Applying Embedded Systems and Sensor Technologies to Trampoline Gymnastics

Diogo Ferreira Tribolet de Abreu

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

June 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

During the entire process of this thesis, many people gave me good advice and helped me make it as good as possible.

First of all, I thank both my supervisors for guiding me the whole way and not letting me ever loose focus on the final goal. Professor Carlos Almeida who was responsible for giving a first good impression of embedded systems and sensor technologies, not to mention his incredible attention for detail while revising this thesis. Professor Rui Rocha aided me significantly on the more practical side, developing hardware and software, and provided me with a great working space and resources in the Tagus Park campus.

Also I thank Sporting Clube de Portugal for letting me use their facilities to test my prototype. To all the coaches and athletes, specifically Tiago Duarte (coach), Luis Santos (coach), Tiago Costa (athlete), Pedro Ferreira (athlete), Diogo Ganchinho (athlete), Duarte Fernandes (athlete), Tiago Lopes (athlete) and Ricardo Santos (athlete) for taking the time to test the system and to give their honest opinion on ways to improve it.

To all my friends and colleagues, mostly those from Instituto Superior Técnico, not only for helping during this thesis but also throughout the whole course.

A very special thanks to my girlfriend Mariana Raposo who put up with me through this entire journey and tirelessly helped me to write this thesis. Without her it would have been a lot harder.

A final thanks to all my family, especially to my parents Margarida and Miguel Abreu, for providing and teaching me everything I know and love, Trampoline Gymnastics and Engineering. Without them I wouldn't be the person I am today.

Abstract

At the highest level of sports, the difference between ending in first or second place can be very small. This is particularly true for the sport of Trampoline Gymnastics, where jumping a little higher and more centered on the trampoline can have a huge impact in the final score. By utilizing Sporting Technologies (STs) in training, coaches and athletes can identify hidden problems and develop new training methods, which would be impossible otherwise. In the case of Trampoline Gymnastics, two parameters were chosen to be measured: the Time of Flight (TOF) and the Horizontal Displacement (HD).

At the time that this work was started, not much research had been done in this area and most of it implicated alterations to the athlete's and/or the trampoline's behavior. Also, there were no commercial products available that measured both parameters.

The designed architecture uses a grid of infrared beams (transmitters and receivers) under the trampoline bed that wirelessly send the TOF and HD to be stored in a database. A program with a Graphical User Interface (GUI) was created to control the system and view the data. Based on this, a functioning prototype was developed and tested in a "real-world" scenario.

Experiments were conducted in Sporting Clube de Portugal, where national team level athletes and coaches tried the system. The general consensus was that the system provided enough accuracy for measuring both the TOF and HD, being a helpful tool to improve their performance.

Keywords: Sporting Technologies, Trampoline Gymnastics, Time of Flight, Horizontal Displacement, Infrared Beam Module Transmitter, Infrared Beam Module Receiver

Resumo

Ao mais alto nível do desporto, a diferença entre acabar em primeiro ou segundo pode ser muito reduzida. Isto é especialmente verdade na Ginástica de Trampolins, onde saltar ligeiramente mais alto e mais no centro do trampolim pode ter um enorme impacto na nota final. Ao utilizar Tecnologias de Desporto (TD) durante o treino, treinadores e atletas podem identificar problemas e desenvolver novos métodos de treino, o que de outra forma seria impossível. No caso da Ginástica de Trampolins, dois parâmetros foram escolhidos para serem medidos: o *Time of Flight (TOF)* e *Horizontal Displacement (HD)*.

Quando esta tese foi iniciada, não existia muita investigação feita nesta área e na maioria dos casos implicava alterações no comportamento do atleta e/ou trampolim. Para além disso, não estavam comercializados nenhuns produtos que medissem ambos os parâmetros.

A arquitectura desenvolvida usa uma grelha de infravermelhos (transmissores e recetores) posicionada debaixo da cama do trampolim que envia sem fios o *TOF* e o *HD* para serem guardados numa base de dados. Um programa com uma interface gráfica foi criado para controlar o sistema e visualizar os dados. Baseado nisto, um protótipo funcional foi criado e testado num cenário real.

Foram feitas experiências no Sporting Clube de Portugal, onde atletas e treinadores da selecção nacional testaram o protótipo. A opinião geral foi que o sistema tinha precisão suficente para medir o *TOF* e o *HD* de forma a ser uma ferramenta útil para melhorar os seus desempenhos.

Palavras-Chave: Tecnologias de Desporto, Ginástica de Trampolins, *Time of Flight*, *Horizontal Displacement*, Módulo Transmissor de Infravermelho, Módulo Recetor de Infravermelho

Contents

List of Tables

List of Figures

List of Listings

Acronyms

AJAX Asynchronous JavaScript And XML.

AP Access Point.

APS Access Point Server.

ATS AirTime Trampoline System.

BOM Bill of Materials.

CAN Controller Area Network.

CSS Cascading Style Sheets.

CTC Mode Clear Timer on Compare Match Mode. *Glossary:* CTC.

FGP Federação de Ginástica de Portugal.

FIG Fédération Internationale de Gymnastique.

GC Grid Controller.

GPIO General Purpose Input and Output.

GUI Graphical User Interface.

HD Horizontal Displacement. *Glossary:* HD.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

I2C Inter-Integrated Circuit.

IMUs Inertial Measurement Units.

IR Infrared.

IR-GRID Infrared Beam Grid.

IRM Infrared Beam Module.

IRM-Rx Infrared Beam Module Receiver.

IRM-Tx Infrared Beam Module Transmitter.

ISR Interrupt Service Routine.

LAN	Local Area Network.
LED	Light Emitting Diode.
LPM	Low Power Mode.
MOSFET	Metal Oxide Semiconductor Field Effect Transistor.
PCB	Printed Circuit Board.
PIR Sensors	Pyroelectric Infrared Sensor.
PLA	Polylactic Acid.
RPi	Raspberry Pi.
RS-485	RS-485.
SoC	System on Chip.
SQL	Structured Query Language.
STs	Sporting Technologies.
SYN	Synchro Score. *Glossary:* SYN.
TD	Tecnologias de Desporto.
TMD	Time Measurement Device.
TOB	Time on Bed. *Glossary:* TOB.
TOF	Time of Flight. *Glossary:* TOF.
UI	User Interface.
URL	Uniform Resource Locator.
USCI	Universal Serial Communication Interface.
UT	User Terminal.
WebApp	Web Application. *Glossary:* WebApp.
WSGI	Web Server Gateway Interface.

Glossary

CTC Clear Timer on Compare Match (CTC) Mode refers to a mode of operation of Timer0 in the ATTINY85 microcontroller. In this mode the counter starts at zero and increments until it reaches a certain user-defined value. When it reaches the desired value it automatically resets the counter to zero and starts incrementing again.

Execution is a measure of the overall perfectionism of an athlete's routine. It refers to the straightness of the legs, torso, "travelling", etc..

HD Horizontal Displacement is a measure of the location where an athlete lands on the trampoline bed after each skill. There are 11 different areas on the trampoline bed (see Figure 1.2) and depending on which of them the athlete lands on after each jump a deduction of 0.0 (A5), 0.1 (A4 and A6), 0.2 (A1, A3, A7 and A9) or 0.3 (A0, A2, A8 and A10) points is given. The sum of all deductions is then subtracted to an initial score of 10.0 points resulting in the HD score.

SCL is the serial clock line in the I2C protocol.

SDA is the serial data line in the I2C protocol.

SYN Synchro Score is a component of the final score of a Synchronized Trampoline Routine. It refers to the degree at which a pair of athletes (jumping in seperate trampolines) fall on their trampoline bed at the same time, after each skill.

Synchronized Trampoline Routine is a Trampoline Routine executed by two athletes side-by-side on two seperate trampolines. The final score includes a parameter called Synchro Score.

TOB Time on Bed is the measured time that an athlete spends on the trampoline bed before taking off and performing a new jump in the air..

TOF Time of Flight is the measured time that an athlete spends in the air. This value does not count with the contact time spent on the trampoline bed. Typical values of Senior Men range between 17.000 and 20.000 seconds for a full trampoline routine [18].

Trampoline Routine is a sequence of 10 consecutive jumps/skills executed on the trampoline by an athlete.

WebApp is a type of computer program based on the client-server structure. The client part is normally a web browser that is used to make requests to the server. The server contains all the code of the Web Application and responds to the client's requests. It has a front-end that contains the User Interface part of the program and a back-end which contains the databases and the code to manage them. Examples of Web Applications are webmail, online stores, messaging services and many others.

1

Introduction

Contents

Nowadays, more and more technologies are being used in the sports world. These technologies are applied in various aspects of sports: some devices are used in competition to help make judging the sport less subjective and more accurate (e.g. Hawk-Eye system in tennis [21]), and others are being used during training to help coaches and athletes achieve better results more efficiently (e.g. Babolat Play [45]). The University of Ulster defines Sporting Technologies (STs) as "...man-made means developed to reach human interests or goals in or relating to a particular sport ..." [54].

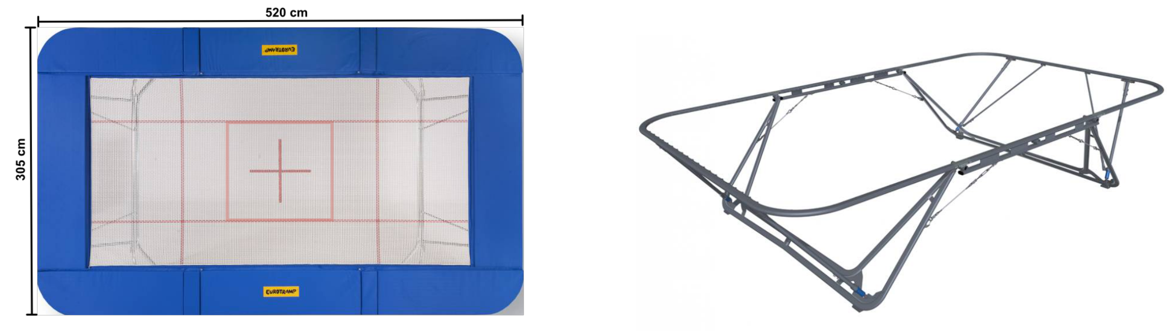
In general, the main purpose of STs during training is to gather data about the athletes performance. This data can then be used by the coaches to improve or develop new training methods, aiming to increase the athletes efficiency and results. In sports that use specific equipment, like cycling and tennis, this data can be analyzed and used to develop better materials or structures and improve existing instruments. Another important aspect of these technologies is that they must be "invisible" to the athletes and cannot change in any way the conditions that exist during a competition. As a paper on STs by Pervasive Computing explained, "...enhancing players so that they want to use sensors in their training, enabling unobtrusive instrumentation so that coaches can analyze the best data available ..." [9].

Trampoline Gymnastics is a competitive sport that has gained much momentum since it was introduced in the Olympic Games in Sydney, 2000. Trampoline Gymnastics competitions consist of an athlete performing two trampoline routines during the preliminaries round (where the total score is the sum of two separate scores for each routine), and one routine in the finals (one final score). Trampoline Routines consist of 10 different skills/jumps performed in a sequence, one after the other. Trampoline Gymnastics has been continuously evolving and in the last years has verified the implementation of various STs. The main components of an Olympic competition trampoline are: a metallic frame structure, as shown in Figure 1.1b, a set of metallic springs that are attached to the inner perimeter of the trampoline frame, and an elastic bed, as shown in Figure 1.1a. The elastic and flexibility of the springs and bed and the rigidity of the metallic frame are the main characteristics that define how a trampoline behaves when an athlete is jumping on it. This is basically the way the forces are transfered from an athlete falling on the trampoline, to the floor where the trampoline is placed. As referenced before, the properties cannot be altered in any way by any STs that want to be used.

In 2010, a new rule was created adding a new variable when scoring a trampoline routine, the Time of Flight (TOF). This new component refers to the measured time that a trampolining athlete spends in the air. This value does not count with the contact time spent on the trampoline bed. To measure the TOF, devices were created that can be attached to the trampoline and differentiate between when an athlete is in the air and when she/he touches the bed. These machines are used during competitions and in training to help athletes understand whether they should work on jumping higher or not.

Another important component of a trampoline routine is the Horizontal Displacement (HD). This parameter refers to the location on the bed where an athlete lands after each skill. A trampoline bed, as shown in Figure 1.1a, has several areas bounded by red lines. The greater the number of jumps inside the center square, the higher the score. The HD used to be included in the Execution part of the routine score [18], but since 2017 it is a completely separate component, as defined in the 2017-2020 Trampoline Code of Points [17]. Right now the HD is is measured by the judges using video cameras, but very recently there has been an attempt to measure this component with a machine (non-official and not Fédération Internationale de Gymnastique (FIG) approved).

These two score components are very important because they can be measured in a very objective way, eliminating human error. This makes them perfect candidates to be measured and analyzed using STs.



(a) Trampoline bed [14]. (b) Trampoline frame [15].

Figure 1.1: Olympic Eurotramp Trampoline Structure

1.1 Motivation

At the moment, the only STs for Trampoline Gymnastics that are commercially available are for measuring TOF. These systems have many problems that are holding back a wide spread of their usage during the training of the gymnasts.

The first problem is pricing: these machines can cost between €900 [2] and €1400 [52]. Most clubs cannot afford a system at these prices because, unfortunately, Trampoline Gymnastics is not a big revenue generating sport yet. Moreover, most clubs have other sports, besides Trampoline Gymnastics, that get more investment and are more well-known. In Portugal, only the Federação de Ginástica de Portugal (FGP) and one other club have these TOF systems for training and for competition.

The second issue is related to utilization of the system. These devices can be very difficult to mount on the trampoline and normally are not very sturdy due to their ineffective attaching system. To work correctly, the machine relies on precise alignment of the sensors so any impact on the trampoline might incapacitate the system (this happens very often during competition and training). Also the User Interface (UI) of both systems is not very user friendly and intuitive to use.

The final problem, and probably the most important one, is the utility of the machine. These systems were designed mainly for the competition, so their output is merely quantitative (final score), even though the main purpose of STs is to help coaches and athletes to improve their performance in training. To fix this, these systems must deliver, in a better way, more processed data that can be relevant for both coaches and athletes.

There is currently no machine to measure the HD of a trampoline gymnast commercially available. However, the documents and regulation needed by the FIG are being created to allow these systems to be used in competitions.

1.2 Goals

The main objective of this thesis is to create a functioning prototype of a system that can accurately and precisely measure the TOF and HD of a trampoline routine.

The trampoline system must be able to detect, with millisecond resolution (as defined by the FIG in the 2017-2020 Trampoline Code of Points [17]), the time when an athlete lands and leaves the bed. These two moments, in each skill, are essential to be able to calculate the final TOF of the routine, since one is only interested in the time the athlete spends in the air. This system must also be able to determine where the athlete has landed on the bed with some degree of accuracy. Even though there are no official FIG minimum requirements yet, it should detect in which areas, highlighted in Figure 1.2, the athlete lands on after each skill.

The Graphical User Interface (GUI) should be intuitive and easy to understand for athletes and

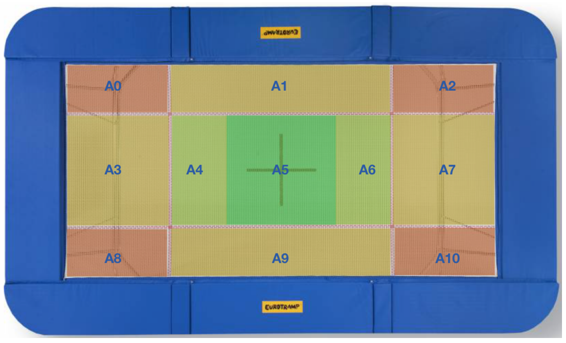


Figure 1.2: Trampoline bed with different scoring areas highlighted (based on [14])

coaches to use during training. Timing information for each jump will be recorded and in the end of a routine the total TOF is displayed to the user. The location where a gymnast lands on the bed, in each jump, will also be determined and displayed. The User will manually signal the system to start/stop measuring a given routine. A simple database will also be provided to save athlete profiles and routine (TOF and HD) information.

When finished, the system should be cheaper than the other devices for measuring TOF. It should also be non-invasive to the trampoline, meaning that, in no way, can it alter any variables influencing the jumpers performance. A trampoline has two main components: a moving one, which is comprised of the springs and bed, and a static one, which is comprised of the metallic frame, shown in Figure 1.1b. The most important part that defines "how a trampoline jumps" has to do with the flexibility/elastic properties of the springs and the bed. This means that the system should be mounted on the static part of the trampoline, the metallic frame, minimizing its influence on the athlete's performance. Mounting the machine must be very easy and it must be robust enough to withstand normal day-to-day use of the trampoline.

The final result will be a system that is attached easily to the trampoline and will "feed" the coach real-time data (after each routine) about TOF and HD of the athlete.

