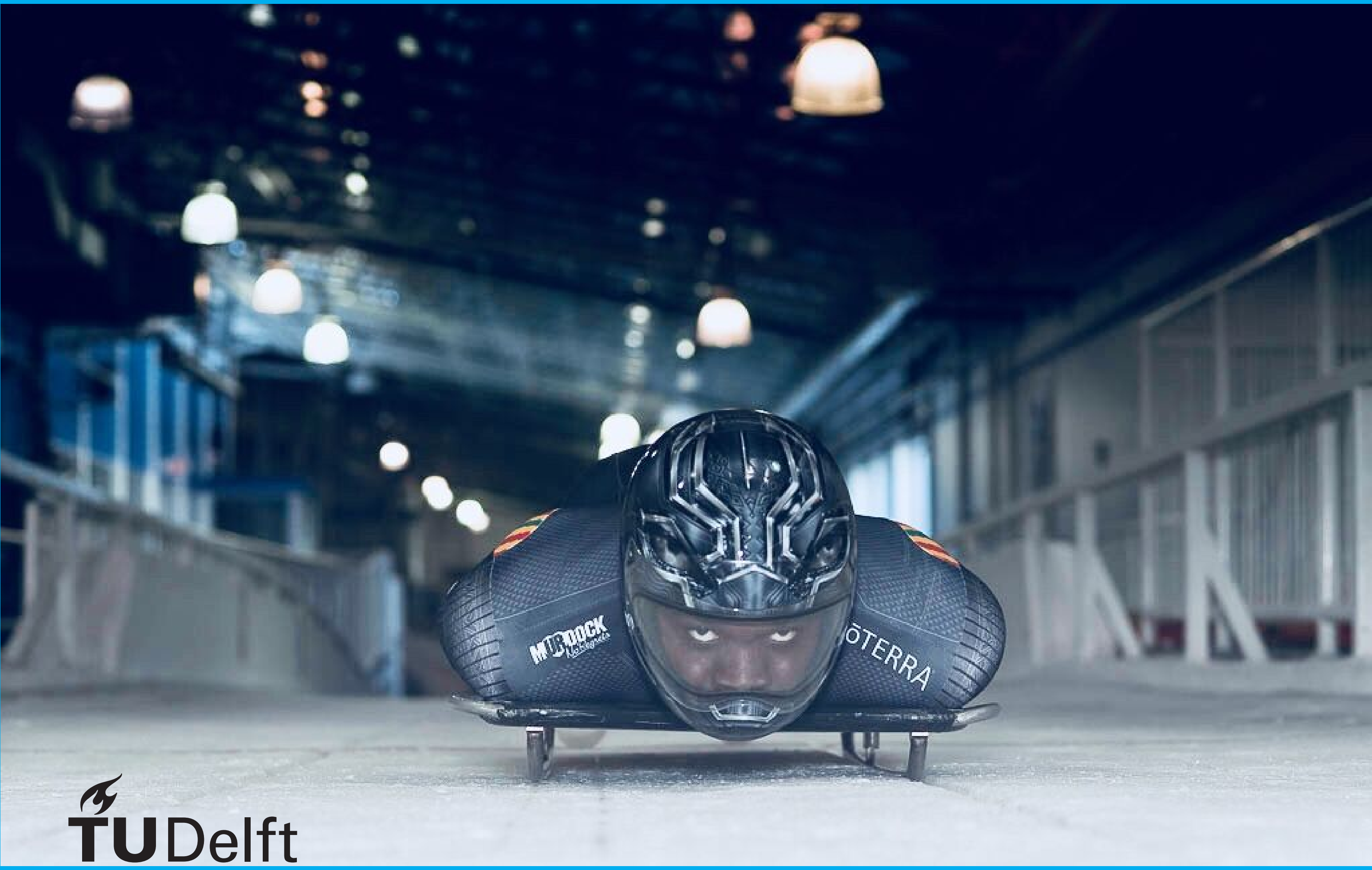
Instrumented Sled for Skeleton

Bachelor Thesis

W.M. van Dijk
K.N. van der Werff

Focusing on power management
and the sensors for localisation, velocity & temperature



TU Delft

Instrumented Sled for Skeleton

Bachelor Thesis

by

W.M. van Dijk
K.N. van der Werff

focusing on localisation, velocity, temperature & power management,
as part of the Bachelor Graduation Project,
at the Delft University of Technology.

Student number:	Werner van Dijk,	4464346
	Karen van der Werff,	4478657
Project duration:	April 22 – July 5, 2019	
Supervisors:	Prof. dr. P. J. French	
	Dr. ir. A. Bossche	

TU Delft

Abstract

There is only a very limited number of moments a skeleton athlete can train at a skeleton track. Therefore, it is important to train as efficiently as possible. To do so, an instrumented skeleton sled is designed that provides useful feedback. This instrumented sled is able to monitor the force exerted on the sled by the athlete and link this to the position and speed. Also, the ice temperature of the track as well as the G-forces acting on the sled are measured. This report covers the design and implementation of the temperature sensor, the power system, the PCB and the system to determine the location and velocity of the athlete which is necessary for the instrumented sled.

Preface

This thesis has been written in context of the Bachelor Graduation Project of the TU Delft. The project was commissioned by Akwasi Frimpong, a professional skeleton athlete that requested us to design a product to can provide him with information about his performance while practising his sport.

We would like to express our gratitude to our supervisors prof. dr. Paddy French, dr. ir. André Bossche and Ing. Jeroen Bastemeijer for their guidance during the project. Furthermore, we would like to thank Akwasi Frimpong for the topic and we wish him the best of luck on the Winter Olympics in Beijing in 2022. Finally, we would like to thank our colleagues: Martijn Heller, William Hunter, Tijs Moree and Jan de Jong, for an enjoyable and productive collaboration.

Werner van Dijk & Karen van der Werff
Delft, June 2019

Contents

1 Introduction

Skeleton is a winter sport in which an athlete lies in a prone position (face down and head first) on a small sled with two metal runners underneath, and goes down a winding track which is approximately 1800 meters long and covered in ice. An example of such a track can be seen in Fig. 1.1, which depicts a computer mock-up of the track that will be used at the 2022 Winter Olympics in Beijing. The sport has a high intensity: during a run, the athlete is subject to high G-forces and speeds which can exceed 130 km/h [1]. The steering of the sled is done by pushing the shoulders or knees into the sled: “the sled contorts as a response to the athlete’s steering control movements. When this happens, the left or right runner knife is forced into the ice, creating an asymmetry in ice friction resulting in a steering moment. That is, when the left runner is forced into the ice, the sled will turn left. Athletes thus use their shoulders and knees to contort the sled; for a more dramatic steering movement, they ‘tap a toe’ onto the ice, creating a larger steering moment” [2].

A run starts with the athlete sprinting from the starting point with the sled, the so-called push start, a critical part of the run (this feature also appears in bobsled racing, but is absent from the similar sport of luge). This can be seen schematically in Fig. 1.2: after about fifteen to thirty metres, the athlete mounts the sled at full running speed and manoeuvres it around a series of (often) high-banking corners in the desired path, to maximise his speed [2].

This project focuses on creating an instrumented skeleton sled. In the following sections, the goal of the project will be laid out, followed by the problem definition and an elaboration on the structure of this thesis.

1.1. The goal of the project

Akwasi Frimpong is a Dutch skeleton athlete of Ghanaian descent. His aim is to become the first athlete of African descent to win a golden medal at the Winter Olympics in 2022. In order to have a realistic chance of winning that medal, he must train harder, better and faster than his fellow skeleton colleagues. One of the ways to speed up the learning curve, is through improved feedback. Knowledge on the effects of certain movements will give valuable insights and pave the way to a more targeted training method. Thus, the goal of this project is to make an instrumented sled that is able to track the movements and present the data in a way that is useful for Akwasi. This way, he can practice skeleton in a smarter way than his competitors - a smarter way with a smarter sled.

1.2. Problem definition

The current training methods in skeleton are not yet very advanced. At the moment, only video imagery and visual feedback (such as photos) from the coach are used as feedback. Because of the high speeds at which skeleton athletes can go down the track, the important details from the run can be very hard to spot. Besides the previously mentioned means of feedback, there are no quantitative elements that can be measured, except for the timing measured by the timing eyes on the track. The important thing to know is how the athlete influences the sled during the run, especially in the curves. After all, this is

Figure 1.1: A computer mock-up of the skeleton track to be used at the 2022 Beijing Winter Olympics [3]

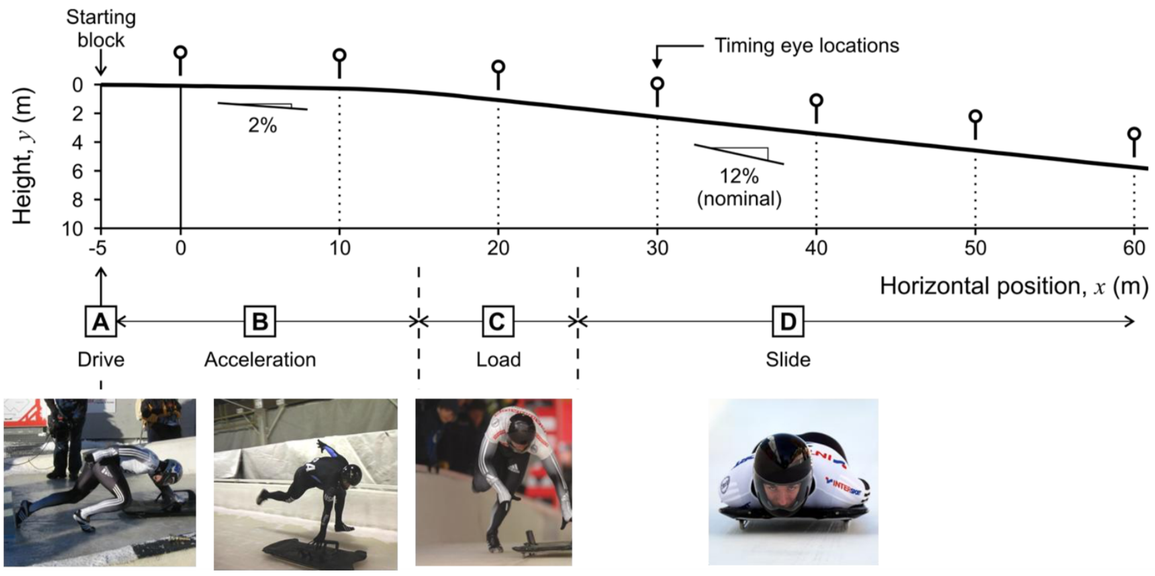


Figure 1.2: An overview of the height profile at the beginning of a skeleton practice track, illustrating the push start [1]

the factor with the highest impact on the run time besides the push start [4]. The steering of the sled is done by exerting forces on it using the shoulders and the knees. These movements of the shoulders and knees are practically impossible to see on video. An example of this can be found in figure 1.3, a photo that has genuinely been used for feedback. The solution to this problem is an instrumented sled which can measure the forces applied by the athlete, coupling the measured forces to the location of the sled on the track. The use of an instrumented sled would result in a significant increase in useful feedback that Akwasi can use to improve his run time.

Figure 1.3: Example of a visual feedback method that is currently being used.

1.3. State of the art skeleton instrumentation

Despite the fact that the sport of skeleton is not very widely practised, a number of studies have been done on the measurement of the forces, speeds and acceleration involved in the sport. These studies were primarily performed in order to create a better understanding of the dynamics at play in the sport, instead of having a goal of being used to actively improve athlete performance. Roberts [1] showed in his work that from measurements of the acceleration of the sled in three axes, velocity and traversed distance can be derived, which can provide useful information about the push start of the run. This is valuable, as it has been shown that the (effectiveness of the) push start has a large impact on the eventual time taken to complete a run [5]. After the push start and later in the run, however, the noise on these measurements becomes too large due to vibrations and other factors [1], making it impossible to integrate this to obtain a meaningful speed and distance reading during the whole descent. Sawade *et al.* [2] studied the factors influencing skeleton steering, showing a correlation between the applied steering force by the athlete and measured accelerometer and gyroscope data.

The aforementioned studies examined a number of relevant parameters involved in a skeleton run by attaching sensors to the sled and logging their output data. Although this is useful data, these studies provide only a limited tangent to the solution to the problem put forth in section 1.2, as further processing of the data and visualisation (to produce the graphs from which conclusions could be drawn) was done after the fact. No special consideration was given to making the data easily and quickly available to the athlete (and his coach) from which the measurements were sourced, relegating their results to the status of reference work for an athlete instead of a product they can use themselves. The general idea of combining computing, sensing and communication as is required for this project, however, isn't new. This way of integrating information processing into user objects without the user being actively aware of the hardware behind it is known as "ubiquitous" or "pervasive" computing and has seen a rapid increase in the past decade [6]. In sports in general, "these ubiquitous computing technologies are utilised to acquire, analyse and present performance data without affecting the athletes during training and competitions" [6].

Nevertheless, these technologies aren't yet prevalent in the skeleton world; the closest similar system was studied by Lee *et al.* for use on a bobsled. This involved fitting an elaborate system of sensors and cameras on the sled, that together produced video imagery overlaid with sensor data, which was then wirelessly transmitted to a monitor on a remote site in real time [7]. This is useful functionality, but it isn't practically usable in the skeleton case: the system is bulky, as can be seen in Fig. 1.4, requiring (amongst other things) a relatively large and heavy control unit, which wouldn't fit on a skeleton sled.

It can thus be concluded that although modern research into the topic of skeleton dynamics is avail-

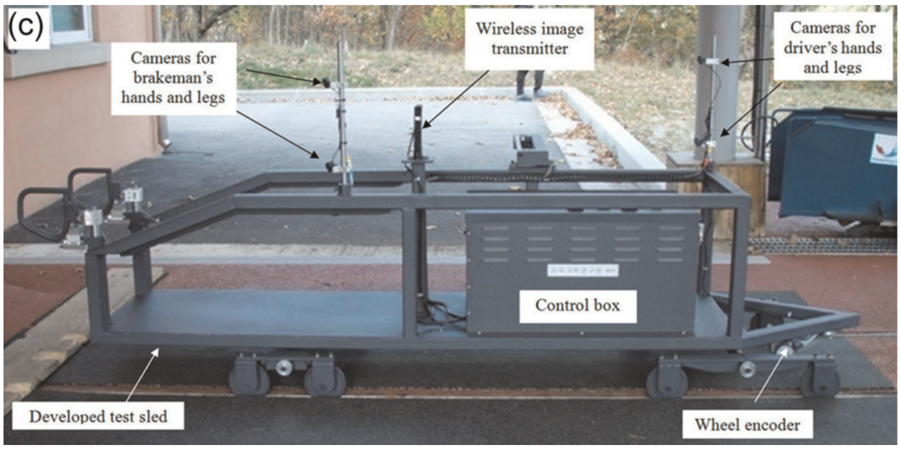

(c)
Cameras for brakeman's hands and legs
Wireless image transmitter
Cameras for driver's hands and legs
Control box
Developed test sled
Wheel encoder

Figure 1.4: An example of an instrumented bobsled, as developed by Lee *et al.* [7].

able, none exists that cover the scope of the system required to provide skeleton run data accurately and quickly in an (from the athlete's point of view) easy-to-understand format.

1.4. Subdivision of the system

In order to realise the instrumented sled, the project has been divided into 3 subgroups, each with its own responsibilities. This division can be seen in figure 1.5. The Data Group is responsible for the storage and the visualisation of the data at the end of each run. Sensor Group A is responsible for the measurements of the forces applied by the athlete, as well as the measurements with respect to the *g*-forces and the orientation of the sled. Sensor Group B is responsible for the localisation of the sled, measuring the ice temperature and designing the power management system. Furthermore, this subgroup focused on the integration of the system, connecting all subsystems on a PCB. This thesis focuses on the work of Sensor Group B.

1.5. Structure of the thesis

The thesis covers the choices made in the process of designing the subsystem for the instrumented sled, as well as the results. First, the programme of requirements will be explained in chapter 2. After that, chapters 3 to 6 will discuss each aspect of the subsystem more in depth. The design considerations as well as the implementation for the temperature measurements, the localisation system, the power management system and the PCB design will be discussed respectively. Then, the results will be presented and discussed in chapter 7. Lastly, a conclusion will be drawn in chapter 8, followed by recommendations for future work.

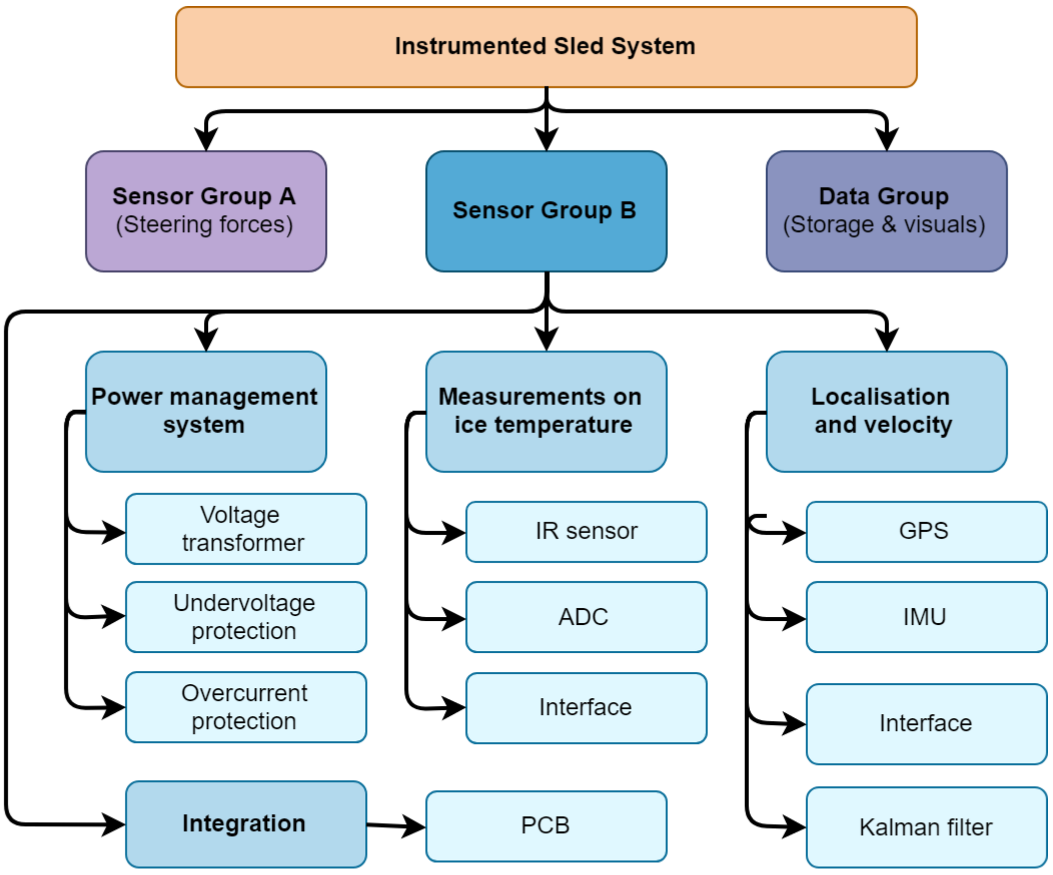


Figure 1.5: The division of tasks for the project, focusing on Sensor Group B.

