Bachelor Thesis

GonioTrainer for sport

D. Kester

J. Overdevest

TU Delft

Bachelor Thesis

GonioTrainer for sport

by

D. Kester
J. Overdevest

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended on Thursday July 2, 2015 at 9:00 AM.

Student:	D. Kester	4221745
	J. Overdevest	4215109
Project duration:	April 20, 2015 – June 22, 2015	
Thesis committee:	Prof. dr. ir. R. Fastenau,	TU Delft
	Dr. ir. N. P. van der Meijs,	TU Delft
	Ing. J. Bastemeijer,	TU Delft
	MSc. O. den Braver,	O'Sports

This thesis is confidential and cannot be made public until December 31, 2018.

TU Delft Delft University of Technology

Preface

The choice for the ”GonioTrainer for sports” project for our bachelor thesis was really not difficult. A combination of sport and engineering makes this project ideal for our personal and professional growth. Now that the project is coming to an end, we realise the impact it had on us. During the project, we were able to use all our acquired knowledge to identify, solve, test and document problems and solutions *on our own*.We are thankful to the other group members, Bas Bosma, Rene Miedema, Ingo Schilken and Mitchel Scoop, for being a huge inspiration. The project would certainly not have been possible without the help of our technical supervisor Jeroen Bastemeijer. Not only would we like to thank him for his technical support, but also for his encouragement when we faced difficulties. We would also like to thank Otto den Braver en Diederik van der Steen (O'Sports) for their enthusiasm, quick response and supply of resourses. The project was supervised by Prof.dr. K.A.A. Makinwa.

D. Kester
J. Overdevest
Delft, June 2015

Contents

Abstract

In speed skating, the frontal area of the athlete is one of the major factors that influences the aerodynamic drag he/she experiences [1]. The athlete can minimise the frontal area by going lower, and thus bending more his/her knees. O'Sports would like to manufacture the GonioTrainer, a device that will measure the knee of the skater and provide a stimulus to remind the skater to stay low. In order to make this possible, sensors to measure the knee angle and an Inertial Measurement Unit are used to record the athlete gestures. Data storage needs to be implemented to analyse the data afterwards on a dedicated computer. The system must be designed as efficient as possible with scalability in mind, resulting in a system ready for production. This is the motivation behind the BSc thesis work presented in this document.

This thesis project focuses on the development of the GonioTrainer's measurement module, or goniometer: a microcontroller connecting the peripherals and controlling the signals. Measuring the knee angle and an Inetrial Measuremet Unit was done through I^2C at a sample rate of 100Hz. Storing the raw data on a flash memory was possible through SPI and as well as transferring the contents to a smartphone though Bluetooth. Once the information is on a smartphone, it can be uploaded to an internet server for easy access. Also, a special peak detection algorithm for speed skating was developed and explained in appendix A of this document. Transferring the sensors measurements in real time to the smartphone was not possible due to software/hardware conflicts. Note that the GonioTrainer emerged from the idea to improve the performance of ice-skaters, but the end product is ready to be used within different sport disciplines, once the conflicts are solved.

1

Introduction

One of competitive forms of ice skating is known as speed skating. The winner is the one who travels a predetermined distance in the least amount of time. Today, time differences between competitors are incredibly small and are being measured in milliseconds. Every physical aspect that could contribute to decrease time by a fraction of a second is currently profoundly scrutinised to improve the performance. O'Sports, which will be referred to as "the client", has shown that the knee angle (see figure 1.1.) is the most representative parameter to measure the performance of the skater. With this in mind, the client wants to create the GonioTrainer. The GonioTrainer is a system that will continuously measure the knee angle of the athlete during the training and will provide direct feedback when needed in the form of a physical stimulus to the body.

The client already has prototypes of the GonioTrainer, but are still struggling with the electrical part of the system. They would like to manufacture the first batch of 100 GonioTrainers by the end of 2015. This is why they came to us, a group of 6 electrical engineering students. Our task (which will be our Bachelor End Project) is to design, implement and test the robustness of the whole electrical system. A first attempt to design the electrical part has already been done by a group of two students. From their conclusions, the client was able to make specific choices about the sensors, the actuators and other (minor) specifications. This left the client four major concerns about the electrical system: data storage, wireless communication, microcontroller and sensors configuration and the power supply. In this thesis the link between the sensors and microcontroller together with the data storage of the GonioTrainer will be addressed.

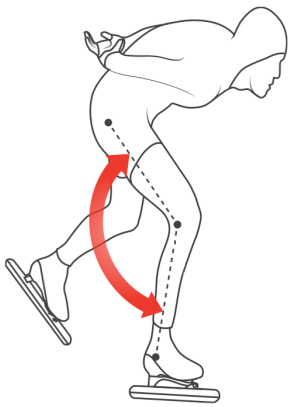

Figure 1.1: The knee angle has been shown to be the most representative parameter to measure the performance of the skater [2]

Figure 1.2 is a simple representation of the GonioTrainer system. The goniometer is the device that will measure the knee angle and will send the measurements wireless to a smartphone. If a stimulus is needed, the smartphone will instruct the feedback module to give a stimulus to the athlete. The smart phone will also provide some statistics during the training, so the coach/athlete can see the progress during the actual training. All measured data during a training will be stored entirely for a more advanced analysis after the training. The GonioTrainer has also attracted the attention of other sports, where measuring a particular joint within the body could be of great interest. Some examples of these sports include rowing, skiing and even sailing have shown to have interest in this product. The different conditions where the GonioTrainer could be used will be studied in order to develop a multi-sport GonioTrainer.

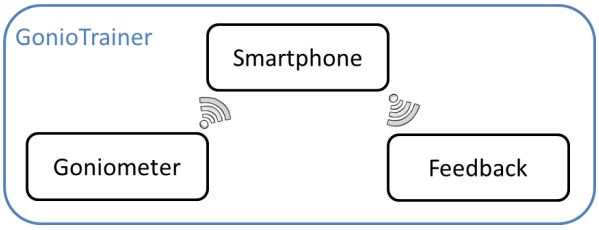

Figure 1.2: An overly simplified block schematic of the GonioTrainer.

2

Problem definition

O'Sports, our client, would like to design, produce and sell the GonioTrainer. Such product is currently unavailable on the market and the client wants to be the first. The client has some experience with electronics, but not enough to develop the electronics needed for the GonioTrainer. A group of six electrical engineering students has the task of doing so. This group is split in three subgroups of two students each addressing specific tasks. Ideally, at the end of the project, all the efforts will be combined to result in the electrical system of the GonioTrainer.

2.1. Problem Scope

In this report the design, implementing and testing of the data acquisition and logging of the GonioTainer will be described. Wireless communication and power supply will not be addressed here, this is the task of the others subgroups. Although power and communication is beyond the scope of our task, it will still be taken into consideration in order to have the most efficient GonioTrainer.

The client wants to measure the knee angle (or any other joint in the body) with a contactless angle encoder. The measurements will determine how an athlete is performing on a specific sport activity, such as speed skating. The measurements need to be transferred to a smartphone for data analysis, or need to be recorded in another way in the case the smartphone is not available. The last case corresponds to an official competition, were measurements devices are allowed, but no smartphones. After the competition, the measurements will be uploaded to the smartphone. When finishing a session, the user should have easy access to the data. With this data, more detailed analysis can be performed on a dedicated computer.

Using the measurements from the joint angle and dedicated algorithms, it is possible to determine whether the athlete needs to change its posture. If this is the case, the feedback module will make sure the athlete is aware of it. Because the user is won't notice that he/she is doing something wrong and his/her focus may lay on something else than his posture (e.g. during high intensity training, posture and technique is often sacrificed for speed and power), it is important to provide the feedback on the *right* moment. According to the client, the perfect moment to give feedback is 50ms after the placement of the skate on ice (for speed skating). The client also concluded that it is not possible to detect this placement moment with only the angle measurements. For this, the client wants an Inertial Measurement Unit (IMU) to be available on the GonioTrainer. With this unit, the placement moment should be easily found using accelerometer measurements.

The previous issues will be addressed in this report, taking the communication with the smartphone, power supply and feedback module for granted. Algorithms regarding any kind of sports are not discussed here. The intention of this project is to provide the client with the electrical system of the GonioTrainer, so they can further develop their product with their knowledge on a robust system.

2.2. Design Requirements

The GonioTrainer consists of three parts: the goniometer, the smartphone and the feedback module. In order to use the GonioTrainer, the athlete will place the goniometer on the joint to be measured;

Figure 2.1: Artist impression of the goniometer on an ice skater [2]

the feedback module is placed where he/she feels comfortable receiving feedback. Finally he/she will link both devices to his smartphone via Bluethooth. The athlete is now ready to start his training. The GonioTrainer must be able to run continuously for at least 3 hours. This implies that the power supply should at least provide energy for the specified 3 hours. The power supply is implemented in [3].

2.2.1. Physical specifications

The goniometer should weight less than 67 grams and fit into cylindrical enclosure of 5mm height and 35mm diameter. The goniometer will have two arms with straps at the ends to attach the device to the athlete. Figure 2.1 shows an artist impression of the goniometer on an ice skater. The feedback module should weight around 33 grams and fit into a 10mm x 17.5mm x 30 mm enclosure.
The GonioTrainer needs to be water resistant. This implies that the goniometer and the feedback module must have as little as possible physical ports. The less ports, the more easy it is to make the enclosure waterproof. The GonioTrainer will be used in damp, sweaty and/or wet environments with temperatures ranging from -10 to 40 degrees celsius.

2.2.2. System specifications

The GonioTrainer must measure the joint angle with a sample rate larger than 60Hz, preferably 100Hz. In order to measure the angle, the client specified the AS5600 magnetic encoder. The magnetic encoder will be placed in the centre of the goniometer. The measurements will be send real-time via Bluetooth to the smartphone. The smartphone will decide if it is appropriate to give the athlete a stimulus. If a stimulus is to be given, the smartphone will instruct the feedback module via Bluethooth to do so. Besides the magnetic encoder, the goniometer must be equipped with an IMU (inertia measurement unit) with the same specifications as the IMU of the iPhone 4s smartphone.
The measurements from the magnetic encoder and the IMU need to be recorded. How the measurements are recorded is not specified by the client, as long as all the data is available at the end of a training ready to be analysed by a personal computer.
The GonioTrainer must also meet the following specifications:

- The software licenses and tools must be freely available.
- Real time calculations such as a peak detection must be possible.
- IMU with interrupt system
- The whole electrical system for each GonioTrainer must fit in a €150 budget for a batch of 10 pieces, and €60 budget for a batch of 100 pieces.
- The GonioTrainer is required to have a minimum life span of 3 years.

3

Related Search

Before the implementation and design of the GonioTrainer is described, the different components will be discussed and explained.

3.1. Microcontroller

The goniometer can be considered to be an embedded system, therefore a MicroController Unit (MCU) will be used. The client expressed great interest in the nRF51822 System on Chip (SoC) by Nordic Semiconductors. This SoC combines a 2.4 GHz transceiver with a ARM® Cortex-M0 CPU together with a range of peripherals and memory options. The ARM Cortex Microcontroller Software Interface Standard (CMSIS) hardware abstraction layer is implemented on the SoC and is available for the CPU. The interesting peripheral blocks for the goniometer are the Serial Peripheral Interface (SPI) which enables full duplex synchronous communication, and the Two-wire interface (TWI). The TWI makes it possible to implement I^2C protocol to interconnect up to 127 individual devices. The nRF51228 has also 31 general purpose I/O pins, which should be more than enough for our purposes.
Nordic Semiconductor is not the only one offering a Bluetooth Smart SoC solution. Argenox Technologies[1] offers a guide explaining the most relevant Bluetooth SoC currently available on the market [4]. In this guide, besides the nRF51822, the following SoC are nominated as good candidates:

1. Texas Instruments CC2650
2. Dialog Semiconductor DA14580

The solution provided by Texas Instruments is discarded for the GonioTrainer because only Code Composer Studio and IAR's Embedded Workbench can be used for software development[5]. Both tools are not open source nor free. second option, from Dialog Semiconductor is a very attractive option, featuring the lowest current consumption of all the options mentioned on the guide. Unfortunately, the software development is based on µVision IDE from Keil[6], which is again not open source nor free. The nRF51822 results to be the best option for the GonioTrainer. Besides meeting the hardware specifications, it also provides open source and free software development kits.

