Directional E-Link

A position and attitude determination- and directional system for WLAN communication antennas onboard stratospheric balloons

Gustav Johansson
Johan Selinder

Luleå University of Technology
MSc Programmes in Engineering
Department of Computer Science and Electrical Engineering
EISLAB
1 Abstract

The thesis discusses the problem and solution, as well as design and construction, of an attitude and pointing system intended to be used for increased efficiency of communication onboard stratospheric balloons. It also briefly explains the advantages of utilizing such a system during balloon flights.

To be able to point something in a specific direction, information has to be extracted regarding the exact position of the system; continuously and in real time. There are several different methods of doing this, for example by using GPS-receivers, gyros, accelerometers and magnetic sensors. These systems and solutions are discussed in some extent, where advantages and disadvantages to each method is debated, in order to finally come to a design solution. The same scenario is valid for the actual pointing system. A number of different design ideas are considered, highlighting assets and drawbacks for each individual solution.

The proposed and implemented design houses a combination of GPS-receivers and accelerometers, as well as magnetoresistive circuits to allow for a complete and redundant attitude determination system.

Several additional tests have to be performed for the prototype to be ready for a first flight, many of which are suggested and discussed in this report. A section outlining additional changes and modifications that should be done in order to increase and enhance performance of the system is also included, as well as a paragraph considering future work.
## 2 List of Contents

1 Abstract........................................................................................................................................... 1

2 List of Contents.................................................................................................................................. 3

3 Introduction ........................................................................................................................................ 7

4 Description of the Thesis Project ......................................................................................................... 7

4.1 Purpose and Background ................................................................................................................. 7

4.2 Goal.................................................................................................................................................. 8

4.3 Implementation ............................................................................................................................... 8

4.4 Required Knowledge ....................................................................................................................... 8

4.5 Extent............................................................................................................................................... 9

5 Theoretical Background, Attitude Determination ............................................................................. 9

6 Design Ideas ..................................................................................................................................... 13

6.1 Mechanical ..................................................................................................................................... 13

6.1.1 A³ ............................................................................................................................................... 13

6.1.2 Rotary Joint ............................................................................................................................... 14

6.1.3 Smart Antenna ........................................................................................................................ 14

6.1.4 Tipper² ....................................................................................................................................... 16

6.1.5 Bender ....................................................................................................................................... 17

6.2 Sensors .......................................................................................................................................... 17

6.2.1 GPS ........................................................................................................................................... 17

6.2.2 Accelerometers ....................................................................................................................... 18

6.2.3 Gyros .......................................................................................................................................... 18

6.2.4 Rotation Sensors ..................................................................................................................... 19

6.2.5 Signal Strength Sensors ............................................................................................................ 19

6.2.6 Magnetometers ....................................................................................................................... 20

6.3 Control System ............................................................................................................................... 21

6.3.1 PC/104+ ..................................................................................................................................... 21

6.3.2 Programming ............................................................................................................................ 21

6.3.3 Regulator ................................................................................................................................... 22

6.4 Antenna ......................................................................................................................................... 22

7 Design Solutions ............................................................................................................................... 22

7.1 Balloon Data ................................................................................................................................... 22
Index

7.2 Mechanical............................................................................................................... 24
7.3 Electronics................................................................................................................ 30
7.4 Sensors...................................................................................................................... 34
  7.4.1 GPS ....................................................................................................................... 34
    7.4.1.1 Accuracy...................................................................................................... 36
    7.4.1.2 Timing......................................................................................................... 36
    7.4.1.3 Manual Calculation ................................................................................... 37
    7.4.1.4 Doppler-Shift Measurements ................................................................... 38
    7.4.1.5 High-Frequency Switch ............................................................................ 39
    7.4.1.6 Tests ............................................................................................................. 40
    7.4.1.7 Conclusions................................................................................................. 42
  7.4.2 Accelerometers.................................................................................................... 42
  7.4.3 Magnetometers ................................................................................................... 43
  7.4.4 Temperature Sensor ........................................................................................... 44
  7.4.5 Rotational Sensors .............................................................................................. 44
7.5 Control System........................................................................................................ 45
7.6 Antenna.................................................................................................................... 45
8 Programming..................................................................................................................... 47
  8.1 GPS Data Retrieval, Analysis and Communication............................................. 47
    8.1.1 Tasks..................................................................................................................... 48
    8.1.2 Packages............................................................................................................... 51
    8.1.3 Conclusions on PC/104+ programming........................................................... 53
  8.2 Motor Control.......................................................................................................... 54
  8.3 Attitude Sensors........................................................................................................ 55
9 Tests ............................................................................................................................... 55
  9.1 Mechanical............................................................................................................... 55
  9.2 Sensors...................................................................................................................... 56
  9.3 Control System........................................................................................................ 57
10 Results............................................................................................................................. 57
  10.1 Mechanical............................................................................................................... 57
  10.2 GPS.......................................................................................................................... 58
  10.3 Accelerometers and Magnetic Compass.............................................................. 58
  10.4 Electronics.............................................................................................................. 58
11 Future Work........................................................................................................................................59
11.1 GPS...............................................................................................................................................59
11.2 Programming...................................................................................................................................59
11.3 Tests...............................................................................................................................................59
12 Credentials........................................................................................................................................60
13 Appendices........................................................................................................................................60
13.1 Design Flowchart.........................................................................................................................61
13.2 Budgets..........................................................................................................................................62
13.2.1 Financial Budget.......................................................................................................................62
13.2.2 Mass Budget.............................................................................................................................63
13.2.3 Power Budget............................................................................................................................64
13.3 System of Schematics Numbering ...............................................................................................65
13.4 Mechanical Drawings.....................................................................................................................66
13.5 Electronic Schematics.....................................................................................................................88
13.6 Antenna Data Sheet......................................................................................................................95
13.7 Program Code ..............................................................................................................................96
13.7.1 Balloon Data Analysis, ADA..................................................................................................96
13.7.2 PWM Motor Controller PIC, C .............................................................................................103
13.7.3 Attitude Sensor PIC, C ..........................................................................................................109
3 Introduction

This thesis discusses the problem of, and presents a solution to, an onboard attitude determination and pointing system for stratospheric balloons. The main and primary intention is to replace the omni-directional antenna used for communication between the balloon and the ground station today, with a pointed antenna, hence being able to save power as well as increase the bandwidth of the system.

For this to be possible, the exact position and orientation of the gondola in space has to be known at all times, as well as the position of the receiving ground station. Attitude determination is today commonly used within a wide variety of systems. There is however no available commercial system specifically designed and intended for use onboard stratospheric balloons. The system developed is also custom made to interact with the existing E-Link system which is in use at Esrange today.

Different combinations of sensors can be used in order to determine the attitude; several of these systems are also discussed throughout the thesis. Trade-off solutions and other design issues are thoroughly debated in order to finally reach a definitive prototype proposal, which also has been built and proven to be operational, even though additional work has to be done to obtain a flight ready version.

The entire design and development phase is carefully documented in this thesis, together with enough background information for anyone not completely up to date with today’s technology to easily understand the entire process.

4 Description of the Thesis Project

4.1 Purpose and Background

At present time, E-Link is used for high-speed communication between stratospheric balloons and the ground station at Esrange. The communication link is used in order to obtain data for analysis as soon as possible, preferably even in real-time, instead of waiting for the return of the gondola which can take days. The antenna used today is omni-directional, which limits the bandwidth severely from what would be possible to
obtain with a pointed antenna using the same amount of output power. Hence, the purpose of the thesis project is to make a thorough investigation concerning how to design and construct a pointed antenna system which will increase the data rate, while maintaining or improving the current signal to noise ratio. The project also includes the development of a prototype which fulfills these demands.

### 4.2 Goal

The goal of the thesis project is to develop and construct a prototype of a pointed antenna which is to be integrated into the existing E-Link-system currently used at the Esrange facilities. General demands of the pointing system are as follows; low power consumption, an accuracy of ±2 degrees, high reliability as well as user-friendliness. The signal losses between antenna and receiver must also be kept at a minimum.

### 4.3 Implementation

The thesis will be performed as a project; developing and integrating the new hardware with the existing system. During the first week time will be set aside in order to establish a detailed time plan for the entire project. The report will be written in English and a presentation of the thesis will be held at Esrange as well as at Luleå University of Technology. No confidential information will be included in the final report which is available to the public.

### 4.4 Required Knowledge

Fields of study where the student should have a thorough knowledge:

- Antenna theory
- Mechanical design
- Electronics
- Communication
- Programming
- Control systems
4.5 Extent

The extent of the project is planned to demand at least 40 weeks of full time work, which is why the project will be carried out by two students. The thesis-project is meant to result in a prototype, not a finished product, in order to prove that the technique works as well as fulfills the demands put on the system.

5 Theoretical Background, Attitude Determination

In order to determine the attitude in three-dimensional space, the position of at least three different points on the platform has to be known at all times. Alternatively, information on the movement around the three axles pitch, roll and yaw must be available, combined with an initial reference setting. A classic approach to solve this problem is by using components such as gyros which measures rotation rates, and hence the change in attitude can be calculated and determined. The rapid increase in availability of cheap, reliable and accurate GPS\(^1\)-receivers will probably make them a very attractive option to the somewhat conservative attitude systems regularly constructed and used today.

The accuracy of a low-cost non-military GPS-receiver is at present time a few meters. Hence the receiver antennas have to be located relatively far apart, approximately a few hundred meters, in order to obtain a reliable attitude measurement with accuracy better than one degree. This is of course not physically possible in most situations where these systems are supposed to be used. A short distance between the antennas is a demand if GPS is to compete with the very space-saving setups of modern inertial systems. Something called relative GPS (RGPS) can be used in order to achieve this. The theory behind relative GPS is to use the information from the exact same set of satellites, combined with an identical timestamp for a number of receivers. The overall

\(^1\) GPS – Global Positioning System, a constellation of 24 active satellites split up into six orbital planes, together with a number of tracking ground station around the world and a ground master system located in Colorado, USA.
absolute performance and accuracy of the GPS will not increase when using RGPS, but the antennas position relative to each other will be significantly better.

![Illustration of rotation axles in space, coupled to the movements of the Space Shuttle. Picture courtesy of NASA.](image)

This is due to the fact that if all the error sources for all receivers are the same, they will cancel each other. Hence the calculations of the receivers’ relative position will be virtually free from errors. The position can then be determined to an accuracy of a few millimeters, giving a theoretical attitude accuracy better than one degree at an antenna separation distance of less than one meter.

To get a better absolute position, something called differential GPS (DGPS) is used. In this setup the receiver is relating its position not only to the satellites, but also to a known stationary receiver on the ground which has a well defined position. This eliminates many of the errors that accumulate through the system otherwise, and hence resulting in an absolute position with accuracy in the centimeter range.

Another way to increase the performance of the system is by using carrier phase measurements. Simply put, by counting the number of wavelengths the received information has traveled, it is possible to determine the distance to the satellite very
precisely. This aids in getting a more accurate overall position than by normal GPS operation which counts code-bits that are spaced much further apart than the carrier phase. By combining these and other available methods to increase the accuracy, such as advanced signal processing and more complicated and better algorithms, a very good position can eventually be achieved.

![Schematic illustration of the DGPS theory](image)

**Figure 2.** Schematic illustration of the DGPS theory. Corrections to all received positions are sent to the remote position from the known base station. Hence, most of the errors in the signals can be accounted for.

Accelerometers and gyros are traditionally known as inertial sensors, reacting to changes of velocity and position. One initial condition for this to work is that the starting position and attitude for the system is completely known. Today both accelerometers and gyros are very accurate and have a very high resolution. The downside is that any errors induced in for example the transformation from the measured units to designation of position will accumulate, which might eventually give rise to a significant error. This can be avoided by frequently comparing the calculated position and attitude with a known reference point. Another issue is the
commonly known gyro drift which has to be compensated for whenever the measurements stretches over a longer period of time.

RLG\textsuperscript{2} and FOG\textsuperscript{3}, probably represent the peak of accuracy at the moment, and also inherit a much lower drift rate than conventional gyros. Unfortunately they also represent the absolute peak when it comes to purchase price due to their complexity. A conventional gyro consists of moving parts, which leads to increased power consumption and also to generation of friction and heat.

\begin{center}
\includegraphics[width=0.5\textwidth]{ring_laser_gyroscope.png}
\end{center}

\textbf{Figure 3.} \textit{Schematic of a Ring Laser Gyroscope. The Dither motor applies a very small oscillatory rotation in order to overcome the problem of frequency lock at small rotation rates.}

\textit{Illustration from A. D. King, B. Sc., F.R.I.N.}

Accelerometers are a cheap alternative when the demand on actual accuracy is reduced to around 0.1°. Another advantage is that accelerometers can be used as an absolute reference when relating their measurements to the Earth's gravitational field. A combination of gyros and accelerometers would hence create a reliable base from which to extract the attitude in a satisfactory way for many applications.

\begin{itemize}
\item \textsuperscript{2} RLG – Ring Laser Gyro
\item \textsuperscript{3} FOG – Fiber Optic Gyro
\end{itemize}
6 Design Ideas

6.1 Mechanical

6.1.1 A³

The Advanced Antenna Arm, A³, idea was based on endoscope instruments used by surgeons. This solution will avoid and eliminate rotation of the RF-cable to the antenna, as well as rotation of signal and power cables to motors and sensors. The flexible tube was to be controlled by guiding wires using DC-Motors as actuators. This design would allow the antenna to scan the entire half-dome of space required. The A³ design was supposed to be a financially beneficial solution, as no expensive joints or special orders had to be made when constructing the prototype.

Figure 4. Photo montage of what the A³ design could look like.

A very simple prototype was created in order to test the basic idea in real life. Unfortunately it turned out to not be stable enough in order to hold the antenna with desired accuracy. There were also complications as how to efficiently fasten and secure the guiding wires. It is the firm belief of the authors that this is a good idea, but it demands a severe amount of time and effort in order to get the mechanics to work properly. It is also doubtful whether it is possible to design a simple mechanical construction that will work according to the specifications. There was a major risk that
the system would be very expensive to build, and hence the main objective of the design is lost.

6.1.2 Rotary Joint

The rotary joint design is based on movement in two separate planes. The first plane is handling azimuth rotation, while the second plane takes care of the change of elevation angle. The basic idea is similar to the design used when aiming antenna dishes on the ground, only on a miniature scale and turned upside-down. The main problem experienced with this design is how to transfer the RF₄-signal through a slip ring system at a moderate cost. Several manufacturers were contacted in order to find a possible solution to this problem.

Another solution was to rotate the entire data-to-RF converter, and hence only having to transfer normal signals and power cables through the rotary joint. This would make the actual slip ring much easier to construct, but would complicate the system as a whole, i.e. make it a lot heavier, which increases the power consumption. The sheer size of the complete design would also increase severely as the existing data-to-RF converter is not initially designed and built to be rotated.

6.1.3 Smart Antenna

There are so called “smart antennas” where the lobe is controlled piezoelectrically or by means of phase-control. We consider this to be a superior solution for the actuating section as the need for moving parts is completely eliminated. An array of smart antennas with a slight overlap of lobes could easily cover the half-dome of space required.

The major downside with phase-controlled antennas is the fact that they quickly turn into extremely complex structures as they grow, as they need more and more phase

4 RF – Radio Frequency
shifters and processing for increased flexibility. Paratek Microwave Inc. is to our knowledge the company furthest ahead regarding development of smart antennas for WLAN\(^5\) use. They did not have an antenna or prototype that was able to fulfill our list of demands at this time.

![Figure 5](image.png)

**Figure 5.** Left image shows radiated signal when two radiating elements are transmitting with the same phase. The right image shows the radiation pattern when the lower elements output is shifted by 10 degrees. The elements are not shielded, which is why the back lobe is as large as the front one. Picture courtesy of Christian Wolff.