Figure 1.3 shows a general architecture of the final system. The main blocks are: a sensor system mounted on the trampoline that detects when and where the gymnast lands on the bed; a database that receives the data from the sensor system and stores it; a user terminal that accesses the database and displays the information in a graphical manner. The User Terminal is also used to give the commands to start or stop the routine measurements.

Data communication between the different blocks of the system should be wireless to avoid the usage of long electric wires in gyms. The system should also be able to function without Internet connection, meaning that all data storage and communication should be done locally in the gym.

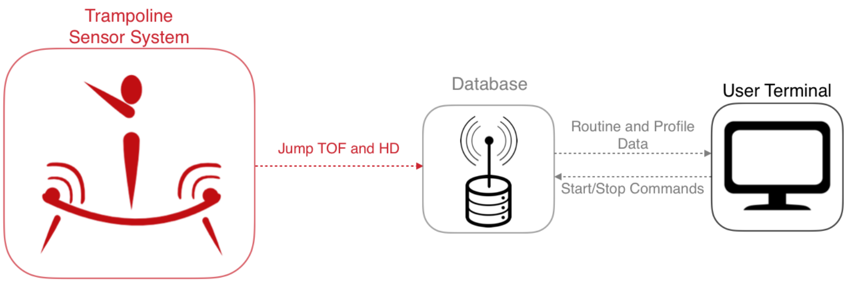


Figure 1.3: General Diagram of System Architecture

1.3 Challenges

The two main challenges that this project faces are: guaranteeing the resolution needed for the TOF part and developing a new kind of sensor(s) that can calculate where the athlete has landed on the trampoline after each skill.

The first problem has to do with the speed at which the sensor(s) can detect if a person has contacted or left the trampoline bed. This means that the sensor's measurements must be accurate to the true value of TOF in milliseconds (as referred above in Section 1.2). To address this, the performance of the sensors will have to be carefully analyzed and eventual computation and data transmission times will also have to be considered. Another issue, related to the timing, is guaranteeing that the processing of data and all final results should be finished and accessible right after the routine has ended.

The second problem is related to the lack of systems that can measure and localize surface deformations with the timing constraints needed for this project. Specifically, the challenge is to create a sensor system that can accurately differentiate in which trampoline area an athlete lands on after each jump (position of the deformation on the trampoline bed surface).

Furthermore, the sensor(s) chosen must be somewhat cheap and cannot change in any way the trampoline's elastic/flexible properties.

1.4 Thesis Structure

This thesis is organized as follows. In Chapter 2, other technologies and academic studies related to Trampoline Gymnastics will be presented and analyzed. After that, in Chapter 3, the requirements necessary for the desired system to work and a general architecture of the different sub-systems that constitute it will be proposed. In Chapter 4, the hardware and software implementations of these sub-systems will be described and the inner working of the system will be explained. In Chapter 5, the testing and validation of the system will be discussed and which results were obtained. To terminate, in Chapter 6, the conclusions about how the system as a whole stacks up to the previously defined requirements will be presented. Some future work and improvements that can possibly be added to the system will also be referenced.

2

The State of Sensor Technologies Applied to Trampoline Gymnastics

Contents

This chapter will focus on existing studies and systems used to measure various parameters in Trampoline Gymnastics. Some of the referred systems are actual products that are being sold and used both in training and competitions while others are academic studies or patents. This chapter will focus more on the technology/sensor components of these projects as it will be more relevant in order to elaborate an architecture to make the proposed system work. Also the most significant contribution of this thesis is about the trampoline sensor system and not as much as the user terminal and the database (although they are all important for the whole system to work as desired).

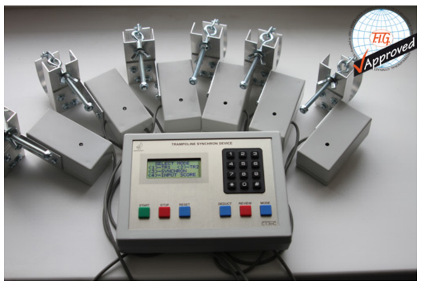
Because technology in Trampoline Gymnastics is a very specific topic, not many studies or systems have been developed. This also reinforces the important contribution to the area that this thesis might give.

2.1 Existing Products

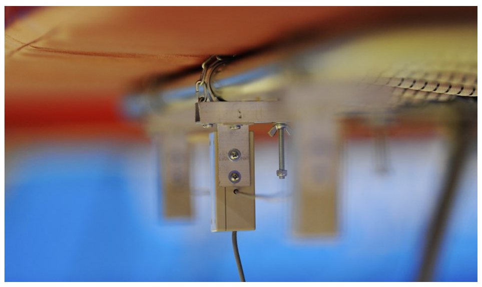
As referred in the Introduction, there are already Sporting Technologies (STs) that are used in Trampoline Gymnastics. The two main products that are being used during training and competition are: the AirTime Trampoline System (ATS) by Trampoline Timing Systems [52] and the Time Measurement Device (TMD) by Acrosport [2]. Both of these devices are very similar as they measure the TOF of a trampoline routine and can also be used to measure the synchro score during a Synchronized Trampoline Routine. This thesis will focus only on Individual Trampoline so this feature will not be explored, although this system can be easily adapted to also measure the synchro score in the future.

2.1.1 Time Measurement Device (TMD)

The TMD is comprised of three pairs of infrared (IR) beam emitters and receivers. These are attached under the trampoline bed onto the metallic frame (the receiver is attached on the opposite side of the emitter). Figure 2.1 shows an image of the whole system and how it is mounted on the trampoline.



(a) The whole TMD system [2].



(b) TMD system mounted on the trampoline frame (receiver side) [18].

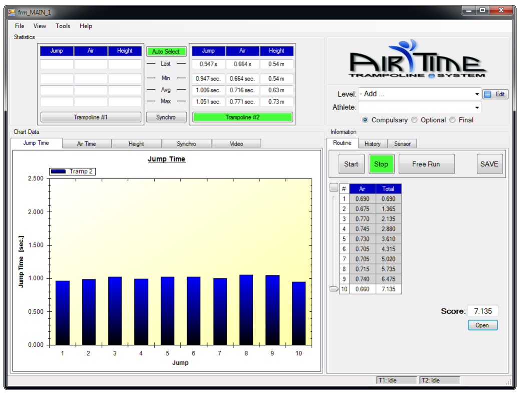
Figure 2.1: Images of the Time Measurement Device by Acrosport.

The system works by detecting when one of the three beams is interrupted, caused by the deformation of the trampoline bed when an athlete lands on it. By being able to detect the moments when an athlete leaves and lands on the trampoline bed, the machine can then calculate the Time of Flight (TOF) of each skill that the athlete executes. The product can register the moment when an athlete lands and leaves the trampoline bed with an accuracy of 1 millisecond.

As seen in Figure 2.1a the user only interacts with the TMD using a low resolution LCD Display and some physical buttons to control the machine. It can only store data from one routine at a time and display the information about each jump individually. It is also important to refer that this product is FIG approved.

2.1.2 Air Time System (ATS)

The ATS system is comprised of two (or three, for competitions) industrial 24VDC sensors that are also attached to the trampoline frame below the bed. Unfortunately a detailed description of the components in the system is not available and there isn't much information about how the system works. The user interacts with a GUI that runs on Windows, as can be seen in Figure 2.2. The ATS has two versions, one for competition and one for training.



	Jump	Air	Height
Last	0.947 s	0.664 s	0.54 m
Min	0.947 sec.	0.664 sec.	0.54 m
Avg	1.006 sec.	0.716 sec.	0.63 m
Max	1.051 sec.	0.771 sec.	0.73 m

Level: - Add ...
Athlete:
[✓ Compulsary] Optional Final

#	Air	Total
1	0.690	0.690
2	0.675	1.365
3	0.770	2.135
4	0.745	2.880
5	0.730	3.610
6	0.705	4.315
7	0.705	5.020
8	0.715	5.735
9	0.740	6.475
10	0.660	7.135

Score: 7.135

Figure 2.2: Air Time System GUI [52].

For the training version, the system can store information of different athletes in a database located in a personal computer. It also has the functionality to set specific TOF goals for each athlete.

The competition alternative has the ability to add the starting list (athletes order for performing) and associate the TOF score to each of them accordingly to the FIG code of points.

The main specifications of this systems are that it has 1 millisecond resolution and has a USB computer interface. This product is also FIG approved.

2.1.3 Summary

These two products are very similar in what they can do, still several different aspects can be identified. The most relevant difference, in my point of view, is the user experience. The ATS includes a GUI that displays information graphically and makes it easier for users to control the system. It also includes a database for storing the athlete's routine information so it can be consulted in the future. The only problem is that the database is located in the user's computer, limiting the flexibility and security of the system.

On the other hand, the TMD only has a small LCD screen with physical buttons that can only display text information to the user. The only advantage of the TMD is the price: for two trampolines the ATS is about €1400, while the TMD is about €900.

2.2 Other Measuring Systems for Trampoline

In this section the systems described are not commercially available and are not used during competion (not FIG approved). Some of these systems do not directly measure the parameters defined in the trampoline code of points [17], like TOF or HD, but they do measure other relevant factors related to the practice of Trampoline Gymnastics.

2.2.1 Automated Classification of Trampoline Motions Based on Inertial Sensor Input

This title refers to a Master's Thesis submitted by Heike Brock in Saarland University [8]. The purpose of the thesis is to be able to identify and classify different trampoline skills using inertial sensors (to measure, for example, acceleration or rotation). I will be focusing mainly on which sensors were used to capture the motion on the trampoline.

There are many different systems to capture the human body's motion. The most used systems are: optical systems (marker-based or marker-less), mechanical systems, magnetic systems and inertial systems. All of these alternatives provide different information formats and use different setups to capture motion.

Optical systems use images from several cameras to get a three-dimensional position of the human body. Some of these systems use special markers that very distinctively reflect light, attached to the subject, to help the cameras locate the points of interest in the movement. The main problem with these marker-based systems is that they can be very expensive and complex to setup. Others don't use any markers and take advantage of powerful computer vision algorithms to track the subject. These marker-less systems have the advantage that almost no setup is needed. The main problem is that computer vision algorithms are still not very accurate and require very powerful computers to process them.

Mechanical systems use a type of physical exoskeleton, attached to the body, to track the precise motion of various joints used in movement. The main problem with this system is that it restricts the movement of the actor, limiting her or his performance.

Magnetic systems use magnetic sensors attached to the subject's body and a fixed magnetic field generator. By measuring the variations in the sensor's values within the field, the system is able to precisely pin point the location of the person inside the field. The main problem with this system is that metallic objects can seriously interfere with the results, like wiring, lights or computers. Also, these magnetic sensors have to be tethered to a central processing unit via long wires which limits a lot the actor's movements.

Inertial systems use small inertial sensors that are attached to the body and gather orientational and dynamic data (not positional) of the motion that can be fed into biomechanical models to infer the precise movement of the subject. These sensors can contain an array of "sub-sensors" like accelerometers and gyroscopes. The main problem with this system is that the biomechanical models must be very precise for it to work well.

Table 2.1: Comparison of motion capture technologies based on [8]

System	Capture Volume	System Setup	User Mobility	Outdoor Factors	Marker Occlusion	Cost
opt. marker-based	Small	Very Hard	Very High	Very Big Influence	Very High Probability	Very High
opt. markerless	Very Small	Hard	Very High	Small Influence	Not Applicable	High
mechanical	Very Large	Easy	Very Low	Very Small Influence	Not Applicable	Low
magnetic	Very Small	Hard	Low	Very Small Influence	Not Applicable	Low
inertial	Very Large	Very Easy	Very High	Very Small Influence	Not Applicable	Low

Table 2.1 shows some of the most important parameters, relevant to Trampoline Gymnastics, that

were used to compare different motion capture systems. Given the particular nature of Trampoline Gymnastics, technologies that significantly limit the athlete's movements cannot be used. This discards mechanical and magnetic systems. Moreover, because of the speed and capture volume needed, optical systems present many disadvantages compared to other technologies. These technologies require a very complex setup to be able to acquire a big volume of data, their measurements are highly influenced by outdoor lighting and marker occlusion that can occur many times (given the many twists and turns that an athlete performs). In conclusion, given the requirements, the study found that inertial sensors were the best candidate to successfully capture Trampoline skill motion.

Ten Inertial Measurement Units (IMUs) were attached at key points in the athletes body, as can be seen in Figure 2.3. Each inertial sensors contained three accelerometers, three rate gyroscopes and three magnetometers. All of the data gathered by the system can then be analyzed and used to classify specific motions of the athletes body.

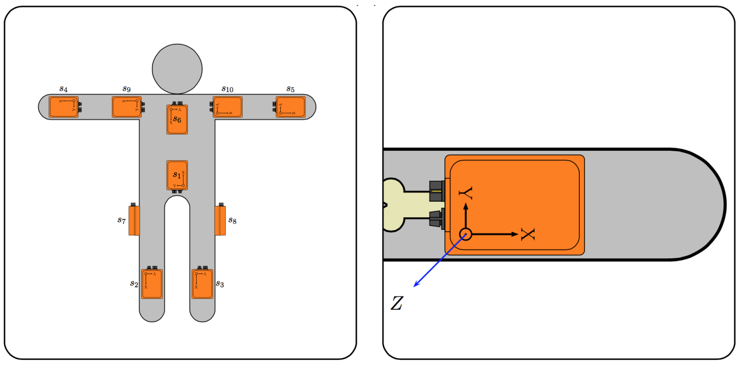


Figure 2.3: Placement of the IMUs in the athlete's body (left) to measure the limbs orientation (right) [8].

By repeatedly testing the system and gathering large quantities of data, motion templates can be generated that are then used to infer which skills are being performed in real-time. Also, using the equations for movement, many important variables can be calculated (angular velocity, acceleration, torque, linear momentum, force, etc.) that are then associated with specific motions that the human body performs.

2.2.2 Sports System Monitoring Intensity of Trampoline Jump

This work [10] was conducted by researchers from Coimbra University, in an effort to develop a system to determine the TOF of an athlete jumping on a trampoline.

To achieve this, the researchers used a HCSR04 ultrasound sensor and a MSP430 microcontroller to measure the distance between the floor and the trampoline bed. The HCSR04 sends an ultrasound signal towards the trampoline bed, which then bounces back and reaches the ultrasound sensor. The time interval between the emission of the ultrasound signal and the reception of the echo can be used to calculate the distance to the trampoline bed. This process is repeated over and over again showing the progression of the distance between the floor and the bed of the trampoline as an athlete jumps.

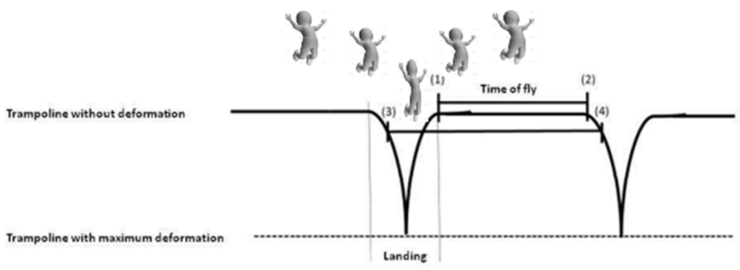
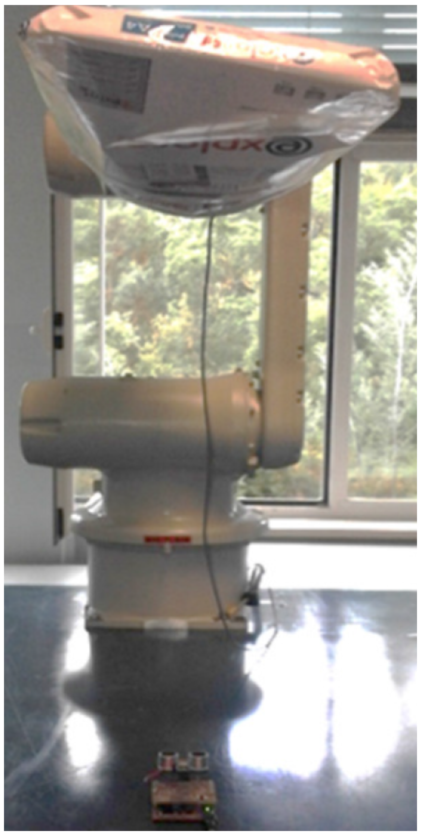
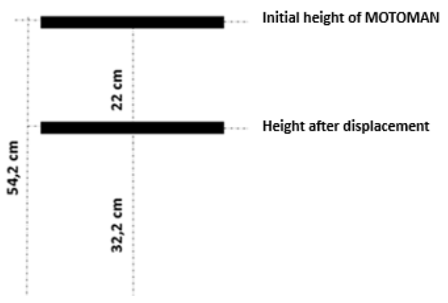

Trampoline without deformation
(1)
Time of fly
(2)
(3)
(4)
Trampoline with maximum deformation
Landing

Figure 2.4: Trampoline bed deformation with time [10].

