Delft University of Technology
Master's Thesis in Embedded Systems

Design of a Real-Time Micro-Winch Controller for Kite-Power Applications

Andres Moreno

embedded software

TU Delft Delft University of Technology

Design of a Real-Time Micro-Winch Controller for Kite-Power Applications

Master's Thesis in Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Andres Moreno
A.L.MorenoWandurraga@student.tudelft.nl

5th November 2015

Author
Andres Moreno (A.L.MorenoWandurraga@student.tudelft.nl)
Title
Design of a Real-Time Micro-Winch Controller for Kite-Power Applications
MSc presentation
13th November 2015

Graduation Committee

Prof. dr. K. G. Langendoen (Chair)	Delft University of Technology
Dr. Venkatesha Prasad R (Supervisor)	Delft University of Technology
Dr.-Ing R. Schmehl (External Member)	Delft University of Technology
MSc. Uwe Fechner (Daily Supervisor)	Delft University of Technology

Abstract

Airborne wind energy is a technology to extract energy from high altitude winds. This technology is under heavy development by several companies and universities. An actual problem with the commercialization of the technology is the reliability and safety of the system. In this thesis a real time environment suitable to perform research and further development of the prototype steering and depower control is proposed. Additionally, the overload prevention of the kite lines is researched.

This thesis presents a method to estimate the tension on the kite control tapes using only one tension sensor. Thus, reducing the amount of hardware needed to protect the kite from overloads. The method relies on the characterization of the powertrain efficiency and can be used to estimate the tensions at high loads.

An algorithm to limit the forces on the steering lines by depowering the kite is shown; it controls the depower state of the kite based on the desired depower state, the actual tension, and previous tensions on the KCU's tapes. The tensions history is used to calculate a higher depower state to prevent future overloads, this reduces the amount of action needed by the motors and enable the system to use a brake to save energy. The limiter output is used as an input to a position controller, which allows the project to use off the shelf solutions to build the KCU prototype.

The controller was implemented in a real time system and is able to run as fast as 20 Hz being the communication protocol the execution time bottleneck. The control algorithms were tested using a mathematical model of the kite, the environment, and trajectory control inputs from FreeKiteSim. Three scenarios were considered for the model test, normal operation, overload operation without tension limitation, and overload operation with tension limitation. The apparent wind speed during the reel out phase of the normal scenario is approximately 30 m/s and 35 m/s for the overload scenarios.

During the overload scenario the limiter spent roughly 22% more energy than the normal operation scenario to counteract an increase of 5 m/s in the apparent wind during 3.5 hours of operation, but it spent 15% less energy than the overload scenario without tension limitation.

Contents

Chapter 1

Introduction

The steering and depower (S/D) mechanisms of flexible kites have been investigated by many authors [10], [22], [37]. Several kite controllers have been designed using these principles [9]. However, there is a lack of published implementation details and limitations about these controllers mainly because the majority of them have not been tested in prototypes, and the ones that have been are kept confidential.

The actual TU Delft S/D controller has its own limitations, it does not measure or limit the forces exerted on the steering lines, there is no guarantee of its response times, there are powertrain robustness problems, and the debugging information is lacking (mainly because of the motor driver hardware). These issues have impacted the reliability of the technology and are barriers that have to be surpassed for a successful commercialization of the technology.

What this thesis proposes is a real time S/D controller for a flexible kite to overcome these previous limitations. The scope is limited to flexible kites only, and it should be understood that any mention to kites refers to leading edge inflatable kites unless stated otherwise.

1.1 Wind Energy

We live in a power hungry world with an ever increasing need of energy. During the last years a lot of research has been done on greener and renewable ways to generate power [2] driven by a more environmental aware public.

Wind is one of the main energy sources on which the research has been focused. Using the wind as energy source is not new, it has been used since ancient times for vessel propulsion and wind mills. Nowadays wind energy is extracted mainly by the means of wind turbines [11], and its exploitation is very attractive because it is clean, renewable, and has the potential to supply all the energy needed by the world [3].

1.2 Airborne Wind Energy

Wind turbines are not the only option to extract energy from the wind. During 1960's and 1970's Miles Loyd and others researched the idea to generate power from wind using airborne airfoils [26], [31], [24], but the idea has been around at least since 1930's [27].

During the 1990's Wubbo Ockels started working on the idea of generating power using a laddermill concept [30]. He then formed a group in 2004 at Delft University of Technology to further investigate the idea of the laddermill that was eventually changed [23] to focus on a pumping cycle using Lloyd's idea of crosswind kite power generation [25].

The are several reasons why there is an interest in kites to generate power. Kites allows the system to reach winds at higher altitudes than a conventional wind turbine, these higher winds located at the atmospheric boundary layer are more consistent and have greater speeds, hence, a greater power density [19]. The kite works similar to the tip of a wind turbine, which is the fast moving part that generates most of the power [11]. However, the kite does not need a big tower to support itself, hence kite power systems use a lot less materials than wind turbines. This can be translated into several advantages for the system, like smaller size, reduced construction costs, portability, cheaper maintenance, and a safer operation because there is no need to lift heavy loads or perform works at height.

As of the writing of this thesis high wind energy technology is used commercially for boat traction, but there is no commercial offering for electrical power generation yet. There are several research groups across the globe researching various technology prototypes. The University of Torino with a carousel concept [14], Sky WindPower with a flying electric generator [36], the Makani project backed by Google with an auto lift off system, Ampyx with a pumping cycle glider, and TU Delft with flexible kites working in a pumping cycle [11], just to name a few. Each research team has variations of the concept, some have on-board generation instead of ground generation, which make them heavy and limit the power output; some have rigid bodies which can achieve higher speeds than a flexible kite, but are also heavier and riskier in inhabited areas. And there are some combinations in between, like rigid-flexible kites, and kites mixing on-board and ground generation. The research performed at the TU Delft group is mainly focused on flexible kites and on ground generation [11].

1.3 TU Delft System

The TU Delft system is shown in Figure 1.1. It is composed of a ground station where the power generation takes place, a tether line that transfer the forces of the kite to the ground station, a kite, and hanging between the

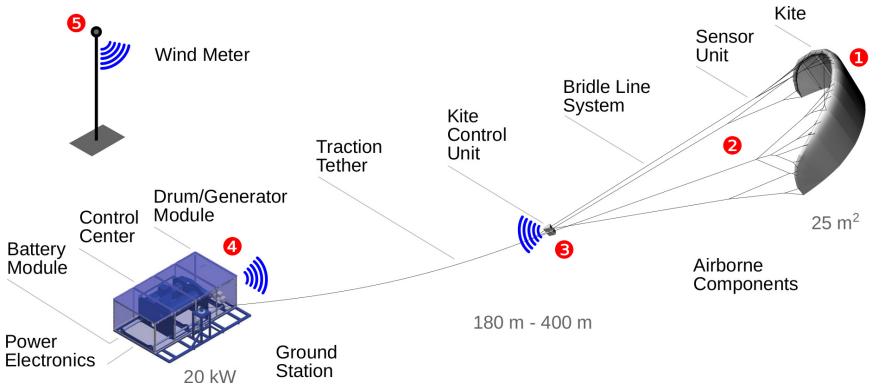

Figure 1.1: System components [11].

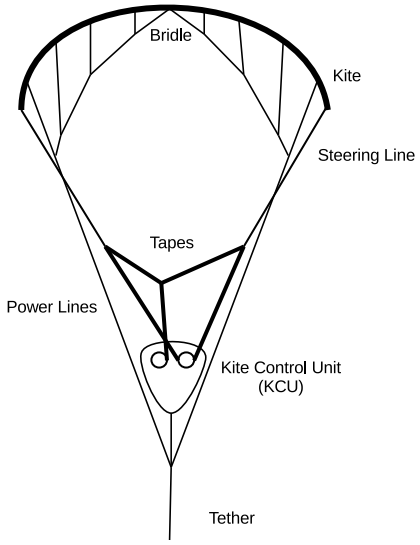

Figure 1.2: Airborne components.

kite and tether is the Kite Control Unit (KCU).

Figure 1.2 shows the airborne components of the system in more detail. The tether is attached to the kite by the bridle or power lines, these lines transmit 80% of the generated force to the tether. The steering lines are connected to the KCU tapes (bold lines) trough pulleys. The central tape is the depower tape and the external tapes are the steering tapes. These three tapes transmit 20% of the force generated by the kite trough the KCU to the tether line. The KCU contains 2 motors, one manipulates the total length of the steering lines (left motor), and the other manipulates the differential length of them (right motor).

1.4 Thesis Organization

This thesis is divided in 7 chapters. Chapter 2 “Kite Power” shows the technical background and the state of the art of the kites S/D controllers. Chapter 3 “System Model” explains the different system components models. Chapter 4 “Controller” shows the design and components of the controller. Chapter 5 “Real Time” shows the real time considerations and partial implementation of the control system. Chapter 6 “Results” shows the results of the controller model simulations and the partial implementation tests. Chapter 7.1 “Conclusions” list the main conclusion points of the thesis, and the proposed future work.

Chapter 2

Kite Power

There are several disciplines involved into developing a controller for a flexible kite. This chapter presents a general overview of flexible kites models focused on the models' steering behavior, the control sub-systems and strategies used to control the kite, and the actual controller limitations and proposed improvements.

2.1 Pumping Cycle

The system generates power by flying the kite in a wind field using a repetitive pattern called a pumping cycle. One pumping cycle is composed by one reel out and one reel in phase, and it is executed in such a way that the power generated during the reel out phase is bigger than the power consumed during the reel in phase, thus the system works as an electrical power generator over a single pumping cycle. The reel out phase and reel in phases are shown in Figure 2.1

The power generated during the reel out phase depends on the tether's reel out speed and the lift force generated on the kite. The reel out speed is controller by the ground station. If the tether line is not being reeled out, or if it is being reeled out at maximum speed then no power is generated. The optimum reel out speed is somewhere in between [11].

The lift force depends on the relative wind speed and angle of attack of the kite among other factors [25]. To increase the relative speed, the kite is flown in a crosswind direction, more exactly, in figure of eight patterns to avoid tether twisting. The force generated by the kite is transmitted to the ground through the tether and electrical power is produced by using the tether to drive an asynchronous generator. Part of this energy is stored in batteries for the own system needs.

When the tether line has reached its maximum length, the angle of attack of the kite is lowered and the kite is pulled back to its initial position using the ground generator as a motor; this phase is called the reel in phase, and

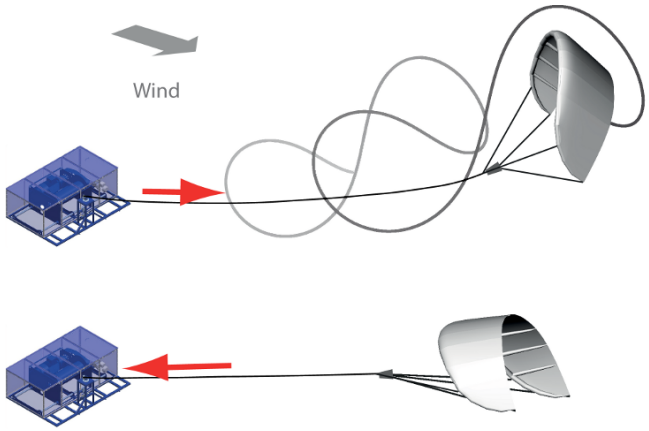

Figure 2.1: Pumping cycle. Reel out (up). Reel in (down) [11]

it consumes some of the electrical energy previously stored in the batteries. No power is generated during this phase.

It can be inferred from this operation description that the kite needs to be highly maneuverable to perform the figure of eight maneuvers. It also needs to react fast to wind gusts to prevent the snapping of the weak link, damage to the kite, the tether, or the generator, and be predictable while remaining controllable.

2.2 Kite Steering Models

In order to understand the steering of the kite it is necessary to understand the kite behavior. The kites used at TU Delft at the moment are three inflatable leading edge kites with 25, 25 and 14 square meters of surface. These kites have flexible structures that complicate the modeling and analysis of the kite mainly because the aerodynamics and structure of the wing are heavily coupled. There have been several models proposed aiming at different needs. Each model has a different degree of detail and simulation time needs. In general, a more detailed model needs more computational time, and research is been done to find a good compromise between complexity and run time for real time applications.

Some of the models are explained below regarding its steering concept. For a more detailed comparison of different models, Ruppert [37] and Bosch [7] present a more complete analysis of the advantages and disadvantages of each one, including the tether, bridle, and control unit models.

2.2.1 One and Four Points Models

The one point model represents the kite like a point mass that moves under the influence of external forces [43]. The steering is modeled as a rotation of the lift vector caused by a tilt on the kite. This model is used to bootstrap the simulations but lacks vital information to model the kite steering. Therefore it is suitable for flight-path simulations because of its simplicity but not for steering analysis of the kite.

The four point model represents the kite as 4 masses. This model includes the rotational inertia of the kite and an additional mass for the KCU. It is a simple and fast model suitable for the development and optimization of flight-path control algorithms and is more accurate and stable than the one point model [16]. The steering of the kite is modeled such that the steering input changes the angle of attack of the side surfaces, the forces, and produce a yaw movement on the kite. This model needs the steering sensitivity of the kite to be identified from experimental data.

2.2.2 Multibody Dynamics

The multibody approach represents the kite and ropes by discrete rigid component attached to each other using different types of joints that constrain the movement of the elements. There are different ways to construct a multibody model of the kite. Williams used multiple plates [44], [45], and Breukels [10] used three different construction blocks, which improved Williams results by adding more deformation modes and a more sophisticated method to calculate aerodynamic forces. In this type of models the deformation of the kite and the acting aerodynamic forces on the body elements are used to model the kite steering.

Breukels' model is able to calculate the kite deformation and steering lines displacements caused by a steering input. It is important to note that the steering input to his model is a force applied on one steering line under which the kite deforms. According to the simulation results the kite reaches a steady state deformation after a couple seconds when a constant force is applied. This is different from models that use the steering lines difference as an input.

The problem with force inputs is that the kite will take different shapes in steady state depending on the apparent wind speed for the same steering inputs. And as tested and verified by Breukels, the deformation of the flexible structure is the main factor in the cornering ability of the kite, hence a constant force on a steering line will not produce the same steering under different wind speed conditions.