Another idea would be to use pointed antennas with a fairly large opening lobe, and combine these in a switching array. If an antenna with a 60° opening angle is used, a set of six will cover the full azimuth plane. By using two identical sets of six antennas positioned with an angular offset between them, the receiving ground station would most likely be visible for an antenna at all times. The only exception being when the balloon is located straight above the ground receiver. Care must also be taken to assure that the switching between different antennas is fast and synchronized enough to not lose any information during communication.

---

\(^5\) WLAN – Wireless Local Area Network, a LAN that uses high frequency radio-waves rather than wires to communicate between nodes.
6.1.4 Tipper$^2$

The Tipper$^2$ design uses two perpendicular axles to obtain a 360° rotation, as well as coverage of ±90° perpendicular to the base plane, see Figure 6. The advantage of this system over the rotary joint is the fact that there is no need for a slip ring system. One major downside of this solution is the size and weight of the construction. There is also an issue of balance in order to minimize the motor-torque needed, which in turn will add additional weight.

Several gimbaled designs have been examined as alternative solutions to the Tipper$^2$, and even though many of them might fulfill the requirements, it seems unlikely that they would work better than a rotary joint design, since they experience what is called “gimbal lock”. They are also generally much more complicated mechanically speaking.

![Figure 6. The basic idea of a typical two-axis gimbal system. The design of Tipper$^2$ would have been somewhat more complicated due to the fact that the antenna must be offset from the first rotation axis in order to not get a blocked line of sight at large tilt angles.](image)

To avoid that the system will experience gimbal lock, a design where more than two rotational axles exist can be designed. This will however complicate the mechanical construction even further, and also add weight and increase the power consumption of the system. Yet another downside with more axles is that the system will consist of more moving parts, which is definitely not a desirable feature.
6.1.5 Bender

The Bender design is based on access and availability to smart antennas that are controllable in two dimensions. It was basically a combination of the advantages from both the A³ and the Smart Antenna solution. By positioning a smart antenna at a fixed 45° inclination to the rotational plane, and fastening the antenna to a design similar to A³ gives it the opportunity to rotate. This takes care of the issue of rotating the RF-cable, as it will turn inside the hollow mounting tube. Hence, the azimuth rotation is handled by a DC-Motor, and the elevation would be controlled by means of phase control. The strength and advantages of this solution is the need for mechanic rotation in only one plane, and use of only one smart antenna instead of an array. Unfortunately, no smart antennas available could fulfill the requirements.

6.2 Sensors

6.2.1 GPS

In order to point the antenna towards the receiver on the ground, the position and attitude of the transmitting platform has to be determined. The absolute position in space can be obtained from a single GPS unit, which for a good receiver may be accurate to approximately 80 cm when using WAAS⁶ or EGNOS⁷ for L1 and L2. Without any external help-systems, high end receivers usually determine their position down to about 1.5 meter. On the distances at which the DE-Link system will operate, the available absolute accuracy of position is more than enough.

To determine the attitude of the gondola, a minimum of three GPS receivers mounted at fixed distances relative to each other must be used. If each individual GPS can be

---

⁶ WAAS – Wide Area Augmentation System, a number of ground stations located across the US which correct the signal transmitted by the GPS-satellites and hence increasing the accuracy.

⁷ EGNOS – Euro Geostationary Navigation Overlay System, the European alternative to WAAS, consisting of a series of groundstation located in Europe.
completely controlled in terms of deciding which satellites to use, and the fact that the antennas would be mounted less than one meter from each other, it seems safe to assume that the error-sources will be the same for all three receivers. This implies that the relative accuracy between the three receivers should be very good; the estimated accuracy of such a system has been approximated to about one degree in all directions.

6.2.2 Accelerometers

An alternative to GPS receivers is accelerometers. The biggest difference between these systems being that one is active while the other is not, at least not traditionally speaking. When using accelerometers, the system has to be tuned prior to each launch to create a reference system relative to the position in which to point. As the gondola moves, the accelerometers sense the change of speed, and properly mounted also rotation, and by a series of calculations a position can be obtained. Accelerometers today can be made very small and use a minimal amount of power, making them ideal for space related designs.

Another advantage is that the accelerometers are small enough to be mounted directly on the antenna; hence the pointing position relative to the reference system could be determined immediately. The downside is that even though accelerometers themselves are very accurate even at very low g-levels, flaws induced during e.g. calculations or in the A/D-conversion will be added continuously, giving a significant error in position after some time. Another way to utilize accelerometers is to relate their output to the gravitational field of the Earth, hence giving an absolute reference of the tilt. This will however not yield any information about the rotation of the flight train and has to be combined with other sensors for a complete attitude determination.

6.2.3 Gyros

A gyro is another passive system for attitude measurements, and probably also the most widespread and used system today. Gyros react to movement relative the inertial reference frame of space, which later can be transformed into a position relative a previously determined reference system. Gyros are very accurate and relatively simple
to obtain relevant information from. Downside being that they tend to be power consuming as they have revolving parts, alternatively very expensive as is the case with ring laser gyros, but also the fact that they drift with time. This drift has to be calculated and compensated for, and if not done correctly it might eventually lead to significant errors in both position and attitude.

6.2.4 Rotation Sensors

To be able to aim the antenna in the correct direction, the position of the antenna has to be known at all times. If the attitude of the gondola can be completely determined, that platform can be used as a reference plane for the tilt and rotation of the antenna. A sensor that gives an absolute reference is to prefer, as it would simplify the design of the control system as well as minimize the amount of calculations needed to get a position. It also has the advantage of not accumulating any calculating errors during transformation. It is possible to use accelerometers and gyros as rotation sensors for the antenna as well; however, it would include more circuitry and more mathematics in order to work.

6.2.5 Signal Strength Sensors

There are sensors available that can determine the signal strength of a received signal. By combining a few sensors on the part that needs to be aimed in a particular direction, it is possible to use this to keep the antenna focused towards the ground station where the transmitter is. This is a simple and elegant solution, but unfortunately it has the drawback that it only gives the direction to the receiver; in order to obtain the actual attitude a series of calculating steps must be performed.

More importantly, even though the accuracy of a system based on this type of sensor will be enough for the antenna that is supposed to be pointed; it might not be accurate enough for other applications such as pointing a scientific instrument. It also requires a land-based system in addition to the on-board part, unless it is possible to tune the sensors to focus and lock on, for example, the normal communication frequency for the balloon at all times. This also limits the operational distance of the system as the
balloon may never go below the horizon, and hence it can not be used for other applications than transmitting information to the ground station. That would make the system almost impossible to use at e.g. longer flights when signals are transmitted via satellites.

6.2.6 Magnetometers

In order to get an absolute compass reference, without using dead-reckoning together with accelerometers and gyros, magnetometers are the natural choice. This is another active system as it continuously senses the direction of the present magnetic field. Unfortunately the accuracy of magnetic compasses tends to be fairly low; in the area of ±2 degrees. This is due to the fact that the Earths magnetic field is far from constant, and hence the reliability is somewhat reduced.

![Figure 7. Chart of the total magnetic intensity over Scandinavia. Since the balloons travel at least to the Russian border, the magnetic model used to calculate the compass direction must be frequently updated.](image)

The sensors are also quite sensitive and hence respond to all other magnetic fields present. This means that the direction given from the system might deviate significantly from the real value due to for example a large iron deposit in the ground, large structures or parts of the gondola or payload onboard the balloon. At the
operational altitude of a stratospheric balloon these effects are fairly low as most disturbing magnetic fields are relatively far away.

6.3 Control System

6.3.1 PC/104+

The existing system, E-Link, flies with a 500 MHz Pentium III processor mounted on a PC/104+ card. This system has proven to be very efficient and reliable, hence it was suggested to use a similar design as the control center and mainframe for the DE-Link system as well. A rough estimate shows that computational time should not be a problem with this setup. The PC/104+ system is also very effective in terms of power, as well as lightweight and space saving.

6.3.2 Programming

DE-Link is meant to be a real-time system, and one of the main real-time languages used in the space industry today is ADA. Unfortunately ADA is not that widely used in industrial engineering as a general, hence drivers for necessary hardware might not be available. These will then have to be written manually, e.g., for communication with the attitude system and the I/O-cards. This is a time consuming task, but the result would be drivers that are honed for this specific system and hence also very effective.

C/C++ is the most widespread language today, and it is easy to find drivers and ready-made applications for it. It is however not a real-time system, which is vital in order to guarantee a communication system that works as intended at all times. It is desirable with a system that is easy to communicate with during tests as well as operation, which rules out programming languages such as assembler and other more hardcore alternatives since a user-friendly interface between the ground station and DE-Link is considered a necessity.
6.3.3 Regulator

To get a stable antenna which has a good pointing accuracy the natural choice is a PID-s regulator for the control system. Depending on what data and sensors that are available, it might also be the easiest system to design in a well functioning way. Power-wise, it might prove better to have a fuzzy logic regulator instead of a conventional regulator process. The disadvantage of this choice is reduced pointing accuracy and hence bandwidth and efficiency for the communication system.

6.4 Antenna

The antenna the system will aim has to have as small outer dimensions as possible. If the antenna grows too big, it will be very power consuming to continuously move it around. The construction itself will also quickly become very bulky and heavy, which of course will lead to a massive increase in used power. Other important criterions are the total gain and the size of the lobe. A regular circular antenna tends to become quite large when putting demands such as a focused lobe on them. The strongest candidate seems to be a flat antenna, which can have nicely aimed lobes even though their outer dimensions are very small.

7 Design Solutions

7.1 Balloon Data

To confirm the performance needed of the mechanical design, data from earlier flights was analyzed in order to extract the peak values for rotation speed and tilt angle. Unfortunately, only two different sets of data were available which makes the statistical significance of the outcome somewhat uncertain. However, it should at least give an idea about the expected movements.

Start time is decided manually from the corresponding .FTD-file (radar tracking data) in order to only calculate movement once the balloon has left the launch pad. Cut-off

---

8 PID – Proportional, Integral and Derivative control.
time (COT) is decided in a similar manner as we are only interested in data prior to cut-off. Once the flight-train is released from the balloon, there will be no need of transferring data via DE-Link.

Data is excluded if the time between two consecutive data blocks is more than one second. This is due to the fact that the information extracted from the data will be inaccurate if sampling is done too slowly. Data is also excluded if the rotational acceleration of the balloon at any time exceeds 90 deg/s². An acceleration of that magnitude is clearly unreasonable, and thus the sample is faulty and hence discarded. Table 1 summarizes the results from the extracted data.

The result from the analysis is that maximum speed of both pitch and roll is always less than approximately 30°/s. Maximum speed of rotation is about 54°/s.

<table>
<thead>
<tr>
<th>Flight Date</th>
<th>1998-12-08</th>
<th>2000-12-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>COT</td>
<td>[hh:mm:ss]</td>
<td>15:13:00</td>
</tr>
<tr>
<td>Vₐₘₓ Pitch</td>
<td>[deg/s]</td>
<td>2.382E+01</td>
</tr>
<tr>
<td>Vₐₘₓ Roll</td>
<td>[deg/s]</td>
<td>2.857E+01</td>
</tr>
<tr>
<td>Vₐₘₓ Yaw</td>
<td>[deg/s]</td>
<td>5.350E+01</td>
</tr>
<tr>
<td>Pitch</td>
<td>[deg]</td>
<td>1.770E+01</td>
</tr>
<tr>
<td>Roll</td>
<td>[deg]</td>
<td>3.620E+01</td>
</tr>
<tr>
<td>Yaw</td>
<td>[deg]</td>
<td>3.598E+02</td>
</tr>
</tbody>
</table>

Table 1. Summary of results from the extracted data. Unprocessed data was obtained from Prof. Sheila Kirkwood at IRF, Kiruna.

According to the data, the largest roll/pitch of the balloon before cut-off is less than 36°. The flight data indicates that a reasonable height above the horizon is around 10° at the maximum distance from the ground station at Esrange. Hence, the mechanical design has to be able to handle the maximum elevation, and at the same time guarantee that our construction never blocks the line of sight from our antenna to the ground station. What this means in reality is that the distance between the centre of the antenna and
the bottom of the main platform needs to have a minimum value. Calculations show that this distance has to be at least 105 mm. This will allow the mechanical design to compensate 115° in total. I.e. at 10° above the horizon the gondola may tilt a maximum of 35° away from the ground station, while the antenna is still aimed at Esrange. At 5° above the horizon a maximum tilt of 30° will be possible to correct and so on.

7.2 Mechanical

The mechanical construction is split up into two different sections. One part contains a GPS attitude system and the mainframe, while the other houses a redundant inertial attitude system, the antenna, the actuators and the power supply.

There were two main reasons for this segmentation. Firstly, DE-Link is supposed to complement an existing system without having to change it. As the antenna has to be contained at the very bottom of the flight train to have a non-obstructed line of sight towards the ground station at all times, while the GPS antennas cannot be covered by any signal reducing constructions upwards, a split into two different segments was necessary.

![Figure 8. GPS base plane with the three antennas mounted in position. The base plane is slightly over-dimensioned in order to guarantee that there will be no internal movement between the antennas.](image)

Secondly, by designing it in two parts, the GPS attitude system can be used for many other applications than pointing an antenna. It is for example possible to use the...
system to get detailed real-time information about the movements of whatever part of the flight-train it is attached to. The GPS attitude part will be designed so that it only needs to be hooked up to a 5 VDC power line in order to be completely operational. If necessary the redundant inertial attitude system can be housed in the same box, and on the same frame, as the GPS system.

A rotary joint solution was decided to be the most beneficial and reliable system for the antenna suspension. By combining a conventional industrial slip ring system with a center bore, and an off-the-shelf miniature rotary joint for single coaxial cables, the problem of transferring a complete set of signals with an acceptable loss was solved.

The design handles continuous rotation of 360° in the azimuth plane as well as more than 230° in the elevation plane. Even though 115° elevation compensation is enough, the design does not suffer from allowing a wider range. The wider range will also be an advantage to the control system, which obtains greater flexibility in moving the antenna into a correct position. However, at inclinations greater than 115° from the normal of the base plane the antennas line of sight might be obstructed by the rest of the design. As shown in paragraph 7.1 a maximum tilt of 115° should be enough for this application.

Initially, a homemade solution took care of the combination of transferring the five necessary power and signal connections as well as the suspension of the rotating parts. The idea was to use non-lubricated deep groove roll bearings in order to transfer the electrical connection through the rotation joint.

However, after discussing this solution with personnel at Esrange who had conducted a similar experiment a few years earlier, it was decided that the connection most likely would not be stable enough. Large performance fluctuations were experienced while testing the system; the signals had been heavily distorted and sometimes even completely lost.
Figure 9. CAD drawing of the homemade rotary joint and suspension system. Left image shows the system put together. Right image shows an exploded view, from the top: RF-rotary joint, inner cylinder, five sets of roller bearings with non-conducting distances between, and outer cylinder.

An available industrial slip ring assembly which fulfills our criterion is manufactured by Fabricast, Inc. in California, USA. This particular slip ring transfers up to six parallel signals, as well as handles the nominal axial and radial loads of the rotating part of the design. The slip ring is guaranteed to operate in hard vacuum environments as well as in temperatures down to -40°C without significant changes in mechanical or electrical characteristics. The center bore of the slip ring is sized to fit a miniature RF rotary joint made by Diamond Antenna in Massachusetts, USA.

There are complete solutions available on the market that can handle both RF and electrical signals, but these are extremely expensive as they tend to be special order items only. Modifying the design by using a combination of two different off-the-shelf systems saves a vast amount of money. No significant loss in signal strength is predicted compared to a complete special order system, neither is any increase in torque needed to make the system rotate compared to other systems.
Figure 10. Mechanical antenna suspension, here viewed without bearings, motor, angular sensors and gears.

