al. *Towards amphibious robots: Asymmetric flapping foil motion underwater produces large thrust efficiently*. Tech. rep. Massachusetts Institute of Technology. Sea Grant College Program, 2009.

[35] Pål Liljebäck et al. “Mamba-A waterproof snake robot with tactile sensing”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 294–301.

[36] Christian Meurer et al. “Nonlinear Orientation Controller for a Compliant Robotic Fish Based on Asymmetric Actuation”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 4688–4694.

[37] Edward Z Moore. “Leg design and stair climbing control for the rhex robotic hexapod”. In: (2002).

[38] Nordic Semiconductor. *nRF Connect Technical Documentation*. Accessed: 2024-08-23. 2024. url: https://infocenter.nordicsemi.com/topic/nrf_connect_sdk/.

[39] ams OSRAM Group. *14-Channel Multi-Spectral Sensor v4-00*. Datasheet DS001046. ams OSRAM. Dec. 2022. url: https://ams.com/en/as7343.

[40] Martin Molberg Overrein et al. “Biomass estimations of cultivated kelp using underwater RGB images from a mini-ROV and computer vision approaches”. In: *Frontiers in Marine Science* 11 (2024), p. 1324075.

[41] Talat Ozyagcilar. *Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference*. Application Note AN4246. Freescale Semiconductor Inc. 2015. url: https://www.nxp.com/docs/en/application-note/AN4246.pdf.

[42] Caroline Peres et al. “Development of a low-power underwater nfc-enabled sensor device for seaweed monitoring”. In: *Sensors* 21.14 (2021), p. 4649.

[43] PLATYPUS. *PLATYPUS video demonstration*. Created: 2024-8-21. 2024. url: https://youtu.be/o2NdyeQ-mwo.

[44] Nikhila Ravi et al. “Sam 2: Segment anything in images and videos”. In: *arXiv preprint arXiv:2408.00714* (2024).

[45] Blue Robotics. *Watertight Enclosures (WTE) Technical Specifications*. Datasheet Rev C (Jan-2024). Blue Robotics Inc. Jan. 2024. url: https://bluerobotics.com/wp-content/uploads/2019/11/WTE-DATASHEET-RevC-JAN2024.pdf.

[46] Uluc Saranli, Martin Buehler, and Daniel E Koditschek. “Design, modeling and preliminary control of a compliant hexapod robot”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 3. 2000, pp. 2589–2596.

[47] Uluc Saranli, Martin Buehler, and Daniel E Koditschek. “RHex: A simple and highly mobile hexapod robot”. In: *The International Journal of Robotics Research* 20.7 (2001), pp. 616–631.

[48] Uluc Saranli, Martin Buehler, and Daniel E Koditschek. “RHex: A simple and highly mobile hexapod robot”. In: *The International Journal of Robotics Research* 20.7 (2001), pp. 616–631.

[49] Ashutosh Simha et al. “A Flapped Paddle-Fin for Improving Underwater Propulsive Efficiency of Oscillatory Actuation”. In: *IEEE Robotics and Automation Letters* (2020), pp. 3176–3181.

[50] Torfinn Solvang et al. “Automation concepts for industrial-scale production of seaweed”. In: *Frontiers in Marine Science* 8 (2021), p. 613093.

[51] Ivan Stenius et al. “A system for autonomous seaweed farm inspection with an underwater robot”. In: *Sensors* 22.13 (2022), p. 5064.

[52] Ke Sun et al. “Deep High-Resolution Representation Learning for Human Pose Estimation”. In: *CVPR*. 2019.

[53] Jean-Philippe Tarel and Nicolas Hautiere. “Fast visibility restoration from a single color or gray level image”. In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 2201–2208.

[54] United Nations. *Sustainable Development Goal 14: Conserve and sustainably use the oceans, seas, and marine resources*. https://sdgs.un.org/goals/goal14. 2021. url: https://sdgs.un.org/goals/goal14.

[55] Waveshare. *ST3020 Servo - Waveshare Wiki*. [Online; accessed 20. Aug. 2024]. Aug. 2024. url: https://www.waveshare.com/wiki/ST3020_Servo.

[56] Waveshare. *ST3022 Servo - Waveshare Wiki*. [Online; accessed 20. Aug. 2024]. Aug. 2024. url: https://www.waveshare.com/wiki/ST3025_Servo.

[57] Wenqing Yang and Bifeng Song. “Experimental investigation of aerodynamics of feather-covered flapping wing”. In: *Applied bionics and biomechanics* 2017 (2017).

[58] Bin Zhong et al. “On a CPG-based hexapod robot: AmphiHex-II with variable stiffness legs”. In: *IEEE/ASME Transactions on Mechatronics* 23.2 (2018), pp. 542–551.