2.2.3 Finite Element

Using the finite element method (FEM) the structural behavior of the kite has been modeled by Schwoll [38] and expanded by Bosch [7] to include fluid structure interactions. Geshiere included power generation into Bosch's model and updated the kite model [18]. So far, FEM based models are the most accurate but they are also very slow.

The steering input for these models is the steering lines length difference. Bosch's and Geshiere's models are able to calculate the kite structure deformation and aerodynamic forces generated by this steering input.

The cornering of the kite is modeled by applying the forces found by the fluid structural interaction (FSI) solver to the kite mass that was previously distributed over the bridle attachment points. This allowed Bosch to find the forces on the steering and power lines, although his configuration is different from the current prototype.

2.2.4 Empirical Models

As seen from the previous sections, there are mainly two ways to steer a kite in these models: Using a force input, or a steering line length input. These inputs deform the kite and change its aerodynamic properties, generating a torque on the kite. However, neither of these models give an explicit analytical expression to relate the control action to the turning of the kite.

There is a third approach called the turn rate law [12] that uses an empiric model and experimental observations to correlate the steering lines length difference with the turn rate of the kite. This relation is shown in equation 2.1, where c_1 and c_2 are kite's parameters to be identified, v_a is the apparent wind, δ is the steering input, $\mathbf{g}$ is the gravity, and $\mathbf{y_B}$ is the y axis of the kite-fixed reference frame ($\mathbf{y_B}$ goes from the left to the right wing tip). This empirical relation has been validated by others authors and found to be valid if the kite is in a sufficiently powered state [22], [7], [37].

$$\dot{\psi} = c_1 v_\mathrm{a} \delta + c_2 \, \frac{\mathbf{g} \cdot \mathbf{y_B}}{\mathrm{g}} \tag{2.1}$$

Under normal flight conditions the seconds term is much smaller than the first one and can be neglected to simplify the design of the steering controller.

2.3 Kite Control

The system control can be divided in three parts [5], the ground station control, the trajectory control, and the S/D control. The ground station controls the tether line length, the trajectory controller guides the kite in a path perpendicular to the tether and generates the desired steering and

depower settings for the S/D controller. In this section only the trajectory and S/D controller are explained as they are the only two involved in the steering process. The ground control is omitted but can be consulted in Fechner work [15].

2.3.1 Trajectory Control

The trajectory control is responsible for making the kite follow a predefined trajectory in the sky. This trajectory is generated by a flight planning algorithm that optimizes the kite trajectory for power output and robustness.

The trajectory controller depends on the steering and depowering controllers and assumes that these states of the kite can be manipulated. Albeit flight planning controllers are the main internal client of the steering/depower controller they are not the aim of this thesis. A lot of research has been done in trajectory planning and control, including model predictive control, neural networks, parameter variation, etc. Baayen did a good analysis on the different researched approaches [4].

2.3.2 Steering

In the reviewed literature the steering controllers are heavily coupled with the trajectory tracking controllers, sometimes the boundaries of each controller are not clear. In this thesis we are concerned only with the steering mechanism, not how to determine the correct heading vector. Only kites steered by an airborne control unit are considered. The approaches revisited in this section were designed to test kite models or trajectory controllers, very few of them have been tested on real prototypes.

The steering control of a flexible kite has several challenges. The modeling of the turning effect is really hard to do, and actual models differ enough from reality or are too slow as to be used in model based control as shown in Section 2.2. Some work on trying to correct the models' errors using adaptive control was tried by Baayen [4], [5], but his controller did not work on the TU Delft prototype's tests.

More accurate but complex models pose the difficulty of depending on hard to measure variables and are not suitable to perform control. The controllers that have been successfully tested in prototypes are mainly based on the turn law [13], [22]. The main difference between Erhard's and Jehle's approaches to the turn law is the consideration of actuators effects into the control loop.

Turn Rate Based Controllers

Jehle's controller is based in the turn law but as the Baayen's controller it is also very complex. It uses adaptive control to estimate unknown parameters to perform a dynamic system inversion. As expressed by him, this

estimations are suspected to be a source of errors for the controller. This controller had a very limited time of testing with the prototype. Several systems failures not related to the controller occurred, including loss of data and insufficient time to tune the controller. Overall, the controller was able to flight figures of eight maneuvers successfully.

Bosch [7] based his controller on Jehle's and included a dependency of the steering gain on the apparent wind speed, furthermore, he considered the manipulation of each steering line separately, which is not possible with the current TU Delft prototype, and finally, he considered a coupling between the steering controller and the depower controller, although this is to limit the forces on the lines to stabilize his model and not to account for the loss of maneuverability of the kite. This was needed because his model uses a stiff straight tether. These improvements were successfully tested with his model, but not on the prototype.

On the other hand Erhard's [13] steering controller has been tested successfully in prototypes. It is simpler and more practical than the previous approaches. It does not use system models except the turn rate law and includes implementation details like delays and actuators limits that are missing in other controllers. However, in this model the actuators are also simplified models consisting on limiters and rate limiters, therefore motor torque constraints were not considered. His controller consists of a dynamic system inversion (simpler than Jehle who included the tracking controller into the inversion) and cascaded feedforward/feedback controllers, very similar to a conventional position controller. This steering controller is coupled with the trajectory controller trough the feedforward inputs.

The power state of the kite affects heavily the turning ability of the kite, Jehle [22] mentions that the powered state of the kite can be included into a fitting coefficient c_1 that models the kite yaw rate, and Erhard includes the power state of the kite implicitly into the air path velocity using the glide ratio. Both controllers do not consider the manipulation of the power state of the kite and assume a constant power state (either powered or depowered). This consideration is important as varying the angle of attack of the kite during the reel-out phase can help limit the forces on the steering lines.

2.3.3 Depowering

Depowering a kite means reducing the aerodynamics forces on it by reducing its angle of attack. It is achieved by reeleing out both steering lines at the same time, and by the same length. The power/depower controllers are simple, they translate a depower setting into the corresponding line length and reel in or out the corresponding amount of tape.

Because depowering the kite changes the forces on the tether there is a coupling between the ground station controller and the depower controller. Hence both controllers need to coordinate the depower state of the kite,

otherwise instabilities could appear in the system.

It is important to mention that the power setting of the kite affects the maneuverability of the kite, so there is a tradeoff between depowering the kite to consume less energy and being able to steer the kite during the reel-in phase. This effect has been researched and explicitly included in Fechner's steering model [16].

2.4 Kite Control Unit

There has been only a handful of KCUs built and tested, and unfortunately most of the details of these KCUs are confidential [9]. In general, the KCUs run the trajectory and S/D algorithms and manipulate the S/D lines by using small DC motors.

The main differences between the KCUs are how they are powered (Batteries or cable from ground station) and how they manipulate the steering lines. There are two main ways to arrange the actuators in the KCU; one is to operate each steering line with one motor, in which case the kite is depowered by reeling out tape on both motors at the same time; and the other arrangement is to operate both lines with one motor, and the difference between them with another.

The TU Delft KCU is powered from internal batteries and uses one motor for depower and one for steering. The advantage over the single line actuation is that the depower motor can be blocked with a brake to save power. This motor arrangement is shown in Figure 1.2.

2.5 Software Safety Considerations

Airborne systems have strict safety requirements because of the risks associated with system failures. Anticipating the commercial use of the system, the possibility to develop the embedded software controller conforming to the DO-178[1] was assessed, as this is the primary document used for the FAA and EASA for software certification. It is important to note that because of the research nature of the project, the current prototype is excluded from EASA regulations [1], and this might be the case as well for the final commercial product depending on its weight. If this is the case then the certification would have to be done on a country basis [42].

After a quick review of the DO-178, it was concluded that the current research nature of the project is not ideal to develop software under those requirements. There are several reasons for this conclusion:

- There is no free certifiable operating system [35], and the available ones are very expensive for this phase of the project.

[1]Software Considerations in Airborne Systems and Equipment Certification

- The development of conforming software require expertise in the field. It is a very complex task that requires specialized methodology and tools.
- The process is very time consuming and could not be achieved in the allocated time. It also needs more resources that are currently not available.
- The software can only be as safe as the hardware it runs on, and safety oriented processors are more expensive and time consuming to program.

2.6 S/D Controllers Limitations

Some of the reviewed S/D controllers were implemented in MATLAB or similar software but were not tested in real hardware. As expressed in section 2.4 there is limited information on the controller implementation done by different projects. Therefore, the limitations presented here are based on the available published information and the actual TU Delft KCU implementation.

The S/D controllers tested on hardware were able to control the kite heading successfully, as shown by the Skysails propulsion system and the TU Delft prototype experimental results. However, there are opportunities to improve the current controller and to make the system more robust, safe, and responsive.

The presently implemented controllers do not measure or limit steering lines forces. Including this features could be useful for safety and research purposes. It will help to understand the effect of wind gusts on the controllability of the kite, identify and validate kite models, react faster to overloads, and provide more information for research.

These steering controllers also assume that the kite is in a sufficiently powered state. But if the power state of the kite is going to change dynamically to control the forces, it is necessary that the steering controller adapts to the kite power setting. Using the model proposed by Fechner [16] it could be possible to develop an steering controller able to compensate for varying depower states and not be limited to only two previously identified operation points.

Implementation details are very important as the system needs to be reliable, fast, and safe. The robustness of the steering controller could be improved if the actuator characteristics are included into the steering model. This aspect becomes more important because the certification of the technology will require response time guarantees. It is estimated that between 40% and 50% of the crashing events of the TU Delft prototype happened because of KCU steering reliability problems. Overloads and thermal issues

are the main suspects. To mitigate these problems, not only the hardware has to be replaced, but also the software has to be rewritten to be more robust and with more debugging/logging capabilities.

A crucial limitation of the present controller for complete automated flight is the correction of the steering lines bias. At the moment an operator is needed to compensate the bias at the system start time, but the bias changes with time as the steering lines creep and knots tighten.

2.7 Proposed Improvements

Out of the listed limitations, the proposed controller objective is to provide a method to estimate the load on the lines and limit them to safe values using the least amount of sensors to reduce the weight and complexity of the hardware.

To guarantee the controller response time, the mechanical and electrical response times are included into the controller design, furthermore the controller is implemented in a real time environment. To improve the reliability of the system the mechanical limits and a thermal model of the motor are included in the controller design.

As an additional improvement not related to the previous section limitations, the code is made portable by avoiding frameworks with limited chip support. This is desired as the KitePower 2.0 is a research platform and the controller is expected to be ported to different processors in the future.

2.8 Summary

A brief explanation of the technology was given. Several models explaining the kite steering mechanisms were shown and why they are (or not) practical to implement a real time controller.

There are several groups researching the airborne wind energy production, but only a couple prototypes exist that use a flexible kite[2]. State of the art controllers are able to steer the kite and do proper trajectory planning, but there is still room for safety and reliability improvements.

The turn rate law is the steering model we are going to work with because it is simple and has worked on the field. The controller should receive the depower and steering inputs from the trajectory controller, control the S/D line lengths, estimate the forces on the KCU tapes, and limit them below a desired value.

A proper real time platform and partial implementation of the system is proposed. The compliance with DO-178C was determined to be out of the scope of this work.

[2]At the time of writing this thesis DARPA has presented a new prototype called TALONS using this technology.

Chapter 3

System Model

3.1 Introduction

In this chapter models of the system's components are shown. The system is composed of a kite, the wind field, and the KCU. The wind model generates the apparent wind seen by the kite, including the turbulence in the expected worst case scenario. The kite generates the forces on the lines and the KCU controls the kite. Figure 3.1 shows the system model overview.

3.2 KCU

The KCU contains two control subsystems (for depower and steering). Each subsystem has a motor to reel in or out a tape on drums to control the attached tape length. These drums are connected to the motors using planetary gears. The reeled in tape changes the effective radius of the drum, hence it increases the torque on the motor and the speed at which the motor can change the steering line length. This section shows the modeling and interaction of these components.

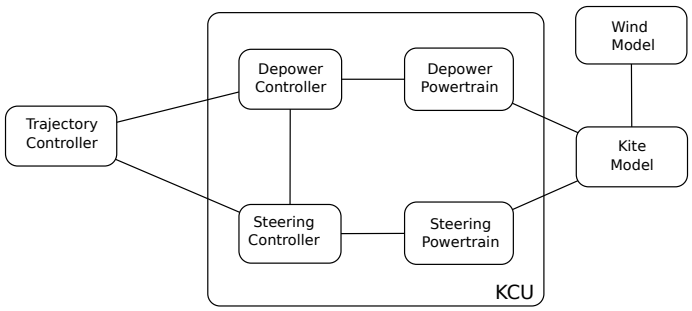

Figure 3.1: Relations between system components models.

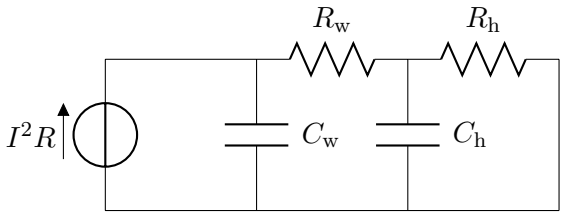

Figure 3.2: Equivalent thermal circuit.

3.2.1 Motors

The preselected motors for the KCU are brushless DC motors (BLDC) and they are modeled as armature controlled motors. These motors are often confused with permanent magnet synchronous motors (PMSM) because both have similar construction. The main difference between these two types of motors, is that BLDC motors have a trapezoidal back electromotive force (BEMF), while PMSM have a sinusoidal BEMF [33]. PMSM are driven with vector control (sinusoidal input) and their windings are built for this purpose. The BLDC motors are commonly driven by commutating a DC voltage between the windings (like brushes would do). The motor and driver preselected for the KCU use block commutation.

The motors' state space model is given by equations 3.1 and 3.2. These equations are a modified version of the models shown in the mechatronic books [8], [20]. A detailed explanation of BLDC motors' friction losses (negligible), core losses (substantial), and how to compensate for them in the model is explained by Stemme [41].

$$\begin{pmatrix} \dot{\theta} \\ \dot{\omega} \\ \dot{I} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -b/J & K_{\mathrm{m}}/J \\ 0 & -K_{\mathrm{m}}/L & -R/L \end{pmatrix} \begin{pmatrix} \theta \\ \omega \\ I \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -1/J \\ 1/L & 0 \end{pmatrix} \begin{pmatrix} V \\ T_{\mathrm{l}} \end{pmatrix} \tag{3.1}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta \\ \omega \\ I \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V \\ T_{\mathrm{l}} \end{pmatrix} \tag{3.2}$$

Where θ is the angular position of the rotor, ω is the angular speed of the rotor, I is the current consumed by the motor, b is the friction coefficient, J is the inertia, K_{m} is the torque constant, R is the winding resistance, L is the winding inductance, V is the input voltage, and T_{l} is the load torque.

