Delft University of Technology
Master’s Thesis in Computer Science

Design of a Fast and Reliable Wireless Link for Kite Power Systems

Antonio Ramos Salido Maurer

Design of a Fast and Reliable Wireless Link for Kite Power Systems

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

Antonio Ramos Salido Maurer
ramosmaurer@gmail.com

17th August 2012

Author

Antonio Ramos Salido Maurer (ramosmaurer@gmail.com)

Title

Design of a Fast and Reliable Wireless Link for Kite Power Systems

MSc presentation

28th August 2012

Graduation Committee

Prof. Dr. K. G. Langendoen	Delft University of Technology
Dr. Ir. E. Onur	Delft University of Technology
Dr. Ing. R. Schmehl	Delft University of Technology
MSc. Uwe Fechner	Delft University of Technology

Abstract

Kite Power Systems present technological advantages over traditional windmills. In this work, the design of a dual wireless link for kite applications requiring high reliability and low delays is presented. The link provides real-time data exchange between the system components that allows researchers to run closed-loop controllers, and to validate kite models. The proposed solution meets the quality of service requirements (delay and packet losses) and other requirements specific for airborne wind energy applications like small form factor, low weight, and power consumption. The most relevant design choices for frequencies, devices, and configuration are described. A pair of controllers for flying crosswind were tested under different QoS conditions with positive results. Performance measurements during flight were used to build statistical models. Such models were implemented in a simulator to better understand the effects of the network over autopilot controllers and to find the minimum communication performance requirement for the system.

Preface

The ASSET institute is an inter-faculty institute within TU Delft directed by Prof. Dr. Wubbo Ockels. MSc. Aart de Wachter and MSc. Rolf van der Vlugt are currently building a Laddermill Prototype with an output capacity of 25kW. Before moving forward to a bigger system they have the milestone to achieve 24 hours of continuous flight using an autopilot. A PhD. researcher MSc. Uwe Fechner is working on the development of a methodology for the design of optimal kite power control systems.

My motivation to work in this project is twofold. I am concerned about the world energy supply and demand and I agree with the development of alternative energy sources to reduce pollution and to move away from the dependency on fossil fuels. I am also interested in the development of Linux-based embedded systems with tight timing requirements that are related to control applications. My previous experience was with C based systems where a simple scheduler was enough for the application. In this case we are working with larger devices and using different software tools to speed up the development and build up a reliable system.

Antonio Ramos Salido Maurer

Delft, The Netherlands
17th August 2012

Contents

Chapter 1

Introduction

The Applied Sustainable Science Engineering and Technology (ASSET) institute at TU Delft is designing a wind energy production system based on kites as an alternative to the traditional Windmill. This power generation device, known as the Laddermill, has the advantage of harvesting wind at a higher altitude, which is faster and less gusty. This alternative presents some advantages, like the possibility to access stronger wind, but it also comes with new problems to research and solve. Developing this kind of systems requires collaboration between different fields including aerospace, electrical engineering, embedded systems, and mechanical engineering. For this early research part of the project, the researchers at ASSET need a real-time platform where they can acquire real-time data during system operation, test autopilot controllers, and validate kite models.

1.1 Kite Power Systems

The idea behind Kite Power Systems is to provide a cheaper and more efficient alternative to windmills. The main components of a windmill are a big mast, a generator, and a big rotor. By placing the generator at a certain altitude the rotor will turn due to wind forces and transfer that energy to the generator. As a new proposal, the generator can be placed in the ground, with no need for a large structure, and the energy from the wind can be harvested using a large kite. The kite is attached to the generator with a tether, making it is possible to access higher altitude wind —which is less gusty and stronger— and eliminating the need of a large structure. This results in cost savings compared to the windmill solution.

The current system prototype uses 25 or 14 square meter kites made of fabric with an inflatable structure at the leading edge. They are connected with a bridle system to the steering and power lines that are controlled by a control pod. The pod is a battery-powered system with an embedded computer, a wireless communication system, motor controllers, and a pair of motors. Its objective is to control the steering and angle of attack of the kite and to monitor the kite state.

The generator at the ground station is an induction motor with a gearbox that converts mechanical forces into electricity and vice versa. The ground station can store up to 20 kWh with a battery array. By rolling the generator drum in and out, a tether transfers mechanical energy between the kite and the ground station. The tether is made of a highly resistant synthetic fibre named Dyneema. The tether has a length of 980 meters, but during normal operation only 600 meters are used.

Power generation occurs during pumping cycles that have two phases: a reel-out phase where the kite pulls the generator producing power, and a reel-in phase where the generator pulls the kite investing some power. The angle of attack determines how much force the kite will generate by pulling the tether. The force at the generator is controlled by setting the angle of attack of the kite; a low angle of attack means low forces at the kite and a high angle of attack means high pulling forces. It is obvious that a high force from the kite is desirable during the reel-out phase to generate power, and a low force during the reel-in phase to save as much power as possible.

To maximize power output the kite flies two patterns. When in parking mode the kite keeps a low angle of attack and it is just hovering with its heading pointed to the zenith. This is the ideal flight mode to reel-in the tether and allow a new cycle to be started. To harvest energy, the kite flies crosswind, pulling the generator with high force. The path of the kite resembles the number eight but it may also fly in slightly different patterns always with a high angle of attack.

It is important to note the relation between the flight pattern and the phase at which the generator operates. The reel-out/reel-in speed, the angle of attack, and the wind speed determine the power generation performance of the system. The ground station and the pod must work well coordinated and communication between them is fundamental.

1.2 Communications and Control

The kite and the ground station work as a single system by exchanging data through a wireless link. Attitude, heading, apparent wind speed, and other sensor values are sent down to the ground station. Steering and power inputs are sent up to the pod. The Laddermill Prototype uses wireless to avoid installing a cable along the tether which would increase the cost and would significantly reduce the performance of the system.

The Laddermill prototype development started in 2008. Since then, it has been changing at every iteration of the design process. At the beginning of this project, the system had a simple wireless link implemented with XBee-

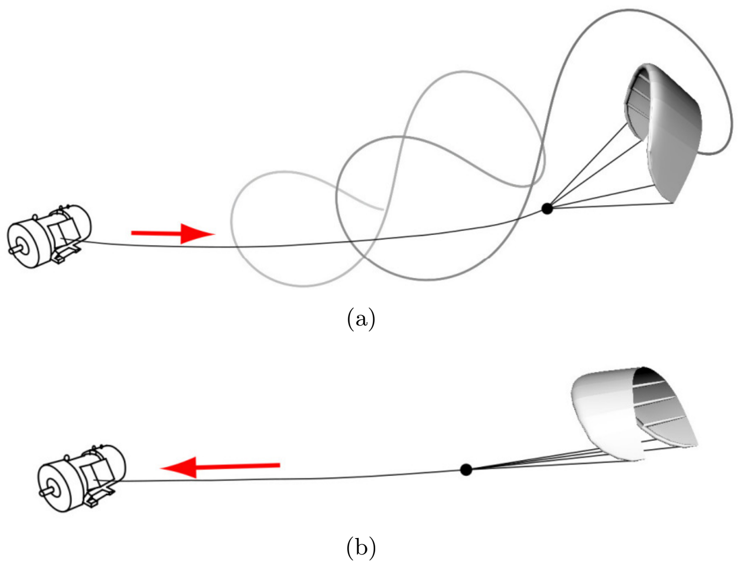


Figure 1.1: Reel out phase (a), flying figure of eight, high angle of attack. Reel-in phase (b), low angle of attack. [27]

Pro radios. The delay and packet loss parameters of the link were never measured. They proved, however, to be too large when a sophisticated controller was tested. An extra unidirectional wireless link is also available as a backup to avoid crashes. The backup link overrides the main link whenever communications fail. Figure 1.2 shows the initial wireless link.

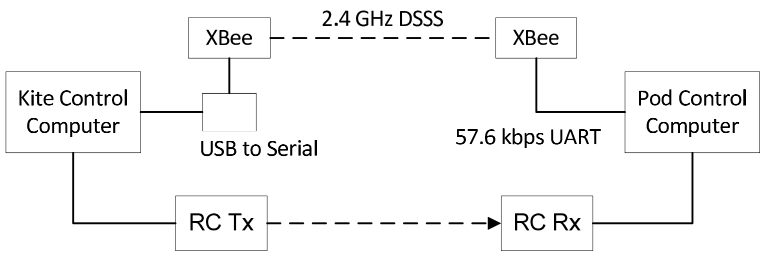


Figure 1.2: Communications System.

Researchers will use the system to test and tune different kind of controllers and to evaluate the accuracy of their kite models. The controllers under test will run in a computer at the ground station. This allows to use computation intensive algorithms like Non-Linear Model Predictive Control (NMPC) without introducing a heavy payload in the pod, and is convenient for developers who may not have experience developing embedded software applications. Sensor data goes through the wireless link after it is acquired by the pod and is sent down to serve as an input to the controller. After some calculations, the controller sends its output back to the pod computer

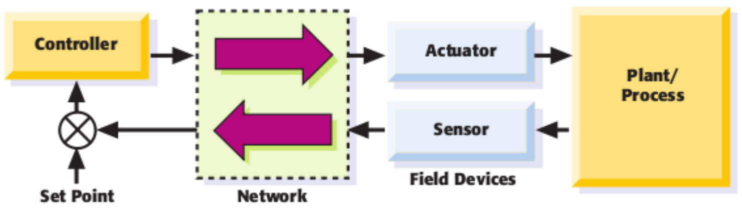


Figure 1.3: Control scheme. [12]

and to the actuators. This control scenario is depicted in Figure 1.3.

Any communication system is affected by packet losses and will introduce a delay between the source of the data and the sink. The source may be the output of the controller and the sink are the actuators that execute the desired output to the plant. In any control loop there is a limited delay that it tolerates after which the controller becomes unstable. Therefore the performance requirements are defined by the amount of delay and packet losses that the controller is able to endure. It is the task of the controller to keep the kite flying on the desired trajectory and the wireless link has to provide a fast and reliable connection between the controller and the plant.

1.3 Wireless Link Requirements

It is difficult to define the requirements clearly from the beginning, especially those related to the control system, which are in turn specified by the physical system. There are iterations in the design process, and each of them comes with changes and improvements to the system components. Not even a validated model for the kite is available that can be used to design and tune a controller. Given this situation the requirements were set with an estimated controller frequency of 20 Hz. In order to allow some computation time the maximum packet delay of the communications system should not exceed 35 ms. Packets that take longer than 35 ms shall be considered as packet losses. The packet loss rate must be lower than 1 for every 2000 packets sent considering a round trip for the data.

Other implicit requirements are the range and performance under bad weather conditions of the wireless link. The link must operate correctly at the maximum distance supported by the Laddermill Prototype (1000 m), and to achieve 24 hour flight it must operate correctly under rain. The solution must be as light as possible and must fit inside the pod. Power consumption should be less than 5 W.

1.4 Problem Statement

The goal of this work is to implement a fast and reliable wireless link between the Kite Control Computer and the Pod Computer that meets the application requirements and that allows the researchers to gather real-time data of the flight, test kite physical models, and test controllers to implement automatic flight control. It is part of this thesis work to determine the limits at which the system can operate in terms of distance, weather conditions, and Quality of Service (QoS) of the wireless link.

1.5 Methodology

We start with an initial wireless link with unknown performance metrics and a set of requirements based on the expected controller performance. In order to implement a fast and reliable wireless link we determined the delay and packet loss rate of initial system to find how they compare to the required target. A solution with redundant radios that combines the main link with an independent link was implemented to provide higher reliability. We explored different options on how to combine both links and performed experiments to select the best implementation. A statistical model of round trip time and packet loss for each link and for the final solution were built to compare and measure the improvement.

The models were used to implement a simulator to explore the stability of a two controllers. The results of the simulation were used to determine more formal requirements, nevertheless they are valid only for the particular case of this controller and the kite model used during the simulation.

The rest of the document is organized as follows: Chapter 2 contains the literature study, Chapter 3 shows the design of an improved wireless link, Chapter 4 shows the simulations of the controller running on top of the wireless link, and Chapter 5 concludes and proposes future work.

Chapter 2

Literature Study

The goal of this chapter is to describe our application from the Networked Control Systems (NCS) point of view, and to provide other examples of similar studies. We divided this chapter into three sections. First, we present some background theory on NCS and real-time systems. We review basic control theory concepts, and describe delay and packet loss models found in literature. Second, we present a small survey on wind energy generation using kites, with some controller designs, and some systems that have been implemented. We describe the communication systems used and the performance requirement of their controllers. Third, we list similar applications, with wireless communications and mobility patterns, where latency and packet loss measurements have been studied.

2.1 Background Information

The Laddermill Prototype will be running autopilot controllers in a ground station computer, closing the loop through a wireless link with the actuators and sensors in the pod. The system requires certain QoS from the network and also real-time capabilities at both computers. This section gives an introduction to Networked Control Systems and explores some options on how to implement real-time systems in Linux.

2.1.1 Networked Control Systems

The field of Networked Control Systems (NCS) studies feedback control systems where the loops are closed through a real-time network. In this section we introduce the two types of NCS found in literature, the effects of adding a network in the control loop, and how these effects have been modelled by some authors.

