ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Adrian Friedli

Steering an autonomous Weather Balloon

Semester Thesis
Spring Semester 2013

Tutors: Samuel Welten, Jara Uitto
Supervisor: Roger Wattenhofer

Abstract

This report documents the work on an autonomously flying glider. A glider, which may be used to recover equipment attached to weather balloons. It builds upon an earlier student project.
The hardware part has been improved and consists of two servos connected to an Android phone. Each servo rotates a wheel, which then winds up a string in order to steer the glider. The software part has been rewritten and is an Android app running on the phone.
Flight tests have been performed by launching the glider from a viewpoint tower and from a helium balloon and the results have been analyzed.

Acknowledgements

I would like to thank my tutors Samuel Welten and Jara Uitto for their support and Prof. Dr. Roger Wattenhofer for giving me the opportunity to write this semester thesis in the Distributed Computing Group.
Furthermore I would like to thank the following people who helped me besides my tutors with the experiments: My parents Silvia and Paul Friedli, Angela Botros, Silvio Frischknecht and Klaus-Tycho Förster.

Contents

Chapter 1

Introduction

1.1 Weather Balloons

Weather balloons are employed by meteorologists to perform radio-soundings. To accomplish that the balloons are equipped with sensors to measure temperature, humidity, wind speed and sometimes ozone concentration at higher altitudes in the atmosphere. Sensor and GPS position data are transmitted using some radio channels to a ground control station. In Switzerland twice a day a weather balloon is launched in Payerne by MeteoSwiss. Worldwide more than thousand weather balloons are launched daily.
These balloons are made of natural or synthetic rubber. They are inflated by hydrogen, helium or methane. The gas is often kept in bottles whereas hydrogen can also be produced locally by a hydrogen generator. Hydrogen and methane are less expensive than helium, but since of their inflammable and explosive character, they are dangerous. Whereas helium is an inert gas and therefore it is safer to handle. Like methane, which is the main component of natural gas, helium is also drilled for in subterranean caverns. These gas fields need to be surrounded by radioactive rock producing the helium, hence helium is a rare natural resource. These gases have densities smaller than air, causing the balloon to lift.
The balloon, often with a diameter of about 2 m, lifts up to an altitude of about 35 km within 90 minutes. Due to the low air pressure at these altitudes, the balloon expands until it bursts. The radio sonde then falls to the ground with a parachute. MeteoSwiss does not actively recover used radio sondes, rather they pay a reward for every returned sonde. [1][2][3][4]

1.2 Gliders and Kites

A glider is an aircraft without an engine. There are many different types of gliders, their varieties range from simple paper airplanes over toy and model building gliders, hang gliders and paragliders, and sailplanes to the space shuttle[1]. The four forces acting on an aircraft are lift, drag, weight and thrust. Drag, the force directing oppositely to the flight direction, and lift, the force opposing weight, are generated by moving through the air. In a motorized aircraft drag is opposed by thrust. Because of the missing propulsion, a glider doesn't have thrust. A glider has to trade potential energy to kinetic energy by lowering its altitude in order not to slow down and not being able to generate lift anymore.
Hang gliding and paragliding are similar air sports with different equipment. Hang gliders have a wing consisting of a fixed frame, mostly of triangular shape, with a fabric spanned over it. The pilot lies in a harness under the wing having ones back directed to the sky. Paragliders don't have a frame, the wing consists of fabric and strings. The form of the wing is maintained by caverns filled with air. The pilot sits in a harness held by strings. There are four bundles of strings from the wing to the harness, two of them are to hold the pilot and two are for steering. Pulling a steering string causes the glider to turn into that direction, pulling both strings causes the glider to reduce speed. Paragliders are an enhancement of parachutes.
A kite is an unmotorized aircraft bound to an object on the ground with one ore more strings. Kites are similar to gliders as both not having an engine. A kite may stay in its position by using

[1] The rocket engines in the space shuttle are only used for liftoff, it lands on earth as a glider.

wind to generate lift. There are different types of kites, some have a fixed frame and some don't. Some are made for fixed flying, may reach high altitudes and have only one string. And some are maneuverable and have two or more strings, which can be used for steering. [5][6]

1.3 Motivation

The problem of the equipment attached to weather balloons in radio-sounding is that it falls uncontrolled to the ground with a parachute and may be hard to recover. The impact point of the radio sonde may be as far as 300 km away from the balloon's launch position. The recovery of measuring equipment would be significantly easier if, instead of a parachute, a paraglider could be used to autonomously fly to a programmable destination.
An earlier project [7] showed that steering a paraglider with a box of less than two kilogram in weight is possible. The hardware and software from that project was operational and could be therefore reused, but there was room for improvement.
The task of this project was to improve the existing hardware and software of the previous autonomous glider project. Especially the release mechanism of the box had to be made more reliable. From the software part we could use the algorithms, but the architecture needed a cleaner design.

1.4 Overview

Chapter 2 explains the design of the hardware that was built or improved and software that was written for this thesis, Chapter 3 describes the tests that were performed and explains their results and Chapter 4 concludes the thesis with a summary and proposes possible future work.

Chapter 2

Design

This chapter describes the hardware parts that were built or improved from the previous thesis. Furthermore it explains the software that was implemented.
In preparation of a flight with a real weather balloon one would have to equip our box (Section 2.1.6) with measuring equipment and configure the Android application (Section 2.2.1) with a target destination in the area of the estimated impact point[1]. After launch during lift up only the measuring equipment will do its work, until the balloon rose up that high that it bursts[2]. Then the release mechanism (Section 2.1.4) will open to loose the leftovers of the balloon and let the glider[3] (Sections 2.1.1 and 2.1.2) unfold. After flight could be stabilized the glider will steer (Section 2.1.3) to the previously configured landing point.

2.1 Hardware

There is a box attached to a glider. The box contains two servos for steering, a release mechanism for opening the glider, a smartphone, a battery and an I/O interface connecting the smartphone to the servos. The box has been built by a group of people writing a prior thesis.
Of the original box at first we only wanted to replace the release mechanism, because it was not working reliably. But due to the box's destruction, where we had to rebuild the box, we improved more parts of it (see Section 2.1.6). As a replacement for the model paraglider we also used kites, which are much simpler than the paraglider. An overview of the kite and the box is shown in Figure 2.1.

2.1.1 Paraglider

The original thesis used a model paraglider as shown in the upper half of Figure 2.2. This paraglider has four bundles of strings, which had to be attached to the box. Two bundles were attached directly to the box for bearing the load and the two others were connected to the servos for steering. The model paraglider was stable and good at bearing loads. However it had a lot of strings which made it difficult to pack and open properly, because they entangle more easily. The diameter of the model paraglider we used was about 2 m and its weight was 197 g.

2.1.2 Kite

For this thesis, two simple kites were used instead of the model paraglider. Our kites were similar to the model paraglider except they only had two bundles of strings. No bundles were connected to the box directly and the two available bundles were only connected to the servos. In contrast to the model paraglider, the kite's velocity can't be controlled, as it only has two

[1] Balloon Trajectory Forecasts: `http://weather.uwyo.edu/polar/balloon_traj.html`

[2] Detection of the balloon's burst has not been implemented yet. A barometer as available in newer Android phones may be used to detect the declining altitude, considering commercially available GPS receivers may not work in heights above 60 000 ft (=18.29 km) due to export restrictions of the DOD.