It is necessary to note that the inertia J depends on the whole system, not only the motor, and T_{l} is the motor load torque referred to the motor side of the gear. The motor brake is modeled as a 1 or 0 gain on the input load.

As it is desired to monitor the motor temperature it is necessary to have a thermal model. The model used is based on the equivalent electrical circuit of the system heat transfer as shown in Figure 3.2. This model combines the three basic mechanisms of heat transfer, conducting, convecting, and radiating [41]. It assumes that the motor is surrounded by air and that the KCU allows the air to flow around the motor.

The thermal behavior of the motor is then described by the following state space model.

$$\begin{pmatrix} \dot{T_{\mathrm{w}}} \\ \dot{T_{\mathrm{h}}} \end{pmatrix} = \begin{pmatrix} -1/(R_{\mathrm{w}}C_{\mathrm{w}}) & 1/(R_{\mathrm{w}}C_{\mathrm{w}}) \\ 1/(R_{\mathrm{w}}C_{\mathrm{h}}) & -1/(R_{\mathrm{w}}C_{\mathrm{h}}) - 1/(R_{\mathrm{h}}C_{\mathrm{h}}) \end{pmatrix} \begin{pmatrix} T_{\mathrm{w}} \\ T_{\mathrm{h}} \end{pmatrix} + \begin{pmatrix} 1/C_{\mathrm{w}} \\ 0 \end{pmatrix} (I^2 R) \tag{3.3}$$

3.2.2 Gear

The planetary gears are very difficult to model and unfortunately the efficiency curves are not provided by the manufacturer; only the maximum efficiency is provided but that is not enough as the tension estimator depends heavily on understanding the torque transmission trough the powertrain.

The efficiency of the gear depends on the internal losses which are caused by static and dynamic friction between the gear teeth. The friction depends on the higher torque, speed, temperature, construction, and lubrication [32].

Experiments were carried out to identify the efficiency curves of the system, and the gear. In these experiments a constant lubricant temperature was assumed. Figure 3.3 shows the efficiency of the system. The efficiency was calculated by measuring the DC power supply current with an ammeter and calculating the mechanical output using the known load and speed.

Figure 3.4 shows the efficiency of the gear; it was calculated using the current reported by the controller (which is proportional to the torque) and the known load and speed. It can be observed that there is a big range of operation efficiencies, the speed affects the lighter loads more than the heavier ones, and the efficiency hits zero when the load is not able to drive the motor over a certain speed.

To be able to use these results in the tension estimator the motor current is used as input instead of torque or efficiency. The drive train model is assumed to be piecewise linear on the angular speed ω and torque T_{l} as described by equation 3.4.

$$I(\omega, l) = \begin{cases} a_1\omega + b_1 T_{\mathrm{l}} + c_1 & \omega < 0 \\ a_2\omega + b_2 T_{\mathrm{l}} + c_2 & \omega > 0 \end{cases} \tag{3.4}$$

From this model it can be see that the powertrain efficiency is direction dependent and that the current is not defined when the drivetrain is not moving. This is so because the static friction will oppose any force below

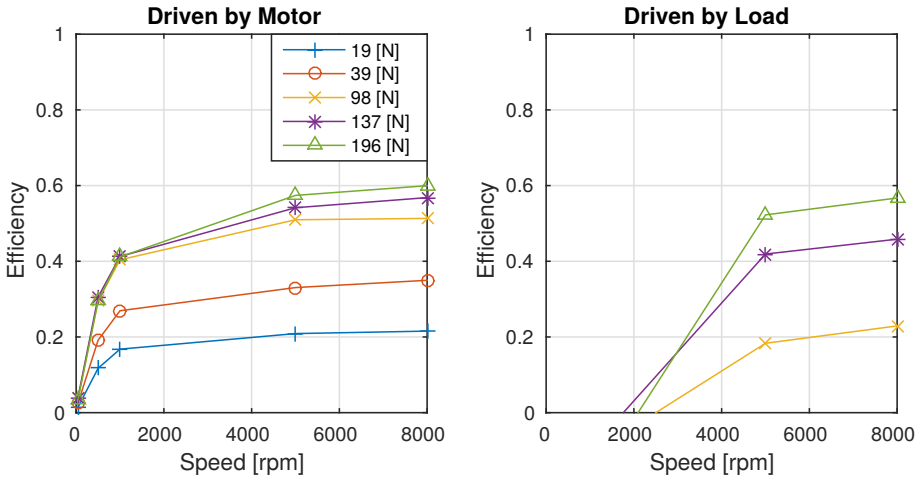

Figure 3.3: Experimental data of the system's efficiency when the motor's torque is higher than the load's torque (left), and when the load's torque is higher than the motor's torque (right).

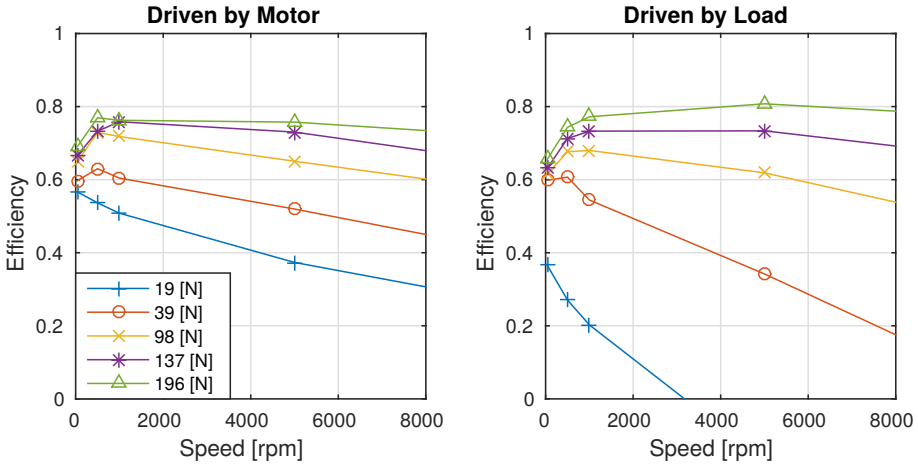

Figure 3.4: Experimental data of the gear's efficiency when the motor's torque is higher than the load's torque (left), and when the load's torque is higher than the motor's torque (right).

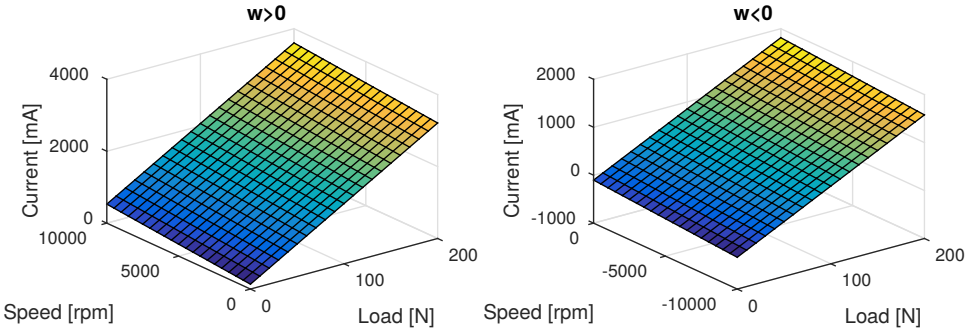

Figure 3.5: Curve fitting of the powertrain characteristics in terms of motor current for a positive speed (left) and negative speed (right).

a certain threshold, hence it is not possible to determine the magnitude of this force when the motor is not moving unless the force is higher than the static friction.

Several tests were performed to identify the model coefficients using different loads and different velocities. The results of the curve fitting are shown in Figure 3.5. The equation coefficients, the R-square coefficient of determination and root mean square error (RMSE) are listed in Table 3.1.

Table 3.1: Model Coefficients

$w > 0$		$w < 0$		Units
a_1	135.3	a_2	-102.3	mA/rpm
b_1	9.573	b_2	100	mA/N
c_1	0.03936	c_2	0.0235	mA
R^2	0.9971	R^2	0.9923	
RMSE	32.01	RMSE	32.32	mA

For the synthesis of the controller the gear is modeled with two gains, one for the velocity relation between the gear input and the gear output, and another for the gear torque input and output.

3.2.3 Drums

Each motor has a drum connected to the gear. In the case of the depower motor, the drum reels in or out a single tape to change the angle of attack of the kite. On the other hand, the steering drum is double, when it rotates in one direction one steering tape is reeled in and the other tape is reeled out.

The drums are modeled using Archimedes spirals in order to relate the

reeled in tape length and the drum effective radius (which affects the motor torque) with the motor revolutions:

$$L = \int_a^b \sqrt{r_o^2 + \left(\frac{dr}{d\theta}\right)^2} d\theta \tag{3.5}$$

Where a is the initial position in radians, b is the final position in radians, r_o is the initial radius, and $\frac{dr}{d\theta}$ is the tape thickness divided by π.

The moment of inertia of the drum changes with the amount of tape that is reeled in. However, because of the high gear ratio these changes are negligible when referred to the motor side of the gear. Hence, the moment of inertia was calculated analytically assuming only half of the tape was reeled in.

3.3 Wind

They main purpose of the wind model is to generate the forces on the steering lines, not to simulate the complete 3D wind field the kite is flying in. Therefore, the wind speed vector is aligned with the kite, no side slip angle is assumed and no rotational effects of the turbulence are considered. There is experimental data of the apparent wind speed at the kite height that is useful to adjust the model parameters of the wind gust model.

The main methods used to simulate the wind gusts are the von Kármán wind turbulence model, the Dryden turbulence model, and the one minus cosine discrete model. A simplified way to generate the wind field speed was chosen. The wind model is based on the Dryden continuous turbulence with the superposition of a one 1-cosine discrete gust. These models are specified in the MIL-F-8785C [28] standard, the MIL-STD-1797A [40] standard and the FAA 14 CFR 25 subpart C [34] code. No wind shear model was included. The 1-cosine model assumes the wind gust to be stationary in space, that is, the raise time of the apparent wind is determined by the wind gust length and the kite speed.

The idea is to simulate the controller under worst normal operation. It is expected that if the weather is very adverse (storm) the kite is landed for safety reasons. Hence, the parameters for the generation of the wind gust were extracted from Gage [17] for a moderate-severe wind gust (10^{-4} probability of exceedance) and these values were compared to experimental kite flight data to define the gust window size. The worst case for the altitude is 300 m (the kite maximum operation height) because the wind speed increases with altitute within the kite operation range.

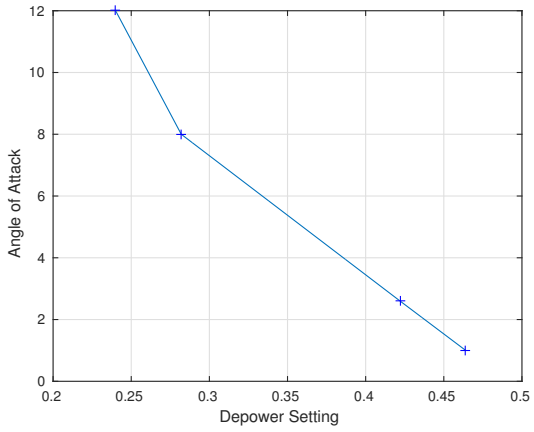

Figure 3.6: Linear interpolation of the depower setting vs angle of attack.

3.4 Kite

The kite model inputs are the apparent wind, the normalized depower setting, and the normalized steering setting. Its outputs are the forces generated on the KCU tapes. It is expected that the trajectory controller calculates the needed turning rate using the turn rate law, and translates it into normalized S/D inputs for the steering controller.

The generated force on the kite F_a is modeled using the equation 3.6. The apparent wind v_a is provided by the wind model; the projected area A of the kite is given by its physical dimensions; the air density ρ is calculated using the barometric formula for a height of 300 m; and the total force coefficient C_r is given by equation 3.7, where C_l is the lift coefficient and C_d is the drag coefficient.

The aerodynamic coefficients depend on the angle of attack of the kite, and are given by experimental data included into FreeKiteSim[1] by Fechner and by the characterization work of the system done by Ruppert [37]. The angle of attack is calculated by linear interpolation of experimental data provided by Fechner as shown in Figure 3.6. This relation between the angle of attack and steering input is an average obtained during figure of eight flights.

$$F_\mathrm{a} = \frac{1}{2}\rho A v_\mathrm{a}^2 C_\mathrm{r} \tag{3.6}$$

$$C_\mathrm{r} = \sqrt{C_\mathrm{l}^2 + C_\mathrm{d}^2} \tag{3.7}$$

[1] FreeKiteSim is distributed under an open source software license and can be obtained from https://bitbucket.org/ufechner/freekitesim

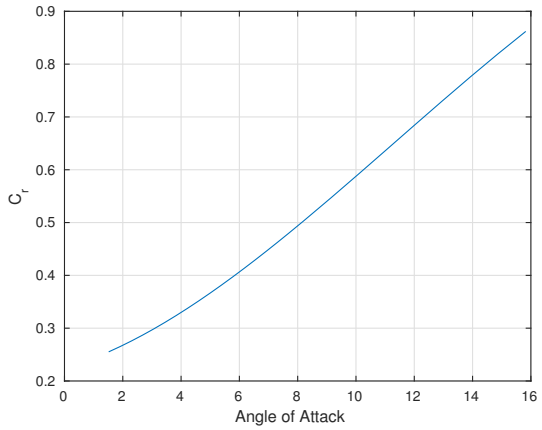

Figure 3.7: Angle of attack vs C_r.

3.4.1 Force Distribution

Out of the total force that the kite generates, 80% of it is transmitted to the tether line trough the power lines, and only 20% is transmitted trough the KCU.