3.2. Sensors

Dedicated sensors are used to measure the posture of the athlete. The goniometer must be equipped with the magnetic encoder (already specified by the client) and an Inertial Measurement Unit (IMU).

3.2.1. Joint angle

The client specified the use of the AS5600 (or AS5601) by AMS for the angle measurement, one is being sold as a contactless potentiometer and the other as a contactless encoder. The sensing part is the same for both. The AS5600 has a complementary analog or PWM (Pulse Width Modulation)

[1] Argenox Tecnologies is a company that provides costumers with design, manufacture, assembly, test and certification solutions for Bluetooth integration in their projects

output while the AS5601 has a push-button output [7][8]. The ICs have a 12 bit resolution, measuring 0 to 360 degrees and need to receive a supply voltage of either 3.3V or 5V with 6.5mA supply current. The sample rate is 6667kHz. The sensor supports random/sequential read via the Fast-mode Plus I^2C protocol (up to 1Mhz [9, p. 3]). The MCU only needs to read the values from the ANGLE register to obtain the current value at the moment of reading.

3.2.2. Inertial Measurement Unit (IMU)

The Inertial Measurement Unit(IMU) is needed to find the exact placing moment of the skate on ice. The IMU could also provide significant information when using the GonioTrainer in sports other than skating. According to the specifications, the GonioTrainer must be equipped with an IMU with the same specifications as of the iPhone 4S or later. The IMU of the iPhone 4S consists of accelerometer, gyroscope and magnetometer [10]. For the GonioTrainer, only the accelerometer and the gyroscope are of interest. An accelerometer is a transducer that measures the force exerted on a mass when subjected to an accelerating motion [11, p. 48]. A gyroscope is used to measure orientation and is based on the principle of conservation of angular momentum [12, p. 313]. A more detailed explanation about accelerometers and gyroscopes is beyond the scope of this thesis.
Unfortunately, the are no public available specifications of the IMU from Apple (manufacter of the iPhone). In order to chose the right IMU, the specifications were based on teardown websites, where the iPhone 4s is disassembled in order to identify its component parts. We used the information published by TechInsights. According to TechInsights, the iPhone 4S uses the LIS331DH accelerometer and the L3G4200D 3-axis gyroscope, both from STMicroelectronic [13]. The specification from both sensors can be found in table 3.1.

Device	scale	rate (Hz)	resolution(bits)
LIS331DH [14]	±2/±4/±8/±16g[2]	0.5 to 1000	16
L3G4200D [12]	250/500/2000 °/s	100/200/400/800	16

Table 3.1: Specifications of the accelerometer and gyroscope in the iPhone4S

For a more efficient use of the available space on the goniometer, an accelerometer and gyroscope on the same silicon die is desirable. This will lead to less physical connections and a more seamlessly integration. The search for this device resulted in the MPU6050 by InvenSense. The MPU6050 unit ”is the world’s first and only 6-axis MotionTracking devices designed for the low power, low cost, and high performance requirements of smartphones, tablets and wearable sensors”[15]. Some important specifications are listed next:

- 400kHz Fast Mode I2C for communicating with all registers
- Input Voltage: 2.3 - 3.4V
- Temperature Range: -40°C to +85°C
- Programmable interrupt system.
- 10,000g shock tolerant

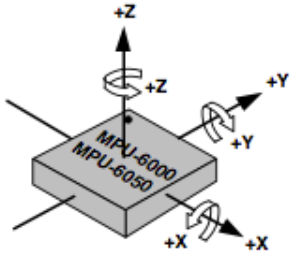

Figure 3.1: Orientation of Axes of Sensitivity and Polarity of Rotation [16]

Besides the raw acceleration and gyroscopic measurements, the programmable interrupt system makes the MPU an attractive choice. Each interrupt source may be enabled and disabled individually. The data ready interrupt is of special interest for timing purposes. The following three interrupts are of interest for the GonioTrainer: Free Fall, Motion Detection and Zero Motion Detection. Each interrupt is generated based on two parameters: *threshold* and *duration*. Duration is the time the threshold condition

[2] 1g = $9.8m/s^2$

MPU-6050

Device	scale	rate (Hz)	resolution(bits)	operating current(A)
accelerometer	±2/±4/±8/±16g	1000	16	500µ
gyroscope	250/500/1000/2000 °/s	4 to 8000	16	5m

Table 3.2: Specifications of the accelerometer and gyroscope of the MPU6050

must be valid before generating the interrupt. The threshold condition for the interrupts are explained in table 3.3. When a motion detection interrupt is generated, the Motion Detection Status register can be read. This register reports the axis and polarity of motion which generating the interrupt (see figure 3.2). For example when a Zero Motion is detected, this register will also indicate when the Zero Motion ceases to take place. The advantage of the interrupt system lies in decreasing the workload of the microcontroller. By setting the threshold and duration parameters, it should be possible to detect the placing moment of the skate. If it turns out that it is not possible to detect the placing moment solely using the interrupt system, a peak detection algorithm must be implemented in the microcontroller (see Appendix A).

+Z	-Z	+Y	-Y	+X	-X		Zero motion

Figure 3.2: 8 bit Motion Detection Status register. If an Motion Detection interrupt is generated, the 1's in this register will indicate the direction and orientation of the accelerometer that exceeded the threshold

Threshold	Description [17]
Free Fall	Detected when the absolute value of the accelerometer measurements for the three axes are each less than the detection threshold.
Motion	Detected when the absolute value of any of the accelerometer measurements exceeds the Motion detection threshold.
Zero Motion	Detected when the absolute value of the accelerometer measurements for the 3 axes are each less than the detection threshold. Unlike Free Fall or Motion detection, Zero Motion detection triggers an interrupt both when Zero Motion is first detected and when Zero Motion is no longer detected.

Table 3.3: The MPU-6050 has a programmable interrupt system which can generate an interrupt signal based on this three thresholds.

3.3. Data storage

Besides direct feedback, O'sports wants to analyse the posture of the rider *after* a training or a specific exercise. Before the analysis takes place, the measurements of the sensors need to be stored during the to-be-analysed race (data logging). In the near future, additional sensors might be added to the system. This issue will be also taken into consideration for the data logging.
Starting with one magnetic encoder which will provide a 12-bit word ($n = 1$), equation 3.1 can be used to calculate how much storage capacity is needed to store the raw data of the sensor with a sampling frequency of 100Hz. A top-athlete can train up to 3 hours (10800 seconds) continuously, so this value will be used as the sampling time. Filling in these values, one finds approximately 1,6 Mbyte[3] of memory storage is needed (for one sensor).

$$\text{Capacity} = \text{word length (bits)} \times \text{frequency (1/seconds)} \times \text{sampling time (s)} \times \text{n} \tag{3.1}$$

[3] 1 byte = 8 bits. 1kbyte = 1024 byte. 1 Mbyte = 1024 kbyte

In the search for a more lighter, efficient and smaller goniometer, data logging on the device is disadvantageous. If the storage of the data is done in the goniometer, this will slightly increase the weight and size, and impose some constrains on the energy consumption. In order to store the measurements, advantage of the smartphone can be taken. During the direct feedback, the smartphone will receive the measurements and determine if a stimulus is necessary or not. Because the smartphone is already receiving the measurements, it could also log the entire ride. An extra advantage of this method is that the goniometer will have one less physical port, making it easier to make it waterproof. The smartphone data logging will be discussed in the section 3.3.2. Although the idea of the GonioTrainer is to provide feedback during and after a training, how the athlete performed during a real race is also valuable information. During a race, the rider will not receive direct feedback nor will he/she carry a smartphone. The GonioTrainer will thus need on-board memory to record the data during the race. After the race, the information could be transferred to the smartphone and proceed as usual. This will be discussed on the section 3.3.1.

3.3.1. On-board

The goniometer must store data when the smartphone is not available. Once the smartphone is available, the goniometer can upload the data to the smartphone. Because it is not known when the smartphone will be available again, the data storage in the goniometer must be non-volatile. EEPROM (Electrical Erase Programmable Read Only Memory) and flash are the most known types of non volatile memory [18]. Flash is actually the name for a specific type of EEPROM. The difference lies in the fact that flash erases large blocks of memory bits in parallel. This makes the erase operation faster.

Flash memory by Spansion The 512 Mbit (64 Mbyte) MirrorBit® Flash Non-Volatile Memory from Spansion (S25FL512S) was chosen as the storage device in the goniometer. The amount of memory available in this device is more than twice as needed to store both the measurements from magnetic encoder and the IMU. Still, 64 Mbyte was chosen based on the multi-sport and the sensor extension idea. The most relevant specifications of the S25FL512S are listed next[19]:

- Serial Peripheral Interface (SPI)
- Read at 6.25 Mbytes/s
- 512-byte Page Programming buffer (1500 kbytes/s)
- 256-kbyte Logical Sector Erase (500 kbytes/s)
- 100,000 Program-Erase Cycles on any sector typical
- 20 Year Data Retention typical
- Core Supply Voltage: 2.7V to 3.6V
- Temperature Range: -40°C to +85°C
- Maximum current consumption of 16mA while reading, 100mA while programming or erasing.
- Standby current consumption of typically 70µA

The flash IC consists of a main flash array divided in 256 sectors, containing each 512 pages with 512 bytes. A 32-bit byte resolution address is used for the address space (see figure 3.3).

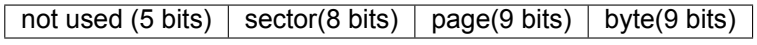

Figure 3.3: 32-bit address format of the flash memory.

It is possible to read and program the flash array one byte at the time, but it is only possible to erase per sector. Erasing a sector means setting all the bits in the sector to logic 1. In order to (write data) program the flash array, the data is first passed to a 512 byte buffer (the size of a page), so a maximum of a page of data can be programmed at the time. Programming means turning the logic 1's on an

erased byte to logic 0 when needed. Once a bit is set to 0, it can only be set to 1 by erasing the entire sector where the bit is located. To read from the flash array, it is only necessary to specify the start address, subsequently the device will output the data and automatically go to the next higher address in sequential order until stopped by the MCU, in Spansion words: ”The entire memory can therefore be read out with one single read instruction and address 000000h provided”[19, p. 90].

3.3.2. Off-board

During a training or exercise, the smartphone will process the data received from the sensors to determine if a stimulus is needed. Because the client is interested in analysing the performance after the ride, the smartphone will also record the measurements. The client showed interest in being able to access the recorded data through the internet. When a training session is finished, the smartphone will upload the measurements to a server, where the client can have easy access to the data. The communication between the goniometer and the smartphone will be through Bluetooth Smart and the smartphone will be running on the Android OS. Bluetooth Smart is only supported in Android version 4.3 or later [20]. The memory requirements for this version states that the smartphone must have at least 512MB of non-volatile storage available for application data [21]. Thus the smartphone used for the GonioTrainer will have enough storage capacity to record the measurements of a training. The requirements for the smartphone also state that it must include support for one or more forms of data networking (either WIFI or cellular network coverage). Transfer of the measured data to a server should be possible, this will be discussed in the next section. The smartphone will act as a buffer between the goniometer and the database. Once the data is uploaded to the database, it can be removed from the smartphone.

3.3.3. Data format

The client has been working with the comma separated format (CSV) format for the data processing. This format has widely been used for exchanging data between programs. There is no a formal specification, but the RFC 4180 standard defines that each record is separated by a line break and each field in a record is separated with commas. Common usage for the records is ASCII [22]. At the end of each training, the client will have easy access to a CSV file with the measurements ready to be analysed.

3.4. Software

The software part of this project can be split up into three parts: a Java application for user interaction, a website for easy access to the measurements and a database to store the information regarding each training session. The application on the smartphone can send a request to a PHP-script found on the server[23], which does all the processing work and registers all data in a MySQL (Structured Query Language) database. Eventually the file can be retrieved from a web page for extensive analysis of the training session or race. Retrieving the data measurements on a PC gives rise to the opportunity to do extensive calculations and graphical analysis.

3.4.1. Java application

The first version of the smartphone application will be developed for Android. Although Android is our focus, other operating systems are explored for similar file upload capabilities. From Android OS 2.2+ onwards file upload is supported, however, the GonioTrainer requires a minimum of Android 4.3 for the Bluetooth Smart link. iOS 6.0 and later, as well as Windows Phone 8.1 and higher are supporting operating systems [24], which makes the coverage on the smartphone industry practically completely for later development of the GonioTrainer.