[3] In this report with "glider" we may refer to "model paraglider" or "kite" in the same manner.

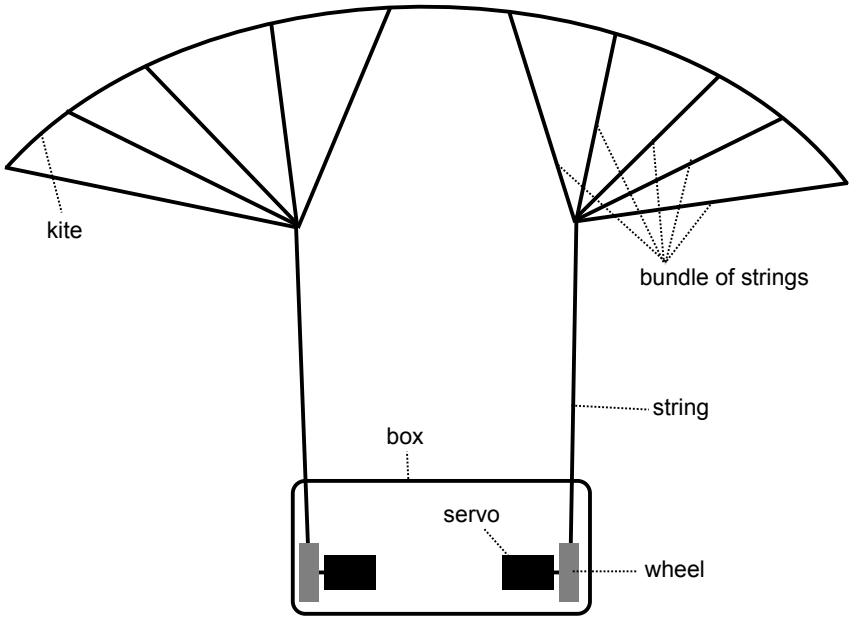

Figure 2.1: Schematic overview of the kite and the box with the servos.

bundles of strings. Because we wanted to keep things simple, we think it is a good idea not to control the speed.
We had a small kite with a diameter of 2 metres and a weight of 218 g and a large kite with a diameter of 4 metres. The small kite can be seen in the lower half of Figure 2.2 and the large kite in Figure 2.3. Although the large kite was much larger than the original model paraglider, the experiments showed that it has about the same capability of bearing loads. The advantage of this kite is that it has fewer strings, such that the risk of entangling them is much lower which made setting up the experiments much easier.

2.1.3 Steering Servos

A servo is an electric motor with some additional electronics to measure the current position and activate the motor to move into a target position. The value for the target position can be set using pulse width modulation. Our servos are controlled from the IOIO (Section 2.1.5).
The servos we used for steering are continuous rotation servos. For a specific range of values they have a position they can be moved to and held on. Usually they rotate for more than one turn in this value range. For values out of this range the servos continuously rotate in the corresponding direction with a speed depending on the value they are set to. For our application only rotating to a fixed position was used, we didn't make use of the continuous rotation function. In our application a wooden wheel is mounted on each servo. The wheel gets rotated by the servo and then winds up a string connected to it in order to steer the glider.

2.1.4 Release Mechanism

The original release mechanism had a rotating wheel, which wound up a string, which then pulled a bar. The bar was initially held in its position by a magnet. The original release mechanism was rather unreliable. To get it lighter and more reliable, we chose to build another release mechanism. The old release mechanism weighted about 70 g whereas the new improved one is only 27 g.
Our implementation of the release mechanism consists of two rings, a coupling, a piece of wire, a fabric strap and a small servo. In the closed state the rings hold each other and the small ring is locked by the coupling and the piece of wire. They are arranged so that most of the force is on the strap and almost none of it is on the piece of wire holding the ring. To open the mechanism, the servo pulls the piece of wire out of the coupling, which then releases the small ring. Then

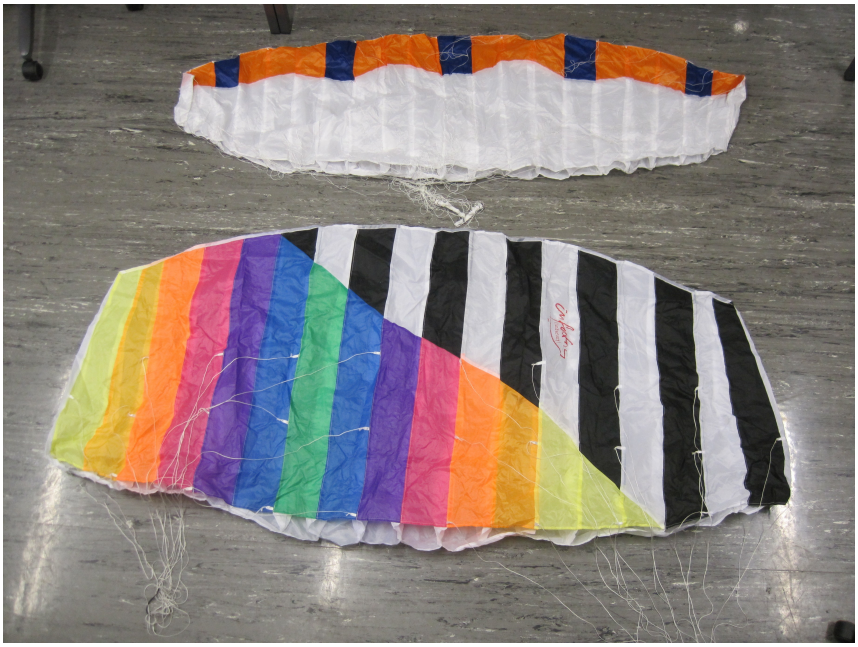

Figure 2.2: The model paraglider (top) and the small kite.

Figure 2.3: The large kite.

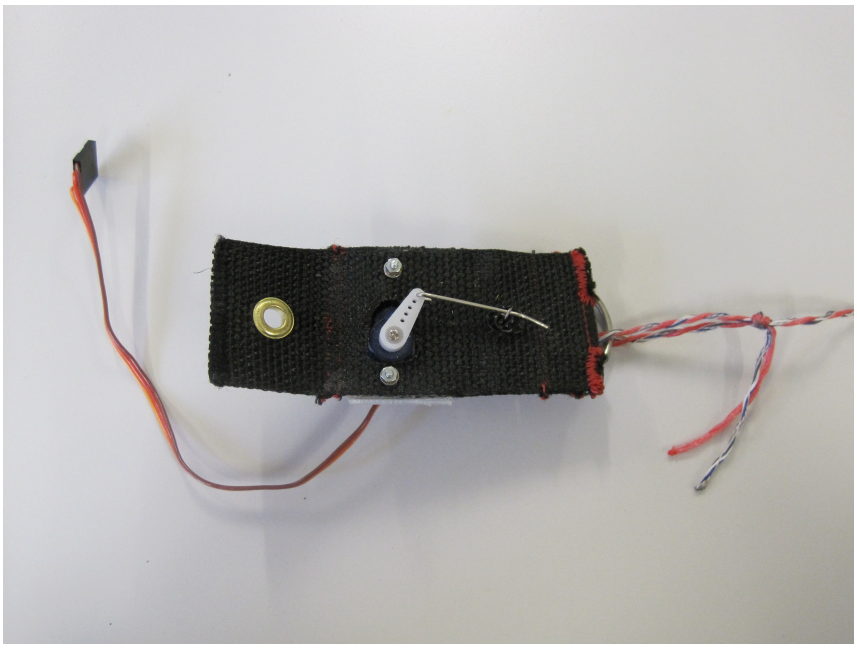

Figure 2.4: The release mechanism from the back side, closed.

when force is on the string, the large ring will move the small ring and get free. For the smaller ring, we at first chose a round shape but experiments showed, a rectangular shape to be more reliable, as less force to open is needed. See Figures 2.4 to 2.6 for details.