Tipsuwan et al. [30] present a direct versus hierarchical distinction between control systems. Direct controllers are connected to a plant through a network, while the plant has immediate access to the sensors and actuators. The plant is sampled at a fixed interval T. Samples are sent through a network to an event-driven controller that, after receiving the input, generates a control output. The output travels back through the network to reach the actuators that operate on the plant as shown in Figure 2.1.

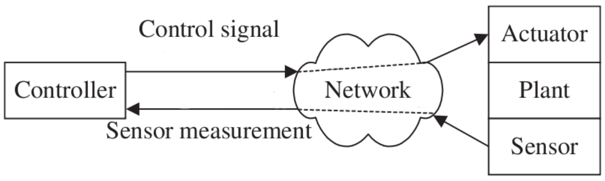


Figure 2.1: Direct control system.

In a hierarchical configuration, a fast controller that accesses data from sensors and actuators is running at the plant and a network connects the plant to a high-level controller as depicted in Figure 2.2. Direct mode requires higher network performance than hierarchical mode, which requires more processing power at the plant. A direct control strategy was selected by ASSET researchers to take advantage of the processing power of the computer at the ground to evaluate complex NMPC controllers, and to allow developers without any background on embedded systems to test their designs. Sensor information from the ground station must be used in combination with the kite sensor information to estimate the kite state. That is another reason why the hierarchical mode is not used in the system.

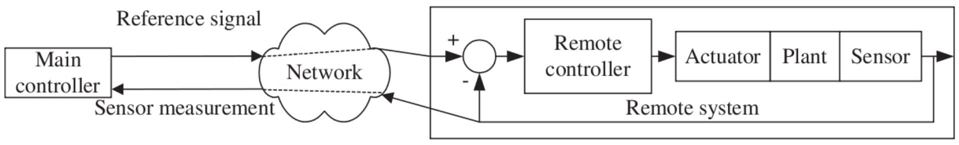


Figure 2.2: Hierarchical control system.

We follow the general delay model for direct control systems of Tipsuwan et al. [30]. In Figure 2.3, **r** is the reference signal, **u** is the control signal, **y** is the output of the sensors, k is a time index and T is the sampling period. We are interested in the sensor-to-controller delay τ^{sc}, which we will refer to as *downlink*, and the controller-to-actuator delay τ^{ca} or *uplink*.

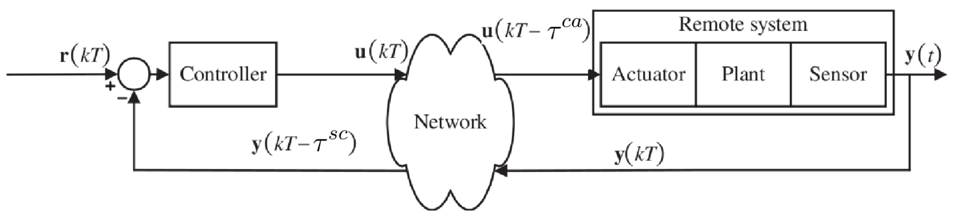


Figure 2.3: Loop delays.

Figure 2.4 shows the timing details for the delays on the network. τ^{se} is the time at which the plant sends feedback to the controller, τ^{cs} is the time at which the controller starts processing the input data, τ^{ce} is the time instant when the controller sends a control packet through the network, and τ^{rs} is the time at which the plant receives the control signal. The controller processing delay τ^{c} and both network delays can be added together as the control delay τ.

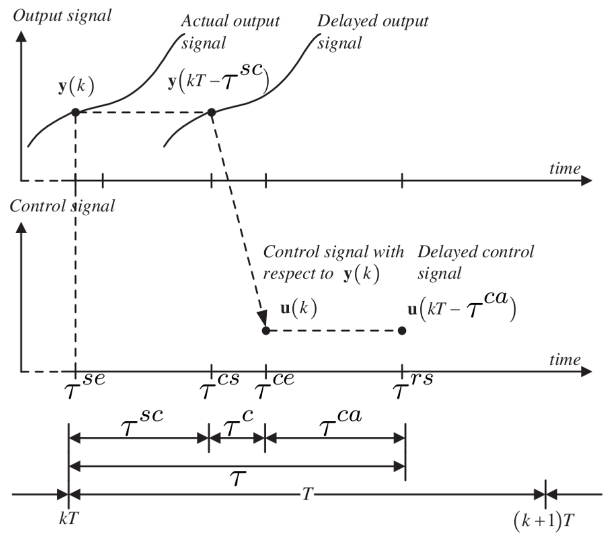


Figure 2.4: Timing diagram.

$$\tau^{sc} = \tau^{cs} - \tau^{se}$$

$$\tau^{ca} = \tau^{rs} - \tau^{ce}$$

$$\tau = \tau^{sc} + \tau^{c} + \tau^{ca}$$

Networks have random time varying delays and packet losses that affect the stability of control loops. These are the main source of performance degradation in NCS [14]. The kind of networks required for NCS do not need high bandwidth, but reliable and secure data transmission. Network delays are determined by the Media Access Control (MAC) algorithm, routing path variations and number of hops, processing delays in the controller, retransmission of frames due to physical layer faults, etc. Out-of-order arrivals, which are counted as packet losses, can be avoided by keeping the variation of the delay lower than the sample period. The challenge is to "keep control delay as short as possible to avoid phase lag, and to avoid out-of-order arrivals" [22].

Consider a simple example of an RS-232 physical layer connecting two UARTs. The delay model in this case is a function of the amount of data to be transmitted and it is affected by the physical layer encoding and frame structure. For every byte of information there will be 10 bits in a 8N1 configuration. The data rate sets the duration of a bit and with short buffers the channel can be fully loaded. When considering more complex protocols like CAN or Ethernet the models become more complex. A simple linear model is no longer enough because of media access contention and encoding in shared media work. A statistical model can describe the behaviour of these interfaces as explained by Nilsson [22].

2.1.2 Delay Models in NCS

According to Nilsson [22] and Eriksson [14], there are three types of delay models:

Constant delay model: it can be the average or the maximum measured delay in a network. It is a simple model that does not include delay variations, or jitter.

$$\tau_1(t) = C$$

Single distribution delay model: delays are modelled using a probability distribution, for example a normally distributed delay model with a specific variance and mean.

$$\tau_2(t) = x_1(t), X_1 \sim N(\mu, \sigma^2)$$

Eriksson [14] presents a probability distribution of network delay. Nilsson [22] proposes a model for a CAN network, and presents experimental results that verify the accuracy of the model.

Markov chain: it can combine multiple distribution delay models, for example, Ethernet is a random access network that can be modelled as

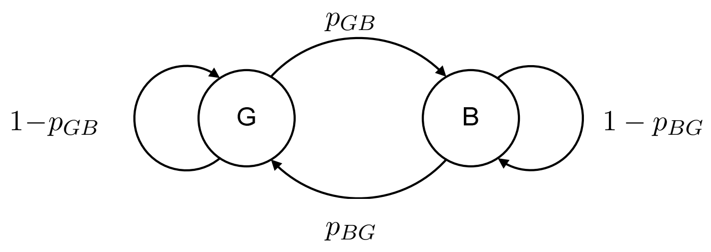


Figure 2.5: Simple Gilbert model with two parameters: p_{GB} and p_{BG}. The transition probabilities depend on the current state.

a Markov chain of different probability distributions as described by Tipsuwan et al. [30].

2.1.3 Packet Loss Models

There are two simple models for describing the success or failure of an event like packet reception. A Bernoulli process is a simple example of an event where the outcome can be a success with a probability $p_{success}$ or a failure with probability $p_{loss} = 1 - p_{success}$. In this case the effective reception of the packet after the round trip is a success, whereas a lost packet is a failure. This is a simple model because it assumes that the events are independent from each other and identically distributed. Sinopoli et al. [28] modelled packet loss between sensor-controller and controller-actuator as independent Bernoulli processes.

Since packet losses appear to have a bursty nature in wireless connections, the two-state Markov chain known as Simple Gilbert model (a simplified version of the Gilbert-Elliot model) may be a better fit. In this case there are two states, "good" and "bad". In such a model there are four transitions, meaning that not only the probability of a failure is modelled, but also the probability of failure once a failure has already occurred. Figure 2.5 shows the transition probabilities p_{GB} of going from a good state to a bad state, and p_{BG} of going from a bad state to a good state.

The transition probabilities can be expressed as a function of measurable properties on a packet loss samples as in Salsano [26]. The transition probabilities can be calculated as follows:

$$p_{GB} = \frac{1}{E(B)}$$

$$p_{BG} = \frac{p_{loss}}{E(B)p_{loss}}$$

In this case $E(B)$ is the average length of consecutive lost packets, or average error run length, and p_{loss} is the percentage of lost packets (the Bernoulli model). The transition probability matrix P is composed as follows.

$$P = \begin{bmatrix} 1 - p_{GB} & p_{GB} \\ p_{BG} & 1 - p_{BG} \end{bmatrix}$$

2.1.4 Linux Real-Time Systems

We are interested in the performance of Linux as a real-time operating system to find out how to implement low latency devices that can run controllers, operate the actuators, and perform computations in general in a timely fashion. This is required for the ground station and pod computers. Real-time systems are evaluated on the basis of their latency and jitter. The time between an event and correct reaction is called latency, and the variation of such latency is known as jitter. This section describes some results found in literature about real-time Linux systems and their latencies.

The main sources of latency in the standard kernel are timer resolution (also known as scheduler latency problem), and large non-preemptible code sections in the kernel that can block the processor from dispatching interrupts during tens of milliseconds. There are two types of solutions to the latency problem: a mini-kernel that runs at a lower level and handles interrupts with low latency before passing them to Linux if required like ADEOS [1], or patches to the kernel that allow preemption like the Preemption and Low latency patch [3].

Bruzzone et al. [9] used a PC104, which is a mature product that has a high precision timer and PCI interface. They used a kernel patched for preemption in two applications and measure the waiting times for threads in the system. They concluded that with their system they can implement applications with timing requirements in the order of 1 ms.

Altenberg et al. [6] compare the Fast Context Switch Extension patch for ARM devices. They use an ARM9 and an ARM11 and present their results as a guide for developers to select between these two targets. They also use the Preempt RT patch. They use some applications to generate load on the CPU and the network interface. ARM11 devices are recommended for applications where the time required to service an external interrupt from user space is less than 1 ms, and where the worst case latency should be less than 250 us. Systems with looser constraints can be implemented with an ARM9, in it which is possible to service external events from user space with a 1 ms latency.

In Abeni et al. [4] they study the latency of the a preemptable Linux kernel under different stress tests. Memory accesses, I/O load, caps lock, LED toggling, and accesses to the */proc* file system are executed in the background while a task uses the *usleep* function to wake up periodically.

They show how the Preemptable kernel reduces the latency during reads to procfs and during console switch tests but not during the memory test. No specific test related to interrupts is presented.

The requirements for the network performance are clear. Nevertheless, the whole application has to provide real-time capabilities to be able to comply with such requirements. It is possible to use commercial off-the-shelf hardware and free software to implement a real-time system. Linux is becoming more popular for embedded systems because of its portability and the wide variety of platforms where it runs. It was designed to provide fairness of resource distribution for users and applications but it can, however, be patched to provide real-time operation for control applications.

2.2 Related Work

2.2.1 Kite Power Systems

We present a small survey on literature related to kite controller design. We summarize the performance characteristics required by the systems or the controllers to compare them to the initial set of requirements we received.

There are some publications about kite controller proposals designed to keep a kite flying figure-of-eight and parking modes, like Ahmed et al. [5] Houska et al. [16] Ilzhofer et al. [17], Novara et al. [23], and Williams et al. [32]. Most of them tackle the problem from the control theory point of view and the results presented are simulations. These papers do not define where the actuators are, they disregard the processing power of the device executing the controller, and usually they do not use a discrete time analysis with an inherent sample frequency and delays. The types of controllers presented are NMPC, Feedback PID, and evolution of control agents through genetic algorithms. As in any controller, the inputs from the sensors are given to the controller and the output signal is sent to the actuators. For a Kite Power System, the actuators providing a differential steering input to control the kite may be located in a pod —close to the bridle system— or at the ground.

In the case of Ahmed et al. [5], some real-time considerations are mentioned. The NMPC controller presented is implemented with fixed point arithmetic to enhance performance, but no results on execution time are reported. According to Novara et al. [23], building a reliable controller for a Kite is very difficult. Most of the real life controllers require two steps: first develop a model of the plant, then design a controller using the model. To implement successful NMPC it is important to have an accurate model, and this is particularly difficult for flexible structures like kites. In this case, the authors use a discrete time analysis, they consider an embedded computer with limited resources, and they provide pseudocode for their controller blocks.

Ilzhofer et al. [17] present an NMPC controller with a frequency of 1 Hz. They show that the execution time is lower than 500 milliseconds, nevertheless the delay between input and output is neglected as most of the calculations can be executed in advance with only a few operations related to the feedback from the plant. No information is provided on the system implementation, so the timing results are difficult to interpret.

SkySails [13] is using kites to tow cargo ships and save fuel. This product faces similar challenges as Kite Power Systems, namely to control a kite. They ruled out MPC because of the complexities of building a model for the kite. The system has a control pod next to the kite's bridle system that is running a feed-forward cascaded PI controller at 10 Hz to provide a differential steering input to the kite. They used a delay block to model the aggregated delay of all the components, but no detail on the sources or amount of delay are given.