2 Program of Requirements

As has been described in chapter 1, this project is about creating an instrumented skeleton sled to enable the skeleton athlete to train faster and smarter. In order to achieve this, a couple of aspects must be monitored, such as the location, the velocity and the applied force. These values must be stored and visualised for the athlete and his coach. This system is subject to various requirements, which are listed below. The final product will be tested against these requirements.

2.1. General requirements

General requirements are those requirements that are relevant for the entire system and that should be met by every subgroup. They are listed as follows:

G.1 The product must be able to measure G-forces, rotation, force applied by the athlete, ice temperature of the track and must be able to determine the location of the skeleton sled.

G.2 The product must be able to work in a temperature range from -20 °C to 40 °C, since it will be used in an environment with temperatures in this range.

G.3 The product must be able to withstand momentary accelerations of up to 5 g [4, p. 198].

G.4 The complete system should not weigh more than 1.5 kg, to prevent that the characteristics of the skeleton sled are different from match conditions during training.

G.5 The dimensions of the product cannot exceed dimensions of 31.5 × 14.7 × 2 cm, since this is the size of the available box inside the skeleton sled.

G.6 The update rate of the force sensors and localisation system should be such that data points are at most 1 metre apart. Working with a maximum speed of 147 km/h [8], this gives a minimum frequency of 41 Hz.

G.7 It should not be necessary to open the space inside the skeleton sled, where the circuitry will be located, in between runs. Therefore the user must be able to start and stop the measurement from the outside.

G.8 The product must be able to be easily installed or removed from the sled, without leaving any (permanent) traces on the skeleton sled.

G.9 The product should influence neither the aerodynamic properties nor the mechanical properties, apart from the weight, of the skeleton sled.

G.10 The product can not have any wired connections outside the skeleton sled and must be able to operate for the time it takes to do 3 runs and the time in between runs.

G.11 The system must be robust, being able to handle the vibrations of the skeleton sled during a run.

G.12 The acquired data must be available within 5 minutes after each run for the athlete and its coach.

G.13 The product should be easy to use.

G.14 The total cost of making the prototype must fit in the budget of €250,-.

2.2. Specific requirements

There are several requirements that apply solely to the subsystems described in this thesis. For Sensor Group B, the following requirements must be met:

S.1 The power system should have the capacity to last for at least three hours at temperatures down to -20 °C, enabling the athlete to complete three consecutive runs.

S.2 The power system should have an undervoltage protection to prevent damage and capacity loss of the battery.

S.3 The power system should have an overcurrent protection to prevent damage and increase the safety.

S.4 The battery should have a hard case to protect it from a direct hit or puncture, to prevent unsafe use.

S.5 The battery should be rechargeable.

S.6 The system should be able to deliver voltages of 3 V, 3.3 V and 5 V.

S.7 The temperature sensor can measure temperatures as low as -20 °C.

S.8 The temperature sensor should be able to obtain the temperature without physically touching the ice.

S.9 The temperature sensor must obtain readings with an accuracy of 1 °C.

S.10 The accuracy of the location measurements should be at least 1 metre.

S.11 The PCB should be able to connect all components in a reliable way.

3 Measuring Ice Temperatures

The skeleton sport knows three phases: the sprint phase, followed by the loading phase and finishing off with the sliding phase. The interactions between the ice and the runners of the sled, mainly friction, have a large impact on the performance of the run. It has been estimated that about 40% of the energy dissipated in the sliding phase is transmitted into the ice, and the effects of this energy transfer are temperature-dependent [9]. Research has shown that changes in the ice temperature have a large impact on the sliding velocity, even more so than the weight of the sled and roughness of the metal bars. The relationship between the temperature and the velocity can be seen in figure 3.1 [10]. Therefore, temperature is an important factor that is beneficial to include in the feedback system for Akwasi.

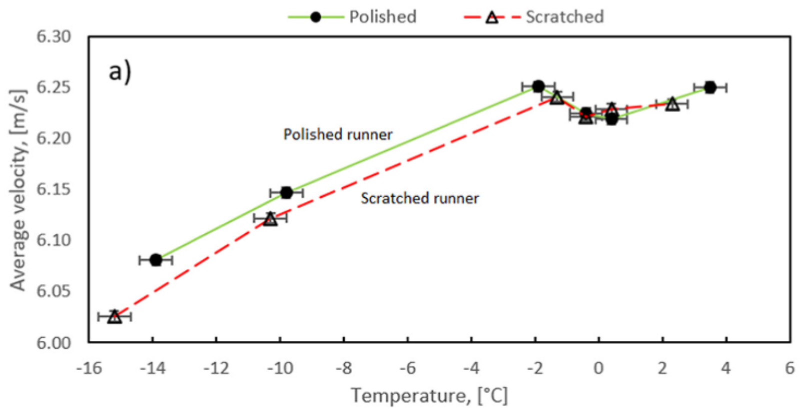


Figure 3.1: Relationship between the temperature of the ice and the velocity of the sled. [10]

3.1. Infrared sensors

There is a large variety of thermometers that can be found on the market. While they all have one thing in common - correlating a physical, thermometric property to a temperature - not all of them are suitable for this application. As the temperature measurement of the ice should be done without affecting the athlete's performance, it should also be done without physically touching the ice. A logical choice would thus be one of the following non-contact sensors: fluoroptic sensors, interferometric sensors, fiber-optic temperature sensors or infrared sensors [11].

While each sensor has their benefits and disadvantages [12] [13] [11], the infrared (IR) temperature sensor seems to be the best option for this system. This is because IR sensors are able to operate in the freezing cold environment of a skeleton track, indicating temperatures with high accuracy. Furthermore, they are lightweight and can be found in small sizes, while still being sold at an affordable price.

3.1.1. The principles of an IR sensor

One of the ways that heat transfer takes place, is through thermal radiation. This 'heat' radiation is an electromagnetic wave with a wavelength ranging from 1 up to 100 micrometers [14]. An object can radiate and absorb electromagnetic waves. A *blackbody* is a theoretical, ideal body that absorbs all radiation incident on it, while it can also emit radiation at all wavelengths. The spectrum of this radiation is determined solely by the temperature; the shape or material of the body have no influence. The characteristics of this radiation can be described with a number of laws [15], of which Planck's law is the overarching one. However, this law is very abstract, so practical applications usually make use of the derived *Stefan-Boltzmann law* [16].

The Stefan-Boltzmann law relates the absolute temperature T to the radiated energy per unit area E, through a proportionality constant σ. While this theory has been devised for blackbodies, it can be used for real, physical bodies as well. Physical bodies emit energy at a portion of the blackbody energy, this portion being determined by the object's emissivity [17]. Incorporating the emissivity, ϵ into the Stefan-Boltzmann relation, the following equation is obtained:

$$E = \sigma \cdot \epsilon \cdot T^4 \tag{3.1}$$

From equation 3.1, it follows that the emitted energy density of an object increases as the temperature increases. Using this fact, the temperature of an object can be calculated when the energy density and the emissivity of a material are known. The emissivity of a blackbody is 1, while real bodies have an emissivity of less than 1 [16]. The emissivity of ice is between 0.97 (smooth) and 0.98 (rough) [18].

3.1.2. Sensors available on the market

Equation 3.1 is indispensable when calculating the temperature of an object and should be kept in mind when choosing which sensor to use. The value for the emissivity used by the sensor must be adjustable to increase the accuracy of the readings. Table 3.1 gives an overview of IR sensors that have been found to be available and affordable, while being able to handle temperatures well below 0 °C. After considering these options, the *MLX90614* has been chosen due to its high accuracy and resolution. It is able to withstand temperatures even lower than -20 °C, fulfilling requirement G.2. Furthermore, it can measure temperatures in the right range, in a contactless manner with an accuracy of 0.5 °C, thus fulfilling requirements S.7, S.8 and S.9 respectively.

Sensor	Temperature range	Supply voltage	Resolution	Max. error
TMP007 [19]	-40 to 125 °C	2.2 - 5.5 V	0.03125 °C	±5 °C
MLX90614 [20]	-70 to 380 °C	3.3 V or 5 V	0.02 °C	±0.5 °C
ZTP-135SR [21]	-20 to 100 °C	*not listed*	*not listed*	*not listed*
Phidget IR [22]	-70 to 380 °C	4.8 - 5.3 V	0.02 °C	±4 °C

Table 3.1: An overview of possible IR sensors.

3.2. Implementation of the sensor

As stated above, the *MLX90614* has been chosen. More specifically, the 3.3 V variant *MLX90614ESF-BAA* has been used, as it complies with the maximum voltage rating of 3.6 V of the microcontroller (*ESP32*). The following sections will elaborate on the implementation of this sensor. The *MLX90614ESF-BAA* will be abbreviated to 'MLX' in the rest of the text.

3.2.1. Setting up the MLX

Since the sensor already contains an amplifier, a 17-bit ADC and a DSP unit, no further circuitry needs to be designed in order to obtain the sensor readings. In fact, only a power supply decoupling capacitor is necessary in order to keep the noise low. A 100 nF capacitor must be placed in between V_{dd} and V_{ss}, which is shown in figure 3.2 [20]. Multiple sensors will be integrated through the ESP32, thus adding extra pull-up resistors is not necessary here - the microcontroller contains internal pull-ups already.

The MLX will communicate with the *ESP32* through *SMBus*, a two-wire bus derived from I^2C. SMBus can operate at frequencies up to 100kHz, however sampling at 50 Hz will be enough for this application.

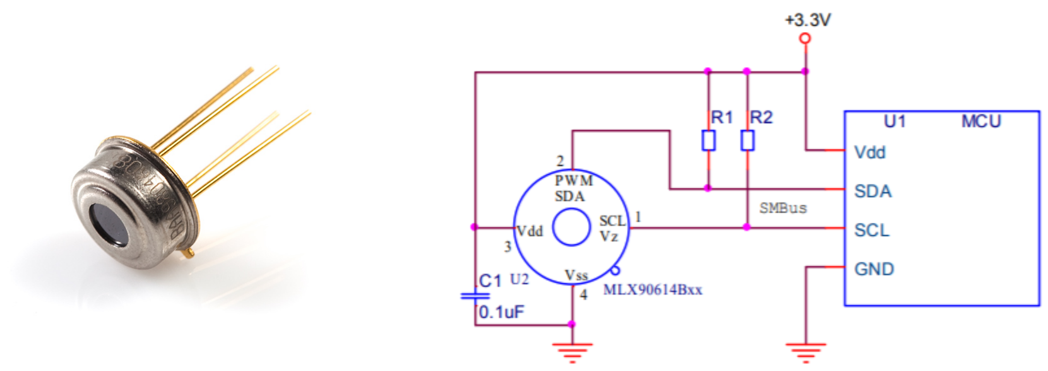


Figure 3.2: Left: the *MLX90614*. Right: Schematic of the connections. [20]

Obtaining the sensor readings will be done using the *Adafruit-MLX90614* library, a basic library that outputs temperature in either degrees Celsius or Fahrenheit and that is compatible with the *ESP32*.

3.2.2. Changing the emissivity

While the factory setting of the MLX's emissivity is set to 1, an emissivity of either 0.97 or 0.98 is necessary for this application, as mentioned in section 3.1.1. Thus, some reprogramming had to take place. Nevertheless, Sparkfun's *Change emissivity* sketch proved to be incompatible with the *ESP32*, while other *ESP32*-compatible MLX libraries did not include such functionality. Therefore, a custom code was written using [23] while taking inspiration from the *Sparkfun MLX90614* library.

The MLX stores the emissivity value in EEPROM at register addres $0x04$. These values must be stored in hexadecimal and are obtained using the following formula:

$$0x04_d = 2^{16} \cdot \epsilon - 1 = 65536 \cdot \epsilon - 1 \tag{3.2}$$

Using equation 3.2 for 0.97 yields $0x04_d$ = 63568.92. Rounding it off and converting decimal to hexadecimal gives the value $0x04_h$ = $0x$F851. In a similar fashion for 0.98, $0x04_d$ equals 64224.28. Rounding it off and converting decimal to hexadecimal yields $0x04_h$ = $0x$FAE0. To set one of these values, the register needs to be erased first by writing $0x0000$ to it. Then, the new value $0x$F851 or $0x$FAE0 can be written. Afterwards, the register value can be read by sending a repeated start, to ensure that the correct value has been set.

Furthermore, a Packet Error Code (PEC) must be sent after each read or write to check if the correct message has been received. This PEC is calculated through a 8-bit Cyclic Redundancy Check (CRC-8), in which the message is XOR'd with the polynomial $c(x) = x^8 + x^2 + x^1 + 1$ [24]. A message on which the CRC-8 is performed has the following structure:

[Slave Address (Read/Write), Command, LSB, MSB]

The default address of the sensor is $0x$5A or $0b01011010$. In write commands, the address is followed by a '1', resulting in $0b010110101$ or $0x$B5 in the case of 'Slave Address Write'. Furthermore, the Command is $0b00100100$, [23] or $0x24$. Lastly, the LSB and MSB are taken from the hexadecimal emissivity values. Thus, the following PECs have been calculated:

- **Erase 0x04**: the message equals $[0xB5, 0x24, 0x00, 0x00]$, the PEC equals $0x3E$.
- $\boldsymbol{\epsilon = 0.97}$: the message equals $[0xB5, 0x24, 0x51, 0xF8]$, the PEC equals $0xC1$.
- $\boldsymbol{\epsilon = 0.98}$: the message equals $[0xB5, 0x24, 0xE0, 0xFA]$, the PEC equals $0x95$.

These steps have been incorporated into the code for the *ESP32* and can be found in appendix A. The measured effects of changing the emissivity can be found in section 7.1.

4 Measuring Location and Velocity

Akwasi Frimpong mainly trains on the the skeleton track in Park City, Utah. This track is roughly 1.3 km long, with a total of 15 curves. These curves have varying dimensions, where radii of 25-30 metres are quite common [25]. In order for the feedback to be useful, Akwasi and his coach need to be able to see where on the track certain quantities have been measured. Especially in the curves, this information is useful. Hence, requirement G.6 has been set to obtain enough data points, with requirement S.9 to obtain enough accuracy. The measurements of the location and velocity will be done with two different devices: a GNSS receiver and an Inertial Measurement Unit, which are described in section 4.1. The output of these devices will be fused to create a more accurate system, as described in section 4.2

4.1. Tracking sensors

This section describes the Inertial Measurement Unit and the GNSS receiver:

Inertial Measurement Unit

The Inertial Measurement Unit (IMU) used in the sled system is a *MPU-9250*, as ordered from Sparkfun. This is a 9-DoF unit, which contains an accelerometer, a magnetometer and a gyroscope. The considerations made when choosing the sensor can be found in [26], as well as a more elaborate description of its functionality. While Sensor Group A is home to the g-force measurements, this sensor has been included in this thesis as well because of the sensor fusion as described in the next section. For obtaining the measurements from the IMU, the Sparkfun library *MPU9250-9-DoF-IMU-Breakout* has been used. Besides outputting the results of the accelerometer, magnetometer and gyroscope, this library also has the functionality to compute the yaw, pitch and roll angles.

GNSS receiver

GNSS stands for Global Navigation Satellite System. GNSS receivers use satellites to determine the latitude, longitude and altitude of their current position. By looking at the difference between time signals received from different satellites, their position can be calculated. For the sled system, the update rate of the GNSS receiver is important - it should be as high as possible. Furthermore, it should be able to communicate with the *ESP32*, either through UART, SPI or I^2C.

In table 4.1, different GNSS receivers are presented that could possibly be used in the sled system. Each GNSS receiver can withstand temperatures as low as -40 °C. As can be seen, the *SAM-M8Q* from Sparkfun has the highest possible update rate. The Time-To-First-Fix (TTFF) is a little high, yet it does not pose a problem for this application. After all, the system can already be turned on in advance, allowing the GNSS receiver to find its first (cold) fix. Besides that, in case of a hot start the TTFF is 1 second, quick enough to obtain a fix before the athlete goes down the track. Thus, the SAM-M8Q has been found to be the most suitable GNSS receiver for the sled system. In the rest of the text, the receiver will simply be referred to as 'SAM'.

GPS	Max. update rate	TTFF	Horizontal accuracy
SAM-M8Q [27]	18 Hz	26 - 30 s	2.5 m
Quectel L76-M33 [28]	10 Hz	<15 s	< 2.5 m
Adafruit Ultimate [29]	10 Hz	34 s	<3 m
NEO-M8P [30]	10 Hz	26-29 s	2.5 m

Table 4.1: An overview of possible GNSS receivers.

The SAM is a 72-channel GNSS receiver, that can receive and track GPS, Galileo and GLONASS. By default, GPS and GLONASS are enabled simultaneously, increasing the coverage and reliability. However, in this case the maximum update rate is 10 Hz. When only the GPS is enabled, the maximum update rate reaches 18 Hz [27]. Since skeleton athletes can move at high speeds, the update rate should be as high as possible. Thus, for this application GLONASS has been disabled. This has been done by programming the GNSS receiver using the UBX protocol as specified in [31]. The structure and content of this message can be found in appendix B.1, the implementation of the message in code can be found in appendix B.2. This code only needs to be run once, as the configuration is stored even after power off. This is achieved by the tiny battery that has been included on the break-out board, that maintains a back-up voltage V_{bckp}. Further communication with the SAM has been done using the Sparkfun *Ublox-Arduino-Library*, in which a large variety of functions have been programmed. Readings on latitude, longitude, altitude are easily obtained.

4.2. Sensor fusion

Both the SAM and the IMU in section 4.1 have their benefits and disadvantages when using either sensor to obtain information on location and velocity. While the GNSS receiver is able to provide absolute values for the location, the update rate is relatively slow for the purpose of an instrumented sled. At speeds of up to 140 km/h and a frequency of 18 Hz from the GNSS receiver [27], the distances between each data point at maximum speed can be calculated as follows:

$$\frac{1}{f} \cdot v\,[m/s] = \frac{1}{18} \cdot \frac{140}{3.6} = 2.16\,m \tag{4.1}$$

Noting that the accuracy of the SAM is 2.5 m [27] and taking into account the results of calculation 4.1, distances between consecutive data points can be up to 4.6 metres apart. This is considerably more than requirement G.6 dictates.

In contrast, the IMU has relatively high update rates. It can be configured to run at speeds in the order of kHz [32]. This can be used to create a much more detailed map of the velocity and acceleration at specific locations. Nevertheless, the IMU can only measure changes relative to a starting position. Hence, it can be subject to a large amount of drift as time passes by.

Therefore, a system will be implemented that combines both sensors. In this way, the issue of drift as well as that of low update rates can be countered. The data of the IMU can be corrected on a regular basis with the GNSS receiver data, while running at a higher frequency than the 18 Hz of the SAM.

A popular method for fusing GNSS receiver and IMU data, is through the use of a *Kalman filter*, hereafter abbreviated to KF. Such a filter is widely used in navigation applications, from urban navigation [33] to user tracking in augmented reality [34]. This filtering technique is able to take the measurement errors from the GNSS receiver and the IMU into account, in order to provide an optimal estimate of the state. This results in more accurate data obtained from the GNSS-IMU fusion. When comparing a KF to an alternative filter such as a *particle filter*, it has a relatively low computational cost as well [35]. Thus, a KF seems to be well-suited for the instrumented sled, where both computation time and memory space are limited. The following subsections will elaborate on the theory behind such a filter.

4.2.1. The Kalman filter

A KF is an algorithm that consists of two steps [36]:

- **Step 1: Prediction** In this step, an estimate is produced of the state variable $\boldsymbol{x}_k$ - the *a-priori* estimate. This is done using the information from the previous state estimate, $\boldsymbol{x}_{k-1}$, as well as the current sensor input, $\boldsymbol{u}_k$.