The mechanical design allows easy and logically mounted angular sensors. By making the design handle rotation in the azimuth and elevation plane separately, and by mounting the sensors directly on the two rotational axles, the sensors return the values wanted by the control system with no need of additional calculations or transformations, hence saving valuable computational time.

The system uses two conventional DC-Motors as actuators, one handles azimuth rotation while the other takes care of changes in elevation. The system is designed to handle nominal and continuous rotations of up to 244°/s in the azimuth plane, and 712°/s in the elevation plane.

**Azimuth DC-Motor data:**
- Continuous speed, $C_{rpm}$: 9300 rpm
- Torque, $T$: 2.68 mNm
- Total gear ratio, $R$: 228.75:1
- Efficiency, $e$: 61%

\[
\frac{C_{rpm}}{R} = 40.66 \text{ rpm} \\
T \cdot R \cdot e = 373.96 \text{ mNm}
\]

**Elevation DC-Motor data:**
- Continuous speed, $C_{rpm}$: 12700 rpm
- Torque, $T$: 1.04 mNm
- Total gear ratio, $R$: 107:1
- Efficiency, $e$: 61%

\[
\frac{C_{rpm}}{R} = 118.69 \text{ rpm} \\
T \cdot R \cdot e = 67.88 \text{ mNm}
\]
Data from earlier balloon campaigns were analyzed, and the results suggest that this design performs according to the requirements for the system. The oversized rotational speed in elevation is a trade-off effect as the next available gearbox ratio resulted in close to insufficient speed. In order to have a reasonable safety margin it was decided to use a gearbox with over-dimensioned performance. In case the system capacity turns out to be inadequate after flight tests, the motors and/or gearboxes can easily be replaced in order to improve either rotational speed or torque as needed.

The motors are not guaranteed to function in a vacuum environment, neither have they been tested in very low temperatures. However, tests done with old standard stepper motors earlier showed no reduction in performance or any increase in power consumption. The motors were at the time of test exposed to the pressure found at an altitude of 28000 meters and a temperature of around -40° C. The motors used in the design of DE-Link are high performance miniature DC-motors normally used in medical equipment and other high accuracy applications. They are assumed to perform better than the ones used during tests.

![Figure 11. Exploded view of the Namiki DC coreless motor used in the design of DE-Link.](image)

1 – Brushes, 2 – Commutator, 3 – Moving Coil, 4 – Magnet, 5 – Housing, 6 – Flange, 7 – Pinion Gear.

A worm gear design was supposed to assure that the system would be able to self-lock in order to save power. It turned out hard to find reliable off-the-shelf miniature worm gears that would easily fit, so conventional planetary gear-heads are used instead. However, most gearboxes have a self-impeding effect when the gear ratio exceeds...
approximately 75:1. The ratios chosen to guarantee the designated requirements regarding torque and rotational speed are 229:1 and 107:1 for the azimuth and elevation motors systems respectively. Hence the desirable locking effect of a stepper motor can be achieved without the need of running current through the motors all the time. This design is predicted to save a fair amount of power compared to a set of stepper motors with similar performance, since the motors then would be among the largest consumer of current in this system. See Appendix 13.2.3 for a complete power budget.

Figure 12. CAD model of the mechanical system viewed from below, without the bottom of the protective housing, which normally covers the antenna.

One 28 V, 13000 mAh lithium battery pack is mounted in the actuator section of the design. The battery has to be separated from the rest of the system, as well as ventilated in order not to contaminate the entire system in case a leak should occur. The actuator section contains less sensitive equipment, and could also house the pack with minimal adjustments to the overall size of the system. In addition, the center of
gravity is pushed in an advantageous direction by placing it as far down on the flight-train as possible since the battery pack is among the heavier parts of the design. This single power cell is estimated to last for approximately 15 hours of operation.

As the antenna is located on the bottom end of the flight-train, it is essential that the system can survive the impact when the payload returns to Earth. Hence, some kind of impact protection that does not significantly reduce signal strength was needed. After discussing the problem with several material experts, a protective housing was designed. The material used is Divinycell, which has proven to work without flaws in several previous balloon campaigns. As Divinycell also protects against the extreme temperatures at the operational height of the stratospheric balloon, it was decided to make similar housings for the complete design.

It is vital to the overall performance of the system that the attitude section and the actuating section of the design are perfectly aligned relative each other, or at least that the offset between them are constant during the entire flight so this can be compensated for in the software. To assure this, new mounting holes has to be made in the top part of the E-Link protective housing to attach the attitude section of DE-Link. These are the only modifications made to the existing system. The actuator part will be mounted on existing aluminum bars located on the bottom of the E-Link box.

The entire mechanical system has been developed and designed using SolidWorks 2005. Rymdbolaget in Solna, Sweden, has manufactured the custom aluminum details. A complete set of mechanical drawings can be found in Appendix 13.4.

7.3 Electronics

The DC-motors require driver stages in order to be controlled. These were designed based on an H-bridge circuit from Analog Devices. Signals are sent from the control system, which then are buffered and magnified through the driver circuit. The output from the driver stages is 12 Volts. The signals to the motors are pulse width modulated in order to have high efficiency at both low and high speeds. The electronics schematic for the control card is included in Appendix 13.5.
Figure 13. Sketch of basic H-bridge circuit. Depending on the position of the switches A1, A2, B1 and B2 the current is forced in different directions through the output. This makes it possible to alternate the direction of rotation for the DC-Motor.

A microcontroller, PIC18F2331 provided by Microchip Inc, was used in order to generate the PWM\(^9\)-output for the DC-motor control. In addition to this, it also handles the acquisition and conversion of data from the two rotational sensors giving the position of the antenna relative to the platform. Communication between the microprocessor and the PC/104+ is handled via the RS-232\(^{10}\) port.

The PIC18F2331 was chosen as it has built-in dedicated ports for pulse-width modulation, as well as an A/D\(^{11}\)-converter with sufficient resolution for the angular sensors. It is also possible to run it at a clock frequency of 40 MHz, making it a back-up alternative to handle parts of the control system for DE-Link in case the PC/104+ would be overloaded from running the GPS calculations.

---

\(^9\) PWM – Pulse Width Modulation, by pulsing a fixed voltage output at a high frequency, the effect of a lower mean voltage is obtained. This method increases the efficiency of DC-motors when varying the speed.

\(^{10}\) RS-232 – Recommended Standard 232, Standard that defines the mechanical and electrical interface for serial communication.

\(^{11}\) A/D – Analog to Digital
The A/D-converter of the microcontroller has a resolution of 10 bits, hence returning the angular position from the angular sensors with a resolution of approximately 0.35 degrees. This is more than enough as the sensors themselves have a resolution of 0.5 degrees. The interface between the PC/104+ and the microcontroller is handled via a series of single character commands sent over the RS-232-port, where certain hexadecimal byte series initializes and control the microprocessor functions. The PIC\textsuperscript{12} is passive in the sense that the PC/104+ has to ask for angular position in order to receive a value; it is only updated and sent to the RS-232-port when asked. The output is just like the input a series of hexadecimal bytes, including a few control bytes. For additional information, see section 8.2 Motor Control.

During the first tests it turned out that the power-PWM output pins would not function properly, so the PWM output signal was rewired to the CCP\textsuperscript{13} ports which can be configured as PWM modules. This was a last minute solution as there was not enough time to analyze why the power-PWM pins behaved strangely. However, the function of the circuit is exactly the same. The only difference being that the PIC needs

\textsuperscript{12}PIC – Programmable Integrated Circuit

\textsuperscript{13}CCP – Capture, Compare, PWM - module
a little more computational time in order to produce the output. This is not a problem as the microcontroller is used for this particular purpose only.

A second circuit card was designed and created primarily as a complement to the GPS receivers for redundancy, but also in order to increase the overall stability and accuracy of the attitude determination system. This card includes accelerometers, magnetoresistive circuits, A/D-converters, a temperature sensor and connections and power supply for up to four external heaters. The basic idea is to relate the position of the platform to the Earths gravitational field for the two different axles of tilt; pitch and roll, with the help of accelerometers. The magnetoresistive circuits are used to determine compass-direction, or yaw.

Combining this information with a rough position in space given by a single GPS-receiver completely establishes the 6-dimensional pose of the platform. A second microprocessor, identical to the one used for motor control, keeps track of the movements of the platform at all time, and continuously updates the mainframe of DE-Link via RS-232, where the information is available to the control system. Internal communication on the sensor card is handled via the I²C protocol which is available on all chips used in the design. The electronics schematic for the control card is included in Appendix 13.5.

The temperature sensor has a span of -40° C to +125° C, and a resolution of 0.5° C. The temperature is read by the microprocessor at a given frequency, and a set of heaters will start in case the ambient temperature around DE-Link drops below a predetermined value. This control loop is handled entirely by the PIC-processor.

The accelerometers used are provided by Analog Devices Inc. and are capable of resolving changes in acceleration down to 2 mg. When using them as pure tilt sensors the resolution is, according to the datasheets, around 0.1° when measuring the

14 I²C – Inter Integrated Circuit bus, a serial data protocol created by Philips to simplify communication between different IC’s.
deviation from the gravitational field of the Earth. The resolution of the magnetic compass is similar to that of the accelerometer, making it possible to see changes as small as 0.1°. However, the accuracy is limited by the fact that the magnetic field varies significantly depending on location. The sensors might also be disturbed if in the vicinity of an object surrounded by a magnetic field. This gives an actual compass accuracy estimated to be in the order of ±2°. For further information on the sensors, see section 7.4.

This circuit design is very effective in terms of power consumption, averaging at around 120 mW when the heaters are off. This is partly thanks to the lack of moving parts in the sensors chosen. A significant increase in power consumption would be the result if miniature gyros were to replace the accelerometers and by changing the magnetoresistive circuits to regular coils measuring the magnetic field strength. Other reasons for choosing this design are the fact that accelerometers are fairly cheap compared to gyros, while at the same time it is possible to relate the output to the Earths gravitational field and by that obtain an absolute value.

MPLAB IDE was used to edit the source code; the C-compiler used was bought from CCS. A programming circuit was built in order to use the In-Circuit Serial Programming (ICSP) capabilities of the processor, see Appendix 13.5 for schematics. For additional information on how the microcontroller is setup and how it works, see Appendix 13.7.2 and 13.7.3 which contains the C source-code for the controller- and sensor- cards. For more information on the PIC18F2331, please refer to http://www.microchip.com.

7.4 Sensors

7.4.1 GPS

The GPS receivers used for DE-Link are manufactured by u-blox. The receivers are based on the ANTARIS GPS Engine, developed by Atmel and u-blox. The chipset
Design Solutions consists of three IC’s; one RF front-end IC, a low noise amplifier IC and a baseband IC. The three chips combined comprise a complete GPS solution from antenna input to navigation output. The chipset features 16 channels, a 4 Hz position update rate as well as DGPS and SBAS\(^{16}\) (WAAS, EGNOS) support. Power consumption is less than 100 mW at an update rate of 1 Hz. For additional information on the GPS receiver-chip, see http://www.u-blox.com.

The chipset is mounted on a circuit card designed by Lars-Olov Jönsson at Esrange, in order to combine the receiver with a TCP/IP module for easy and fast communication. The receivers are also completely operational for use above 18 km, which is a necessity when flying onboard a stratospheric balloon, as well as thoroughly tested for the harsh conditions at the operative altitude. The majority of other available chipsets in the same price range tend to be limited in use at high altitudes due to the temperature and vacuum environment.

**Figure 15.** Left picture shows the active antenna made by Sarantel which is used by the GPS receivers on DE-Link. Right picture shows the TIM-LP module manufactured by u-blox. The chipset consists of three separate IC’s, all supplied by Atmel Corporation.

As previously stated, the idea is to use RGPS to determine the attitude. In order to do so, at least three GPS receivers has to be used for a three dimensional value to be

\(^{15}\) IC – Integrated Circuit

\(^{16}\) SBAS – Satellite-Based Augmentation System, a mutual name for ground based systems that increase GPS performance and accuracy, such as WAAS and EGNOS.
possible to extract. A complete system would probably use an additional receiver for redundancy, but for the prototype three will suffice.

7.4.1.1 Accuracy

With the same error sources the relative position and accuracy between the three receivers should be more or less perfect. In theory, the accuracy is then limited only by the communication protocol and the accuracy of the phase measurements of the receivers, which currently allows a resolution of approximately 1.5 mm for each receiver. This converts to an angular accuracy of about 1°, which is well within the objective for the project. As the antennas are mounted within 0.5 meters from each other, the error sources are guaranteed to be the same as long as an identical set of satellites are used by the receivers.

It is possible to manually limit the amount of satellites viewed by a receiver. By setting this number to e.g. five it should theoretically mean that the same five satellites are used by the three identical receivers, since they are mounted so close together. To ensure that this actually is the case one receiver should be configured as master, locking on a specific set of satellites. The master then tells the other two receivers which satellite-numbers it is using, and claiming that all other satellites are unhealthy. This is a vital and fairly basic function which unfortunately is not supported by the u-blox receiver.

7.4.1.2 Timing

One of the main problems experienced with the GPS modules made by u-blox is a timing-issue. All data need to be read and extracted almost simultaneously from all three receivers in order to achieve good accuracy attitude-wise. This data need to have a timestamp that does not differ by more than one nanosecond for the system to be accurate enough.
There is unfortunately no easy way to synchronize the time between different receivers made by u-blox. A hardware line with an external clock pulse could have solved this problem, but the u-blox chip does not have any input ports. An adjustable software clock is another alternative that might have worked; the navigation calculations could then have been timed to a certain GPS/UTC\textsuperscript{17} time. Such a function is not available for the u-blox chipset. A third option would be to manually time the three GPS-receivers by polling messages at the same time.

Unfortunately, this will not solve the entire problem since the individual position-calculations still will be performed at different times, and this time can only be approximated based on when the poll-message was sent. Yet another solution is to start all the receivers at the exact same time and simply hope that they will do the calculations at the same time. In theory this might work as the receivers are, or should be, identical. However, theories rarely interconnect perfectly with real life, and this case is not an exception.

7.4.1.3 Manual Calculation

All these issues lead to the conclusion that the relative position calculations need to be done manually within the software designed specifically for DE-Link. The timing problem is then reduced to only include the measurements, not the calculations as well. Which satellites to use can then be chosen without regard to what satellites the receiver uses, thus solving the problems indicated in the two previous paragraphs. In addition, the actual amount of calculation is reduced severely, since there really is no need to calculate three complete position solutions, but rather only calculate the difference between the antennas. To achieve this, the phase measurements have to be extracted in raw data format from the three receivers.

\textsuperscript{17} UTC – Coordinated Universal Time
### 7.4.1.4 Doppler-Shift Measurements

Another measurement received from the u-blox receivers is the Doppler-shift of the received signal. Doppler-shift is the induced shift in base-frequency of each satellite due to the relative motion between receiver and satellite.

**Figure 16.** Example of the Doppler-shift of a GPS Satellites transmission frequency seen from the receiver located on the surface of the Earth.

When studying the Doppler-shift readout from the receivers an interesting error was found; the u-blox chipsets seems to always see the exact same difference in shift between measurements, no matter which satellite they are referring to. The error also drifts with time, at about 200 Hz per hour. The difference in Doppler-shift should be zero between the receivers, or at least constant over time, since it depends only on relative speed. All three antennas are mounted on a rigid aluminum frame guaranteeing that their relative speed is zero at all times, any difference in speed due to rotation is too small compared to the satellite velocities to impose any aberration. This shows that the Doppler-shift is an excellent way of checking whether the receivers are reliable or not.

The Doppler measurements also resulted in unreasonable values of the shift. The maximum possible shift on the ground is around ±6 kHz, while our own program as
well as the program supplied by u-blox always shows negative values, and with magnitudes as large as 18 kHz. These effects both indicate that the oscillators used on the RF-chipset are of very low quality. They are not synchronized and they also drift with time.