A

Appendix

Parameter	Unit	Value	Description
leg_width	mm	25.00	width of leg
flap_count	-	4	number of flaps
radius	mm	55.00	the radius of the leg disk
alpha	deg	5.0	arc passing center line
mount_radius	mm	23.8 mm / 2	the radius of the leg mount circle
desired_len	mm	170.00	the desired length of the leg arc
flap_margin_angle	deg	25.0	the amount margin between the bolt holes from the start and end of the flap
straight_spacer_count	-	4	

Table A.1: Physical design parameters of the different evaluated legs.

	implementation	frames_w_detected_markers	avg_time_per_frame_seconds	speedup	precision
0	base_implementation	232	0.046646	100.00%	1.000000
1	smaller_window	238	0.043871	106.33%	1.025862
2	bigger_window	228	0.048700	95.78%	0.982759
3	down_up	225	0.017380	268.38%	0.969828
4	down_up_early	221	0.027450	169.93%	0.952586
5	only_green_blue_down_up	224	0.015282	305.23%	0.965517
6	only_green_blue	231	0.039863	117.02%	0.995690
7	only_green	225	0.037239	125.26%	0.969828
8	only_green_down_up	223	0.014997	311.03%	0.961207
9	only_green_down_up_atm_fast	226	0.024833	187.84%	0.974138
10	skip_transmission_refine	199	0.030136	154.79%	0.857759
11	transmission_gaussian	228	0.033422	139.57%	0.982759
12	smaller_window_with_gaussian	222	0.031504	148.07%	0.956897
13	transmission_mean	228	0.032615	143.02%	0.982759
14	skip_gray	13	0.042718	109.20%	0.056034
15	single_color_dehazing	3	0.217706	21.43%	0.012931

Figure A.1: Table showing all implementations with the number of frames with detected markers, and the speedup compared to base implementation. These were all benchmarked on a MacBook (M1, 2022).

B

PCB and Schematic

The following appendix contains various details on the design of the Platynode PCB

Battery insertion. To deal with possible flipped battery insertion, a connector type was chosen (XT-60) that can only be plugged in one direction, additionally a reverse polarity circuit was added. A P-ch MOSFET was chosen. We use a power MOSFET as it is placed in-line with the battery input. Care was also taken to select one with a low on-resistance (RDS(on)) to prevent thermal issues. Based on estimations, if all servos were to stall, you would have a max load current of 2x4.4 + 2x2.7 = 14.2A = 15A with some margin for error. With this, you can calculate the junction temperature T_j. A large copper ground plane is also added to the drain of the MOSFET to assist with heat dissipation.

RDS(on) at V_{gs}(-4.5V) = 5.5 mOhm

PD = I^2 * RDS(on) = 15 * 15 * 0.0055 = 1,2375 W

T_j = 1,2375W * 55 C/W = 68,0625C

$T_J = P_d \theta_{JA} + T_A$

Where,

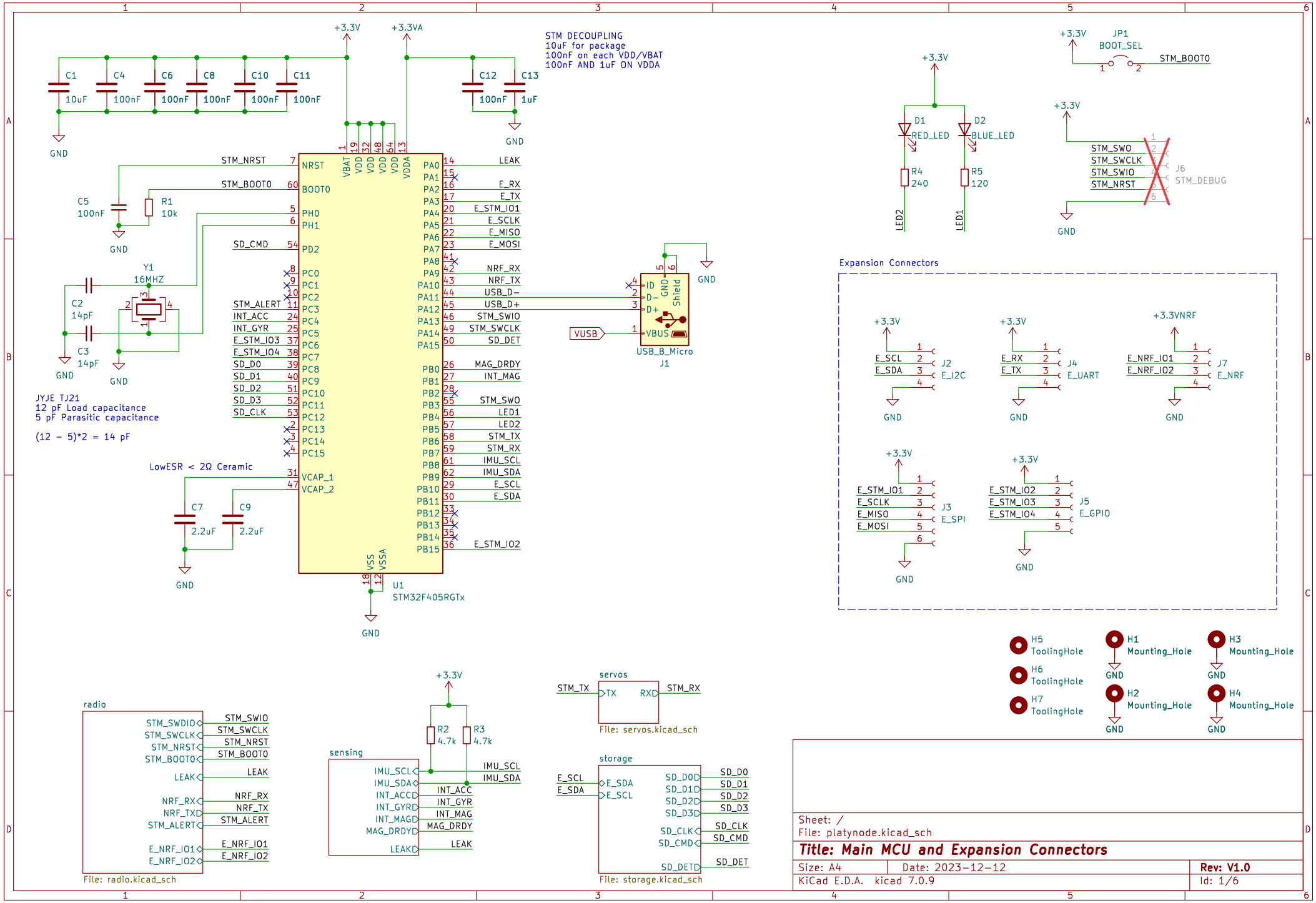
θ_{JA} = Junction-to-ambient thermal resistance