- **Step 2: Update** Using the a-priori estimate and the sensor measurement, the *a-posteriori* estimate is calculated.

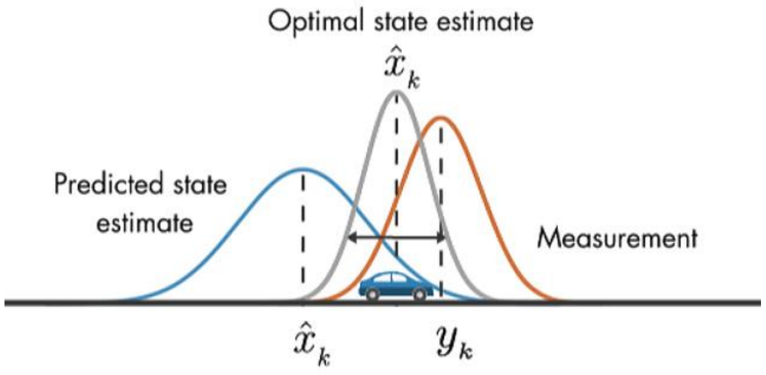


Figure 4.1: Visualisation of a Kalman filter [37]

This is visualised in figure 4.1. Here, the blue curve shows the probability distribution of the location of the car using the a-priori estimate from the prediction phase. The orange curve shows the probability distribution of the measured location. Using these two distributions, the optimal state estimate is obtained. As can be seen, it has a higher and a more narrow curve, implying that this is the most probable location for the car to be located. Thus, through the fusion of the data, a more accurate output is obtained.

4.2.2. A motion model

First, we will take a look at an *observer*. An observer is a filter that approximates the state vector of a real dynamical system, using measurements of the input and output of that system. Imagine the LTI state-space model [38]:

$$\boldsymbol{x}(k) = \boldsymbol{A}\boldsymbol{x}(k-1) + \boldsymbol{B}\boldsymbol{u}(k) \tag{4.2a}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \tag{4.2b}$$

In the upper equation, **x**(k) is the state vector, **A** is the state transition matrix, **B** is the control matrix and **u**(k) is the control vector. In the lower equation, **y**(k) is the measurement vector and **C** is the observation matrix [39]. **D** is the feedforward matrix, which incorporates the direct influence of the input on the output. As the sled system will measure position and velocity indirectly, this matrix is considered to be a null-matrix.

On the other hand, matrices **A**, **B** and **C** still need to be determined. These will be defined using the following equations of motion:

$$\boldsymbol{p_2} = \boldsymbol{p_1} + \boldsymbol{v_1} \cdot t + \frac{1}{2} \cdot \boldsymbol{a} \cdot t^2 \tag{4.3a}$$

$$\boldsymbol{v_2} = \boldsymbol{v_1} + \boldsymbol{a} \cdot t \tag{4.3b}$$

In discrete form, these equations can be written as:

$$\boldsymbol{p}(k) = \boldsymbol{p}(k-1) + \boldsymbol{v}(k-1) \cdot Ts + \frac{1}{2} \cdot a(k-1) \cdot Ts^2 \tag{4.4a}$$

$$\boldsymbol{v}(k) = \boldsymbol{v}(k-1) + a(k-1) \cdot Ts \tag{4.4b}$$

Where Ts is the sampling period and k the sample number. These equations can be captured in matrix form as follows:

$$\begin{bmatrix}\mathbf{p}_x(k)\\\mathbf{p}_y(k)\\\mathbf{p}_z(k)\\\mathbf{v}_x(k)\\\mathbf{v}_y(k)\\\mathbf{v}_z(k)\end{bmatrix}=\begin{bmatrix}1&0&0&Ts&0&0\\0&1&0&0&Ts&0\\0&0&1&0&0&Ts\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}\mathbf{p}_x(k-1)\\\mathbf{p}_y(k-1)\\\mathbf{p}_z(k-1)\\\mathbf{v}_x(k-1)\\\mathbf{v}_y(k-1)\\\mathbf{v}_z(k-1)\end{bmatrix}+\begin{bmatrix}\frac{Ts^2}{2}&0&0\\0&\frac{Ts^2}{2}&0\\0&0&\frac{Ts^2}{2}\\Ts&0&0\\0&Ts&0\\0&0&Ts\end{bmatrix}\begin{bmatrix}\mathbf{a}_x(k)\\\mathbf{a}_y(k)\\\mathbf{a}_z(k)\end{bmatrix} \tag{4.5}$$

Here, the vector on the left-hand side of the equal sign is the state variable **x**(k). The vectors on the right-had side of the equation denote the state variable **x**(k-1) and the acceleration **a**(k) respectively. In a system where an IMU provides the input, **a**(k) can be seen as the control vector **u**(k). Furthermore, 6-by-6 matrix represents the state transition matrix **A** and the 6-by-3 matrix represents the control matrix **B**. Consequently, equation 4.5 is an implementation of equation 4.2a. In a similar fashion, equation 4.2b can be implemented for the motion equations as:

$$\begin{bmatrix}\mathbf{p}_x(k)\\\mathbf{p}_y(k)\\\mathbf{p}_z(k)\end{bmatrix}=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\end{bmatrix}\begin{bmatrix}\mathbf{p}_x(k)\\\mathbf{p}_y(k)\\\mathbf{p}_z(k)\\\mathbf{v}_x(k)\\\mathbf{v}_y(k)\\\mathbf{v}_z(k)\end{bmatrix} \tag{4.6}$$

Here, the vector on the left-hand side represents the measurement vector **y**(k). These measurements can, for instance, be given by a GNSS receiver. The right-hand side resembles the observation matrix **C** multiplied by the state vector **x**(k).

4.2.3. The model according to Kalman

The state-space equations for the Kalman filter look similar to that of the observer as discussed in section 4.2.2, except that a KF also takes the noise into account. More precisely, it models the process noise **w**(k) and the measurement noise **v**(k). The process noise comes from mismatches between the modelled motion equations and the reality. The measurement noise comes from the sensor inputs. Equations 4.2 then become [38]:

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k)+\boldsymbol{w}(k) \tag{4.7a}$$
$$\boldsymbol{y}(k)=\boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{v}(k) \tag{4.7b}$$

Knowing **w**(k) and **v**(k) would enable one to obtain perfect values for a state **x**(k). However, the noise parameters are unknown - and this is where the Kalman filter shows its added value. Both **w**(k) and **v**(k) are assumed to have covariance matrix **Q** and **R** respectively. In mathematical form:

$$E[\boldsymbol{w}(k)\boldsymbol{w}(k)^T]=\boldsymbol{Q},\qquad E[\boldsymbol{v}(k)\boldsymbol{v}(k)^T]=\boldsymbol{R} \tag{4.8}$$

These matrices **Q** and **R** are known, as they can be obtained by looking at the standard deviations of the components of state variable **x**(k) and measurement variable **y**(k) respectively. Assuming that the variables are independent and uncorrelated, these matrices have the following structure:

$$\boldsymbol{Q}=\begin{bmatrix}\sigma^2(x_{p_x})&0&0&0&0&0\\0&\sigma^2(x_{p_y})&0&0&0&0\\0&0&\sigma^2(x_{p_z})&0&0&0\\0&0&0&\sigma^2(x_{v_x})&0&0\\0&0&0&0&\sigma^2(x_{v_y})&0\\0&0&0&0&0&\sigma^2(x_{v_z})\end{bmatrix}\qquad\boldsymbol{R}=\begin{bmatrix}\sigma^2(y_{p_x})&0&0\\0&\sigma^2(y_{p_y})&0\\0&0&\sigma^2(y_{p_z})\end{bmatrix} \tag{4.9}$$

Both matrices **Q** and **R** are called the tuning parameters of the filter. While measuring the standard deviation is a good starting point for the covariance matrices, they should be further adjusted to obtain the best performance [39].

4.2.4. The Kalman algorithm

Now that the theory behind the KF has been covered, the actual algorithm can be discussed. As mentioned in section 4.2.1, the algorithm can be subdivided into two phases: prediction and update. The prediction phase is described by equations 4.10, the update phase is described by equations 4.11. Here, the superscript '-' indicates an estimation made with the knowledge of the previous state, while the superscript '+' indicates an estimation made with the knowledge of the current state [39]. These equations will be executed in a loop for every sample in the dataset.

$$\text{Predicted state estimate} \quad \hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1}^+ + \boldsymbol{B}\boldsymbol{u}_{k-1} \tag{4.10a}$$

$$\text{Predicted error covariance} \quad \boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}^+\boldsymbol{A}^T + \boldsymbol{Q} \tag{4.10b}$$

$$\text{Measurement residual} \quad \tilde{\boldsymbol{e}}_k = \boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k^- \tag{4.11a}$$

$$\text{Meas. residual covariance} \quad \boldsymbol{S_k} = \boldsymbol{R} + \boldsymbol{C}\boldsymbol{P}_{\boldsymbol{k}}^-\boldsymbol{C}^T \tag{4.11b}$$

$$\text{Kalman gain} \quad \boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{C}^T\boldsymbol{S}^{-1} \tag{4.11c}$$

$$\text{Updated state estimate} \quad \hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\tilde{\boldsymbol{e}} \tag{4.11d}$$

$$\text{Updated error covariance} \quad \boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_k^- \tag{4.11e}$$

4.3. Implementation

For the instrumented sled, the KF will be implemented according to equations 4.5 and 4.6. The IMU will provide the 3D acceleration information as input **u**(k). The SAM will provide the 3D position information **y**(k). The state vector **x**(k) will track the position and velocity in three dimensions each. The matrices **A**, **B** and **C** will be according to the motion model. The tuning parameters **Q** and **R** will be obtained from taking the measured standard deviation, squaring these to obtain the variance and adjusting these until the KF performs best. The implementation of the KF is shown schematically in figure 4.2.

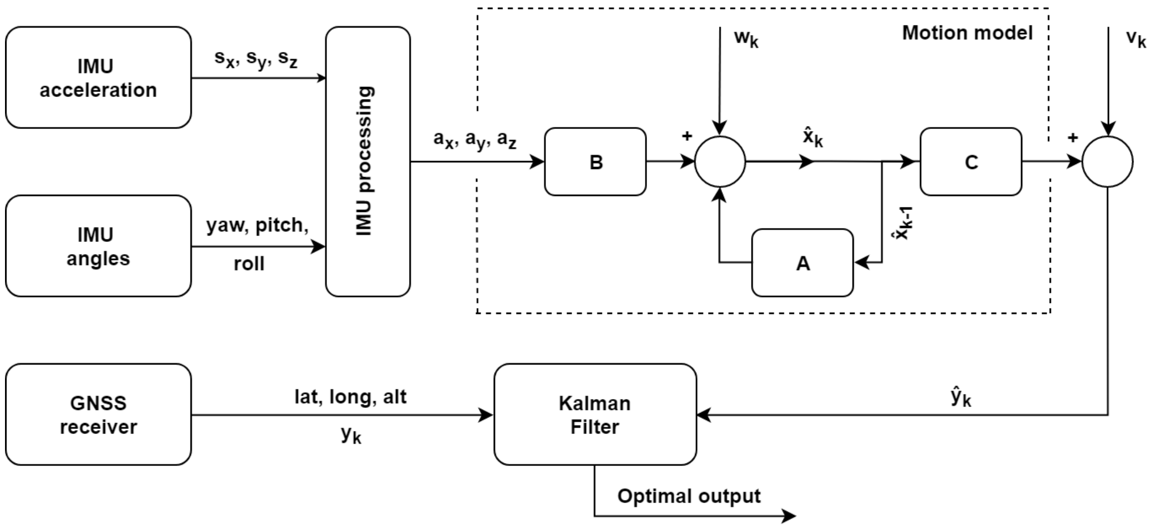


Figure 4.2: Schematic of the KF implementation.

There are various libraries to be found online that are able to implement a Kalman filter on the *ESP32*. These functions often need only two values: the tuning parameters **Q** and **R** as described in section 4.2.3. To obtain these tuning parameters, the KF has first been implemented in MATLAB, in which multiple simulations have been done. The code can be found in appendix C.1. After initializing the necessary variables, data is imported from a .txt file. The number of samples needs to be set for N, after which the Kalman algorithm will run for N times. For each iteration, the data from the IMU and the GNSS receiver are fused into one estimate $\hat{x}$.

However, the IMU outputs its data in a body coordinate frame (x, y, z), while the GNSS receiver works with latitude, longitude and altitude. These frames must be rearranged to be able to fuse the data. The GNSS data can easily be transformed into the navigation coordinate frame, (N, E, D), by using the MATLAB function 'geodetic2ned'. The IMU body frame must be rearranged to the navigation frame through a rotation matrix [40]. Such a matrix has also been implemented in MATLAB. In Euler angles, the rotation matrix $\mathbf{R}$ from the body frame *(z,y,x)* to the navigation frame *(N,E,D)* can be calculated as $\mathbf{R}_b^n(\boldsymbol{\theta}_{nb}) = \mathbf{R}_b^n(z,\psi)\cdot\mathbf{R}_b^n(y,\theta)\cdot\mathbf{R}_b^n(x,\phi)$. Here, ψ is the yaw, θ is the pitch and ϕ is the roll. The rotation is then performed as follows: first a rotation of angle ψ around the z-axis, followed by an angle θ around the y-axis, concluded by an angle ϕ around the x-axis. $\mathbf{R}_b^n$ can then be written as [41]:

$$\mathbf{R}_b^n(\boldsymbol{\theta}_{nb}) = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \tag{4.12}$$

Where $\boldsymbol{\theta}$ gives the attitude of the system in Euler angles: $\boldsymbol{\theta} = [\phi,\theta,\psi]^T$. While a quaternion representation is generally more numerically stable [40] than euler angles, Akwasi will not experience angles of more than 90 °on the track. Hence, the issue of *gimbal lock*, in which a degree of freedom is lost, will not be encountered here. Therefore, the use of euler angles can be justified.

In chapter 7, the validation of this system will be discussed.

5 Power Management and Battery System

In this chapter, the battery system used to power the instrumented sled will be discussed. The choice for a specific battery type will be addressed, as well as the required specifications and characteristics of the battery. Furthermore, the implementation of the battery will be discussed, which includes a protection circuit and other power electronics.

5.1. Requirements of the battery

The weight and the size of the battery are of utmost importance; the battery has to fit in a small space inside the skeleton sled and should not increase the weight of the total instrumented sled significantly, seeing that the characteristics of the skeleton sled should not be affected. Therefore, a characteristic that is important for this project is the *energy density* of the battery. Furthermore, the battery needs to be rechargeable and be able to supply energy for about 3 hours, even at low temperatures. As the battery will be placed close to the athlete, it needs to be safe in operation as well. Lastly, a battery management system is required to protect the battery and the connected system.

5.1.1. Energy density

As has been mentioned, the energy density is an important aspect, since a limit of 1.5 kg has been set for the total system (requirement G.4). However, while an upper limit has been set, a lower weight is beneficial in order to minimise the differences between the regular sled and the instrumented sled. Furthermore, the battery should be rather flat, so that it fits inside the sled (requirement G.5). In figure 5.1, the densities of common rechargeable batteries are shown. Three battery types have been listed with the highest densities:

- Li-Polymer batteries
- Cylindrical type Li-ion batteries
- Prismatic type Li-ion batteries

These batteries have a high *volumetric energy density*, as well as a high *gravimetric energy density*. Moreover, these batteries are rechargeable, which would satisfy requirement S.5. At the moment, batteries with an even higher energy density are available on the market, yet these are either not yet stable (safe) enough, not rechargeable, or simply too expensive. As Lithium-based batteries are widely used these days, the price has become relatively low compared to their potential successors [42]. Therefore, these three battery types would be suitable options for this project.

5.1.2. Safety

The safety and stability of the battery is another important aspect. G-forces of up to 5 g are no exception and a skeleton sled will heavily vibrate during a run [4]. Spontaneous fires in Lithium-based batteries have occurred before and should be taken seriously [44]. Li-Polymer batteries appear to be more safe and may be better-suited to these conditions [45]. However, they should be well protected by a hard case,

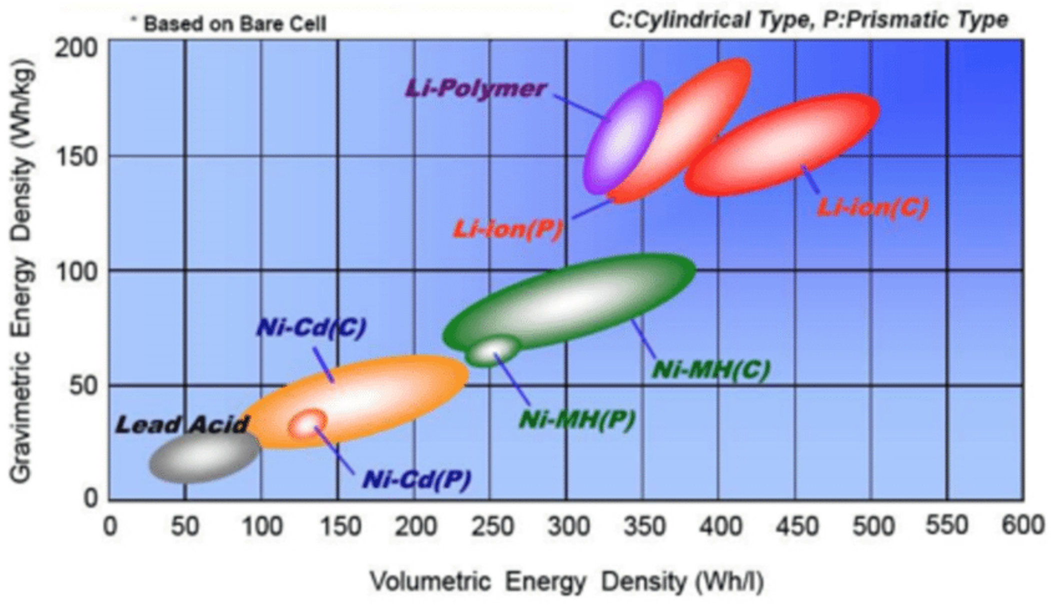


Figure 5.1: Comparison of Energy Density in Battery Cells [43].

as a direct hit or puncture could set the battery on fire (requirement S.4). Lithium-based batteries are also used in space applications where high g-forces are exerted on the batteries. This makes them suited for the high g-forces that can be experienced during a run on the skeleton track as well. Furthermore, Li-Polymer batteries are able to function in cold conditions as low as 20 °C below zero. However, it should be taken into account that the usable capacity decreases with temperature and that the batteries should be charged at temperatures above 0 °C, as will be discussed in section 5.1.3.
All things considered, a Li-Polymer battery will be used for this project.

5.1.3. Capacity

In order to determine what the minimum capacity of the battery should be, it has to be known how much energy is used by the connected components and devices. In table 5.1, the electrical specifications of the components that will be used for this project are given [27] [32] [46] [47] [48] [49]. For this project, the battery should be able to power the system for at least 3 hours, allowing the athlete to complete his training programme (requirement S.1 and G.10). To be able to supply a high enough voltage to the subsystems, a two-cell Li-Polymer battery will be used, with a nominal voltage of 7.4 V.