The distribution of the forces on the depower and steering tapes are affected by several factors like the kite shape, the wind speed, the depower state and the steering state of the kite [21]. To model the distribution of the forces it is assumed that the drag forces on the lines are much smaller that the force generated by the kite, and that the kite wing span is constant

The forces on the S/D tapes are distributed using a linear interpolation between the corner cases as shown in Figure 3.8, total depowered no steering, total depower total steering, complete powered no steering, and complete powered complete steering.

The steps to calculate the forces on the lines are:

1. Obtain the depower state of the kite.
2. Obtain the angle of attack from the depower state. See Figure 3.6
3. Obtain the C_r coefficient using the angle of attack. See Figure 3.7
4. Obtain the force generated by the kite using Equation 3.6
5. Obtain the 20% of that force.
6. Assume no steering and distribute the force using only the depower state.
7. Redistribute the forces by applying the steering state.

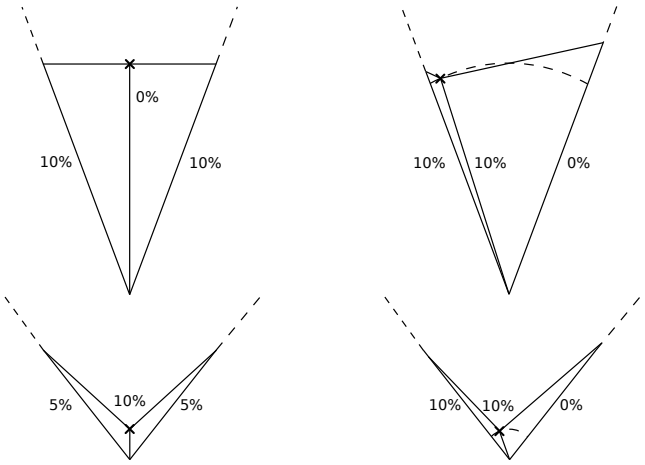

Figure 3.8: Load distribution corner cases in percent of the tether force. 100% depower and 0% steering (top left), 100% depower and 100% steering (top right), 0% depower and 0% steering (bottom left), 0% depower and 100% steering (bottom right).

To the knowledge of the author, there have not been experimental measurements of the forces on the S/D tapes, but fortunately, Bosch's [6] model uses the forces on the bridle and S/D lines attachment points as the interface between the kite and the tether models. The results of his simulations support the idea that the distribution of the forces on the steering lines is linear with respect to the steering input of the kite (ΔL in his paper).

Chapter 4

Controller

The current TU Delft kite power prototype is going to be scaled up to generate more power. This new prototype needs a new KCU that is able to handle higher forces and is robust enough to achieve a 24 hours continuous flight time. Part of the requirements to achieve this is that the controller should be able to monitor and limit the forces on the tape, and limit the motor's current to prevent any damage to them.

Although the old KCU and the new one have the same working principles, the actuators and test data is sized up to match the requirements of the new prototype. The aim of the controller is to provide a minimal reel speed of 0.3 m/s [9], be able to monitor the forces on the depower and steering tapes, and limit them below 800 N while using less sensors to keep the KCU mass and complexity low. The system actuator selection was done previously by Braun for the KitePower 2.0 KCU [9].

The controller was designed with MATLAB using the component models explained in chapter 3. The design was tested with data from FreeKiteSim and a partial implementation was done in C to test the estimators accuracy and the ability of the algorithms to run in a real time environment.

The controller inputs are the normalized steering and depower settings from the trajectory controller, and at least one tension sensor for the steering force estimator. The depower input is a normalized value beetwen 0 and 1, being 1 the most depowered state of the kite; the steering input is normalized between -1 and 1, being -1 the maximum steering to the left and 1 the maximum to the right.

In this chapter the design of the controller is presented.

4.1 Controller Model

The complete controller is divided into the steering and depower subsystems. Each subsystem has the same basic architecture as shown in Figure 4.1, and there is additional communication between the subsystems in order to use

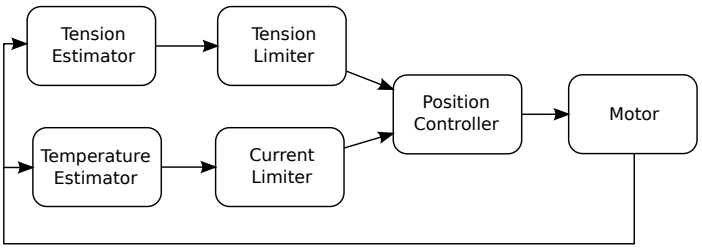

Figure 4.1: Communication between subsystem's components.

the steering tapes as input for the tension limiter.

4.1.1 Position Controller

The position controller input is the desired reel in length corresponding to the desired depower or steering command. It uses three nested PI control loops, the inner loop for the motor current, a middle loop for the motor speed, and an outer loop for the position. Each PI controller has anticlamping to avoid output signal overshoot, and the current loop controller has a saturated output to model the physical limitations of the real system. The position controller output is a voltage value; this voltage is used as input for the motor model, which includes the driver power state as explained in section 3.2.1.

4.1.2 Estimators

One of the goals of this work is to estimate and limit the forces on the KCU tapes using the smallest amount of sensors. To achieve this, an observer was implemented that uses the motor state and the system model to estimate the tension on the lines. An additional observer was implemented for the winding temperature of the motor as this is not made available by the physical controller but it is needed for post mortem analysis, limiting the amount of force the motor can exert, and ultimately affects the controllability of the kite.

The inputs of the observers, current and speed, were filtered by low pass filters (50 Hz and 5 Hz respectively) and sampled with a zero order hold before making the readings available. This simulates the delay experienced by the realtime system and helps to evaluate if the implementation is fast enough to keep the system stable.

Temperature Estimator

The motors of the KCU have a physical limit of the winding temperature. The actual temperature of the winding limits the maximum current that can be feed into the motor to keeps the temperature low enough such that the

isolation is not damaged and the motor rendered useless. Unfortunately it is not possible to measure the winding temperature directly but it is believed to be one of the causes of failures in the current KCU.

As the amount of weight of the KCU reduces the amount of power that can be generated by the system, it was necessary to reduce the size and weight of the actuators. The motors were selected such that they work overloaded for short periods of time. However, the thermal analysis of the motors was not done with data corresponding to the kite flight patterns as there was not a complete model of the KCU to analyze this phenomena. Hence it was not known if the motors could be reduced in size or how much could they be overloaded.

The temperature observer is based on the state space model presented in section 3.2.1 with the difference that it allows an input from a housing temperature sensor to correct the model deviations. The inclusion of a housing sensor through a Luenberger gain allows the system to correct itself to the true temperature of the system. This correction is necessary because the current on the motor can change faster than the sampling time of the S/D controller, hence, the winding can get hotter or cooler depending on the current peaks between the samples.

Tension Estimator

The tension estimator for the steering and depower controllers work on the same basis: Knowing the system state and the motor current, it is possible to determine the torque on the drum. In the depower tension observer the torque on the drum is exerted only by the depower tape, while in the steering observer the torque is the difference of both steering tapes actions, hence is not possible to determine both tensions without at least one sensor.

To determine the differential tension on the drum the total addition of the torques on the system as expressed in equation 4.1 is used; where α is the motor angular acceleration, J is the system rotational inertia, and T_i are the torques on the system.

$$\Sigma T_i = \alpha J \tag{4.1}$$

In the case of the depower controller the equation reads:

$$T_\mathrm{l} = \alpha J + T_\mathrm{m} \tag{4.2}$$

Where T_l is the torque of the load referred to the motor side of the gear, T_m is the motor torque, α is the motor angular acceleration, and J is the system moment of inertia referred to the motor side of the gear.

The load torque can be estimated if the motor torque and the acceleration of the load can be determined. Fortunately, the torque of a motor with permanent magnets on the rotor is directly proportional to the stator

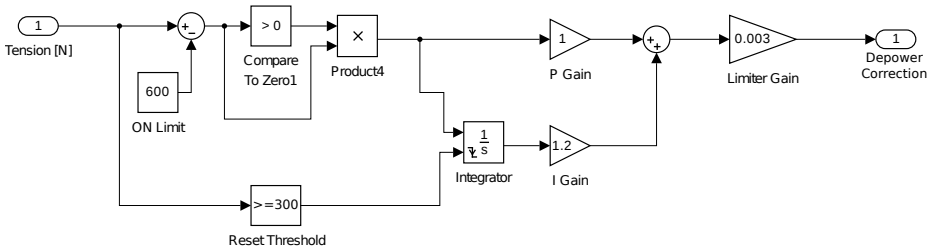

Figure 4.2: Depower tension limiter.

current (by a factor K_m); the moment of inertia of the system can be found analytically and the acceleration of the system can be obtained by differentiating the angular speed of the motor which can be read from the physical controller.

Replacing the known variables into equation 4.2 the tension on the line referred to the motor side of the gear can be calculated by equation 4.3:

$$T_\mathrm{l} = \dot{\omega} J - I K_\mathrm{m} \tag{4.3}$$

To estimate the load at the drum side of the gear (the real tape tension), the mathematical model of the gear derived in section 3.2.2 is used.

4.2 Tension Limitation

The forces generated on the kite depend mainly on the angle of attack and the apparent wind as shown in equation 3.6. The ground station can reduce the apparent wind on the kite by reeling out the tether line faster, but the tether line acts as a delay and damper in the system [10]. The ground station will detect the kite overload after the KCU does, hence, is more effective to perform the tension limitation in the KCU.

In the KCU there are two actuators and hence two possible actions to reduce the tension on the tapes. One way is to redistribute the forces and the second way is by depowering the kite. The first method poses the problem that by redistributing the load on the tapes, the steering action is reduced, effectively making the kite harder to steer, or uncontrollable. It also does not reduce the total forces on the kite, it just reduces the load on the higher loaded tape and increases the load on the other ones.

Depowering the kite is a more appropriate method to reduce the tension on the tapes but the range of motion in the depower tape is bigger than in the steering tape, hence it takes longer than the redistribution method to limit the forces. The advantage is that it preserves the maneuverability of the kite.

Figure 4.2 shows the block diagram of a single tape tension limiter. It is necessary to use this mechanism in each tape because the brake on the

depower tape limits the ability to estimate its tension. The correction of each block is summed up and used to increase the depower state of the kite.

Each limiter acts when the tension on its own tape is bigger than a given threshold. It increases the actual kite depower state by an amount proportional to the overload on the tapes. This action reduces the overload on the tapes effectively, but creates oscillation around the threshold value. To prevent the oscillations, the limiter includes an hysteresis behavior, hence, the limiter is turned off at a lower value than the one used to turn it on.

To avoid excessive operation of the motors an integrator keeps the correction even after the overload of the tapes is gone (its input is strictly positive). This allows the controller to use a brake on the depower tape to prevent motor overheating and excessive energy consumption.

4.3 Current Limitation

The current limitation of the motor is divided into instantaneous and overload current protection. The instantaneous current is limited to a fixed value even if the thermal model of the motor permits to use a higher current.

If we assume that the motor housing remains at a constant temperature (the limiter runs much faster than the housing temperature time constant) between sampling times, we can use the first state space equation 3.3 to find the maximum overload current during the sample time duration:

$$\dot{T_{\mathrm{w}}} = a_{11}T_{\mathrm{w}} + a_{12}T_{\mathrm{h}} + b_{11}I^2R \tag{4.4}$$

Where T_{w} is the winding temperature, T_{h} is the housing temperature, I is the motor current, R is the motor electrical resistance, $a_{11} = -1/(R_{\mathrm{w}}C_{\mathrm{w}})$, $a_{12} = 1/(R_{\mathrm{w}}C_{\mathrm{w}})$, and $b_{11} = 1/C_{\mathrm{w}}$.

If we solve the differential equation and replace the final temperature with the maximum winding temperature, then we are able to find the maximum allowed current:

$$I^2_{\mathrm{max}} = \frac{T_{\mathrm{max}} + (1 - e^{a_{11}T_{\mathrm{s}}})\frac{a_{12}}{a_{11}}T_{\mathrm{h}} - T_{\mathrm{w}}e^{a_{11}T_{\mathrm{s}}}}{(1 - e^{a_{11}T_{\mathrm{s}}})\frac{b_{11}R}{-a_{11}}} \tag{4.5}$$

Where I_{max} is maximum allowed current during the next sampling time T_{s}.

4.4 Summary

The controller reads the actual current consumption and speed of the motors, estimates the tension on the KCU tapes and the motor temperature. It checks if any of the tensions are above a safety threshold, if they are, it generates a correction term to depower the kite; this term is subtracted from

the position control loop, increasing the depowered state of the kite. With the temperatures estimates the maximum allowed current is calculated and a clipping function is inserted before the current loop. The final controller for the depower subsystem can be seen in Figure 4.3.

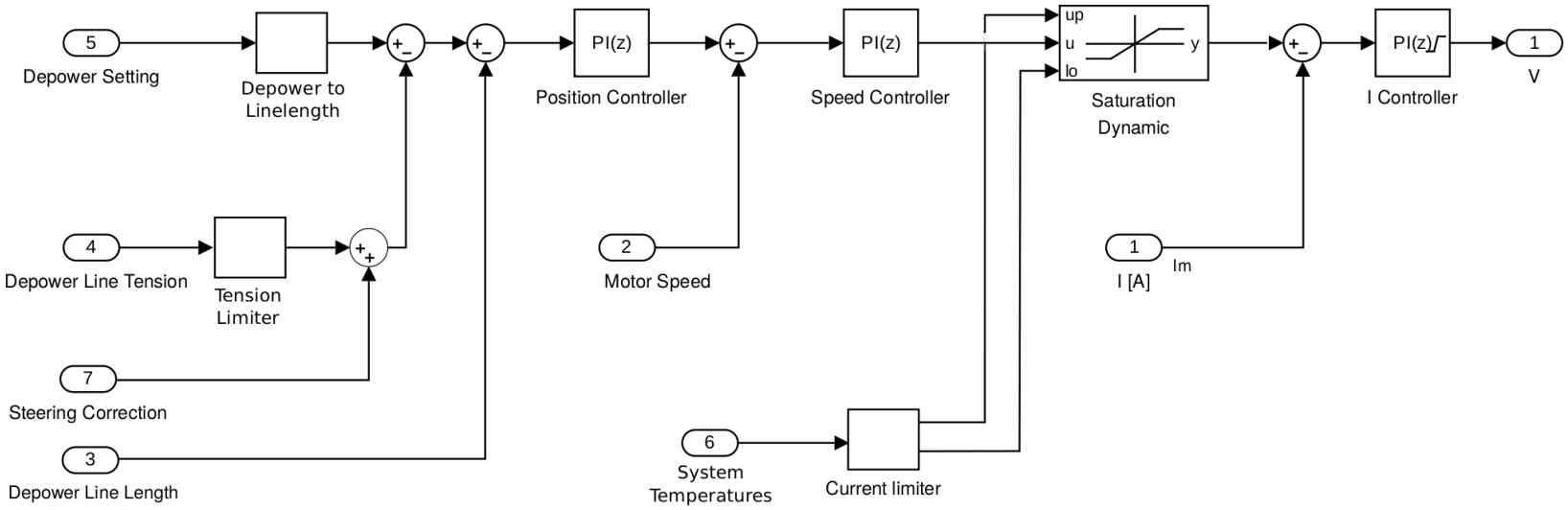

Figure 4.3: Depower controller.