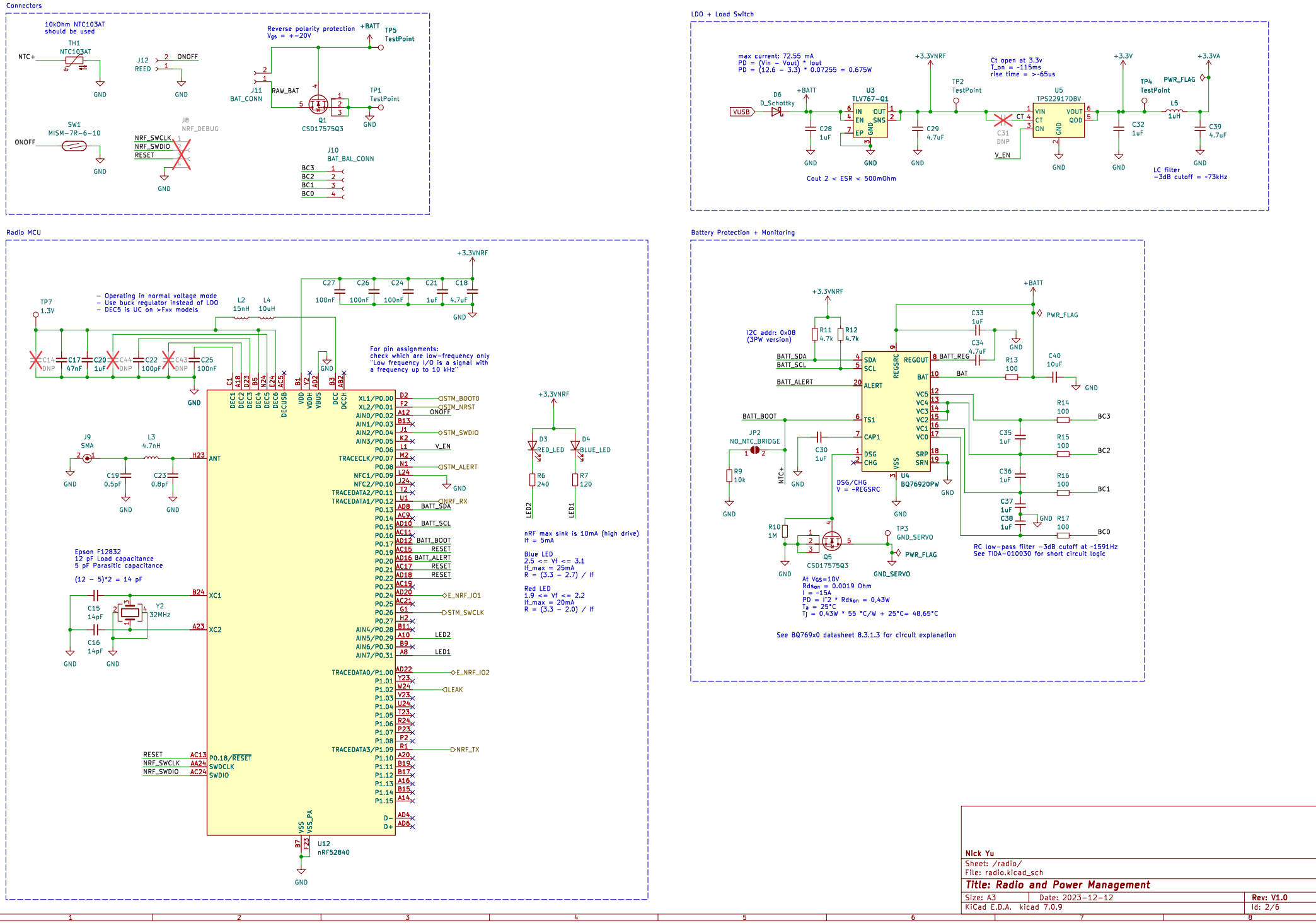
T_A = Ambient temperature

P_d = Core power + I/O switching power + ODT Power

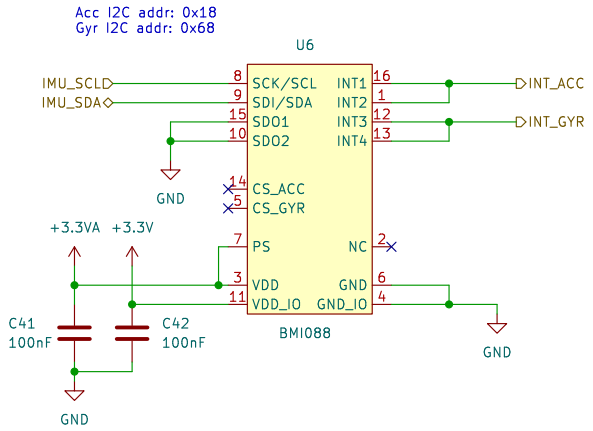
Antenna. a matching circuit is used to used for the antenna and nRF52. We use a pi-filter for this. Care is taken to select stable capacitors and inductors, e.g. only using C0G capacitors and selecting an inductor with a self-resonant-frequency (SRF) above a 4 multiple of the transmission frequency (4 x 2.5GHz = 10 GHz), the chosen inductor has an SRF of 10.80 GHz. As for the antenna transmission line, a coplanar waveguide is used with fencing vias, trace edges were also tapered. We also take care to only use a single continuous ground plane under the transmission line and remove any solder resist to not affect the

dielectric constant. From calculations, we arrived at a transmission line width of 0.3545 mm.

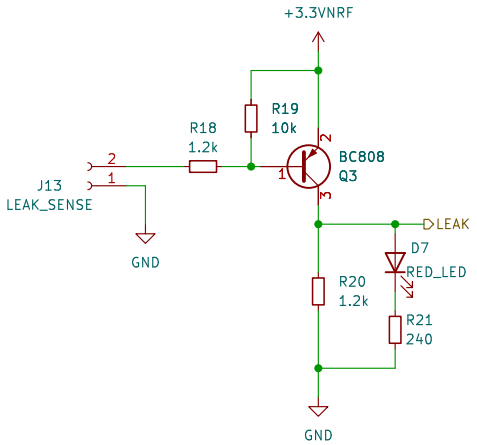

STM DECOUPLING
10uF for package
100nF on each VDD/VBAT
100nF AND 1uF ON VDDA
C1 10uF
C4 100nF
C6 100nF
C8 100nF
C10 100nF
C11 100nF
C12 100nF
C13 1uF
+3.3V
+3.3VA
GND
U1
STM32F405RGTx
NRST
BOOT0
PH0
PH1
PD2
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