Table 5.1: Electrical specifications of the components

Device	Minimum Voltage [V]	Maximum Voltage [V]	Typical Voltage [V]	Maximum Current Consumption [mA]	Typical Current Consumption [mA]	Typical Power [mW]	Energy in 3 hours [mWh]
Esp32	1.8	3.6	3.3	240	100	330	990
GPS	2.7	3.6	3.0	67	32	96	288
IMU	2.4	3.6	2.5	3.7	3.7	9.3	27.8
Temperature sensor	2.6	3.6	3.0	2.0	1.0	3.0	9.0
Force sensors	3.0	5.0	3.3	15	10	33	99
ADC	2.7	5.5	3.3	0.4	0.3	1.0	3.0
Power electronics	6.3	15	7.4	18	17	126	377

From table 5.1, it can be calculated that the total dissipated energy in three hours will be 1766 mWh. For this calculation, the typical current consumption is used. As a two-cell Li-Polymer battery with a nominal voltage of 7.4 V will be used, 1766 mWh can be written as 239 mAh. It is recommended that the Li-Polymer battery should not be discharged below 20% SOC, so in order to maintain the lifetime of the battery, the required capacity is then 239/0.8 = 29 mAh.

Since the instrumented sled will be used in cold conditions, an even higher capacity is needed - the ambient temperature has a significant effect on the usable energy capacity of Lithium-based batteries

[50][51]. As can be seen in figure 5.2, at -20 °C the voltage of a typical Li-ion battery drops with about 0.5 V compared to the voltage at 23 °C. Using equation 5.1, it can be estimated that this results in an energy capacity decrease of about 25%, which is a typical energy decrease for Li-ion polymer batteries at these temperatures [52]. Therefore the battery's energy capacity should be at least 298/0.75 = 318 mAh.

To meet this capacity with a good margin and to compensate for ageing, a two-cell Li-Polymer battery with an energy capacity of 1800 mAh, or 13.3 Wh, will be used for this project. This allows the athlete to complete a full training while having some capacity left.

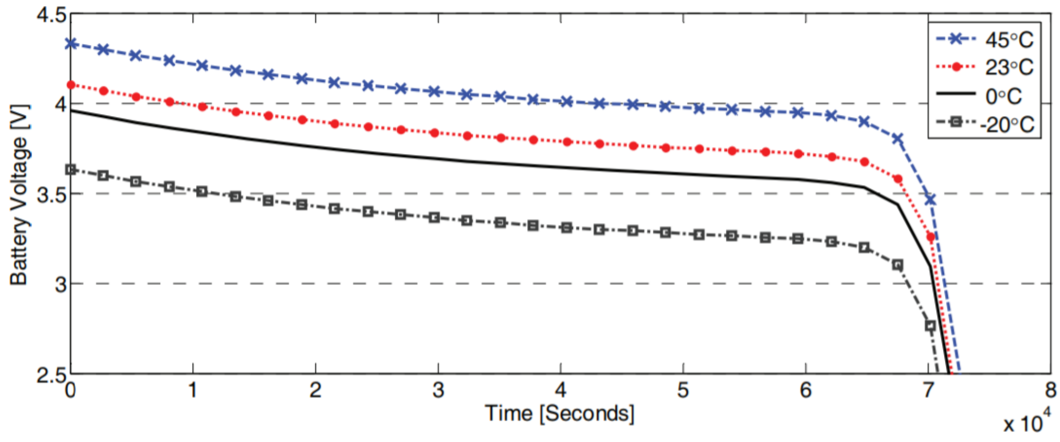


Figure 5.2: Effect of temperature on voltage of Li-ion cell [51].

$$E = \frac{U^2 \cdot t}{R} \tag{5.1}$$

5.2. The implementation

To implement the battery in the system, a battery management system is necessary. This system should be able to convert the battery voltage to the desired voltage of the connected subsystems. Also, the battery management system should include an undervoltage protection to prevent an early wear-out of the battery, as well as an overcurrent protection to protect the battery from high currents and to make the overall system more safe [53]. The following subsections elaborate on these aspects.

5.2.1. Undervoltage Protection

Discharging a Lithium-based battery too deep directly results in a capacity loss of the battery [54]. Naturally, this is an unwanted effect and therefore an undervoltage protection circuit is necessary. The requirement of this circuit is that it should disconnect the battery when its voltage is too low (the voltage corresponds to the state of charge), according to requirement S.2).

The first design made to fulfil the requirements is shown in figure 5.3. Here, R30 and R25 set the voltage threshold at which the battery should be disconnected. When this voltage is reached, Q9 starts conduction, activating Q10. R24 is added as pull-up resistor. When Q10 is conducting, Q15 starts conducting as well, which activates Q17. Q17 is a transistor capable of conducting relatively high currents. R26 is added to limit the base current of Q17. Then, the load can be connected to the emitter of Q17. Although this system has been tested and turned out to work well, it consumed relatively much power. Therefore, a new iteration of the design was made, to save energy and increase the usable capacity for the total system.

For the second design, a dedicated battery protection IC, namely the *R5460N212AF*, has been used to protect the battery against undervoltage and overcurrent. This IC consumes a considerably lower amounts of power [55]. The implementation of this IC is shown in figure 5.4. The IC is shown as U15. LMT2 is a connector that is used to read the individual voltages of both cells in the battery pack. R32 and R31 are used to set the threshold at which the MOSFETS Q11 and Q12 are switched on or off. Capacitors

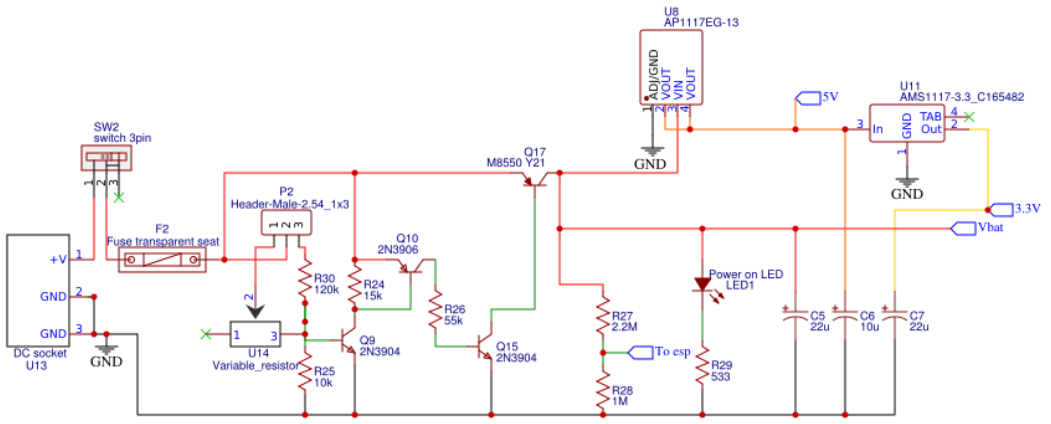


Figure 5.3: First power electronics design

are added to stabilise the system. For the MOSFETs, the *NMOS WSF30100* has been used, because it is able to withstand high currents and has a low $R_{DS(ON)}$ to minimise the voltage drop over the MOSFETs [56]. Also, this MOSFET was chosen because it is surface-mount, to make the system as flat as possible. Button K1 was added after the design was tested. In section 7.3 it is explained why.

When the protection IC detects a voltage per cell that is lower than the threshold set by R32 and R31, it disconnects the battery by disabling the MOSFETs. The threshold voltage per cell is set at 3.0 V. This prevents the system from discharging the battery to even lower voltages that could cause permanent decrease of the battery's stability and capacity, and eventually total failure.
Some batteries are already equipped with such a protection circuit, but the battery used in this project is not. Furthermore, most batteries with a built-in protection IC have a threshold voltage of less then 3.0 V, sometimes even as low as 2.4 V per cell. At this voltage, the battery is already damaged. Therefore, whether the battery has a protection IC or not, it is strongly recommended to use the protection circuit as described.

Because it is desirable that the battery is disconnected before it reaches the absolute minimum threshold of 3.0 V, a buzzer is added to warn the user of the system for low battery voltages. From a voltage of 3.6 V per cell and lower, the buzzer is enabled with an interval that decreases as the voltage decreases. A voltage divider is used to divide the battery voltage to a level that is safe to measure by the *ESP32*.

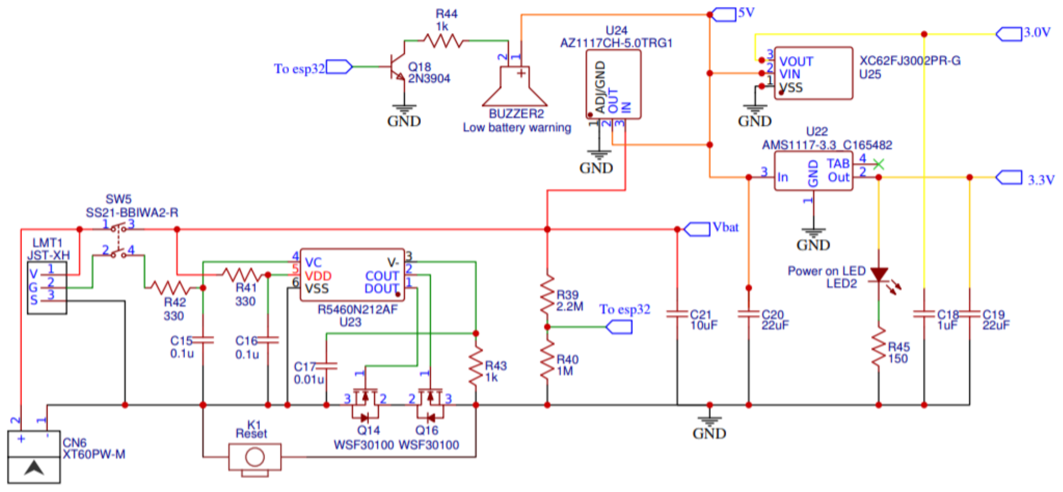


Figure 5.4: Second power electronics design

5.2.2. Overcurrent Protection

According to requirement S.3, an overcurrent protection system is necessary. This prevents large, dangerous currents to flow through the system when a short circuit or other fault occurs. To implement an overcurrent protection, a fuse could simply be added to the system. It is connected directly to the battery, to ensure that the battery is disconnected when a short circuit occurs. To calculate the breaking capacity of the fuse, all the maximum currents of the system are added, with a certain margin, giving the minimum breaking current. This is shown in figure 5.3.

As discussed in section 5.2.1, the design of figure 5.3 has been changed - the new design is shown in figure 5.4. The protection IC is also able to detect a short circuit, and disconnects the battery ground when a short circuit occurs. The advantage of this IC over the fuse is that the new design results in a lower voltage drop between the battery and the load.

5.2.3. Voltage regulation

According to requirement S.6, the system requires three voltages; 5 V, 3.3 V and 3 V. For this purpose, voltage regulators will be used. Most components work reliably on 3.3 V, such as the GNSS receiver, IMU, processor, temperature sensor, the ADC for the force sensor and the op-amps of the force sensor circuits. The 3.3 V regulator input will be connected to the output of the 5 V regulator. As most components are connected to the 3.3 V, it has to be made sure that this regulator is able to deliver enough current. As this regulator is connected to the 5 V regulator, the 5 V regulator should be able to deliver at least as much current. Therefore, a 5 V and 3.3 V regulator with a maximum current of 1.35 A is used to power these components, which should be well enough according to table 5.1 [57]. The 5V regulator will also be used for the force sensors and the battery warning buzzer. Furthermore, a 3 V regulator will be connected to the 5 V regulator. The 3 V regulator will be used to supply a reference voltage to the force sensor circuit. As this regulator only has to supply a reference voltage, a low maximum current suffices. Therefore, a regulator with a maximum current of 200 mA is used [58].

As the 3 V and 3.3 V regulator inputs are connected to the 5 V regulators output, it has to made sure that the $V_{dropout}$ is below 2 V and 1.7 V respectively. Also, the 5 V regulator needs a low $V_{dropout}$, as it is connected to the battery of which the voltage decreases when it is being discharged. Therefore, regulators with a low $V_{dropout}$ have been chosen [58] [57]. To make the final design as flat as possible, only surface-mount regulators have been chosen. Lastly, capacitors are added to improve the transient response and the stability of the power supply. This can all be seen in figure 5.4.

6 PCB Design and Prototype Implementation

Eventually, all the subsystems of the project - the temperature sensor, force sensor, IMU, GNSS receiver, visualisation, SD data logger, power electronics and *ESP32* - must work together as one. To do so, a limited number of pins of the *ESP32* are available. Therefore, a pin-out scheme was made to organise this. To save space as well as the number of available pins, mostly *I2C* and SPI have been used for the data interfaces. These interfaces allow one to connect multiple devices to the same pins, as long as the addresses of the devices are different. The circuit diagram of this total system is shown in appendix D.

Furthermore, according to requirement S.11, all subsystems must be connected in a robust and secure way, while keeping in mind that most of it must fit in a box of 31.5×14.7×2 cm as described in requirement G.5. The battery, power electronics, force sensor circuit, IMU, data logger and ESP32 board must all fit inside this box. To do so, a printed circuit board (PCB) is designed, to which all these systems can be connected. Also, connection terminals for the GNSS receiver, temperature sensor and force sensors can be found on this PCB. Naturally, the force sensors themselves will be on top of the skeleton sled, and the temperature sensor at the bottom of the sled. The GNSS receiver will also be located on top of the sled, to prevent any blocking of the GNSS signals through the carbon fibre of the sled. Lastly, a button together with an indicator LED will be placed on top of the sled, allowing the athlete to start and stop the measurements during his run and to indicate when it is measuring. This button and LED are also connected with terminals to the PCB. The PCB design is shown in figure 6.1. All terminals are labelled, making it easy to connect all sensors properly. To connect all components on the PCB, two layers are used to rout all traces. The remaining area is filled with a copper plane, which is connected to the ground. This is done to reduce the voltage drop caused by the impedance of the ground traces, as well as to reduce electrical noise and cross-talk between traces.

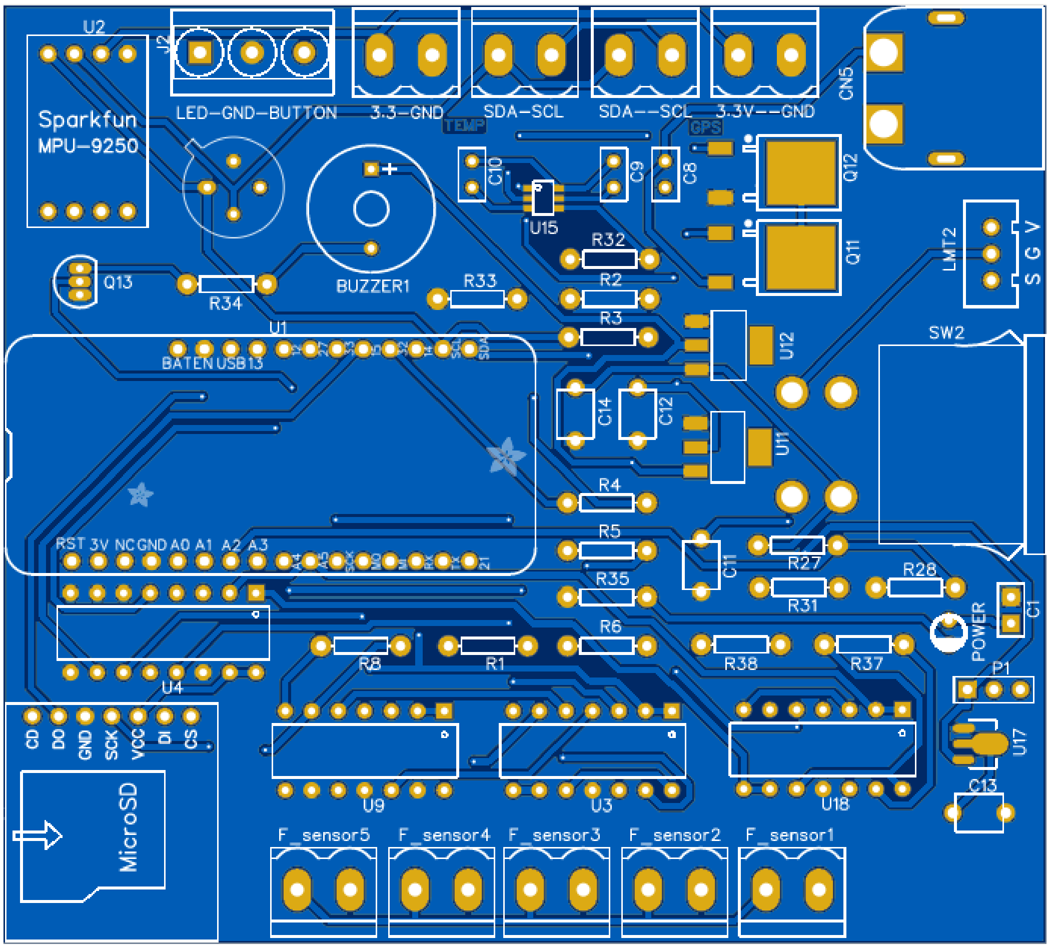


Figure 6.1: The final PCB design.

7 Results and Discussion

This chapter lays out the results of the tests that have been done to test the subsystem, followed by a discussion of these results. Each aspect of the subsystem, as discussed in chapters 3 to 6, will be covered.

7.1. Temperature measurements

The temperature sensor was tested with the *ESP32* and it worked. The temperature output of the sensor was compared with the output of a commercially-available IR temperature measurement device and was closely related. Also, the sensor was tested by measuring the temperature of ice, down to -15 °C. This test resulted in plausible values. By changing the emissivity settings, the output could be tuned. However, it was not possible to test the accuracy of the temperature sensor when measuring ice, because there was no accurate reference available to compare it with. The temperature sensor was able to output data with a frequency of 25 Hz.

7.2. Localisation system

The localisation system consists out of the IMU, the GNSS receiver and the KF. The following subsections will elaborate on these individual aspects as well as the system as a whole.

7.2.1. Inertial Measurement Unit

Figure 7.1 shows the orientation of the axes of the IMU. In order to gain insight into the accuracy of the acceleration and angle measurements, a set of measurements has been done. The IMU has been turned around each axis at an amount of 45°and of 90°. These increments can clearly be seen in the results.

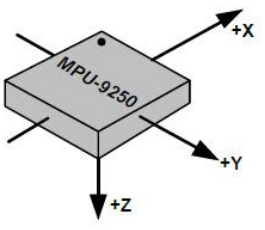


Figure 7.1: The orientation of the axes of the IMU.

Figure 7.2 shows the results of these measurements. First, the IMU was at rest on a table. This can be seen clearly in the specific force on the z-axis, which fluctuates around 10 m/s^2 and can be related to the gravity. At the same time, the x and y-axis experience no force. After that, the roll angle is increased in two steps, in which the positive y-axis is turned towards the positive z-axis. The force of gravity is then reflected on the y-axis. Consequently, the IMU is again returned to its original position. After that, the pitch angle is increased, due to which the positive x-axis is turned towards the positive z-axis. Again, the reflection of gravity can be seen in the specific force. Lastly, the IMU has been turned around the z-axis at 45°and 90°. Where these increments are clearly visible in the pitch and roll, they are hard to see for the yaw. It can be noted that these yaw angle measurements are less accurate than the roll and pitch measurements. Luckily, the roll and the pitch angles are more important for the instrumented sled as these have an impact on the measured G-forces.