Programming language Java is programming language used to develop Android applications. Java was created later than C and C++. It added improved functionality that lacked in both programming languages. Building applications in Java is especially advantageous because of the ease it has to import libraries. The Android SDK is a developer tool that helps building, testing and debugging apps. It contains many standard Java libraries (data structure libraries, math libraries, graphics libraries, networking libraries and everything else you could want) and some additional special Android libraries. Uploading a file requires one additional library not included in either the standard Android nor standard Java library: the *HttpComponents* library. HttpComponents is made available by Apache Software

Foundation [25]. HttpComponents provides several classes that can be used to accomplish Hypertext Transfer Protocol (HTTP) connections and stream data through it. HTTP is a communication protocol between a web client (Android device in our case) and the web server [26]. An HTTP request is a class that consists of HTTP style requests, line requests, method requests, URL requests, header field and body content. HTTP posting requests will be able to send the file from the smartphone to the server. The most common methods that are used by a client in an HTTP request are: GET, HEAD, POST, PUT, DELETE, TRACE and OPTIONS. These are included in the HttpComponents library and are named as: HttpGet, HttpPut, HttpHead, HttpPost, HttpTrace, HttpDelete, and HttpOptions [27]. HttpPost will be used to request for acceptance of the origin server to enclose the entity in a posting request and the actual posting the entity to the URL.

Developing environment Eclipse is a popular IDE (Integrated Development Environment) to develop Android applications. It's is open source and has many options to customise the environment and adding useful plug ins [28].

3.4.2. Website

O'Sports wants the measurements to be easy accessible at the end of each training session. No further specifications were given. Taking advantage of the smartphone Internet connection nowadays, a clever solution is to upload the data directly to an online server and retrieving them via a website. Most of the time, websites are build via the HyperText Markup Language (HTML). HTML is the most frequent markup language used to describe the structure of a website. HTML contains only plain text if no style is added. HTML files are hosted on at least one web server, which are only accessible via the Internet through an Uniform Resource Locator (URL). [29]
HTML is static and therefore needs to be combined with Hypertext Preprocessor (PHP) to meet the GonioTrainer's needs to update the website whenever files are added or deleted. PHP is an HTML-embedded server side scripting language, which gives web developers the opportunity to create dynamic web pages. The PHP-code is executed on the web server and afterwards its results are send to the user's computer to display it on the browser [30].

3.4.3. Database

Databases are digitally organised collections of data. Preferably all information regarding the GonioTrainer should be stored in a database. In a future, hundreds of users might use the GonioTrainer, so it is important that the data on the database is well-arranged. Every user should be able to request for a specific training session from himself/herself from any date, any time and/or any training session.
MySQL is the most widely used Relational DataBase Management System (RDBMS). It's capable of being accessed and adjusted by PHP language. PHP is able to connect to the MySQL server, run queries, and retrieve the results. SQL is the language used for managing data used in RDBMS. Queries are the main SQL statements capable of selecting, inserting, updating and deleting specific data [31].

4

Implementation

Figure 4.1 is a block diagram showing how the described components of this thesis are connected. The chosen joint angle sensor and IMU, as well as the external flash memory, will be driven by the the nRF51822 System on Chip (SoC). The nRF51822 may perform digital processing on the measurements before sending it via a Bluetooth link to the smartphone. The smartphone will do some further processing on the measurements in order to determine if a stimulus is necessary (how this is determined is beyond the scope of this thesis) as well as storing the raw data on an Internet server. The client or end-user will be able to retrieve the measurements though this server to perform a more detailed analysis on the data.

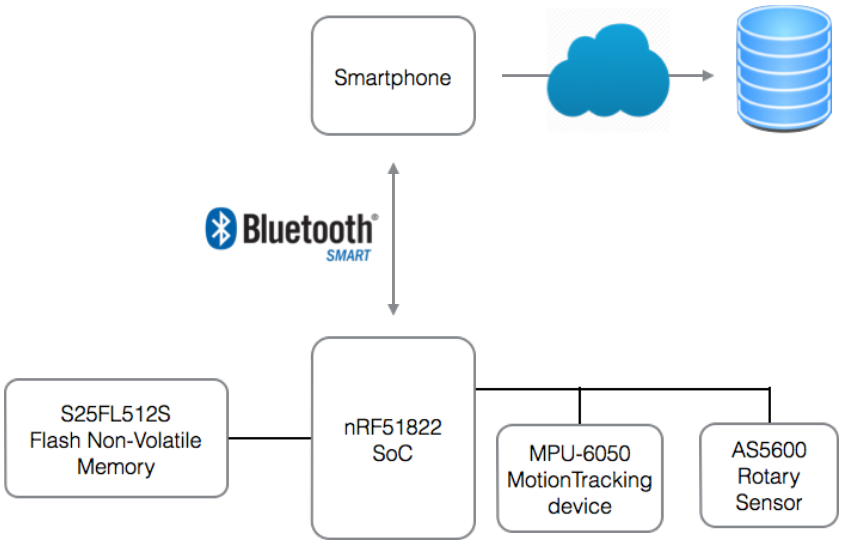

Figure 4.1: Block diagram showing the interconnection between the micro controller of the goniometer and its peripherals: memory, IMU, rotary sensor, smarthone and server (feedback module is not shown).

In this chapter a detailed description will be given about the implementation of the system. The starting point will be the nRF51822 SoC, which can be considered the heart of the whole system. The second section will explain how the sensors are controlled through the I^2C protocol. In the third section, SPI will be explained and how it is used to access the external memory, which is needed when a smartphone is not available. Last section of this chapter deals with the connection between the smartphone and a server for storing the measurements in the cloud. The reader should have a basic understanding of Object Oriented programming. For the implementation of the GonioTrainer smartphone application please refer to [32]. Figure 4.1 and the remainder of this chapter provide an abstract representation how the different components are connected. For a more detailed representation on pin layout, the reader is referred to appendix B.

4.1. Microcontroller

The RedBearLab nRF51882 development board was chosen to develop the system around the nRF51882 SoC (see figure 4.2). Nordic Semiconductor also provides their own development board, the nRF51 DK, but RedBearLab was chosen over Nordic Semiconductors because of its price and easy to acquire, while still providing all the features needed. The SoC may be also referred as *the host* throughout the remainder of this thesis.

There are three ways to program the nRF51882: using GCC with Nordic nRF51822 SDK, through the mbed platform developed by ARM and with an Arduino library for nRF51882 developed by RedBearLab themselves. The Nordic nRF51822 SKD provides an open source compiler free of charge where one can take full advantage of the SoC, the price for this method is its complexity compared with the other options. The Arduino library is an easy and simple way for people that are new to programming. Arduino did not provide some advanced options, so this method was soon discarded for this project. Using the Nordic SDK method, we could completely program and test the system ourselves with Eclipse, but due to lack of time and complexity, the system will be implemented using Mbed. Mbed is a platform developed by ARM which provides free software libraries and on-line tools for professional prototyping. This platform gives an extra layer of abstraction to program the system. The goniometer will be programmed in C++.

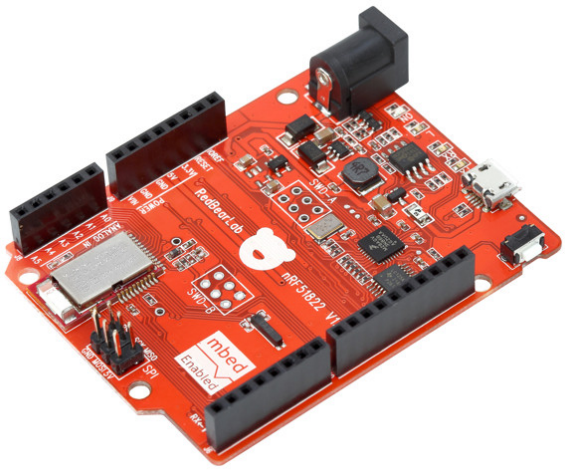

Figure 4.2: RedBearLab nRF51882 development board [33].

4.2. Sensors

Both sensors, the AS5601[1] and the MPU6050, will be controlled via the Inter-Integrated Circuit (I^2C) interface. The I^2C-bus is a bidirectional 2-wire, Serial Data (SDA) and Serial Clock (SCL), bus for efficient inter-IC control. The advantage of this protocol is that only two wires are used to communicate with more than one device. Each device has an unique address.

The single `Sensors` class was created to use both sensors. The class uses the I2C library provided by mbed. This library takes care of the signals between the nRF51822 and the sensors and with the functions `read()` and `write()`, it is possible to write or read bytes to and from the sensors. The important functions implemented by the `Sensors` class are described in table 4.1. When an instance of the class is created, both setup functions are called. The update functions ensures that the sensor measurements, at the moment, will be copied from the sensor registers to the nRF51822 register via I^2C. Once the update functions are executed, the get functions can be used to read the measurements of the sensors from nRF51822 memory.

The 12-bit sample from the magnetic encoder is found in two successive 8-bit registers. The first register contains the 4 most significant bits (MSB), and the second register the remaining 8 bits. The same applies for six 16-bit samples from the MPU6050 (one for each of the three axes of the accelerometer and gyroscope). Besides the 14 bytes of sensor measurements, two additional register from the MPU

[1] This variant was provided by the client, no particular choice was made between the AS5600 and the AS5601

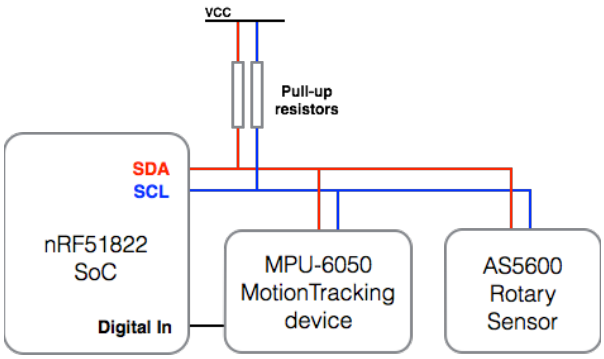

Figure 4.3: Representation of I^2C interconnections between sensors and nRF51822 SoC. The MPU6050 Int pin is connected to one of GPIO of the SoC

Function	Description
`setupAngle()`	The current angle measured by the AS560x will be set as the zero position
`setupIMU()`	Set the interrupt system, the range from the accelerometer and gyroscope and the sample frequency
`updateAngle()`	The two 8-byte registers containing the 12-bit angle value is read via I^2C and copied to the nRF51822 RAM
`updateIMU()`	The 12 8-byte registers containing the 16-bit accelerometer and gyroscopes values for each axis is read via I^2C and copied to the nRF51822 RAM
`getAngle()`	Returns the latest's updated angle measurement
`getIMU(n)`	Returns the latest n byte updated IMU measurement. The first 6 bytes corresponds to the accelerometer, and the remaining 6 from the gyroscope

Table 4.1: Functions for Angle and IMU read out. Each sensor has its own setup, update and get function.

are read. The registers contain information regarding the MPU interrupt system. The first byte indicates which interrupt was generated, and the second register is the Motion Detection Status (see figure 3.2). The maximum collection of data that can be obtained at a time from the GonioTrainer is 16 bytes.
The time it takes to copy the data from the sensors to the host registers depends on the SCL. For the GonioTrainer, the SCL is set to 400kHz, which is the maximum SCL supported by the MPU6050. Each data byte send via I^2C is accompanied with an acknowledgement bit [9]. Before we can read the data from the sensors register, the register 8 bit address need to be send from the host to the sensor, thus to read the values from one register byte, 17 bits are needed. To read data from successive registers, only the start address need to be specified. The time to read n successive registers can be calculated using equation 4.1. To obtain the whole collection of data previously described, 4 successive measurements are done: angle, IMU, interrupt status and motion status. Filling in the values in equation 4.1, the readout of the sensor will take approximately 0.5ms (theoretically).

$$\text{time(s)} = \frac{n \cdot 9 + 9}{\text{frequency(Hz)}} \tag{4.1}$$

4.2.1. Programmable Interrupt System

The MPU6050 is able to generate a *Data Ready* interrupt. The interrupt indicates that new data is available from the accelerometer and gyroscope register. It will be used to indicate the microcontroller to read (update) the new values. The interrupt is generated at the same sample rate as the gyroscope,

which can be set from 33 to 8000Hz. Using this interrupt, there is no need to implement extra timers on the microcontroller to read out the sensors. The *Free Fall*, *Motion*, *Zero Motion* interrupts can be used alone or in combination to detect a specific gesture of the athlete. When one of this interrupt will be generated was explained in the previous chapter.
For the specific case of identifying the positioning of the skate on the ice, a special peak detection implementation for speed skating was designed. The reader is referred to Appendix A.