At the Politecnico di Torino, Canale et al. [11] and Fagiano et al. [15], developed a prototype of KiteGen, a system with actuators at the ground that provides steering inputs to the kite and converts mechanical force into electricity. They use a pair of triaxial accelerometers, magnetometers, and GPS to determine the position of the kite. Sensor values are sent down by radio signals. For the controller they use a fast version of NMPC with sample frequencies of between 5 and 10 Hz.

The Laddermill prototype developed at TU Delft uses a control pod like [13], but the controller is running at the ground station, that is, through a wireless link between the ground station and the pod. The controller output has to travel through the network to get to the actuator, in contrast to [15], where the sensor data, once it is processed in the ground station, is ready to be applied to the actuators. However the usage of a single line to connect the kite and ground station reduces the cable drag forces. As pointed out by [15], the TU Delft system gives more control possibilities, but is more susceptible to faults because of the wireless link.

Baayen [7] developed a controller with proven stability, minimal model requirements, and low implementation complexity. The controller was tested at the Kite Power System prototype from TU Delft with negative results due to delays. They found out through simulation that delays longer than 100 ms seriously degrade control performance.

Wireless communications are found in other systems [11, 13, 15], however, they are using low sampling rates, or they have a configuration where the actuators are located at the ground station. There is not a lot of information on the characteristics of the wireless links used and a performance analysis is missing. In the next section we present an assessment of the initial XBee-Pro wireless link to be used as a starting point to make further enhancements.

2.2.2 Communications System Assessment

An initial assessment of the communications system was performed to measure the delay and packet losses between the ground station and the pod. The ground station or Kite Control Computer (KCC) is a laptop taking inputs from a human pilot or an autopilot and sending them to the Pod Control Computer (PCC) across a wireless link. The wireless link is also used to report flight data from the on board sensors to the pilot. The initial measurements provided information on the quality of the communications and served as a starting point for improving the performance.

A pair of XBee-Pro radios is used for communications, which operates at 2.4 GHz in the free Industrial, Scientific, and Medical (ISM) band. The devices are operating as a serial cable replacement with a serial interface at 57.6 kbps and a wireless data rate of 250 kbps. A radio controller is used as a backup system to allow the pilot to override the commands sent by the KCC. This connection is not bidirectional, it can only send steering and power inputs to the kite. This is used to avoid crashes when the XBee-Pro or the KCC fails. Figure 2.6 shows the communication system.

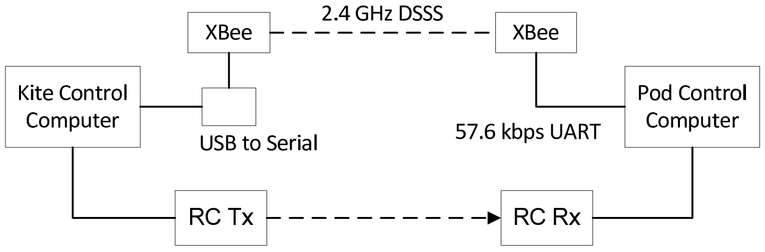


Figure 2.6: Communications System Overview: Kite Control Computer connected to the Pod Control Computer using XBee-Pro devices. A USB-to-serial converter is used at the Kite Control Computer.

We executed a basic set of tests for determining packet losses and delay with a simple ping application to measure the round trip time for information going from the KCC to the PCC and back to the KCC. To get a clear idea of the wireless link performance in the different scenarios we tested at three distances in both operation modes. We consider a long range to be 400 meters and tested at steps of 100 meter until a practical limit is reached. Operating the kite closer than 200 meters is not common during the pumping cycles.

The PCC has an ARM9 device running a 2.6.33.5 Linux kernel. The KCC was a dual core Dell laptop running Ubuntu with a 2.6.33 kernel. The measurement software was developed in C++. In normal operation, the commands sent via the uplink are around 20 bytes and the packets coming down from the pod are around 100 bytes. The same data sizes were used for the assessment. The measurements are for total round trip time with no distinction between the uplink and downlink. If a packet is lost in either path —going up or coming down— it will be counted as just one packet loss. The KCC is polling the serial port at a fixed interval of 6 ms and the PCC is doing it at 4 ms.

Figure 2.7 shows the results for the initial assessment of the wireless link. We can observe that with longer distance the packet losses increase and also that packet losses are higher when flying figure of eight than when in parking mode. Average round trip times have a similar average regardless of the distance and the flight pattern. The maximum round trip time for all the modes and distances is 85.58 ms.

For XBee modules the problem is that packet losses are dependent on the distance between the ground station and the kite, and on the flight mode. These are the packet loss probabilities for each test in Figure 2.7.

These are the packet loss probabilities for each of the experiments:

Flight mode	400 m	300 m	200 m
Figure-of-eight	9.22 %	9.94 %	4.11 %
Parking	8.13 %	2.01 %	0.47 %

Table 2.1: Packet loss rate for X-Bee link.

The average delay of the XBee-Pro link is around 64 ms, but the plots show that whenever a packet completes the round trip the delay is less than 90 ms. For the case of packet losses we can see a clear problem whenever the kite is flying figures of eight. The packet loss rate for parking mode is much lower than for the figure-of-eight. Antenna alignment and mobility are the primary suspects for such behaviour. In the next section we present some of the wireless technologies that we considered to solve the problems.

2.2.3 Wireless Connectivity Solutions

In this section we start by presenting an overview of the radio frequency allocation in the Netherlands. This gives a general idea on which bands can be used without licensing and the power limitations that they are subject to. Then we present some wireless technologies considered for the application.

Free License Frequency Bands

The *Vergunningsvrije radiotoepassingen* document specifies the frequencies that can be used and the allowed transmission power limit for each band in the Netherlands. Table 2.2 lists some of the free license frequencies that can be used for telemetry and other general applications.

Table 2.3 lists the power limit for broadband devices. Equipment using IEEE 802.11 RLANs, HomeRF^{TM}, Bluetooth^{TM} wireless technologies, Zigbee^{TM}, etc. are considered wide band data transmission equipment. [2]

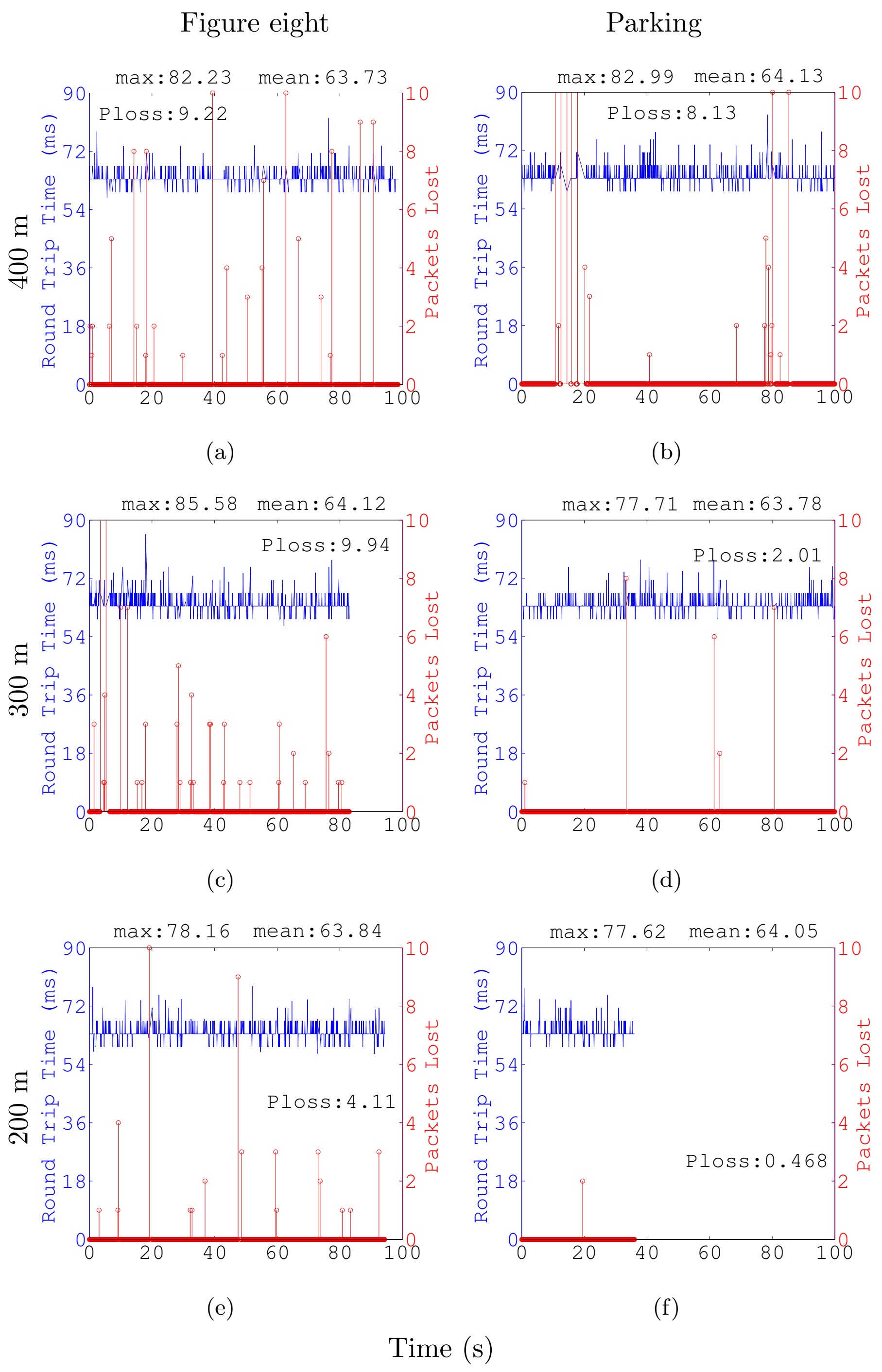


Figure 2.7: XBee wireless link initial assessment results. Figure eight (left column) and parking (right column).

Frequency Band	Power	Duty Cycle
6,765 - 6,795 MHz	42 dBA/m at 10 m distance	-
13,553 - 13,567 MHz	42 dBA/m at 10 m distance	-
863,000 - 865,000 MHz	25 mW e.r.p.	$< 0.1\%$
865,000 - 868,600 MHz	25 mW e.r.p.	$< 1.0\%$
868,700 - 869,200 MHz	25 mW e.r.p.	$< 0.1\%$
869,300 - 869,400 MHz	10 mW e.r.p.	-
869,400 - 869,650 MHz	500 mW e.r.p.	$< 10\%$
869,499 - 869,650 MHz	25 mW e.r.p.	-
869,700 - 870,000 MHz	5 mW e.r.p.	-
869,700 - 870,000 MHz	25 mW e.r.p.	-
2400 - 2483,5 MHz	10 mW e.i.r.p.	-
5725 - 5875 MHz	25 mW e.i.r.p.	-
24,00 - 24,25 GHz	100 mW e.i.r.p.	-
61,0 - 61,5 GHz	100 mW e.i.r.p.	-

Effective isotropically radiated power is the amount of power that a theoretical isotropic antenna (which evenly distributes power in all directions) would emit to produce the peak power density observed in the direction of maximum antenna gain. Effective radiated power is similar to e.i.r.p but may use some other reference antenna than an isotropic antenna, e.g. a half-wave dipole

Table 2.2: Frequencies available for general applications such as telemetry and alarms [29].

Table 3.2 shows the different bands around 5 GHz that are available for radio LANs. The lower bands are restricted to indoor operation so they can be discarded for our application. The highest band can be used with a 1 W transmitter power but has to comply with *Transmitter Power Control* (TPC) and *Dynamic Frequency Selection* (DFS) standards.

DFS detects weather radars to avoid operating in the same frequencies. Radar detection and uniform loading are required on the frequency ranges between 5.250 GHz to 5.350 GHz and 5.470 GHz to 5.725 GHz. TPC is a technique that ensures a mitigation factor of at least 3 dB to reduce interference between systems. It is not required if the channel falls completely within the 5.150 to 5.250 GHz band.

The DFS procedure starts by assuming that all channels are unavailable. Before transmitting on any channel, a *channel availability check* (**CAC**) must be performed. There are two options for checking a channel: a normal CAC or an off-line CAC. A normal check is done by monitoring a channel continuously for one minute, if no radar is find the channel becomes available and can be used. The off-line CAC can be done when using another frequency for communication. In this case the channel being checked is monitored only during small time slots. Off-line checks have to run as long as 24 hours to guarantee that no radar is transmitting on that channel. When-

Frequency Band	Power	Power Density
2400 - 2483,5 MHz	100 mW e.i.r.p.	
57,0 - 66,0 GHz	40 dBm e.i.r.p.	13 dBm/MHz e.i.r.p. density (1)
57,0 - 66,0 GHz	25 dBm e.i.r.p.	-2 dBm/MHz e.i.r.p. density (2)

1) Applications outdoors are excluded.
2) Fixed outdoor installations are excluded.

Table 2.3: Wideband data systems

Frequency Band	Power	Power Density
5150 - 5250 MHz (3)	200 mW e.i.r.p.	10 mW/MHz (2)
5250 - 5350 MHz (1,3)	200 mW e.i.r.p. (2)	10 mW/MHz
5470 - 5725 MHz (1)	1 W e.i.r.p.	50 mW/MHz (2)

1) The output power is with TPC (Transmitter Power Control) and DFS (Dynamic Frequency Selection).
2) The maximum average power density.
3) Only indoor use is permitted.
4) With DFS only, the max. e.i.r.p. is 500 mW.