Chapter 5

Real Time Implementation

5.1 Introduction

The implementation of the new controller as a real time system was deemed necessary for several reasons including safety, future certification purposes, response time analysis, implementation of time sensitive components, and consistent data logging.

The next generation KCU is going to use the open hardware UDOO board. This board has two processors, one processor is an ARM-A9 (MPU) processor able to run a complete Linux distribution including and X server, and the second processor is a Cortex-M3 microcontroller (MCU).

The controller code was divided into two parts. The core or real time controller, and the supervisor. The real time controller runs on top of the M3 processor and the supervisor on the A9 processor. The communication between both processors is done trough a UART port.

The motor driver is an off the shelf (OTS) controller made by the same company that manufactures the motors, this simplifies the prototype construction. This driver contains the power stage to drive the motor, a position controller, a setpoint generator and also performs the data acquisition from the encoder and hall sensors including the signal conditioning. Figure 5.1 shows the hardware used in the system.

This chapter explains the implementation of the real time controller.

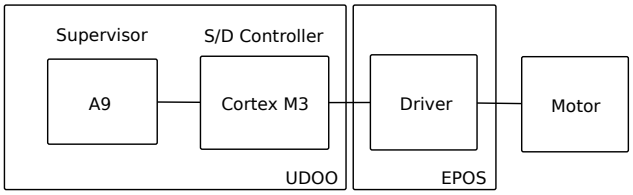

Figure 5.1: Overview of the system hardware components.

5.2 Timing Considerations

The controller designed in chapter 4 contains three nested control loops: The current, speed and position loop. We can run these three loops inside the OTS driver hardware, or we can run the loops in a separate controller and use only the driver's current loop.

Running the control loops outside the driver's hardware allow us to include feedforward control and our own current limiter, but we have to guarantee that we are able to communicate with the driver fast enough to keep the system stable. In other words: each loop has to run faster than the dynamic system is has to control. The electrical time constant for the motor is calculated using the RL circuit of the winding, its value is 0.4 ms. The mechanical time constant is approximately 3 ms, it was calculated using the response of the system given by equation 3.1. The OTS driver execution times are shown in Figure 5.1.

Table 5.1: OTS Controller loop times.

Loop	Frequency
PWM	57 MHz
Current	10 kHz
Position	1 kHz
Trajectory generator	1 kHz

As explained in section 5.4.2, reading a single motor state (position, speed, or current) takes approximately 2.5 ms, that means that if we want to run the complete position controller on the real time MCU, it will take up to 10 ms to get the three feedback values and write the output back. This is more than three times larger than the system mechanical constant. Hence, it was chosen to run the complete position controller and current limiter inside the EPOS controller.

5.3 Real Time Operating System

The real time controller was implemented using a real time operating system (RTOS) to facilitate future modifications by the KitePower team and reduce the entry barrier for students who are not from embedded systems.

Selection

There are several RTOS suitable to be run on the M3 core. The main criteria for the selection of the RTOS was the price, support, documentation, and portability. While professional RTOS provide good support for certification purposes, they are also expensive, and free RTOS are normally lacking in

support or maturity. A middle ground was achieved by selecting FreeRTOS because of the following criteria:

- The upfront development cost is zero.
- It is actively developed and has been for a long time.
- Good community support.
- It is well documented.
- It has been ported to several processors.
- It is open source (GPL).
- If needed an OpenRTOS license can be purchased to receive paid support.
- Already ported to Arduino DUE boards, so no board support package has to be written.
- The project can be ported later to SafeRTOS[1] without major changes.

Software Stack

There are at least two ways to use FreeRTOS on the UDOO board. One is to use FreeRTOS as an Arduino library, and the second one is to dismiss the Arduino framework and use the Atmel[2] software framework (ASF) directly.

Using the Arduino library makes the development easier for new students, but as more control over hardware is required, more hacks are needed to be able to compile the Arduino framework with the FreeRTOS kernel. This will complicate the project down the road and require the system to be redesigned around the needed hacks and incompatibilities. This is the reason why this option was not used.

The ASF is harder to use as each peripheral have to be configured and powered up individually, hence a more detailed knowledge of hardware is required, but in the long run the system will suffer less compatibility issues, and because the ASF supports several processors, the porting of the controller to a different processor not supported by the Arduino framework (as expected) won't require a complete rewrite of the code.

Because the UDOO board is compatible with the Arduino DUE processor and pinout, the same board support package (BSP) was used. The only modification needed was the creation of the `__bss_start__` and `__bss_end__` symbols in the linker script to simplify the build process.

[1] SafeRTOS is based on the functional model of FreeRTOS but it allows safety certifications.

[2] Atmel is the M3 processor manufacturer.

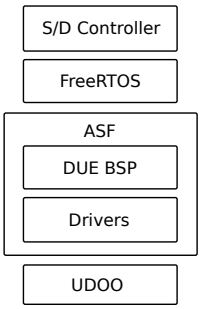

Figure 5.2: The S/D controller runs on top of FreeRTOS, which uses the ASF framework to have access to the UDOO hardware.

Figure 5.2 shows the resulting software stack needed to run the controller (with a *very* simplified ASF model). The drivers module of the ASF framework control the different parts of the processor (serial communications, DMA channels, power management, etc) and the BSP defines how the processor is wired on the board. The toolchain used is based on gcc 4.6.3.

5.4 Tasks

FreeRTOS was configured to use preemption, run the scheduler every millisecond and protect the stack with memory canaries. In case of a buffer overflow or if memory allocation fails, an error is reported to the supervisor.

The real time system was separated into four different tasks. Except the heartbeat task, these tasks are explained in the following subsections.

Controller Task: Reads the sensors, estimates the temperature and line tension, calculates the desired position settings for the motor, and send them to the driver.

Communication Task: Executes the command from the supervisor and reports back the result.

State Report Task: Sends the actual controller state to the supervisor.

Heartbeat Task: Toggles a led on and off to signal that the processor is not blocked.

5.4.1 Controller

The controller task reads the motor state, estimates the line tension, the motor temperature and the actual drum radius, it generates the force limiter correction term, and sets the desired position of the motors that correspond to the desired steering and corrected depower settings. All the calculations

have to be performed in fixed point arithmetic as the M3 does not support floating point arithmetic. The current limiter used is the one built into the motor driver; the tension limiter is a direct translation of the model explained in section 4.2 and its code is listed in section B.3. The details of the implementation of the estimators are presented below.

Temperature Estimation

The coding of the temperature observer poses the problem that it is a stiff system. The winding time constant is of order of 10 seconds, but the housing time constant is of the order of 30 minutes. So the winding heats up 180 times faster than the housing.

If the observer is discretized with a small sample time, the poles associated with the Housing temperature (the slow part) become too close to the unit circle. Because the controller processor does not have floating point arithmetic this number has to be truncated and translated into a fixed point representation. If the number is truncated and rounded down, the system converges to a lower temperature which renders the temperature monitoring useless; if the representation is truncated and rounded up, then the pole becomes one and the system becomes unstable.

If the system is discretized with a big sample time, the short changes in temperature are not observed and damage to the winding will occur. The chosen solution is to code the thermal model in two parts, one with fast dynamics, and one with slow dynamics. The winding temperature is then calculated every 100ms and the housing temperature and mutual effects every 10 s. The final code is listed in the appendix B.1

Tension Estimation

Tension estimation is done using equation 4.2. However, the system inertia is too small (9.4×10^{-5}) and the term αJ can be dropped from the equation. The expression for the line tension referred to the motor side of the gear would be:

$$T_\mathrm{l} - T_\mathrm{m} = 0 \tag{5.1}$$

The real problem in the implementation is to find the load torque referred to the drum side of the gear. Fortunately the gear model in section 3.2.2 was formulated in terms of the motor current. By arithmetic manipulation we can arrive at the estimator formulation:

$$T_\mathrm{l}(I,\omega) = \begin{cases} \frac{1}{b_1} I - \frac{a_1}{b_1}\omega - \frac{c_1}{b_1} & \omega < 0 \\ \frac{1}{b_2} I - \frac{a_1}{b_2}\omega - \frac{c_1}{b_2} & \omega > 0 \end{cases} \tag{5.2}$$

To summarize, the motor current and speed are used to find the torque on the gear output. The tension on the line would be T_l/r where r is the effective

radius of the drum, and the estimation is not updated if $w < 40$ rpm. The final code is listed in the appendix B.2

5.4.2 Communication

The communication task receives the user commands, interprets them, and sends them to the OTS driver. Even though the name is "communication task", it is not the only task using the communication channels. The communication with the driver is shared with the control task and protected with a mutex. The communication with the supervisor is shared with the state report task and is also protected with a mutex [39]. Both mutexes have priority inheritance to prevent priority inversions.

Controller - Motor Driver

The driver offers several communication options: Analog, digital, RS232 and CAN. Analog and digital inputs can not be used exclusively because they do not support the configuration of the driver and they need extra hardware to be used. The communication protocol chosen was RS232 because it is simple and fast to implement despite CAN being in theory almost 9 times faster[3].

The communication with the motor driver is defined by the manufacturer. In general, the driver works only as slave and will not initiate a communication on its own. This simplifies the communication since it is known when to expect data from the driver, and there is no need to keep polling for data.

Each communication frame starts with a handshake, then the data is sent including its CRC and a final acknowledge is received if the frame CRC matches the data received. A complete explanation of this communication protocol can be found in the driver communication guide [29].

All the communication functions are non-blocking to prevent the processor from locking. If communication is not possible a timeout error is raised and an emergency procedure can be performed, like restarting the driver.

At hardware level, this communication is done with the USART0 port at 115200 bits/s. The shortest request to the driver and its response (24 bytes total) takes approximately 2.5 ms to complete. Dropped packages are not resend but reported to the supervisor for an action to be taken.

Controller - Supervisor

The communication with the supervisor uses a serial port but a simpler protocol than the communication with the driver. The supervisor sends a command to the S/D controller and the S/D controller responds with a message containing either the requested data, an acknowledge, or an error

[3]CAN has its own problems, like the small packet size compared to RS232

code. This way, all the errors that occur in the motor driver are able to reach the supervisor and they can be logged.

The frames received and sent from the supervisor to the controller have different structures. Both frames are composed by a length field, a data field and a CRC field. The difference is that the frames sent *to* the supervisor include a start of frame field, and the frames sent *from* the supervisor include a end of frame field.

The end of frame marker of the supervisor commands triggers the release of a semaphore by the RX buffer interruption and the execution of the command is scheduled in the controller for the next task run. This means that even though the communication task waits for a semaphore before running a command, it is not an aperiodic task, it is a periodic task. This simplifies the worst case scenario analysis.

The communication with the supervisor is done through the UART0 port at 115200 bits/s. In case there is an error in the communication the supervisor continues reading the data stream until the next start byte is found and a new frame is received. The controller will discard all the bytes in the buffer and wait for the next end of frame marker to parse the next command.

5.4.3 State Report

The state report task sends the motor position, speed, current, temperature and estimated tension to the supervisor to be displayed on the GUI. This GUI can not be updated more than 10 times per second or it will halt, hence the state report is sent every 100 ms. This task runs independent of the communication task, the reason behind this is that even if the RX line is broken, the supervisor can still receive the data and log it without the need to request it.

5.5 Supervisor

The supervisor is a user interface application to access the S/D controller library from within the MPU. It was built for the purpose of testing the controller prototype and running the experiments, it is not part of the final KCU flight system because the communication with the S/D controller is done by the trajectory controller in the real system.

The supervisor simulates a command line to receive user inputs, displays the actual controller state, logs information into the SD card, and calculates the depower and steering setpoints in qc units. The qc units are quadrature counts, they are the number of steps registered by the encoder, in our case 2000 qc is a motor revolution. The reason why the translation from depower and steering normalized units to qc units is done in the supervisor instead of the S/D controller is because of the floating point support in the MPU.

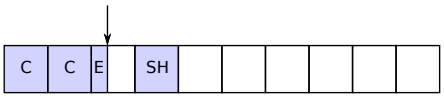

Figure 5.3: Task Scheduling with 100ms hyperperiod.

5.6 Summary

The S/D controller is constantly running the heartbeat (H), control (C), state report (S) and the communication task (E). The communication task executes the user commands but will not run until a end of the command is received.

The worst case scenario for a control command is to arrive at the S/D controller after the communication task just finished as shown by the arrow in Figure 5.3. In the figure the heartbeat and send state tasks were started with significant phases, otherwise the send state task will preempt the communication task and reduce the worst case time.

In this scenario a complete hyper period has to be waited to parse, execute the command, and then wait for the next control task to apply the correct setting. Thus, the command takes 196.8 ms to reach the driver[4]. This time can be lowered to 96.8 ms by using a hyperperiod of 50 ms. If the hyperperiod is modified, only the coefficients of the temperature estimator have to be recalculated (the tension estimator does not depend on time).

The communication with the EPOS controller is protected with a mutex. This can affect the update time of the controller state and logging. The maximum time this mutex can be hold by the communication task is 2.5 ms, hence the state samples and logging can have a 2.5 ms jitter time.

The calculations done by the controller are so simple that all the time is spent in communication routines. If it is possible to reduce the communication time by half using CAN, then it would be possible to reduce the controller hyperperiod to 20 ms.

Table 5.2 shows the summary of the S/D controller tasks and the resource use for a hyperperiod of 100 ms.