To solve this problem the onboard RF-oscillators were removed and replaced with one single very stable clock-signal. The Doppler-shift readouts were suddenly kept within a very reasonable range, and the accuracy in position was improved considerably. By doing this the three receivers are matched at the RF-stage, which should aid considerably in yielding measurements accurate enough to deduce the gondolas attitude.

7.4.1.5 High-Frequency Switch

In between the conclusion that the clock signals needed to be matched, another solution to achieve less error was tested. The theory being that using only one receiver, the errors induced by it would be cancelled. Thus a high-frequency switch that could switch the three antennas within 50 ns was installed and tested. Since the switching was very fast compared to the read-out times (500ms), the switching should not cause any errors in the measurements.

Unfortunately it did so anyway, probably because the receivers have hardware Kalmann-filters attached to the RF-chipset modifying the input signals, which leads to the effect that all three antennas will be filtered together to appear as one. The Kalmann-filters are installed to reduce the errors induced by reflected signals from the satellites, i.e. small phase-shifts in the signals, which is exactly what we achieved by switching the antennas, and thus the difference we tried to measure is completely removed by the filter.
7.4.1.6 Tests

After matching the clocks a number of tests were done to finally decide whether it is possible to use the u–blox receivers for attitude determination or not. To be able to compare the three receivers they were all mounted with one antenna by a high-frequency splitter to guarantee that no phase errors were induced in the transmission lines.

By doing this, the raw measurements of the three receivers should be exactly the same if taken simultaneously. The tests immediately showed large differences in measurements. As previously stated, there is no reliable way to determine the exact time on nanosecond-level with the u–blox receivers. In fact, the tests quite clearly indicate that the time stamps that are attached to the data stream are wrong. If one tries to compensate the measurements for the time-difference, a larger error is induces than there was before the correction.

The reason to this is unknown to us, since neither u–blox nor Atmel are willing to discuss these discrepancies with us. In any case, even if the measurements are done at different times on the receivers, doing a set of measurements should yield a dependence of the difference at least. This was done in a number of ways to try to determine by experiment how the u–blox chipset makes its measurements.

As seen in Figure 17, there is no apparent correlation at all between the receivers. The difference that should be zero is jumping incoherently between measurements by N*128 clock cycles. Again, why this is so is unknown, and u–blox and Atmel are unwilling to discuss any possible causes. What is seen in the plot is an unsuccessful attempt to correlate the phase difference to the Doppler-shift difference since this is the most reasonable correlation to assume. As stated in previous paragraphs, the Doppler-shift difference should be zero. It is however not, thus implicating that the receivers are not accurate enough for our purpose.
Other correlation attempts were done as well in order to find a connection between the different read-outs of three seemingly matched receivers, such as satellite numbers, which channels each satellite is locked on, which order the u–blox chipset sends the raw data in and so on. None of the tests resulted in any better correlation than the figure above. As a consequence to these strange measurements, two chipsets were dismantled to measure the digital signals that are transferred to the base-band processor inside the chip. Even when matching every voltage and clock frequency and using one antenna, the output signal from the RF-chip to the base-band processor were not matched. The authors interpret this result as proof that the Atmel RF-chipset is not accurate enough for attitude determination in the u-blox configuration.
7.4.1.7 Conclusions

In order to determine the attitude by using GPS-signals, our conclusion is that receivers with better performance and accuracy than those supplied by u-blox must be used. Most importantly, they need to be equipped with the ability to synchronize the position measurements, or at least return a very exact timestamp of when the measurement was made so that software corrections can be performed.

An example of a possible replacement receiver that is capable of this is the OEM4-G2L from Novatel; which also has a relatively low power consumption of 1.8W. The time-pulses on these receivers can be transferred from one master to several slaves, hence giving them all the same time, which also is accurate to <50 ns, which should be enough. We need an accuracy of sub nanosecond-level, but the datasheets available for the Novatel chipset does not state the resolution of their timing function. The chipsets are prepared to accept an external oscillator input as is needed to match the measurements.

Position accuracy using single point L1 is about 1.8 meters. The measurement resolution of these receivers is approximately 6 cm RMS when only using the L1 Code. The L1 Carrier Phase measurement has a resolution of 0.75 mm. These values would give DE-Link a theoretical pointing accuracy of 0.5 degrees in its current configuration. The data rate is also significantly higher than the u–blox receivers. Novatel promises a position and measurement data rate of 20 Hz, while u–blox guarantees 4 Hz. The dominant drawback of the Novatel chipset is the price, which is about 50-100 times higher than u–blox. Other brands have been examined, and the general outcome is that the price for a GPS receiver that handles true carrier phase measurements and analysis starts at around $4000 each.

7.4.2 Accelerometers

Accelerometers were chosen as a secondary system mainly because of their size and low power consumption combined with their high resolution and sensitivity. The accelerometers are manufactured and supplied by Analog Devices Inc, and the model number is ADXL203.
Figure 18. Left image shows dimensions of the ADXL203 in mm. Right image shows the functional block diagram.

DE-Link uses three identical sets of dual-axis accelerometers, in order to fully be able to determine pitch and roll movement. Each dual-axis is needed to determine the correct angle towards Earth’s center of gravity since each axis has unsigned sensitivity. In order to obtain full three-dimensional attitude, three pairs are needed. The accelerometers output are referenced to the gravitational field of the Earth; by doing this the problem of dead-reckoning is removed as an absolute value can be continuously extracted. For additional information on the accelerometers, see http://www.analog.com.

7.4.3 Magnetometers

The use of accelerometers provides an absolute measurement of the direction towards Earth’s center of gravity, but it does not give information of the rotation of the gondola. For this purpose DE-Link uses magnetoresistive circuits built by Honeywell. In order to get a reliable compass reading regardless of the inclination of the system, three orthogonally positioned circuits are used. The three sensors together use a maximum of 20 mA at 12 VDC, which makes them very power effective.

As the stratospheric balloon will tilt in different directions, it is very important that the magnetic field is measured in all three directions. These values can then be correlated to the tilt measured by the accelerometers, and with a map containing information of the magnetic fields at the current position. Once all this is established and calculated, a
valid compass direction can be determined. For additional information on the magnetoresistive circuits, see http://www.honeywell.com. As the magnetic field at the position of the balloon must be known to calculate the heading, the WMM\textsuperscript{18} from DoD\textsuperscript{19} was implemented in the code in the PC/104+.

### 7.4.4 Temperature Sensor

The temperature sensor is made by Analog Devices Inc, and has a resolution of 0.25° C and an accuracy of 0.5° C. The total range of the sensor covers -40° to +125°. The IC has a built-in 10-bit A/D-converter and is fully compatible with the I2C serial interface. For additional information on the sensor, see http://www.analog.com. The model number of the temperature sensor is AD7414.

![Analog output from the Vishay Spectrol 360° Smart Position Sensor. Picture courtesy of Vishay Intertechnology, Inc.](image)

**Figure 19.** Analog output from the Vishay Spectrol 360° Smart Position Sensor. Picture courtesy of Vishay Intertechnology, Inc.

### 7.4.5 Rotational Sensors

The Vishay Spectrol 360° Smart Position Sensor was chosen due to its ratio metric output ranging a full 360° with no dead band. It is also completely self contained, not requiring any external electronics; the output is an analog voltage signal according to Figure 19. The angular response is 50 µs, and the output is an absolute value which

---

\textsuperscript{18} WMM - World Magnetic Model  
\textsuperscript{19} DoD - Department of Defense
Design Solutions

reduces the risk of errors when extracting the angular position of the antenna. The resolution of the sensor is 0.5°.

7.5 Control System

The brain of the DE-Link control system consists of a Pentium II 300 MHz MMX-S, with 64 Mb RAM and a compact flash card of 512 Mb working as a hard drive. The operating system installed is FreeBSD 4.10, which is a UNIX based OS. Everything is mounted on a PC/104+ card, which in addition to the processor and memory also has two serial ports, 10 Mbps Ethernet, a parallel port and vga-out.

![A typical PC/104+ card. The outer dimensions of the complete mainframe mounted on DE-Link is 95 x 90 x 20 mm.](image)

Figure 20.

All calculations are made in the mainframe, except for the A/D-conversions which are handled by hardware IC’s for simplicity. Communication between the PIC-processors and the PC/104+ is done via standard RS-232 communication protocols.

7.6 Antenna

Svenska Antennspecialisten AB manufactures the antenna chosen for the mechanical pointing design solution. It is a flat panel antenna with vertical polarization and a gain
of 9 dBi\textsuperscript{20}. The intended use of this antenna is ground based WLAN systems. However, it suits our design as it is very compact and lightweight. For additional information on this antenna, see Appendix 13.6. The choice of antenna was suggested by Lars-Olov Jönsson, who designed the E-Link system, and who also has considerable experience within the RF-field.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{antenna_pattern.png}
\caption{Signal strength of the chosen antenna. Left image shows the azimuth radiation pattern, the right image shows the radiation pattern in the elevation plane.}
\end{figure}

The alternative of using smart antennas for the system has been thoroughly investigated, as it has the very appealing aspect of completely eliminating the need for moving parts in the design. Today, there are no simple available solutions that give the user the possibility to control the lobe the amount necessary for our system to function.

There are phase arrays that with an acceptable loss in gain can steer the lobe. However, in order to cover a half sphere of space, the design would become very bulky as well as quite complicated and power consuming. It is definitely out of the scope for this thesis, and also to our knowledge, to include the development of a miniature phase array antenna system. Several manufacturers of advanced antenna systems has been

\textsuperscript{20} dBi is the strength of the signal that would be transmitted by a non-directional isotropic antenna i.e. radiates equally in all directions.
contacted and dealt with. It is the belief of the authors that systems will be available within a couple of years. At that time it might be advisable to redesign or complement the DE-Link system with a smart antenna.

The antenna was delivered to us as ordered; without an outer plastic cover, since the entire system will be surrounded by a protective housing.

8 Programming

8.1 GPS Data Retrieval, Analysis and Communication

For the mainframe of DE-Link; the PC/104+, ADA was the natural choice of programming language. Partly because the authors have previous experience of this language, but mainly due to the fact that it is a very reliable real-time language for sensitive systems. In order to obtain a neat overall structure as well as information on usage of computer resources, the program was split into several different tasks which all have individual priorities and purposes. Below follows a short description of each task as written in the main file Gps.adb. It should be noted that the code has changed in both appearance and use several times during the development as new tasks were needed and others were phased out. To give a better overview of the program structure, see Figure 22.

To easier be able to change certain data and settings, for instance the GPS addresses and ports, a configuration file named GPS.Conf is provided to avoid having to recompile the program for every little modification made. The file contains complete information on which variables that can be set, and also how to set them. The configuration file is read directly at startup, before the tasks start running, which means any change to the file has to be followed by a hardware reboot for the changes to be initiated.

A secondary program to display received data on a ground computer has also been written in ADA. This is not an optimal solution as ADA’s graphical abilities are quite limited. However, this program is not critical to the function of the DE-Link system,
and was implemented quickly only to aid in development of the design. Because of this the program experiences crashes at different times, which had implications for communication with the onboard computer. A small logger-program was written to provide a buffer between the programs. The logger is stable, and takes care of the communication with the flight computer. All incoming data is saved in a text file that can be read from the graphical program. Hence, in case of a crash the logger keeps saving data and no problems arise for the flight computer. This also has the effect that no data is lost. The graphical program is simply restarted and continues to read the text file.

![Diagram](image)

**Figure 22.** PC/104 Program Flow Chart

### 8.1.1 Tasks

**Startup** - Initializes the system and prepares for navigation. Basically it loads the external Gps.CONF-file that was created in order to be able to change certain variables without the need of recompiling the program. It also sets a flag when the procedure has finished to allow other tasks to begin.
**Prog** – Currently this task only starts the rest of the tasks needed for the program to run, i.e. the Data_Retrieval and Calculation tasks. It also starts the RS-232 communications protocols. The purpose of this task was to be a monitoring task that can restart a crashed task to further improve the reliability of the program. However, due to lack of time, this was never implemented and the task ends directly after performing the mentioned assignments.

**Data_Retrieval** - This task is constructed as a task type, allowing multiple instances of the task to be started. That approach was chosen since all three GPS receivers each needs an identical retrieval task, and thus saving a lot of coding by not having to copy the task, but simply instantiate it several times. The data received from each GPS is stored in protected variables. A small amount of data analysis is done here to be able to detect which variables to store certain data in. In the end, it turned out that only one instantiation of this task was needed anyway, as the GPS only is used to determine the position of the balloon, not the attitude.

**Calc** - Calculates the navigational solution in vector form from the received GPS data. The calculation uses a spherical coordinate system to avoid unnecessary conversion between different coordinates. The end result is two angles, azimuth and elevation, which represent a vector that points from the antenna to the target at Esrange. Originally the task waited for data to arrive from all three GPS’s before running the calculation and then stored the result in a protected variable.

A full implementation of the standard algorithm for calculating GPS position was implemented due to the fact that the receivers are not accurate enough doing these calculations on their own. The raw data was extracted and all the necessary calculations were made in this task. After realizing that the GPS attitude system would not work as planned, these algorithms were not needed any more and could be removed to shorten the code. New algorithms were created to calculate the attitude of the balloon from the data given by the accelerometers and magnetometers, as well as to calculate the azimuth and elevation angles from the balloon relative to the target position on earth. These algorithms are located in the Operations library.
**PID** - This task communicates with the PIC-processor that controls the motors. The task retrieves the latest target angles and sends them to the PIC-processor. Initially this task was used as a full PID-regulator, but as the communication speed degraded continuously it was decided to move the PID regulator to the PIC-processor instead, cutting communications with 60%. This will improve performance since the delays within the regulating process are strongly reduced.

**Operator Input** - This task starts directly after initialization of the system and continuously listens for commands from ground. This is achieved by setting up a UDP\(^{21}\)-port that is only listened to. There were problems with the program crashing when trying to use the same port for two-way communication whenever the connection was lost. Since this is a reasonable event during a flight, the problem had to be circumvented. The solution was to set up two different ports, one in each direction; ground to balloon and balloon to ground. This way the port never got locked in listening mode which prevented new commands to be sent when the connection was re-established with the ground station. This solution is not optimal since two connections should not be needed. Instead, a watchdog should be implemented in the listening code to prevent lockup.

**Print** - A pure development task. If asked to, the task prints different data to the screen for analysis. This task is not used in flight-mode and can be removed or simply not started to minimize the code.

**Resend To Ground** - Resend is a low priority task that sends certain data to ground computers. Resend must be activated from the ground. As the data is selective and only sent when all other tasks are idle, one should not rely on this data to do any calculations. It is purely to show that the system is operational and a minimum data control channel to check that calculations are reasonable. The data sent is always the most updated from the GPS receivers and calculating tasks.

---

\(^{21}\)UDP - User Datagram Protocol, a part of the TCP/IP suite of data transfer protocols.
**Acc** – This task communicates with the sensor PIC-processor and receives the raw measurements from the accelerometers and in the future also the magnetometers. As is the case with the PID-regulator, it would be better to place as much raw data reduction directly in the PIC-processor to minimize communications. This was however impossible to implement due to the fact that the sensor cards were received only two days before the project ended. At the moment the raw data is received and only one of the accelerometer pairs is used to deduce the attitude which limits the roll and pitch to \( \sim 90^\circ \). This can be improved to full 3D attitude determination with a small additional amount of programming, and then preferably implemented in the PIC-processor instead.

### 8.1.2 Packages

Several different packages were created to be able to use protected variables, and using the same library for communication with all GPS receivers. Below follows a short description of each package. It should be noted that many of the packages can be improved both in structure and efficiency.

**Store** - This package contains a couple of procedures for storage of data in protected variables, and functions to read the variables back to whichever task is requesting data. This way there is no risk of changing the data while it is being read and hence avoiding corruption of data.