Table 2.4: Broadband access systems including Radio Local Area Networks (RLAN's)

ever a radar is detected on a channel it becomes unavailable and can not be checked for at least 30 minutes known as non-occupancy period. After this period expires the channel must be scanned again before it can be used.

Frequency Range	Duration	
	CAC	Off-line CAC
Within 5600 MHz to 5650 MHz	10 min	24 hours
Outside 5600 MHz to 5650 MHz	1 min	4 hours

Table 2.5: DFS channel scan duration.

The DFS procedure sometimes results in false detections that block the channels for 30 minutes. To minimize the consequences of a false positive it is recommended to stay away from the 5.600 GHz to 5.650 GHz range to avoid waiting times of 10 minutes to perform the CAC. It is also recommended to scan the region where the device will operate pointing the antennas to different orientations as the DFS hits are directional.

The IEEE 802.11 Standard

To be able to decrease latency, and to provide a more robust connection, we considered the 802.11 standard. The IEEE 802.11 standard had three physical layers: 2.4 GHz in Direct-Sequence Spread Spectrum (DSSS), 2.4 GHz in Frequency-Hopping Spread Spectrum (FHSS), and Infrared (IR). A new physical layer at 5.2 GHz was introduced for 802.11a. Orthogonal Frequency Division Multiplexing (OFDM) made it possible to reach 54 Mbps data rate. 802.11b uses 2.4 GHz but has a different modulation scheme —Complementary Code Keying— to achieve data rates of 11 Mbps. The 802.11g (2003) standard used the same multiplexing techniques as 802.11a, but at 2.4 GHz, achieving high data rates, and maintaining backwards compatibility with 2.4 GHz devices as well. [24]

The 802.11n amendment introduced new techniques at the MAC, and PHY layers to be used at 2.4 GHz and 5.2 GHz. The most compelling characteristic is the use of Multiple Input Multiple Output (MIMO) antennas which can be used to overcome the alignment problem concluded from the initial assessment of the channel.

In addition to 802.11n we considered other low-cost radio solutions that could be added to provide extra connectivity on a different band. The 802.14.5 standard was released in 2003. It is a low-data-rate, low-cost, low-power solution for wireless connectivity. It is targeted for applications where 802.11 is overkill and where long battery life is needed. The properties come from its reduced complexity at both MAC and PHY layers. In our project we used similar radios at the physical level only (not using their MAC, or Zigbee networks). The devices tested were used as serial cable replacement. We present the physical layer characteristics of the devices.

802.14.5 was designed for two physical layer options: 2.4 GHz or 868/915 MHz. When in 2.4 GHz mode the data rate over the air is 250 Kbps. It has 16 channels with 5 MHz separation. This is a high throughput and lower latency mode for the radios. In 868/915 MHz mode the data rate is limited to 20 and 40 Kbps respectively. It has 11 channels with 2 MHz separation. In this case it is possible to get better sensitivity, which translates into a larger coverage area (Callaway et al. [10]).

Petrova et al. [25] performed measurements and simulations on coverage, throughput, and error run length with off-the-shelf 802.15.4 radios. The authors experimented indoors and outdoors at 2.4 GHz with 250 Kbps data rate. They sent 20 packets per second with a 20 byte packet size and averaged the results over several trials. Packet losses were modelled as Bernoulli variables by defining the probability of a packet loss. A two-state Markov chain model (Gilbert-Elliot) was obtained by analysing the error run lengths. The delay models work for distances lower than 20 m, at larger distances the best fit was achieved with the two-state Markov chain.

2.3 Latency and Packet Loss Measurements in Other Applications

We explored other applications with some mobility patterns where wireless networks were considered. The evaluation procedure and results inspired our work and motivated the selection of 802.11n. We present also other papers related to latency and packet loss measurements in a network, which are relevant as they present other models that we can use.

In the field of Inter Vehicle Communications (IVC), Matsumoto et al. [20] and Wewetzer et al. [31] have studied the possibility of using wireless networks to transfer safety information between mobile nodes (cars). The mobility pattern of vehicles on a highway or cruising the suburbs is different than kites. Antenna alignment is not a problem as vehicles remain on the ground and should not yaw or roll under normal operation. There is however a higher probability of obstacles in the line of sight and other networks that may be operating nearby.

Wewetzer et al. [31] compare UMTS with WLAN. They study moving vehicles that exchange safety, traffic control, and entertainment data with different performance requirements. The requirements for safety applications are similar to our requirements; timely, low latency, fast access to media for nodes with mobility. They explore the coverage range, latency, and effects of mobility on latency for both solutions. They executed the test by placing laptop computers on each vehicle and attaching antennas on the roof and using a simple ping application. With one car parked, the other car passing by at 100 and 150 km/h, they tested the coverage range, latency, and effects of mobility on latency for both solutions.

UMTS has a centralized structure with no capability for ad-hoc communications. This provides advantages in the coverage, but the round trip time sometimes exceeds 400 ms. This is not fast enough for safety applications. With WLAN most round trips occurred under 1 ms with some variations of up to 25 ms. The coverage range is 1450 m with the first packet loss at 750 m. They did not found any effect of mobility on latency. In [31], car-to-car outdoor throughput and coverage range measurements were executed. They report 850 m coverage range with one spatial stream, 290 m with two, when using 802.11n. However the results are in terms of throughput versus distance and do not present ping time measurements. The results on latency and coverage range are promising for our application.

Chapter 3

Dual Wireless Link

In this chapter we describe a dual wireless link solution that operates on the free license spectrum that meets the latency and packet loss requirements of 35 ms round trip time and less than one packet loss in 2000 packets.

We start by describing the system components, and the main design choices for the dual wireless link. We explain how the links work together and the performance in terms of round trip time and packet losses. Afterwards we build basic models for the performance of the link, similar to the ones found in literature. Measurements for a range test are presented before concluding with the performance measurements for two autopilot controllers working under the different performance conditions at which the dual link operates.

3.1 System Description

Different parts of the Kite Control system were upgraded during the project. Figure 3.1 shows the updated parts including the dual link. The KCC is a high performance industrial embedded system with no moving parts and a quad core Intel i7 processor. It has 4 GB of RAM, 120 GB SSD, 5 serial ports, and two 1 Gb Ethernet interfaces. This computer communicates with the kite through the dual link, runs the autopilot software, and serves as interface to the human pilot as well.

The PCC is an Electrum-100 embedded computer with a small and light form factor. It has a 400 MHz 32-bit ARM9 processor with 64 MB SDRAM, one 10/100 Ethernet interface, six serial ports, and an external SPI ADC. The PCC is the main processor in the pod, but it is still interfacing a proprietary board to apply the steering inputs. It gathers data from the on-board sensors and communicates to the KCC through the dual link which is described in detail in the following sections.

For the wireless link connecting both computers we decided to use a dual link approach. Taking into account the free license spectrum as regulated

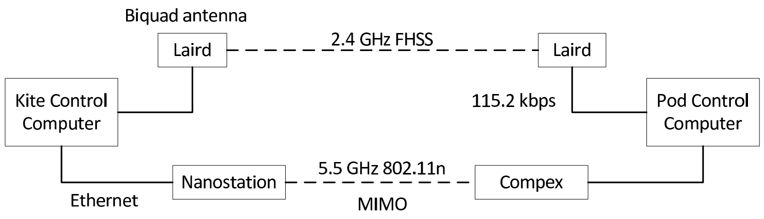


Figure 3.1: System components. KCC connected through UART to Laird modules and to the Nanostation through Ethernet. PCC connected through UART to Laird modules and to the WPE72 through Ethernet.

by the *Agentschap Telecom*, and our conclusions from the initial X-Bee link assessment, we selected two frequencies. After performing the initial assessment of the X-Bee link, we decided to use 802.11n with MIMO antennas to avoid antenna orientation problems. 802.11n devices can operate at 2.4 or 5.5 GHz. In the 5.470 - 5.725 GHz frequency range we can transmit with up to 1 W power if we use TPC and DFS (cf. in Chapter 2). The 2.4 GHz band, which is narrower and limited to 100 mW, is used for a backup link. It is also possible to find a variety of low power radios that operate in the ISM band. We discarded the 869 MHz band because it offers narrow channels with limited power. Even for the 869.400 - 869.650 MHz band, where 500 mW is allowed, the duty cycle is limited to less than 10%. The next two sections describe the hardware selected for the dual link.

3.1.1 Main Link

Table 3.3 shows a list of devices that were considered for the main link. Dual band devices were discarded to avoid interference with the backup link, and have a high power consumption. The Compex WPE72 has a low weight and it includes OpenWRT, which is a linux distribution for embedded devices that can be customized to use the radio in any way. Mikrotik RB433 was considered but it is bigger (105 x 150 mm) than the Compex device (95 x 68 mm). The main link consists of a Ubiquity Nanostation Loco M5 on the ground, and a Compex WPE72 router on the pod.

The KCC is connected to the network through a switch where the Nanostation —working as an access point— is connected. The PCC is directly connected to the Compex router that works as a station. The Nanostation and Compex have a channel width configuration of 5 MHz, and are limited to use the following channels: 5.500, 5.520, 5.540, 5.560, 5.680, 5.700 GHz. Using a subset of the available channels on both devices reduces the time needed to set up the connection. The data rate is fixed to 1.5 Mbps. For this application we do not need a high bandwidth so we disabled any data rate switching.

Manufacturer Module	OS	Band	Antenna	Power Consumption	Weight
Ubiquiti Nanostation	-	5 GHz	3x3 MIMO	5.5 W	180 g
adstec IWL3000	-	Dual	3x3 MIMO	-	-
Hirschmann BAT300		Dual	3x3 MIMO	10 W	-
Mikrotik RB433	RouterOS	5 GHz	2x2 MIMO	4.95 W	137 g
Compex WPE72	OpenWRT	5 GHz	2x2 MIMO	5 W	100 g

Table 3.1: 5.4 GHz devices evaluated for the main link.

Device	Frequency Range	Gain	Vertical/Horizontal Angle
WiMo Antennen	5400 - 5900 MHz	9 dBi	$60^\circ/60^\circ$
Ubiquiti Nanostation	5470 - 5825 MHz	13 dBi	$45^\circ/45^\circ$

Table 3.2: Antenna List

Omnidirectional antennas have lower range and are more sensitive to noise. Directional antennas can cause alignment problems if the opening angle is too small. We use antennas with 60°, and 45° opening angles.

Both devices are capable of performing DFS but do not include the TPC feature. Normally only the master device performs DFS checks. The main link provides a low latency connection with low packet loss, however, the set up time after a reset or after any radar detection event can be one or a few minutes, depending on the available channels and previous radar detection events. The kite cannot be left unattended for that long, even if it only happens once a year. This is where the backup link keeps the kite flying safely.

The packets over the main link include all information collected by the sensors in the pod. The frame size after serialization is on average 240 bytes. Figure 3.2 shows the round trip time for the main link. The average round trip time is 4.72 ms, and the maximum is 8 ms. The green line is a time stamp added at the PCC. We can see that the time spent in the uplink is lower (1.7 ms) than in the downlink (3 ms), which is expected as we have larger data frames coming down the kite.

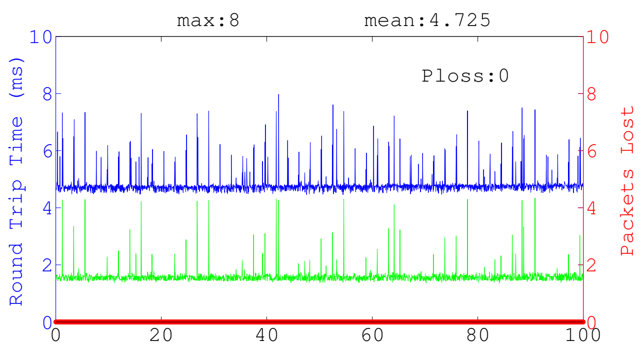


Figure 3.2: Main link round trip time during 100 seconds. The blue line shows the total round trip time in milliseconds. The green line is a time stamp when the packet was received at the PCC. The red stems are packet losses (in this case zero).

3.1.2 Backup Link

The backup link serves as a low data rate and higher latency connection with very low set up time. The advantage of this kind of device is that packet losses degrade gradually, and even in adverse conditions it is possible to communicate some packets so the system can exchange data even when the main link is down.

Table 3.3 shows a list of devices that were considered for the backup link. The Laird LT2510 module was selected because it provides the longest range in the 2.4 GHz band and it can be configured to use two RF data rates. The AC4424 has a longer range, but the data sheet did not specify data rate information, and the device is also near the end of life phase of the product life cycle.

Manufacturer Module	Modulation	Frequency	Data Rate (RF)	Interface	Range
Y-Lynx TRM8053	FHSS	868 MHz	152.3 Kbps	UART	40 Km
MicroLinear ML2722		2,4 GHz	1.53 Mbps	UART	
Laird AC4424	FHSS	2,4 GHz			3.2 Km
Laird LT2510	FHSS	2,4 GHz	280/500 kbps	UART	2.4 Km
Digi XBee	DSSS	2,4 GHz	250 Kbps	UART	1.6 Km

Table 3.3: Low throughput devices evaluated for the backup link.

The KCC has six serial ports so no USB-to-UART converter is required.