As shown in Figure 2.4, there are two key time intervals that are important to differentiate. The first is the interval between moments (3) and (4). This represents the time elapsed for the entire skill, since the moment the athlete lands on the bed (3), until the moment the athlete returns again to the bed after she or he performed the desired skill. The second time interval is between moments (1) and (2), and this corresponds to the true TOF of the skill. It is the time since an athlete leaves the trampoline bed until the moment she or he lands on the bed after executing the desired skill. This means that the true TOF corresponds to the time interval between (3) and (4) minus the time the athlete spends on the bed of the trampoline.

To test the accuracy of the system, a robotic arm, MOTOMAN, was used to simulate the movement of an athlete falling on the bed. Figure 2.5a shows a photograph of the setup with the system hardware below the MOTOMAN. Figure 2.5b shows the distances that were used to simulate when an athlete is in the air, so there isn't any bed deformation (MOTOMAN at 54.2 centimeters), and when the bed is at maximum deformation (MOTOMAN at 32.2 centimeters).


(a) System setup [10].



(b) Distance representation of the system setup [10].

Figure 2.5: Images of the system setup with the robotic arm, MOTOMAN.

The tests that were conducted simulated two landings executed with a TOF of 2.6 seconds. What varied was the speed at which the simulated athlete was falling. Three speeds were chosen, 80 mm/s, 160 mm/s and 240 mm/s. The system was programmed to interpret that there was a deformation on the trampoline bed when the MOTOMAN was at less than 50 centimeters from the sensor. Also a C# program was developed to create a Graphical User Interface (GUI) to control the system and display the desired information via a serial interface.

After the three tests the main conclusion was that as the speed increases so does the error. For the first test, at 80 mm/s, the error was 0.28 seconds. For the second test, at 160 mm/s, the error was 0.337 seconds. Finally for the third test, at 240 mm/s, the error was 0.356 seconds.

2.2.3 Trampoline System for Measuring TOF and the Location on a Trampoline Bed of Jumps

The following description relates to a patent filed by Eurotramp (German trampoline manufacturer) and describes a system that can measure the TOF and detect where an athlete lands on the trampoline bed after each skill [16].

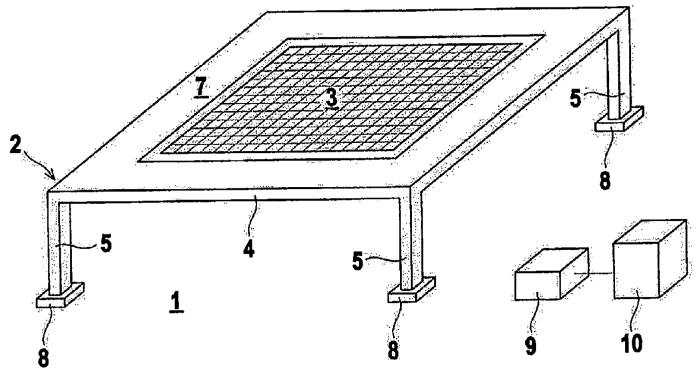


Figure 2.6: Invention of a system for measuring TOF and location on a trampoline bed of jumps [16].

Figure 2.6 shows an illustration of the various components of the system. The most important parts to notice are the cubic blocks that are positioned underneath the four legs of the trampoline (Figure 2.6: 8). These blocks are to be implemented as force sensors in the form of pressure sensors or strain gauges.

When an athlete is jumping on a trampoline with this system, there are two states that sensors must detect. The first is when the athlete is in the air executing the skill and the force applied on the plate is only that of the weight of the trampoline. The second is when the athlete lands on the trampoline bed and transferring her or his energy through the trampoline and in to the sensors. By distinguishing these two time intervals the system can calculate the TOF of the jump.

To be able to detect where the athlete has landed on the trampoline bed the system uses the difference in force applied between the four plates underneath the trampoline legs. In theory, when and athlete lands on the middle of the trampoline bed, the forces should be equal on all four sensors. But as the athlete lands farther away from the center, the forces applied will be greater on the sensors closer to where the athlete landed.

This system has a central processing unit (Figure 2.6: 9) that processes the data from the sensors and outputs the value for TOF and the location where the athlete landed on the trampoline bed. This data can then be converted into a TOF and Horizontal Displacement (HD) score and displayed on a user terminal like a smartphone, tablet or a computer (Figure 2.6: 10).

2.2.4 Other Applications or Technologies for Measuring Trampoline Parameters

Many patents have been filed for several different technologies applied in Trampolining. Figure 2.7 shows an image of the patent for a Sensor, Control and Virtual Reality System for a Trampoline [11].

The present invention aims to display an avatar of the athlete during a trampoline exercise. The system includes a computer module (Figure 2.7: 30) to process the data which is sent by the sensors, a trampoline as a platform for the user to perform the skills on (Figure 2.7: 22) and a sensor module able to detect the users movements (Figure 2.7: 220, 21 ,17, 12, 19, 13).

The sensor module can be a composition of various sensing modules attached to the user or to the trampoline or even remotely mounted. The document describing the patent does not specify exactly which sensors are used.

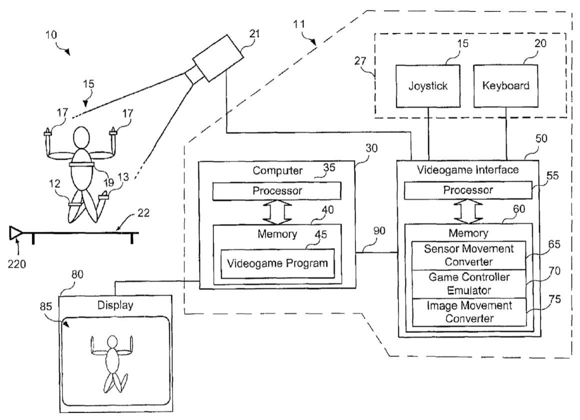


Figure 2.7: Invention for Sensor, Control and Virtual Reality System for a Trampoline [11].

Another interesting invention is from a patent publication of a Trampoline with Feedback System [53]. The general diagram can be seen in Figure 2.8.

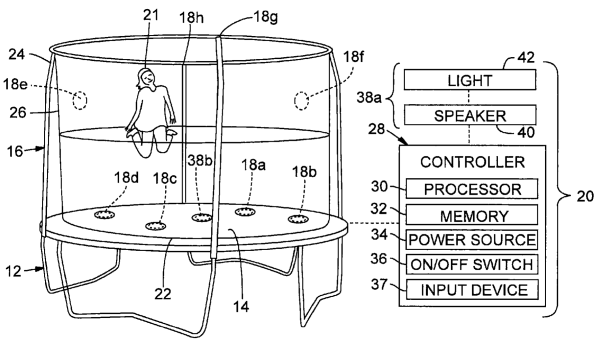


Figure 2.8: Invention for a Trampoline with Feedback System [53].

The system consists of a plurality of sensors that can be attached to the frame and/or to the mat of the trampoline. These sensors can be piezoelectric, proximity sensors, accelerometers, motion detectors or any other type. These devices gather information about when someone touches the bed and even track the movements of a user around the mat. The feedback signals transmitted from the sensors are fed into a controller that can then actuate using LEDs or sounds.

2.2.5 Summary

By analyzing these systems we can conclude that most of them are not intended for Olympic competitive Trampoline Gymnastics during training or competitions.

First of all some of the systems use sensors attached to the bodies of athletes to be able to gather the intended data. This is true for the system in section 2.2.1 that uses inertial sensors strapped to the jumper and the patent represented in Figure 2.7, which also included the possibility for sensors to be attached on the users body. The main disadvantage of this is that the extra gear limits and alters the motion of the athletes while executing very technical skills, which can compromise the jumper's safety.

The second reason is that some of these systems rely on altering essential properties of the trampoline. This can be seen in the patent represented by Figure 2.8 that includes the possibility to attach sensors to the bed of the trampoline. The elastic properties of the trampolines (springs and bed) are of the utmost importance to the performance of the athlete. By altering the feel and characteristics of these components seriously compromises the ability for the jumpers to perform. The patent described in section 2.2.3 shows a very complete solution for the problem that needs to be solved. The main drawback of this system is that the force sensors absorb some of the force that would normally be applied on the floor where the trampoline was mounted on. This creates a dampening effect that may lower the jumping height that the athletes can achieve on that trampoline. It is also important to refer that not all the skills performed on the trampoline are perfectly vertical, which means that not all of the force is applied vertically on the force sensors. This may lead to unreliable results in terms of the HD score.

The work done by Batalha M, Umbelino V and Amaro JP [10] is very good in the sense that it can, theoretically, calculate the TOF of the jumps without altering in any way the athlete's performance. One problem is its lack of accuracy and resolution due to the hardware and the time it takes for the ultrasound waves to physically hit the bed and echo back. Another big problem is the fact that this system does not present a solution for measuring the HD or the location where the skill lands on the trampoline bed.

2.3 Sensor Technology Comparison

As explained in Chapter 1, the main objective of this work is to create a sensor system technology that can measure the TOF and HD of an athlete during a Trampoline Routine. As previously referred, the system cannot be, in any way, invasive to the athletes performance by interfering with her or his mobility or with the properties of the trampoline. Given these limitations, the sensor must be able to detect an object's presence and position on a surface at a distance (without contact). This basically means detecting the "when and where" of the deformation on the trampoline bed.

Based on my knowledge of sensor systems and with some research, the simplest to use and cheapest sensor technologies that can detect an object's presence or distance are ultrasonic and infrared systems. Technologies based on cameras were not included because to have a system with millisecond resolution, the cameras would have to film at least at 1000 frames per second. These equipments are expensive and the computer vision algorithms would be very processing intensive. Another problem with these solutions would be their dependency on color and lighting conditions in the gym where the system would be used. These factors cannot be controlled in a real life scenario. Given these limitations these systems were considered impractical to solve the proposed problem.

Ultrasonic Sensors work the same way as bats use sonar to detect prey. The general idea is that an ultrasound is emitted by a source, being then reflected by the objects that it encounters. The time it takes for the echo to return to the source gives a good estimate of the distance of that particular object (the speed is previously known). The emitted sound wave has frequencies in the ultrasound range, which is inaudible to humans. Figure 2.9 illustrates the working principle of an ultrasound sensor.

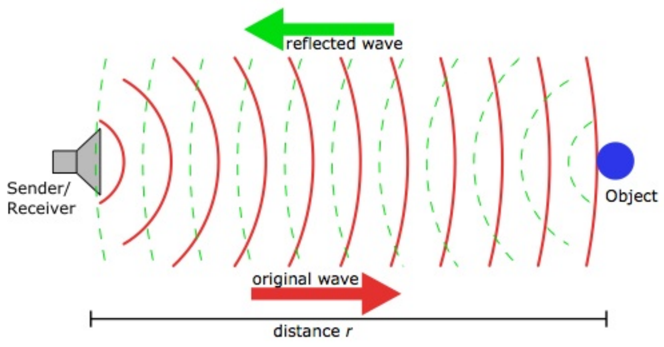


Figure 2.9: Ultrasound Sensor working principle [40].

Infrared Sensors work by detecting IR light from natural (passive) or artificial (active) sources. The main active IR sensors are the break beam and the reflective configurations. The most common passive IR sensor is the Pyroelectric Infrared (PIR) Sensors.

Break Beam IR Sensors work by having an IR emitter directly pointed at a receiver. This type of device can detect when an object is positioned between the emitter and receiver, blocking the beam. Usually the emitter pulses the IR light so that it is not confused with the ambient IR radiation. With this technology two of the most important features are the width of the IR beam and the angle at which the sensor can detect IR light. For the IR sensor, the angle can be managed using physical blockers or just by selecting the right sensor with the best characteristics. The IR beam width can be controlled using lens systems that collimate light or by using highly concentrated IR lasers (dangerous for health). Figure 2.10 shows an application of Break Beam IR sensor technology.

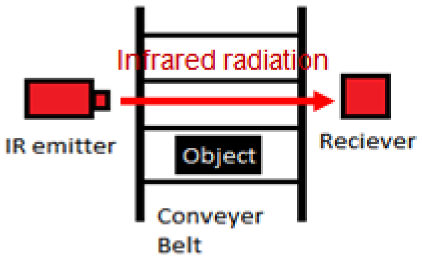


Figure 2.10: Break Beam IR Sensor example [27].

An advanced iteration of this system is achieved by combining several beams into an array that can detect and identify the shape of an object. Pepperl+Fuchs [34] produces light grid sensors that use this principle to detect the shape of packages that goes through the array. This is done by monitoring in real-time which beams are and aren't being interrupted by the object. This system has a range of between 0.3 and 6 meters with a maximum height of 3 meters. Figure 2.11 shows an image of the system being used to detect the shape of a car door.

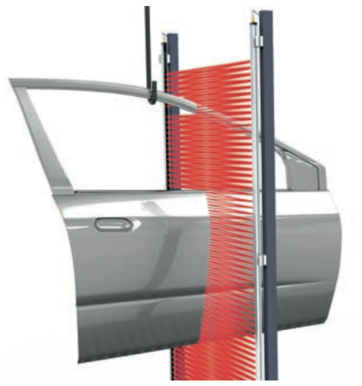


Figure 2.11: Light Grid LGS100 Serie by Pepperl+Fuchs [33].

A patent filed by James L. Levine, Susan A. Luerich and Duane Scott Miller [24], takes this concept a step further by configuring the IR beam arrays into a two-dimensional grid that is integrated as a touchscreen. The invention refers to a touch-sensitive display with a plurality of IR beams placed in front of it. The sensors are configured in two sets of arrays, one for the X-axis and one for the Y-axis. These are paired with the corresponding IR emitters. By triangulating which beams are broken in both the X and Y axis, the device can determine the location of the touch event. Figure 2.12 shows a diagram of the intended invention.

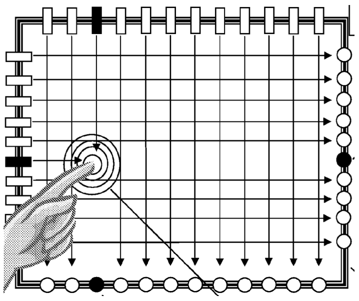


Figure 2.12: Invention for Infrared Touch Screen Gated by Touch Force [24].

Reflective IR Sensors take advantage of the reflective properties of IR light. It works in a very similar way to ultrasound sensors. The difference is that instead of measuring the time that the signal takes from traveling to an object, it detects the angle of the reflected IR light. Figure 2.13 shows an example of how the angle of IR light changes with distance.

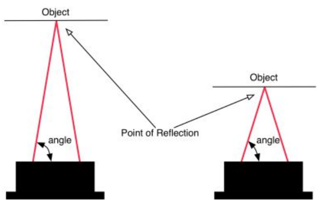


Figure 2.13: IR angle of reflection for near and far object using a Reflective sensor [1].

PIR Sensors work by detecting the natural radiation of IR light caused by warm bodies, analyzing changes in total amount of IR radiation compared to the ambient reference. These sensors include two IR detectors that are used to sense moving objects that pass in front of the PIR sensor. This is done by measuring the difference in IR radiation detected by each of the detectors. Normally a Fresnel Lens is used to increase the detection area of the device. Figure 2.14 shows a general diagram of how a PIR sensor works.

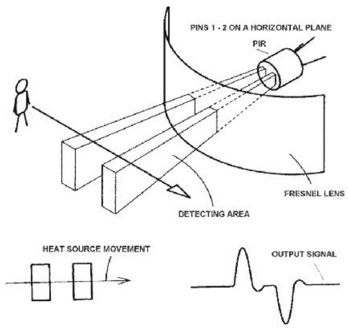


Figure 2.14: PIR sensor general diagram [3].

2.3.1 Comparisons

Table 2.2 shows a general comparison between the previously explained sensor technologies. The parameters chosen for comparison were those deemed more important for the requirements needed to solve the proposed problem.

As referred in the goals, the price is a very important factor in the final system. The prices in Table 2.2 were based on the cost of several component in the Sparkfun website [44]. Given this requirement, the PIR and Reflective IR sensor are not the most suitable choices as it would require several of each device to cover the whole area of the trampoline bed. We can also see that the Ultrasound and the Reflective IR sensors rely on the bouncing of signals to function. This can be very problematic

Table 2.2: Comparison between various sensing technologies [39][41][1][3]

System	Sensed Parameter	Detection Area/Range	Detection Time	Reflection Dependancy	Price
Ultrasound Sensor	Distance and presence of an object	3 to 10 meters (15 degrees angle)	340 millimeters in 1 millisecond	Dependent	Medium (~5-50€)
Break Beam IR Sensor	Presence of an object	10 centimeters to 10 meters (10 degree angle)	Less than 1 millisecond	Independent	Low (~3€)
Array of IR Break Beam Sensors	Presence, morphology, and location of an object	Area of the array plane	Less than 1 millisecond	Independent	Medium (~30-50€)
Reflective IR Sensor	Distance and presence of an object	10 to 100 centimeters	Less than 1 millisecond	Dependent	Medium (~15€)
Pyroelectric IR Sensor (PIR)	Movement of an object	10 meters (45 degree angle)	Dozens of milliseconds	Independent	Medium (~15€)

because the shape of trampoline bed when deformed is very irregular, which could cause unwanted echoes to influence the data gathered. Another issue with the Ultrasound sensor is it's dependency on the speed at which sound travels. This means that for distances above 340 millimeters it will take more than 1 millisecond just for the signal to travel towards the object and back.