2.1.5 IOIO

For the mobile Android phone to be able to control the servos, IOIO was chosen as interface. IOIO is a small board allowing Android applications to use some Input and Output ports.
The IOIO has a USB-A receptacle, connected with a regular Type A to Micro-B USB cable to the mobile phone. IOIO uses Android's debug interface to communicate with the Android application. It also charges the phone's battery with its own battery through this connection.
The IOIO is powered by a 7.2 V model building battery, which it internally converts down to 5V.[4]

2.1.6 The Box

The original box was a plastic storage box, with the servos attached to a middle layer to an acrylic glass plate. The servos were screwed to metal angles which were screwed to the middle layer.
Since the original box was destroyed in an early experiment, a new one had to be built. We saw this as a chance for improvement. We chose to use the same type of plastic storage box. Instead of the acrylic glass, we used a lighter wooden middle layer, and screwed the servos with wooden bars (instead of metal angles) to the ground. This made the box much lighter than the original one. And because we screwed the servos to the ground instead of the middle layer, the middle layer is now free of any components, which makes it more easy to open and close the box to change something inside. The box ready for flight weights about 980 g, about 160 g of which are from the protection cushion below the box. Figure 2.7 shows the inside of the box, with the servos on each side, the release mechanism on the bottom left, the Android phone on the top right, the IOIO module on the top left and the battery below some connection board in the middle.

[4]The battery's charger should not be used to directly power the IOIO board for testing purpose. Because it outputs a much higher voltage than imprinted. After we destroyed two IOIO boards and burned one thumb, we measured the voltage of it and it had peaks at about 18 V.

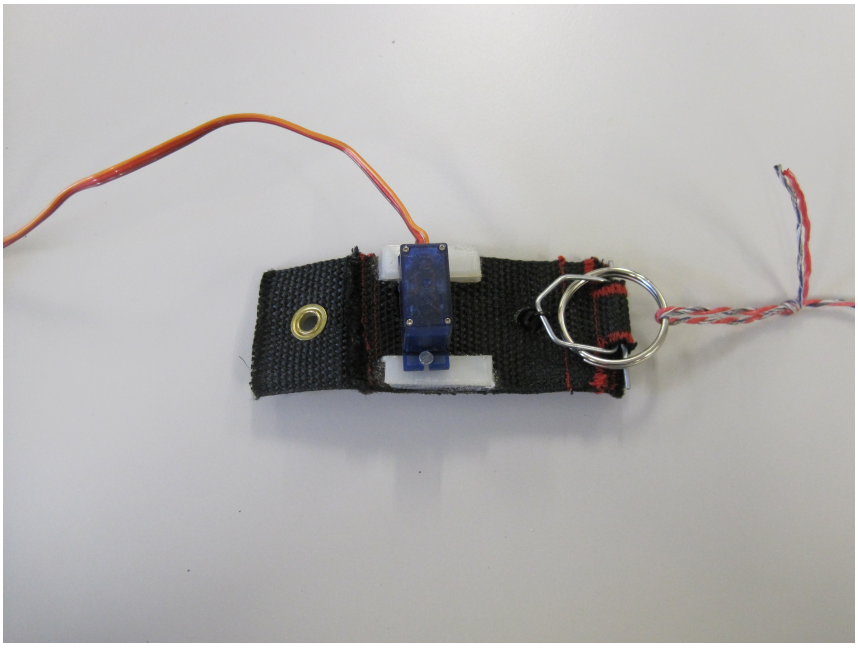

Figure 2.5: The release mechanism from the front side, closed . . .

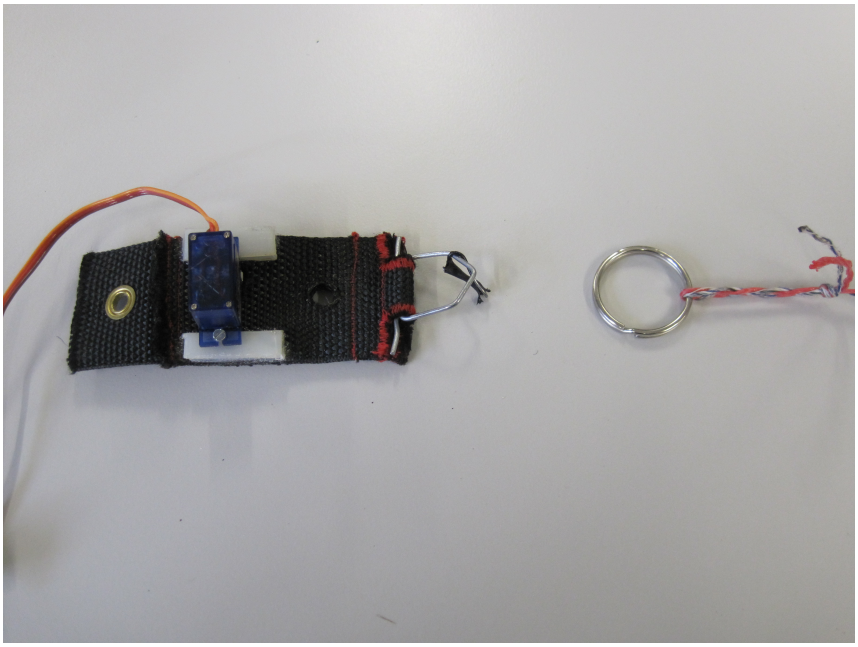

Figure 2.6: . . . and open.

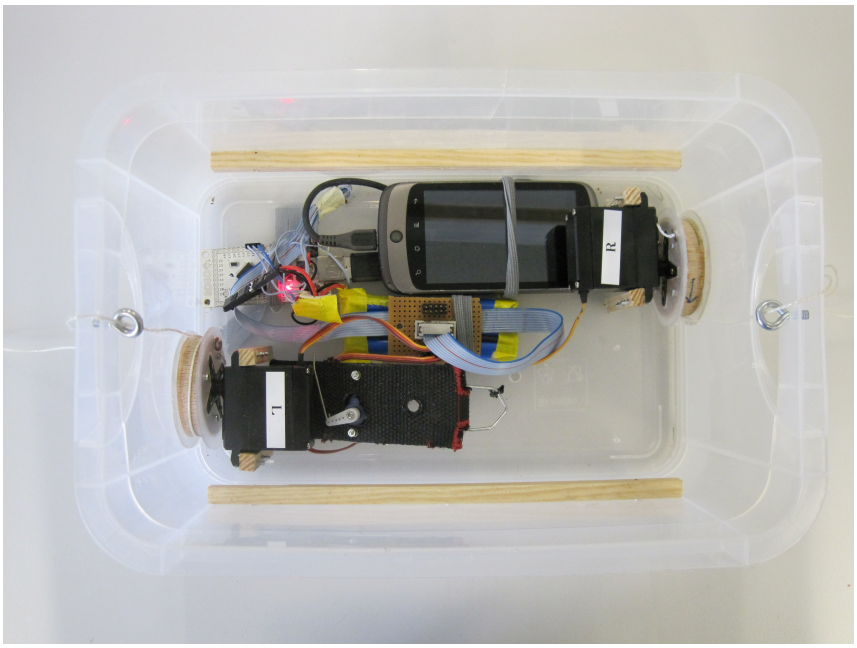

Figure 2.7: The box with middle layer and protection cushion removed, view from top.