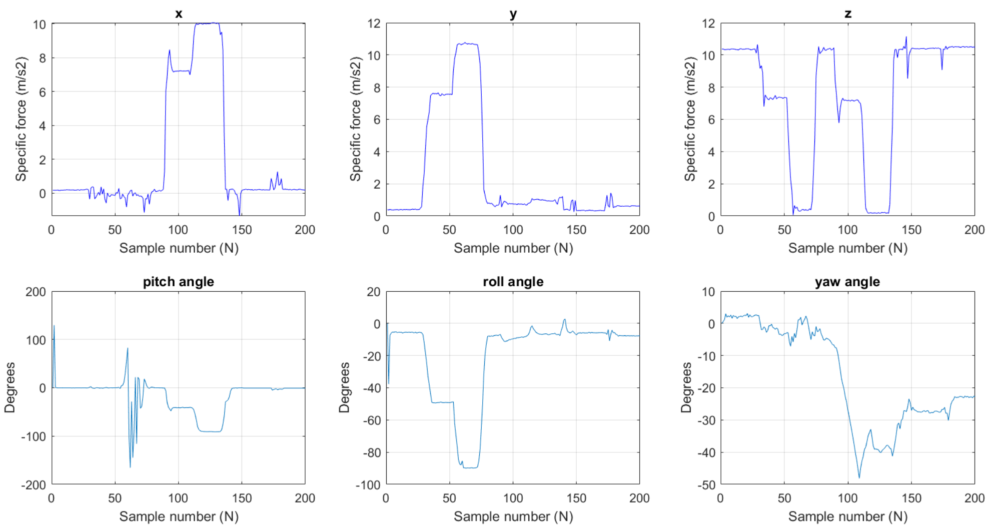


Figure 7.2: Plots of the specific force (upper row) and the angles (lower row) of the IMU.

7.2.2. GNSS receiver

As described in section 4.1 and appendix B, the SAM has been configured to receive only GPS. This has been done to increase the update rate from 10 Hz to 18 Hz. First, a test round has been done by walking on an outside path while GPS and GLONASS were configured. Even though the update rate has been set to 18 Hz, the GNSS receiver was only able to output data at a frequency of 10.3 Hz. After that, all satellite systems other than GPS have been disabled and the same outside path has been taken. Again, the update rate was programmed to 18 Hz and the SAM was now able to reach update rates of 17.8 Hz. Thus, the desired increase in update rate has been reached.

In figure 7.3, the results of the two measurements have been plotted. As can be seen, the GNSS receiver was able to follow the path quite accurately in both cases. However, the accuracy drops when the number of enabled satellite systems drops. This makes sense, since a disable leads to a lower amount of satellites and thus a lower amount of signals that can be used for the computation of the position. While the accuracy of the GPS-only configuration is lower, still the decision was made to keep to this configuration. This is because the update rate is much higher in this case (18 versus 10 Hz). Sampling at 10 Hz and travelling at the maximum skeleton speed, 147 km/h, the data points will be 4.1 m apart. Sampling at 18 Hz while travelling at the same speed, the distance between consecutive data points is almost halved to 2.3 m. While it is a trade-off, the GPS-only configuration seems to have a bigger advantage.

Despite the fact that the frequency has been set to 18 Hz, this frequency is not always constant. Sometimes, gaps in the data acquisition occur, lasting 1 or more seconds. Gaps as long as 4 seconds have been encountered, which correspond to gaps of 160 m. This is an unacceptably large gap that must be fixed. Since the gaps always occur in integer number of seconds, it is suspected that the problem does not lie in a delay of data reception of the SAM, but in the timing of the communication when calling a function. Unfortunately, the cause for these data black-outs has not yet been found. Hence, the *Sparkfun Ublox Arduino* library needs to be inspected in even closer detail to see what could be the cause.

However, a possible solution has already been thought of, that will be worked out in more detail for the protoype as well. This solution relates to the **Q** and **R** matrices of the KF. A timer should be implemented, monitoring the frequency at which the data from the SAM is received. When this time gap

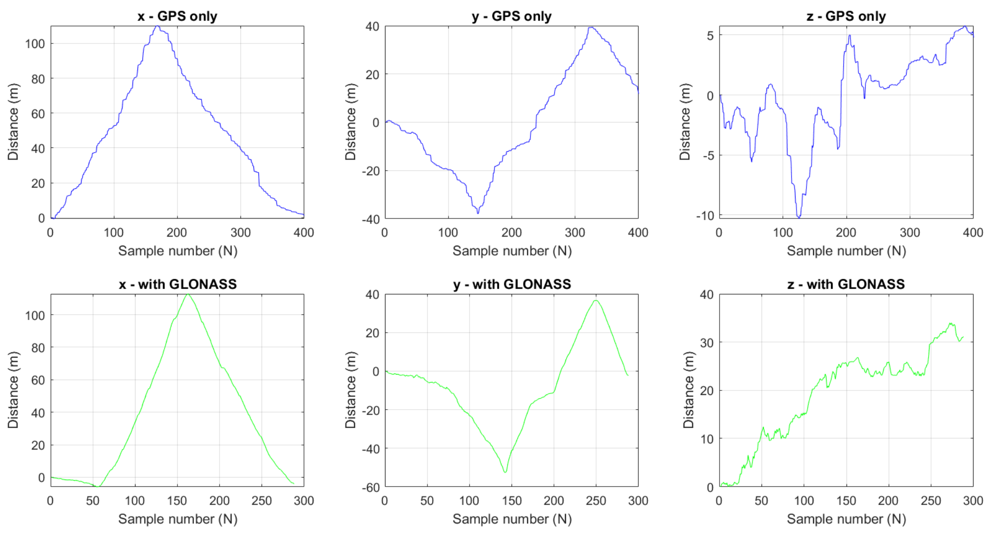


Figure 7.3: The same path with only GPS enabled (above) and GPS + GLONASS enabled (below).

is longer than a specific threshold, the previous values for latitude, longitude and altitude will be used, after which the software goes on to obtain the IMU data. Meanwhile, the measurement noise covariance matrix $\mathbf{R}_{\mathrm{GPS}}$ is set to a high(er) value, indicating that these measurements contain a lot of noise. In contract, the process noise covariance matrix $\mathbf{Q}_{\mathrm{IMU}}$ will be set to a low(er) value, indicating that these measurements contain less noise. Then, the KF will produce the output estimate of the state variable $\hat{x}_k$ in such a way that the IMU measurements will be 'trusted' more than the GPS measurements. This way, the gaps are filled. Once the data black-out of the GNSS receiver is over, the **Q** and **R** matrices return to their original values.

7.2.3. Sensor fusion - Q and R matrices

Now that it has been verified that the IMU and the GNSS receiver work, it is time to validate the fusion of the data through a KF. The MATLAB code of the KF can be found in appendix C.1. For this implementation, the values for the noise covariance matrices **Q** and **R** must be given. As a starting point, the standard deviation of the IMU and SAM outputs have been measured, as stated in section 4.3.

For the IMU, this has been done by positioning the sensor on a table. The outputs of the accelerometer, magnetometer and gyroscope have been logged while the device was kept stationary. From a set of 2000 data points in each of the 9 dimensions, the standard deviation has been calculated. These values can be found in the first three columns of table 7.1. By squaring these values, the variance has been calculated. These values can be found in the last three columns of table 7.1.

IMU	**St. Dev. x**	**St. Dev. y**	**St. Dev. z**	**Var[x]**	**Var[y]**	**Var[z]**
Accelerometer	5,482663	3,373336	4,046577	30,05959	11,37939	16,37479
Magnetometer	7,965674	7,545086	7,464969	63,45196	56,92832	55,72576
Gyroscope	0,049399	0,057063	0,054607	0,00244	0,003256	0,002982

Table 7.1: Standard deviation σ and variance σ^2 of the IMU.

In a similar fashion, the standard deviation of the SAM has been measured. This has been done by positioning the receiver outside, where there were permanently at least 8 satellites in view. A 1000 measurements have been done for the latitude, longitude and altitude. The standard deviation and the variance obtained from these measurements can be found in table 7.2. The **R** matrix follows directly from table 7.2. However, the **Q** matrix requires some more thought. This matrix should contain the variances

GNSS receiver	Latitude	Longitude	Altitude
Standard deviation	210,2243	374,0705	18904,79
Variance	44194,27	139928,7	3,57E+08

Table 7.2: Standard deviation σ and variance σ^2 of the GNSS receiver.

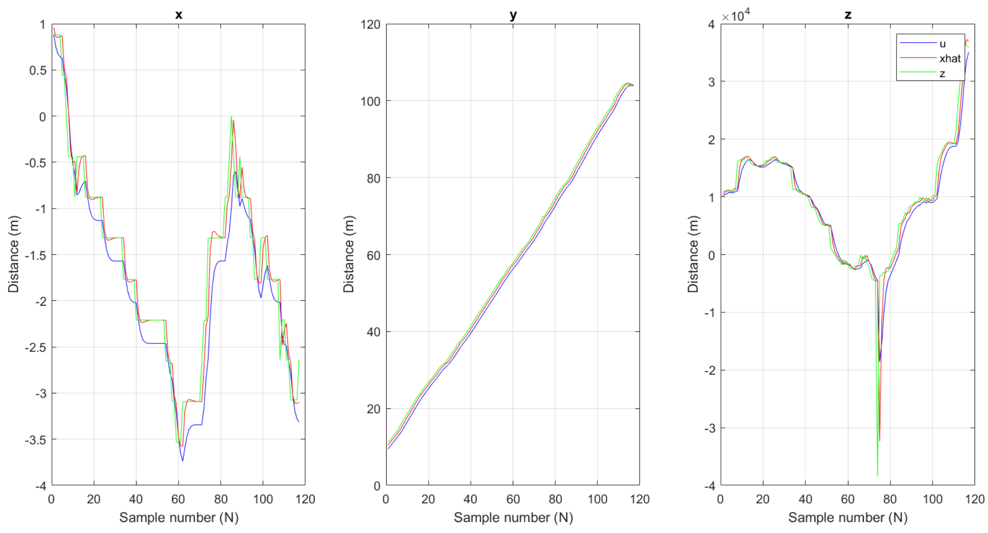


Figure 7.4: Data obtained while walking down lat. 52'0. Kalman filter uses the 6x6 identity matrix for **Q** and the 3x3 identity matrix for **R**.

of the position and velocity, as shown in equation 4.9. However, the variances have been measured for the acceleration. Noting that the position and velocity are obtained from integrating the acceleration once or twice, the assumption has been made that the variances of the position and velocity are equal to those of the acceleration. Thus, the matrices have been determined as follows:

$$\boldsymbol{Q} = \begin{bmatrix} 30.05959 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11.37939 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16.37479 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30.05959 & 0 & 0 \\ 0 & 0 & 0 & 0 & 11.37939 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16.37479 \end{bmatrix} \tag{7.1}$$

$$\boldsymbol{R} = \begin{bmatrix} 44194,27 & 0 & 0 \\ 0 & 139928,7 & 0 \\ 0 & 0 & 35700000 \end{bmatrix} \tag{7.2}$$

7.2.4. Sensor fusion - Kalman filter

Now that the **Q** and **R** matrices have been determined, the KF can be run. A test has been done by walking over the 52nd degree latitude that is marked on the campus. This line gives a clear reference point for the measurements. Data was collected from the IMU and the GNSS receiver and subsequently run through the Kalman Filter. In figure 7.4, the results can be seen for a KF where both the **Q** and **R** are identity matrices. Here, the blue line shows the IMU input, the green line shows the GNSS receiver input and the red line shows the position output as estimated by the KF. Figure 7.5 shows the results when the same input data is used, yet the **Q** and **R** matrices as given in equations 7.1 and 7.2 are used. A larger size of these figures can be found in appendix C.2, as well as two figures where the impact of individually changing the **Q** or **R** matrix can be seen.

As can be seen in figure 7.5, a smooth estimate is calculated by the KF. Since the GNSS receiver is quite accurate in the x and y directions, the estimate resembles the data from the SAM quite closely. Never-

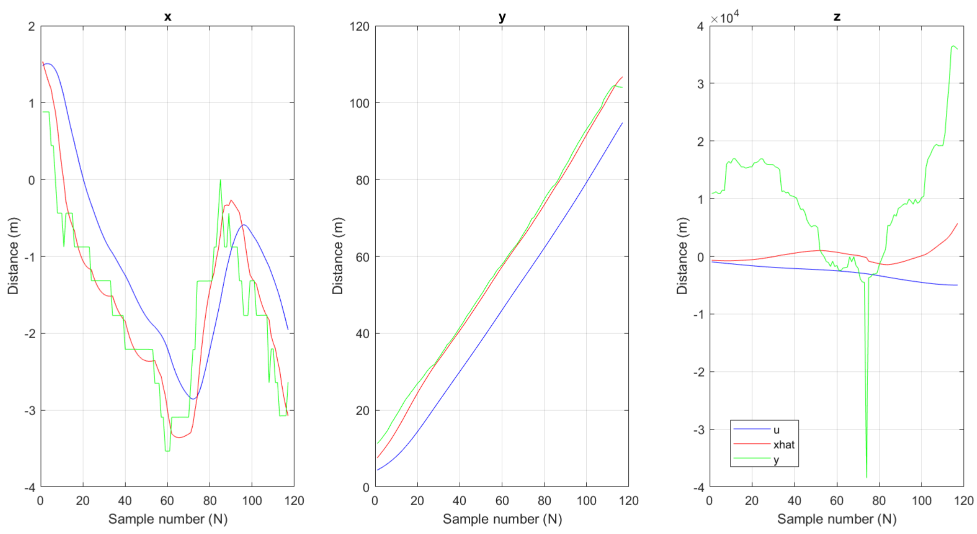


Figure 7.5: Data obtained while walking down lat. 52'0. Kalman filter uses the acceleration variances from table 7.1 for **Q** and the variances from table 7.1 for **R**.

theless, since the receiver is quite inaccurate for the height (z-axis), the data from the IMU weighs more heavily here.

While the **Q** and **R** matrices, as defined in equations 7.1 and 7.2 respectively, form a good basis for the tuning of the KF, the values can be further adjusted to make the Kalman Filter even more accurate. In order to do so, an accurate set of reference data is needed, so that calibration can be done. Furthermore, such a reference set is a good way to validate the Kalman Filter. To obtain this reference data, a visit has been paid to the amusement park of Duinrell, where the rollercoasters can reach high speeds and forces of up to 5 g. Measurements have been done using the IMU and GNSS receiver, as well as using the MATLAB Mobile app. However, upon checking this data, it became clear that the output of the IMU and GNSS receiver was not usable for the purpose of tuning. The data had been logged at too low a frequency and the height readings of the GNSS receiver had not been saved. Especially the angles measured by the IMU exhibit a great amount of noise, but due to the low sampling frequency, only little to no filtering is possible. An example of such noisy IMU data can be found in figure 7.6.

Thus, the final tuning and the true validation of the KF has not yet been possible. Nevertheless, the estimated position (red line) as shown in figure 7.5 already shows that a smooth fusion of the data can be achieved. A new set of reference data must be obtained, logged at a higher frequency. Furthermore, a low-pass filter must be applied over this data. Then, the KF can be tuned even better.

7.3. Power management system

In this section, the results from the design of the power system, including the battery itself and the implementation of the power system, will be discussed.

For this project, a battery with a high volumetric and a high gravimetric energy density is needed. The battery that has been used, is a Li-polymer battery with dimensions of $105 \times 35 \times 15$ mm, and a weight of 91 grams. The nominal voltage of the battery is 7.4 V and the energy capacity is 13.32 Wh, or 1.8 Ah. This means that the gravimetric energy density of the battery is 146 Wh/kg and the volumetric energy density is 242 Wh/l. When looking at figure 5.1, the gravimetric energy density is good and as expected. The volumetric energy density however seems rather low, but this does not pose a problem as there is enough space for this battery in the skeleton sled.

Furthermore, the battery did not fail when exposed to high G-forces or low temperatures, which proves that it is safe to use at the skeleton track - assuming that it is well-protected with a hard case. The

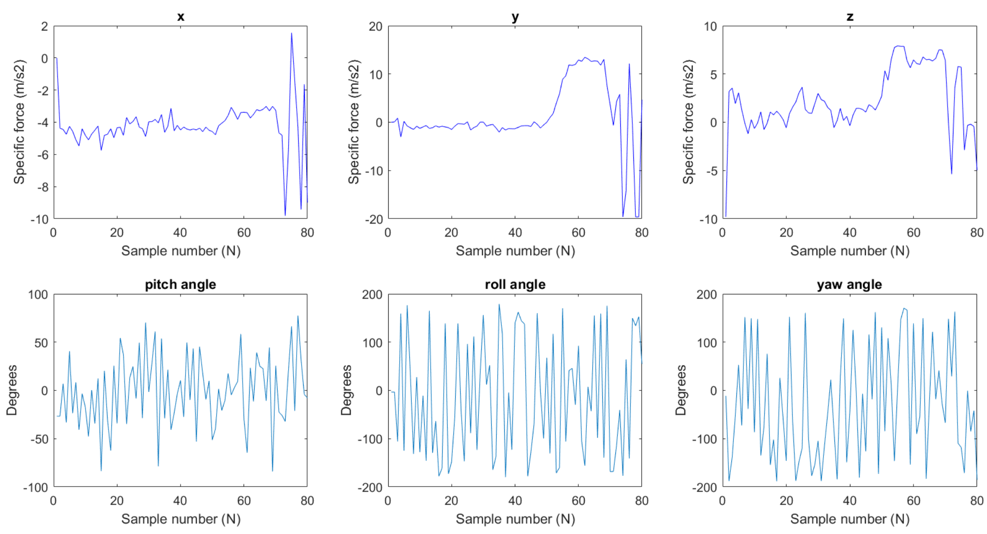


Figure 7.6: Noisy IMU data obtained from the Falcon, a rollercoaster in Duinrell.

exposure to G-forces was done by taking the prototype with a Li-Polymer battery in a rollercoaster, which exposed the battery to forces of 5 g. The performance at low temperatures was tested by freezing the battery to -15 °C and then discharging it while monitoring its voltage and current output.

Also, after powering the system with a fully-charged battery for more than 3 hours, the state of charge of the battery was still 80%. However, it should be noted that this was at room temperature and that at lower temperatures, this remaining state of charge will be lower.
As mentioned before, the battery used for this project was first cooled down to, and then tested, at -15 °C. Also, the battery was tested at +22 °C for comparison. The results are shown in figure 7.7. This figure shows that the voltage indeed drops at lower temperatures, similar to figure 5.2. The measurement was done by connecting a 10 Ω resistor to the fully-charged battery, and then discharging it to 7.1 V while monitoring the current and voltage. The values obtained by this measurement are shown in table 7.3. The begin and end voltage are measured at room temperature. As can be seen, at lower temperatures, the usable capacity is indeed lower. The usable capacity at -15 °C decreased with 23% when compared to the usable capacity at 22 °C, which is well in line with the estimation made in section 5.1.3. A surprising fact is that, when measuring the end voltage of the battery without load at room temperature, the battery that had reached -15 °C is 0.28 V higher than the battery that was +22 °C. This corresponds to a state of charge of about 30%. This suggests that more capacity could have been used, but due to safety reasons as well as to prevent damage to the battery, the battery should not be discharged to an even lower voltage.