4.3. Flash memory

The Spansion S25FL512S 512 Mbit (64 Mbyte) flash non-volatile memory unit will be used to store the measurements of the GonioTrainer when a smartphone is not available. The device connects to the nRF51822 (the host) via a Serial Peripheral Interface (SPI).
The device supports the traditional SPI single serial input and output as well as two bit and four bit serial commands. For our purposes, the single serial input and output option will be used to reduce the complexity of the system and keep the interconnections between devices as low as possible. A block representation is depicted in figure 4.4. Signals from the host to the memory device will be through the MOSI (Master Out/Slave In) link and signals from memory to the host will go through the MISO (Master In/Slave Out) link. The communication between the host and the memory device will be through 8-bit commands, these indicate the type of information transfer or operation to be performed. Before the communication between the host and the memory device can start, the host will drive CS (Chip Select) low. In order to terminate the communication, the host will drive CS high. CS is used to select the device on the SPI bus, this is needed when more than one SPI devices are connected to this bus. SCK is the Serial CLock and marks the transfer of each bit between the host and the S25FL512S.

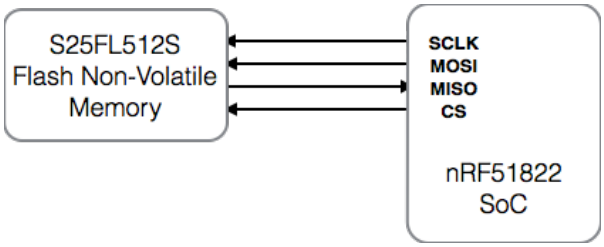

Figure 4.4: Block representation of SPI interconnections between flash memory and nRF51822 SoC.

To use the flash IC, the `Memory` class was created. Before the goniometer will start storing the measurements, the flash array needs to be erased for the new data. For this, the `bulkErase()` function was created. Erasing the entire 64Mbytes array can take up to 2 minutes [19, p. 4]. Therefore, a more efficient sector based erase method was chosen. The time it takes to erase the memory now depends on the amount of sectors used in the previous session. The new `sectorErase()` will check if the first register of the first sector is erased. If not, it will erase the entire sector and proceed to the next until, a sector is reached, where the first register has already been erased. Note that this implementation makes it possible to only store one measurement session at the time. When a write, read or erase operation is in progress, the *Write in Progress* bit in the Status Register will be set. This register can always be read to check if the device is busy. `isBusy()` returns true if this bit is set, otherwise false. This function is used during the erase operation to determine when the flash IC is done erasing and ready to be programmed.
Once erased, the flash IC will be used to store the measurements from the sensors with a desirable rate of 100Hz. This consists of the following steps:

1. reading the measurements from the sensors
2. passing them to the programming buffer
3. when buffer is full, recording the values to the main flash array.

It is necessary to ensure that these processes do not overlap. If so, the flash memory may be still writing data into the registers, while a new measurement is being taken and loaded into the flash buffer or main array. This will create a conflict, and timing issues may appear or run-time error could rise

up. With a sample rate of 100Hz, there is 10ms span between the start of each measurement. In the previous section has been encountered that the sensor read-out takes approximately 0.5ms. The time it takes to load a byte into the memory buffer depends on the chosen SPI clock, which was conveniently chosen to be the same rate as the I^2C clock. The time span between the end of loading into the buffer and the start of the next measurements is therefore approximately 9ms.
The flash buffer is 512 bytes long. Each data read out consists of 16 bytes, meaning that after 32 measurements (or 320ms), the buffer will be full and the data needs to be passed to the main flash array. The time to write the entire flash buffer into the registers, according to the data-sheet, is typically 0.3ms to a maximum of 0.75ms in extreme conditions (90°C). As a result, the previously calculated 9ms interval is more than enough to store the data from the buffer into the registers, or ”memory dump”. In order for this to work, a counter will be incremented on each period. After each 32th period, the counter will be reset and the data on the buffer will be recorded into the main array. Figure 4.5 shows is a time diagram for the reader to get an idea of how the data logging will take place. For convenience, the the time needed to copy the data from the buffer into the registers is taken to be 0,5ms. Starting from 0ms, the first memory dump will occur at 320ms, the second at 640ms and so on. From the figure it is also clear that no timing issues can occur between reading the sensors, loading into the buffer and finally dumping the data into the main flash array.

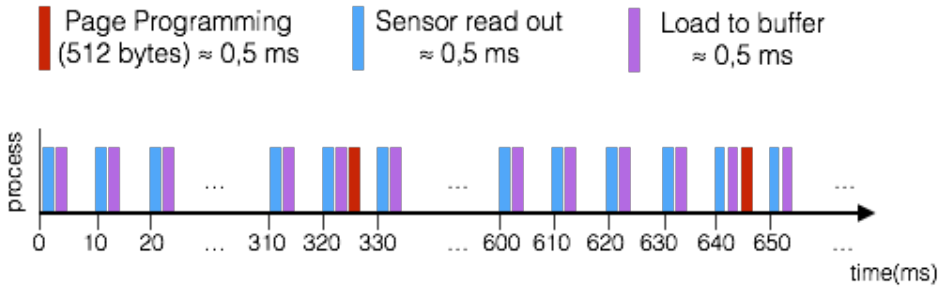

Figure 4.5: Timing diagram of the data logging. First the measurements from the sensors will be taken (red), and passed to the memory buffer (purple). When the buffer is full, the data will be ”dumped” to the main flash array (red). Figure not to the scale

The `pageProgram(address)` and `read(address)` functions are used to specify the starting address for the `write(data)` and `read()` functions. `write(data)` will copy the data byte into the buffer and `read()` will return the value of the register addressed with `pageProgram(address)`.

4.4. Control logic

The goniometer will be able to interact directly with the real world through a push button, which will generate an interrupt and drive a LED (Light Emitting Diode). After careful analysis of the GonioTrainer tasks, five different states were identified: idle, feedback, offline, read and off. Idle is the state at which the power is on or after a reset; then the goniometer is ready and waiting for instructions from the smartphone. During this state the smartphone can choose to set the goniometer into the remaining four states. In the *feedback* state, the goniometer will read the measurements of both sensors, do some data analysis and send the information in real time to the smartphone. The *offline* state is used when the athlete wants to log the training or competition only, without feedback. During this mode the athlete does not need to carry the smartphone (hence the name of this state) and the measurements will be stored in the flash IC. The *read* state is used to read the measurements from the sensors stored in the previous state and then transmitting it to the smartphone. The last state, *off*, will put the whole system in minimum current consumption for storage. To implement each state, an interface (or abstract class in C++ [34]) `State` was created. Each state will be derived from this interface. The `State` class consists of four pure virtual functions: `initialise()`, `execute()`, `pushButton()` and `exit()`. The class also contains functions accessible to the derived state classes to control the LED.
In order to select the right mode and to make sure these perform as intended, a controller, the `Controller` class, is created. The controller has a list with a reference to all the possible states and a variable indicating the current state. The controller will set the current mode and call their functions based on the inputs of the goniometer: the Bluetooth connection, the push button and the interrupt generated by the MPU6050. When a new state is selected, the `exit()` function of the previous state

	`initialise()`	`execute()`	`pushButton()`	`exit()`
`IdleState`	N/A	N/A	N/A	N/A
`FeedbackState`	N/A	Send measurements to smartphone	Start reading sensors	Stop reading sensors
`OfflineState`	N/A	write measurements to flash IC	Erase flash memory and wait for button to be pushed	Clear flags
`ReadState`	N/A	N/A	Pass on data from flash to smartphone	N/A
`OffState`	Set everything in minimun current consumption	N/A	N/A	N/A

Table 4.2: For each state, the controller can call four different functions. Each function has a different task depending on the current state, which is described on this table

is called. Thereafter, the `initialise()` function of the new state will be called. When the push button is pressed, the `pushButton()` function will the executed, and when the MPU6050 generates an interrupt, `execute()` will run. See table 4.2 for a description of what the state functions do during a particular state. The previous implementation is based on the **strategy pattern**, a software design pattern that allows to change the behaviour of the program at runtime [35, p. 349]. An UML class diagram of the goniotrainer is depicted in figure 4.6. It shows the relation between the different classes. Note also that the `GonioService` class was included. This class is created by Rene Miedema and Ingo Schilken for the Bluetooth communication.
When the nRF51822 is powered on (or reset), it will create an `Controller` object. After its creation, the controller will initialise the sensors, create the different derived states objects and select the *idle* state as default state. Once in the *idle* state, the controller will wait for the Bluetooth connection to select a different state and call its functions based on the push button and MPU6050 interrupt. Interrupts[2] can make things easy, but special care need to be taken. Spending to much time in an interrupt could cause timing problems and even run-time errors. Interrupts can form large queues when not used properly, and prevent the system from returning to the main function.
The `Controller` is implemented to be the goniometer's highest level of abstraction. It should be considered as a black box. The input is the mode selector and a push button, and the output are the sensor measurements and a LED (see figure 4.7). The black box model provides a way to integrate the system described in this thesis together with the communication system developed by Ingo Schilken and Rene Miedema. The LED is used to indicate the status of the goniometer.

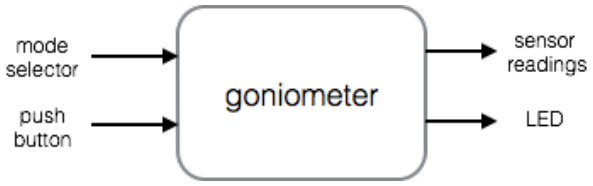

Figure 4.7: Input/output goniometer

4.5. Cloud service

Succeeding the sensors read-out, conveying the data over Bluetooth to the smartphone and storing it on a file created by the smartphone, the data will be uploaded to a server. This gives the user the ease to easily download the measurements and perform extensive analysis on his/her training sessions. Figure 4.8 depicts a simple flow-diargram for uploading the data, from the client side to the server side.

[2] An interrupt is a digital signal into a CPU that indicates some event has happened [36]

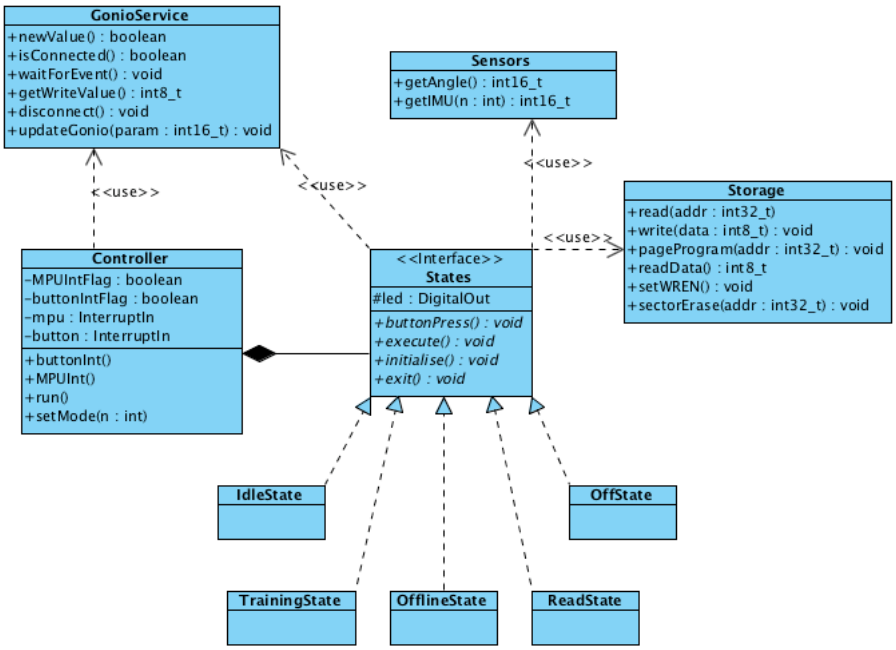

Figure 4.6: UML class diagram of the goniometer (mbed library not shown)

The smartphone implements the client side, creating the data file and setting up a connection with the server. On the server side, a PHP script ensures that the file is copied on the server.