2.2 Software

Because the software architecture from the previous thesis was not designed in a modular fashion and did not allow us to easily exchange or improve parts of it, we decided to completely take a new design. The software implemented for this thesis has been mostly written from scratch. Some algorithms were used from the previous thesis.

2.2.1 Android Application

The Android application is a software running on the Android phone in the box and controlling the glider. Java was the programming language used for it. The central part of the software implementation is the glider model. Controllers and views can subscribe as listeners to the model and get notified about changes to the model. Controllers may also change values in the model. The general design of our Android application can be seen in Figure 2.8 on page 17, where dashed lines show notifications to listeners and solid lines depict either reading (from the model) or calling functions (of other controllers).
Android leaves some application data structures like the glider model open and initialized to their old values, when the application is closed [5]. Therefore, a mechanism was implemented to re-initialize the data structures at the intended start of the application.

Glider Model

The central part of the design was the glider model, which stores states of the different parts of the glider application. Each controller and each view can be registered as a listener to changes in the glider model, or parts thereof. The model maintains a list of listeners. Controllers may write to the model, whereas on each change, the model takes care of notifying its listeners.

Flight Controller

The flight controller gets notified by the glider model about state changes. Using the sensor data it has, it will then calculate movements of the servos to either fly in a fixed target direction or a

[5] The Java process of the application is kept running in the background for some time and only the views get destroyed, this allows Android to open a previously closed application more quickly.

target direction calculated from a target position. It writes servo states to the glider model.
There are two parameters. $\mathrm{correctionTime}$ defines how often steering corrections should be performed and $\mathrm{pullingFactor}$ defines the magnitude of the corrections. Both parameters may be configured remotely during flight.[6]
A steering correction is done when new sensor data are available and at least $\mathrm{correctionTime}$ milliseconds have passed since the last servo move.
First an $\mathrm{adjustment}$ factor is calculated from the difference between target direction, and the orientation[7] of the box ($\mathrm{directionDiff}$) and the time since the last servo movement in milliseconds ($\mathrm{timeDiff}$).

$$\mathrm{adjustment} = \frac{\mathrm{directionDiff}}{\mathrm{timeDiff} \cdot 1000} - \mathrm{rotationSpeed} \tag{2.1}$$

If $\mathrm{rotationSpeed}$ is greater than a defined maximum of 90 degrees per second and the calculated $\mathrm{adjustment}$ is in that direction, that it would just increase the rotation speed, then $\mathrm{adjustment}$ is set to 0. With the goal to limit the rotation speed.
From the $\mathrm{adjustment}$ a relative movement of the servos is calculated.

$$\mathrm{servoMove} = \frac{\mathrm{pullingFactor} \cdot \mathrm{adjustment}}{2000} \tag{2.2}$$

The relative movement of the servo is then added to the current displacement to get a new displacement.

$$\mathrm{servoDiff} = \mathrm{servoLeft} - \mathrm{servoRight} + \mathrm{servoMove} \tag{2.3}$$

The new displacement is then trimmed to be in the range $[-1, 1]$.

$$\mathrm{servoDiff}_{\mathrm{trimmed}} = \min(1, \max(-1, \mathrm{servoDiff})) \tag{2.4}$$

The new states for the servos then get taken from the trimmed displacement.

$$\mathrm{servoLeft}_{\mathrm{new}} = \max(0, \mathrm{servoDiff}_{\mathrm{trimmed}}) \tag{2.5}$$

$$\mathrm{servoRight}_{\mathrm{new}} = -\min(0, \mathrm{servoDiff}_{\mathrm{trimmed}}) \tag{2.6}$$

Servo Controller

The Servo controller runs in an IOIO Service. Using a service instead of just a view was chosen, because now the application can be put in the background and is still running. The servo controller gets regularly called from within IOIO's main loop and then reads the servo states from the glider model and sets the IOIO output ports accordingly.

Sensor Controller

The sensor controller takes care of registering to the relevant Android sensors and updating the glider model with sensor data. We used an orientation sensor and a location sensor.
The orientation sensor is a virtual sensor provided by the Android framework, which calculates the device's orientation using different hardware sensors, depending which are available. On the device we used, the orientation was calculated by using a magnetic compass and accelerometers. Advanced Android phones also have gyroscopes included, which would give more accurate orientation data. The orientation data coming from the Android framework with a rate of about $18\mathrm{Hz}$ gets filtered by a simple low-pass filter of the form:

$$x_n = 0.9 \cdot x_{n-1} + 0.1 \cdot x_{\mathrm{input}} \tag{2.7}$$

From the filtered orientation data, the rotation speed is calculated. To allow the box to turn around more then once, the sensor controller internally stores the orientation in a float which would not get normalized. The float increases or decreases by 360 degrees for every turn.

[6] The remote control is described in Section 2.2.2.

[7] In the beginning we used the bearing reported by GPS for this purpose, but tests showed, that this was not suitable, so we switched to the orientation sensor, which is a magnetic compass in the case of our Android phone. See Section 3.7 for details.

The location sensor gets it's data from the GPS receiver of the Android phone, which provides amongst others longitude, latitude, height, bearing and speed.

2.2.2 Remote Control

The remote control allows us to change flight parameters, configure the autopilot and also manually control the flight direction, which one needs to being able to for autonomous aircrafts due to regulations. The remote control software is split into three parts, a server, a web client and the network controller as part of the Android application.

Server

The Server is a RESTful HTTP server written in Python. It communicates with the Android application and the web client using objects serialized into JSON. Additionally it also serves static files needed for running the web client.
The server maintains a glider state object and a flight control object, which can be accessed as RESTful resources. The Android application pushes updates of these resources to the server. The web client regularly polls these resources.
The web client sends control commands to the server which appends them to a command list, which is another RESTful resource. The Android application polls new commands regularly. For the Android application to be able to get only new commands, the server assigns each command an incrementing id and the Android application asks only for commands with ids larger than the id it had last seen.

Web Client

The web client comes as static files from the Server. It then uses JavaScript in combination with the JQuery library to communicate with the server. It regularly polls the glider state from the server and it puts control commands to the server when required.