PC12
PC13
PC14
PC15
VCAP_1
VCAP_2
VBAT
VDD
VDDA
VSS
VSSA
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
STM_NRST
STM_BOOT0
SD_CMD
STM_ALERT
INT_ACC
INT_GYR
E_STM_IO3
E_STM_IO4
SD_D0
SD_D1
SD_D2
SD_D3
SD_CLK
LEAK
E_RX
E_TX
E_STM_IO1
E_SCLK
E_MISO
E_MOSI
NRF_RX
NRF_TX
USB_D−
USB_D+
STM_SWIO
STM_SWCLK
SD_DET
MAG_DRDY
INT_MAG
LED1
LED2
STM_TX
IMU_SCL
IMU_SDA
E_SCL
E_SDA
E_STM_IO2
C5 100nF
R1 10k
Y1 16MHZ
C2 14pF
C3 14pF
JYJE TJ21
12 pF Load capacitance
5 pF Parasitic capacitance
(12 − 5)*2 = 14 pF
LowESR < 2Ω Ceramic
C7 2.2uF
C9 2.2uF
USB_B_Micro
J1
VUSB
ID
D−
D+
VBUS
Shield
D1 RED_LED
D2 BLUE_LED
R4 240
R5 120
LED2
LED1
JP1 BOOT_SEL
STM_BOOT0
J6 STM_DEBUG
STM_SWO
STM_SWCLK
STM_SWIO
STM_NRST
Expansion Connectors
J2 E_I2C
J4 E_UART
J7 E_NRF
J3 E_SPI
J5 E_GPIO
+3.3VNRF
H5 ToolingHole
H6 ToolingHole
H7 ToolingHole
H1 Mounting_Hole
H2 Mounting_Hole
H3 Mounting_Hole
H4 Mounting_Hole
radio
File: radio.kicad_sch
sensing
R2 4.7k
R3 4.7k
servos
File: servos.kicad_sch
storage
File: storage.kicad_sch
Sheet: /
File: platynode.kicad_sch
Title: Main MCU and Expansion Connectors
Size: A4
Date: 2023-12-12
Rev: V1.0
KiCad E.D.A. kicad 7.0.9
Id: 1/6