[4]This analysis assumes the supervisor is not sending more commands before the current one is completed, otherwise the USART interruption will add 0.625 ms more to the execution time.

Table 5.2: Task Properties

Task	Execution Time	Priority[a]	Period	Stack[b]
Control	19.7 ms	1	100 ms	200
State	9.4 ms	2	100 ms	130
Communication	3.2 ms	3	100 ms	500
Heartbeat	1.4 us	4	100 ms	500
Comsuption	33.7%	-	-	16.2%

[a] Priority values are inverted in the M3 architecture.
[b] In bytes.

Chapter 6

Results

This chapter presents the results of the KCU model simulations using MATLAB , and results of the temperature and load estimators implemented in the UDOO board. Unfortunately the new KCU is not built and the controller could not be tested in flight with a real kite.

6.1 KCU Model Test

The MATLAB model of the system and the controller were tested with data generated by the FreeKiteSim (FKS) simulation software. This software simulates a complete wind field (3D), the kite response, and the desired control action to keep the kite on the planned trajectory. Our model uses FKS generated apparent wind (including turbulence), and its trajectory controller depower setting and steering setting as inputs. A brake signal is generated manually when needed as it exists on the current prototype but not in FKS. In appendix A the physical characteristics of the motor, gear, and drum are listed. This data corresponds to the specific components models that are expected to be used in the new KCU.

The total kite depower tape range is 6 meters and ± 1.8 meters for the steering tape. The kite model is a 8000 N leading edge kite with 16.5 m^2 projected area. A typical KCU control input to perform a single pumping cycle is shown in Figure 6.1. During normal operation the kite is never completely powered or depowered, the steering signal used is between -0.4 and 0.4, and the brake is engaged when the its value is 1 and off when it is 0. The brake is only used at the depower motor and is activated once it reaches the preset powered or depowered states.

6.1.1 Force Limitation

Both methods explained in section 4.2 are simulated here to observe the effects on the kite tape tension.

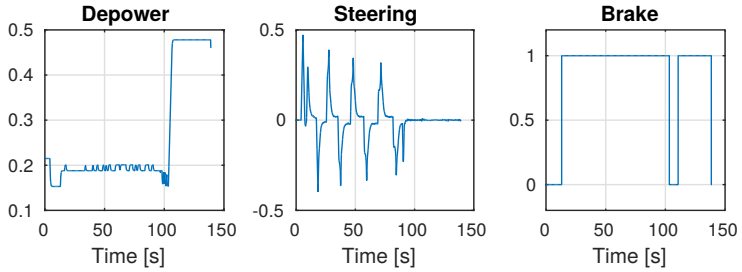

Figure 6.1: Controller's inputs for a single pumping cycle.

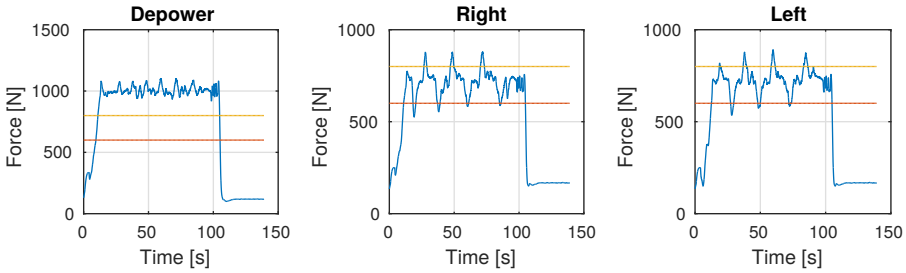

Figure 6.2: KCU tapes overload by a wind speed of 35 m/s. The steering tapes reach values over 800 [N].

No Limited Tension

The tensions on the control tapes during flight are affected mainly by the apparent wind speed and the kite angle of attack. During normal operation the ground station controller keeps the tension on the tether low by reeling out the tether. We assume that the ground station is not controlling the tether force and the kite can be overloaded by sudden wind gusts or a permanent increase of the wind speed. The simulation in Figure 6.2 shows the overload on the control tapes caused by a continuous high apparent wind of 35 m/s. The tapes tensions are maximum 600 N under normal operation and should never reach more than 800 N to prevent them to snap or get damaged (they deform under tension and can get tangled inside the drums).

Effects of controlling the force by load balancing

The load balancing method consists on redistributing the forces on the lines by diminishing the steering motor action. The simulations in Figure 6.3 show that redistributing the loads does not reduce the tension enough to reach a safe value even though it prevents the steering tapes tensions from reaching the hard limit of 800 N. Additionally it can be observed that the steering capabilities are severely degraded and completely lost after the 50s mark (the forces on the left and right tapes are the same).

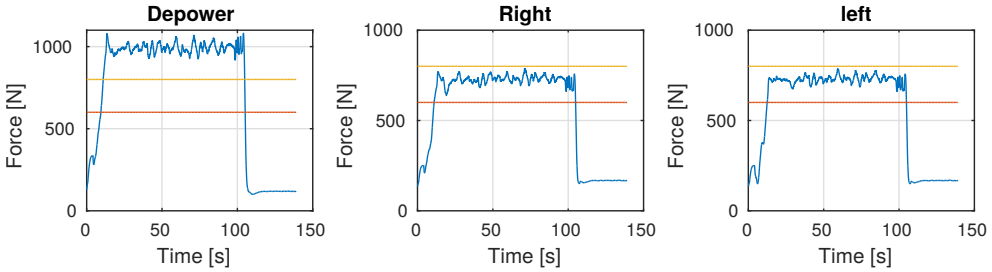

Figure 6.3: Load redistribution on the control tapes. The tension on the steering tapes is decreased below 800 N but not below 600 N. The depower tape is unaffected.

Force limitation by Depowering

This method works by reeling out the depowering tape. There are two ways this method affects the tension on the tapes. The first effect depends on the mechanics of the motor. If the motor inertia is omitted (it is very small), equation 4.2 can be formulated as:

$$T_l = T_m = IK_m \tag{6.1}$$

Hence, the torque exerted by the tape is equal to the motor torque. This torque has a physical limit, after which the force control is lost but the tape will continue to be reeled out and reduce the angle of attack. Keep in mind that if the tension on the lines is bigger than the maximum motor torque, the depower subsystem becomes uncontrollable but not unstable. This is the second effect of reeling the depower tape, the actual depowering of the kite.

Tests Two scenarios were investigated to test the force limitation by depowering: The response of the system to a sudden wind gust and the limitation of forces during two consecutive pumping cycles under high wind speeds.

The discrete wind gust simulation tests the response time of the controller under harsher conditions than normal operation. To make it easy to observe the controller response, the desired depower setting was set to 0.22. The wind speed was recreated as explained in section 3.3. The results including the apparent wind are shown in Figure 6.4

It can be seen that without the limiter the tension on the depower tape reaches almost 900 N (and stays high) and the steering tapes reach more than 600 N. On the other hand the limiter is able to keep the tension of the lines below 700 N at any time and below 600 N at the end. It can be also observed that the depower override is incremented each time the tension goes over 600 N and is maintained even when the tensions goes below 600 N.

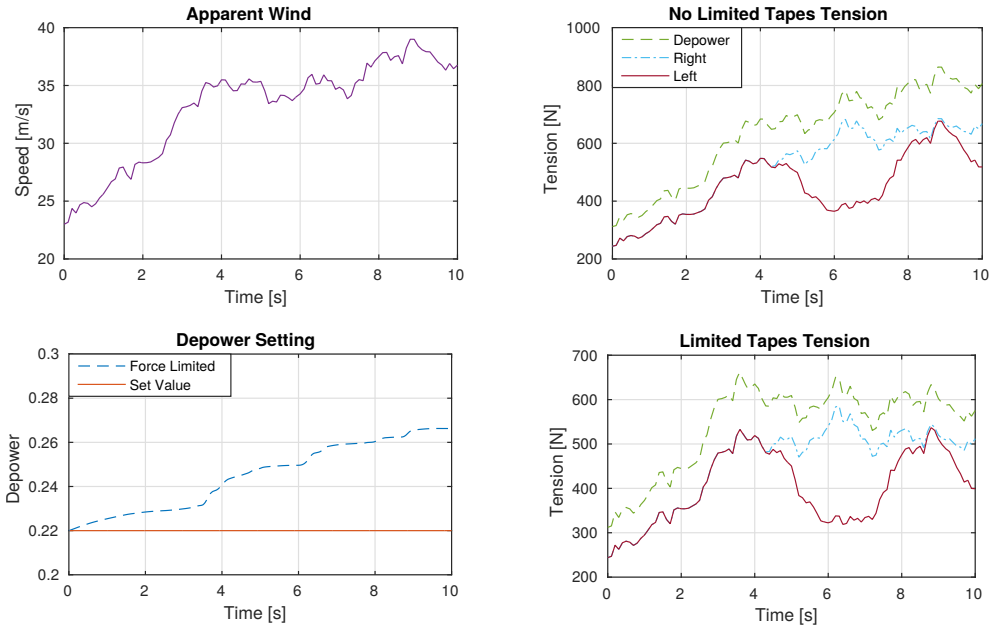

Figure 6.4: Wind gust response with and without tension limitation.

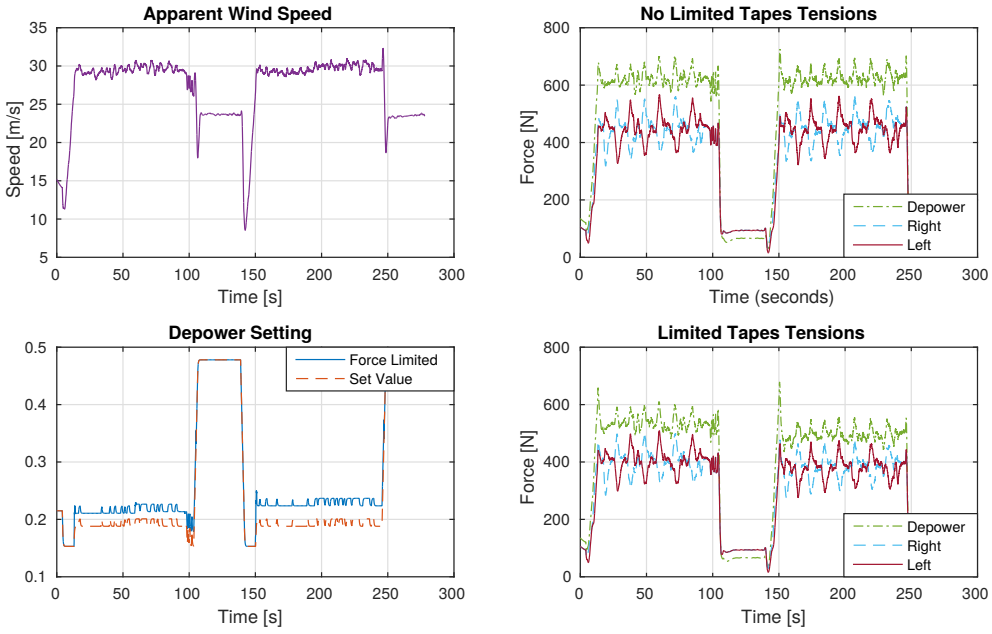

Figure 6.5: Operating under high wind speed conditions with and without tension limitation.

This is done to prevent oscillation and going over the maximum tension again during the same pumping cycle.

Figure 6.5 shows the force limitation using the depowering tape under high wind conditions. For this simulation, the apparent wind speed was increased to 30 m/s but the desired depower setting was kept as if the pilot was not aware that the kite is flying under higher wind speeds.

It is observed that the limiter overrides the depower setting and is able to keep the forces on the lines below 600 N during both pumping cycles. Also, the operation tensions are higher during the first cycle than during the second cycle because the integral buffer is discharged after each pumping cycle and the second pumping cycle started with higher forces which made the controller operates at lower tensions to prevent overloads.

6.1.2 Temperature

The kite controller should by safe enough not to overheat and damage the motors by excessive action. However, a constant change in depower setting to control the forces on the tapes could prevent the use of a brake and overheat the depower motor.

To investigate the system reliability a simulation of a single pumping cycle was simulated. The current consumed to control the kite was obtained and used as an input to the thermal model of the motors to find the thermal steady state of the kite.

Figure 6.6 shows three pumping cycle scenarios and the shared controller inputs; normal operation (left), overload operation (center), and overload operation with tension limitation (right). The normal scenario apparent wind is the same shown in Figure 6.5 and the overload scenarios were created by adding 5 m/s to the normal scenario apparent wind. Below each force graph the steady state temperature is obtained by simulating the system for 3.7 hours (90 pumping cycles) with an ambient temperature of 25°C. During these simulations the brake is activated manually during 107 s for the first two scenarios, and 108 s for the limited scenario (the tension limiter is activated during one extra second at 36 s).

It can be observed that during normal operation the depower motor winding reaches a peak of 51°C and the steering motor winding 27°C. During overload operation (provided the tapes do not snap) the temperatures raise to 73°C and 30°C because the motor has to overcome a higher load on the tapes to be able to power and depower the kite in a controlled way.

What is notable is that the limiter does not increase the temperature of the motors by a significant amount; the depower winding reaches a peak of 54°C and the steering a peak of 28°C. This means that the additional limiter action does not pose a problem is terms of overheating the motor. Instead, it helps keeping the temperatures low by preventing the kite to reach overload states where the motor will need more energy.