**Tibbo** - This is a large and somewhat badly structured package that controls communication with the GPS modules. The name comes from the TIBBO modules mounted on each GPS card that controls the TCP/IP protocols. All of the commands that are sent to the GPS modules can be found in this package, which makes it quite large.

**Operations** - Takes care of conversion of data from the GPS into float values. This conversion works from float to GPS values as well. It also includes a few other operations as well, such as changing coordinate system and calculating the antenna angles that are used to position the motors.
Resend - A smaller version of the Tibbo-package. This package contains the communication library for communication with ground.

Code_List - Contains a look-up table for converting message-codes to text for easier understanding of the data. This package is for debugging only, and can safely be removed in a flight version.

Raw - This package reads the raw data format coming from the receivers. It uses the operations package to convert each variable to a correct value which will be used during calculations. It also contains a lot of read-functions to allow the program to read any value necessary. It includes a couple of correction functions used to adjust, among other things, each receivers data in accordance to the difference in clock information.

This package is quite advanced as the way the data is processed in the u-blox receivers is somewhat vague. However, since raw-data will not be used to calculate our position in the latest version, the package is obsolete. It is still mentioned in the thesis as it is operational and can be used if more accurate GPS-receivers are to replace the u-blox receivers.

Ephemeris - This package interprets the Ephemeris messages that are received. These messages contain the orbit elements of the satellites, and can later be used to calculate the satellite position in the ECEF\(^{22}\) coordinate system. It also contains read functions for the stored variables. Just as the previous package, this is obsolete as the position is not calculated manually anymore.

Satellite - Contains the calculation of the satellite position. It reads data from the Raw and Ephemeris packages and calculates the current position. The satellite position has to be accurately known in order to be able to calculate the receiver position. This package is obsolete.

\(^{22}\) ECEF – Earth Centered, Earth Fixed reference frame
SVs_used - Compares which satellites are used by each of the receivers, and then chooses the ones that are supposed to be used to calculate the current position. This package is obsolete based on the same facts as before.

Matrix - Package that contains a couple of basic matrix- and vector functions. The matrix calculations were first used to solve the over-determined equation system that arises in the position calculation, but found its use in the current version of the program as well, by calculating vectors and coordinate systems in the control system.

Position - Calculates the position of the receiver by using the standard algorithm for GPS position calculation. Further information on this subject can be found in ICD-200c. This package is obsolete.

Mag - Calculates Earth's magnetic field components at a location and time. The result from this package is three vectors in the ECEF coordinate system. The package uses the WMM-2005. The data from the model is written directly in the code to save development time, in the future it would be better to read the data as is given by DoD for easier upgrade when the model changes.

8.1.3 Conclusions on PC/104+ programming

During programming and reprogramming time and time again of the PC/104+, degradation in performance were noted. Especially within the RS-232 communications with the PIC-processors, which were suffering heavily from time-lag and loss of data at the end of the project. But also the Ethernet communications suffered from data loss. This was probably due to the fact that a Compact Flash memory card was used as a hard drive, and that FreeBSD uses the hard drive to store the communications buffers rather than using the internal RAM. Compact Flash-drives are not very fast compared to RAM and are known to have issues handling frequently written data, i.e. fragmentation problems. This would explain the degradation in communications the further along the project got, even though the protocols were improved to send less data and check bits were implemented.
8.2 Motor Control

The program for the microprocessor controlling the DC-motors and the acquisition of data from the angular sensors is written in C, due to lack of ADA-compilers for PIC-processors. This should not affect the real-time nature and functionality of the system as a whole though, since the microcontroller is handling a very limited amount of tasks.

DC-motor steering was originally handled by the control system in the mainframe computer, but was later implemented directly in the PIC-processor to reduce the amount of information transferred over the serial port. The DC-motors are controlled by PWM. This makes it possible to control and set the speed in 256 steps, each step adding approximately 0.05V to the bridge output voltage. The angular sensors are continually read by the PIC-processor using a timer and interrupt routines to achieve good accuracy. The data is immediately processed to remove the jump that occurs when crossing the zero-pole of the sensor and also rotated to define the local reference frame of the gondola.

A PID regulator was implemented and is run with another timer and interrupt routine to provide accurate times between calculations. This is essential to be able to create a good discrete regulator. The regulator is of increment-type to prevent integrator wind-up and the discrete formula used is provided below. The formulae is a standard PID-regulator as described on Åbo University’s homepage. The PIC is programmed to include a safety system to assure that the antenna does not rotate more than the allowed 230° around the elevation axis. For additional information, see the source-code in Appendix 13.7.2.

\[
u(k) = K_c \left(1 + \frac{h}{2T_1}\right) e(k) + \frac{h}{T_1} e(k-1) - \left(1 - \frac{h}{2T_1}\right) e(k-2) - \frac{2T_1}{h} \left(y(k) - 2y(k-1) + y(k-2))\right) + u(k-2)
\]

Formulae 1. Discrete implementation of an increment-type PID-regulator
8.3 Attitude Sensors

The program for the microprocessor controlling the attitude sensors is just as for the DC-motor card written in C. The conversion time of the A/D-converter is small, ~10 μs, and hence the real-time nature of the overall system should not be affected significantly in a negative way.

The PIC is programmed to read the complete set of attitude sensors in a straight sequence. The read results are continuously sent to the mainframe via RS-232. No data analysis or conversion of this data is done by the PIC or its peripherals, except for A/D-conversion. All internal communication is handled via the I2C-protocoll, where the PIC is set to be the master.

This microprocessor also handles temperature control of the part of the system housing the antenna and most of the electronics. A temperature sensor is read at a given frequency, and whenever the temperature drops below a predetermined value a number of heaters are set to start. This part of the program loop is completely internal and does not interact with the rest of the system. The attitude data acquisition starts automatically at power-up. For additional information, see the source-code in Appendix 13.7.3.

9 Tests

9.1 Mechanical

The mechanical design works as intended, the only flaw found is a small gap in the transition between the two conical gears handling rotation in the elevation plane. The glitch seems to be within the gearbox of the DC-motor, which a new elevation-motor and gearbox should take care of. This glitch result in the fact that the antenna might have a minor problem to stay fixed whenever the flight train is stable. It also induces small oscillating effects for the control system.

Depending on the thickness and sturdiness of the protective housing, which has not been built yet, our base plane could be somewhat over-dimensioned. It was designed
to be the main support of the system, which is not necessary if the housing is tough and stiff enough. By reducing the thickness of the base plane, about 450 grams of weight might be saved. The same is valid for the GPS base plane, which can be reduced to about half the thickness if the housing is made stiff enough, which would save another 430 grams.

There are no significant disturbances or losses of signals through the slip rings. The DC-motors run smoothly and are capable of holding the antenna steady thanks to the gearboxes. We are confident that the rotational speed of the system is sufficient based on previous calculations, and the mechanical system is very reliable overall. Before a first test flight the motors and angular sensors need to undergo tests in a vacuum chamber, as well as be tested under very cold conditions in order to assure functionality at an altitude of 30,000 meters.

9.2 Sensors

The angular sensors have been tested to see if they maintain an absolute read-out even after a significant number of reads. To this respect they fulfill the demands put on them accurately. However, further tests would be of interest to gain some statistically significant data on the absolute accuracy of the sensor. These tests will take some time and can preferably be combined into one single test to determine the overall accuracy of the pointing system. I.e. the sensors and motor driver system as a whole.

![Azimuth motor step response](image)

*Figure 23.*  *Azimuth motor step response*
9.3 Control System

The control system was tested by manually giving different angles towards a fictive target and letting the control system calculate the correct commands to the motors to then position the antenna accordingly. Hence, a step-response was tested. The results of the step response of each motor can be seen in Figure 23 and Figure 24. Also a continuous drift was added to test the systems ability to cancel the effect of a rotation of the balloon which is a probable occurrence during flight. Both tests were successful and the data was used to calculate the PID-variables, although more fine-tuning of the regulator variables are needed to achieve as smooth operation as possible.

![Elevation Step Response Graph](image)

*Figure 24. Elevation motor step response*

10 Results

10.1 Mechanical

The mechanical design works flawlessly in the lab environment that it has been exposed to. It provides a reliable base to continue developing a full scale attitude determination system on. It has been designed to withstand the environment at the operational height of a stratospheric balloon. The speed of the system is fully compliant with the design parameters, the final accuracy has not yet been determined.
10.2 GPS

The U-blox GPS receivers that were used to determine the attitude of the balloon were too inexact. To get the desired accuracy the antennas would have to be mounted on at least 100 meters distance from each other, which is not feasible. The position of the gondola is determined correctly and accurately.

![Figure 25. Final mechanical assembly, the antenna seen at the bottom is approximately 9x9 cm.](image)

10.3 Accelerometers and Magnetic Compass

The accelerometers provide attitude information with an accuracy of less than 0.35°. They are currently limited to a maximum tilt of 90°, so full 3D attitude determination is not implemented as is. The magnetic compass has not been implemented.

10.4 Electronics

The electronics designed for DE-Link has been proven to work very well in every aspect.
11 Future Work

11.1 GPS

The idea of using GPS for attitude determination is a good one. By using for instance one DSP\(^{23}\) instead of the three base-band processors, the problems with timing and choices of different satellites can be significantly reduced. New problems might arise, such as multiplexing three antennas to one input channel, but this can be solved.

This solution would be quite time consuming, and hence costly to develop, but might be a good investment in the long run compared to using three $4000 chips in each attitude determination system.

11.2 Programming

The ADA programming needs to be cleaned up since a lot of the routines created are obsolete at this point. The overall structure is made to provide easy to understand code rather than ultimate efficiency. In future flight models, restructuring can be necessary to improve performance.

11.3 Tests

A number of tests of the system should be done to gain a complete sense of how the system is performing and which improvements that can be made. Testing of the total pointing accuracy is needed. This can be done by fastening a laser diode on the antenna and then repeatedly letting the antenna point toward the same angles. By measuring the size of the area that the light is always within on a wall or similar some distance away, the overall accuracy can be practically decided and proven.

The total accuracy of the attitude sensor must also be tested. This can be done in a similar way by fastening a laser diode to the frame holding the sensors. Pointing the

\(^{23}\) DSP – Digital Signal Processor
frame toward the exact same point time after time and reading the sensor values will provide a statistically correct estimate of the accuracy of the sensors.

The whole system needs to be tested in a vacuum chamber to test its capability to withstand very low pressures and temperature in operating mode. By placing the system in a vacuum chamber the heat distribution of the system can also be tested. Both by allowing no external heat sources, thus measuring only internal heat contributors, and also by simulating one-sided sun exposure as would be the reality on day-time flights.

Long time system tests will provide exact knowledge of performance as well as accurate power consumption, which is only roughly estimated at this point.

12 Credentials

Rymdbolaget – Esrange, for letting us do our own thing our own way, and putting down the money to make this project happen.

Lars-Olov Jönsson, for helping us with all of our questions as well as giving us a new perspective on the development of electronic systems.

Mattias Robertsson at SolidWorks Corp. Northern Europe for supplying us with a time limited full version license of SolidWorks 2005.

13 Appendices
13.1 Design Flowchart
## 13.2 Budgets

### 13.2.1 Financial Budget

<table>
<thead>
<tr>
<th>Financial budget DE-Link</th>
<th>Amount</th>
<th>Cost</th>
<th>Currency</th>
<th>Total SEK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical Construction</td>
<td>1</td>
<td>34 300,00</td>
<td>SEK</td>
<td>34 300,00</td>
</tr>
<tr>
<td>GPS Antenna</td>
<td>3</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>GPS Receiver Module</td>
<td>3</td>
<td>1 550,00</td>
<td>SEK</td>
<td>1 550,00</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>1</td>
<td>38,00</td>
<td>SEK</td>
<td>38,00</td>
</tr>
<tr>
<td>PC/104+</td>
<td>1</td>
<td>149,00</td>
<td>SEK</td>
<td>149,00</td>
</tr>
<tr>
<td>Protective Housing</td>
<td>1</td>
<td>672,05</td>
<td>USD</td>
<td>4 704,35</td>
</tr>
<tr>
<td>Cable</td>
<td>TBD</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Misc. Nuts and bolts</td>
<td>TBD</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Slipring</td>
<td>1</td>
<td>3 675,00</td>
<td>SEK</td>
<td>3 675,00</td>
</tr>
<tr>
<td>RF Rotary Joint</td>
<td>2</td>
<td>136,50</td>
<td>SEK</td>
<td>273,00</td>
</tr>
<tr>
<td>Bearing</td>
<td>1</td>
<td>38,00</td>
<td>SEK</td>
<td>38,00</td>
</tr>
<tr>
<td>Gear, Rotation Motor</td>
<td>1</td>
<td>149,00</td>
<td>SEK</td>
<td>149,00</td>
</tr>
<tr>
<td>Gear, Rotation Sensor</td>
<td>1</td>
<td>149,00</td>
<td>SEK</td>
<td>149,00</td>
</tr>
<tr>
<td>Gear, Mounting Frame</td>
<td>2</td>
<td>47,50</td>
<td>SEK</td>
<td>95,00</td>
</tr>
<tr>
<td>Gear, Conical</td>
<td>2</td>
<td>470,20</td>
<td>SEK</td>
<td>940,40</td>
</tr>
<tr>
<td>Sensor</td>
<td>1</td>
<td>505,00</td>
<td>SEK</td>
<td>505,00</td>
</tr>
<tr>
<td>DC-Motor SCL12</td>
<td>1</td>
<td>665,00</td>
<td>SEK</td>
<td>665,00</td>
</tr>
<tr>
<td>DC-Motor SCL16</td>
<td>1</td>
<td>880,00</td>
<td>SEK</td>
<td>880,00</td>
</tr>
<tr>
<td>Gearbox SPG12</td>
<td>1</td>
<td>820,00</td>
<td>SEK</td>
<td>820,00</td>
</tr>
<tr>
<td>Gearbox SPG16</td>
<td>1</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Antenna</td>
<td>1</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Battery</td>
<td>1</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>PIC Circuit Board</td>
<td>1</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Electronic Components</td>
<td>1</td>
<td>2 759,52</td>
<td>SEK</td>
<td>2 759,52</td>
</tr>
<tr>
<td>Protective Housing</td>
<td>1</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Cable</td>
<td>TBD</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
<tr>
<td>Misc. Nuts and bolts</td>
<td>TBD</td>
<td>TBD</td>
<td>SEK</td>
<td>-</td>
</tr>
</tbody>
</table>