Table 7.3: Results of the battery test

Temperature [°C]	Used capacity [Wh]	Begin voltage [V]	End voltage [V]
22	13.3	8.40	7.22
-15	10.3	8.40	7.50

The undervoltage protection circuit has been tested by connecting a power supply to the battery terminals of the power circuit, as depicted in figure 5.4. The battery warning buzzer starts beeping at voltages lower than 3.6 V per cell, with an interval that decreases as the voltage decreases. At 3.0 V per cell or lower, the power input is completely disconnected. This behaviour is as it was designed and therefore it works properly. One flaw of the undervoltage protection, is that when the power input is disconnected at low voltages, it does not reconnect when the voltage is again increased to a safe level. It only reconnects when a charger is connected after the protection circuit. This could be solved by using the *R5460N212AE* protection IC instead of the *R5460N212AF*, which reconnects automatically when the input voltage is at

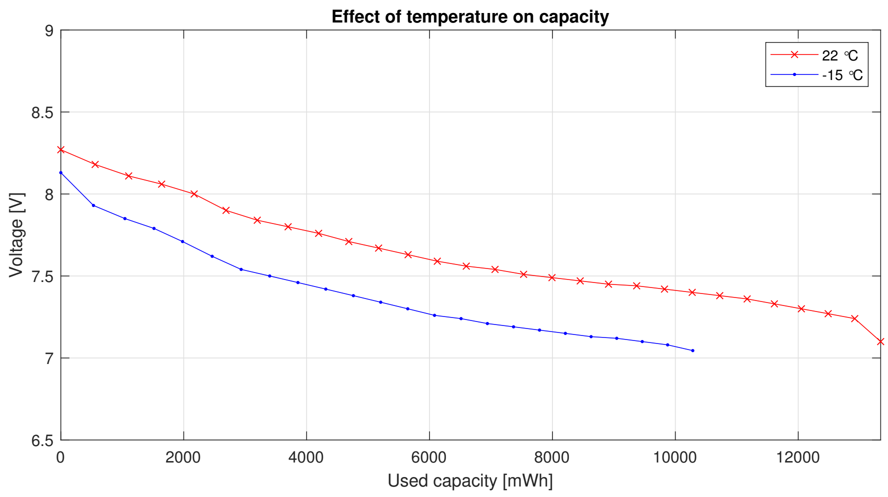


Figure 7.7: Effect of the temperature on the capacity of Li-Polymer battery [51]

a safe level again. However, this component could not be acquired and therefore this problem is solved by using a reset button. This button bypasses the MOSFETS when pushed, to reset the protection IC by simulating that a charger is connected, which is actually the battery itself, assuming that the battery voltage is at a normal level.

The overcurrent protection could not be properly tested, because the risk of damaging any components could not be afforded, as this would cause a delay in the project. However, as the overcurrent protection is done by the same IC that proved to execute the undervoltage protection properly, it is likely that the overcurrent protection works as well. Nevertheless, it is recommended that the overcurrent protection is tested when more resources are available.

The 5 V, 3.3 V and 3.0 V regulators output a stable voltage of 5.06 V, 3.31 V, and 3.06 V respectively. The output was monitored with an oscilloscope and it showed a small ripple. This could simply be noise, yet it could also be due to the internal switching of the regulators. This noise is already suppressed by electrolytic capacitors, but by using ceramic capacitors, which have a lower equivalent series resistance and can filter higher frequencies, the noise could be suppressed even further. These capacitors were however not available in time, but could be added in the future to further improve the output of the voltage regulators.

7.4. PCB design and prototype implementation

The final PCB design is 10×9 cm and with all components on it, it is 1.8 cm high, which allows for the system to fit perfectly inside the skeleton sled. In order to let the PCB fit in even smaller spaces, smaller and more SMD components could be used. Furthermore, instead of one 2-cell Lithium-ion polymer battery, two 1-cell batteries in series could be used, making the battery pack flatter. The PCB was tested with all components on it and proved to behave correctly. All connections were good and no errors were noticed. The design is depicted in figure 6.1. However, it is not yet proven that the data output of the final prototype implementation is correct.

8 Conclusion

8.1. Conclusions

The goal of this project was to design an instrumented sled that can provide useful feedback for skeleton athletes. To reach this goal, several parts have been designed: the temperature measurement system, the localisation system using the IMU and GNSS receiver, the power management system and the integration on a PCB.

Temperature measurements

The temperature sensor is able to measure the temperature of its surroundings and therefore satisfies requirement G.1. Also, the sensor is able to measure the ice temperature without touching it, thus satisfying requirement S.8. The accuracy could not be tested properly, however according to the data sheet the accuracy should satisfy requirement S.9. Lastly, the sensor was able to measure ice temperatures down to -15 °C, which makes it likely that the temperature sensor satisfies requirement S.7, which is theoretically confirmed by its data sheet [46].

Localisation system

The localisation system fused the data obtained from an IMU and a GNSS receiver. This was done using a Kalman Filter. While the GNSS receiver has a frequency of only 18 Hz, the frequency of the IMU can be set to a much higher value - somewhere in the order of kHz. Thus, the localisation software can be run at a higher frequency, with correction updates from the GNSS receiver coming in at a lower rate of 18 Hz. This system has not been implemented yet, but sure is a feasible implementation (requirement G.6). Furthermore, a requirement has been set on the accuracy of the location measurements - these should be at least 1 metre (requirement S.10). Unfortunately, no useful reference data has been obtained in Duinrell and thus no real conclusion can be drawn here. However, new reference data will be obtained such that this specification can be validated. In case the requirement is not met, the KF will be tuned further to improve the accuracy. Lastly, the data has to be available within 5 minutes after each run for the athlete and the coach to use (requirement G.12). This specification has been met for the MATLAB implementation of the KF - all position simulations were finished in less than 0.4 seconds in MATLAB. Looking ahead to the implementation of the KF on the ESP32 microcontroller, the time slot of 5 minutes seems feasible as well.

Power management system

The chosen Li-Polymer battery used for this project was able to fulfil its requirements. First of all, it was able to fit in the space of the skeleton sled, as the dimensions are within the limits of requirement G.5. Furthermore, the high gravimetric energy density resulted in a weight that was relatively low and also well within the limits set by requirement G.4. Also, the battery is rechargeable and is able to supply the system with electrical energy for more than 3 hours, which satisfies requirement S.5 and S.1 respectively. Requirement G.3 is satisfied, as the tested Li-Polymer battery was able to survive G-forces of 5 *g*. As the battery is just 15 mm thick, there was enough room left to protect the battery with a hard case, which satisfies requirement S.4. To implement the battery, according to requirement S.2 and S.3, an undervoltage and overcurrent protection was necessary. This was successfully done by using a battery protection IC. The appropriate voltage needed for the different components of the system, as described by requirement S.6, were successfully made available by using the power electronics as described in section 5.2.3. Overall, by using the power system as described in chapter 5, requirement G.10 is satisfied.

Integration on PCB

The PCB proved to fulfil its requirements as described by requirement G.5, G.11 and S.11. Therefore, it is suited to be used as final product and to be put inside the skeleton sled.

General requirements

All quantities stated in requirement G.1 can be measured individually as described in chapter 7 and by the other subgroups[26] [59]. However, due to a still unknown error in the GNSS receiver implementation, the sample time sometimes gets delayed to several seconds, as explained in section 7.2. Because of this, the sample rate is not always high enough to satisfy requirement G.6.
Since the means to measure the temperature of the environment in which the product will be used and the vibrations on the track due to irregularities in the surface of the ice are not available, it can not be checked whether requirements G.2 and G.11 are met. However, as mentioned in chapter 7, a prototype of the system was tested during several roller coaster rides. The prototype was not yet mounted on the dedicated PCB that was designed, as the PCB had not yet been delivered at that time. Since the final product will be mounted on the PCB, it will be even more robust.

The product must be easily removed and installed on the sled, as dictated by requirement G.8. The inside of the sled is fairly easily accessible, therefore the product can be easily removed or installed. As the force transducers are attached to the sled using double-sided tape, they could leave some easy-to-remove traces on the sled. The measurements, however, should be started without accessing the circuitry inside the sled, as stated in requirement G.7. Just like the force transducers and the GNSS receiver, a momentary switch is mounted on the outside of the sled. This momentary switch sends a signal to the *ESP32* to start the measurements.

The dimensional requirements listed in G.5 and G.4 have been met. The dimensions of the part of the product that should fit inside the sled are 11.5 × 12.5 × 1.8 cm and the weight of the total system is 261 grams. The product, however, does have a minimal influence on the aerodynamics of the sled due to parts of the system being mounted on the outside of the sled, such as the GNSS receiver and the switch for starting the run G.9.

As discussed in chapter 7, the system has been tested at G-forces of up to approximately 5 g. The system was able to acquire data in these conditions, however, due to an error in the code, the measured data turned out to be of doubtful use. This means that it can not be said whether requirement G.3 is met. As described by subgroup C, the processed information is sent to a mobile device via Wi-Fi [59], meaning that no external devices have to be connected to the product in order to use it, in accordance with requirement G.10.

All that is needed to use the product, is a push on the button to start the measurements and a push after the run, as well as a mobile device to read the data, satisfying requirement G.13. According to requirement G.12, the acquired data should be available to the athlete and the coach within 5 minutes. As described in the thesis of subgroup C [59], the time it takes to process the data and present it on the web page takes less than 20 seconds.
The final general requirement, G.14, states that the total costs of making the prototype should be within the budget of €250,-. This requirement was not met, the overall costs of the prototype are approximately €400,-.

8.2. Recommendations and future work

To make sure that the temperature sensor is accurate, it is recommended that it is tested on ice, while comparing the output to a reference of which it is known to be accurate.

For the location system, a high-quality set of reference data must be obtained. Using this data, the Kalman Filter should be validated and tuned to meet the specifications S.10 and G.6. Then, a good Kalman filter library should be found that is compatible with the *ESP32*, into which the **Q** and **R** matrix can be loaded. Furthermore, it is recommended to add a low-pass filter, as it would improve the accuracy by removing much of the noise caused by vibrations. Also, finding a GNSS receiver with an even higher update rate could improve the accuracy of the system.

Furthermore, to get rid of the need to have a reset button for the battery protection IC, it is recommended to use the *R5460N212AE* IC instead of the *R5460N212AF*.

To meet requirement G.14, cheaper components can be used. Also, due to the fact that this was still the research phase, the product became more expensive than necessary, as for instance different pressure sensors have been ordered. The product could be implemented cheaper once it is know what exactly is needed. Lastly, by scaling up the production, the price could be further reduced.

References

[1] I. Roberts, “Skeleton Bobsleigh Mechanics: Athlete-Sled Interaction”, PhD dissertation, Univ. Edinburgh, Edinburgh, United Kingdom, 2013.

[2] C. Sawade, S. Turnock, A. Forrester, and M. Toward, “Assessment of an Empirical Bob-Skeleton Steering Model”, *Procedia Engineering*, vol. 72, pp. 447–452, 2014, The Engineering of Sport 10, ISSN: 1877-7058. DOI: `https://doi.org/10.1016/j.proeng.2014.06.078`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S1877705814005943`.

[3] S. Xiaochen. (Jun. 1, 2018). Plans Unveiled for Building All 2022 Olympic Venues, [Online]. Available: `http://www.chinadaily.com.cn/a/201806/01/WS5b1081d9a31001b82571d8c3.html` (visited on 06/21/2019).

[4] F. Braghin, F. Cheli, S. Maldifassi, S. Melzi, and E. Sabbioni, *The Engineering Approach to Winter Sports*. Springer, 2016, ISBN: 978-1-4939-3019-7.

[5] F. M. Impellizzeri, A. La Torre, G. Merati, E. Rampinini, and C. Zanoletti, “Relationship Between Push Phase and Final Race Time in Skeleton Performance”, *J. Strength Cond. Res.*, vol. 20, no. 3, pp. 579–583, 2006.

[6] A. Baca, P. Dabnichki, M. Heller, and P. Kornfeind, “Ubiquitous Computing in Sports: A Review and Analysis”, *J. Sports Sciences*, vol. 27, no. 12, pp. 1335–1346, Oct. 2009. DOI: `10.1080/02640410903277427`.

[7] S. Lee, T. Kim, S. Lee, S. Kil, and S. Hong, “Development of Force Measurement System of Bobsled for Practice of Push-off Phase”, *Proc. IMechE Part P: J. Sports Engineering and Technology*, vol. 229, no. 3, pp. 192–198, 2015. DOI: `10.1177/1754337114565383`.

[8] I. J. M. Roberts, “Skeleton bobsleigh mechanics: Athlete-sled interaction”, PhD thesis, The University of Edinburgh, Jul. 2013.

[9] A. Seymour-Pierce, B. Lishman, and P. Sammonds, “Recrystallization and damage of ice in winter sports”, *Royal Society Publishing*, vol. Microdynamics of ice, 2017. DOI: `https://doi.org/10.1098/rsta.2015.0353`.

[10] E. Jansons, J. Lungevics, K. Stiprais, L. Pluduma, and K. A. Gross, “Measurement of sliding velocity on ice, as a function of temperature, runner load and roughness, in a skeleton push-start facility”, *Cold Regions Science and Technology*, vol. 151, pp. 260–266, 2018. DOI: `https://doi.org/10.1016/j.coldregions.2018.03.015`.

[11] J. Fraden, *Handbook of Modern Sensors*. Springer, 2016, ISBN: 978-3-319-19302-1.

[12] G. Alvarez-Botero, F. E. Baron, C. C. Cano, O. Sosa, and M. Varon, “Optical sensing using fiber bragg gratings: Fundamentals and applications”, *IEEE Instrumentation Measurement Magazine*, vol. 20, no. 2, pp. 33–38, Apr. 2017, ISSN: 1094-6969. DOI: `10.1109/MIM.2017.7919131`.

[13] B. G. Liptak, *Instrument Engineers’ Hanbook, Volume One: Process Measurement and Analysis*. CRC Press, 2003, ISBN: 9781420064025.

[14] G. S. Ranganath, “Black-body radiation”, *Resonance*, vol. 13, no. 2, pp. 115–133, 2008.

[15] T. P. Merritt and F. F. Hall, “Blackbody radiation”, *Proceedings of the IRE (IEEE)*, vol. 47, no. 9, pp. 1435–1441, 1959. DOI: `10.1109/JRPROC.1959.287032`.

[16] B. Ning and Y. Wu, “Research on non-contact infrared temperature measurement”, *2010 International Conference on Computational Intelligence and Software Engineering*, 2010. DOI: `10.1109/CISE.2010.5677034`.

[17] H.-Y. Chen and C. Chen, “Determining the emissivity and temperature of building materials by infrared thermometer”, *Construction and Building Materials*, vol. 126, pp. 130–137, 2016.

[18] *Ir thermometers & emissivity*, Bacto Laboratories Pty Ltd, 2005.

[19] *Tmp007 infrared thermopile sensor with integrated math engine*, Texas Instruments, Apr. 2014.

[20] *Mlx90614 family, single and dual zone infra red thermometer in to-39*, 11th ed., Melexis, Jul. 2017.

[21] *Thermopile ir sensor*, Thermometrics - Amphenol Sensors, 2014.

[22] Phidgets. (). Phidgettemperaturesensor ir, [Online]. Available: `https://www.phidgets.com/?tier=3&catid=14&pcid=12&prodid=1041`. (accessed: 02.05.2019).

[23] *Mlx90614 changing emissivity, how to ... (example included), including unlocking cell 0x0f*, 2nd ed., Melexis, Jul. 2013.

[24] *Application note, smbus communication with mlx90614*, 4th ed., Melexis, Jan. 2008.

[25] J. S. D Baranowski H van Boerum, *Plan and track technical data, bobsled/luge track & buildings bid package 2*, Park City skeleton track, 2001.

[26] M. J. Heller and A. J. de Jong, "Instrumented Skeleton Sled: Focusing on Force and Orientation Sensing", BSc Thesis, Delft University of Technology, 2019.

[27] *Sam-m8q data sheet*, u-blox AG, 2017.

[28] *Quectel l76 compact gnss module*, Quectel Wireless Solutions Co., 2014.

[29] *Fgpmmopa6h gps standalone module data sheet*, GlobalTop Technology Inc., 2011.

[30] *Neo-m8p u-blox m8 high precision gnss modules*, u-blox, 2017.

[31] *Sam-m8q receiver description including protocol specification*, UBX-13003221, u-blox AG, May 2017.

[32] *Mpu-9250 product specification*, 1.0, InvenSense, Jan. 2014.

[33] M. M. Atia and S. L. Waslander, "Map-aided adaptive gnss/imu sensor fusion scheme for robust urban navigation", *Measurement*, vol. 131, pp. 615–627, 2019, ISSN: 0263-2241. DOI: `https://doi.org/10.1016/j.measurement.2018.08.050`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0263224118307899`.

[34] E. Bostanci, "Motion model transitions in gps-imu sensor fusion for user tracking in augmented reality", Dec. 2015.

[35] M. d. T. Peral, F. G. Bravo, and A. MartinhoVale, "State variables estimation using particle filter: Experimental comparison with kalman filter", in *2007 IEEE International Symposium on Intelligent Signal Processing*, Oct. 2007, pp. 1–6.

[36] MATLAB, *Understanding kalman filters, part 4: An optimal state estimator algorithm*, May 2017. [Online]. Available: `https://nl.mathworks.com/videos/understanding-kalman-filters-part-4-optimal-state-estimator-algorithm--1493129749201.html`.

[37] ——, *Understanding kalman filters, part 3: An optimal state estimator*, May 2017. [Online]. Available: `https://nl.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html`.

[38] V. V. M. Verhaegen, *Filtering and System Identification, a least squares approach*. Cambridge, 2007, ISBN: 9781107405028.

[39] Y. Kim and H. Bang, "Introduction to kalman filter and its applications", in. Nov. 2018. DOI: `10.5772/intechopen.80600`.

[40] I. Skog, *Sensor fusion gps+imu, module 1 - sensing and perception*, KTH Royal Institute of Technology, Stockholm, 2016.

[41] M. Kok, J. D. Hol, and T. B. Schön, *Using Inertial Sensors for Position and Orientation Estimation*. now, 2017, ISBN: 1680833561. [Online]. Available: `https://ieeexplore.ieee.org/document/8187588`.

[42] D. Deng, "Li-ion batteries: Basics, progress, and challenges", *Energy Science & Engineering*, vol. 3, no. 5, pp. 385–418, 2015. DOI: `10.1002/ese3.95`.

[43] V. Beggi and L. Loisel, "Microgrid in usth campus : Architecture and power management strategies", Master's thesis, Ecole Centrale de Marseille, 2018.

[44] F. Lambert, "Tesla model s battery caught on fire 'without accident', says owner – tesla is investigating", *Electrek*, Jun. 2018. [Online]. Available: `https://electrek.co/2018/06/16/tesla-model-s-battery-fire-investigating/`.

[45] T. RAVPower, *Lithium ion vs. lithium polymer batteries – which is better?*, Aug. 2017. [Online]. Available: `http://blog.ravpower.com/2017/06/lithium-ion-vs-lithium-polymer-batteries/`.

[46] *Mlx90614 data sheet*, 5th ed., Melexis, Mar. 2009.

[47] *ESP32 series*, ESP32, Version 3, Espressif Systems, 2019.

[48] lady Ada and D. Nosonowitz, *Adafruit huzzah32 - esp32 feather*, Adafruit, 2019.

[49] *Mcp3204/3208: 2.7v 4-channel/8-channel 12-bit a/d converters with spi serial interface*, English, version 1 Revision E, Microchip Technology Inc., 2008, 29 pp.

[50] J. D. Dogger, B. Roossien, and F. D. J. Nieuwenhout, "Characterization of li-ion batteries for intelligent management of distributed grid-connected storage", *IEEE Transactions on Energy Conversion*, vol. 26, no. 1, pp. 256–263, 2011. DOI: `10.1109/TEC.2009.2032579`.