Connectors
10kOhm NTC103AT
should be used
TH1
NTC103AT
NTC+
GND
J12
REED
ONOFF
Reverse polarity protection
Vgs = +-20V
+BATT
TP5
TestPoint
J11
BAT_CONN
RAW_BAT
Q1
CSD17575Q3
TP1
TestPoint
SW1
MISM-7R-6-10
ONOFF
J8
NRF_DEBUG
NRF_SWCLK
NRF_SWDIO
RESET
J10
BAT_BAL_CONN
BC3
BC2
BC1
BC0
LDO + Load Switch
max current: 72.55 mA
PD = (Vin - Vout) * Iout
PD = (12.6 - 3.3) * 0.07255 = 0.675W
D6
D_Schottky
VUSB
+BATT
U3
TLV767-Q1
IN
OUT
EN
SNS
GND
EP
C28
1uF
C29
4.7uF
+3.3VNRF
TP2
TestPoint
Ct open at 3.3v
T_on = ~115ms
rise time = >~65us
U5
TPS22917DBV
VIN
CT
ON
VOUT
QOD
C31
DNP
V_EN
C32
1uF
+3.3V
TP4
TestPoint
PWR_FLAG
+3.3VA
L5
1uH
C39
4.7uF
LC filter
-3dB cutoff = ~73kHz
Cout 2 < ESR < 500mOhm
Radio MCU
+3.3VNRF
C27
100nF
C26
100nF
C24
100nF
C21
1uF
C18
4.7uF
- Operating in normal voltage mode
- Use buck regulator instead of LDO
- DEC5 is UC on >Fxx models
TP7
1.3V
L2
15nH
L4
10uH
C14
DNP
C17
47nF
C20
1uF
C44
DNP
C22
100pF
C43
DNP
C25
100nF
For pin assignments:
check which are low-frequency only
"Low frequency I/O is a signal with
a frequency up to 10 kHz"
J9
SMA
L3
4.7nH
C19
0.5pF
C23
0.8pF
ANT
Epson F12832
12 pF Load capacitance
5 pF Parasitic capacitance
(12 - 5)^2 = 14 pF
C15
14pF
C16
14pF
Y2
32MHz
XC1
XC2
XL1/P0.00
XL2/P0.01
AIN0/P0.02
AIN1/P0.03
AIN2/P0.04
AIN3/P0.05
P0.06
TRACECLK/P0.07
P0.08
NFC1/P0.09
NFC2/P0.10
TRACEDATA2/P0.11
TRACEDATA1/P0.12
P0.13
P0.14
P0.15
P0.16
P0.17
P0.18
P0.19
P0.20
P0.21
P0.22
P0.23
P0.24
P0.25
P0.26
P0.27
AIN4/P0.28
AIN5/P0.29
AIN6/P0.30
AIN7/P0.31
TRACEDATA0/P1.00
P1.01
P1.02
P1.03
P1.04
P1.05
P1.06
P1.07
P1.08
TRACEDATA3/P1.09
P1.10
P1.11
P1.12
P1.13
P1.14
P1.15
D-
D+
STM_BOOT0
STM_NRST
ONOFF
STM_SWDIO
V_EN
STM_ALERT
GND
NRF_RX
BATT_SDA
BATT_SCL
BATT_BOOT
BATT_ALERT
RESET
RESET
E_NRF_IO1
STM_SWCLK
LED2
LED1
E_NRF_IO2
LEAK
NRF_TX
RESET
NRF_SWCLK
NRF_SWDIO
P0.18/RESET
SWDCLK
SWDIO
VSS
VSS_PA
U12
nRF52840
+3.3VNRF
D3
RED_LED
D4
BLUE_LED
R6
240
R7
120
LED2
LED1
nRF max sink is 10mA (high drive)
If = 5mA
Blue LED
2.5 <= Vf <= 3.1
If_max = 25mA
R = (3.3 - 2.7) / If
Red LED
1.9 <= Vf <= 2.2
If_max = 20mA
R = (3.3 - 2.0) / If
Battery Protection + Monitoring
+3.3VNRF
+BATT
I2C addr: 0x08
(3PW version)
R11
4.7k
R12
4.7k
BATT_SDA
BATT_SCL
BATT_ALERT
BATT_BOOT
JP2
NO_NTC_BRIDGE
R9
10k
NTC+
C30
1uF
DSG/CHG
V = ~REGSRC
U4
BQ76920PW
SDA
SCL
ALERT
TS1
CAP1
DSG
CHG
REGSRC
REGOUT
BAT_REG
BAT
VC5
VC4
VC3
VC2
VC1
VCO
SRP
SRN
VSS
C33
1uF
C34
4.7uF
R13
100
C40
10uF
PWR_FLAG
R14
100
R15
100
R16
100
R17
100
C35
1uF
C36
1uF
C37
1uF
C38
1uF
BC3
BC2
BC1
BC0
RC low-pass filter -3dB cutoff at ~1591Hz
See TIDA-010030 for short circuit logic
R10
1M
Q5
CSD17575Q3
TP3
GND_SERVO
PWR_FLAG
GND_SERVO
At Vgs=10V
Rdson = 0.0019 Ohm
I = ~15A
PD = I^2 * Rdson = 0,43W
Ta = 25°C
Tj = 0,43W * 55 °C/W + 25°C= 48,65°C
See BQ769x0 datasheet 8.3.1.3 for circuit explanation
Nick Yu
Sheet: /radio/
File: radio.kicad_sch
Title: Radio and Power Management
Size: A3
Date: 2023-12-12
Rev: V1.0
KiCad E.D.A. kicad 7.0.9
Id: 2/6