Network Controller

The network controller is the part of the remote control that runs in the Android application. It runs in its own thread, so it would not block the execution of the rest of the application. It polls the server every second to get new commands and it receives updates to states from the glider model and sends them to the server at most once a second.
For the Android application to not execute all old commands, which may still be stored at the server, after it has been restarted, the Android application discards all commands it gets from its first successful query. In normal operation it will only discard an empty list. This allows us to restart or even update the Android application without restarting the server.
Restarting the server while the Android application is kept running is not supported.

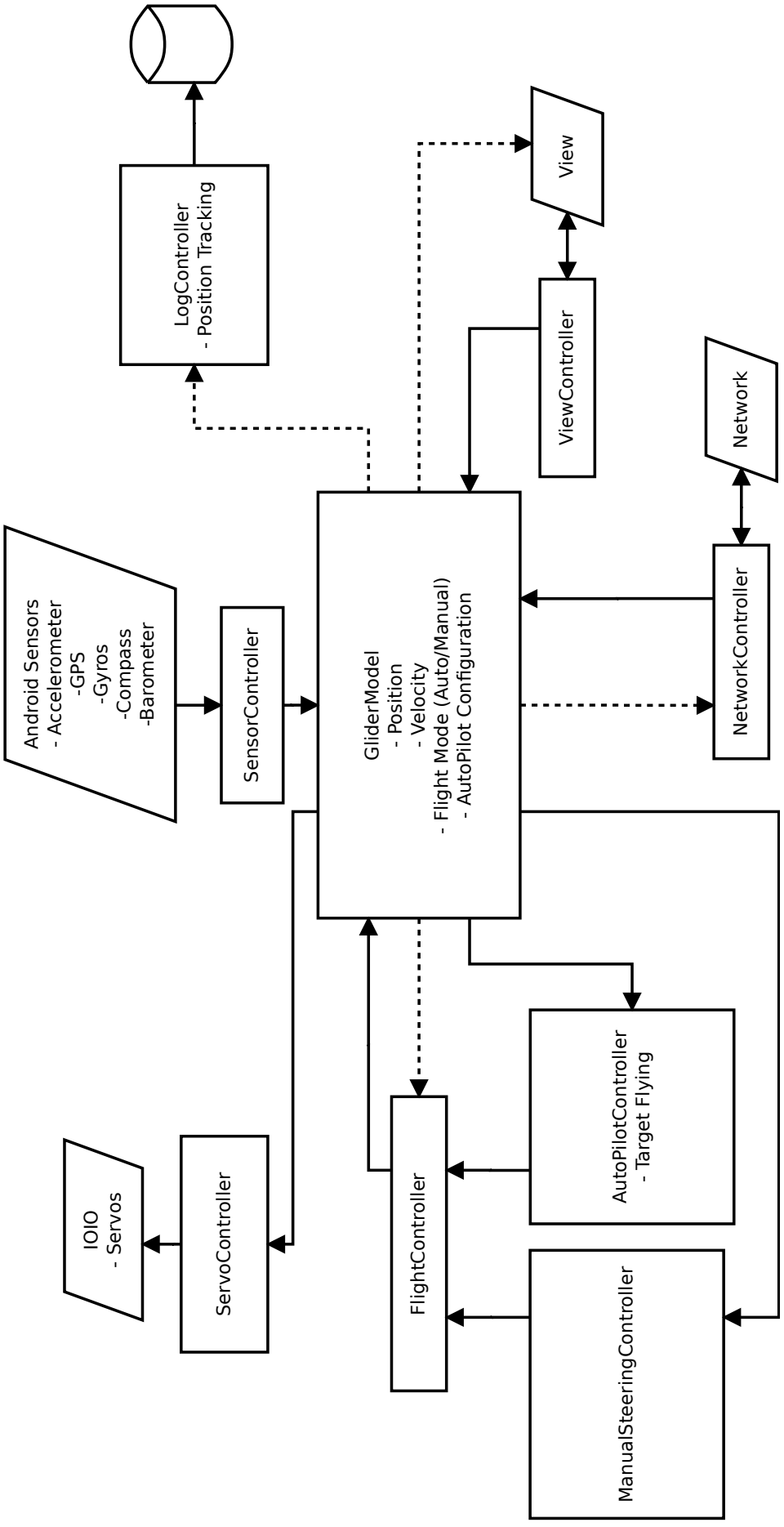

Figure 2.8: Overview of the software design.

Chapter 3

Experiments

After we did some preliminary experiments to compare the kites and the model paraglider (Section 3.1) and to test the release mechanism and the unfolding of the glider (Section 3.2), we performed some test flights from a viewpoint tower (Sections 3.4 and 3.5) and from a helium balloon (Sections 3.6 to 3.8).

3.1 Testing different Gliders

The model paraglider from the prior thesis was still available, and we additionally ordered two kites, one with a diameter of 2 metres and another with a diameter of 4 metres. To see how well these three gliders perform we did some tests to be able to compare their behaviour.

3.1.1 Setup

To test the model paraglider and the kites, we put a weight of about 1 kg to them and launched them from a window at the ETZ building. In some cases, we also tried different lengths of strings.

3.1.2 Results

The small kite with no extension of the strings[1] falls really fast and unstable. Adding a string of 30 cm length to the bundle of strings made it fly nicer, with 70 cm added to the strings it flies stable but still rather fast in comparison to the model paraglider and the large kite.
Looking at the results from the small kite tests, we recognized the need for longer strings here, too. We chose a length of 40 cm and added it to the large kite. It flew nice and slowly. This was sufficient for our purpose so no other tests were performed with the large kite for now.
The model paraglider flew best of all, but it was difficult to setup the experiment because of the large number of strings, which got entangled rather often.

3.2 Testing the Opening of the Glider

First we tested the new release mechanism on the ground while the Android phone was connected to the ETH's public wireless LAN[2] and the remote control server was running on the student's laptop. It has proven to open reliably. After that it had to be tested if the kite can unfold when thrown from some height.

3.2.1 Setup

In preparation of the experiment a rope was spanned from a viewpoint tower to the ground. The tower has a height of 30 metres. The box with the folded kite was lowered down the rope, so

[1] Just tying the weight to the point where the different lines of the kite join to a bundle.

[2] To not have to login at the landing page all the time, the Port 1194, which is not filtered in the public wireless LAN, has been used for the server.

it was further away from the tower, it then was about 20 to 25 metres above ground. The box was then released via the remote control interface. Where the Android phone was the access point, the server and the web client were running on a Nokia N900, which was connected to the Android phone via wireless LAN.

3.2.2 Results

The drop height of 20 to 25 metres showed to be too less for the kite to unfold properly, the box crashed into the ground and had to be fixed. Thus, no further release experiments were performed except the balloon experiments, where larger drop heights were possible (Sections 3.6 to 3.8).

3.3 Testing the Autopilot

In order to test the autopilot and see if it will output the correct commands to steer the glider in the direction we wanted, we configured the autopilot to a destination near our current location, then watched the steering commands while walking around. The commands showed to steer in the direction of the configured target point. Thus we conclude the autopilot is working correctly.

3.4 Testing Flight I

For the purpose of testing, if the kite can be steered at all and later to see if it also steers in the direction we want it to, we chose a high drop-off point, where we could manually launch the glider. The viewpoint tower at Pfannenstiel[3] proved to be good for this, as long as the wind was not coming from north. The tower is about 30 metres high and there is a large meadow in the north to east direction of the tower, where the glider could land safely.