[51] O. Erdinc, B. Vural, and M. Uzunoglu, "A dynamic lithium-ion battery model considering the effects of temperature and capacity fading", in *2009 International Conference on Clean Electrical Power*, Jun. 2009, pp. 383–386. DOI: `10.1109/ICCEP.2009.5212025`.

[52] B. G. Kim, F. P. Tredeau, and Z. M. Salameh, "Performance evaluation of lithium polymer batteries for use in electric vehicles", in *2008 IEEE Vehicle Power and Propulsion Conference*, Sep. 2008, pp. 1–5. DOI: `10.1109/VPPC.2008.4677513`.

[53] "Ieee standard for rechargeable batteries for multi-cell mobile computing devices", *IEEE Std 1625-2008 (Revision of IEEE Std 1625-2004)*, pp. c1–79, Oct. 2008. DOI: `10.1109/IEEESTD.2008.4657368`.

[54] H. Maleki and J. N. Howard, "Effects of overdischarge on performance and thermal stability of a li-ion cell", *Journal of Power Sources*, vol. 160, no. 2, pp. 1395–1402, 2006, Special issue including selected papers presented at the International Workshop on Molten Carbonate Fuel Cells and Related Science and Technology 2005 together with regular papers, ISSN: 0378-7753. DOI: `https://doi.org/10.1016/j.jpowsour.2006.03.043`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0378775306004277`.

[55] *R5460x2xx series data sheet*, RICOH.

[56] *Wsf30100 data sheet*, WINSOK Semiconductor, Dec. 2014.

[57] *Az1117c data sheet*, 3.2, Diodes Incorporated, Oct. 2014.

[58] *Xc62fj data sheet*, TOREX SEMICONDUCTOR LTD.

[59] A. W. G. Hunter and T. Moree, "Instrumented Skeleton Sled: Focusing on the Data Processing and the User Interface", BSc Thesis, Delft University of Technology, 2019.

A
IR sensor - setting emissivity

```
//////      Bachelor Graduation Project - Instrumented Sled for Skeleton
//////      Code for the implementation of the IR sensor, MLX90614
//////      This file can be used to change the emissivity of the sensor;
//////      This calibration needs to be done once before using the sensor
//////      After the setting is complete, please turn the sensor off once
//////      After restarting, the device is ready to be used with proper calibration
//////      Program can differentiate between rough and smooth ice,
//////      requests input from keyboard
//////      Implementation is largely inspired by the Sparkfun MLX90614 library


// Include the necessary libraries
#include <Wire.h>                          // I2C library
#include <Adafruit_MLX90614.h>             // Library to read temperature from MLX
#include <Streaming.h>                     // Library for easy printing

// Declare global variables
const uint8_t deviceAddress = 0x5A;        // Default address of the sensor
const uint8_t clearPEC = 0x3E;             // PEC used to clear the 0x04 memory cell
const int command = 0b00100100;            // Opcode: command 001 to access EEPROM,
                                           // 00100 to address mem cell 0x04
// Create object from class
Adafruit_MLX90614 mlx = Adafruit_MLX90614(deviceAddress);


void setup() {
  Serial.begin(9600);                      // Set baud rate (bits/sec)
  mlx.begin();                             // Prepare communication with MLX
  Serial << "Communication with MLX90614 started" << endl;

  Serial << "Is the structure of the ice smooth or rough? Please enter r for rough or
  s for smooth " << endl;
}


void loop() {
  uint8_t LB = 0, HB = 0;                  // Variables for High and Low Data Bytes
  uint8_t  newPEC;                         // Variable for the PEC for writing
  char iceConsistency;                     // Stores user's choice: smooth/rough

  if (Serial.available() > 0)              // Read if there is data available
  { iceConsistency = Serial.read();        // Store the user's input

    if (iceConsistency == 's')             // User chose smooth ice
     // Send; LB_smooth, HB_smooth, PEC_smooth (LB and HB come from 0xF851)
     {  execute(0x51, 0xF8, 0xC1);  }

    else if (iceConsistency == 'r')        // User chose rough ice
     // Send; LB_rough, HB_rough, PEC_rough (LB and HB come from 0xFAE0)
     {  execute(0xE0, 0xFA, 0x95);  }
  }

  return;
}


void execute(uint8_t LB, uint8_t HB, uint8_t PEC)
{
  // In order to upload a new value, the EEPROM cell needs to be erased first
  clearAddress(deviceAddress);

  // Set the new value
  setNewValue(deviceAddress, LB, HB, PEC);

  // Check if the new value has been set correctly
  checkSetValue(deviceAddress);

  Serial << "Emissivity setting MLX90614 complete" << endl;
}


// Address needs to be cleared before it can be altered. Steps come from manual.
void clearAddress(char devAddress) {
  Wire.beginTransmission(devAddress);        //1. Send START bit 2. Send Slave Address
  Wire.write(command);                       //3. Send Command
  Wire.write(0);                             //4. Send Low data 0x00
  Wire.write(0);                             //5. Send High data 0x00
  Wire.write(clearPEC);                      //6. Send PEC
  Wire.endTransmission(deviceAddress);       //7. Send STOP bit

  delay(5);                                  //8. Wait 5ms
  Serial << "Clearing EEPROM cell complete" << endl;
  return;
}


// Set the new value in 0x04. Steps come from manual.
void setNewValue(uint16_t devAddress, char LB, char HB, char PEC) {
  Wire.beginTransmission(devAddress);        //1. Send START bit 2. Send Slave Address
  Wire.write(command);                       //3. Send Command
  Wire.write(LB);                            //4. Send Low Byte
  Wire.write(HB);                            //5. Send High Byte
  Wire.write(PEC);                           //6. Send PEC
  Wire.endTransmission(devAddress);          //7. Send STOP bit

  delay(5);                                  //8. Wait 5ms
  Serial << "Emissivity setting MLX90614 complete - await check" << endl;
  return;
}


// Read the contents of 0x04 to doublecheck. Steps come from manual.
void checkSetValue(int devAddress) {
  // Variables to store the checked bytes (Data Byte Low and Data Byte High)
  uint16_t DBL = 0, DBH = 0, CRC = 0;

  Wire.beginTransmission(devAddress);        //1. Send START bit 2. Send Slave Address
  Wire.write(command);                       //3. Send Command
  Wire.endTransmission(false);               //4. Send Repeated START_bit
  Wire.requestFrom(devAddress, 3);           //5. Send Slave Address + Rd\-Wr bit**
  DBL = Wire.read();                         //6. Read Data Byte Low (master must ACK)
  DBH = Wire.read();                         //7. Read Data Byte High (master must ACK)
  CRC = Wire.read();                         //8. Read PEC (master can send ACK or NACK)
  Wire.endTransmission(devAddress);          //9. Send STOP bit

  delay(5);                                  //10. Wait 5ms
  Serial << "Check complete - this has been set: " << DBL << " and " << DBH << endl;
  Serial << "Please turn the device off and on for the procedure to take effect" << endl;
  return;
}
```

B
GNSS receiver - SAM-M8Q

B.1. *UBX-CFG-GNSS* message structure

The UBX message *UBX-CFG-GNSS* must be sent to enable GPS and disable the other systems. The message has the following structure:

Header	Class	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x06	0x3E	4 + 8*numConfigBlocks	Actual message	CK_A CK_B

Table B.1: Structure of the *UBX-CFG-GNSS* message.

The payload contains the actual message: the information on the configuration of the different satellite systems. This payload is structured as follows:

msgVer	numTrk ChHw	numTrk ChUse	numConfig Blocks	gnssId	res TrkCh	max TrkCh	Reserved	Flags
Byte 0	Byte 1	Byte 2	Byte 3	4+8*N	5+8*N	6+8*N	7+8*N	8+8*N

Table B.2: Structure of the payload.

Here, *msgVer* and *numTrkChHw* describe the message version and the number of available tracking channels in the hardware respectively. After that, *numTrkChUse* sets the number of channels that will be used, followed by *numConfigBlocks* that sets the number of configuration blocks. In this application, the number of configuration blocks is 4. Namely, 1 block for enabling GPS and 3 blocks for disabling BeiDou, QZSS and GLONASS respectively. Then, *gnssId* up to *flags* describe the settings for each satellite system. For GPS, the *gnssId* is 0 while BeiDou has *gnssId* 3, QZSS 5 and GLONASS 6. The number of channels for GPS is set to 32, while the number for the other systems is set to 0. After that, the flag is set to 1 for GPS, and 0 for the other systems. The payload then looks as follows:

Payload	**msgVer**	**numTrkChHw**	**numTrkChUse**	**numConfigBlocks**	
	0x00	0x32	0x32	0x04	
GNSS	**gnssId**	**resTrkCh**	**maxTrkCh**	**Reserved**	**Flags**
GPS	0x00	0x16	0x32	0x00	0x01
BeiDou	0x03	0x00	0x00	0x00	0x00
QZSS	0x05	0x00	0x00	0x00	0x00
GLONASS	0x06	0x00	0x00	0x00	0x00

Table B.3: Contents of the payload.

B.2. *UBX-CFG-GNSS* message code

```
///////    Bachelor Graduation Project - Instrumented Sled for Skeleton
///////    Code for the implementation of the GNSS receiver - SAM-M8Q
///////    This file can be used to enable GPS and disable all other systems;
///////    This needs to be done once before using the receiver,
///////    Such that the maximum update rate can be increased to 18 Hz
///////    Code makes use of the Sparkfun Ublox library to work with the UBX struct

// Include the necessary libraries
#include <Wire.h>
#include "SparkFun_Ublox_Arduino_Library.h"
#include <Streaming.h>

// Create object from class
ubxPacket packetCfg;
SFE_UBLOX_GPS myGPS;

void setup() {
  Serial.begin(9600);                 // Set baud rate (bits/sec)
  while (!Serial);                    // Wait for terminal to open
  Wire.begin();                       // Prepare communication with ublox

  if (myGPS.begin() == false)         // Connect to the Ublox module using Wire port
  {
    Serial.println(F("Ublox GPS not detected at default I2C address. Please check wiring. Freezing."
    ));
    while (1);
  }

  myGPS.setI2COutput(COM_TYPE_UBX); // Set the I2C port to output UBX only

  enableGPS();                        // Enable only GPS, turn GLONASS off
  Serial << "Glonass off" << endl;

  myGPS.saveConfiguration();          //Save the current settings to flash and BBR
  Serial << "Config saved " << endl;
}

void loop() {
  Serial << "check " << endl;
}

boolean enableGPS()
{
  packetCfg.cls = 0x06;               // Class for UBX-CFG-GNSS
  packetCfg.id = 0x3E;                // ID for UBX-CFG-GNSS
  packetCfg.len = 36;                 // Message length (bytes) for UBX-CFG-GNSS
  packetCfg.startingSpot = 0;

  if (myGPS.sendCommand(packetCfg, 250) == false) // This will load the packetCfg.payload array with
      current settings of the given register
    return (false);                   // If command send fails then bail

  //packetCfg.payload is now loaded with current bytes. Change only the ones we need to
  packetCfg.payload[0] = 0x00;       // Message version, 0 here
  packetCfg.payload[1] = 32;         // numTrkChHw
  packetCfg.payload[2] = 32;         // numTrkChUse, use 32 channels
  packetCfg.payload[3] = 4;          // No. of ConfigBlocks, 4 configurations will follow

  packetCfg.payload[4] = 0x00;       // Enable GPS, gnssId 0
  packetCfg.payload[5] = 16;         // resTrkCh (minimum number of reserverd tracking channels)
  packetCfg.payload[6] = 32;         // mxTrkCh (maximum number of TrkCh)
  packetCfg.payload[7] = 0;          // Reserved
  packetCfg.payload[8] = 0x01;       // Flag, 1 to enable GPS L1C/A
  packetCfg.payload[9] = 0x00;
  packetCfg.payload[10]  = 0x00;
  packetCfg.payload[11]  = 0x00;     // End of repeated block

  packetCfg.payload[12]  = 0x03;     // Disable BeiDou, gnssId 3
  packetCfg.payload[13]  = 0;        // resTrkCh, reserve 0 channels
  packetCfg.payload[14]  = 0;       // mxTrkCh (maximum number of TrkCh)
  packetCfg.payload[15]  = 0;       // Reserved
  packetCfg.payload[16]  = 0x00;    // Flag, 0 to disable
  packetCfg.payload[17]  = 0x00;
  packetCfg.payload[18]  = 0x00;
  packetCfg.payload[19]  = 0x00;    // End of repeated block

  packetCfg.payload[20]  = 0x05;    // Disable QZSS, gnssId 5
  packetCfg.payload[21]  = 0;       // resTrkCh, reserve 0 channels
  packetCfg.payload[22]  = 0;       // mxTrkCh (maximum number of TrkCh)
  packetCfg.payload[23]  = 0;       // Reserved
  packetCfg.payload[24]  = 0x00;    // Flag, 0 to disable
  packetCfg.payload[25]  = 0x00;
  packetCfg.payload[26]  = 0x00;
  packetCfg.payload[27]  = 0x00;    // End of repeated block

  packetCfg.payload[28]  = 0x06;    // Disable GLONASS, gnssId 6
  packetCfg.payload[29]  = 0;       // resTrkCh, reserve 0 channels
  packetCfg.payload[30]  = 0;       // mxTrkCh (maximum number of TrkCh)
  packetCfg.payload[31]  = 0;       // Reserved
  packetCfg.payload[32]  = 0x00;    // Flag, 0 to disable
  packetCfg.payload[33]  = 0x00;
  packetCfg.payload[34]  = 0x00;
  packetCfg.payload[35]  = 0x00;    // End of repeated block

  return (myGPS.sendCommand(packetCfg, 250));
}
```

C
KF implementation

C.1. MATLAB code

```
%% Initialization
Ts = 1/18;                        % Sampling period (18 Hz)
N = 576;                          % Number of samples, should be adjusted manually
i = 1;                            % Variable for loop

% 6xN:
xhat_k_k = ones(6,N);             % State vector
xhat_kPlus_k = ones(6,N);         % Estimation state vector
xhat_kPlus_kPlus = ones(6,N);     % Updated estimation state vector

% F = 6x6: State transition matrix
F = eye(6);
F(1,4) = Ts;
F(2,5) = Ts;
F(3,6) = Ts;

% G = 6x3: Control matrix
a = (Ts^2)/2;
G = [a 0 0; 0 a 0; 0 0 a; Ts 0 0; 0 Ts 0; 0 0 Ts];

% H = 3x6: Observation matrix
H = eye(3,6);

% u = 3xN: IMU input
euler = ones(3,N);                % [yaw; pitch; roll], angles from IMU
s = zeros(3,N);                   % [s_x; s_y; s_z], specific force from IMU
u = zeros(3,N);                   % [a_x; a_y; a_z], acceleration after R_n_b and - g
g = [0; 0; 1];                    % Gravitation vector, points down the z-axis
IMU = zeros(3,N);                 % Will store position from IMU (u)
sIMU = zeros(3,N);                % Will store position from IMU (s)

% z = 3xN: GNSS input
z_geodetic = ones(3,N);           % [lat; long; height]
z = zeros(3,N);                   % [p_x; p_y; p_z] in NED

% Navigation vectors
xNorth = zeros(1,N);              % Used for NED conversions
yEast = zeros(1,N);
```

```
zDown = zeros(1,N);

% Reference ellipsoid for the conversion from Geodetic to NED
WGS = referenceEllipsoid('wgs84', 'm');

% e = 3xN: error
e = ones(3,N);                  % Error vector

% 6x6:
Q = eye(6);                     % Process noise (IMU)
I = eye(6);                     % 6x6 identity matrix
P_k_k = eye(6);                 % Error covariance
P_kPlus_k = eye(6);             % Estimation error covariance
P_kPlus_kPlus = eye(6);         % Updated estimation error covariance
P_archive = ones(6,6,N);        % To store error covariance matrices

% 3x3 matrices:
S = eye(3);                     % Innovation covariance
K = eye(3);                     % Kalman gain
R = eye(3);                     % Observation noise (GNSS)
R_n_b_archive = ones(3,3,N);    % To store rotation matrices


%% Data importeren
A = dlmread('52_lat_testrun');  % Load data

s(1,:) = ((A(:,5)).')./1000;    % Import IMU data to vector s
s(2,:) = ((A(:,6)).')./1000;    % Divide by 1000 to convert from mm/s to m/s
s(3,:) = ((A(:,7)).')./1000;

euler(1,:) = (A(:,14)).';       % Import yaw, pitch, roll to vector euler
euler(2,:) = (A(:,15)).';
euler(3,:) = (A(:,16)).';

z_geodetic(1,:) = ((A(:,2)).')./10000000;  % Import raw data (lat, long, height)
z_geodetic(2,:) = ((A(:,3)).')./10000000;
z_geodetic(3,:) = ((A(:,4)).')./1000;

zN1 = z_geodetic(1,1);          % Define begin position for reference position
zE1 = z_geodetic(2,1);
zD1 = z_geodetic(3,1);


%% Kalman algorithm
for i = 1
   xhat_k_k(:,1) = zeros(6,1);  % Define first state as all zeros
   i = i + 1;
end

for i = 2:N                     % Loop through all samples

    % Calculate values rotation matrix, to transform body to navigation frame
    R_yaw = [cos(euler(1,i)) sin(euler(1,i)) 0;
    -sin(euler(1,i)) cos(euler(1,i)) 0; 0 0 1];

    R_pitch = [cos(euler(2,i)) 0 -sin(euler(2,i));
    0 1 0; sin(euler(2,i)) 0 cos(euler(2,i))];

    R_roll = [1 0 0; 0 cos(euler(3,i)) sin(euler(3,i));
    0 -sin(euler(3,i)) cos(euler(3,i))];

    R_n_b = R_yaw * R_pitch * R_roll;       % Calculate rotation matrix
    R_n_b_archive(:,:,i) = R_n_b;           % Save the rotations

    % Convert IMU output to the right frame (body to NED) and compensate for gravity
    u(:,i) = (R_n_b * s(:,i)./9.81)-g;

    % Convert GNSS output to the right coordinate frame (lat/long to NED)
    [xNorth(i), yEast(i), zDown(i)] = geodetic2ned(z_geodetic(1,i), z_geodetic(2,i),
    z_geodetic(3,i), zN1, zE1, zD1, WGS);
    z(:,i) = ([xNorth(i), yEast(i), zDown(i)]).';

    % Predict xhat and P
    xhat_kPlus_k(:,i) = F * xhat_k_k(:,i-1) + G * u(:,i);
    P_kPlus_k = F * P_k_k * F.' + Q;

    % Update e, S and K
    e = z(:,i-1) - H * xhat_kPlus_k(:,i);
    S = H * P_kPlus_k * H.' + R;
    K = P_kPlus_k * H.' * inv(S);

    xhat_kPlus_kPlus(:,i) = xhat_kPlus_k(:,i) + K * e;
    P_kPlus_kPlus = (I - K * H) * P_kPlus_k;

    % Feedback xhat and P for next iteration
    xhat_k_k(:,i) = xhat_kPlus_kPlus(:,i);
    P_archive(:,:,i) = P_k_k;
    P_k_k = P_kPlus_kPlus;

    % Increment i
    i = i+1;
end
```