Given all of these factors, the best solution is to use IR beams, specifically in a array configuration. It is a low-cost system that is able to detect the position of an object in a wide area. The idea would be to use two IR arrays to form a grid-like structure with two axis and place it below and parallel to the trampoline bed. This application would be similar to the patent filed by James L. Levine, Susan A. Luerich and Duane Scott Miller [24] but in a much larger scale. In this case it isn't a finger that interacts with the IR grid but the deformation caused by an athlete landing on the trampoline bed.

2.4 Inter-Sensor Communication Comparison

The trampoline mounted sensors will need to send data through to a common central controller so a communication protocol needs to be chosen. Completely wireless communication has the advantage of using less hardware (wires in particular) and being easy setup. The main problem with this solution is that the sensors would need to be powered by batteries and would need to be charged every now or then. Given that the system is suppose to be mounted on the trampoline for long periods of time, charging the sensors every "x" mount of days or hours, is completely unpractical. Seeing that powering the system with batteries isn't feasible, the best solution is to use wires to feed power to the sensors. Because wires will already need to be used for power, it only makes sense to also use a wired communication protocol between the sensors as it is simple to implement, reliable and cheap.

To keep the price point and complexity of the system as low as possible, a serial protocol was chosen. The most important features chosen to compare protocols, for the desired application, were: number of signal lines, possible network size, data rate, overall complexity of implementing the software and hardware and the approximate price to make a working prototype between two nodes. The protocols chosen for comparison were: Controller Area Network (CAN), Inter-Integrated Circuit (I2C), RS-485 and as a baseline a solution with parallel wires, one for each module in the network, was also taken into consideration. Table 2.3 summarizes this comparison.

Because of the the size of the trampolines, the serial protocol must be able to deal with relatively

Table 2.3: General Comparison between Serial Communication Protocols [12][26][47][30][31][23]

Protocol	Number of Signal Lines	Network Size	Data Rate	Implementation Complexity	Hardware Cost
CAN bus	2	50-1600 meters	50kbps-1Mbps	Very complex without many examples online	- MCP2515 CAN controller **(1.5€)** - MCP2551 CAN transceiver **(1€)** - MSP430G2553 **(2€)**
I2C bus	2	Limited by 400pF capacitance on bus (reported 10 meters)	100kbps-3Mbps	Easy and has a many examples online	- MSP430G2553 **(2€)**
RS-485	2	12-1200 meters	100kbps-10Mbps	Medium and not that many examples online	- MAX485 transceiver **(2€)** - MSP430G2553 **(2€)**
Parallel Wires	Equal to number of nodes	Depends on wire resistance	Depends on sampling speed of central node	Very easy in software Very complex in hardware	- MSP430G2553 **(2€)**

large distances (at least 10 meters). This requirement is accomplished by all of the chosen protocols although the distance that can be achieved sometimes depends on data rate or bus capacitance. In terms of data rate, all of the protocols can achieve enough speed to send the necessary data to the central module. If we assume that there are ten nodes in the network and all of them must send 1 byte (on/off state of the sensor) in less than 1 millisecond then that would mean a minimum of 80 kbits per second would be needed. Of course there is all of the processing behind the protocol that also takes up some time during the process.

Then there is the question of the physical/hardware complexity of the protocols. All use only 2 wires for communication, except the parallel solution where every node in the network has an individual signal wire that converges on the central processing node. Looking at software complexity the hardest protocol is the CAN because it includes many complex features that may not be necessary for the application at hand. The easiest protocols to implement would be the I2C and the parallel solution. The first because it is simple, easy to understand and has many support and documentation online, and the second because it simply needs the master to check the signal level of every wire to know the sate of all the sensors in the network. Finally in terms of overall cost, the cheapest solutions are the I2C and the parallel wires solution as they only need to use a generic microcontroller. All of the hardware configurations in the above Table 2.3 were based on using the generic MPS430G2553 microcontroller. This decision was done due to the availability of the microcontroller in the lab, it's low cost, high processing power and energy efficiency and because it has a large support community online. The prices of these different solution were based on the costs of the components in the Sparkfun [44] and Microchip [28] online shops.

Given all these limitations and analyzing the characteristics of all the protocols the most appropriate one for the application at hand is the I2C. This is mainly because of its simplicity to implement, good and abundant online documentation and it is one of the cheapest to develop working prototypes. Most of the other protocols offer better features but they can be considered "overkill" in terms of what is needed to solve the proposed problem. These better features make them more time consuming to understand and to implement. The parallel wires solution in theory seems very appealing but due to the high number of wires necessary, in practice, it would be what is called a “living hell”!

2.5 I2C Primer

In the I2C protocol there are two main bidirectional wires, serial data (SDA) and serial clock (SCL), that are used to transmit information between the devices connected to the bus. Devices can usually be both transmitters and receivers of data. There are also masters, which generate the clock and always initiate the data exchange, and slaves which respond to the masters’ requests. Each device is also identified in the network by a unique 7 or 10 bit software programmable address. There are 4 speed modes defined in the I2C specification [31]: Standard-mode at 100 kbps, Fast-mode at 400 kbps, Fast-mode Plus at 1 Mbps and High-speed mode at 3.4 Mbps.

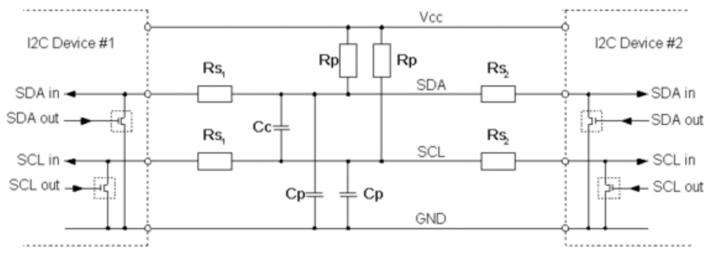


Figure 2.15: General circuit diagram for an I2C connection between two devices [22].

In I2C the SDA and SCL lines are open-drain, which means that the master or slave device can only drive them to be LOW (ground) or open. As shown in Figure 2.15, there are termination resistors Rp connected to Vcc that pull these lines HIGH (Vcc) when they are left open. Depending on which speed mode the network is operating at, certain timing requirements must be guaranteed. This depends mainly on the total capacitance Cp (wires and device pins) that bus has. One of the main factors that increases bus capacitance is the length of the wires. To counteract this effect, the Rp must be sized correctly so that the timing requirements are met. Figure 2.16 shows how the maximum Rp allowed decreases with the increase of Cp (increase in wire length).

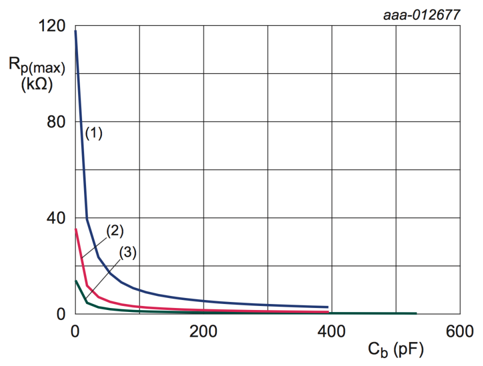


Figure 2.16: Maximum Rp as a function of Cp for Standard-mode (1), Fast-mode (2) and Fast-mode Plus (3) [31].

The I2C protocol allows for clock stretching. This happens when a device needs to slow down the clock so it holds the SCL line LOW for more time before letting it rise HIGH again.

Figure 2.17 shows a complete I2C transaction between one master and one slave. All data exchanges are initiated by the master. They begin with a START condition and end with a STOP condition. A START condition occurs when there is a transition of the SDA line from HIGH to LOW while the SCL line is HIGH. A STOP condition occurs when there is a transition of the SDA line from LOW to HIGH while the SCL line is HIGH. When the devices on the bus, that are not involved in the transaction, detect a START condition, they remain silent until a STOP condition occurs. After every byte transmitted, an acknowledge (ACK) bit must be transmitted by the receiver. The ACK is positive when the receiver holds the SDA line LOW during the HIGH period of the ninth clock pulse. If more than one master is transmitting in the bus, an arbitration process occurs. During the HIGH period of the clock pulse, if the value in the SDA line does not correspond to what that master transmitted then it means that another master with a "higher" priority is transmitting.

After the master generates a START condition it sends the 7 bit slave address. The eighth bit indicates if the master wishes to READ (logic "one") or to write (logic "zero") data to the slave. After the data is transfered the master can generate a STOP condition or generate a repeat START condition to initiate a transaction with a new slave.

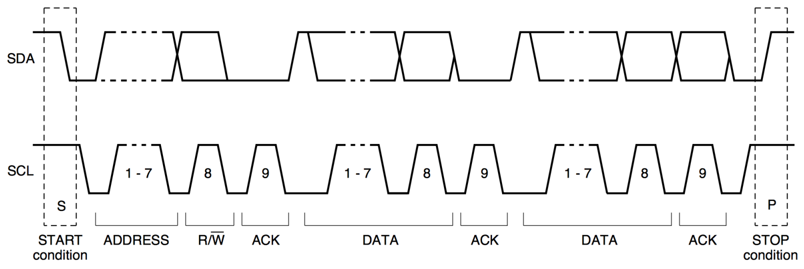


Figure 2.17: Example of a complete data transaction in I2C [31].

3 System Architecture and Requirements

Contents

In this chapter a proposed architecture for the system will be presented. Before being able to elaborate a specific architecture, the system requirements need to be well defined. These will be divided into functional and non-functional requirements. The first type relates to the actual behavior of the components when the system is being used, while the latter defines specific criteria that can be useful for evaluating the system's performance.

3.1 Requirements

Table 3.1: Functional and Non-Functional System Requirements

Type	Parameter/ Component	Requirement	Degree of Necessity
Non Functional	Computer application on user terminal	Run on OS X, Windows or Linux	Should Comply
	Database	Store at least 30 athlete profiles	Should Comply
	Sensor system to database distance	Transmit wirelessly at 15 meters	Must Comply
	TOF resolution	1 millisecond	Must Comply
	HD accuracy	100% accurate within 10 centimeters of each limit of the trampoline areas	Must Comply
	Connection between sensor system and user terminal	Wireless	Must Comply
	Trampoline compatibility	All olympic format trampolines	Must Comply
	Price	Less than 300 euros to manufacture (not including user terminal)	Must Comply
	Internet connectivity	No Internet connection must be required	Must Comply
	Power limitations	No significant limitations	Must Comply
Functional	User terminal	- Graphically display the HD and TOF of the routine - Provide Start/Stop functionality for the routine - Provide management interface of the athletes database (Add/Save/ Remove)	Must Comply Must Comply Must Comply
	Trampoline sensor system	- Detect location where athlete lands on the trampoline bed - Calculate total TOF of each jump - Correct setup visual validation (e.g. green LED) - Cannot interfere with "jumping" properties of the trampoline	Must Comply Must Comply Should Comply Must Comply
	Database	- Contain athletes data - Store routine information	Must Comply Must Comply

Table 3.1 shows all the requirements, divided into type and parameter/component, that the system needs to meet to achieve all the desired goals. The requirements are also classified by their "degree of necessity" in the system, which refers to the importance of meeting a particular requirement in the final prototype.

The main non-functional requirements are related to the accuracy and resolution of the trampoline sensor system. The measurement of Time of Flight (TOF) must have millisecond resolution or else it will not comply with the Fédération Internationale de Gymnastique (FIG) requirements [17]. The measurement of Horizontal Displacement (HD) must be able to differentiate with 100% accuracy in which of the highlighted areas (see Figure 1.2) the athlete landed on, within 10 centimeters of each side of the area lines. This requirement was defined by consulting national team level athletes and coaches. Both of these requirements are mandatory (**"Must Comply"**) in the final system. Another non-functional requirement that is important is related to the price of the machine. As referred in Section 1.2, the goal is to make a system that is cheaper than previous similar technologies, so a requirement of being able to manufacture the machine for less than €300 was defined as **"Must Comply"**. This value was obtained by consulting coaches from five different national clubs to understand what would be a reasonable price for the system (not including the user terminal). The average price obtained was €600. Given that in hardware products it is usual for the selling price to be double of that to produce the product, the €300 maximum cost to manufacture was reached. Finally, to make the system as easy to assemble and as unobstructive as possible, the connection between the user terminal and the sensor system must be wireless.

The main functional requirement has to do with the user terminal, as it is where most of the interaction with the user will take place. The User Interface (UI) should clearly display the TOF and HD of the routine in a graphical and intuitive manner. The interface must also have the functionality to manage routine information in the athletes database (Save, Delete, ...). It will also be from the terminal that the user controls the system by commanding it to start or stop the measurement of a routine. All of these requirements are defined as **"Must Comply"**.

In the next section a solution that attempts to meet all the requirements, as shown in Table 3.1, will be presented. The architecture of the whole system and of each individual component will be explained in more detail.

3.2 Architecture

By analyzing the system requirements that needed to be met and by comparing the different sensors that are available today, an architecture was elaborated and is generally represented in Figure 3.1. The system can be divided into three separate main components: the Infrared Beam Grid (IR-GRID) (in red), the Access Point Server (APS) (in gray) and the User Terminal (UT) (in black). Each one of these will now be explained in further detail.

3.2.1 IR-GRID

As shown in Figure 3.2a, the IR-GRID is comprised of a two-dimensional grid of Infrared Beam Module (IRM) containing a X-axis and Y-axis. Each IRM is comprised by an emitter and a receiver of infrared (IR) beams. IR light was chosen because it is relatively cheap to buy hardware that uses it and it is invisible to the human eye. This property avoids distracting athletes while jumping if they ever look at the beams. All the data created by the IRMs is sent to the Grid Controller (GC) to be processed. The IR-GRID will be attached, beneath and parallel to the trampoline bed, onto the metallic frame, as shown in Figure 3.2b.

The idea behind the IR-GRID is that the deformation of the trampoline bed, caused when an athlete lands, will interrupt some of the IR beams. Depending on where the athlete lands on the trampoline bed, different beams will be interrupted at different moments in time. The closer the IRM is to where the athlete landed on, the sooner the deformation of the trampoline bed will interrupt its beam. This means that the first beams to be interrupted, from each one of the axis, are the ones closest to where

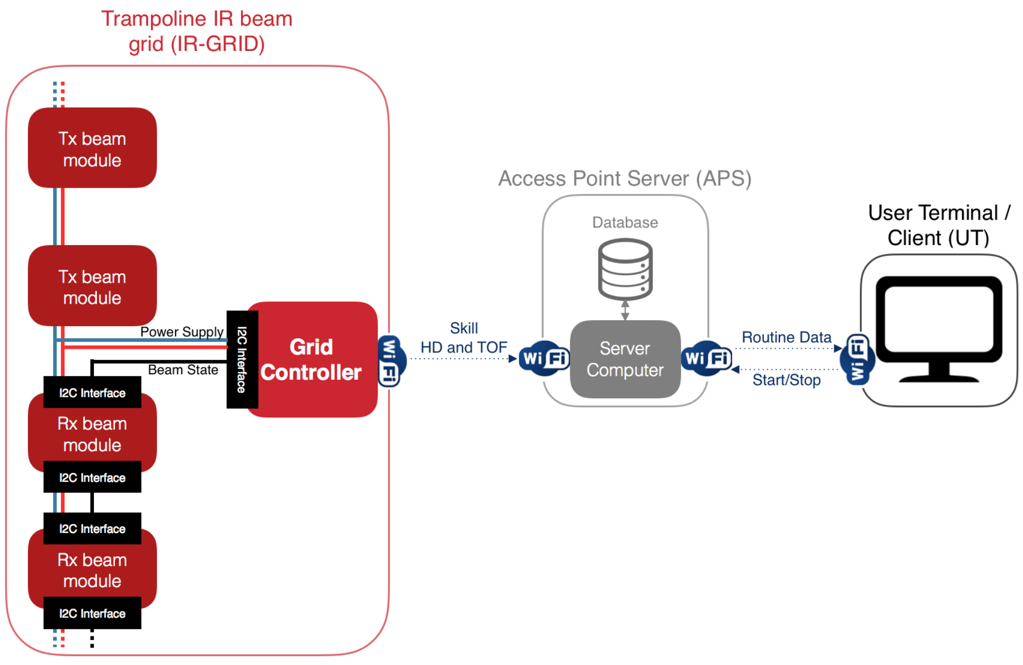
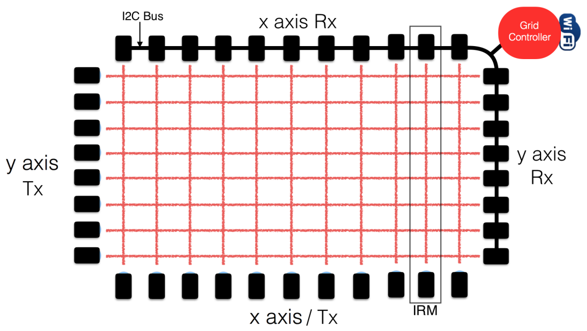
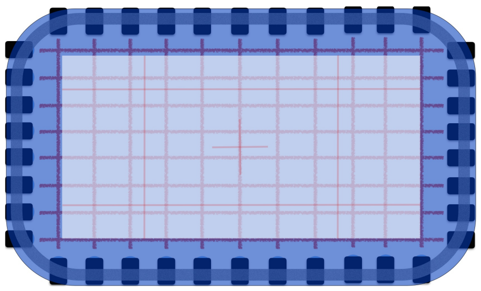


Figure 3.1: Diagram of the System Architecture.