3.4.1 Setup

Two people were standing on the viewpoint tower and held the large kite, with the box attached to it. They released the open kite simultaneously. As target direction 30°[4] was chosen. Just after releasing the kite, a command to start the flight controller was issued from the remote control. Earlier tests showed that the new, lighter box was too light for the large kite and it didn't fly well, so we had to add an additional weight of about 720 g to the box. We used GPS direction as an input for the flight controller, this showed not to be a good idea (more on that in Sections 3.6 and 3.7). The orientation sensor was only used to calculate rotation speed.

3.4.2 Results

In the beginning the glider steers right and does a turn to the right. As can be seen in Figure 3.1 the box is steering to the right until the target direction is almost reached after about 5 seconds (it starts turning left earlier, because of the high rotation speed to the right). And then it fully steers left just before landing.
We can see, that the height of 30 meters is just enough to do a bit more than a 180° turn. For further testing the flight controller we would need higher heights.

3.5 Testing Flight II

3.5.1 Setup

The setup for this test was the same as in "Testing Flight I".

[3] Pfannenstiel is a hill in the region of Zurich. Coordinates of the tower: 8.67338°E, 47.29076°N
[4] Target direction is given in azimuth.

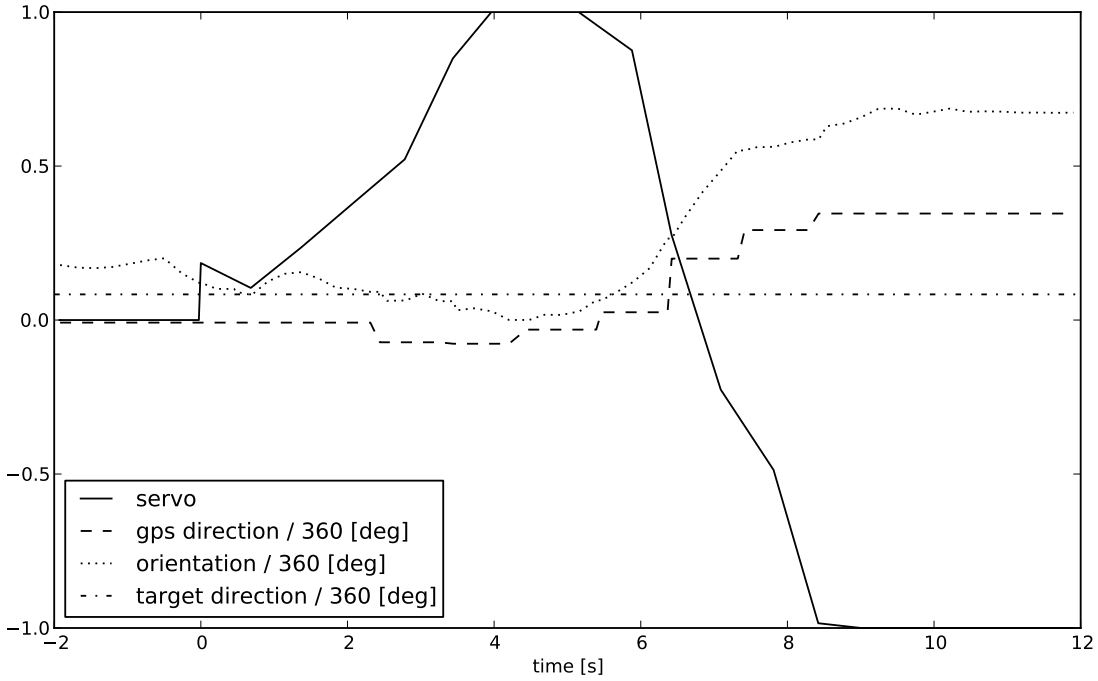

Figure 3.1: Data plot.

3.5.2 Results

There is a strong wind coming from west. The glider does not have any chance to change its flight direction, it just gets pulled along by the wind. The glider is steering to the left. As can be seen in Figure 3.2, there are two peaks (just right after the beginning and after about 8 seconds) where the glider is steering (or starting to steer) to the right, this is because of a large rotation speed to the left at these points.
This experiment has shown, that we can't control the glider if the wind is too strong. Also the box was heavily rotating, although the glider was flying straight the orientation sensors show rotations of about 180° (comparing values at seconds 4 and 7.5).

3.6 Experiment with a Balloon I

As we have learned from the experiment described in Section 3.4, we needed higher altitude differences to further test the flight controller. These experiments were performed using a tethered helium balloon to lift up the glider and the box.

3.6.1 Setup

A large balloon with a diameter of about 2 metres filled with helium was used to carry up the box with the glider. A string of length 100 metres was attached to the balloon, such that we could pull the balloon down and reuse it after each experiment.
Again the Android phone in the box was the access point and the remote control server and the web client ran on the Nokia N900.
For this test we wanted the glider to stabilize the tumbling first before it starts steering to the target direction. Also we only wanted to use meaningful flight direction data. Direction data reported by GPS may have an arbitrary value if the GPS receiver is standing still, as it needs movement to calculate bearing. For achieving this goals we changed the flight controller to ignore the direction difference of the target direction and the direction reported by GPS when

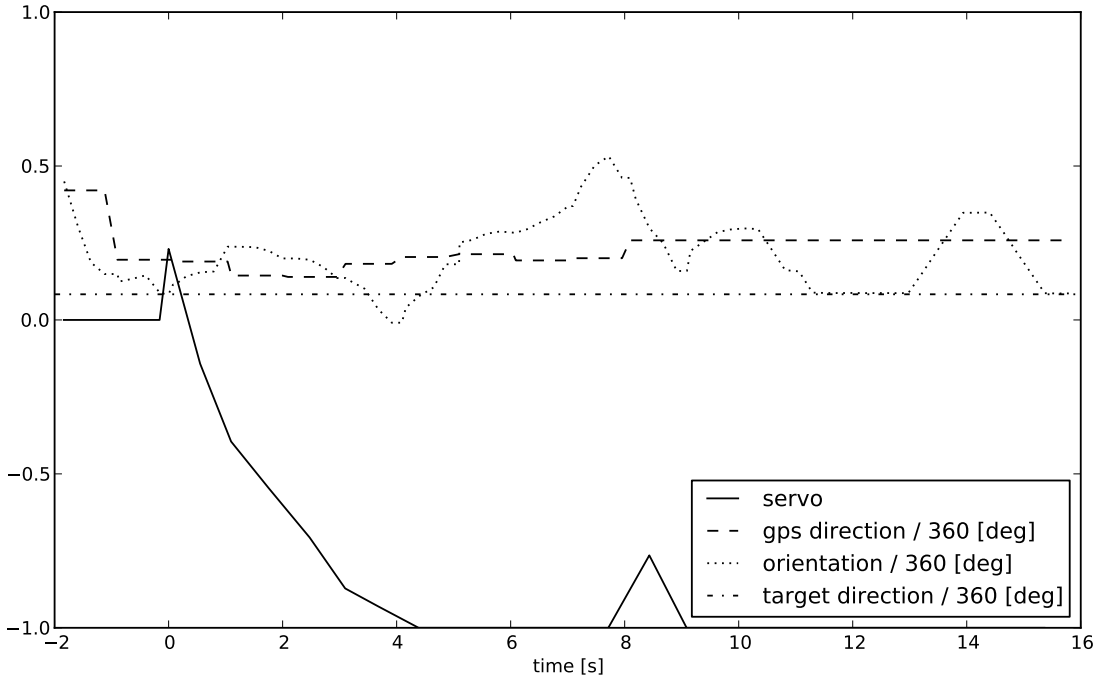

Figure 3.2: Data plot.

the speed reported by GPS was below 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$.[5] The configured target direction was 270°.
Because the box with the large kite, which had an additional weight to fly well, was too heavy for the balloon to lift up, we chose to use the small kite and removed the additional weight from the box for this experiment.