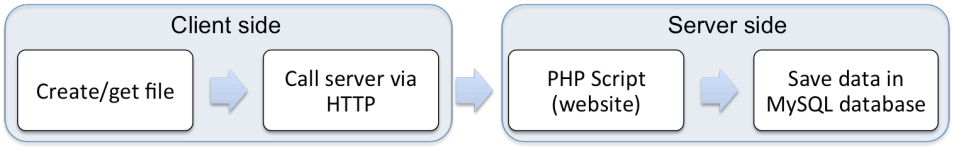

Figure 4.8: File upload split up into a client side and server side

4.5.1. Client side

On the client side, an Android application is run, which starts the process of uploading. An instance of the (`serverTask`) class is created, which is a subclass of `Asynctask` that allows to perform background operations and at the same time execute tasks on the main threat[3] (User Interface threat). `Asynctask` automatically invokes four asynchronous steps when it is executed: `onPreExecute()`, `doInBackground()`, `onProgressUpdate()` and `onPostExecute()` [38]. Figure 4.9 visualises the implementation of subclassing the Java application.

`onPreExecute()` is invoked on the UI thread before a task is executed. Right after the previous function is finished, `doInBackground()` is executed. `onProgressUpdate()` is only invoked when `publishProgress(Progress...)` is called, so this function is optional and not used on the GonioTrainer since the text files are relatively small (at most a few Mbytes for long measurements). Finally `onPostExecute()` is invoked immediately after the background computation finishes.

`AsyncTask` is an abstract class provided by Android and is inevitable for tasks like file and network accessing. As requests are send out to the server, response will be awaited. Android is a service with single thread modes, so in case of waiting for response equals to non-responsibility of the screen. In case of long operations in the UI thread, the screen is thus non-responses for a long time. Android won't even let the smartphones connect to internet without the use of `Asynctask` and returns the Main Thread Exception error. This is also the reason `AsyncTask` is used for uploading the file to the server,

[3] A thread is a concurrent unit of execution[37]

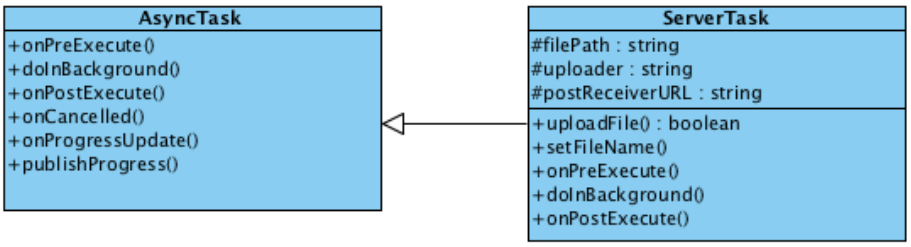

Figure 4.9: ServerTask is a subclass of AsyncTask. ServerTask makes it possible to upload a local file to an internet server.

so that the UI threat stays active.
`uploadFile()` is the function that is invoked in `doInBackground()` and is responsible for uploading the file to the server. `uploadFile()` uses a small library called HttpComponents, which is explained in section 6.3.1. Figure 4.9 shows that an instance of ServerTask contains the following three variables:

1. *filePath* defines the path of the file on the smartphone.
2. *uploader* defines the user that uploads the data file.
3. *postReceiverURL* is the URL to which the file is going to be posted.

Those three strings are arguments of the function `uploadFile()`.
After establishing connection with the use of two classes from *HttpComponents* library (`HttpClient` and `HttpPost`), data and additional information (time, date and name of uploader) is ready to be posted. First a MultipartEntity needs to be set up, so it has a request entity suitable for an HTTP multipart POST method. This creates our to be send data bundled and thus allows us to send multiple parts at once. Once bundled, the data is send to the server using *HttpPost* and a response from the server is received if an error has occurred.
A pseudo code of `uploadFile` is shown in algorithm 1. This is written such that it could be implemented in other programming languages, especially for future development of the iOS application [39]. *AFNetworking* is a method that works on similar basis for iOS applications using HTTP post requests.

Algorithm 1 Uploading file

$httpclient =$ new DefaultHttpClient
$httppost =$ new HttpPost
$entity =$ new Multipartentity
Create new File from filePath
Add file and additional information to entity
Post $entity$
Execute $httppost$
Receiving responses

4.5.2. Server side

The server side consists of a website segment and a database segment.

Website

The server side responds to a posting request using a PHP script, which executes and stores the information that was send on the client side. The PHP script can be divided into three parts:

1. Initialising variables.
2. Checking whether the file path exists in the server storage.
3. Storing the file and concurrently saving its name and appended information in a MySQL database.

Initialising variables At first the file folder, where the file is ought to be stored, is defined. After which the file name of the requested file is gathered and combined to a new file path. This is the path where the file can be eventually found.

Checking file path This is where we get to point at which the existence of the file path is checked. This is done by using the existing PHP function: `file_exists()`. A while-loop will go through this function, as the return value is true. In the while-loop the file is split up in parts using `pathinfo()`, which is able to return the directory name, basename, filename and extension. This enables us to set up a counter and walk through the file paths and rename the file as long as it exists. This is how a eventually an unique file path is brought forth.

Storing file As a unique file path is found, the file is ready to be stored. This is done through a standard function defined in PHP: `move_uploaded_file()`. If the file is stored succesfully, the address of the file and the supplemented information is then inserted into a MySQL database using a SQL query. The address, user, time and date of transmitting can be found in the MySQL database. `Date_default_timezone_set()`, `date()` and `time()` are standard defined PHP functions that are able to return the current date and time. This opens up the possibility to search for specific training sessions on the website using database queries.

Database

Organising data into overviewable tables is the key to a well-ordered database. Since the GonioTrainer doesn't retrieve heaps of data only one table is created (see figure 4.10): *SessionList*.

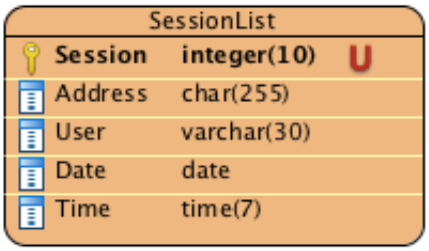

Figure 4.10: Table *SessionList* and its field values

Session is an an integer-type unique field that keeps record of the sessions that have been recorded. It is defined with a Primary Key that serves as a unique ID and can easily be used to request for specific rows in the table. *Address* is utilised as string that refers to the path where the specific training session is stored. The string is captured in the MySQL database as char of length 255. The user that has uploaded the file is recorded as *User* in the database. The field has type varchar and has a maximum length of 30. *Date* and *Time* are two fields that represent the specific moment of uploading the file.

5

Ethics

Behind every engineering project, the same question always arises: is the project actually ethical? According to the Dutch ”3m-responsibility” model [40], the engineer is responsible for the well-being of the people (*Mens*), the society (*Maatschappij*) and the environment (*Milieu*). The engineer should always take these three elements into consideration during the project, taking care that no element is left behind or neglected. When the three elements are in balance, the project is said to be sustainable, and therefore ethical.

5.1. People

The GonioTrainer has the ability to measure, collect and analyse data and behaviour from a human being and upload this information on the Internet. With this point of view, the GonioTrainer can be considered an element of the ”Internet of Things” (IoT). The IoT is a term formulated by Kevin Ashton in 1999 [41]. Although the term is more than 15 years old, it started to gain popularity the last couple of years. The idea is not only that people can interact with the Internet, but ”things” can too. Nowadays, the number of new connected devices (except smartphones) could reach to 8 billion per year, which results in 28 billion connected devices in 2020 [42]. These products will collect a vast amount of data and make them available to the manufacturer. With all this information, the manufacture has a better way to understand the customer. Understanding the user means that he/she will receive a product that better meets his/her needs. All in all, it may sound great that in the future products will better suit our needs, but this comes with the cost of privacy.
The end user may not be aware that information about him is being recorded and passed on. With the information gathered by the client, gender, age and health may be deduced. It is the responsibility of O’Sports to make this information not public nor sell it to third parties.

5.2. Society

The GonioTrainer is a new concept. People may dislike the idea that in future, traditional coaching will be replaced by smartphones. If every athlete is to be trained the same way, sports may lose their charm. O’Sports made clear that the GonioTrainer does not intent to replace the coach, but aid him. Not everyone might believe these words, which could lead to the fact that the GonioTrainer will eventually face the same situation the Clap skates faced when introduced. Not everyone believed that it could significantly improve the performance during a race, which let to big discussions. It took some years, but nowadays, the Clap skate is almost a must-have for the experienced skater[43]. The GonioTrainer has the advantage that it is not intended to be used during competition, which could make the acceptance more easy and avoid starting controversies.

The GonioTrainer has the potential to increase the enthusiasm of the ”normal” people about sports. The idea to train with a professional coach is the dream of many. With the GonioTrainer, this dream can come true. Planning more practises with real-time feedback and having the availability to analyse your own performance between the training sessions is a great way to stay in shape and avoid injuries. This could lead to a more healthy society, currently dominated by the fast food chains [44].

5.3. Environment

When developing a product, the designer must not only regard the function of the product, but also think about the impact it could have on the environment, during and *after* its life span. Devices using rechargeable batteries, like the GonioTrainer, should take special care of the batteries when these are replaced or disposed, as they can case several damage to the environment [45]. A battery recycling program is a great way to avoid environmental contamination.

6

Results

A frame that can be attached with two straps on the legs of a test person was used to test the implementation design explained in chapter 4 . On this frame, the sensors were mounted and connected to the microcontroller through flexible cables (see figure 6.1). With this frame, a working prototype of the goniometer can be realised and tested.

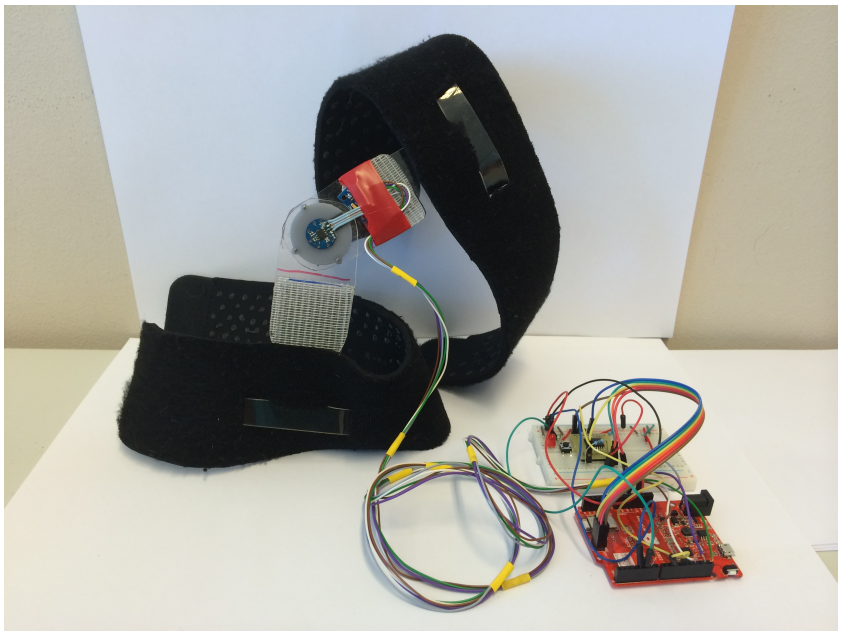

Figure 6.1: Test settings consisting of the frame and the microcontroller and its peripherals

6.1. Sensors

The serial output of the RedBearLab nRF51822 was used to pass on the measurements of the sensors to a computer. A serial port terminal application on the computer made it possible to print the measurements on the computer screen. With this method, it was straightforward to certify the measurements from the sensors. Figure 6.2 shows a screen shot of the sensor read-out on a flat table. The acceleration of approximately 1g on the z-axis (see figure 3.2.2) of the accelerometer corresponds with the acceleration exerted by the Earth on every object due to gravity. Printing the measurements on the computer screen, also verified that the *Data Ready* interrupt works as intended and the sensor read out can be done without fault at 100Hz.