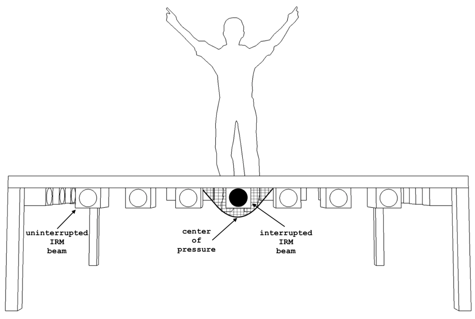
(a) General IR-GRID diagram.



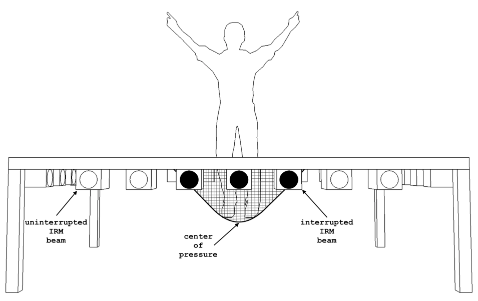
(b) Trampoline bed with integration of IR-GRID below the surface.

Figure 3.2: General IR-GRID.

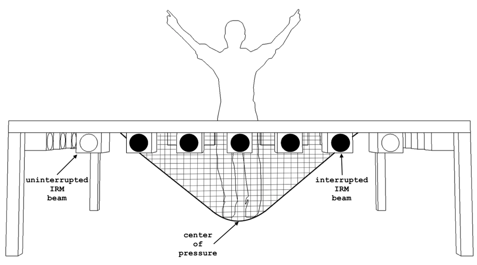
the athlete landed. By having a 2D grid, the system can analyze the chronological order of events on the X-axis and on the Y-axis to obtain a point in 2D space that corresponds to the origin of the deformation on the trampoline bed. When the athlete is in the air, no beams are interrupted. The moment the athlete lands on the bed, the deformation will start interrupting some of the IR beams, first the ones closest, then the ones farthest. As the deformation of the bed increases (athlete falling deeper) more and more beams become interrupted, until the point of maximum deformation. At this point the highest number of beams interrupted is reached. This number can be equal to all of the IRMs or not. This process is clearly illustrated in Figure 3.3. The progression of events seen from the Y-axis is similar, as the deformation of the trampoline bed can be approximated to a conical shape. When the athlete starts leaving the bed, the process is symmetrical. The beams farthest from where the athlete landed are the first ones to stop being interrupted. This description of how the deformation of the trampoline bed, caused by an athlete landing on it, interacts with the IR-GRID is only a theory. It is important to refer that the point the system is actually measuring is the lowest point of the deformation of the trampoline bed, indicated in Figure 3.3. This point is the center of pressure applied by the gymnast when impacting the trampoline bed, which should be in the middle of both feet and slightly in front of the heel. Further tests will prove if the theory is correct or not.



(a) Minimum bed deformation with one IR beam interrupted.



(b) Medium bed deformation with three IR beam interrupted.



(c) Maximum bed deformation with five IR beam interrupted.

Figure 3.3: Bed deformation progression interacting with the IR-GRID (viewed from the X-axis side).

The IR-GRID is also able to calculate the TOF of each skill by measuring the time between when the first beam of any IRM is interrupted (athlete landed) and when all the beams in the IR-GRID stop being interrupted (athlete in the air).

The resolution of the IR-GRID for detecting where an athlete lands on the trampoline bed is directly proportional to the number of IRMs on each axis. It is similar to a touch screen, the higher the number of pixels, where the higher the resolution. Although the system needs to have good resolution, in practical terms, what is actually needed is only to detect in which of the areas, highlighted in Figure 1.2, the athlete lands on. Inside each area the system doesn't need to know where the athlete actually is. Given this optimization, the minimum number of IRMs is 10, as shown in Figure 3.4 (six IRMs in the X-axis and four in the Y-axis). During the rest of this thesis many references to the IR-GRID will be made. To help visualize and understand how the system works use Figure D.1, in Appendix D.

Olympic trampolines have eleven different areas with different HD scores when an athlete lands on them. Delimiting these areas are four vertical lines (LX1, LX2, LX3 and LX4) and two horizontal lines (LY1 and LY2). To determine whether an athlete landed on one side or the other of the trampoline lines, there must be one IRM equally distanced from and on each side of the line. This can be seen clearly in Figure 3.4. Some IRMs can be used in common for two trampoline lines, like for example the IRM between LX1 and LX2. If the areas in the trampoline bed formed a perfect grid there would be one X-axis IRM and one Y-axis IRM for each one of the areas. But because this in not the case and some areas are bigger than others, there are some that have more than one X-axis IRM and/or more than one Y-axis IRM, like for example in area A5. This would mean that in theory, this IR-GRID could be able to differentiate more areas than those that exist in the trampoline. But, as explained before, it doesn't matter where the athlete has exactly landed inside the same area as the score deduction is the same.

The positioning of the IRMs is limited by the dimensions and structure of the trampoline and by the dimensions of the areas. The distances between the IRMs in the X-axis are defined by the need to have an IRM at equal distance between lines LX1 and LX2 and one at equal distance between lines LX3 and LX4. In the Y-axis the distances between the IRMs were defined by the structure of the trampoline itself that limited where the hardware could be attached to the metallic frame.

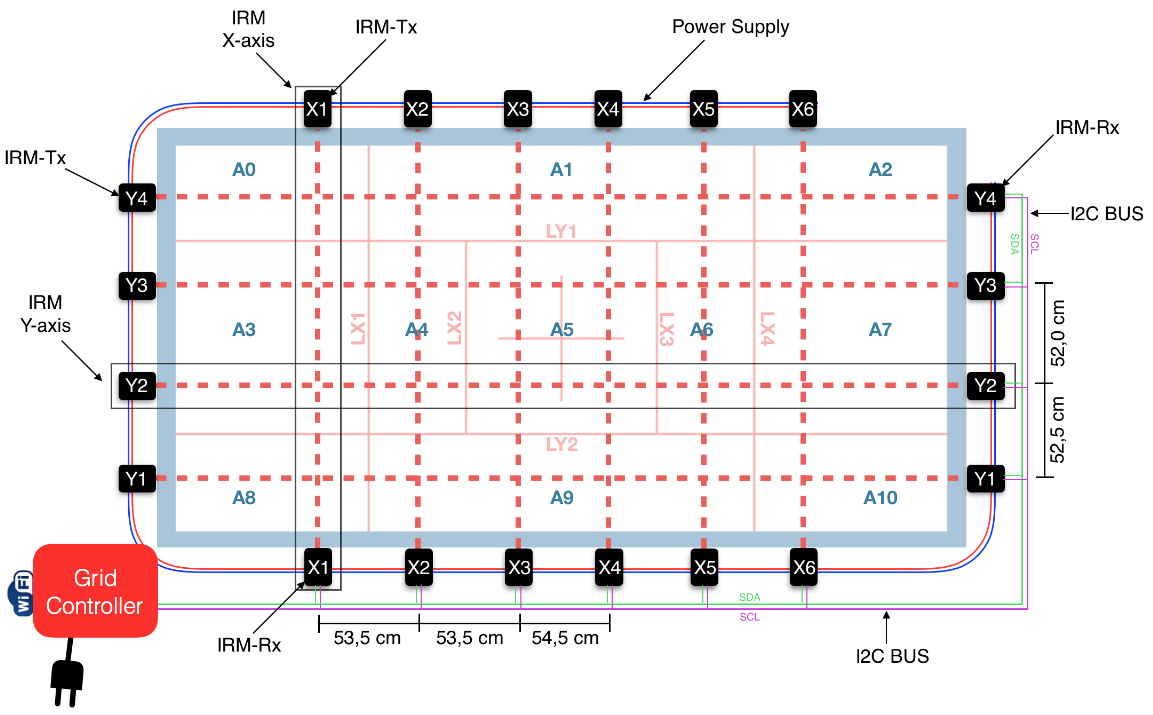


Figure 3.4: IR-GRID configuration.

The IRMs are all connected to the GC via an Inter-Integrated Circuit (I2C) bus and are powered by ground and positive lines. For more information about I2C refer to Section 2.4 and Section 2.5. The GC is connected to the power grid via a transformer.

3.2.2 IR Beam Module (IRM)

The IRMs are the basic building blocks of the IR-GRID. Each one is responsible for producing an IR beam and detecting whether it is interrupted or not by the trampoline bed. To produce the IR beam, the IRM has a submodule called the Infrared Beam Module Transmitter (IRM-Tx). To detect the presence of the beam there is a second submodule called the Infrared Beam Module Receiver (IRM-Rx). The beam produced by the IRM-Tx should be narrow enough to only hit its corresponding IRM-Rx and not interfere with other neighboring IRM-Rx.

All the IRM-Tx are powered by a common power supply that connects to the GC. All the IRM-Rx are also powered by a common power supply that is also connected to the GC. This solution was chosen over using batteries to avoid the trouble of charging all the modules every “x” amount of days, requiring mounting and dismounting the system several times. All the IRM-Rx are also connected to the GC via an I2C bus. This common bus is used to send data relative to the state, interrupted or not, of the corresponding IR beam. Figure 3.5 shows a basic diagram of an IRM.

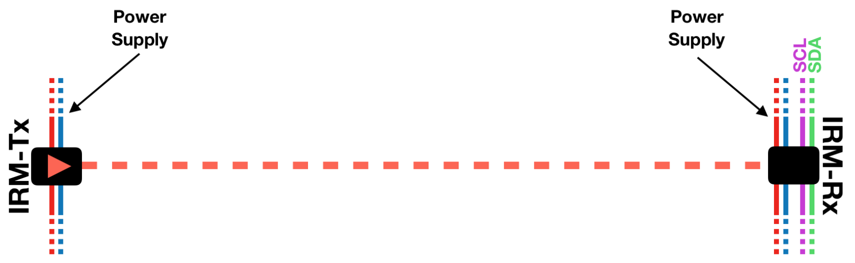


Figure 3.5: IRM diagram.

3.2.2.A IR Transmitter (IRM-Tx)

As mentioned before, the IRM-Tx must emit a narrow beam of IR light (so it does not interfere with its neighboring IRM-Rx modules) with enough power to be detected on the other side of the trampoline, which has 520 centimeters on the X-axis and 305 centimeters on the Y-axis (Figure 1.1a). Another important factor is that the IR light from the transmitter must be clearly distinguishable from the other IR sources, like the sun. The easiest way to solve this is by pulsating the IR Light Emitting Diode (LED) at a certain frequency so it can easily be distinguished from the ambient IR light [43].

Given these requirements, the IRM-Tx must contain a modulating circuit to pulse the light, an IR light source with enough power to emit a strong beam and a lens to concentrate the IR radiation into a narrow beam. Figure 3.6 shows a simple diagram of the components needed for the IRM-Tx.

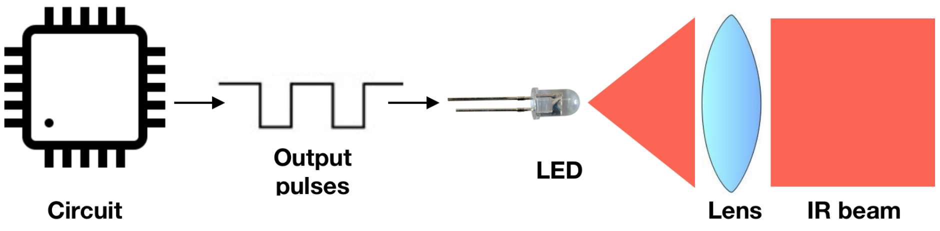


Figure 3.6: Diagram of the IRM-Tx.

3.2.2.B IR Receiver (IRM-Rx)

As said before, the IRM-Rx must be able to detect the pulsed IR beam and differentiate it from other IR sources. It will sense if the beam is broken or not (athlete on or off the bed). This module must have an I2C interface to send data, relative to the state of the IR beam, through the I2C bus to the GC. It must also have a processing unit to implement the I2C protocol and to perform any calculations on the data read from the IR sensor.

Given these requirements, the IRM-Rx must contain an IR receiver/sensor that can detect a specific frequency of modulation and a microcontroller with an I2C interface. Figure 3.7 shows a diagram of the components in the IRM-Rx.

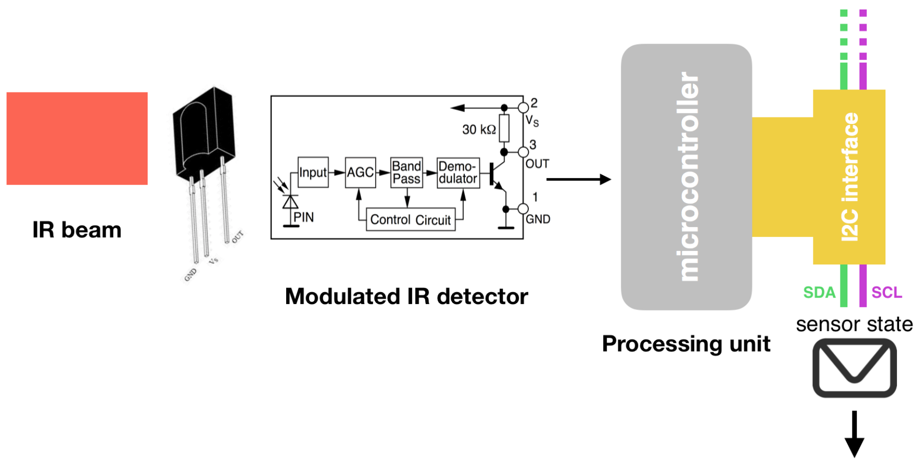


Figure 3.7: Diagram of the IRM-Rx.

3.2.3 GRID Controller (GC)

The GC is the main component in the IR-GRID and connects it to the rest of the system. It must have a I2C interface to gather the data from the sensors in the IRM-Rx, related to the state of the IR

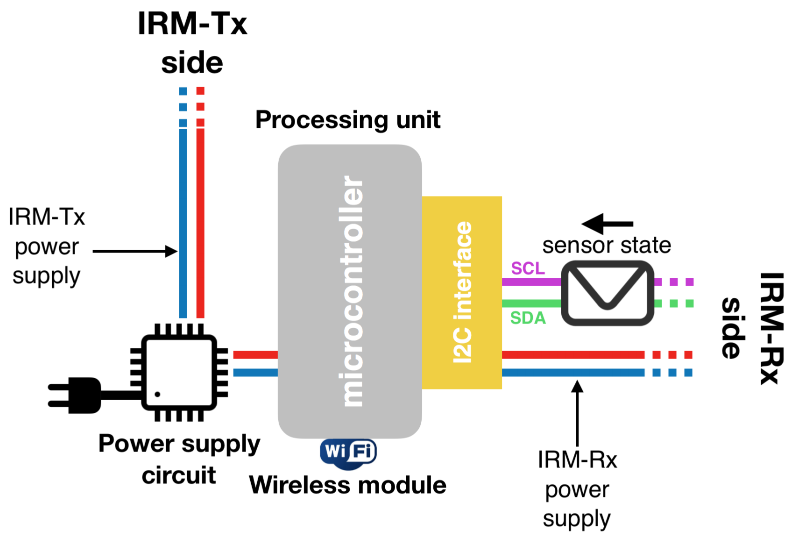


Figure 3.8: Diagram of the GC.

beams. This information is processed by a processing unit that will output the TOF and the trampoline area where the athlete landed on (which translates into a certain HD value) after every jump that is performed on the trampoline. The GC must also have a wireless module to send data to the APS. The GC will send the HD and TOF data of every skill that is performed on the trampoline to the APS, but only the most recent jump is stored at any given time. The GC is connected to the power grid via a transformer. A power supply circuit then regulates the power supply lines that go to the IRM-Rx and the IRM-Tx. Figure 3.8 shows a diagram of the components in the GC.

3.2.4 Access Point Server (APS)

The APS is the "middle man" between the user and the IR-GRID. It must have a wireless module to communicate between the GC and the UT. The main idea is to configure the APS as an Access Point (AP) so that both the IR-GRID and UT can connect to it, creating a small Local Area Network (LAN).