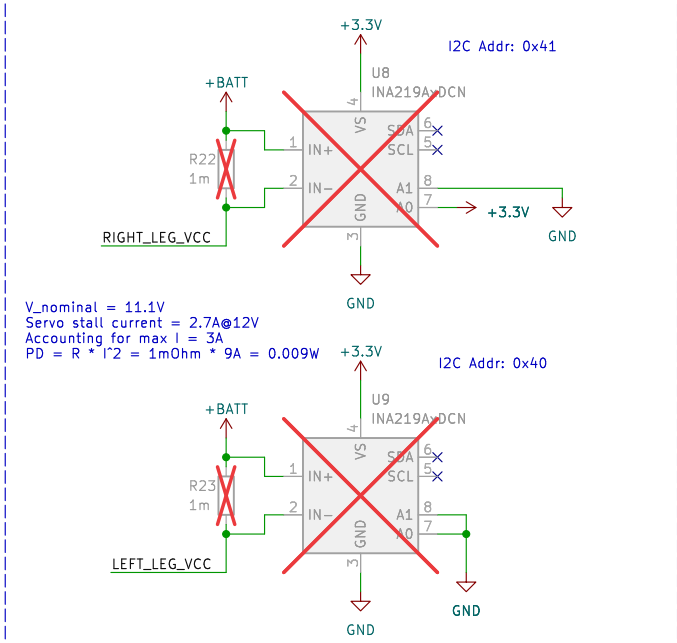
IMU



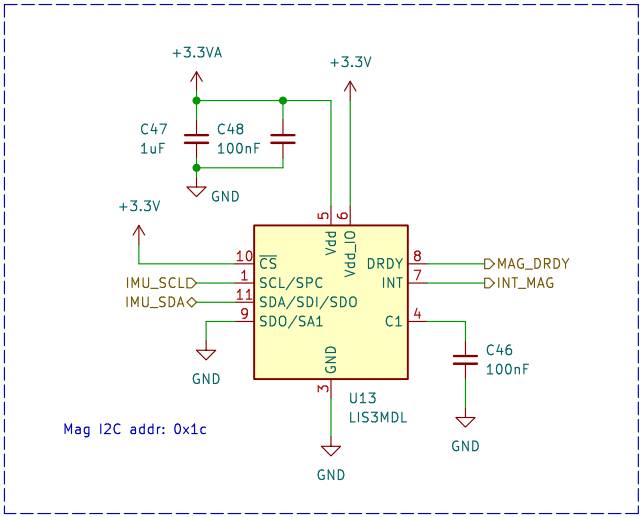
Water Leak



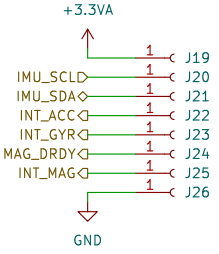
Current Sense



Magnetometer

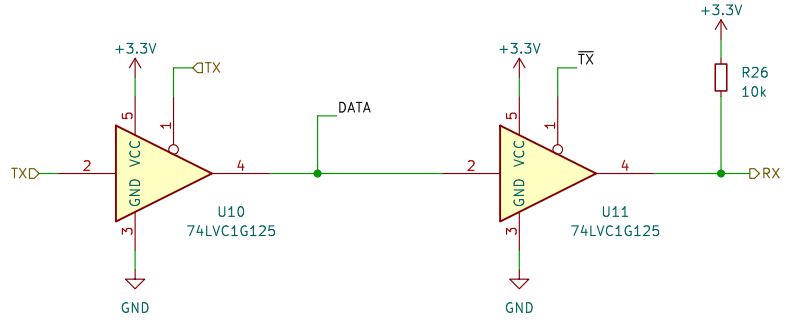


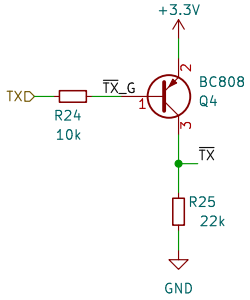
(opt) Daughter Board connector



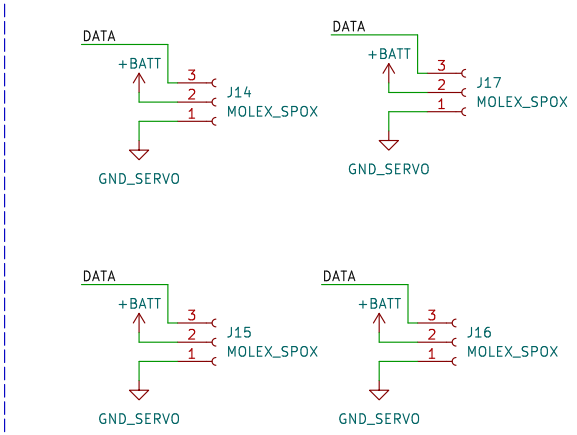
Sheet: /sensing/		
File: sensing.kicad_sch		
Title: Sensing		
Size: A4	Date: 2023-12-12	**Rev: V1.0**
KiCad E.D.A. kicad 7.0.9		Id: 3/6

Servo communication





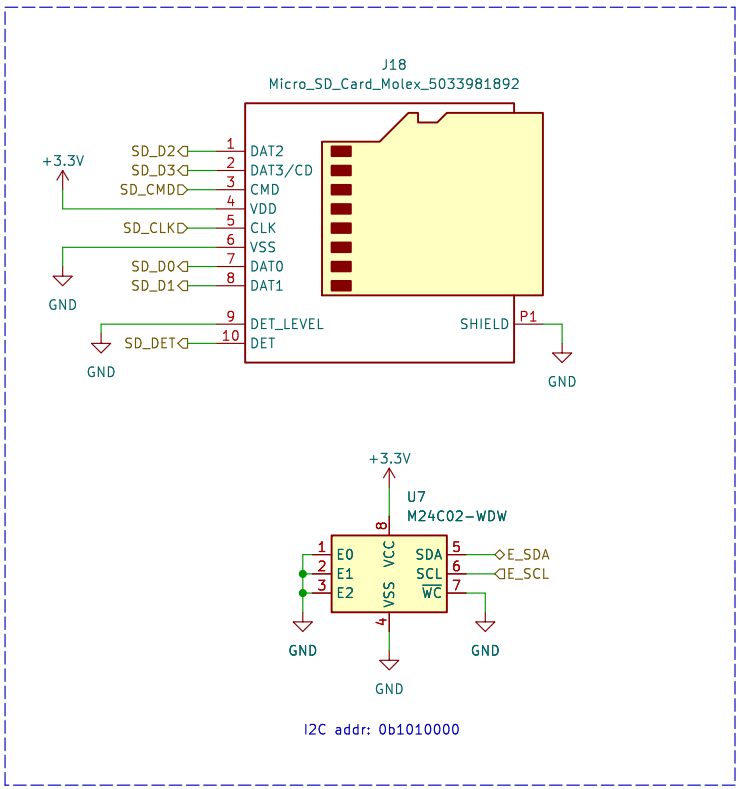
Servo connectors



Servos can be daisy-chained
Only add additional connectors if power consumption per connector is too much

Sheet: /servos/		
File: servos.kicad_sch		
Title: Servo Communication and Connectors		
Size: A4	Date: 2023-12-12	**Rev: V1.0**
KiCad E.D.A. kicad 7.0.9		Id: 4/6

Non-volatile storage



Sheet: /storage/		
File: storage.kicad_sch		
Title: Storage		
Size: A4	Date: 2023-12-12	**Rev: V1.0**
KiCad E.D.A. kicad 7.0.9		Id: 5/6

Notes:
– For the Oscillators and RF matching circuit: choose C0G capacitors
– For the RF matching inductors: choose ones with high S.R.F, preferably > 5*f_{max}, so >5*2.5GHz
This is because you want the impedance to be the expected value at the frequency range used

Battery Monitor:
– When connecting the board to a lab bench power supply via the battery connector, the balance connector will be floating values, so will Undervoltage (UV) be constantly triggered?

Note

To support flexible cell configurations within BQ76920, BQ76930, and BQ76940, UV is ignored on any cells that have a reading under $UV_{MINQUAL}$. This allows cell pins to be shorted in implementations where not all cells are needed (for example, 6-series cells using the BQ76930).

Reed switch
– The thickness of the BlueRobotics case is 6.35mm, so activation distance must be more than this value

Oscillators
– For the nRF: it is recommended to use a crystal with low maximum load capacitance and/or shunt capacitance. A low load capacitance will reduce both start up time and current consumption. Important for deep sleep.