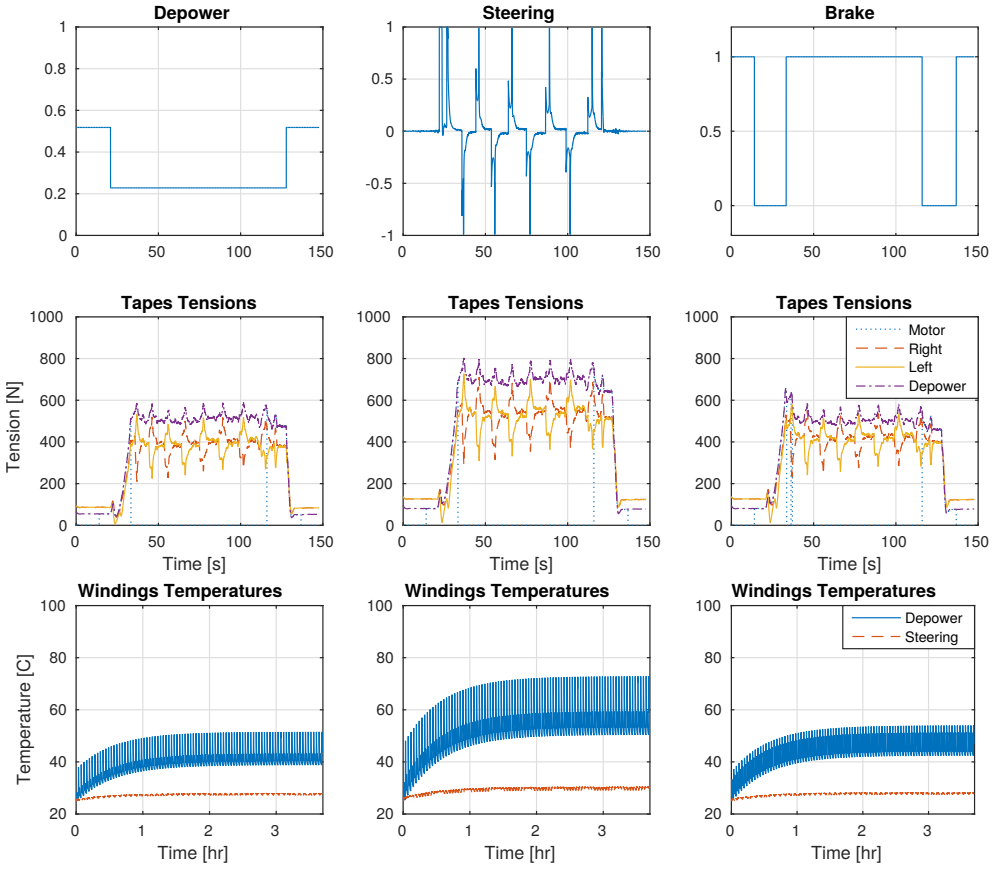

Figure 6.6: Steady state temperature simulation. Controller inputs (top), Normal scenario (left), overload scenario without tension limiter (center), overload scenario with tension limiter (right).

There is however one problem with the brake of the depower motor; it makes the motor "blind". In the tension limited scenario (right) the limiter is not triggered when the brake is engaged even if the depower tape reaches more than 600 N. This was solved by lowering the steering tapes limits to 550 N, this is the reason why the limiter is triggered at 36 s. An additional problem is the discharge of the integral buffer, when the brake is engaged the tension as seen by the motor falls below the reset threshold, to solve this, the buffers only reset when the three tapes tensions fall below 300 N.

6.1.3 Energy

The model's energy consumption was calculated using the kite's mechanical power output and input while the motor is operating, and the system efficiency data. It was assumed that the brake was engaged as soon as the final tape length is reached and the speed is zero. It is relevant to note that the tension on the lines are mostly above 100 N when the motor is operating, and as shown in Figure 3.3 the efficiency for these loads are between 60% and 40% for high speeds while powering the kite, and between 60% and 20% when depowering the kite. By using an average system efficiency of 50% for the powering phase and 40% for the depowering phase, the estimated system average power for the normal scenario is $\approx$ 7.9 W, or 26% of what Braun [9] estimated from experimental data for the current prototype (30 W).

There are a couple of hypothesis on why the simulation energy consumption results are below the experimental measurements. As expressed by Braun, the 30 W estimation can be lowered if the kite is flown in automatic mode, which is the case for the simulated results. Furthermore, even though the current prototype is a 4000 N system and the new one is a 8000 N system, the simulated kite is operating below 6000 N.

There is also a difference in the cycle of the pumping cycle, the simulation data includes eight figure of eight flights per pumping cycle, while it is very probable that the experimental data only made four figure of eight per pumping cycle, hence the experimental data includes additional power and depower actions increasing the power consumption per pumping cycle. And finally, there could be factors not accounted for, like the KCU case temperature which lowers the system efficiency, and sensor noise that triggers steering corrections. Hence, the energy results of the simulation are used only to compare the energy consumption of the test scenarios but can not be compared directly with the experimental data.

In terms of energy the overload scenario consumes 43% more power than the normal scenario and the limited scenario 22%. Hence the limiter cost is relatively low and it can help reduce the amount of energy needed to control the kite.

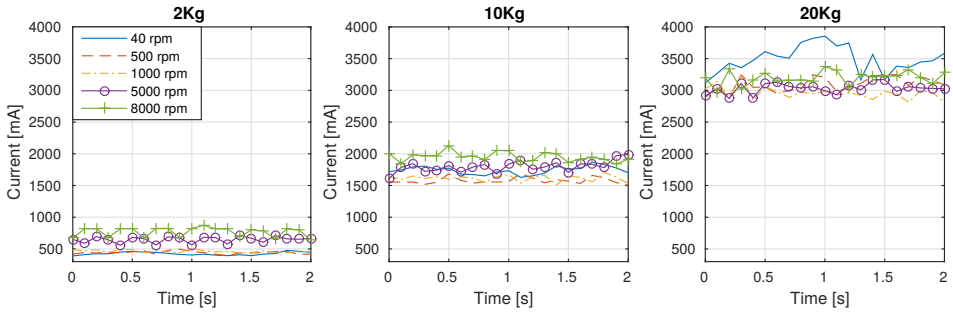

Figure 6.7: Current at different motor speeds and different loads.

6.2 Partial Implementation

A partial implementation was done in a UDOO board using C and FreeRTOS connected to a Maxon EPOS2 50/5 driver. The controller was energized by a 600 W 45 V 10 A power supply and a constant current load was included to consume the energy flowing back from the motor when a sudden stop occurred (which would cause an overvoltage at the power supply and trigger the protections otherwise). The total depower motion range was set to 2 m.

6.2.1 Load Estimation Test

To test the load estimator masses of 2 Kg, 4 Kg, 10 Kg, 14 Kg and 20 Kg were suspended from the tape attached to the drum. The controller was configured to maintain a constant speed and the system state was recorded every 100 ms. Figure 6.7 shows two seconds of recording of the current consumed by the motor to lift 2 kg 10 kg and 20 kg loads at different constant speeds and at an initial drum radius of 34 mm. In this figure, the motor torque is driving the gear. Similar curves are obtained if the load drives the gear.

It can be seen that the load is a bigger factor than the speed when determining the motor current. A steady current is expected as the load is moving at a constant speed and the drum radius does not change significantly in 2 seconds (it is increased by 1.7% at most at 8000 rpm). Excluding sensor noise there are several factors why the current values change so much: The position of the motor poles affects the motor current[1]. The load inertia induces oscillations on the torque. The drum imperfections like the tape attachment point, seams, tape folding and misalignment affect the effective drum radius and hence the current.

Table 6.1 shows the results of the load estimation tests. These results are an average over the different speeds tested. At lighter and heavier loads the estimated load error increases because of the linear curve fitting assumed.

[1]This is the biggest factor observed in the current signal Fourier analysis

Table 6.1: Load estimation results.

$w > 0$				
Load [kg]	Avg Estimation [kg]	Avg Error [%]	RMSE [kg]	RMSE [%]
2	1.95	2.50	0.24	12.31
4	4.01	0.25	0.29	7.23
10	9.94	0.60	0.48	4.83
14	13.75	1.78	0.61	4.44
20	19.44	2.80	0.82	4.22
$w < 0$				
Load [kg]	Avg Estimation [kg]	Avg Error [%]	RMSE [kg]	RMSE [%]
2	1.87	6.50	0.30	16.04
4	4.16	4.00	0.37	8.89
10	9.92	0.80	0.49	4.94
14	14.51	3.64	0.57	4.07
20	20.74	3.70	0.76	3.66

However, the average error is less than 7% in all cases. It can also be observed that the RMSE value in kilograms increases with higher loads, but it decreases when normalized (coefficient of variation). In summary, the accuracy of the estimator is degraded if the load is too light or too heavy, and the precision is decreased with heavier loads. If the estimator is used only as input to the tension limiter, then it is possible to increase its accuracy of the curve fitting by assigning more weight to values near the tension limits.

6.2.2 Motor Thermal Model Validation

These tests were performed with the controller configured in constant current mode and the motor rotor blocked, hence friction and magnetic loses were not taked into account. Currents of 1 A, 2 A, 3 A and 4 A were used to compare the estimated housing temperature with the real one. The housing temperature was measured with a BaseTech IRT-350 IR thermometer with a 2°C accuracy. Because the motor surface is unpainted (shiny) the thermometer is not able to correctly read the temperature, hence an opaque 3cm×2cm thin tape was added on the surface to perform the measurements. The obtained results are shown in Figure 6.8 and the RMSE values are listed in Table 6.2.

The results show that the thermal model is a good approximation, and that separating the model into two models at different update rates does not have a huge impact on the temperature estimation. In general, the predicted

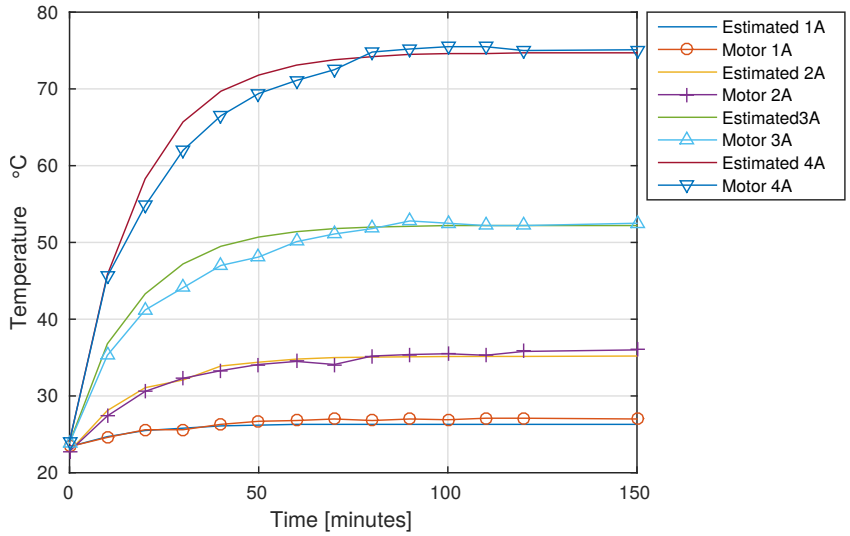

Figure 6.8: Measured temperature values vs estimated temperature values.

temperature raises faster than the real one (this is not a problem for the motor integrity), but at steady state the motor temperature is higher than the estimated one (by less than a degree C).

The observed error can be caused by the rounding of the model coefficients, the added tape (including the glue) resistance, the neglected friction loses, and the neglected magnetic loses. It is necessary to add a security factor to the protection implementation to account for the neglected loses.

Table 6.2: Temperature results root mean square error values.

Current [mA]	RMSE [°C]
1000	0.53
2000	0.50
3000	1.52
4000	1.90

6.2.3 Tension Limitation

As it is not possible to test the real time implementation in a real system, the controller was tested by applying a reduced load to the drum at the laboratory. The controller was configured to maintain the tape tension below 120 N and reset the override at 20 N, the target depower state given was 0.5.

The results are shown in Figure 6.9. It can be seen that the tension is

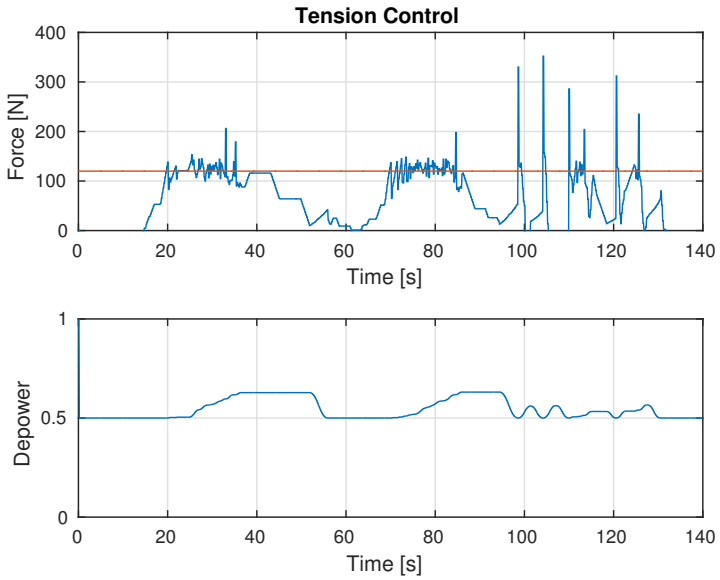

Figure 6.9: Implemented tension limiter.

successfully limited and maintained in the first 100 seconds. The motor acts as expected and increase the depower state to keep the tension down. However, several tension peaks appear after this time mark. The reason for this is that the tension controller is running too slow compared to the time the tension takes to raise. These peaks represent an increase of the load of more than 250N in less than a second which is more than the observed load increase in the kite in experimental data. Even then, the controller is able to reduce these peaks in a couple of control ticks. (200 ms – 300 ms).

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this work a method to estimate the tension on the S/D lines using the current and speed of the motor was demonstrated. The average error of the measured value of the tension estimator is below 7% in the range it was designed and tested (19 N-200 N). The method relies on the correct characterization of the drive train and can be used to limit the forces on the tapes.

At very low and very high speeds the estimation is degraded by the linear curve fittings assumption. A especial case is the zero speed. When the drum is not moving, the tension on the lines can not be estimated accurately as the static friction force is undetermined, nonetheless, the tension at which the tension limiter should actuate is bigger than the maximum static friction and the estimator can be used as input to the limitation algorithm.

If a brake is used on the depower tape to save energy the motor loses its ability to sense the tension on the tape. Fortunately, the high tensions are present mostly during the reel-out phase and the steering lines (which do not have brakes) can be used to estimate the forces.

The proposed limiter algorithm output is used as an input of the outer position loop allowing us to use an OTS controller simplifying the construction of a prototype. The limiter has an internal integrator that prevents future overload of the tapes during the pumping cycle. This reduces the amount of corrections the limiter has to do and enables the KCU to use a brake and hence have a reduced impact on the power consumption.

The effects of using the depower and steering motors to control the tension are explained. Balancing the forces on the lines by reducing the steering of the kite is not sufficient and degrades the maneuverability of the kite. Reducing the forces on the tapes by depowering the kite is effective, but it

needs to act before reaching the hard limit.