The APS contains two distinguishable databases: a small one to store the TOF and HD data of the most recent skill performed on the trampoline and a larger one to save coach profiles, athlete profiles and routine data performed on the trampoline by a certain athlete. The main idea here is that a gym can contain one or more coaches, a coach can teach one or more athletes and each athlete will have a history of saved routines with the corresponding HD and TOF data for each skill. The recent skill database is only written by the GC and is read by the UT. The coach/athlete/routine database is only managed by the UT. When a user commands the system to start a routine, the UT reads and temporarily stores 10 different skills from the recent skill database. If the user then chooses to store the routine, the data from the 10 skills is transfered to the APS and stored in the coach/athlete/routine database.

One way to easily implement these requirements with a wireless protocol is to develop a Web Application (WebApp) that runs on the APS. This means that the APS must also host a web server. The front-end will provide an UI to display and manage the data stored on the databases in the back-end. The recent skill database only sends its data to the front-end. The coach/athlete/routine database sends and receives data from the front-end. Figure 3.9 shows a general diagram of the APS.

3.2.5 User Terminal (UT)

The UT represents a "window" into the entire system. As referred above, it will serve as a Web Client to the WebApp implemented in the APS and as the Graphical User Interface (GUI) for the user. This

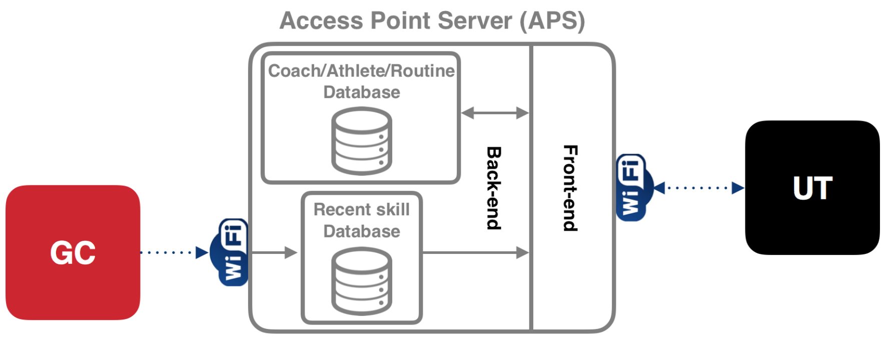


Figure 3.9: Diagram of the APS.

means that it will send requests for information to the APS. The main function of the UT is to display the data stored in the APS in a simple and graphical manner and to send the user's commands of START and STOP of the routine. The most important data displayed in the UT is the TOF and HD information of the most recent skill. Secondarily, it will also serve as a way for the users to manage the coach/athlete/routine database.

Because all the data is centralized in the APS, there is a lot of flexibility for the UT. It can be used by any device with a web browser (tablet, smartphone, laptop, etc.) and with wireless capabilities.

Figure 3.10 shows a mock-up of the Web Application's main page GUI, implemented in the APS. As defined in the system requirements (Table 3.1), the TOF and HD are displayed in a simple and graphical way, enabling easy analysis by the athlete and the coach. In the lower right corner there is an interface to choose in which athlete profile the coach wishes to save the routine and to remove an existing profile if needed. The buttons to send the START and STOP commands to the APS are also highly visible. This mockup serves only to give a general idea about how the main page of the GUI should look like and function.

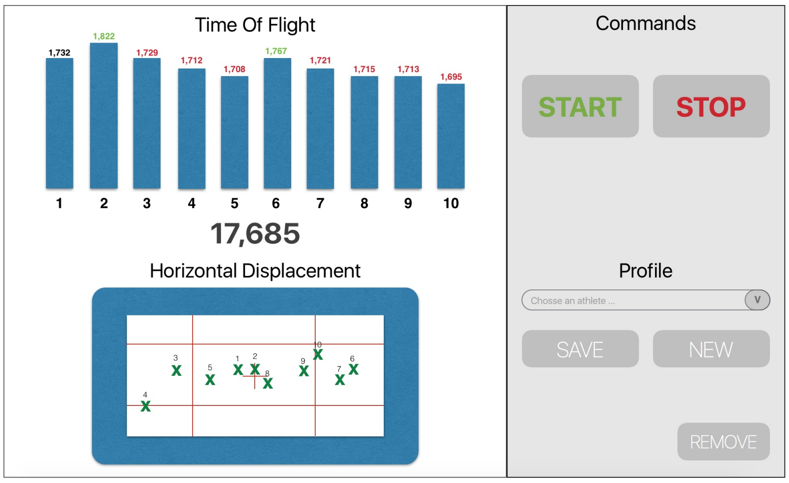


Figure 3.10: General mock-up of the Web Application's main page GUI.

3.2.6 System Connectivity

The global system is comprised of many modules and sub-modules that need to communicate with each other. As defined in Table 3.1, some connections will be wired and some wireless. The most important communication channels are between: the various IRM-Rx and GC; the GC and APS; the APS and UT.

3.2.6.A IRM-Rx – GC

In this connection data flows between all the IRM-Rx and the GC in the IR-GRID. As referred before this channel uses the **I2C** protocol to communicate. For more information about the I2C protocol refer to Section 2.4 and Section 2.5. The GC sends a request for data to each IRM-Rx. One at a time, each IRM-Rx responds with a message containing the state (interrupted or not) of its corresponding IR beam. To guarantee that the GC can detect changes in the IR-GRID with a resolution of at least 1 millisecond (as defined in Table 3.1), this exchange between all of the IRM-Rx and the GC must be done at least once every millisecond. Figure 3.11 shows a diagram of the communication between a IRM-Rx and the GC.

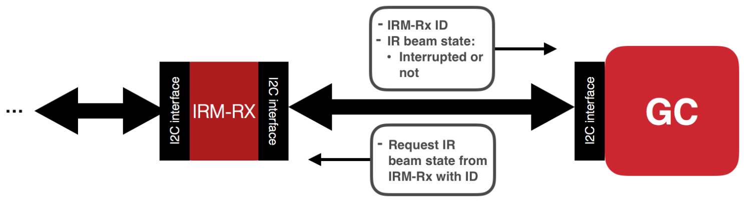


Figure 3.11: Diagram of communication between IRM-Rx and GC.

3.2.6.B GC – APS

The GC sends the final information about each jump to the APS. This data contains the TOF value and the HD area (one of the 11 represented in Figure 3.4) from the most recent skill. For future reference, for the rest of this document HD data refers to the trampoline are where an athlete has landed on. The actual HD score is calculated in the APS. The APS does not need to send any data to the GC so this channel is essentially one way. The APS will be positioned somewhere in the gym and the GC will be attached to the trampoline that is possibly very far away from it. To avoid having many long wires laying around on the floor, a **Wi-Fi** connection was chosen.

This wireless protocol was chosen because of its relative low price, high availability and high compatibility with many other devices on the market. Figure 3.12 shows a simple diagram of the communication between the GC and the APS.

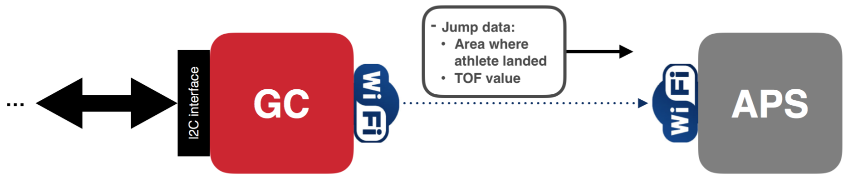


Figure 3.12: Diagram of communication between GC and APS.

3.2.6.C APS – UT

This channel of communication works as a simple server-client Internet protocol. As previously referred, the APS contains a Web Application that manages the databases. The UT is a client that will use the application with a web browser and send or request certain commands or information.

This connection will be bidirectional. The APS sends data related to the HD and TOF of the most recent skill, previously stored routine data and athlete profile data. The UT will issue the START and STOP commands to the APS and will be used to manage the databases, in terms of deleting or editing existing profiles. The UT also sends information of new coach or athlete profiles to be stored in the APS.

Once again, to avoid long cables in the gym and to give more flexibility to the user, a wireless connection was chosen and the preferred protocol is again **Wi-Fi**. Figure 3.13 shows a simple diagram of the communication between the APS and the UT.

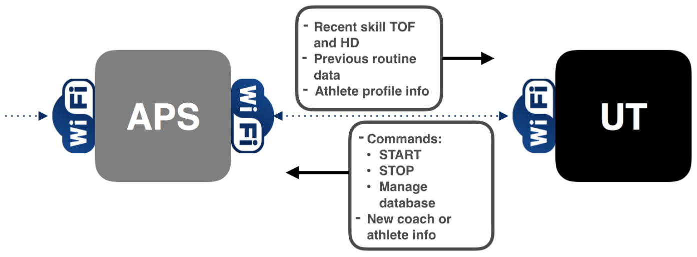


Figure 3.13: Diagram of communication between APS and UT.

4 Hardware and Software Design and Implementation

Contents

This chapter will focus on describing how the proposed architecture in Chapter 3 was designed and implemented, both in hardware and software. There will also be reference to some initial tests that were done in order to validate and help to decide about certain design choices. The specific electrical components and software frameworks were chosen based on the requirements that needed to be met (see Table 3.1): price, availability, documentation, laboratory resources and prior knowledge on the subject. There are four main components that were developed: the hardware and software for the Infrared Beam Module Transmitter (IRM-Tx); the hardware and software for the Infrared Beam Module Receiver (IRM-Rx); the hardware and software for the Grid Controller (GC); and the software for the Web Application (WebApp) in the Access Point Server (APS). The Graphical User Interface (GUI) of the WebApp runs on the User Terminal (UT) but the actual implementation is in the APS.

4.1 IRM-Tx Design and Implementation

As described before, the IRM-Tx generates an infrared (IR) beam that is received on the other side of the trampoline by the corresponding IRM-Rx. The main requirements for the IRM-Tx are that the generated beam needs to be narrow enough not to affect IR sensors other than the on from its corresponding IRM-Rx, it needs to be pulsed in order to be distinguishable from natural IR light sources and it needs to have enough power to reach an IR receiver at 520 centimeters on the X-axis and 305 centimeters on the Y-axis. In Figure 3.4 it is shown that the minimum distance between any two IRM-Rx is 52 centimeters which means that the diameter of the IR beam formed at the distance of any IRM-Rx must be smaller that 52 centimeters.

Given these requirements, in terms of hardware, what is needed is (1) a circuit to produce a pulsed signal, (2) an IR led with a relatively small emission angle, (3) a lens to focus the IR light even more and (4) a box that can correctly encapsulate the entire module. The software for the IRM-Tx is dependent on the hardware implementation.

4.1.1 IRM-Tx Hardware Design

The frequency chosen for the pulsated signal was 38kHz which is one of the most widely used frequencies for these types of applications and there is a large availability of cheap IR receivers for this specific frequency. There are two basic circuit solutions for creating a pulsating signal. The first solution is to use a hardware timer like the NE555 Timer from Texas Instruments [49], which uses external resistors and capacitors to control the frequency and duty cycle of the output signal. The second solution is to use a simple microcontroller and program it to create the desired signal on one of its General Purpose Input and Output (GPIO) pins. The latter option was chosen because it offers more flexibility and adaptability: for example, if in the future other frequencies need to be used to pulse de IR beams it is easy to change the microcontroller's programming to do so. The microcontroller used was the **ATtiny85** [7] because of its large availability, good online documentation and low price.

In terms of the IR light source the component chosen was the **TSAL6100** [55]. This Light Emitting Diode (LED) was used due to its low price, high availability and relatively low angle of half intensity. The angle of half intensity is the angle where the intensity of the light emitted is half of that at 0 degrees (straight ahead). In the case of the TSAL6100, this angle is 10 degrees to each side, meaning that at 520 centimeters the beam formed by the IR light would have a diameter of about 183.4 centimeters and at 305 centimeters in would have a diameter of 107.6 centimeters. Given the configuration illustrated in Figure 3.4, these numbers are not acceptable, as the IR light from one IRM-Tx would affect the IR receivers neighboring its corresponding IRM-Rx. To solve this problem, a focusing lens was used to decrease the emitting angle of the IRM-Tx. Figure 4.1 shows a simple diagram of how the lens is used to focus the light emitted from a LED.

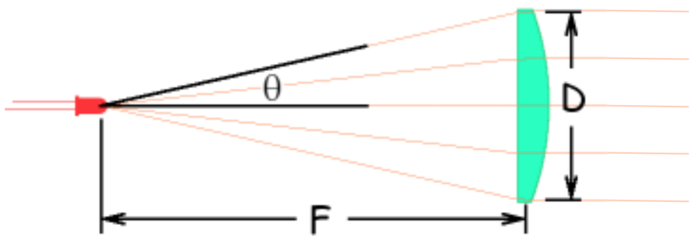


Figure 4.1: Diagram of a lens being used to focus the light from a LED, where D is the diameter of the lens, F is the focal length and θ is the angle of half intensity on the LED [25].

The simplest lens shape that achieves the desired effect is a plano-convex lens or an asymmetric double-convex lens. The minimum ratio between the lens diameter D and the focal length F is given by Equation 4.1a. As mentioned before, the angle of half intensity of the TSLA6100 is 10 degrees, forcing the ratio between D and F to be greater than or equal to 0.352. Obviously, the diameter and the focal length of the lens must also have reasonable values so that the IRM-Tx is small enough to be attached to the trampoline. Given the limitations of the structure of the metallic frame of the trampoline, the diameter of the lens should not be greater than 50 millimeters and the focal length should not be greater than 60 millimeters (Equation 4.1b and Equation 4.1c).

$$\frac{D}{F} \geq 2 * \tan(\theta), \tag{4.1a}$$

$$D < 50\ millimeters, \tag{4.1b}$$

$$F < 60\ millimeters. \tag{4.1c}$$

Based on the availability, price and dimension requirements referred above, the best option found was to use the same lenses that are used in the Google Cardboard Virtual Reality project [19]. These are **asymmetric double-convex lenses** having a diameter of 25 millimeters and a focal length of 45 millimeters. This means that they have a D to F ratio of 0.56 which meets all of the requirements in Equation 4.1a, Equation 4.1b and Equation 4.1c.

After analyzing all of the main components needed for the IRM-Tx, the electric circuit, illustrated in Figure 4.2, was designed.

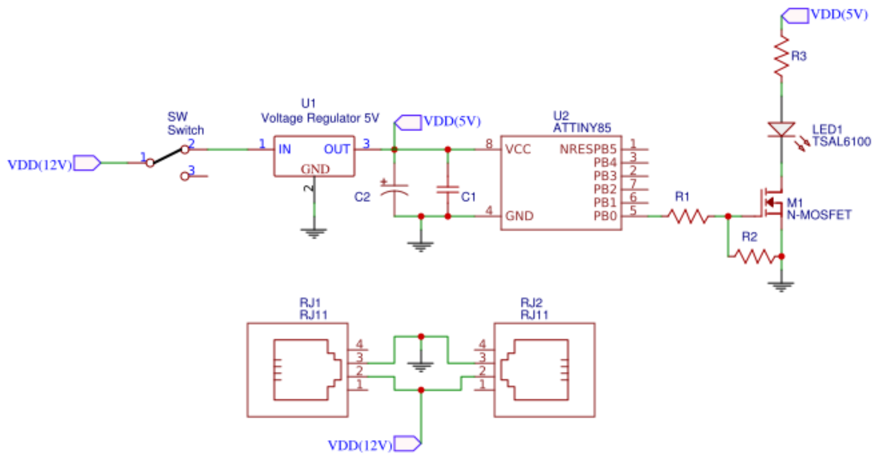


Figure 4.2: Schematic of the electrical circuit implementation for the IRM-Tx.

In the top left corner of Figure 4.2 is the voltage regulating part of the circuit. It uses a **linear voltage regulator** to convert the higher voltage that is fed into the IRM-Tx to a more usable 5V. The exact value of voltage that will be used to power the system will be determined later. Moreover there are two **decoupling capacitors (C1 and C2)** connected to the VCC pin of the ATtiny85 that are used to suppress noise in the power supply lines. Typically, there is a ceramic capacitor (C1) with a small value between 0.01μF and 0.1μF to short high frequency noise away from the microcontroller and a electrolytic capacitor (C2) with a higher value between 10μF and 100μF to smooth out lower frequency oscillations in the power lines [4]. These capacitors should be positioned as close as possible to the power supply pins of the microcontroller. There is also a **switch** to turn on or off the power supply to the IRM-Tx.

Connected to the decoupling capacitors is the ATtiny85 microcontroller, which generates the 38kHz signal. It is powered by VDD(5V) and the output signal is generated at PB0. This signal is then fed into the LED power circuit in the top right corner of Figure 4.2. This circuit uses a **N-Channel Metal Oxide Semiconductor Field Effect Transistor (MOSFET)** to supply current to the TSAL6100 LED. A transistor is used because the ATTiny85 cannot safely supply enough current to the TSAL6100 LED for the IR beam to reach a distance of 520 centimeters. The amount of current needed was determined using a testing prototype of Figure 4.2 and will be described in subsection Subsection 4.1.3. The **R1 resistor** is used to reduce the current surge when the microcontroller drives the MOSFET from “off” to “on”. The **R2 resistor** is a “pull-down” resistor used to ensure the MOSFET is “off” when the PB0 pin’s logic level is not defined (floating) and to discharge the accumulated charge caused by the capacitance between the Gate and the Source of the MOSFET. The **R3 resistor** is used to limit the amount of current needed for the TSAL6100 LED.