Serial interfaces are configured at 115.2 kbps 8N1. The Laird modules are configured to work at 500 kbps wireless data rate, and to use 8 retries for each packet.

The module connected to the KCC is configured as server. Most of the measurements were taken with a 7 dBi dipole antenna. On the latest tests we used a home made 11 — 13 dBi Antenna with much better performance. At the PCC we used a dipole antenna with the module configured as client.

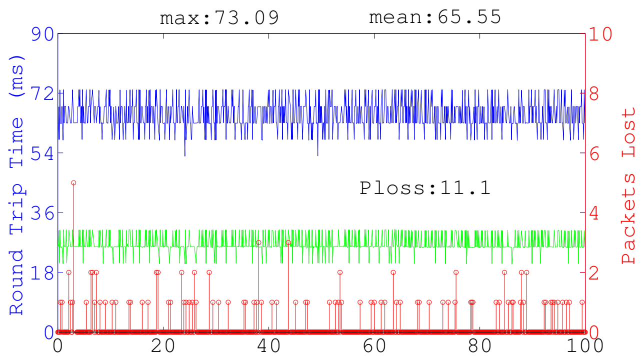


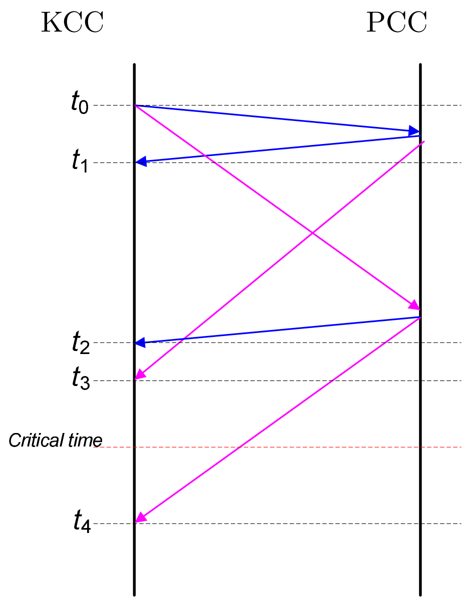
Figure 3.3: Backup link round trip time during 100 seconds. The blue line shows the total round trip time in milliseconds. The green line is a time stamp when the packet was received at the PCC. The red stems are packet losses.

For the backup link, the packet loss rate increases with the size of the data in the frame. We keep the frame size as small as possible to reduce packet losses. The backup link reports only critical data for the autopilot. The average frame size is 120 bytes. Figure 3.3 shows the round trip time for the backup link. The average round trip time is 65.55 ms, and the maximum is 73.09 ms. The time spent in the uplink is lower (26 ms) than the downlink (39 ms).

3.1.3 Dual Link

We developed a set of C++ applications using QT 4.7.4 libraries and threading to make the main and backup links work together as a dual link. We are using the Google Protocol Buffers library to serialize data and convert it into packets. The library can generate access classes in many programming languages, and it performs variable length encoding where some fields can be omitted at runtime. The serialization library converts the whole set of data acquired by the sensors into strings that we can send through the main link —using UDP— or through the backup link. There are two basic frames involved in the communications: the pod control frame and the pod state frame.

Figure 3.4 shows a timing analysis for a basic data transaction between the ground station and the kite. A data transaction starts when a pod control frame is sent from the KCC to the PCC. After receiving the request for data, the PCC replies with the pod state frame. The main link is represented by blue arrows and the backup link by pink arrows. The time taken for each frame is indicated in the vertical direction. At t_0 the KCC sends the request for data frame using both links. The PCC receives each message and replies also through both links. In this scenario both requests are replied through both links resulting in a total of six messages traversing the link (uplink plus downlink).

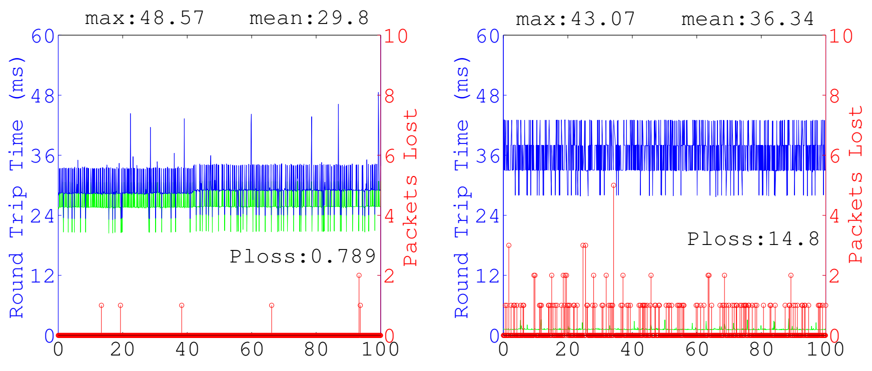


(a)

Figure 3.4: Timing sequence for data exchange between KCC and PCC through the dual link.

At t_1 the reply from the PCC, through the main link, is received at the KCC. At t_2 we show the reception of two messages, one from the main link and another from the backup link. The timing is not exactly the same for both paths due to the different frame size, however, they complete the round trip in an average less than 37 ms, below the 50 ms latency limit for an update rate of 20 Hz. We have the requirement to maintain a 20 Hz update rate which is not possible to meet for messages that only use the backup link.

Figure 3.5 shows the average delay for combined modes where the main and backup link are used in the same transaction. These modes could support 20 Hz as their average delays are 29.8 ms, and 36.3 ms. Even the maximum delays are lower than 50 ms. In practice, however, the main link requires a connection and soon after the uplink or the downlink fails the connection is lost. As a consequence the combined modes do not last longer than a few seconds.



(a) (b)

Figure 3.5: Dual link round trip time during 100 seconds. The blue line shows the total round trip time in milliseconds. The green line is a time stamp when the packet was received at the PCC. The red stems are packet losses. Subfigure (a) shows the round trip time when the frames go up through the main link and down through the backup link. Subfigure (b) shows the round trip time when the frames go up through the backup link and down through the main link.

The backup link cannot keep up at 20 Hz. To reduce the backup link usage we tried two approaches. The first one was to wait for a reply over the backup link before using it again to send a frame. In this mode we had some packets sent as frequently as in the main link (whenever the delay is lower than 50 ms) and some other packets sent every 100 ms (whenever the backup link was not ready to handle more frames. This mode uses the backup link as much as possible, however the variation between frames is uneven and unpredictable. The second option was to set the backup link at a frame rate of 10 Hz. In this case only the odd numbered frames are sent or replied by both computers. We selected this mode for the final version of the link and explore the effects of a lower frame rate in Section 3.6.

Figure 3.6(a) shows a data exchange through the main link. This is the normal mode of operation. Duplicated frames received at a later time through the backup link are ignored. The data is old and has already been received through the main link. Figure 3.6(b) depicts the case where the frame replied by the PCC, through the main link, is lost (dashed blue line). In this case the most important data is received through the backup link at t_2.

Both links are working simultaneously. When the main link operates

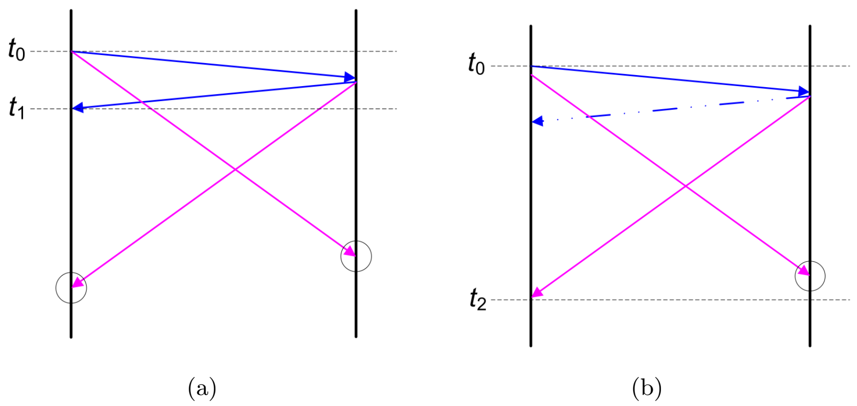


Figure 3.6: Timing sequence for data exchange between KCC and PCC through the dual link. The blue lines show data going through the main link. The pink line shows the data going through the backup link. The circles show discarded packets that have outdated information received earlier through the main link. Main link data exchange (a) and combined main and backup link data exchange (b).

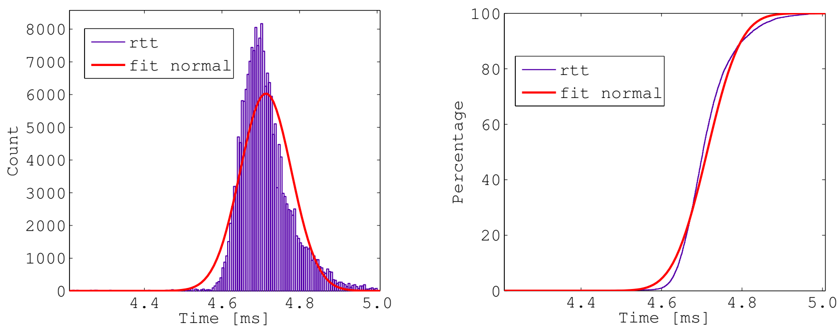
normally we have a low round trip time with a 20 Hz update rate. There are occasions when the router disconnects —sometimes because of radar detections, sometimes for unknown reasons— and it has to perform CAC on other frequencies before connecting again. For these periods the backup link is used to communicate with the kite. Combined modes occur when some packets are lost, but, since 802.11n requires a connection, as soon as the uplink or downlink stops working, the connection is lost. In the next section we build a model for the delay on each link to use later to simulate the effects of the dual link on the autopilot controllers.

3.2 Delay Models

As described in the previous sections we have two wireless links working together. The delay measured in each link is independent of the distance and mode of operation, but it is clearly different depending on the round trip path that a packet takes. Packets normally go through the main link, but when failures occur packets may use a combination of main and backup link. In both the main and backup link we used normal distributions to fit the data.

3.2.1 Main link

We used a long run of 5,200 seconds and fitted the data to a normal distribution. The histogram for the round trip times is shown in 3.7(a), along



(a) (b)

Figure 3.7: Probability distribution (a) and cumulative density function (b) for the main link.

with the fitted normal distribution in red. The fit in this case considers also the periodical high round trip times that are distributed differently. Figure 3.7(b) shows the CDF for the data, and for the fitted normal distribution.

The parameters for the normal distribution are: $\mu = 0.00471323$, and $\sigma = 0.0000661047$.

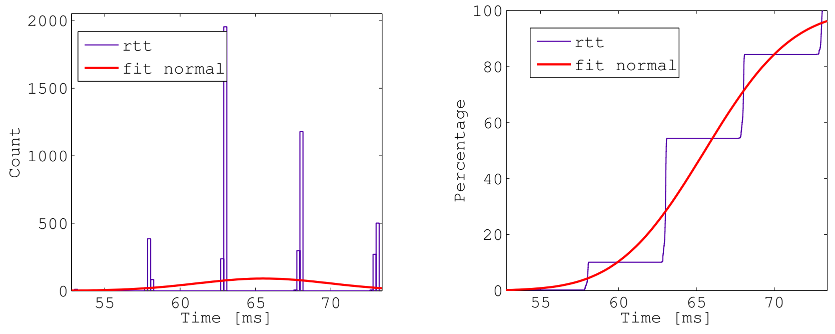
3.2.2 Backup Link

For the backup link we use one of the sample sets of 200 seconds. The histogram for the round trip times is shown in 3.8(a), along with the fitted normal distribution in red. Note that the CDF in Subfigure 3.8(b) shows clear steps. These are caused by the UART being polled every 5 ms.

Figure 3.8(a) shows a normal fit for the round trip time values over the backup link. To implement this model we use a function that rounds the delay values to the time of the polling events which occurs every 5 ms. Any reception that occurs after a polling event is rounded up to the value of the next polling event. In this way a normal distribution model can be used to generate round trip time values as if they were obtained with a polling mechanism. The parameters for the normal distribution are: $\mu = 0.0651851$, and $\sigma = 0.004665$. Further validation and error measurements of the model are presented in Chapter 4.

3.3 Packet Loss Models

We base our work on the paper by Lemmon [19]. The objective is to develop a model that can be used to simulate the statistical properties of packet losses on our system. In the case of delays we use a random variable to generate round trip times with a certain distribution and mean value. In



(a) (b)

Figure 3.8: Probability distribution (a) and cumulative density function (b) for the backup link.

this case we want to model the statistical properties of consecutive successful transmissions and losses. After obtaining this model we can compare the statistical properties of the model with reality and other models.

We apply a bit error rate model, which is used in literature to model bit errors in wireless channels, to model packet losses at the application level. Our dual link uses different channels and modulation techniques to transmit data in the most efficient way (like FHSS in the case of the backup link). We do an analysis on how packet losses are related to distance, but more data is required to include this relation into the model.

In this section we apply the two simplest instances of the Gilbert-Elliot bit error rate model to the packet loss rate in the dual link. In the simplest form, the Gilbert-Elliot model reduces to a Bernoulli random variable with a probability p_{loss} of loosing a packet. This is a memoryless model where losses are independent of each other. We compare this model to the Simple Gilbert model that has memory; the next state is related to the current state, or the probability of receiving a packet now depends on the previous outcome of the transmission event.