C.2. Results Kalman filter

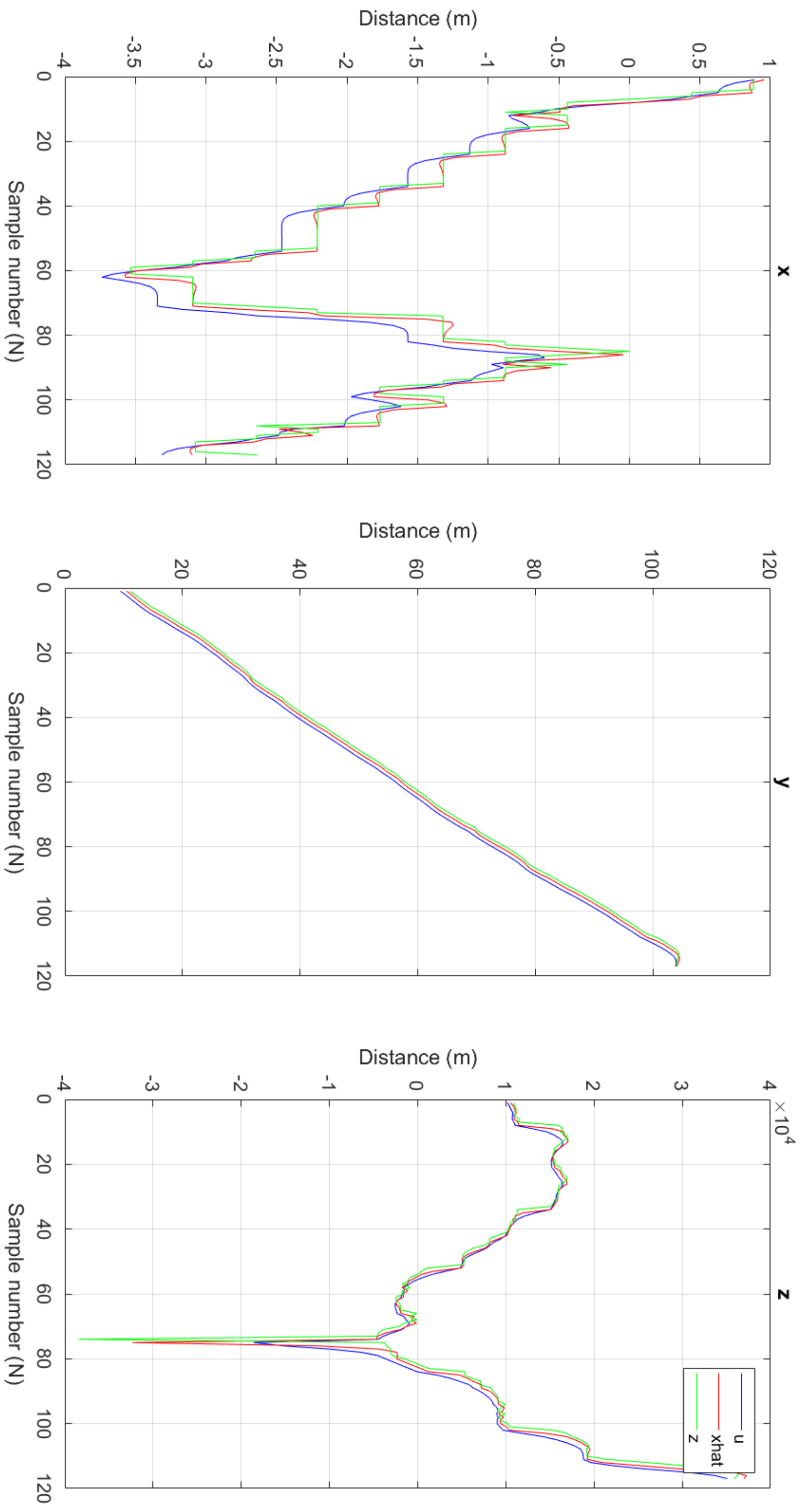


Figure C.1: Data obtained while walking down latitude degree 52. The Kalman filter uses the 6x6 identity matrix for **Q** and the 3x3 identity matrix for **R**.

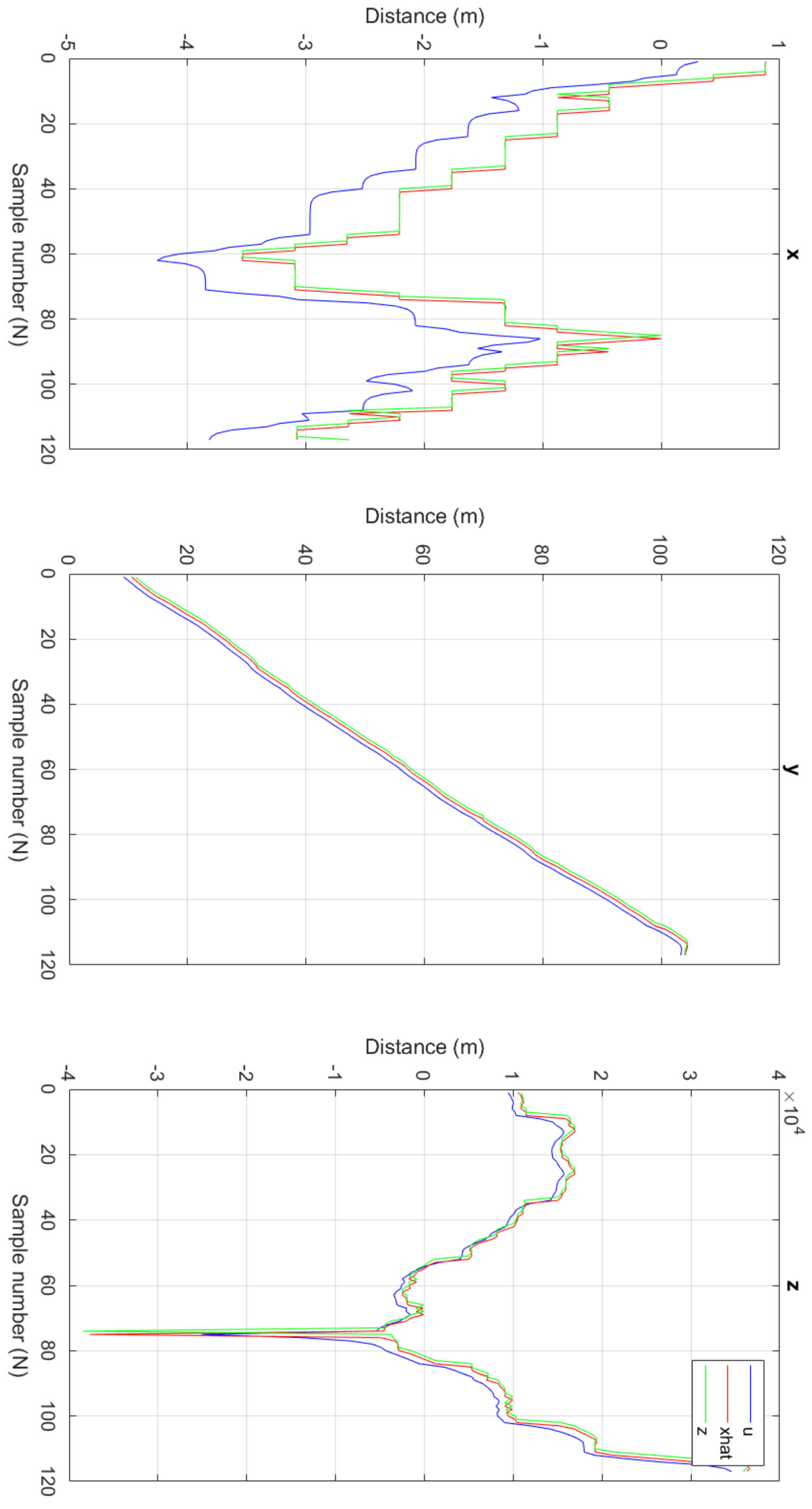


Figure C.2: Data obtained while walking down latitude degree 52. The Kalman filter uses equation 7.1 for **Q** and the 3x3 identity matrix for **R**.

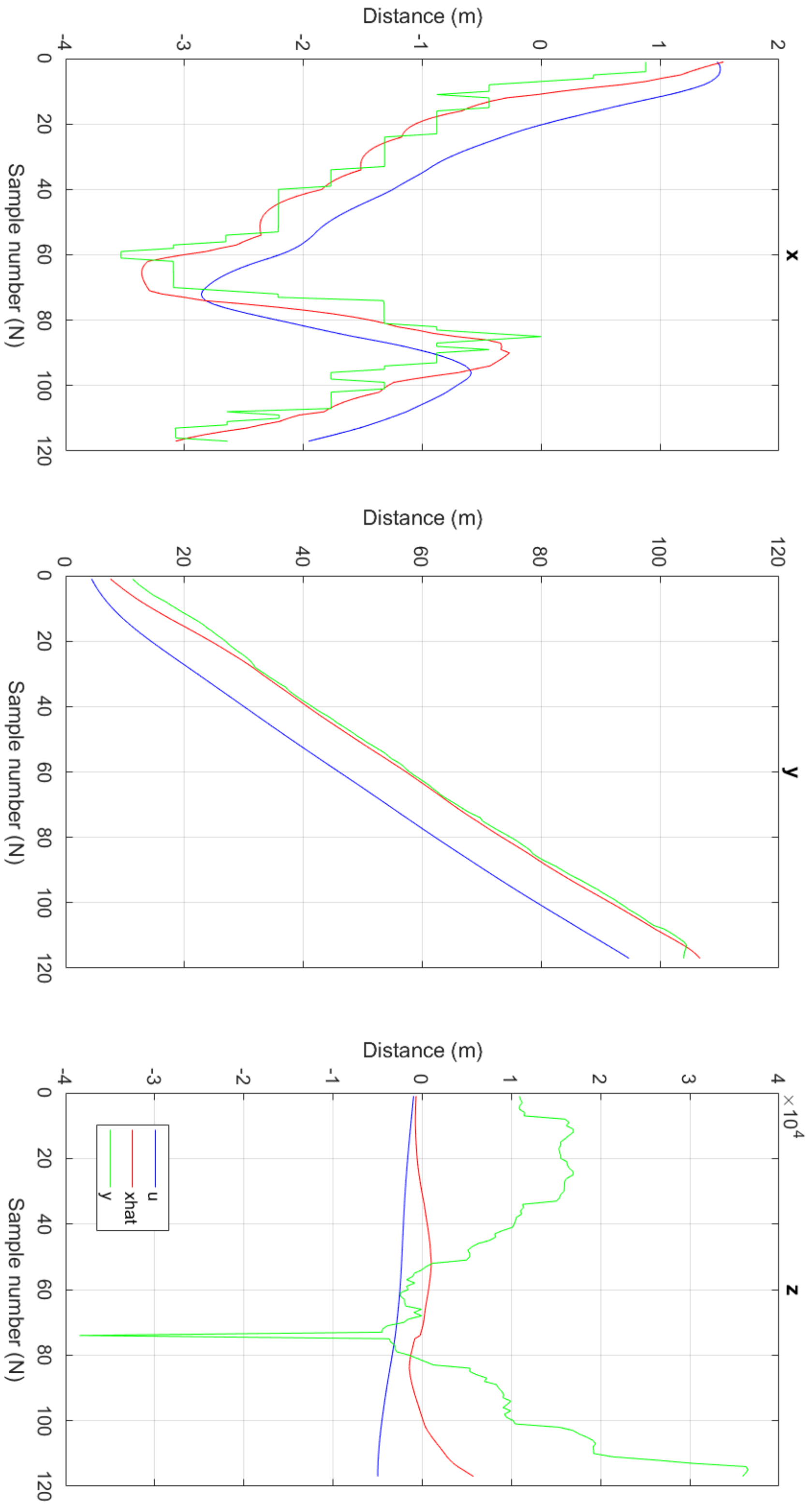


Figure C.3: Data obtained while walking down latitude degree 52. The Kalman filter uses the 6x6 identity matrix for **Q** and equation 7.2 for **R**.

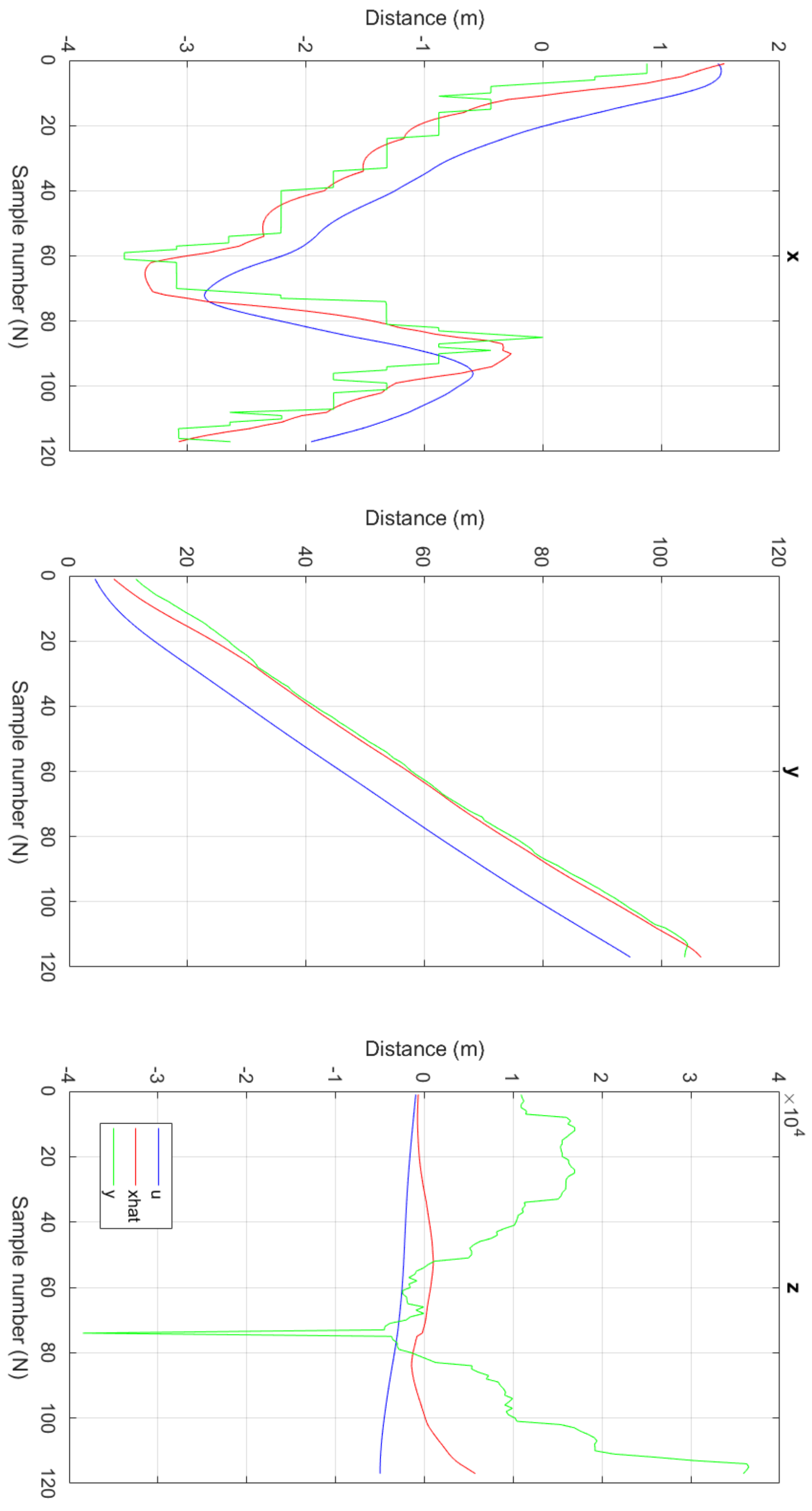


Figure C.4: Data obtained while walking down latitude degree 52. The Kalman filter uses equation 7.1 for **Q** and equation 7.2 for for **R**.

D

Total circuit diagram

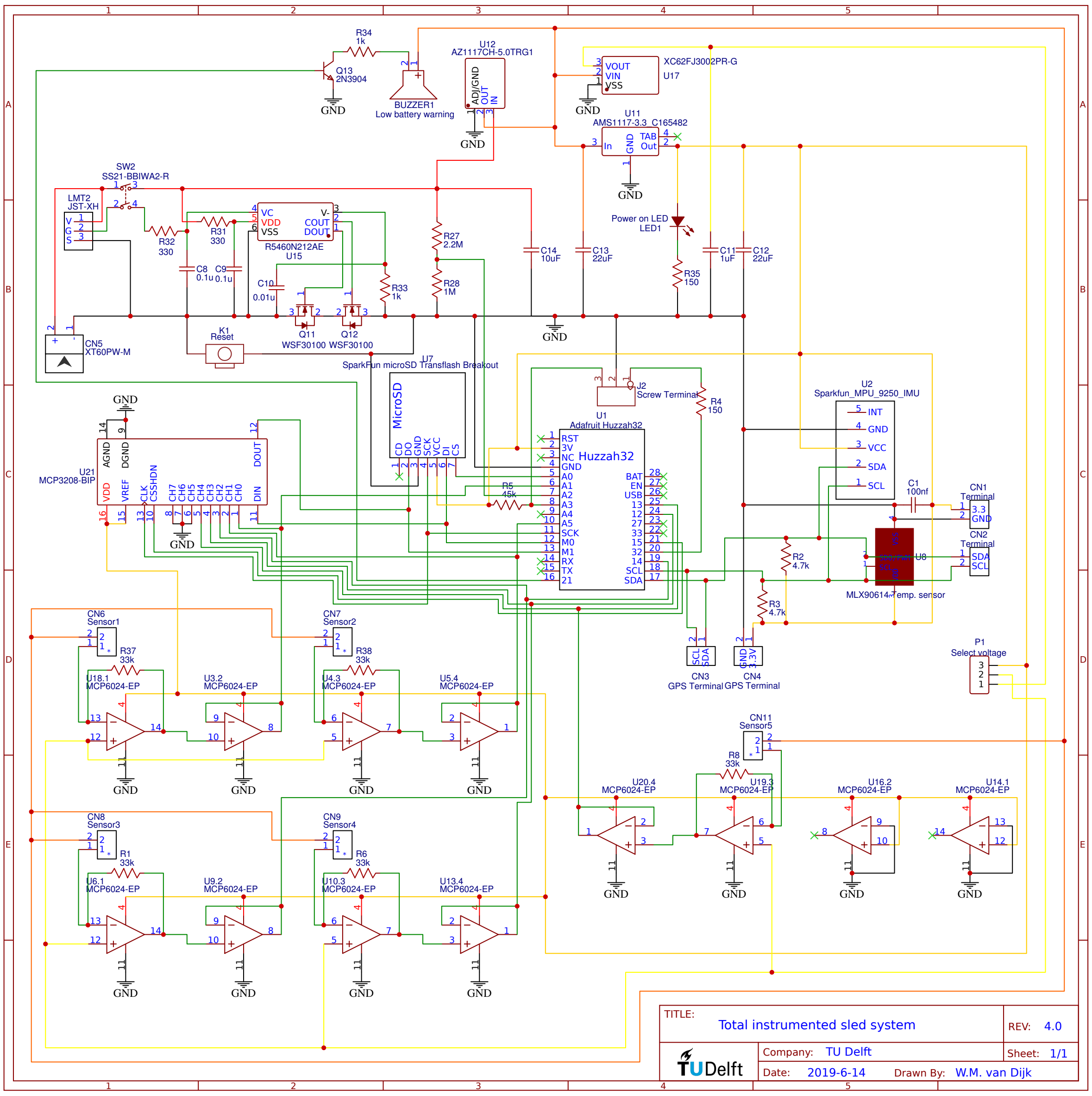


Figure D.1: Circuit diagram of total system