In the bottom part of Figure 4.2 are the **RJ11 female connectors**, used to receive and relay the power supply signals from and to the neighboring IRM-Tx. Although these connectors have four pins, only two are needed to transmit VDD(12V) and ground. RJ11 connectors were chosen because they are cheap, they provide a solid asymmetrical connection (no danger of switching the wires) and they do not require any special tools to crimp the connectors to the wires.

4.1.2 IRM-Tx Software Design

In terms of software, what is needed is a program that will switch the logic value of the PB0 pin between “high” and “low” every 13.15μs (half a period for a frequency of 38kHz). This is accomplished using the Timer0 in Clear Timer on Compare Match (CTC) Mode. In this mode the counter starts at zero and increments until it reaches a certain user-defined value called the Counter Compare Value. When it reaches the desired value it automatically resets the counter value to zero and starts incrementing again. Timer0 is also defined to be in toggle mode, which enables the microcontroller to automatically toggle the logic value of a specific pin every time the counter reaches the desired value. In this case, the associated pin was PB0. Timer0 was set to use the CPU clock frequency with no prescalar. The ATtiny CPU was programmed to run at 8MHz, which means that Timer0 increments the counter every $1/8000000 = 0.125\mu$s. This in turn means that the Counter Compare Value must be approximately $13.15/0.125 \approx 105$. The resulting code is shown in Listing B.1 (Appendix B).

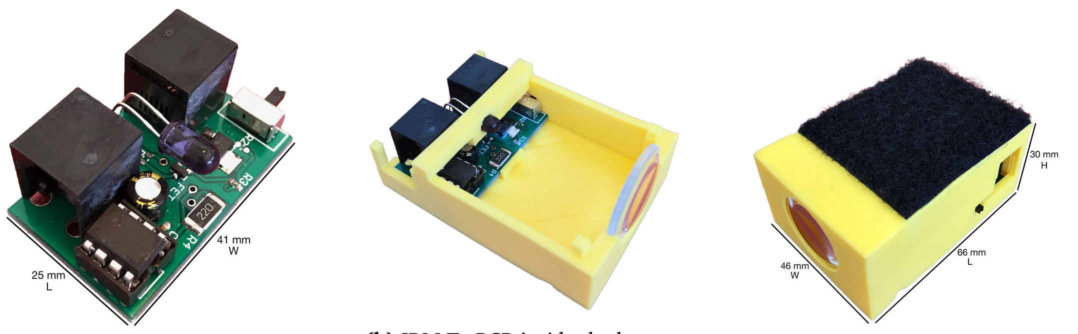
4.1.3 IRM-Tx Prototyping, Testing and Final Implementation

Even though the main components have all been defined, there are still some parameters that need to be tested to be correctly determined, including the IR beam diameter at 520 centimeters (where it is greater), the 38kHz signal and the value of R3, which in turn determines the amount of current needed for the IR beam to reach 520 centimeters. In order to test these parameters, the circuit in Figure 4.2 was implemented on a small breadboard using through-hole electronic components and with the R3 resistor being substituted by a potentiometer. The breadboard was encased in a 3D printed box with a socket for the lens.

The first thing to check was whether the program on the ATtiny85 created a correct 38kHz square wave form. After checking the signal in all ten microcontrollers that are needed for the entire system (as described in Chapter 3), it was noted that not all the frequencies were the same and could have a deviation of about 3% to either side of 38kHz. This is probably due to small differences between the chips, temperature, voltage and in the factory clock calibration, as described in the ATtiny85 datasheet [7]. Consequently, there was a need to adjust the value of the counter compare value for each individual microcontroller in order to obtain a signal with a frequency as close as possible to 38kHz. This calibration allowed the frequency deviation to be limited at ±0.6%.

The next test was to determine the value of the R3 resistor and the diameter of the IR beam at 520 centimeters. To achieve this a simple IR receiver circuit was implemented on a small breadboard using a TSOP4838 IR sensor. This circuit will be further described in Section 4.2. At this time the objective of the receiver circuit was to simply detect or not the presence of the IR beam generated by the IRM-Tx prototype. The transmitter and the receiver were aligned at a distance of 520 centimeters facing each other and there was a multimeter attached to LED to measure the current that was being drawn. According to the TSAL6100 datasheet [55], the maximum allowed forward continuous current is 100mA or 200mA at a 50% duty cycle. This means that while adjusting the potentiometer's resistance, the current read on the multimeter should not exceed 100mA (the multimeter only shows the average current). The potentiometer's resistance started at its maximum value and was gradually decreased until the receiver detected a consistent presence of the IR beam. The resulting resistance was 22Ω, providing approximately 160mA of current to the LED (an average of 80mA). After verifying that the sensor detected a consistent IR signal, the receiver was moved sideways to check where it stopped receiving IR signal. By doing so, the diameter of the IR beam at 520 centimeters was measured to be approximately 43 centimeters. This value is within the desired requirements as it ensures that each IRM-Tx only affects its corresponding (directly in front) IRM-Rx and offers some degree of "wiggle room" such that they do not need to be perfectly aligned to work.

Based on the schematic in Figure 4.2 and on the tests with the breadboard prototype, a Printed Circuit Board (PCB) layout was developed. The main requirements for the PCB were that it needed to be as small as possible (prioritizing a smaller length) and the TSAL6100 LED should be placed in one of the extremities of the PCB. The final PCB with all of the electrical components soldered to it is shown in Figure 4.3a, measuring 25 millimeters in length and 41 millimeters in width.



(a) IRM-Tx PCB with all electrical components soldered to it.

(b) IRM-Tx PCB inside the bottom part of the IRM-Tx box with the lens.

(c) Full IRM-Tx box encapsulating the IRM-Tx PCB and the lens.

Figure 4.3: Hardware implementation of the IRM-Tx and all of its components (PCB, lens and box).

To encapsulate the IRM-Tx PCB, a box was designed and 3D printed using Polylactic Acid (PLA) material. This box needed to have a support for both the TSAL6100 LED and the lens in order to ensure proper alignment between them. It also needed to have openings for the RJ11 female connectors and for the ON/OFF switch. The final version of the box and how the PCB fits inside it is shown in Figure 4.3b and Figure 4.3c, measuring 46 millimeters in width, 66 millimeters in length and 30 millimeters in height.

Another important requirement for the system was to keep the cost of production as cheap as possible. Table A.1 (Appendix A) shows the IRM-Tx Bill of Material (BOM) which was based on ordering the minimum quantities needed of each component from the cheapest supplier. The total cost of the materials for each IRM-Tx is **5.20€**.

4.2 IRM-Rx Design and Implementation

As described before, the IRM-Rx receives the IR beam emitted by the IRM-Tx and sends it's current state (receiving or not receiving) to GC via Inter-Integrated Circuit (I2C). The main hardware requirements for the IRM-Rx are that it needs to be able to sense IR light at 38kHz and not be influenced by natural IR sources, it must have a microcontroller to communicate via I2C with the GC and it must have a way to visually indicate to the user whether it is aligned or not with it's corresponding IRM-Tx.

Given these requirements, what is needed is (1) a 38kHz IR sensor, (2) a microcontroller with I2C compatibility to send information to the GC, (3) a LED that indicates whether the IRM-Rx is aligned or not and (4) a box to encapsulate de electronics. In terms of software, what is needed is a program that reads the signal from the IR sensor and transmits that information via I2C to the GC.

4.2.1 IRM-Rx Hardware Design

The IR receiver chosen was the **TSOP4838** from VISHAY [56]. This electrical component is comprised of an IR sensor and a control circuit to filter all frequencies except 38kHz, all in one package. It has three pins in total: one for the supply voltage; another for ground; and a third one for the output signal. The TSOP4838 works as a binary digital sensor in the sense that it can only output two values, either “high” when it is not receiving a 38kHz IR signal, or “low” when it is. There are many variants of TSOP4838 receivers but the main differences have to do with the receiving frequency and the IR code protocols that they support. Given that, for this application, the IR receiver is only used to indicate whether or not an IR signal is being received (no data needs to be transmitted), the main factors for choosing which TSOP4838 to use were price, availability and size.

In terms of the microcontroller, the model selected was the **MSP430G2553** from Texas Instruments [48]. The main reasons for choosing this particular microcontroller was the fact that there is a Development Kit called the MSP430 LaunchPad [50] that comes with a MSP430G2553, and because it has an Universal Serial Communication Interface (USCI) module that has a full I2C protocol implementation compliant with the I2C specification [31]. This microcontroller is also relatively cheap, easily available to purchase and has good documentation online. By having a development kit for the MSP430G2553 it was easier to test and develop a functioning prototype.

After analyzing all of the main components needed for the IRM-Rx, the electric circuit, illustrated in Figure 4.4, was designed.

In the top right corner of Figure 4.4 is the voltage regulating part of the circuit. It uses a **linear voltage regulator** to convert the higher voltage that is fed into the IRM-Tx to a more usable 3.3V. The exact value of voltage that will be used to power the system will be determined later. Moreover there are two **decoupling capacitors (C1 and C2)** connected to the VCC pin of the MSP430G2553 that are used to suppress noise in the power supply lines. The circuit also includes a **switch** to turn on or off the power supply to the IRM-Rx.

In the bottom part of Figure 4.4 is the TSOP4838 IR receiver which has its OUT pin connected to the MSP430G2553 microcontroller via the GPIO pin 2.3. Also connected to this line is a **LED** with a current limiting **R1 resistor**. When the TSOP4838 is receiving IR light, the OUT pin is pulled “low”, allowing current to flow through the LED and turning it on. When there is no IR light being received,

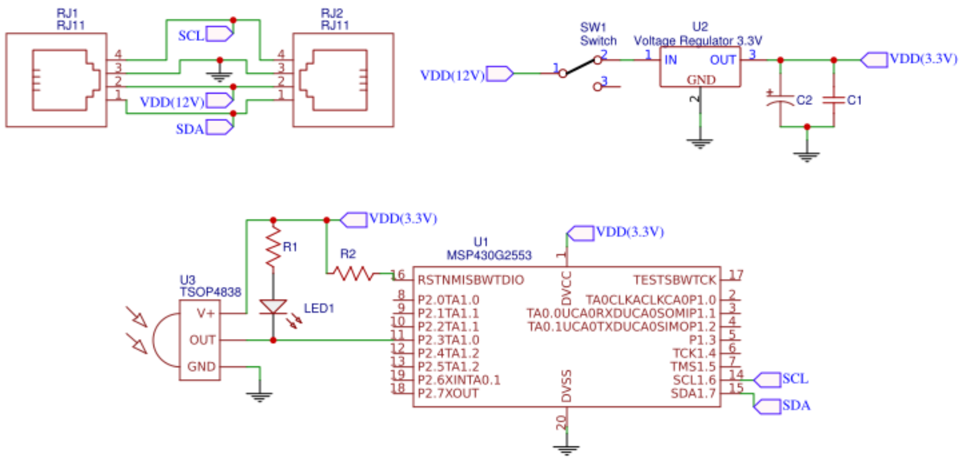


Figure 4.4: Schematic of the electrical circuit implementation for the IRM-Rx.

the OUT is pulled “high”, turning the LED off. Given that the main function of this LED is to give the user a visual indication if the IRM-Tx and IRM-Rx are aligned, it was implemented independently from the microcontroller. In other words, even if the microcontroller is not working correctly, it will not influence the LED’s behavior. The **R2 resistor** serves as a “pull-up” in order to maintain the RST pin “high” to avoid the microcontroller from hardware restarting.

In the top left part of Figure 4.4 are the **RJ11 female connectors**, used to receive and relay the power supply signals and the SCL and SDA lines from and to the neighboring IRM-Rx. The SCL and SDA lines are connected to pins 1.6 and 1.7 of the MSP430G2553, respectively.

To facilitate comprehension, the MSP430G2553 will hereafter be referred to as MSP430, and the TSOP4838 sensor as TSOP sensor.

4.2.2 IRM-Rx Software Design

In terms of software, what is needed is a program that reads the binary digital value in pin 2.3 and sends it via I2C to the GC. Based on the architecture defined in Chapter 3, the IRM-Rx is used as a slave in the I2C bus that only sends one byte of data when the master (GC) requests it. Based on examples provided by Texas Instruments on how to use the USCI module as I2C, the code shown in Listing B.2 (Appendix B) was developed.

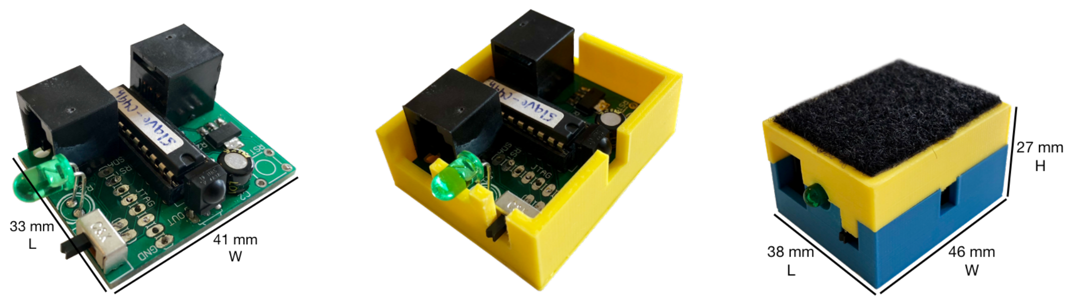
The program starts by calibrating the CPU clock to 16MHz, setting pin 2.3 as an input, setting pins 1.6 and 1.7 to their I2C functions as SCL and SDA, respectively, and configuring the MSP430’s USCI_B module in slave mode with a unique address (each IRM-Rx will be programmed with a different I2C address).

Most of the time the MSP430 is in Low Power Mode (LPM) with the system interrupts enabled. When the master wants to read the value of the TSOP sensor, it sends a START condition followed by the desired slave’s address byte (the eighth bit is “1” to indicate a READ operation). When the slave’s USCI_B detects that a START condition and its own address was sent on the bus, it sets USCI_B START detection interrupt flag (`UCSTTIFG`), triggering the corresponding Interrupt Service Routine (ISR) in function `USCIAB0RX_ISR()`. Given that only one byte of data will be transmitted, the `UCSTTIFG` is immediately set to “0”. After this, the USCI_B enters slave transmitter mode and waits for new data to be put on the `UCB0TXBUF` register by setting the USCI_B transmit interrupt flag (`UCB0TXIFG`), therefore triggering the corresponding ISR in function `USCIAB0TX_ISR()`. The value of pin 2.3 is represented by the binary value of the fourth bit in the `P2IN` register. The result from the AND operation between register `P2IN` and `BIT3` constant (0b00001000) is then copied to the `UCB0TXBUF` register so that it can be transmitted on the SDA line of the I2C bus. When the TSOP sensor is receiving an IR signal from the IRM-Tx, pin 2.3 will be “low” and the value put on the `UCB0TXBUF` will be 0b00000000, in binary, or 0x00, in hexadecimal. When the TSOP sensor is not receiving an IR signal pin 2.3 will be “high” and the value put on the `UCB0TXBUF` will be 0b00001000, in binary, or 0x08, in hexadecimal.

4.2.3 IRM-Rx Prototyping, Testing and Final Implementation

Before creating a PCB of the electrical circuit represented in Figure 4.4, a breadboard version using through-hole electronic components was developed in order to perform some tests. The first test, which was already partly described in Subsection 4.1.3, was to check if the IRM-Rx could consistently receive the IR signal from the IRM-Tx at the distances of 305 and 520 centimeters and if the indicating LED was working as designed. The second test was to simply check if the program and the I2C bus were also functioning as designed. This was done by using code examples provided by Texas Instruments, where the MSP430 LaunchPad was the master, and the IRM-Rx was the slave. More tests relating to the I2C bus were performed while designing and implementing the GC.

After passing all the tests, a PCB layout was developed, based on Figure 4.4. The main requirements for the PCB were that it needed to be as small as possible (prioritizing a smaller length) and the TSOP sensor should be placed in one of the extremities of the PCB. The final PCB with all of the electrical components soldered to it is shown in Figure 4.5a, measuring 33 millimeters in length and 41 millimeters in width.



(a) IRM-Rx PCB with all electrical components soldered to it.

(b) IRM-Rx PCB inside the bottom part of the IRM-Rx box.

(c) Full IRM-Rx box encapsulating the IRM-Rx PCB.

Figure 4.5: Hardware implementation of the IRM-Rx and all of its components (PCB and box).