Matlab includes a function to calculate the transition probabilities and another to generate sequences of states when giving a transition probability as an input. We used both methods —Matlab *hmmgenerate* function and the formulas presented in the literature study— to calculate the transition probability matrix with similar results. With the transition probability matrix we can estimate the percentage of time that will be spent in each state in the long run by solving for the steady state vector π.

3.3.1 Losses in the main link

In the main link we can assume that packet losses are independent of the flight mode and of the distance between system components. Table 3.4 shows the estimated packet loss rates for different data sets. These values were taken from the longest data sets that we sampled.

Log file	Sample size (seconds)	p_{loss} (per thousand)
June 22, 12:10:07	4800	27.5
June 22, 13:32:39	900	0.43
June 23, 12:35:23	1000	0.59
June 23, 12:54:32	7800	0.057
June 23, 15:14:19	2100	0.023

Table 3.4: Main link packet loss rate.

In the first case a router disconnection occurred. There was a 180 seconds failure of the main link, possibly triggered by a radar detection event, where the backup link was used to control the kite and avoid a fast landing event. We use the Bernoulli model only since the error rates are very low and most of the times related to disconnection events.

The Simple Gilbert model calculation results in a matrix with no probability to go from good to bad state because of the low packet loss rate. We are using data series from flights where no disconnections occurred. We do not have enough flight data to get a statistic of how often disconnection events occur. Disconnection events have occurred sometimes once per flight test with the latest settings.

The disconnection model is implemented in the simulator as a disconnection event that disables the main link for a selected number of seconds. This model fits the behaviour defined by DFS and can be triggered at any time by the user.

3.3.2 Losses in the backup link

Figure 3.9 shows a set of measurements performed at different distances with each of the flight modes. We disabled retries to obtain the probability of a single transmission. There are higher packet losses on parking mode. This is the opposite of our observation when performing the initial assessment of the X-Bee wireless link, where losses were higher when flying figures of eight. This could be related to the antenna alignment. We tried to find a correlation between the tension in the tether and packet losses, assuming that when the tension is higher the alignment is better, but it did not show conclusive results.

With fading channels, the losses increase with the distance. However, we do not have enough information on how the packet losses increase as a

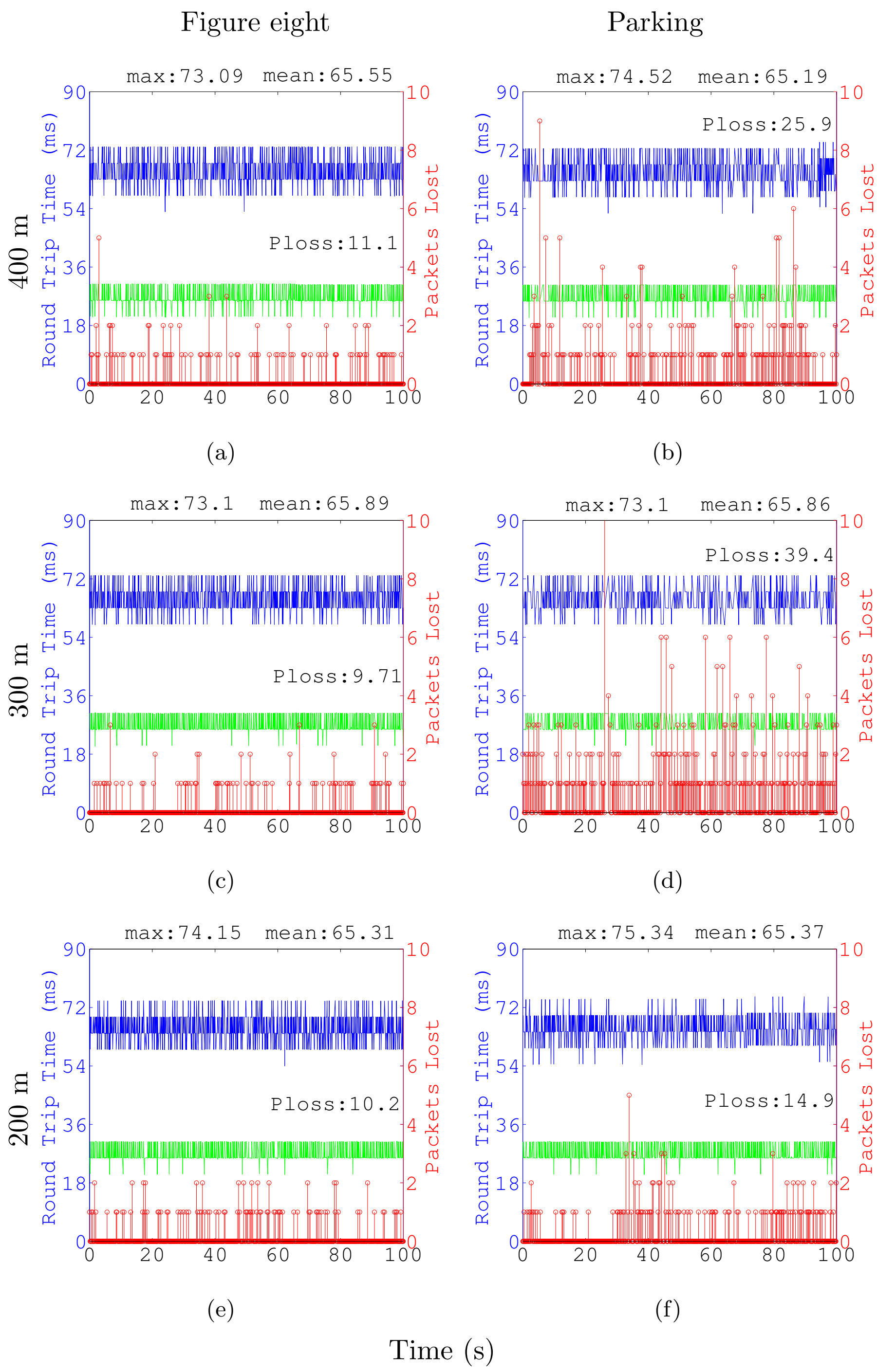


Figure 3.9: Backup link at 500 kbps data rate, single transmission, results. The blue line is the round trip time, the green line is the time stamp at the PCC, and the red stems are packet losses.

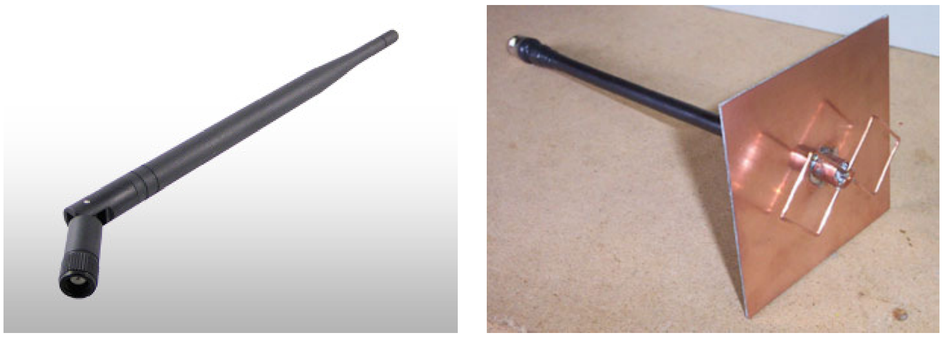

(a) (b)

Figure 3.10: Dipole (left), and Biquad (right) antennas.

function of the distance so we simplified the analysis by using the worst case of parking at 400 m to derive the model. Hence, the model is not related to the distance or the flight mode of the kite. The probability of loosing a packet is $p_{loss} = 0.258$.

$$P_{transition} = \begin{bmatrix} 0.776 & 0.223 \\ 0.639 & 0.360 \end{bmatrix}$$

$$\pi^T = \begin{bmatrix} 0.7411 & 0.2589 \end{bmatrix}$$

$P_{transition}$ was estimated using the average length of the error sequences and the loss probability, and with Matlab's *hmmgenerate* function, with similar results. It is evident that the steady state vector corresponds, obtaining by calculating the eight power of $P_{transition}$, corresponds to p_{loss} for the bad state and $1 - p_{loss}$ for the good state. Details on the model error and validation are presented in the simulator chapter.

3.4 Backup Link Range Test

To test the range of the dual link we obtained permission to fly the kite at 1 km. During that test the wind speed was very close to the limits of the system so we performed the test in parking mode at 730 m. At this distance we measured packet losses on the backup link and signal strength on the main link. We took this opportunity to compare between a dipole antenna and a home-made biquad antenna. Figure 3.10 shows both devices.

We configured each device (transmitter and antenna) to achieve the highest allowed transmission power. For 2.4 GHz the limit is 20 dBm, and for 5.5 GHz it is 30 dBm. EIRP is calculated as follows:

$$EIRP[dBm] = Transmitterpower[dBm] - cableloss[dB] + antennagain[dBi]$$

We considered a low loss on the cable of -0.6 dB. We reduced the transmission power of the Laird devices to stay within the maximum transmission range.

Device/Antenna	Tx Power	Cable Loss	Antenna Gain	Total	Direction
Laird/small dipole	17 dBm	-1.4 dB	3 dBi	18.6 dBm	downlink
Laird/big dipole	11 dBm	-1.4 dB	7 dBi	16.6 dBm	uplink
Laird/biquad	8 dBm	-1.4 dB	12 dBi	18.6 dBm	uplink
Nanostation	14 dBm		13 dBi	27 dBm	uplink
Compex	17 dBm	-1.4 dB	9 dBi	24.6 dBm	downlink

Table 3.5: Packet loss for dipole and biquad antenna.

In this case a higher antenna gain improves packet reception only. Data packets over the downlink are bigger, and packet losses are more on the downlink as shown in Chapter 3. Increasing the antenna gain for the receiver by 5 dBm decreased packet losses ten times as shown in Table 3.6.

Antenna	Gain	p_{loss}
Dipole	7 dBi	64.9%
Biquad	11 — 12 dBi	6.42%

Table 3.6: Packet loss for dipole and biquad antenna.

3.5 Backup Link Retries

The Laird modules include a retransmission option to repeat frames when acknowledges are lost. To decrease packet losses during normal operation we set the retries to eight. Figure 3.11 shows the results for parking and figure-of-eight modes at 300 meters.

Packet losses are 3.4 % in figure-of-eight mode, and zero for parking mode. Retries had an impact on the round trip time, which has an average of 77.59 ms and a standard deviation of 13.7 ms. There is a trade-off between the round trip time and packet losses. When enabling retries, the average round trip time increases, even if the first transmission succeeds, because it has to wait for the reception of an acknowledge. Standard deviation also increases because packets may be received at the first through seventh retry.

3.6 Controller Test

Two automatic flight controllers were tested under different dual link conditions. The first controller is a three loop non-linear controller called 3LAP

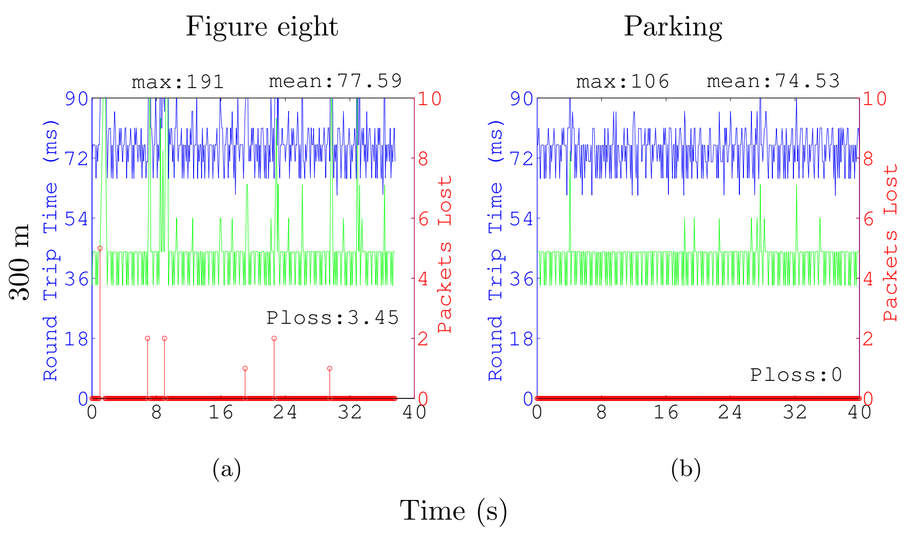


Figure 3.11: Backup link at 500 kbps data rate, 8 retries. The blue line is the round trip time, the green line is the time stamp at the PCC, and the red stems are packet losses.

[18]. The second controller is simpler, it sets n points to be reached by the kite without specifying the trajectory, hence its name, n-point. The input to n-point are desired points that the kite is heading to, following any trajectory. For 3LAP, the input is a specific trajectory to be followed by the kite. Both controllers work at 20 Hz, and whenever there is a packet loss they use the latest value available. The objective of the test is to assess the effects of delay and packet loss on each controller. We tested the 3LAP controller with default gains and the optimal gains found during the tuning process to identify the performance difference when working with different link performance conditions. A single cycle of figure of eight was executed over the main link and over the backup link.

The pink line show the desired trajectory of the kite. The blue line plots the measurements obtained from the GPS unit attached to the kite. In Figure 3.12 we can see the kite flying a little on top of the desired path, with a small overshoot in the lower curves of the path. This is one cycle of operation using the main link.