Programming the STM
If the STM is in bootloader, then USART1 can be used to reprogram the flash, which is connected to the nRF

> The boot loader is located in system memory. It is used to reprogram the Flash memory by using USART1 (PA9/PA10), USART3 (PC10/PC11 or PB10/PB11), CAN2 (PB5/PB13), USB OTG FS in Device mode (PA11/PA12) through DFU (device firmware upgrade)
(From STM32F40X datasheet)

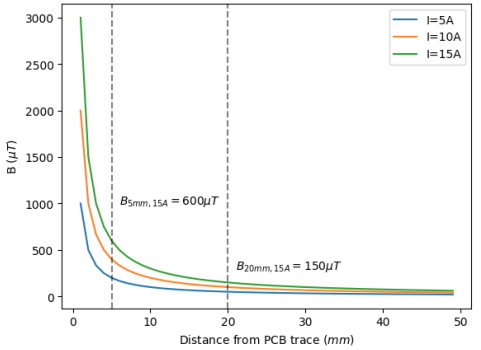
STMF405 can be put in bootloader through
Boot0(pin) = 1 and Boot1(pin) = 0
(From ST bootloader application node AN2606)

Magnetometer
– Preferably choose one with a high range, as to prevent saturation caused by magnetic fields in proximity
– The strongest causes of magnetic interference are the servos, and servo power lines.

Servo magnetic field
– The strongest static magnetic field I could get is 200 uT, that is 3 servos inline.
– The strongest magnetic field I measured is −472uT, that was by moving the servo around the magnetometer manually.
– 3 servos sine waves with different phases, max is 127uT.

PCB trace magnetic field
– Assuming a current of 15 A and the power PCB trace being placed next to the magnetometer at 5mm, then we would have:

B_{5mm} = (0.2 * 10**−6 * 15) / (5 * 10**−3) = 0.0006T = 600 uT
(From NXP AN4247)



Ideally a range of +−1000uT per axis
But in practice, the max B will probably be:
B_{servo} = 200uT
$B_{pcb\_trace,20mm,15A}$ = 150uT
B_{earth} = 25-65uT
B = 200 + 150 + 65 = 415 uT
That is, if the fields align, the average will probably be lower

Sheet: /design_notes/		
File: design_notes.kicad_sch		
Title: Design Notes		
Size: A4	Date: 2023−12−12	**Rev: V1.0**
KiCad E.D.A. kicad 7.0.9		Id: 6/6

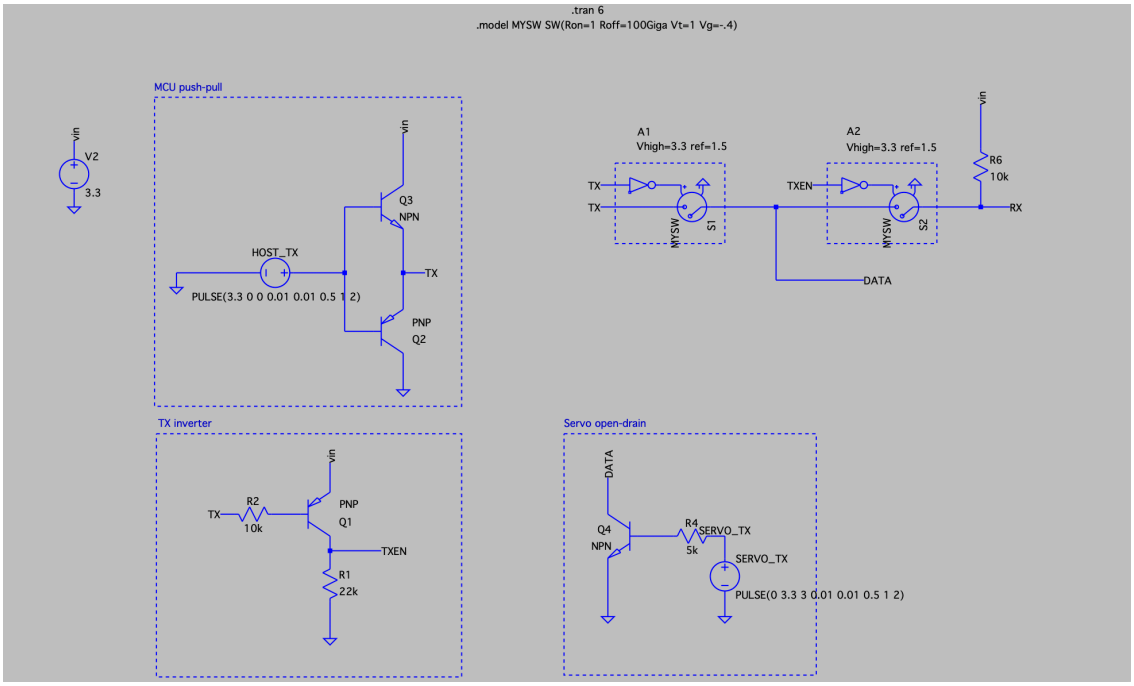


Figure B.1: SPICE simulation of the UART-to-TTL circuit