Exchange Rate in Calculations; 1,00 USD = 7,00 SEK

| Total Cost | 51 515,27 |
## 13.2.2 Mass Budget

### Massbudget DE-Link

<table>
<thead>
<tr>
<th>Part</th>
<th>Amount</th>
<th>Mass [gr]</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS Baseplane</td>
<td>1</td>
<td>913,0</td>
<td>913,00</td>
</tr>
<tr>
<td>GPS Antenna</td>
<td>3</td>
<td>TBD</td>
<td>3,00</td>
</tr>
<tr>
<td>GPS Receiver Module</td>
<td>3</td>
<td>TBD</td>
<td>3,00</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>1</td>
<td>297,0</td>
<td>297,00</td>
</tr>
<tr>
<td>PC/104+</td>
<td>1</td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Protective Housing</td>
<td>1</td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Cable</td>
<td></td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Misc. Nuts and bolts</td>
<td></td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Baseplane</td>
<td>1</td>
<td>856,0</td>
<td>856,00</td>
</tr>
<tr>
<td>Holder Rotation Motor</td>
<td>1</td>
<td>12,1</td>
<td>12,10</td>
</tr>
<tr>
<td>Holder Rotation Sensor</td>
<td>1</td>
<td>24,3</td>
<td>24,30</td>
</tr>
<tr>
<td>Slipping</td>
<td>1</td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>RF Rotary Joint</td>
<td>1</td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Adapter Tube</td>
<td>1</td>
<td>10,5</td>
<td>10,50</td>
</tr>
<tr>
<td>Mounting Frame Bottom</td>
<td>1</td>
<td>34,6</td>
<td>34,60</td>
</tr>
<tr>
<td>Mounting Frame Side</td>
<td>2</td>
<td>20,3</td>
<td>40,60</td>
</tr>
<tr>
<td>Holder Tilt Motor</td>
<td>1</td>
<td>30,5</td>
<td>30,50</td>
</tr>
<tr>
<td>Holder Tilt Sensor</td>
<td>1</td>
<td>27,4</td>
<td>27,40</td>
</tr>
<tr>
<td>Fastener Tilt Motor</td>
<td>1</td>
<td>6,7</td>
<td>6,70</td>
</tr>
<tr>
<td>Tilt Axis</td>
<td>1</td>
<td>10,5</td>
<td>10,50</td>
</tr>
<tr>
<td>Holder Antenna</td>
<td>1</td>
<td>34,5</td>
<td>34,50</td>
</tr>
<tr>
<td>Adapter Sensor Axis</td>
<td>1</td>
<td>0,8</td>
<td>0,80</td>
</tr>
<tr>
<td>Adapter Rotation Axis</td>
<td>1</td>
<td>0,1</td>
<td>0,10</td>
</tr>
<tr>
<td>Adaptor E-Link</td>
<td>4</td>
<td>45,5</td>
<td>182,00</td>
</tr>
<tr>
<td>Bearing</td>
<td>2</td>
<td>1,2</td>
<td>2,40</td>
</tr>
<tr>
<td>Gear, Rotation Motor</td>
<td>1</td>
<td>12,8</td>
<td>12,80</td>
</tr>
<tr>
<td>Gear, Rotation Sensor</td>
<td>1</td>
<td>56,1</td>
<td>56,10</td>
</tr>
<tr>
<td>Gear, Mounting Frame</td>
<td>1</td>
<td>49,5</td>
<td>49,50</td>
</tr>
<tr>
<td>Gear, Conical</td>
<td>2</td>
<td>2,8</td>
<td>5,60</td>
</tr>
<tr>
<td>Sensor</td>
<td>2</td>
<td>30,0</td>
<td>60,00</td>
</tr>
<tr>
<td>DC-Motor SCL12</td>
<td>1</td>
<td>28,4</td>
<td>28,40</td>
</tr>
<tr>
<td>DC-Motor SCL16</td>
<td>1</td>
<td>57,0</td>
<td>57,00</td>
</tr>
<tr>
<td>Antenna</td>
<td>1</td>
<td>160,0</td>
<td>160,00</td>
</tr>
<tr>
<td>Battery</td>
<td>1</td>
<td>876,0</td>
<td>876,00</td>
</tr>
<tr>
<td>PIC Circuit Board</td>
<td></td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Protective Housing</td>
<td>1</td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Cable</td>
<td></td>
<td>TBD</td>
<td>-</td>
</tr>
<tr>
<td>Misc. Nuts and bolts</td>
<td></td>
<td>TBD</td>
<td>-</td>
</tr>
</tbody>
</table>

| Total Mass [gr]                     |        | 3799,40   |
### 13.2.3 Power Budget

#### Power Budget DE-Link

<table>
<thead>
<tr>
<th>Part</th>
<th>Amount</th>
<th>Voltage</th>
<th>Current</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC/104+</td>
<td>1</td>
<td>5,0</td>
<td>1,15</td>
<td>5,75</td>
</tr>
<tr>
<td>GPS Receiver</td>
<td>1</td>
<td>12,0</td>
<td>0,08</td>
<td>0,96</td>
</tr>
<tr>
<td>GPS Antenna</td>
<td>3</td>
<td>3,3</td>
<td>0,02</td>
<td>0,15</td>
</tr>
<tr>
<td>Sensor Card</td>
<td>1</td>
<td>5,0</td>
<td>0,024</td>
<td>0,120</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>1</td>
<td>28,8</td>
<td>0,08</td>
<td>2,40</td>
</tr>
<tr>
<td>DC-Motor SCL16-2515</td>
<td>1</td>
<td>12,0</td>
<td>0,10</td>
<td>1,20</td>
</tr>
<tr>
<td>DC-Motor SCL12-2038</td>
<td>1</td>
<td>12,0</td>
<td>0,05</td>
<td>0,60</td>
</tr>
<tr>
<td>Losses (Approximation)</td>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
</tr>
</tbody>
</table>

Total Power Consumption [W] | 11,18
Total Available Power [W]  | 374,00
Operational Time [Hrs]     | 33,45
13.3 System of Schematics Numbering

General system of numbering has the following format:

DE-Link AA-BBCC

Where:

**AA**  –  Main section / type

01  Mechanical antenna section
02  Mechanical GPS section
03  Mechanical assembly
04  Electronics antenna control
05  Electronics sensors

**BB**  –  Individual part number, 01 - 99

**CC**  –  View

01  Front
02  Back
03  Left
04  Right
05  Top
06  Bottom
07  3D
08  Combination
09  Electrical
Appendix 13.4 – Mechanical Drawings
Appendix 13.4 – Mechanical Drawings

Dimensions are in millimeters. Tolerance: Linear 0.1 mm. Angular 0.1 mm.

Finish: A florine

Company Name: SSC - Esrange

Drawing Title: DE-Link Baseplane, 3D View

Drawing No: DE-Link 01-0107

Material: Aluminium

Weight: 896.0 gr

Scale: 1:12

Sheet 3 of 3
Appendix 13.4 – Mechanical Drawings

DE-Link Rotation Motor Holder
Top, Front, Right and 3D View

Company Name: SSC - Esrange

Material: Aluminium

Drawing: DE-Link 01-0208

Dimensions are in millimeters
Tolerances: Linear 0.1 mm
Angular 0.1 mm

NAME SIGNATURE DATE

DE-Link Rotation Motor Holder
Top, Front, Right and 3D View

Material: Aluminium

Drawing: DE-Link 01-0208

Dimensions are in millimeters
Tolerances: Linear 0.1 mm
Angular 0.1 mm

NAME SIGNATURE DATE

DE-Link Rotation Motor Holder
Top, Front, Right and 3D View

Material: Aluminium

Drawing: DE-Link 01-0208

Dimensions are in millimeters
Tolerances: Linear 0.1 mm
Angular 0.1 mm

NAME SIGNATURE DATE

DE-Link Rotation Motor Holder
Top, Front, Right and 3D View

Material: Aluminium

Drawing: DE-Link 01-0208

Dimensions are in millimeters
Tolerances: Linear 0.1 mm
Angular 0.1 mm

NAME SIGNATURE DATE

DE-Link Rotation Motor Holder
Top, Front, Right and 3D View

Material: Aluminium

Drawing: DE-Link 01-0208
Appendix 13.4 – Mechanical Drawings
Appendix 13.4 – Mechanical Drawings
Appendix 13.4 – Mechanical Drawings

DIMENSIONS ARE IN METERS
TOURANCES:
LINEAR: 0.1 mm
ANGULAR: 0.1 mm

COMPANY NAME:
SSC - Esrange

DE-Link Fastener Tilt Motor
Top, Front, Right and 3D View

NAME
DRAWN
CHECKED
APPROVED
MFG
G.A.

MATERIAL: Aluminium

DE-Link 01-0908

A4

WEIGHT: 67 gr

SCALE: 1:1

SHEET 1 OF 1
Appendix 13.4 - Mechanical Drawings

DE-Link 01-1108

Dimensions are in millimeters.

Tolerances:
- Linear: ±0.1 mm
- Angular: ±0.1 mm

Material: Aluminium

Weight: 34.5 g

Scale: 1:1

Sheet 1 of 1
Appendix 13.4 – Mechanical Drawings
Appendix 13.4 – Mechanical Drawings

---

**Dimensions**
- **Diameters**: 50.50
- **Other Dimensions**: 22.59, 15, 4.50

**Company Name**: SSC - Eranges

**Drawing Details**
- **Drawing Title**: DE-Link Rotation Gear, Mounting Frame
- **Drawing Description**: Top, Front, Right and 3D View
- **Material**: Aluminium
- **Scale**: 1:1
- **Drawing Number**: DE-Link 01-1608
- **Revision**: 4.0

**Notes**
- Linear Tolerance: 0.1 mm
- Angular Tolerance: 0.1 mm

---

**Scale**: 1:1

**Sheet**: 1 of 1
Appendix 13.4 – Mechanical Drawings

Dimensions are in millimeters
Tolerances:
Linear: ±0.1 mm
Angular: ±0.1 mm

Finish: Alodine

Company Name:
SSC - Esrange

Drawing Title:
DE-Link GPS Baseplane, Front View

Drawing Number:
DE-Link 02-0101

Material:
Aluminium

Weight:
930 g

Scale: 1:2

Sheet 1 of 3
Appendix 13.4 – Mechanical Drawings
13.5 Electronic Schematics
Appendix 13.5 – Electronic Schematics
Appendix 13.5 – Electronic Schematics
Appendix 13.5 – Electronic Schematics
13.6 Antenna Data Sheet

VP165/24 - Vertical Flat Panel Antenna, 65 deg, 9 dBi

<table>
<thead>
<tr>
<th>Type</th>
<th>Directional flat panel antenna</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polarisation</td>
<td>Vertical</td>
</tr>
<tr>
<td>Frequency</td>
<td>2400-2485 MHz</td>
</tr>
<tr>
<td>Impedance</td>
<td>50 Ω</td>
</tr>
<tr>
<td>VSWR</td>
<td>&lt;1.5:1</td>
</tr>
<tr>
<td>Gain</td>
<td>9 dBi</td>
</tr>
<tr>
<td>H-Peak Width</td>
<td>Azimuth 65°</td>
</tr>
<tr>
<td></td>
<td>Elevation 65°</td>
</tr>
<tr>
<td>Connector</td>
<td>(1) SMA female or</td>
</tr>
<tr>
<td></td>
<td>(2) white RG58 3-10 m with</td>
</tr>
<tr>
<td></td>
<td>male SMA connector</td>
</tr>
<tr>
<td>Power Handling</td>
<td>20 W</td>
</tr>
<tr>
<td>Mount</td>
<td>Wall/mast mount with</td>
</tr>
<tr>
<td></td>
<td>Az/EL tilt function included</td>
</tr>
<tr>
<td>Size</td>
<td>W 92 mm</td>
</tr>
<tr>
<td></td>
<td>H 100 mm</td>
</tr>
<tr>
<td></td>
<td>D 30 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>160 g antenna only</td>
</tr>
<tr>
<td>Material</td>
<td>Element - aluminium, brass</td>
</tr>
<tr>
<td></td>
<td>Rosturm - ABS</td>
</tr>
<tr>
<td>Color</td>
<td>White</td>
</tr>
<tr>
<td>Other</td>
<td>DC grounded</td>
</tr>
</tbody>
</table>

Available models:
- VP165/24 vertical panel antenna, 65 deg, 9 dBi
- VP270/24 vertical panel antenna, 70 deg, 12 dBi
- VP470/24 vertical panel antenna, 70 deg, 14 dBi

Azimuth (H-field) radiation pattern

Elevation (E-field) radiation pattern
13.7 Program Code

13.7.1 Balloon Data Analysis, ADA

--Balloon.adb
--Calculates maximum movement of stratospheric balloon data
with Ada.Text_IO; use Ada;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Direct_IO;

procedure Balloon is

--Creates package for character read
package Files is new Ada.Direct_IO(Character);
use Files;

--Creates package for saving data after calculation
package Float_Output is new Ada.Text_IO.Float_IO(Long_Float);

--Variables needed for calculation
  Data_File : File_Type;
  Input : Character;
  Decade : Integer;
  Data : array (1..65536) of Integer;
  Data_F : array (1..65536) of Long_Float;
  Time : array (1..65536) of Long_Float;
  Compass : array (1..65536) of Long_Float;
  Pitch : array (1..65536) of Long_Float;
  Roll : array (1..65536) of Long_Float;
  Temp : array (1..15) of Character;
  Temp_Index : Integer;
  Index : Integer := 1;
  Index_F : Integer := 1;
  Value : Integer;
  Block : Integer := 0;
  Decade_F : Long_Float;
  Compass_Max_Speed : Long_Float := 0.0;
  Pitch_Max_Speed : Long_Float := 0.0;
  Roll_Max_Speed : Long_Float := 0.0;
  Compass_Max : Long_Float := 0.0;
  Pitch_Max : Long_Float := 0.0;
  Roll_Max : Long_Float := 0.0;
  Data_Out : Text_IO.File_Type;
  Throw : Integer := 0;
  Throw_B : Boolean := False;
  CO_Time : array (1..3) of Integer;
  BS : Integer := 0;

begin
--Get input from user to set up calculation correctly
Appendix 13.7.1 – Program Code Balloon Data Analysis

```plaintext
Text_IO.Put_Line("Please copy data file to 'Data.txt'");
Text_IO.Put_Line("Check the FTD file for number of lines of each data block!");
Text_IO.Put("Enter Block Size: ");
Get(BS);
Text_IO.New_Line;
Text_IO.Put_Line("Check the FTD file for Start Time (UTC)");
Get(Start(1));
Text_IO.Put("Enter Start Time(MM): ");
Get(Start(2));
Text_IO.Put("Enter Start Time(SS): ");
Get(Start(3));
Text_IO.New_Line;
Text_IO.Put("Enter Cut-Off Time(HH): ");
Get(COT(1));
Text_IO.Put("Enter Cut-Off Time(MM): ");
Get(COT(2));
Text_IO.Put("Enter Cut-Off Time(SS): ");
Get(COT(3));
Start_Time := Long_Float(Start(1)*3600 + Start(2)*60 + Start(3));
CO_Time := Long_Float(COT(1)*3600 + COT(2)*60 + COT(3));
Text_IO.New_Line;
Text_IO.Create(Data_Out, Text_IO.Out_File, "Output.txt");
Text_IO.Put_Line(Data_Out, "Start Time: " & Integer’Image(Start(1)) & ": " & Integer’Image(Start(2)) & ": " & Integer’Image(Start(3)));
Text_IO.Put_Line(Data_Out, "COT Time: " & Integer’Image(COT(1)) & ": " & Integer’Image(COT(2)) & ": " & Integer’Image(COT(3)));

--Opens data file and reads in data
Open(Data_File, In_File, "Data.txt");

loop
  --Reads the lines of each block except the last line
  for i in 1..BS-1 loop
    loop
      Data(Index) := 0;
      Temp_Index := 0;
      loop
        Read(Data_File, Input);
        Temp_Index := Temp_Index + 1;
        Temp(Temp_Index) := Input;
        exit when Input = ‘’ or Input = ASCII.CR or Input = ASCII.LF;
      end loop;
      Decade := 1;
      for j in reverse 1..Temp_Index-1 loop
        case Temp(j) is
          when ‘0’ => Value := 0;
          when ‘1’ => Value := 1;
          when ‘2’ => Value := 2;
        end case;
    end loop;
  end loop;
end loop;
```

97
when '3' =>
  Value := 3;
when '4' =>
  Value := 4;
when '5' =>
  Value := 5;
when '6' =>
  Value := 6;
when '7' =>
  Value := 7;
when '8' =>
  Value := 8;
when '9' =>
  Value := 9;
when ' ' =>
  Value := 0;
  Decade := Decade/10;
  Data(Index) := Data(Index) * (-1);
when ASCII.LF =>
  Value := 0;
when others =>
  Text_IO.New_Line;
  Text_IO.Put(Temp(j));
  Text_IO.Put_Line("ERROR");
  Value := 0;
end case;
Data(Index) := Data(Index) + Decade * Value;
Decade := Decade * 10;
end loop;
if Temp_Index > 1 then
  Index := Index + 1;
end if;
exit when Input = ASCII.CR;
end loop;
end loop;