When the backup link is used the inputs from the PCC are received at 10 Hz and are subject to an average delay of 65.5 ms. In this particular test we had no packet losses. The controller performance decreases as shown in Figure 3.13 where the kite is not able to follow the desired trajectory. This flight path uses more steering inputs than the desired trajectory. it will deplete the batteries faster and is a heavier load to the motors. Controller designers will study optimized trajectories for varying wind speed conditions which have to be followed as close as possible.

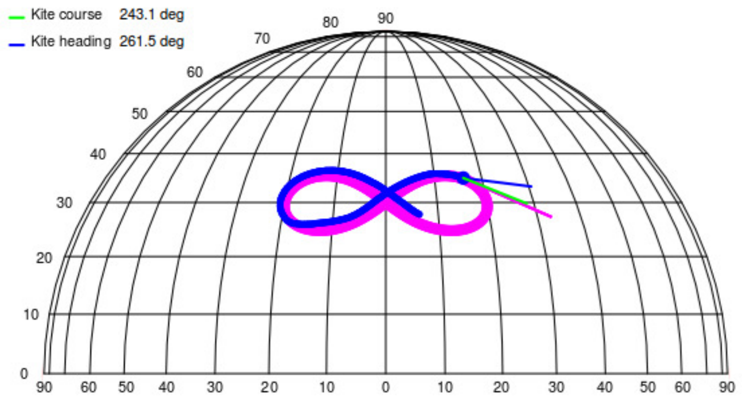


Figure 3.12: 3LAP tested using the main link (20 Hz update rate).

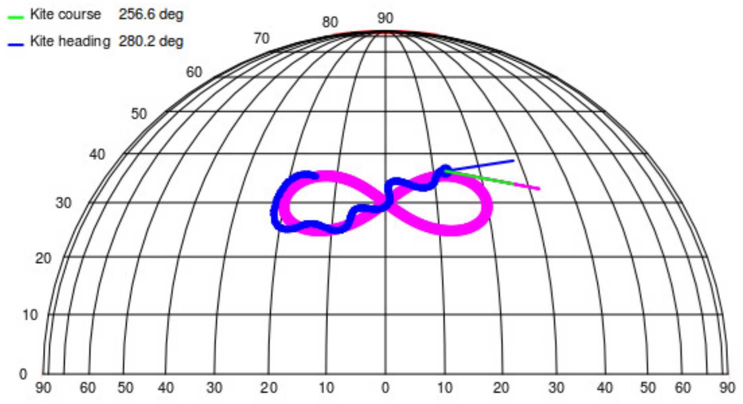


Figure 3.13: 3LAP tested using the backup link (dipole antenna, 10 Hz update rate, $p_{loss} = 5.8\%$).

For the case of n-point, the pink dots represent positions where the kite is attracted to. Once the kite is close enough to one of the points it selects another point and steers towards it approximating a figure of eight. Figure 3.14 shows a segment of the cycle when operating over the main link. The trajectory (the blue line) looks fully connected, with no errors, and following a smooth path.

Figure 3.15 shows n-point working over the backup link with the biquad antenna. The blue trajectory is not connected as a solid line because of a lower update rate in the backup link and lost packets. In this case we have 78 ms average delays and a packet loss rate of 5.8%. The n-point autopilot works correctly even with the lower QoS of the backup link.

The system is subject to other sources of delay and constraints like the maximum rate at which the steering lines can be pulled, however we can see how the communications system parameters affect the controller perfor-

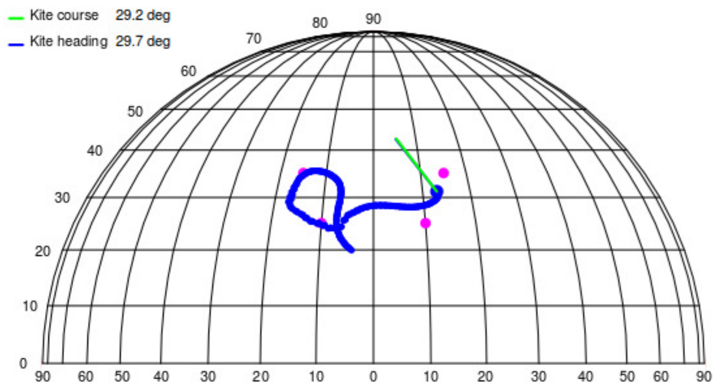


Figure 3.14: N-point tested using the main link (20 Hz update rate).

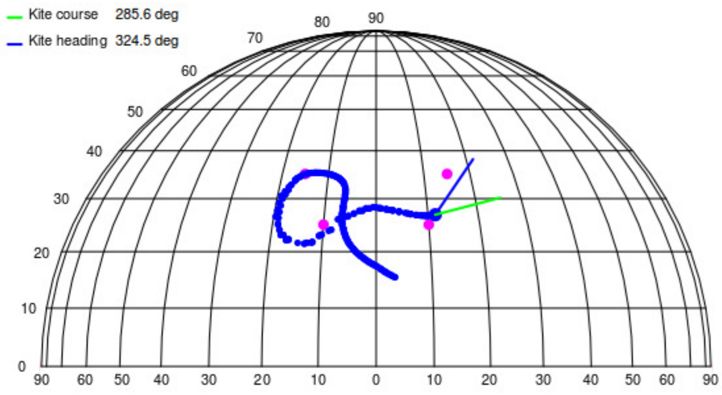


Figure 3.15: N-point tested using the backup link (biquad antenna, 10 Hz update rate, $p_{loss} = 5.8\%$).

mance.

Figure 3.16 shows n-point working over the backup link with 78 ms delays but with higher packet loss rate this time. When the backup link is loosing 57% of packets the GUI updates are slow and the autopilot failed to keep the kite in the air. During this test a human pilot had to take control of the kite to avoid crashing.

We can identify three operating modes for the dual link:

- Full mode: 4.7 ms average round-trip-time, at 20 Hz, packet loss rate close to zero (using the main link) where 3LAP can be used.
- Backup mode: 78 ms average round-trip-time, at 10 Hz, packet loss rate less or equal than 5.8 % (using the backup link) where n-point can be used.
- Critical mode: 78 ms average round-trip-time, at 10 Hz, packet loss

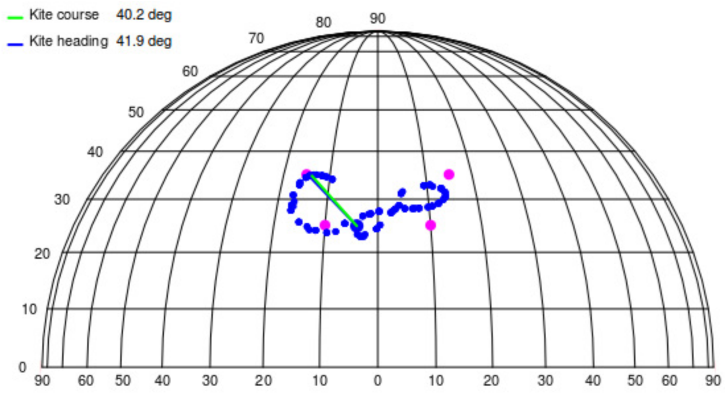


Figure 3.16: N-point tested using the backup link (biquad antenna, 10 Hz update rate, $p_{loss} = 57\%$).

rate greater than 5.8 % (using the backup link) where a parking autopilot running at the PCC can be used.

The difference between each operational mode depends on the performance of the controller at a given packet loss rate or amount of delay. The next step is to find out the limits of operation for each controller. The next chapter describes a simulation tool developed to explore controller stability while modifying the properties of the dual link. The goal is to find critical values for packet losses, delays, and sample frequencies, at which the controllers perform as expected.

3.7 Summary

We described the main design choices and performance results for the dual link during automatic flight control. We were able to implement a dual wireless with a 5.5 GHz, low latency link, that allows 20 Hz sample rate with an average round trip time of 4.7 ms. An extra backup link at 2.4 GHz maintains communications with the kite when the main link fails. With this configuration it is possible to run the basic autopilot controllers developed at ASSET. Three different operational modes for the dual link were defined based on QoS characteristics. For each mode we propose a controller that has proven to perform correctly under those conditions. Further experiments with the simulator are needed to identify the critical values for delay, packet losses and sample frequency that can be tolerated by the controllers.

The system evolved throughout the duration of the project. We presented the final configuration that provided the best performance in terms of independent delays, lowest disconnections due to radar detections (real or false positives), backup link antenna, and other settings. Further tuning is possible with longer data sets and more information on the backup link.

We presented statistical models for delay and packet loss. Such models are implemented in a dual link simulation software that allows control designers to test their controllers off-line and observe the effects of packet losses and delays. This will help to define the different levels of QoS that the controllers need to operate.

Chapter 4

Dual Link Simulator

Throughout the project we had limited access to the kite and limited time to debug the system in normal operation. Each test flight was fully scheduled with different tests, with the risk of rain and bad weather, and with long set up time to get into flight. A simulator will allow control designers to test controllers without having to be in the field with the complete system running. It will also allow researchers to explore different parameters for round trip times and packet losses and to anticipate to problems with the dual link, or to test other links —that provide different QoS— before implementing them.

The goal is to simulate round trip times and packet losses between the KCC and the PCC. The simulator will use the models obtained in the previous chapter to generate time delays and packet losses. In this chapter we present a description of the simulator and how it works, the implementation of the models in the dual link simulator, the error that our models have compared to the data from the flights, and the limitations of the simulator in its current state.

4.1 Simulator Description

The Kite Simulator software will simulate the aerodynamics of the kite and the network properties to allow controller designers to study different algorithms before testing on the field. This project is in progress and the kite model is not yet ready for complete testing. We implemented a link simulator class to simulate the dual link. The software runs in a computer with a wired connection to the KCC network and receives the pod state request frames from the KCC.

The simulator replicates the behaviour of the PCC. The interface to request and report data are exactly the same and the type of data reported is also similar. In this way it is possible to use the same monitoring and control software with the simulator and with the real PCC. The simulator uses UDP packets to implement the main and the backup links. Each link has different statistical models for delay and packet loss.

When a pod state request is received by the simulator, the simulator generates a success or fail event (based on the packet loss probability), and generates a round trip time value if the event was successful. For the main link we have a mean of 4.71 ms with a standard deviation of 0.0661 ms. Packet losses occur at a fixed rate of 0.059%. The simulator has a button to trigger radar detection events that disable the main link for a selected number of seconds. Whenever a packet is lost, the system uses the backup link, which has a mean of 65.2 ms, and a standard deviation of 4.7 ms. Packet losses in the backup link are implemented with the transition probability matrix described in the previous chapter.

The backup link operates at 10 Hz. It is also affected by polling intervals. This is implemented in the simulator as a rounding operation for the value obtained from the normal distribution. We use the polling period and an offset value to replicate the effects of polling. We are using an offset of 3 ms and a polling period of 5 ms. This results in values between 3 ms and 8 ms to be rounded to 8 ms.

4.1.1 Model Implementation

Delays are modelled as normally distributed variables with a specific mean and standard deviation. The model for packet loss is either a Bernoulli variable or a Simple Gilbert Markov chain. We will present the methods used to implement both models.

A normal random variable is distributed around a mean value depending on its standard deviation. In general we can generate values X with a specific distribution μ, σ:

$$X = \mu + \sigma Z$$

where Z is standard normal. We use a method known as the Box-Muller transform [8] to generate a unit standard distribution. This method requires two different random values with a uniform distribution. The normal values are calculated as follows:

$$Z_1 = \sqrt{-2Ln(U_1)}\cos(2\pi U_2)$$
$$Z_2 = \sqrt{-2Ln(U_1)}\sin(2\pi U_2)$$

For each pair of uniform random variables we obtain a pair of normally distributed variables. The functions required to implement these formulas are available in *stdlib* and *math.h*.

Packet loss can be modelled with a Bernoulli variable to implement a fixed loss rate, and with the transition probabilities of the Simple Gilbert Markov chain.

4.2 Model Validation

The simulator interface allows the user to select, for each link, what kind of model is used for delay and packet loss. The parameter for the models (mean, standard deviation, packet loss rate, and transition probabilities) can be specified by the user at runtime.

We used our models to generate round trip time traces and packet loss traces and compared them to the data obtained in the field. For round trip times, we estimate the error of our model calculating the Root Mean Square Error (RMSE) between simulator traces and real flight data. RMSE gives the error in the same unit as the measurement was performed. The smaller the value, the closer the series are to each other. The error is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(D_i - E_i)^2}{N}}$$

For packet losses, in the case of the backup link, we compared the CDF for the length of consecutive receptions and consecutive losses.

Trace	Flight 1	Flight 2	Flight 3	Flight 4
1	0.351	0.364	0.394	0.411
2	0.352	0.364	0.392	0.412
3	0.354	0.366	0.390	0.411
4	0.353	0.366	0.393	0.414
5	0.353	0.362	0.390	0.413
6	0.351	0.364	0.391	0.411
7	0.354	0.366	0.389	0.411
8	0.353	0.364	0.391	0.412
9	0.350	0.365	0.389	0.413
10	0.350	0.363	0.392	0.414

Table 4.1: Main link simulator RMSE values. This is the error (in ms) between ten simulator traces and four series of data taken during the flight.

Table 4.1 shows that for the main link, the highest error is 0.414 ms in flight sample 4. In the backup link (Table 4.2) the highest error is 6.9 ms. In both cases the error is around 10% of the mean value of the round trip time for that link.