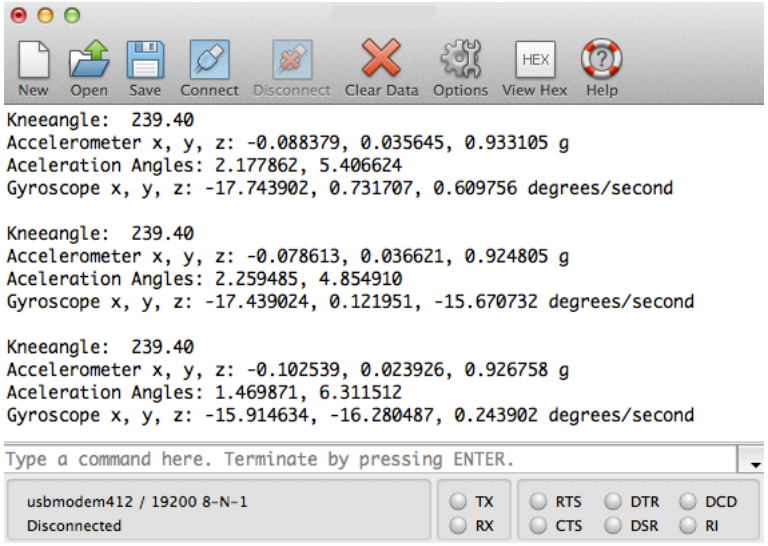

Figure 6.2: Screenshot of the sensors read out on a flat table. The acceleration angles correspond to the angle the z-axis of the MPU is making with respect to the Earth's surface.

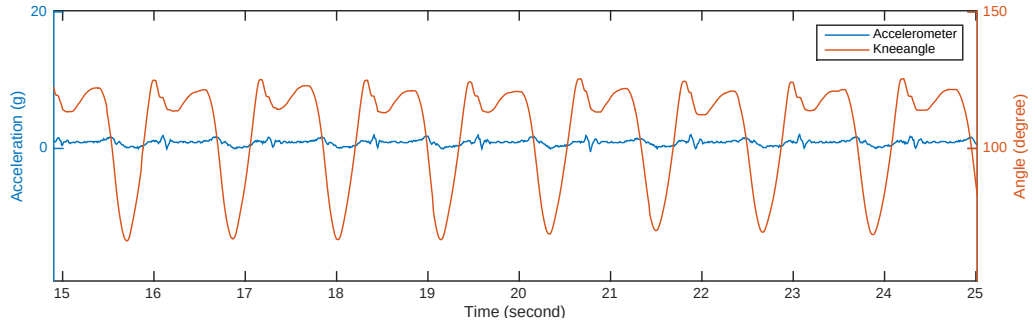

Figure 6.4: Sample from the field test, showing the knee angle during a walk and one axis the acceleromenter.

6.2. Flash memory

Extending the sensor testing, after each sensor read-out the measurements were loaded into the flash array as discussed in section 4.3. An oscilloscope was used to verify the correct behaviour of the data logging. Figure 6.4 demonstrates that our timing prediction in figure 4.5 holds. The yellow signals corresponds to the *Data Ready* interrupt, indicating that the microcontroller is updating the sensor values at that time. After 32 periods, the memory buffer is full and a data dump will take place. Right after CS (blue signal) goes high, the the data dump will take place. Afterwards, the memory buffer will be ready for the next period. As soon as the next Data Ready Interrupt is generated, a small gap of approximately 1ms is observed (sensor read out via I^2C), then CS goes low indicating that the buffer is being filled again.

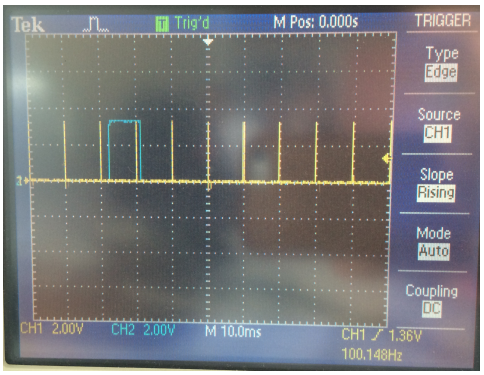

Figure 6.3: Data logging in progress. Yellow signal is the *Data Ready* interrupt, generated at a 100Hz rate. The blue signal is CS, indicating that the memory buffer is being filled.

A field test was done with the prototype, which consisted of walking and recording the measurements. The measurements were downloaded to MATLAB afterwards via the serial port. Figure 6.4 shows a sample of the measurements. The test was done to find out whether the recorded data was meaningful

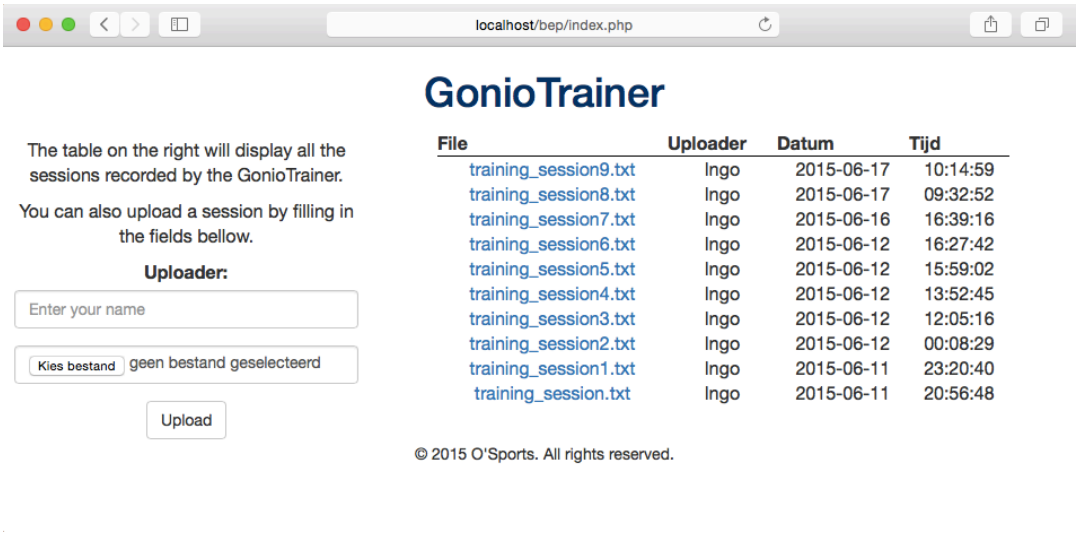

Figure 6.6: Screenshot of the web page.

and to ensure that the system was robust enough for real world testing.

6.3. Cloud service

Cloud services has been tested using the smartphone and the Internet. Both section 6.3.2 and 6.3.1 below show how the results were retrieved on the website.

6.3.1. Android application

A simple Android application was created, containing only one button triggering the upload of a local file on the smartphone to the server (see Figure 6.5). The `startUploading(View view)` function is activated starting a new session. At the same time, a text file is created by `WriteFile()`. After the file has been created, the instance *ServerTask* is created and executed (see section 4.5.1) that is responsible for the file upload.

Figure 6.5: Screenshot of a simple Java application for uploading a file

6.3.2. Website

The web page (see figure 6.6) shows the uploaded files. For testing purposes, a file can also be uploaded from the site, in order to show that the server and PHP scripts worked without the use of a smartphone.
The table shown on the right shows the data retrieved from the database in chronological order with the newest entry showing up first. The entries on the table demonstrates that the file uploading from a smartphone to a webserver works as desired.

6.4. Integration

The `Controller` class in section 4.4 was tested via the serial input. It was possible to choose between the different states of the goniometer from a keyboard. The whole system described in this thesis is now ready to be tested with the Bluetooth communication system from Ingo Schilken and Rene Miedema. The total size of the implementation is 49.5kbytes, far from the maximum of 256kbytes allowed. The first test consisted of setting the goniometer in the desired state. The LED indicated that this worked. Next data logging was tested. The goniometer was successfully set in the *Offline* state. With the *Read* state, the

measurements stored on the flash IC can be transferred to the smartphone. The measurements could be retrieved on the website. At this point, the integration is a success. The last test is the real-time read out of the sensors. As soon as the test started, the whole system crashed. Timing was double checked by both groups, but no timing conflicts were found. For the Bluetooth communication between the goniometer and the smartphone, the S130 SoftDevice was used on the nRF51822[32]. The SoftDevice ”integrates a low energy controller and host, and provides a full and flexible API for building Bluetooth low energy System on Chip (SoC) solutions” [46]. On a later stage, it was discovered that the SoftDevice restricts the access to the Programmable Peripheral Interconnect(PPI). The PPI ”enables different peripherals to interact autonomously with each other using tasks and events without use of the CPU” [47]. This could be the reason that the measurements could not be passed on to the smartphone in real-time mode. Our suspicions where verified by different posts in the forums from Mbed and Nordic Semiconductors. A conflict exists when using the TWI peripheral and the SoftDevice at the same time.

7

Conclusion

The aim of this project was to develop a functional prototype of the GonioTrainer that satisfies the specified design requirements. Chapter 6 describes the results of the tests on the requirements. This section compares these test results with the requirements, and verifies if those are met.

7.1. Evaluation of requirements

7.1.1. Costs

The costs for components in the electrical system (excluding casing) are calculated for respectively a batch of 10 and 100 pieces:

1. 10 pieces - €86.76. Requirement: €150

2. 100 pieces - €67.01. Requirement: €60.

Details of the costs for the specific components can be found in tables C.1 and C.2. The batch for 10 pieces has met the requirements easily. However, the amount for 100 pieces exceeds the requirement by €7.01.
For the batch of 100 pieces, the requirement was not met. However, the memory is the only component in our part that has been ascertained by the client, that is the only component which could be cut down in costs. As the memory size is larger than specified, it could be decreased to lower the costs. Replacing the Spansion S25FL12S 512 Mbit (64 Mbyte)by the Spansion S25FL256S 512 (32 Mbyte) Mbit reduces the costs for a batch of 100 pieces by €1.68. This won't make any sense in achieving the requirement.

7.1.2. Sample rate

Reading out sensors has been successfully done at a sample rate of 100Hz. The requirements state that desired frequency has to been set to a minimum of 60 Hz. This requirement of data acquisition has thus been met.

7.1.3. Data storage

As the results show in figures 6.3 and 6.4 data storage has been implemented successfully using SPI. The Spansion S25FL512S 512 Mbit (64 Mbyte) flash non-volatile memory unit can store sensor data (both the magnetic decoder and the IMU) up to 11 hours. This complies with the requiring 3 hours of data logging. The measurements are easily accessible via a cloud based server. The file is uploaded by the smartphone after each training session and data is stored in a database.

7.1.4. Peak detection

Motion Detection Interrupt is able to give an interrupt successfully in a multi-sport scope. The custom-made peak detection has been implemented with success for speed skaters specifically on the micro-controller, therefore meeting the specifications.

7.1.5. Physical ports

Only one button and one LED were used to implement the goniometer's control logic. The physical ports were therefore kept as few as possible.

7.1.6. Integration

The state of the goniometer can be set via an Android application. The data logging without the smartphone is also possible. The data from the flash IC can be transferred via Bluetooth with success. The real-time sensors read-out failed. Both the Bluetooth Smart connection and I^2C interface didn't experience any problems on a stand-alone basis. However, as the results point out in section 6.4, using Bluetooth Smart and the I^2C interface created a conflict.

8

Discussion

This chapter discusses recommendations meant to improve the behaviour of the GonioTrainer.

8.1. AS5601 push button

The current system uses a physical push button. In future, the AS5601 push button function can be used to reduce the number of physical ports, making the GonioTrainer more resistant to water.

8.2. Programmable Interrupt System

The Programmable Interrupt System could not have been fully tested. A ”dry-skateboard” would have been a perfect way to obtain measurements and find the right parameters.

8.3. Cloud service

The website and database could be improved by adding extra functionality. To secure the athlete’s performances, log in service helps establishing privacy for athletes. The user should have a clear overview on their training sessions and official competition matches separately. Graphical analysis is another feature that is of real importance for the realisation of the final product, as well as the ability to export data elsewhere. Furthermore, the user might want to look for specific sessions in the past, the use of SQL queries is definitely very useful for achieving this. The database could also be expended by user-specific information. The ability to compare performances with the world’s best might also be a feature that would attract a lot of sportsmen.