An electromechanical model on the KCU is presented. It permits the evaluation of different control algorithms in terms of power consumption, response times, actuator limitations, and can be useful for the selection of the actuators of the KCU for future prototypes.

A viable real time platform to perform research and development of the next prototype is proposed. It allows good error reporting and real time logging capabilities. The designed estimators and limiters were implemented and demonstrated to be fast enough to run on a real time system at 50ms. The actual execution time bottleneck is the communication protocol used.

7.2 Future Work

Future work on the S/D controller can be aimed at optimization of the current design or adding more functionality to it. The communication between the controller and the OTS driver should be improved as it is the current execution time bottleneck. The use of CAN as the general communication protocol and using the digital outputs for state updates could be a good option to speed up the control loop although extra hardware is needed to do this.

There is a new iteration of the board family we are using to implement the S/D controller, it is called the UDOO Neo and it has an A9 and a M4 processor on the same die. The M4 processor supports floating point arithmetics, which could make possible to port the trajectory controller onto the real time platform and improve the estimator by doing a better curve fitting. It would also make it possible to use a faster communication protocol (AXI4) to reduce the communication time drastically.[1]

After the new KCU is constructed, the controller should be tested during flights to evaluate the model accuracy and to fine tune the controller. It is necessary to measure the forces on the KCU tapes to validate the results given by Bosch and the force distribution model used in this thesis as currently no experimental measurements are available.

[1] At the time of writing this thesis this board is only available to Kickstarter early backers, but should be available to the general public in a couple of months.

Appendix A

Components Characteristics

In this appendix the basic data of the system components is presented

A.1 Motor

Table A.1: Motor data

Property	Symbol	Value	Units
Brand		Maxon	
Reference		393024	
Volts	U_n	42	V
Power	P	170	W
No Load Speed	n_0	10100	rpm
No Load Current	I_0	230	A
Nominal Speed	n_N	9380	rpm
Nominal Torque	M_N	161e-3	Nm
Nominal Current	I_N	4.24	A
Stall Torque	M_H	2740e-3	Nm
Starting Current	I_A	69.1	A
Line Resistance	R	0.608	omh
Line Inductance	L	246e-6	H
Torque Constant	K_M	39.6e-3	Nm/A
Speed Constant	K_N	241	rpm/V
Rotor Inertia	J_R	53.8	g cm^2
Thermal resistance housing-ambient	R_h	5.21	K/W
Thermal resistance winding-housing	R_w	1.05	K/W
Thermal time constant winding	τ_w	18.7	s
Thermal time constant motor	τ_h	1910	s
Max. permissible winding temperature	$T_{w,max}$	155	°C
Max. Efficiency	η_{max}	89	%

A.2 Gear

Table A.2: Gear data

Property	Value	Units
Brand	Maxon	
Reference	223092	
Reduction	74:1	
Reduction Absolute	147/2	
Mass Inertia	1.72×10^{-6}	kg m^2
Max Motor Shaft Diameter	10e-3	m
Max Efficiency	75	%

A.3 Pulley

Table A.3: Pulley data

Property	Value	Units
Radius	25e-3	m
Inertia	9.37e-5	kg m

Appendix B

Code

B.1 Temperature Estimator

```
/* @brief Calculate the motor temperatures
 *
 * This functions runs a motor model to predict the motor
 * temperature, it can be corrected with a sensor reading
 * using the obserber gain.
 *
 * @note This observer was designed to run at 100ms, if this
 * period changes the observer coefficients have to be
 * recalculated.
 */
void ctrl_temp_tick(void)
{
    /* Ticks counter */
    static int32_t i = 0;

    /* Motor tempertures*/
    int32_t temps[2] = {0, 0};

    /* State space coefficients */
    int32_t A1 = 4074;          /* Q12 */
    int32_t B1[2] = {22, 23};   /* Q12 */
    int32_t A2 = 3970;          /* Q12 */
    int32_t B2 = 105;           /* Q12 */

    /* Luenberger Gain */
    int32_t L[2] = {0, 2};      /* Q12 */
    int32_t current;            /* Q0 [mA] */
    int32_t power;

    /* Sensor variables */
    //int32_t snr_temps_reading[2] = {0, 0};  /* Q24 */

    current = ctrl_status.current;
```

```
    temps[0] = ctrl_status.temp_wnd;   /* Q24 */
    temps[1] = ctrl_status.temp_hou;   /* Q24 */

    /* Observer model */
    power = current;
    power = ((power * power) >> 8);   /* Q-8 */
    /* Q17 = Q-8 * Q25 = power * 0.608 * 1e-3 * 1e-3 * 2^25 */
    power = power * 20.401;
    power = power >> 5;   /* Q12 */

    temps[0] = A1 * (temps[0] >> 12) + B1[0] * (temps[1] >> 12);
    temps[0] = temps[0] + B1[1] * power;

    /* Evaluate the housing model only every 100 ticks */
    if ((i % 100) == 0) {
        temps[1] = A2 * (temps[1] >> 12) + B2 * (temps[0] >> 12);
        temps[1] = A2 * (temps[1] >> 12) + B2 * (temps[0] >> 12);
        //temps[1] += L[1] * ((snr_temps_reading[1] - temps[1]) ↩
            >> 12);
        i = 0;
    }
    ctrl_status.temp_wnd = temps[0];   /* Q24 */
    ctrl_status.temp_hou = temps[1];   /* Q24 */
    i++;
}
```

B.2 Tension Estimator

```
/**
 * @brief Estimate the line tension.
 *
 * @note Tensions are in Q10 format.
 */
void ctrl_force_tick(void)
{
    int32_t current;    /* Q0  Input current in mA */
    int32_t rpm;        /* Q0  Motor speed in rpms */
    int32_t r_eff;      /* Q0  Effective radius in mm */
    int32_t position;   /* Q0  Actual motor position in qc */
    int32_t f_line;     /* Line tension */

    /* Motor state */
    rpm = ctrl_status.speed;
    current = ctrl_status.current;
    position = ctrl_status.position;

    r_eff = ctrl_radius_eff(position);
    f_line = ctrl_status.f_line;

    /* Powertrain curve fitting */
```

```
    if (rpm > 40) {
        f_line = (4243 * current - 168 * rpm - 576431); /* Q16 */
        f_line = f_line >> 6;  /* Q10 */
        f_line = (ctrl_status.f_line * 38) / r_eff;  /* Q10 */

    } else  if (rpm < 40) {
        f_line = (6815 * current - 161 * rpm - 700320); /* Q16 */
        f_line = f_line >> 6;  /* Q10 */
        f_line = (f_line * 38) / r_eff;  /* Q10 */
    }
    ctrl_status.f_line = f_line;  /* Q10 */
}
```

B.3 Tension Limiter

```
/**
 * @brief Limit the forces on the lines.
 * @param[out] correction   Correction term in qc units.
 */
void ctrl_limit_force(int32_t *correction)
{
    const int32_t on_limit = 100;  /* Turn ON limit */
    const int32_t int_reset = 50;  /* Limiter reset */
    int32_t tension;               /* Line tension */
    int32_t p_action = 0;          /* Proportional action */
    static int32_t i_action = 0;   /* Integral action */
    int32_t i_gain = 50;           /* Integral gain */
    int32_t p_gain = 30;           /* Proportional gain */

    tension = ctrl_status.f_line >> 10;  /* Q0 */

    if (tension > on_limit) {
        p_action = (tension - on_limit) * p_gain;
        i_action = i_action + (tension - on_limit) * i_gain;
    }

    if (tension < int_reset) {
        i_action = 0;
    }

    *correction = p_action + i_action;
}
```

Bibliography

[1] European Aviation Safety Agency. Airworthiness Certification of Unmanned Aircraft Systems (UAS), 2009.

[2] International Energy Agency. World energy outlook 2014: Executive summary. Paris, France, 2014.

[3] Cristina L. Archer. Evaluation of global wind power. *Journal of Geophysical Research*, 110(D12):1–20, 2005.

[4] J.H. Baayen. *Automatic trajectory tracking control of kites.* Msc thesis, TU Delft, Delft, 2011.

[5] J.H. Baayen and Wubbo Ockels. Tracking control with adaption of kites. Nov. 2011.

[6] Allert Bosch, Roland Schmehl, Paolo Tiso, and Daniel Rixen. Dynamic Nonlinear Aeroelastic Model of a Kite for Power Generation. *Journal of Guidance, Control, and Dynamics*, pages 1–11, 2014.

[7] H a Bosch. *Finite element analysis of a kite for power generation.* Msc thesis, TU Delft, 2012.

[8] El-Kébir Boukas and Fouad M Al-Sunni. *Mechatronic Systems.* Springer, 2012.

[9] Lukas Carl Braun. *Systematic Approach for a Redesign of a Kite Control Unit.* Msc thesis, Technical University Munich, 2015.

[10] Jeroen Breukels. *An Engineering Methodology for Kite Design.* Phd dissertation, Technische Universiteit Delft, 2011.

[11] Uwe Ahrens (ed), Moritz Diehl (ed), and Roland Schmehl (ed). *Airborne Wind Energy.* Green Energy and Technology. Springer, 2013.

[12] Michael Erhard and Hans Strauch. Control of Towing Kites for Seagoing Vessels. *submitted to IEEE Control Systems Magazine*, pages 1–12, 2012.

[13] Michael Erhard and Hans Strauch. Flight control of tethered kites and winch control for autonomous airborne wind energy generation in pumping cycles. *International Journal of Control*, pages 1–18, 2014.

[14] Lorenzo Fagiano, Mario Milanese, and Dario Piga. Optimization of airborne wind energy generators. *International Journal of robust and Nonlinear Control*, 22(18):2055–2083, 2012.

[15] U Fechner. *A Methodology for the Control of Kite Power Systems.* Phd thesis, Delft University of Technology, 2015.

[16] Uwe Fechner, Rolf van der Vlugt, Edwin Schreuder, and Roland Schmehl. Dynamic model of a pumping kite power system. *Renewable Energy*, 83:705–716, nov 2015.

[17] Stacey Gage. Creating a unified graphical wind turbulence model from multiple specifications. *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (August), 2003.

[18] N.H. Geshiere. *Dynamic modelling of a flexible kite for power generation.* Msc thesis, Delft University of Technology, 2014.
[19] Erich Hau and Horst von Renouard. *The wind resource.* Springer, 2006.
[20] Rolf Isermann. *Mechatronic systems: fundamentals.* Springer Science & Business Media, 2007.
[21] Jonathan Ivan and Ramirez Gutierrez. *Airborne Wind Energy- Data-driven LPV Modeling for Flight Control of a Kite Power Airborne Wind Generator.* Master thesis, TU Eindhoven, 2013.
[22] Claudius Jehle and Roland Schmehl. Applied Tracking Control for Kite Power Systems. *Journal of Guidance, Control, and Dynamics*, pages 1–12, Feb. 2014.
[23] Bas Lansdorp and Wubbo Ockels. Comparison of concepts for high-altitude wind energy generation with ground based generator. *The 2nd China International Renewable Energy and Conference*, pages 1–9, 2005.
[24] L. Lois. Apparatus for extracting energy from winds at significant height above the surface, Dec. 9 1975. US Patent 3,924,827.
[25] M.L. Loyd. Crosswind kite power (for large-scale wind power production). *Journal of Energy*, 4(3):106–111, 1980.
[26] M.L. Loyd. Wind driven apparatus for power generation, Feb. 17 1981. US Patent 4,251,040.
[27] M. S. Manalis. Airborne Windmills: Energy source for communication aerostats. In *AIAA Lighter Than Air Technology Conference*, pages 75–923, 1975.
[28] D Moorhouse and R Woodcock. Us military specification mil–f–8785c. Technical report, Technical report. US Department of Defense, 1980.
[29] Maxon Motors. EPOS2: Communication guide, 2013.
[30] Wubbo J. Ockels. Laddermill, a novel concept to exploit the energy in the airspace. *Aircraft Design*, 4(2-3):81–97, 2001.
[31] P.R. Payne and C. McCutchen. Self-erecting windmill, Oct. 26 1976. US Patent 3,987,987.
[32] Christopher Pelchen, Christian Schweiger, and Martin Otter. Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes. *2nd International Modelica Conference*, 3:257–266, 2002.
[33] Pragasen Pillay and Ramu Krishnan. Application characteristics of permanent magnet synchronous and brushless DC motors for servo drives. *IEEE Transactions on Industry Applications*, 27(5):986–996, 1991.
[34] Federal Aviation Regulations. Title 14 — Aeronautics and Space. *US government Printing Office, Washington*, I, 2015.
[35] L. Rierson. *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance.* Taylor & Francis, 2013.
[36] Bryan W. Roberts, David H. Shepard, Ken Caldeira, M. Elizabeth Cannon, David G. Eccles, Albert J. Grenier, and Jonathan F. Freidin. Harnessing High-Altitude Wind Power. *IEEE Transactions on Energy Conversion*, 22(1):136–144, Mar. 2007.
[37] M B Ruppert. *Development and validation of a real time pumping kite model.* Msc thesis, Delft University of Technology, 2012.
[38] J.F.J.E.M Schwoll. *Finite Elements Analysis of Inflatable Structures Using Uniform Pressure.* Msc thesis, TU Delft, 2012.
[39] David E Simon. *An embedded software primer.* Addison-Wesley Professional, 1999.

[40] Military Standard. Flying qualities of piloted aircraft. Technical report, MIL-STD-1797A, Department of Defense, 2004.

[41] O. Stemme and P. Wolf. Principles and Properties of Highly Dynamic DC Miniature Motors, 1994.

[42] Kimon P Valavanis and George J Vachtsevanos. *Handbook of Unmanned Aerial Vehicles*. Springer, 2015.

[43] Paul Williams. Optimal Wind Power Extraction with a Tethered Kite. In *AIAA Guidance, Navigation, and Control Conference*, number August, Colorado, 2006.

[44] Paul Williams, Bas Lansdorp, and Wubbo Ockels. Flexible Tethered Kite with Moveable Attachment Points, Part I: Dynamics and Control. *Dynamics and Control*, 31(August):793–799, 2007.

[45] Paul Williams, Bas Lansdorp, Richard Ruiterkamp, Wubbo Ockels, and Applied Researcher. Modeling, Simulation, and Testing of Surf Kites for Power Generation. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, number August, Honolulu, Hawaii, 2008. Delft University of Technology, The Netherlands.