3.6.2 Results

Although the lines were twisted, the glider flew well. But the box was rotating heavily back and forth.
The data plot of this flight can be seen in Figure 3.3. At the beginning the glider steers left to counter the rotation of the box. It flew faster than 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$ at about 1.8 seconds, and it started steering to the right. It was flying slower than 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$ again at about 3.8 seconds where it then started steering to the left. Until again it flew faster than 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$ and started steering to the right at about 5.8 seconds. It was rotating to the right rather fast at about 18 seconds, such that it started to steer less hard to the right until it landed.
The small kite showed to fly stable enough with the lighter box. Coupling the GPS data to a lower speed limit showed to be a bad idea, because this limit was reached during normal flight. One could also imagine, that with some wind in the opposite direction of the flight direction, the speed would easily reach this limit, or the glider could even fly backwards with stronger winds.

3.7 Experiment with a Balloon II

3.7.1 Setup

This setup was similar to the one in “Experiment with a Balloon I”, except we abandoned GPS for flight direction correction and added a simple delay for starting flight direction correction at 5 seconds after the release command.

[5] Implementation details: The glider model threw a GPSUnavailable exception when the speed was below 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$. Because of this GPS data for lower speeds were not logged.

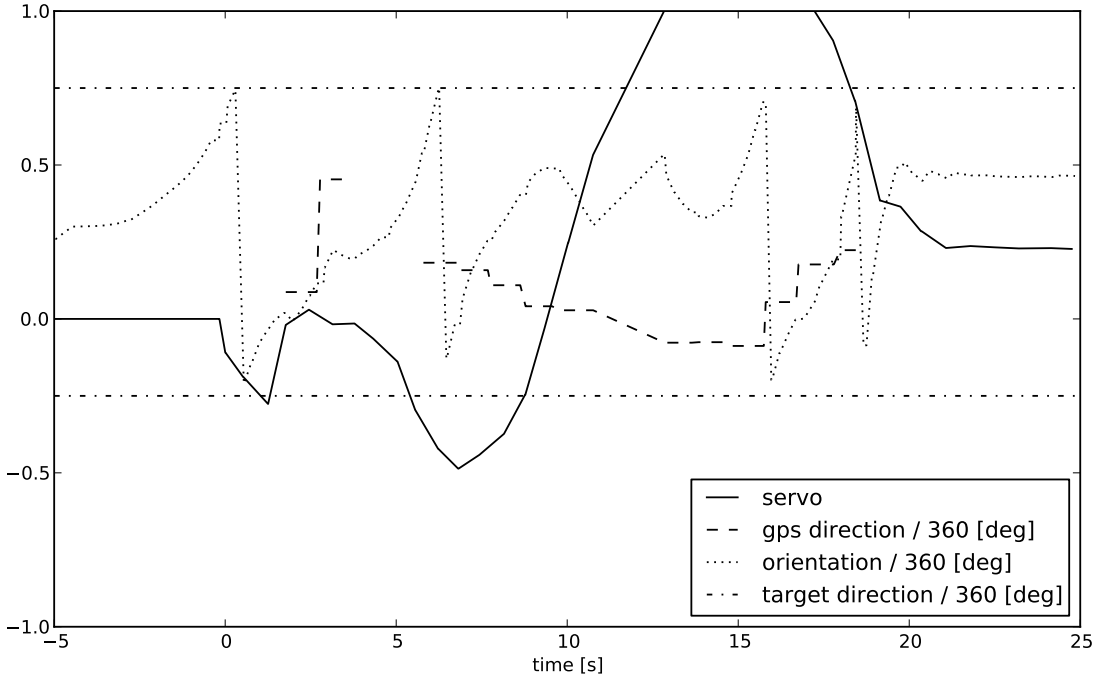

Figure 3.3: Data plot.

In order to overcome the shortcomings by using GPS data for flight correction as learned from the previous experiments and because we hoped for better reaction times by using the orientation from the magnetic compass instead the GPS bearing for flight direction calculation, we implemented a low-pass filter for the orientation sensor, increased its sampling rate and used it as an input for the flight controller. The low-pass filter should help us to minimize distortions coming from the box rotating back and forth, which we could observe in earlier experiments.
For the following tests we also limited the maximal rotation speed in order to get a better flight behaviour when the box was trundling. When the box was rotating faster than 90 degrees per second, we do not move the servo, if a move would be in that direction, that it would cause an even faster rotation.

3.7.2 Results

As can be seen in Figure 3.4, the orientation in the beginning was almost 180° different from the target direction. This, together with the rotation speed of the box explains the back and forth of the steering in the beginning of the flight. At about 9.3 seconds the target direction was reached but the box was still rotating, such that the glider started steering to the other direction.
We can also see, that the glider heavily over-steered. The target direction was reached with a rather high rotation speed. For future tests, one would have to lower the $\mathrm{pullingFactor}$ in the configuration to overcome this.

3.8 Experiment with a Balloon III

3.8.1 Setup

The setup was the same as in “Experiment with a Balloon II”.

3.8.2 Results

The glider started steering to the right because it had a small difference to correct, it rotated and then it started steering to the left. More data is not available, because the phone’s battery went