8.4. Real time sensor read out

The TWI peripheral should be accessible via special functions provided by the SoftDevice API. Once this problem is solved, the GonioTrainer will be completed.

A

Peak detection using the ARM® Cortex-M0

A custom-made peak detection algorithm has been designed for speed skaters that finds the exact placement on ice moment. This appendix provides some background information about the skating technique and how this has lead to the design and implementation of the algorithm in C++ with MATLAB.

A.1. Background

Detection of the peak requires specific information on the technique of the athlete. Research has been carried out by O'Sports to define the knee angle in the different stroke in a race: start stroke, straightaway stroke and the corner stroke. and what impact the knee angle could have on the athlete's performance [2]. It claims that a lower centre of gravity of the speed skater results in less drag (smaller frontal surface) and longer push off distance in the stroke. Due to the longer push off distance, the skater is able to transfer more energy and thus be faster.
Figure A.1 shows a measurement period of two strokes of the knee angle. Different moments can be distinguished in a stroke:

- **PI** is the moment of placement of the skate
- **SP** is where the gliding phase starts
- **EP** represents the end of the push-off.

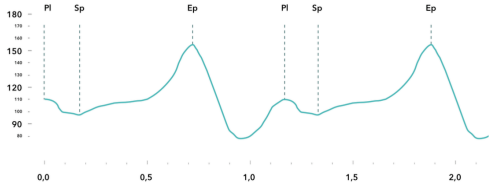

Figure A.1: Knee angle in two straightaway strokes [2].

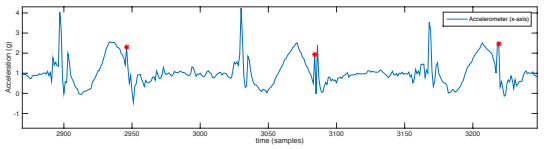

Figure A.2: Acceleration (x-axis) in straightaway strokes.

The moment where stimulus must be given is between the placement of the skate and the start of the gliding phase. The minimum in knee angle that determines *SP* can not be recognised in every speed skater. That's why O'Sports wants to use data from the accelerometer. Figure A.2 shows the accelerometer data in the x-direction of the skater (sideways). The red asterisk shows the peak to be determined. The data preceding the peak is relatively constant, which is because the skate is in air. As the skate touches the ice, the first peak is caused (red asterisk). Now the peak has been detected, an signal should be send to the smartphone indicating that a feedback can be given (if necessary).

A.2. Design

The previous background information is required to find the right peak. The client provided real measurements of the magnetic encoder and the accelerometer of a speed skater. These measurements

were imported to MATLAB to do the processing part and finding particular peaks for that specific speed skater. However, the client pointed out that these measurements are not certain for every speed skater and thus needed a robust algorithm.
With the knowledge that exists for the skating technique specific peaks can always be determined, the algorithm could be designed sturdy. The end of the the push-off (global maximum of one single strike), the valley (global minimum) after and the peak (local maximum) thereafter are detectable for every speed skater. The MATLAB function `findpeaks()` is used to detect every useful peak to finally detect the two local maximum and minimum marked by the red asterisks found in figure A.3. Input arguments give the opportunity to remove unwanted peaks from desired peaks. Three input arguments have been used [48]:

- *minPeakHeight*, returns only peaks higher than a predetermined value.
- *minPeakDistance*, returns only those peaks that are at least the predetermined number of samples apart from each other.
- *Threshold*, used to pick out those peaks that have a minimum height difference as defined by 'Threshold'.

The two marking points assign the interval in which the accelerometer data should be analysed and detect the first peak that is gathered. The black triangle indicates the peak to be found.

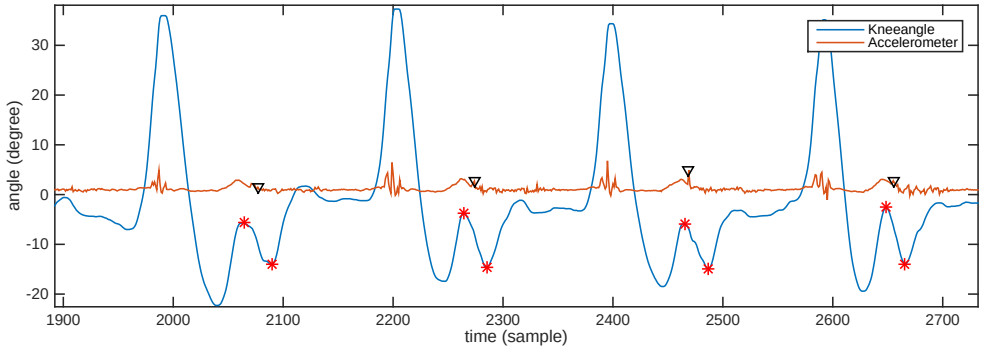

Figure A.3: `MATLAB` simulating the peak detection algorithm.

A.3. Implementation

The algorithm is implemented as a C++ Class, such that it could be executed on the ARM® Cortex-M0 microcontroller. The *Peak* class is created and consists of the attributes found in figure A.4. The input are the measurements from the magnetic decoder and the x-axis of the accelerometer. The following functions are declared in the class:

- `PeakDetection()` is the constructor required for an object to be created.
- `setValues()` is a function that initialises all parameters. Threshold, minPeakHeight and minPeakDistance are defined here for every peak. These parameters are adjustable such that the algorithm can be optimised for every individual.
- `saveAngle()` and `saveAccelerometer()` are two buffers that saves the last three consecutive measurements. Its functionality equals the functionality of a shift register, which shifts the measurements by one position in the array. The newest value added and the latest value is removed from the array.
- Both functions `findMaximum()` and `findMinimum()` are used to detect local minima and maxima. The input arguments (specified in `setValues()`) could be utilised depending on which specific peak is to be found. By checking the three consecutive values stored in the array and taking the input arguments into considerations, the local minima and maxima can be found.

- `update()` runs the algorithm. It detects the interval period in which the accelerometer data is analysed for the placement peak. Since ice skating is a sequence of repeated periodic movements, the end of the push-off (highest peak) can be used to indicate the starting point of one period. Subsequently the minimum is found. Respectively the maximum and minimum afterwards indicate the interval period. As the minimum is not detectable for every individual, a maximum interval is set to 20 samples.
- `isPeak()` returns a boolean at the moment when the peak is found.

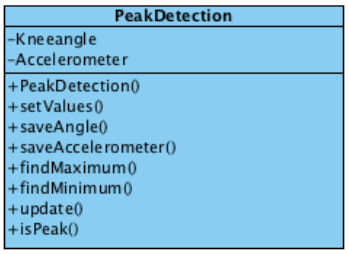

Figure A.4: Class *Peak* and its attributes and functions

A.4. Results

To test the algorithm, one period of the provided knee angle data was chosen, conditioned and imported to the 33220A Fuction/Arbitrary Waveform Generator [49]. This arbitrary waveform simulates the measurements taken from the AS5601. The results are depicted in figure A.5. The yellow signal is the simulated knee angle. The blue signal is the digital output of the microcontroller. It is the correct interval in which the accelerometer data should be analysed. The algorithm proved to be robust up to 3.5 Hz, which is by far larger than the maximum stroke frequency for speed skaters.

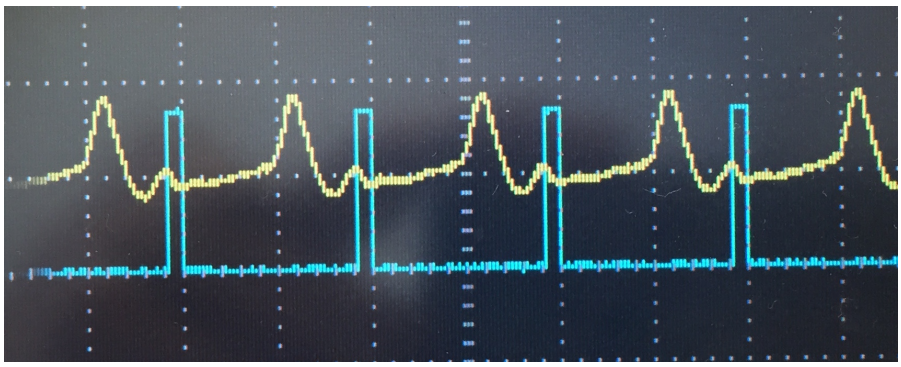

Figure A.5: Results of interval found using peak detection algorithm.

B

Pin layout

B.1. RedBearLab nRF51228

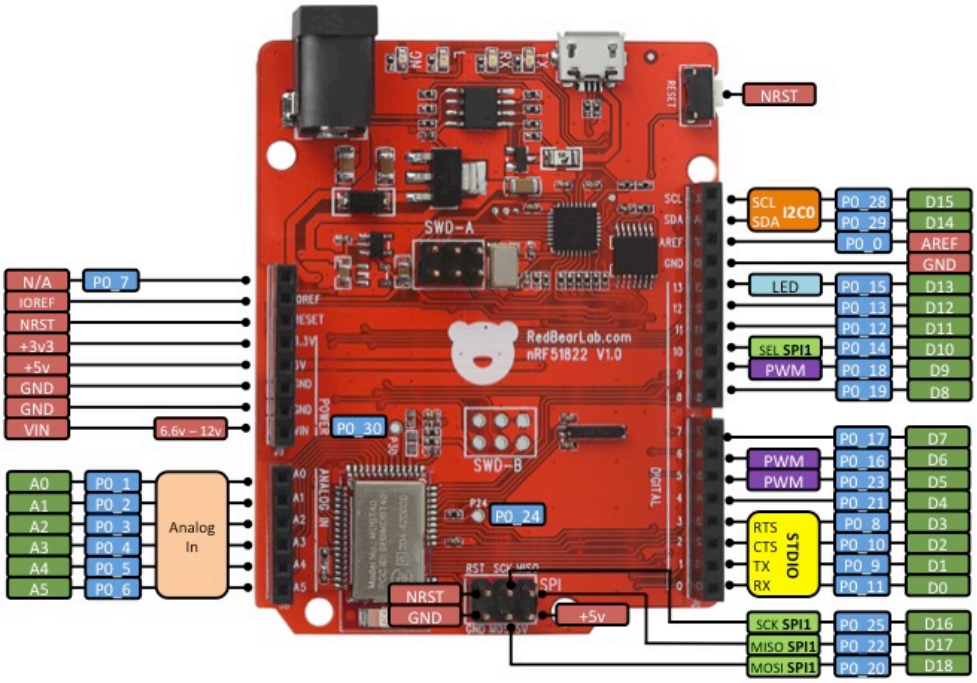

Figure B.1: Pin layout RedBearLab nRF51228[50]

B.2. AS5600 12-bit Programmable Contactless Potentiometer

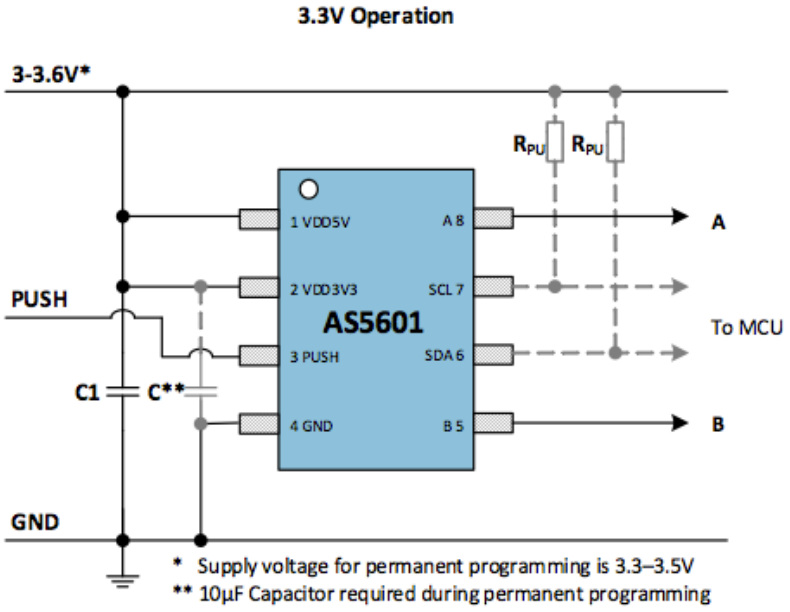

Figure B.2: Schematic of AS5601 Contactless Potentiometer[7]