Subfigure 4.1(a) shows the CDF for the length of consecutive successful packet transmissions, and Subfigure 4.1(b) presents the CDF for the length of consecutive losses.

Trace	Fig-8 400 m	Park 400 m	Fig-8 300 m	Park 300 m	Fig-8 200 m	Park 200 m
0	6.6	6.9	6.4	6.3	6.6	6.3
1	6.4	6.7	6.1	6.2	6.3	6.9
2	6.3	6.6	6.2	6.3	6.3	6.0
3	6.4	6.5	6.2	6.3	6.4	6.2
4	6.4	6.6	6.2	6.0	6.4	6.1
5	6.5	6.7	6.5	6.1	6.8	6.3
6	6.4	6.8	6.3	6.4	6.3	6.4
7	6.2	6.2	6.0	6.0	6.4	6.0
8	6.4	6.3	6.1	6.2	6.2	6.1
9	6.1	6.2	6.0	6.3	6.2	6.9

Table 4.2: Backup link simulator RMSE values. This is the error (in ms) between ten simulator traces and six series of data taken during the flight at different distances and flight modes.

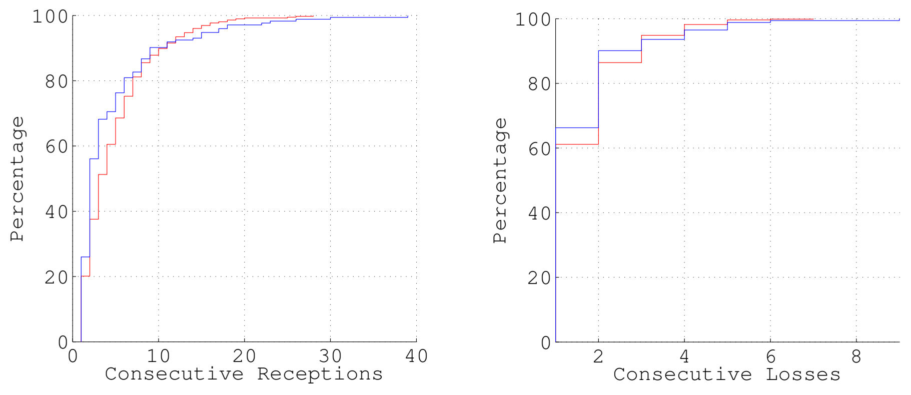
4.3 Simulator Experiments

In its current state, the simulator includes a simple point mass model that behaves like a kite and interacts with the KCC in the same way as the PCC. The simulator is missing a model for gravity, for the slow response of the motors that apply the steering and depower inputs, etc.

We tested the controllers in the simulator by setting the packet loss and delay properties to that of the dual link. Even with high packet loss rate the simulator shows a high stability. It is possible to use 3LAP and n-point to control the kite. The models in the simulator are too general and simple so it is now very difficult to get an unstable controller behaviour. Simulations are not useful yet for finding the critical packet loss rate at which the controllers stop working.

The results presented in this chapter are related to the round trip time model output only. This is only the value that will be used to generate a time delay before providing a reply. One option being considered is the QTimer library. The problem with this method is that the delay is specified in milliseconds. Another option is to use busy waiting, which can generate more accurate delays. Since the simulator runs in a different computer there is no problem with wasting CPU cycles.

It is also necessary to consider the delay between the KCC and the computer that is running the simulator. The computer running the simulator will be connected to the system's network with an Ethernet cable. In this case there will be a base delay (shorter than the wireless case) that has to be subtracted from the estimated round trip time value before executing the actual delay. This will be implemented in the future and the error between



(a) (b)

Figure 4.1: CDF for the length of consecutive packet receptions (a). CDF for the length of consecutive packet losses (b). The red line is the simulator output, and the blue line is real data acquired during flight.

the expected delay, and the real delay will also be measured.

4.4 Summary

Delays and packet losses were modelled using uniform random variables to implement a Box-Muller algorithm. Our models were validated by generating traces and comparing them to the data obtained in the field. The error of the delay models is 0.41 ms and 6.9 ms for the main and backup links, respectively, which corresponds to 10 % of the delays we need to generate. In the case of packet loss we can generate traces with similar distributions of success and error lengths.

We are working on improving the simulator to be able to use it to simulate the controllers under different QoS and determine the minimum requirements for delay and packet loss in the system.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Researchers require a fast and reliable wireless link between the KCC and PCC. We designed a dual wireless link that combines properties of WiFi and a low throughput radio operating at different frequencies as a single communication interface.

The dual link has a main link that uses a pair of 802.11n devices with MIMO antennas operating at 5.5 GHz to provide a low latency and low packet loss rate. The backup link is implemented with a pair of Laird modules operating at 2.4 GHz. Data packets are sent from the KCC to the PCC through both links. In normal operation packets are sent up and down through the main link. When the main link fails, the system uses the data from the backup link to keep the kite in the air.

The main link uses UDP packets over a WiFi connection. When the main link is used, uplink packets are 32 bytes long and downlink packets are 240 bytes long. The delay model is a normal distribution with mean of 4.7 ms and standard deviation of 0.066 ms. The loss model is a constant rate of 0.059 %. The maximum transmission power allowed for 5.5 GHz is 1 W (30 dBm). Due to the presence of radar devices on these frequencies there are some disconnection events that last between one and two minutes.

The backup link uses a UART interface. When the backup link is used, uplink packets are 32 bytes long and downlink packets are 120 bytes long. The delay model is a normal distribution with mean of 77.59 ms and standard deviation of 13.7 ms. The loss model is a constant rate of 3.4 %. The maximum transmission power allowed for 2.4 GHz is 100 mW (20 dBm) In this case we have no disconnection problem. The connection has a short set-up time but packet losses are higher. We use eight retries per transmission, with a trade-off of longer delay versus fewer packet losses.

A range test was performed at a distance of 730 m. Different antennas with 3 dBi, 7 dBi, and 12 dBi gains were tested. We selected the 12 dBi antenna that resulted in a 6.4 % packet loss rate (ten times lower than with the 7 dBi antenna). This antenna is heavy so it is only used in the KCC. Transmission power was adjusted to stay within the legal range, so the benefits are only for receiving packets at the ground station.

A simulator was built to estimate the effects of delays and packet losses. To provide realistic results an improved kite model is needed. The current version of the dual link simulator can generate a constant delay, it can model delays as a normal random variable, it can model packet losses with a Bernoulli variable (constant rate), and implement a Simple Gilbert model for packet losses for the backup link. We measured a 10 % error on the delays generated by the model.

The n-point and 3LAP controllers were tested in different dual link conditions. We defined three operation modes and estimated the QoS parameters that characterize them:

- Full mode: 4.7 ms average round-trip-time, at 20 Hz, packet loss rate close to zero (using the main link) where 3LAP can be used.
- Backup mode: 78 ms average round-trip-time, at 10 Hz, packet loss rate less or equal than 5.8 % (using the backup link) where n-point can be used.
- Critical mode: 78 ms average round-trip-time, at 10 Hz, packet loss rate greater than 5.8 % (using the backup link) where an parking autopilot running at the PCC can be used.

We are still fixing minor technical details but the Laddermill system operators are already enjoying the benefits of the dual link. Sometimes mode switches are not detected by the operators, allowing temporary main link failures without having to use the RC controller to get manual control of the kite.

5.2 Future Work

There are still questions to answer about the dual link. The next milestone of the Laddermill prototype is to achieve one week of continuous operation. This is a great opportunity to get more data to improve the models and find out how the weather affects the link (day, night, rain, snow).

An important goal is to find out the critical packet loss and delay limits for each controller. This can be done with an improved simulator or using the main link to replicate the backup link conditions —or any conditions from a delay/packet loss model— to find an accurate measurement of the delays and packet losses that the controllers can endure.

Bibliography

[1] The ADEOS Project, 2004. `http://home.gna.org/adeos/` retreived on Feb. 2012.

[2] ETSI EN 300 328 Electromagnetic compatibility and Radio spectrum Matters (ERM); Wideband transmission systems; Data transmission equipment operating in the 2,4 GHz ISM band and using wide band modulation techniques. Technical report, European Telecom. Standards Inst. (ETSI), 2006. `http://www.etsi.org` retreived on Feb. 2012.

[3] Linux RT patch, 2011. `http://www.kernel.org/pub/linux/kernel/projects/rt/` retreived on Feb. 2012.

[4] L. Abeni, A. Goel, C. Krasic, J. Snow, and J. Walpole. A measurement-based analysis of the real-time performance of the linux kernel. In *In Proceedings of the Real-Time Technology and Applications Symposium (RTAS02*, page 133, 2002.

[5] M. Ahmed, A. Hably, and S. Bacha. Power Maximization of a Closed-orbit Kite Generator System. In *50th IEEE Conference on Decision and Control and European Control Conference (IEEE CDC-ECC 2011)*, Orlando, FL, USA, 2011. IEEE.

[6] J. Altenberg. Using the realtime preemption patch on arm cpus.

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

[8] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[9] G. Bruzzone, M. Caccia, G. Ravera, and A. Bertone. Standard Linux for embedded real-time robotics and manufacturing control systems. *Robotics and Computer-Integrated Manufacturing*, 25(1):178–190, Feb. 2009.

[10] E. Callaway, P. Gorday, L. Hester, J.A. Gutierrez, M. Naeve, B. Heile, and V. Bahl. Home networking with ieee 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, 40(8):70 – 77, aug 2002.

[11] M. Canale, L. Fagiano, and M. Milanese. Power Kites for Wind Energy Generation Fast Predictive Control of Tethered Airfoils. *IEEE Control Systems Magazine*, (December):25–38, 2007.

[12] Scanlon W. Colandairaj, J. and G. W. Irwin. Understanding wireless networked control systems through simulation. IEE Computing and Control Engineering, vol. 16, issue 2 pp.26-31, April/May 2005.

[13] M. Erhard and H. Strauch. Control of Towing Kites for Seagoing Vessels. pages 1–10.

[14] L.M. Eriksson and M. Johansson. Pid controller tuning rules for varying time-delay systems. In *American Control Conference, 2007. ACC '07*, pages 619 –625, july 2007.

[15] L. Fagiano. *Control of Tethered Airfoils for High Altitude Wind Energy Generation.* Phd thesis, Politecnico di Torino, 2009.

[16] B. Houska and M. Diehl. Optimal control for power generating kites. In *Proc European Control Conference*, pages 1–14, 2007.

[17] A. Ilzhöfer. *Real-time Automatic Control and Estimation with Application to Power Generating Kites under Varying Wind Conditions.* Msc thesis, Ruprecht-Karls-Universität, Heidelberg, 2007.

[18] C. Jehle. Automatic Flight Control of Tethered Kites for Power Generation. MSc thesis, TU Delft, 2012.

[19] J. J. Lemmon and United States. *Wireless link statistical bit error model.* U.S. Dept. of Commerce, National Telecommunications and Information Administration, [Washington, D.C.] :, 2002.

[20] A. Matsumoto, K. Yoshimura, S. Aust, T. Ito, and Y. Kondo. Performance evaluation of ieee 802.11n devices for vehicular networks. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pages 669 –670, oct. 2009.

[21] P. Mieghem. *Performance Analysis of Communications Networks and Systems.* Cambridge University Press, 2006.

[22] J. Nilsson. *Real-Time Control Systems with Delays.* PhD thesis, 1998.

[23] C. Novara, L. Fagiano, and M. Milanese. Direct Data-Driven Inverse Control of a Power Kite for High Altitude Wind Energy Conversion. In *IEEE International Conference on Control Applications (CCA)*, Denver, CO, USA, 2011.

[24] T. K. Paul and T. Ogunfunmi. Wireless lan comes of age: Understanding the ieee 802.11n amendment. *Circuits and Systems Magazine, IEEE*, 8(1):28 –54, quarter 2008.

[25] M. Petrova, L. Wu, P. Mahonen, and J. Riihijarvi. Interference measurements on performance degradation between colocated ieee 802.11g/n and ieee 802.15.4 networks. In *Networking, 2007. ICN '07. Sixth International Conference on*, page 93, april 2007.

[26] Ludovici F. Ordine A. Salsano, S. Definition of a general and intuitive loss model for packet networks and its implementation in the netem module in the linux kernel, 2010.

[27] R. Schmehl. Kiting for wind power. Wind Systems Magazine, No. 7 pp.36-43, 2012. `http://www.kitepower.eu/images/stories/publications/schmehl12b.pdf`.

[28] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. Sastry. An lqg optimal linear controller for control systems with packet losses. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 458 – 463, dec. 2005.

[29] Agentschap Telecom. Vergunningsvrije radiotoepassingen. Technical report, Agentschap Telecom, 2012. `http://www.agentschaptelecom.nl/`.

[30] Y. Tipsuwan and M. Chow. Control methodologies in networked control systems. *Control Engineering Practice*, 11(10):1099 – 1111, 2003. Special Section on Control Methods for Telecommunication.

[31] C. Wewetzer, M. Caliskan, K. Meier, and A. Luebke. Experimental evaluation of umts and wireless lan for inter-vehicle communication. In *Telecommunications, 2007. ITST'07. 7th International Conference on ITS*, pages 1–6. Ieee, 2007.

[32] P. Williams. Optimal wind power extraction with a tethered kite. In *AIAA Guidance, Navigation, and Control Conference*, number August, Colorado, 2006.