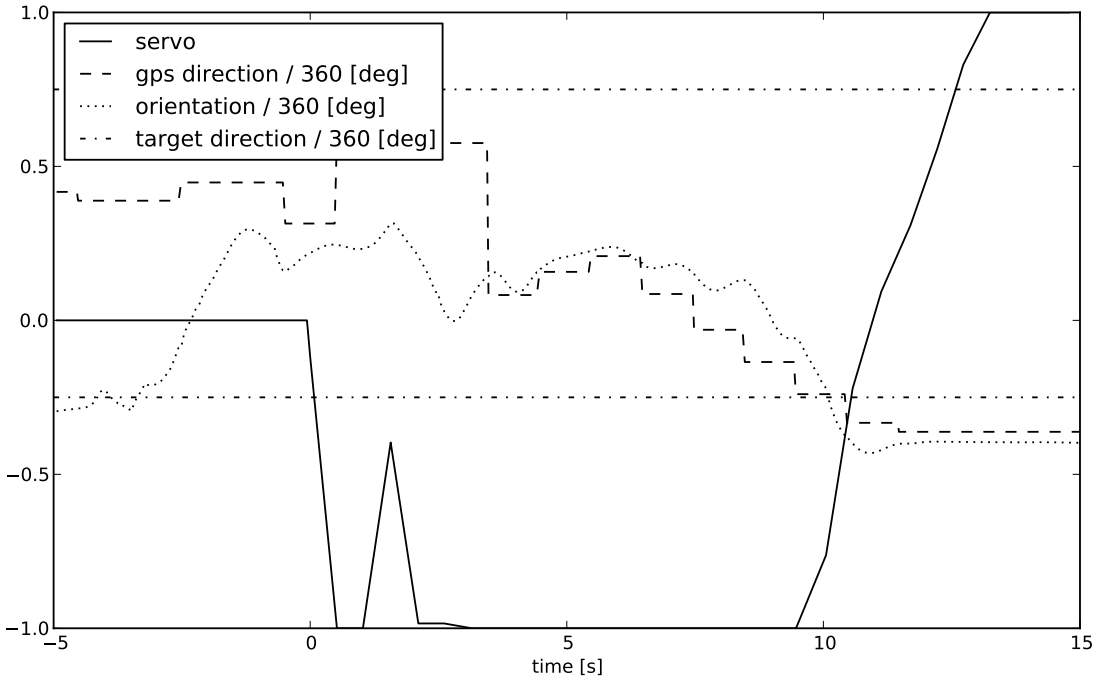

Figure 3.4: Data plot.

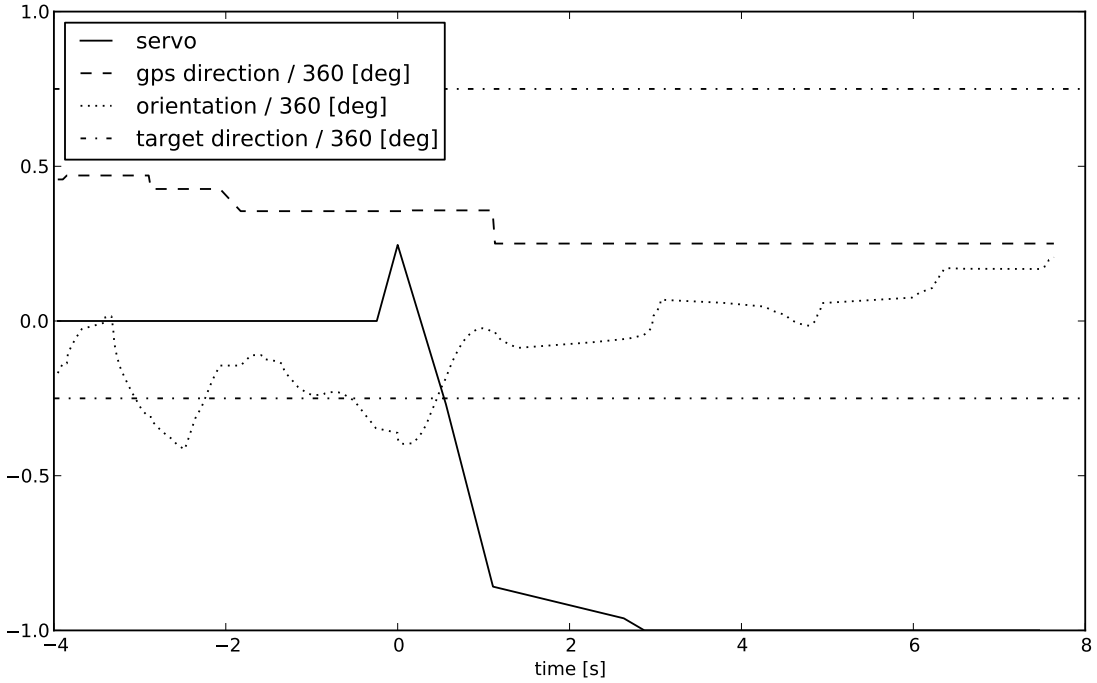

Figure 3.5: Data plot.

empty before it landed. The available data is plotted in Figure 3.5.
The phone, was constantly on for many hours for tests during this day. It was a used phone with a rather old battery and the IOIO seems to provide too few power to charge the battery of the phone when it is in use. We suggest to use a new battery for the phone or at least bring a spare one for field tests.

3.9 Testing Flying on top of a Tree

3.9.1 Setup

The box was launched from the same viewpoint tower that was used for the first flying tests. Two people held the kite, such that it was properly open before releasing.

3.9.2 Results

Because the kite was properly open at release time, it did not lose much height and the suddenly changing wind carried it along. The kite flew on top of a tree, and was stuck there. The box was rescued with a rescue mission and it fell to the ground and broke during rescue. Electronics survived and a new box had to be built. The kite is still on top of that tree.

Chapter 4

Future Work and Summary

4.1 Future Work

The software has been implemented in a modular fashion and in such a way, that it can be extended easily. Future development can build upon the existing code. For example a more sophisticated flight controller could be implemented by replacing the existing flight controller and using the same interface.
On the hardware side, a better packing technique for the kite could be found, such that it does not entangle that easily and unfold more reliably than it did in our experiments.
In order to launch this glider with a real weather balloon at heights of about 35 km one has to think about the low temperature at these altitudes. One should consider adding some insulation or a heating system to the box.

4.2 Summary

It could be shown, that not only paragliders can be steered by an autonomously flying box, but also simple kites can be used for that purpose. The kites are even easier in handling, because of fewer lines which results in easier handling and a lower risk of entangling.
The release mechanism from the previous project has been improved and works reliably now.
After breaking the original box a new, lighter version has been built. The new box with wood instead of acrylic glass has also been proven to be more solid, since it crashed many times without breaking.
Multiple tests have been performed by launching the glider from the Pfannenstiel tower and by lifting the glider up using a helium balloon and let it fly from there. Data of these flights have been logged and analyzed.
The presented hardware is low in cost and most of the used parts are commonly available. For the phone almost any Android phone can be used, as long as it brings the required sensors. The servos and the battery are common in model building. Kites suited for this application can be found at many places. Solely the IOIO module can only be ordered or bought at some specific stores. All the other smaller parts can be found at any do-it-yourself store. The low cost and the common availability of the parts make this project not only suitable for professional use in radio-sounding but also for some hobbyist projects, because filming toys flying into space with a weather balloon seems to be an emerging hobby.

Bibliography

[1] “Federal Meteorological Handbook No.3 – Rawinsonde and Pibal Observations”, Office of the Federal Coordinator for Meteorology, Last access: 25. 9. 2013
URL: http://www.ofcm.gov/fmh3/text/default.htm

[2] “Balloon Lift with Lighter than Air Gases”, Last access: 25. 9. 2013
URL: http://www.chem.hawaii.edu/uham/lift.html

[3] “Radio-soundings”, MeteoSwiss, Last access: 25. 9. 2013
URL: http://www.meteoschweiz.admin.ch/web/en/climate/observation_systems/upperair_observations/radio_soundings.html

[4] Information for finders of radio sondes, Meteolabor AG, Last access: 25. 9. 2013
URL: http://www.meteolabor.ch/find/find-e/

[5] “Gliders”, NASA, Last access: 25. 9. 2013
URL: http://www.grc.nasa.gov/WWW/K-12/airplane/glider.html

[6] “Kites”, NASA, Last access: 25. 9. 2013
URL: http://www.grc.nasa.gov/WWW/K-12/airplane/kite1.html

[7] Group Thesis: Smartphone Skydiving, Roland Meier, Muriel Pauli, ETH Zürich