B.3. MPU-6050 Board GY-521

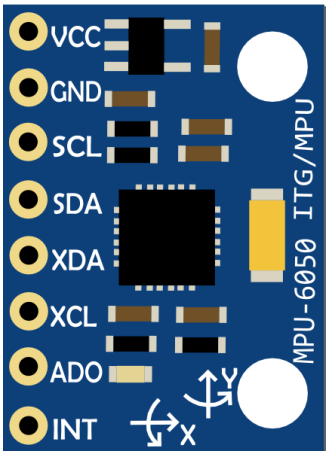

Figure B.3: GY-521 MPU-6050 breakout board [51]

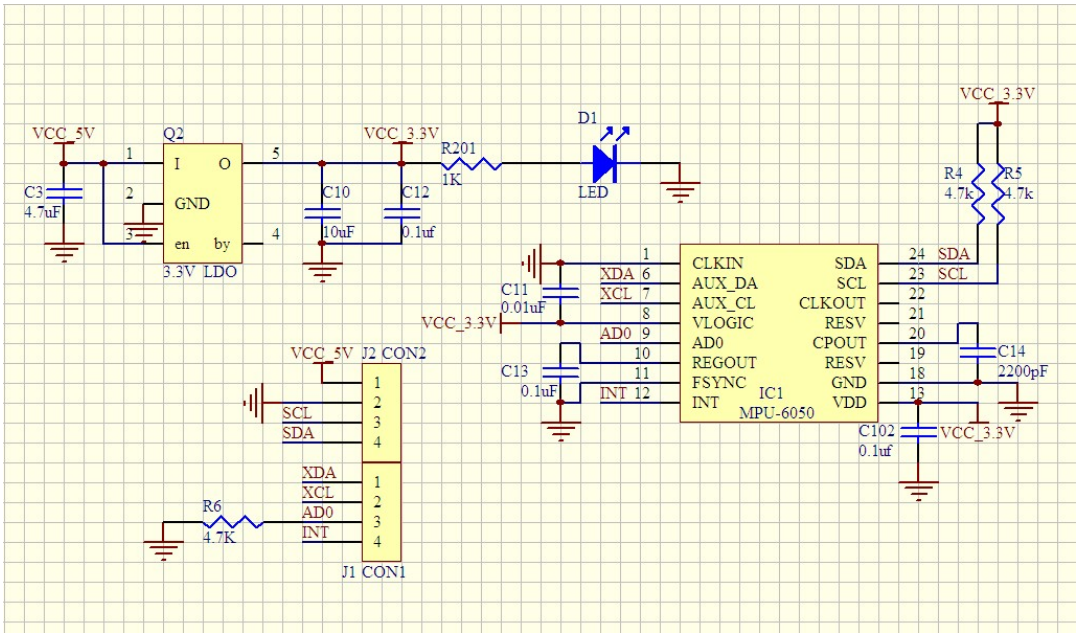

Figure B.4: Schematic of MPU6050 (GY-521 Breakout)[51]

B.4. Spansion S25FL512S 64 Mbyte Flash Non-Volatile Memory

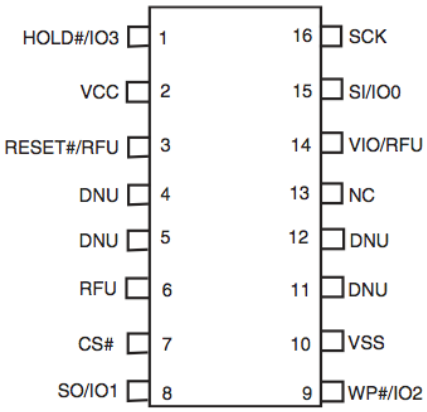

Figure B.5: Pin layout Spansion flash memory[19].

C

Costs

Below the costs are defined for a batch of 10 and 100 pieces respectively.

Table C.1: Costs for a batch of ten GonioTrainers

Component	Quantity	Price (euro)	Amount	Subtotal	Store
Driver	10+	3,7	1,00	3,7	Farnell
Charger	10+	0,479	2,00	0,958	Farnell
DC-DC	10+	6,92	2,00	13,84	Farnell
Bat1(300)	10+	14,9	1,00	14,9	Farnell
Bat2(165)	10+	14,39	1,00	14,39	Farnell
ISP	10+	9,8	2,00	19,6	Texim-Europe
Flash	10+	6,03	1,00	6,03	Farnell
IMU	10+	5,87	1,00	5,87	Invensense
Magnetic decoder	10+	2,44	1,00	2,44	Mouser
Vibration motor	10+	6,77	1,00	6,77	Precision Microdrives
			Total:	88,498	

Table C.2: Costs for a batch of one hundred GonioTrainers

Component	Quantity	Price (euro)	Amount	Subtotal	Store
Driver	100+	2,53	1,00	2,53	Farnell
Charger	100+	0,396	2,00	0,792	Farnell
DC-DC	100+	3,56	2,00	7,12	Farnell
Bat1(300)	100+	10,79	1,00	10,79	Farnell
Bat2(165)	100+	9,8	1,00	9,8	Farnell
ISP	100+	8,33	2,00	16,66	Texim-Europe
Flash	100+	5,42	1,00	5,42	Farnell
IMU	100+	5,59	1,00	5,59	Invensense
Hoeksensor	100+	2,01	1,00	2,01	Mouser
Motor	100+	6,3	1,00	6,3	Precision Microdrives
			Total:	67,012	

Bibliography

[1] L. Bardal. Measurement of frontal area of athletes in wind tunnel experiments. https://www.ntnu.no/documents/11601816/b830b9bd-d256-42c4-9dfc-5726c0ae3596, 2009.

[2] O'Sports. Feedback system for speed skating.

[3] B. Bosma, M.C. Scoop. *Bachelor of Science Thesis - GonioTrainer: Power supply and the Feedback Module*. 2015.

[4] Argenox Technologies. A guide to selecting a bluetooth chipset. http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-guide-to-selecting-a-bluetooth-chipset/, 2015.

[5] Texas Instruments Incorporated. Cc2650 development kit. http://www.ti.com/tool/cc2650dk, 2015.

[6] Dialog Semiconductor. Smartbond™ development tools. http://www.dialog-semiconductor.com/products/bluetooth-smart/smartbond-development-tools, .

[7] AMS. As5601 datasheet: 12-bit programmable contactless encoder, October 2014.

[8] AMS. As5600 datasheet: 12-bit programmable contactless potentiometer, October 2014.

[9] NXPSemiconductorsN.V. I2c-bus specification and user manual. http://www.nxp.com/documents/user_manual/UM10204.pdf, April 2014.

[10] Apple Inc. iphone 4s tech specs. https://www.apple.com/lae/iphone-4s/specs/, .

[11] P. Regtien. *Modern Electronic Measuring Systems*. Delft University Press, 1978.

[12] STMicroelectronics. Mems motion sensor: ultra-stable three-axis digital output gyroscope. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00265057.pdff, .

[13] TechInsights Inc. Apple iphone 4 teardown. http://www.techinsights.com/reports-and-subscriptions/outlook-and-analysis/apple-iphone-4, .

[14] STMicroelectronics. Mems digital output motion sensor ultra low-power high performance 3-axes "nano" accelerometer. http://www.st.com/web/en/resource/technical/document/datasheet/CD00213470.pdf, .

[15] InvenSense Inc. Mpu-6000 and mpu-6050 register map and descriptions revision 4.0. http://store.invensense.com/Datasheets/invensense/RM-MPU-6000A.pdf, .

[16] InvenSense Inc. Mpu-6000 and mpu-6050 product specification revision 3.4. http://www.farnell.com/datasheets/1788002.pdf, 2013.

[17] J. Rowberg. Mpu-6050 6-axis accelerometer/gyroscope. http://www.i2cdevlib.com/devices/mpu6050#registers.

[18] *Introduction to Flash Memory*, volume 91, April 2003. IEEE.

[19] Spansion Inc. S25fl512s 512 mbit (64 mbyte) mirrorbit® flash non-volatile memory cmos 3.0 volt core with versatile i/o serial peripheral interface with multi-i/o. http://www.spansion.com/Support/Datasheets/S25FS512S_00.pdf, .

[20] Android Developers. Bluetooth low energy. http://developer.android.com/guide/topics/connectivity/bluetooth-le.html.

[21] Google Inc. Android 4.3 compatibility definition. http://static.googleusercontent.com/media/source.android.com/en//compatibility/4.3/android-4.3-cdd.pdf, July 2013.

[22] Y. Shafranovich. Common format and mime type for comma-separated values (csv) files. http://tools.ietf.org/html/rfc4180, October 2015.

[23] Stanford University. Tutorial on client-server communications. https://web.stanford.edu/class/ee368/Android/Tutorial-3-Server-Client-Communication-for-Android.pdf.

[24] V. Salminen. File upload support on mobile. http://viljamis.com/blog/2012/file-upload-support-on-mobile/ year = 2002 .

[25] Apache Software Foundation. Apache httpcomponents. https://hc.apache.org/, .

[26] W3C. Hypertext transfer protocol architecture domain. http://tools.ietf.org/html/rfc7230.

[27] O. Kalnichevski, J. Moore, J. van Gurp. Httpclient tutorial. https://hc.apache.org/httpcomponents-client-ga/tutorial/pdf/httpclient-tutorial.pdf.

[28] The Eclipse Foundation. Eclipse ide for java developers. https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunar, .

[29] Html and url. http://www.w3.org/TR/WD-html40-970917/htmlweb.html.

[30] Php manual. http://php.net/manual/en/.

[31] Oracle. The world's most popular open source database. http://www.oracle.com/us/products/mysql/overview/index.html.

[32] I. Schilken, R.D. Miedema. *Bachelor of Science Thesis - GonioTrainer voor Sport: Communicatie en Applicatiedesign*. 2015.

[33] Red Bear Company Limited. Bluetooth low energy boards based on nordic nrf51822. http://redbearlab.com/nrf51822, .

[34] cplusplus.com. Polymorphism. http://www.cplusplus.com/doc/tutorial/polymorphism/.

[35] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[36] S. Rosenthal. Interrupts might seem basic, but many programmers still avoid them. http://www.sltf.com/articles/pein/pein9505.htm, 1995.

[37] Android. Thread. http://developer.android.com/reference/java/lang/Thread.html, .

[38] Android. Asynctask. http://developer.android.com/reference/android/os/AsyncTask.html, .

[39] Alamofire Software Foundation. Afnetworking. http://afnetworking.com, .

[40] L.Royakkers. *Ethiek techniek*. ThiemeMeulenhoff bv, 2014.

[41] K. Ashton. That 'internet of things' thing. *FRID Journal*, 2009.

[42] Dialog Semiconductor. Internet of things: Market overview. http://www.thefaktory.com/wp-content/uploads/2015/01/IoT-market-overview-Final.pdf, .

[43] J. de Koning. Slapskate history and background. http://www.sportsci.org/news/news9703/slapxtra.htm.

[44] J Currie, S. DellaVigna, E. Moretti, V. Pathania. The effect of fast food restaurants on obesity. http://eml.berkeley.edu/~sdellavi/wp/fastfoodJan09.pdf.

[45] D. H. P. Kang, M. Chen , O. A. Ogunseitan. Potential environmental and human health impacts of rechargeable lithium batteries in electronic waste. http://pubs.acs.org/doi/ipdf/10.1021/es400614y.

[46] Nordic Semiconductor. S130 softdevice specification. https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822,.

[47] Nordic Semiconductors. nrf51822. https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51822.

[48] MathWorks Inc. findpeaks. http://nl.mathworks.com/help/signal/ref/findpeaks.html?refresh=true,.

[49] Agilent Technologies Inc. Agilent 33220a20 mhz waveform generator. http://cp.literature.agilent.com/litweb/pdf/33220-90002.pdf,.

[50] Red Bear Company Limited. Redbearlab nrf51822 pinout. http://redbearlab.com/redbearlab-nrf51822/,.

[51] Arduino. Mpu-6050 accelerometer + gyro. http://playground.arduino.cc/Main/MPU-6050.