-- Read in the last line of the block
loop
  Data_F(Index_F) := 0.0;
  Temp_Index := 0;
loop
    Read(Data_File, Input);
    Temp_Index := Temp_Index + 1;
    Temp(Temp_Index) := Input;
  exit when Input = ' ' or Input = ASCII.CR or Input = ASCII.LF;
end loop;
  Decade_F := 0.000001;
for j in reverse 1..Temp_Index-1 loop
  case Temp(j) is
  when '0' =>
    Value := 0;
  when '1' =>
    Value := 1;
end case;
when '2' =>
  Value := 2;
when '3' =>
  Value := 3;
when '4' =>
  Value := 4;
when '5' =>
  Value := 5;
when '6' =>
  Value := 6;
when '7' =>
  Value := 7;
when '8' =>
  Value := 8;
when '9' =>
  Value := 9;
when '.' =>
  Value := 0;
Decade_F := Decade_F / 10.0;
when ASCII.LF =>
  Value := 0;
when others =>
  Text_IO.Put_Line("ERROR");
Value := 0;
end case;
Data_F(Index_F) := Data_F(Index_F) * (-1.0);
Decade_F := Decade_F * 10.0;

if Temp_Index > 1 then
  Index_F := Index_F + 1;
end if;
exit when Input = ASCII.CR;

--Calculate and save the data
Block := Block + 1;
Time(Block) := Long_Float(Data(2)) * 3600.0 +
  Long_Float(Data(3)) * 60.0 +
  Long_Float(Data(4)) + Long_Float(Data(5)) / 1000.0;
Compass(Block) := Data_F(1);
Pitch(Block) := Data_F(2);
Roll(Block) := Data_F(3);
Index := 1;
Index_F := 1;
Text_IO.Put(ASCII.CR & "Reading Block: " & Integer'Image(Block));
if Time(Block) < Start_Time then
  Block := Block - 1;
  goto Skip;
end if;
if Time(Block) > CO_Time then
Appendix 13.7.1 – Program Code Balloon Data Analysis

```
Block := Block - 1;
raise End_Error;
end if;
<<Skip>> null;
end loop;

--Exception handling at end of file
exception
  when End_Error =>
    Close(Data_File);
    Text_IO.New_Line;
    Text_IO.Put_Line("Read Complete!");
    Text_IO.Put_Line("Data Blocks Read: " & Integer’Image(Block));

--Goes through data and retrieves the largest values
for i in 1..(Block-1) loop
  if Time(i+1)-Time(i) < 1.0 then
    if Compass(i+1)-Compass(i) < -180.0 then
      if Compass_Max_Speed < abs((Compass(i+1)-Compass(i)+360.0)/
        Time(i+1)-Time(i)) then
        if abs((Compass(i+1)-Compass(i)+360.0)/(Time(i+1)-
          Time(i))**2) > 90.0 then
          --Unreasonable acceleration! Data thrown!
          Throw := Throw + 1;
          Throw_B := True;
        else
          Compass_Max_Speed := abs((Compass(i+1)-Compass(i)+
            360.0)/(Time(i+1)-Time(i)));
          Text_IO.Put(ASCII.CR & "Compass_Max_Speed: " &
            Long_Float’Image(Compass_Max_Speed) &
            " Block: " & Integer’Image(i));
          end if;
        end if;
      else
        if Compass(i+1)-Compass(i) > 180.0 then
          if Compass_Max_Speed < abs((Compass(i+1)-Compass(i)-
            360.0)/(Time(i+1)-Time(i)) then
            if abs((Compass(i+1)-Compass(i)-360.0)/
              (Time(i+1)-Time(i))**2) > 90.0 then
              --Unreasonable acceleration! Data thrown!
              Throw := Throw + 1;
              Throw_B := True;
            else
              Compass_Max_Speed := abs((Compass(i+1)-
                Compass(i)-360.0)/(Time(i+1)-Time(i)));
              Text_IO.Put(ASCII.CR & "Compass_Max_Speed: " &
                Long_Float’Image(Compass_Max_Speed) &
                " Block: " & Integer’Image(i));
            end if;
          else
            Compass_Max_Speed := abs((Compass(i+1)-
              Compass(i)-360.0)/(Time(i+1)-Time(i)));
            Text_IO.Put(ASCII.CR & "Compass_Max_Speed: " &
              Long_Float’Image(Compass_Max_Speed) &
              " Block: " & Integer’Image(i));
          end if;
        else
          if Compass_Max_Speed < abs((Compass(i+1)-Compass(i))/
            (Time(i+1)-Time(i)) then
```

100
if abs((Compass(i+1)-Compass(i)+360.0)/(Time(i+1)-Time(i))**2) > 90.0 then
  --Unreasonable acceleration! Data thrown!
  Throw := Throw + 1;
  Throw_B := True;
else
  Compass_Max_Speed := abs((Compass(i+1)-Compass(i))/(Time(i+1)-Time(i)));
  Text_IO.Put(ASCII.CR & "Compass_Max_Speed: " & Long_Float’Image(Compass_Max_Speed) & " Block: " & Integer’Image(i));
  end if;
end if;
if not Throw_B then
  if Compass_Max < abs(Compass(i)) then
    Compass_Max := abs(Compass(i));
  end if;
  if Pitch_Max_Speed < abs((Pitch(i+1)-Pitch(i))/(Time(i+1)-Time(i))) then
    Pitch_Max_Speed := abs((Pitch(i+1)-Pitch(i))/(Time(i+1)-Time(i)));
  end if;
  if Pitch_Max < abs(Pitch(i)) then
    Pitch_Max := abs(Pitch(i));
  end if;
  if Roll_Max_Speed < abs((Roll(i+1)-Roll(i))/(Time(i+1)-Time(i))) then
    Roll_Max_Speed := abs((Roll(i+1)-Roll(i))/(Time(i+1)-Time(i)));
  end if;
  if Roll_Max < abs(Roll(i)) then
    Roll_Max := abs(Roll(i));
  end if;
  Float_Output.Put(Data_Out, (Compass(i+1)-Compass(i))/(Time(i+1)-Time(i)));
  Float_Output.Put(Data_Out, (Pitch(i+1)-Pitch(i))/(Time(i+1)-Time(i)));
  Float_Output.Put(Data_Out, (Roll(i+1)-Roll(i))/(Time(i+1)-Time(i)));
  Text_IO.New_Line(Data_Out);
else
  Throw := Throw + 1;
  end if;
  Throw_B := False;
end loop;
--Prints the results to screen
Text_IO.New_Line;
Text_IO.Put_Line("Done! Data saved in ‘Output.txt’!");
Text_IO.Put_Line(Data_Out, "-- Results --");
Appendix 13.7.1 – Program Code Balloon Data Analysis

```plaintext
Text_IO.New_Line(Data_Out);
Text_IO.Put_Line(Data_Out, "Compass_Max_Speed: " &
    Long_Float'Image(Compass_Max_Speed) & " deg/s");
Text_IO.Put_Line(Data_Out, "Pitch_Max_Speed: " &
    Long_Float'Image(Pitch_Max_Speed) & " deg/s");
Text_IO.Put_Line(Data_Out, "Roll_Max_Speed: " &
    Long_Float'Image(Roll_Max_Speed) & " deg/s");
Text_IO.New_Line(Data_Out);
Text_IO.Put_Line(Data_Out, "Compass_Max: " &
    Long_Float'Image(Compass_Max) & " deg");
Text_IO.Put_Line(Data_Out, "Pitch_Max: " &
    Long_Float'Image(Pitch_Max) & " deg");
Text_IO.Put_Line(Data_Out, "Roll_Max: " &
    Long_Float'Image(Roll_Max) & " deg");
Text_IO.New_Line(Data_Out);
Text_IO.Put_Line(Data_Out, "Throws: " & Integer'Image(Throw) & " times");
Text_IO.Close(Data_Out);
end Balloon;
```
# 13.7.2 PWM Motor Controller PIC, C

// Setup and Configuration of PIC
#include <18F2321.H>
#include <STDLIB.H>
#fuses H4,NOBROWNOUT,WDT128,NOPROTECT,NODEBUG,NOLVP
#use delay (clock=40000000)  // Clock speed defined as 40MHz
#use rs232 (baud=9600,xmit=PIN_C6,rcv=PIN_C7, BRGH1OK)

// Masks
- int mask=0x1F;  // Mask to take bit 1-5
- int drive1,drive2;
- int low=0xFF;

// Variables
- int32 tt=0;
- int32 ttt=0;

- signed int32 newelev,  
  newazim;

- signed int32 target[2],  //0 = AZ, 1 = EL
  target_tmp[2];

- int channel, tmp, tmp2,i,j,k;
- int plot=false;
- int pll=false;
- char init,data_in[6];
- unsigned int Ck_A,Ck_B;

// PID Variables
- signed int32 K0[2],  //0 = az, 1 = el
  Ti[2],  //0 = az, 1 = el
  Td[2],  //0 = az, 1 = el
  h;  //time between calculation

- signed int32 Antenna[2][3]; //Stores antenna position
- signed int32 Error[2][3];  //Stores error, including past two errors!
- signed int32 Motor[2][3]; //Stores motor speed
- signed int32 Temp1,Temp2,Temp3,Temp4;
- long target_temp1, target_temp2;
// Appendix 13.7.2 – Program Code Motor Controller

```
void Decode(){ //Decodes recieved characters to target azimuth and elevation
    (target_tmp[0]) = ((data_in[0]) & mask);
    (target_tmp[1]) = ((data_in[1]) & mask);
    target_temp1 = (((target_tmp[0]) << 5) + (target_tmp[1]));
    (target[0]) = target_temp1;
    (target_tmp[0]) = ((data_in[2]) & mask);
    (target_tmp[1]) = ((data_in[3]) & mask);
    target_temp2 = (((target_tmp[0]) << 5) + (target_tmp[1]));
    (target[1]) = target_temp2;
}

void Drive(){
    //Set direction!
    if((Motor[0][0]) >= 0){
        output_high(31760);
        (Motor[0][0]) = (Motor[0][0])/2;
    } else{
        output_low(31760);
        (Motor[0][0]) = (Motor[0][0])/-2;
    }
    if((Motor[1][0]) >= 0){
        output_high(31763);
        (Motor[1][0]) = (Motor[1][0])/2;
    } else{
        output_low(31763);
        (Motor[1][0]) = (Motor[1][0])/-2;
    }

    //Set the speed!!
    drive1 = (Motor[0][0] & low);
    drive2 = (Motor[1][0] & low);
    set_pwm1_duty(drive1);
    set_pwm2_duty(drive2);
}

void PID(){ //Do actual regulation!
    //Buffer the Antenna position!
    (Antenna[0][0]) = (newazim);
    (Antenna[1][0]) = (newelev);

    //Correct antenna angles to set North and Down!
    (Antenna[0][0]) = ((Antenna[0][0]) - 775);
    (Antenna[1][0]) = (475 - (Antenna[1][0]));
    if((Antenna[0][0]) > 1000){
        (Antenna[0][0]) = ((Antenna[0][0]) - 970);
    }
```
else if((Antenna[0][0]) < 30){
    (Antenna[0][0]) = ((Antenna[0][0]) + 970);
}

if((Antenna[1][0]) > 1000){
    (Antenna[1][0]) = ((Antenna[1][0]) - 970);
}
else if((Antenna[1][0]) < 30){
    (Antenna[1][0]) = ((Antenna[1][0]) + 970);
}

for(j=0;j<=1;++j){ //Do twice, one time for each direction!

    //Calculate error
    (Error[j][0]) = ((Target[j]) - (Antenna[j][0]));

    //Correct error to make motor go the shortest way!
    if((Error[j][0]) >= 485){
        (Error[j][0]) = ((Error[j][0]) - 970);
    }
    else if((Error[j][0] <= -485)){
        (Error[j][0]) = ((Error[j][0]) + 970);
    }

    //Do regulation!
    Temp1 = ((16+((Ti[j])/(16*h)))*(Error[j][0]));
    Temp2 = (((Ti[j])/(8*h))*(Error[j][1]));
    Temp3 = ((16-(Ti[j])/(16*h)))*(Error[j][2]);
    Temp4 = (((256*h)/(Td[j]))*((Antenna[j][0])-(2*(Antenna[j][1]))+(Antenna[j][2])));
    (Motor[j][0]) = (((K0[j])*(Temp1 + Temp2 - Temp3 - Temp4))/4096+Motor[j][2]);

    //Prevent too large speeds!
    if((Motor[j][0]) >= 510){
        (Motor[j][0]) = 510;
    }
    if((Motor[j][0]) <= -510){
        (Motor[j][0]) = -510;
    }

    //Save old data!
    for(k=2;k>=1;--k){
        (Motor[j][k]) = (Motor[j][k-1]);
        (Error[j][k]) = (Error[j][k-1]);
        (Antenna[j][k]) = (Antenna[j][k-1]);
    }
}

Drive(); //Call the motor driver!

// -------------------------------
// - Interrupts -
Appendix 13.7.2 – Program Code Motor Controller

// ------------------

#define INT_AD //Handles the input of angles from the sensors
void adc_handler()
{
    if (channel == 2)
    {
        newelev = read_adc(adc_read_only);
        set_adc_channel(3);
        channel = 3;
        if ((newelev < 30) || (newelev > 1000))
        {
            newelev = 1000;
        }
        if ((newelev > 1000) || (newelev < 30))
        {
            newelev = 30;
        }
    }
    else if (channel == 3)
    {
        newazim = read_adc(adc_read_only);
        set_adc_channel(2);
        channel = 2;
        if ((newazim < 30) || (newazim >= 1000))
        {
            newazim = 1000;
        }
        if ((newazim > 1000) || (newazim <= 30))
        {
            newazim = 30;
        }
    }
}

#define INT_TIMER1
void adc_starter() //Starts ADC
{
    set_timer1(64536);
    read_adc(adc_start_only);
}

#define INT_TIMER0
void PID_starter() //Starts PID loop
{
    setTimer0(15536);
    PID();
}

#define INT_RDA
void rs232_data() //Receives data from PC/104
{
    restart_wdt();
    init = fgetc();
    if (init == 0x0E)
    {
        for (i = 0; i <= 5; ++i)
        {
            data_in[i] = fgetc();
        }
        //Calculate checksum!
        Ck_A = 0;
        Ck_B = 0;
        for (i = 0; i <= 3; ++i)
Appendix 13.7.2 – Program Code Motor Controller

```c
Ck_A = (Ck_A + (data_in[i]));
Ck_B = (Ck_B + Ck_A);
if((Ck_A == (data_in[4])) && (Ck_B == (data_in[5]))){
    Decode();
}
```

```c
// ---------------------
// - Main Program -
// ---------------------

void main(){
    // Initial Configuration
    setup_adc_ports(NO_ANALOGS); // All I/O-PINS are digital
    setup_adc(ADC_CLOCK_INTERNAL); // Sets the time for each analog sample
    setup_ccp1(CCP_PWM); // Configure CCP1 as a PWM
    setup_ccp2(CCP_PWM); // Configure CCP1 as a PWM
    setup_timer_0(RTCC_DIV_2); // Used for the PID loop! One count is 0.2us!
    setup_timer_1(T1_INTERNAL | T1_DIV_BY_2); // Used for sampling data! 0.1us!
    setup_timer_2(T2_DIV_BY_1, 255, 1);
    set_pwm1_duty(0); // Initializes motor speeds to zero
    set_pwm2_duty(0); // Initializes motor speeds to zero
    enable_interrupts(INT_AD);
    enable_interrupts(INT_TIMER0);
    enable_interrupts(INT_TIMER1);
    enable_interrupts(INT_RDA);
    enable_interrupts(GLOBAL);
    // Start first read!
    channel = 2;
    set_timer1(64536); // Starts ADC every 100us
    set_timer0(15536); // Starts PID every 2ms
    // Set up PID variables!
    K0[0] = 192;
    K0[1] = 56;
    Ti[0] = 100000;
    Ti[1] = 100000;
    Td[0] = 25000;
    Td[1] = 60000;
    h = 10000; // in us!
    // Set the start values!
```
Motor[0][0] = 0;
Motor[1][0] = 0;
Target[0] = target_temp1;
Target[1] = target_temp2;

output_high(PIN_B0);
delay_us(100);
output_low(PIN_B0);

setup_wdt(WDT_ON);

while(TRUE){
    // Create an infinite loop
}
13.7.3 Attitude Sensor PIC, C

// Program controls sensor readings from accelerometers and magnetometers, and sends the information to the mainframe via RS232.

// Setup and Configuration of PIC

#include <18F2331.H>
#include <STDLIB.H>

#fuses H4,NOBROWNOUT,NOWDT,NOPROTECT,NODEBUG,NOLVP
#use delay (clock=40000000)       // Clock speed defined as 40MHz
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK) // Configures serial-port communication
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C5)

// Definition of variables

- long x1_temp_top, x1_temp_bottom, x2_temp_top, x2_temp_bottom;
- char x1_out_top, x1_out_bottom, x2_out_top, x2_out_bottom;
- long y1_temp_top, y1_temp_bottom, y2_temp_top, y2_temp_bottom;
- char y1_out_top, y1_out_bottom, y2_out_top, y2_out_bottom;
- long z1_temp_top, z1_temp_bottom, z2_temp_top, z2_temp_bottom;
- char z1_out_top, z1_out_bottom, z2_out_top, z2_out_bottom;

- long card_temperature;

- long temporary1, temporary2, temporary3;

- long value_magnetometer_x, magnetometer_x_temp_top, magnetometer_x_temp_bottom;
- long value_magnetometer_y, magnetometer_y_temp_top, magnetometer_y_temp_bottom;
- long value_magnetometer_z, magnetometer_z_temp_top, magnetometer_z_temp_bottom;
- char magnetometer_x_out_top, magnetometer_x_out_bottom;
- char magnetometer_y_out_top, magnetometer_y_out_bottom;
- char magnetometer_z_out_top, magnetometer_z_out_bottom;

- int p = 0;
- int data1, data2;
- int data[12];
- unsigned int Ck_A, Ck_B;
- char data_out[6];

- long acc_value[6];
- long sign;
- long tempvalue;
signed long abstemp;
char send;
char trash;

char LF = 0x0A; // Line Feed
char ok = 0xFF; // 'OK' is sent in the end of each series of data via RS232
char E = 0x45; // Used to sync the communication with the PIC/104+
char F = 0x46; // Following information contains accelerometer X-values
char G = 0x47; // Following information contains accelerometer Y-values
char H = 0x48; // Following information contains accelerometer Z-values
char I = 0x49; // Following information contains magnetometer Y-values
char J = 0x4A; // Following information contains magnetometer Z-values

long top_mask = 0x03E0; // Mask to take bit 9 and 10, all others are set to 0
long bottom_mask = 0x001F; // Mask to take bit 1 to 8, all others are set to 0
int Add = 0xE0;
long value_mask = 0x0FFC; // Mask to take bit 3 to 12, all others are set to 0
long temp_mask = 0xFFC0; // Mask to take bit 7 to 16, all others are set to 0
long sign_mask = 0x0200; // Mask to take bit 10, all others are set to 0
long tempvalue_mask = 0x01FF; // Mask to take bit 1 to 9, all others are set to 0

// Definition of Functions

// Following 3 function reads accelerometer 'X/Y/Z-value', and sends the information to the RS232-port in 2 x 2 char,
// first char gives the information from bit 6-10, second char gives bit 0-5 -> total information is 10-bit.

void accelerometer_x(){
    x1_temp_top = (acc_value[0] & top_mask);
    x1_out_top = ((x1_temp_top >> 5) | Add);
    x1_temp_bottom = (acc_value[0] & bottom_mask);
    x1_out_bottom = (x1_temp_bottom | Add);

    x2_temp_top = (acc_value[1] & top_mask);
    x2_out_top = ((x2_temp_top >> 5) | Add);
    x2_temp_bottom = (acc_value[1] & bottom_mask);
    x2_out_bottom = (x2_temp_bottom | Add);

    //Resave in send variable!
    data_out[0] = E;
    data_out[1] = F;
    data_out[2] = x1_out_top;
    data_out[3] = x1_out_bottom;
    data_out[4] = x2_out_top;
    data_out[5] = x2_out_bottom;

    //Create checksum!
    Ck_A = 0;
}
Appendix 13.7.3 – Program Code Attitude Sensor

```c
Ck_B=0;

for(i=0;i<5;++i)
{
    Ck_A = (Ck_A + (data_out[i]));
    Ck_B = Ck_B + Ck_A;
}

for(i=0;i<5;++i)
{
    putc(data_out[i]);
}
putc(Ck_A);
putc(Ck_B);
putc(0x0A);
putc(0x0D);
delay_ms(10);

void accelerometer_y()
{
    y1_temp_top = (acc_value[2] & top_mask);
    y1_out_top = ((y1_temp_top >> 5) | Add);
    y1_temp_bottom = (acc_value[2] & bottom_mask);
    y1_out_bottom = (y1_temp_bottom | Add);

    y2_temp_top = (acc_value[3] & top_mask);
    y2_out_top = ((y2_temp_top >> 5) | Add);
    y2_temp_bottom = (acc_value[3] & bottom_mask);
    y2_out_bottom = (y2_temp_bottom | Add);

    //Resave in send variable!
    data_out[0] = E;
    data_out[1] = G;
    data_out[2] = y1_out_top;
    data_out[3] = y1_out_bottom;
    data_out[4] = y2_out_top;
    data_out[5] = y2_out_bottom;

    //Create checksum!
    Ck_A=0;
    Ck_B=0;
    for(i=0;i<5;++i)
    {
        Ck_A = (Ck_A + (data_out[i]));
        Ck_B = Ck_B + Ck_A;
    }
    for(i=0;i<5;++i)
    {
        putc(data_out[i]);
    }
    putc(Ck_A);
    putc(Ck_B);
}
```
Appendix 13.7.3 – Program Code Attitude Sensor

```c
#define putc(0x0A);
#define putc(0x0D);
#define delay_ms(10);

void accelerometer_z()
{
    z1_temp_top = (acc_value[4] & top_mask);
    z1_out_top = ((z1_temp_top >> 5) | Add);
    z1_temp_bottom = (acc_value[4] & bottom_mask);
    z1_out_bottom = (z1_temp_bottom | Add);

    z2_temp_top = (acc_value[5] & top_mask);
    z2_out_top = ((z2_temp_top >> 5) | Add);
    z2_temp_bottom = (acc_value[5] & bottom_mask);
    z2_out_bottom = (z2_temp_bottom | Add);

    // Resave in send variable!
    data_out[0] = E;
    data_out[1] = H;
    data_out[2] = z1_out_top;
    data_out[3] = z1_out_bottom;
    data_out[4] = z2_out_top;
    data_out[5] = z2_out_bottom;

    // Create checksum!
    Ck_A = 0;
    Ck_B = 0;
    for(i=0;i<=5;++i)
    {
        Ck_A = (Ck_A + (data_out[i]));
        Ck_B = Ck_B + Ck_A;
    }
    for(i=0;i<=5;++i)
    {
        putc(data_out[i]);
    }
    putc(Ck_A);
    putc(Ck_B);
    putc(0x0A);
    putc(0x0D);
    delay_ms(10);
}
```

// Following 3 functions reads magnetometer X/Y/Z-value, and sends the information to the RS232-port
// The information in value_magnetometer_x is split into two char, first char gives the information from
// bit 9-10,
// second char gives bit 1-8 -> total information is 10-bit. The character 0xFF is sent when
// transmission is completed. Each magnetometer is read twice, once for S/R+ and once for S/R-.
```c
void magnetometer_x(){
    output_low(PIN_C1); // Reset-low
    delay_us(2);
    output_high(PIN_C0); // Set-high
    set_adc_channel(1); // Chooses AN1 (PIN_RA1) as input
    delay_us(10);
    value_magnetometer_x = read_adc();
    magnetometer_x_temp_top = (value_magnetometer_x & top_mask);
    magnetometer_x_out_top = (magnetometer_x_temp_top >> 8);
    magnetometer_x_temp_bottom = (value_magnetometer_x & bottom_mask);
    magnetometer_x_out_bottom = magnetometer_x_temp_bottom;
    // puts(H);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_x_out_top);
    // puts(LF);
    // puts(magnetometer_x_out_bottom);
    // puts(LF);
    output_low(PIN_C0); // Set-low
    delay_us(2);
    output_high(PIN_C1); // Reset-high
    output_high(PIN_C1); // Sets PIN_RC1 high to set/reset magnetometer
    delay_us(2);
    output_low(PIN_C1); // Sets PIN_RC1 low
    set_adc_channel(1); // Chooses AN1 (PIN_RA1) as input
    delay_us(10);
    value_magnetometer_x = read_adc();
    magnetometer_x_temp_top = (value_magnetometer_x & top_mask);
    magnetometer_x_out_top = (magnetometer_x_temp_top >> 8);
    magnetometer_x_temp_bottom = (value_magnetometer_x & bottom_mask);
    magnetometer_x_out_bottom = magnetometer_x_temp_bottom;
    // puts(H);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_x_out_top);
    // puts(LF);
    // puts(LF);
}
// puts(magnetometer_x_out_bottom);
// puts(LF);
}

void magnetometer_y()
{
    output_low(PIN_C1); // Reset-low
    delay_us(2);
    output_high(PIN_C0); // Set-high
    set_adc_channel(2); // Chooses AN2 (PIN_RA2) as input
    delay_us(10);
    value_magnetometer_y = read_adc();
    magnetometer_y_temp_top = (value_magnetometer_y & top_mask);
    magnetometer_y_out_top = (magnetometer_y_temp_top >> 8);
    magnetometer_y_temp_bottom = (value_magnetometer_y & bottom_mask);
    magnetometer_y_out_bottom = magnetometer_y_temp_bottom;
    // puts(I);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_y_out_top);
    // puts(LF);
    // puts(magnetometer_y_out_bottom);
    // puts(LF);
    output_low(PIN_C0); // Set-low
    delay_us(2);
    output_high(PIN_C1); // Reset-high
    set_adc_channel(2); // Chooses AN2 (PIN_RA2) as input
    delay_us(10);
    value_magnetometer_y = read_adc();
    magnetometer_y_temp_top = (value_magnetometer_y & top_mask);
    magnetometer_y_out_top = (magnetometer_y_temp_top >> 8);
    magnetometer_y_temp_bottom = (value_magnetometer_y & bottom_mask);
    magnetometer_y_out_bottom = magnetometer_y_temp_bottom;
    // puts(I);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_y_out_top);
    // puts(LF);
    // puts(magnetometer_y_out_bottom);
void magnetometer_z()
{
    output_low(PIN_C1);  // Reset-low
    delay_us(2);
    output_high(PIN_C0);  // Set-high
    set_adc_channel(0);   // Chooses AN0 (PIN_RA0) as input
    delay_us(10);
    value_magnetometer_z = read_adc();

    magnetometer_z_temp_top  = (value_magnetometer_z & top_mask);
    magnetometer_z_out_top   = (magnetometer_z_temp_top >> 8);
    magnetometer_z_temp_bottom = (value_magnetometer_z & bottom_mask);
    magnetometer_z_out_bottom = magnetometer_z_temp_bottom;

    // puts(J);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_z_out_top);
    // puts(LF);
    // puts(magnetometer_z_out_bottom);
    // puts(LF);

    output_low(PIN_C0);  // Set-low
    delay_us(2);
    output_high(PIN_C1);  // Reset-high
    set_adc_channel(0);   // Chooses AN0 (PIN_RA0) as input
    delay_us(10);
    value_magnetometer_z = read_adc();

    magnetometer_z_temp_top  = (value_magnetometer_z & top_mask);
    magnetometer_z_out_top   = (magnetometer_z_temp_top >> 8);
    magnetometer_z_temp_bottom = (value_magnetometer_z & bottom_mask);
    magnetometer_z_out_bottom = magnetometer_z_temp_bottom;

    // puts(J);
    // puts(LF);
    // puts(ok);
    // puts(LF);
    // puts(magnetometer_z_out_top);
    // puts(LF);
    // puts(magnetometer_z_out_bottom);
    // puts(LF);
# Appendix 13.7.3 – Program Code Attitude Sensor

```c

// --- Initial Configuration ---
void main(){
    // All I/O-PINS are digital
    setup_adc_ports(NO_ANALOGS);
    // Sets the time for each analog sample?
    setup_adc(ADC_CLOCK_DIV_4);
    output_low(PIN_C0); // Set to low
    output_high(PIN_C1); // Reset to high

    for (p=0;p<12;p++){
        data[p] = 0;
    }

    while(TRUE){
        // Tempsensor ---
        i2c_start(); // Initializes the Temperature sensor to make one conversion
        i2c_write(0x92);
        i2c_write(0x01);
        i2c_write(0xC4);
        i2c_stop();
        delay_us(25);
        i2c_start(); // Reads the information from the temperature conversion register
        i2c_write(0x92);
        i2c_write(0x00);
        i2c_start();
        i2c_write(0x93);
        data1 = i2c_read();
        data2 = i2c_read();
        i2c_stop();
        temporary1 = data1;
        temporary1 = (temporary1 << 2);
        temporary2 = (data2 & 0xC0);//take 2 MSB
        temporary2 = (temporary2 >> 6);
        card_temperature = (temporary1 + temporary2);
        sign = (card_temperature & sign_mask);
        sign = (sign >> 9);
        tempvalue = (card_temperature & tempvalue_mask);
        if (sign == 1){
            // Do something
        }
```

\begin{verbatim}
abstemp = ((tempvalue-512)/4);
}
else if (sign == 0){
    abstemp = (tempvalue/4);
}

// Heater starts if temperature is lower than -10 degrees Celsius.
if (abstemp < -10){
    output_high(PIN_C2); // Sets PIN_RC2 high - starting the heaters
}
else if (abstemp >= -10){
    output_low(PIN_C2); // Sets PIN_RC2 low - killing the heaters
}

// -- ADC --
i2c_start(); // Initializes the AD to read the first 6 channels in a sequence
i2c_write(0x42);
i2c_write(0x02);
i2c_write(0x03);
i2c_write(0xF0);
i2c_stop();
delay_us(5); // ???
i2c_start(); // Initializes the AD to be in command mode
i2c_write(0x42);
i2c_write(0x70);
i2c_start(); // Reads the information from the conversion register
i2c_write(0x43); // and saves it into the array data[].
for (p = 0; p < 12; ++p){
    (data[p]) = i2c_read();
}
i2c_stop();

// Puts the read databytes into one single long. (2 LSB in temporary2 SHOULD be 0. MSB in
// temporary2 is an alert flag (?)).
// Bit 13-15 contains information regarding which channel the data comes from, bit 3-12 contains
// the actual value.
// Assuming that channel 1-2 is X, Channel 3-4 is Y and Channel 5-6 is Z...
for (p = 0; p < 6; ++p){
    temporary1 = (data[(p*2)] & 0x0F); //Take only 4LSB!
    temporary1 = (temporary1 << 6); //Shift left 6 bits
    temporary2 = ((data[(p*2)+1]) & 0xFC); //Take only 6MSB!
    temporary2 = (temporary2 >> 2); //Shift out the two LSB!
    acc_value[p] = (temporary1 + temporary2);
\end{verbatim}
// Runs the complete set of functions in order to update the mainframe with the latest readings.

  accelerometer_x();
  accelerometer_y();
  accelerometer_z();
  // magnetometer_x();
  // magnetometer_y();
  // magnetometer_z();