To encapsulate the IRM-Rx PCB, a box was designed and 3D printed using PLA material. This box needed to have openings for the TSOP4838, for the RJ11 female connectors, for the ON/OFF switch and for the LED. The final version of the box and how the PCB fits inside it is shown in Figure 4.5b and Figure 4.5c, measuring 46 millimeters in width, 38 millimeters in length and 27 millimeters in height.

Another important requirement for the system was to keep the cost of production as cheap as possible. Table A.2 (Appendix A) shows the IRM-Rx BOM which was based on ordering the minimum quantities needed of each component from the cheapest supplier. The total cost of the materials for each IRM-Rx is **5.12€**.

4.3 GC Design and Implementation

As described before, the GC receives the sensor readings from the IRM-Rx, processes the data, calculates the Time of Flight (TOF), determines in which trampoline area the athlete landed on (which translates into an Horizontal Displacement (HD) score) and sends these values via Wi-Fi to the APS. This is done for every jump performed on the trampoline. The main hardware requirements for the GC are that it must have a microcontroller to communicate via I2C with the IRM-Rx and process the received data, Wi-Fi capabilities and a way to visually indicate to the user whether the system is working correctly or not.

Given these requirements, what is needed is (1) a microcontroller with I2C compatibility to receive information from the IRM-Rx, (2) a LED that indicates whether the system is working correctly or not, (3) a Wi-Fi module connected to the microcontroller to send the HD data and TOF score to the APS and (4) a box to encapsulate the electronics. In terms of software, the GC must be able to read the values of all the IRM-Rx and do the necessary processing every millisecond. This is required because the system needs to have a TOF resolution of one millisecond. It must also have a program that configures the GC as a Wi-Fi client and sends the HD and TOF data to the APS.

4.3.1 GC Hardware Design

The microcontroller chosen was the same as the one in the IRM-Rx, the **MSP430G2553** from Texas Instruments [48]. The reasons are the ones explained in Subsection 4.2.1. The Wi-Fi module chosen was the **ESP-12E** [42], which is based on the ESP8266EX System on Chip (SoC) [13] and integrated in the NodeMCU Development Kit V1.0 [59]. This module was chosen due to its very cheap price, high availability, extensive documentation online and easy to use Arduino libraries. To send the jump data from the MSP430 and the NodeMCU, a serial connection was made between the two. To facilitate comprehension, the ESP-12E/NodeMCU Development Kit will hereafter be referred to as ESP module.

After analyzing all of the main components needed for the GC, the electric circuit, illustrated in Figure 4.6, was designed.

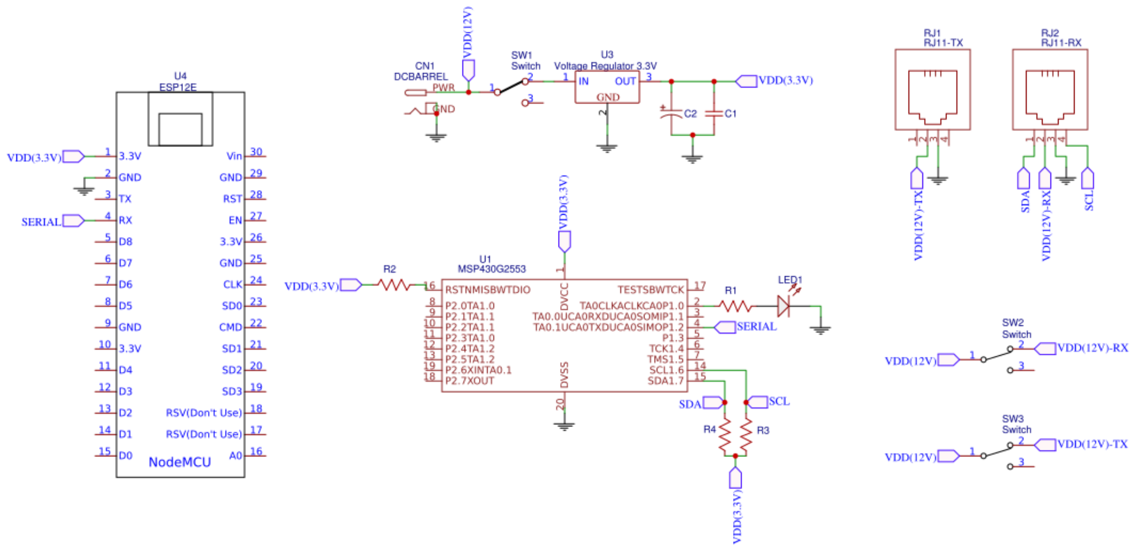


Figure 4.6: Schematic of the electrical circuit implementation for the GC.

In the top area of Figure 4.6 is the voltage regulating part of the circuit. It uses a **linear voltage regulator** to convert the higher voltage that is fed into the GC to a more usable 3.3V. The exact value of voltage that will be used to power the system will be determined later. Moreover, there are two **decoupling capacitors (C1 and C2)** connected to the VCC pin of the MSP430G2553 that are used to suppress noise in the power supply lines. **Switch 1** is used to turn on or off the power supply to the GC. Connected to this switch is a **barrel jack** that is used to connect the AC-DC power supply that will power the entire system.

In the bottom part of Figure 4.6 is the MSP430G2553 microcontroller. Connected to pin 1.0 is a **LED** with a current limiting **R1 resistor**. This LED was used for debugging during development and, in the future, for the user to know if the system is working correctly or not. The **R2 resistor** serves as a “pull-up” in order to maintain the RST pin “high” and avoid the microcontroller from hardware restarting. Connected to pins 1.6 and 1.7 (SCL and SDA lines) are **resistors R3 and R4**, respectively. These resistors serve as a “pull-up” for the I2C lines, as both SCL and SDA are open-drain, which means that they need to be externally pulled “high” when a digital “1” needs to be transmitted. In the left part of the circuit is the NodeMCU, which is connected to the MSP430’s TX pin 1.2 via the RX pin 4.

In the top right part of Figure 4.6 are the **RJ11 female connectors**. One is used to connect the power lines to the IRM-Tx and the other is to connect power and I2C lines to the IRM-Rx. In the bottom right part of Figure 4.6 are **switches 2 and 3** that turn on or off the IRM-Rx and IRM-Tx, respectively.

4.3.2 GC Software Design

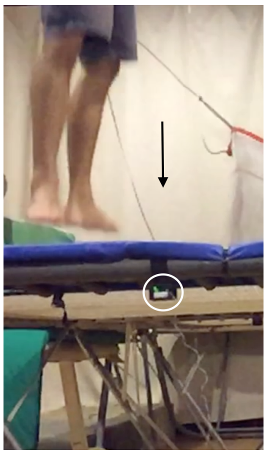
In terms of software, what is needed is a program that constantly reads the state of the IRM-Rx sensors and depending on how those values change over time (while the athlete is jumping), can calculate the TOF and HD data for each jump performed. There will also need to be a program that defines the GC as a Wi-Fi client and sends the relevant jump data to the APS. The first part of the software needed will be implemented on the MSP430 and the second part on the ESP module.

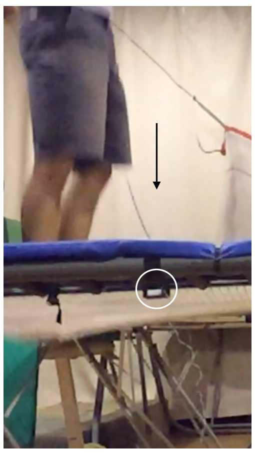
The most approximate depiction of how the trampoline bed deforms when an athlete lands on it is a conical shape where the vertex corresponds to the center of pressure (location where the athlete landed on) and the base width and height increase until the point of maximum deformation. This progression of deformation is depicted in Figure 3.3, viewed from the X-axis. The increase in base width of this cone is what interrupts the IR beams at different moments in time depending on how close they are to the vertex (closer IR beams are interrupted first). Theoretically, this means that by determining which IR beam, in both the X-axis and Y-axis, was interrupted first, the system can know in which area the athlete landed on. During the take-off phase of the jump, the progression of the deformation of the trampoline bed is the inverse. By analyzing the states of the IRM-Rx sensors, the GC must also be able to determine when the athlete leaves and lands on the trampoline bed in order to determine the TOF of the jump. This can be easily done by simply detecting when any IR beam is interrupted (athlete just landed on the bed) and when all IR beams are uninterrupted (athlete just left the bed). All this sensor reading and data processing must be executed in less than one millisecond as determined in the requirements on Table 3.1.

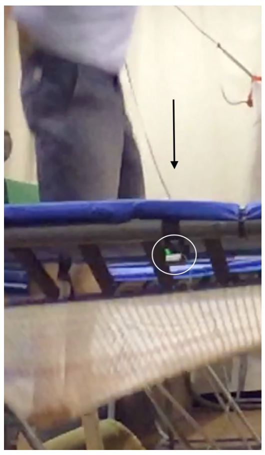
In order to verify if this theory is a close description of how the trampoline bed deforms, some preliminary tests were performed. Using two pairs of IRM-Rx and IRM-Tx, in positions X1 and X2 (see Figure D.1), and a slow motion camera (to view the LED state in the IRM-Rx), the idea was to check if there was discernible difference in the IR beam interruptions between jumping on areas A3 or A4. After performing the test, it was found that the trampoline bed does indeed behave as theorized. For example, when the athlete landed on area A3, the first IR beam to be interrupted was the one in position X1 and the second was one in position X2. Similarly, when the athlete left the trampoline bed, the last IR beam to become uninterrupted was the one in position X1.

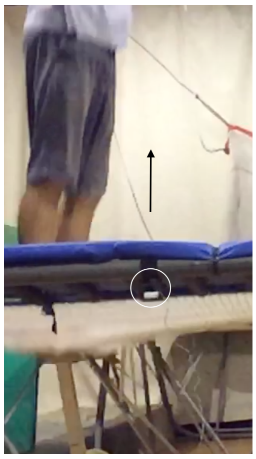
One very important characteristic that was unexpectedly found, is the fact that, during the deformation of the trampoline bed, the IR beams can become uninterrupted for some time. This is mainly due to the fact that when the trampoline bed stretches enough, the light from the IR beams can pass through the springs and the bed and reach the IRM-Rx sensors. There are six phases for the behavior of an IR beam: (1) the athlete is in the air and the beam is uninterrupted, (2) right after the athlete lands on the trampoline bed the beam is interrupted; (3) some time after, the beam can become uninterrupted while the bed reaches its maximum deformation; (4) the trampoline bed starts to reverse the deformation and the beam can continue to alternate between uninterrupted and interrupted; (5) right before the athlete leaves the bed the beam becomes interrupted once again; (6) when the athlete leaves the bed, the beam becomes uninterrupted during the rest of the time that the athlete is in the air. This process is illustrated in Figure 4.7 (phase 4 is not represented), where the green LED shows when the beams are interrupted (LED off) or not interrupted (LED on).

Because of this uncertainty of states during the deformation of the trampoline bed (phases 3 and


(a) Phase (1).


(b) Phase (2).


(c) Phase (3).


(d) Phase (5).

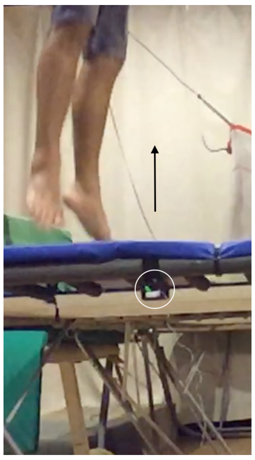

(e) Phase (6).

Figure 4.7: Progression of the IR beam state when an athlete lands on the trampoline bed.

4), it is harder for the GC algorithm to differentiate between when the athlete is on the trampoline bed or in the air. To solve this problem, the algorithm needed to leverage the fact that, during any jump, the time spent on the bed of the trampoline is always less than the time in the air. One should also note that, the higher someone jumps, the less time they spend on the bed. The proposed solution was to use a timer to check when the IR beams stopped being interrupted for a pre-defined amount of time in order to be certain that the athlete has definitely left the trampoline bed. This time interval needed to be bigger than the maximum amount of time that anyone could spend on the trampoline bed while jumping, but lower than the minimum TOF that an athlete needs to perform any trampoline skill. This way the algorithm could differentiate between when the IR beams are uninterrupted and the athlete is on the trampoline bed (less than the pre-defined time) or when the IR beams are uninterrupted and the athlete is in the air (more than the pre-defined time).

Based on all this information, the code implemented on the MSP430 and shown in Listing B.3 (Appendix B) and the code implemented on the ESP module and shown in Listing B.4 were developed. The corresponding flowcharts are shown in Figure B.1 and Figure B.2 (Appendix B).

Starting with the MSP430 code, the program begins by configuring the necessary components: CPU clock calibrated to 16MHz; timer0 sourced from the ACLK with the VLO at 12kHz; pin 1.0 as output; USCI_B in I2C master mode with pin 1.7 as SDA and pin 1.6 as SCL at 400kHz speed (maximum speed supported); and USCI_A in UART mode with pin 1.1 as RX and pin 1.2 as TX at a 115200 baud rate.

After this, the program enters an infinite loop, polling all of the IRM-Rx sensor values, using the I2C bus. This polling process is divided into two "`for`" loops, corresponding to the IRM-Rx in the X-axis and in the Y-axis. As described above, the program is searching for the first IR beam to be interrupted in each axis, which will indicate that an athlete has landed on the trampoline bed and the HD area where the athlete landed on. When an IR beam is interrupted (`RXData == 8`) the program saves the corresponding IRM-Rx I2C address by leaving the "`for`" loop of the corresponding axis, preserving the iterator value (`x` or `y`). It then starts to only poll the IRM-Rx from the remaining axis to search for the first beam to be interrupted in that axis. In the case of the Y-axis `for` loop, when an IR beam is found to be interrupted, it stores the value of the counter from timer0 (`TA0R`), which will be translated into the TOF in the ESP module, and the actual I2C address of the Y-axis IRM-Rx (`currentAddr`). When it finds the first IR beam to be interrupted in each axis (`foundXFlag` and `foundYFlag` are set) the program sends the HD area code and TOF data (`areaCodeList[x][y]` and `tofInClockCycles`) via the serial connection to the ESP module. The values in the `areaCodeList[6][4]` vector correspond to the areas shown in Figure D.2, which will be described later in Subsection 4.4.2. The program then enters a function called `pollForTakeOff()` to detect the exact moment when the athlete leaves the trampoline bed.

In this function, the program is basically polling only one Y-axis IRM-Rx (`currentAddr`) to check when it stops being interrupted for more than 400 milliseconds (the so-called pre-defined time interval), meaning that the athlete has definitely left the trampoline bed. The value was chosen based on several tests that were conducted and will be described further in this thesis. When the IR beam is uninterrupted (`RXData == 0`), the program starts timer0, as there is a possibility that the athlete left the trampoline bed. It then continues to poll the IRM-Rx such that if the IR beam is uninterrupted for 400 milliseconds (`TA0R >= 4800`) the athlete has definitely left the trampoline bed, if the beam becomes interrupted again then the athlete is still on the bed and the polling must continue. For saving the TOF counter value and checking when the athlete leaves the bed, the IRM-Rx of the Y-axis were chosen because it has more IR beams per centimeter which means that it measures a more accurate moment of when the athlete arrives and leaves the trampoline bed.

Also important to note, is that the Watchdog timer is used in this code to ensure that during the I2C communication the program doesn't freeze if anything is wrong with the I2C bus. The Watchdog timer was defined to restart after 43 milliseconds, using the ACLK as the clock source.

In terms of the ESP module, the code developed is fairly simple as it is based on two pre-made libraries for the ESP8266 family, the ESP8266WiFi and ESP8266HTTPClient libraries. The program starts by opening a serial communication at 115200 baud rate and connecting to the Wi-Fi network "Pi_AP" created by the APS. It then enters an infinite loop waiting for the HD area and TOF data of each jump to be sent by the MSP430. After reading this data, the program converts the TOF, that is in timer0 cycles, into seconds and creates a message with a predefined format: "areaCode/TOF(seconds)/jumpTmpID/trampID". Each jump has a temporary identifier that rolls over at 100 and each GC or trampoline has a permanent identifier. This message is then sent to the APS using a Hypertext Transfer Protocol (HTTP) POST request directed at the Uniform Resource Locator (URL) "http://192.168.42.1:5000/post_sensor".

4.3.3 GC Prototyping, Testing and Final Implementation

Before implementing the final PCB of the electrical circuit represented in Figure 4.6, some requirements of the GC, which controls the entire Infrared Beam Grid (IR-GRID) system, needed to be tested. The two main problems were the resolution of the system, which was directly related to how fast the GC could poll/read the sensors of all ten IRM-Rx and if the theory about how the IR beams behave when an athlete lands on the trampoline bed is correct or not. To do these tests, a prototype of the GC was developed using the MSP430 LaunchPad Development Kit with the MSP430G2553 microcontroller, connected to the NodeMCU Development Kit and to a breadboard with the R3 and R4 "pull-up" resistors and the two RJ11 connectors. This setup is shown in Figure 4.8.

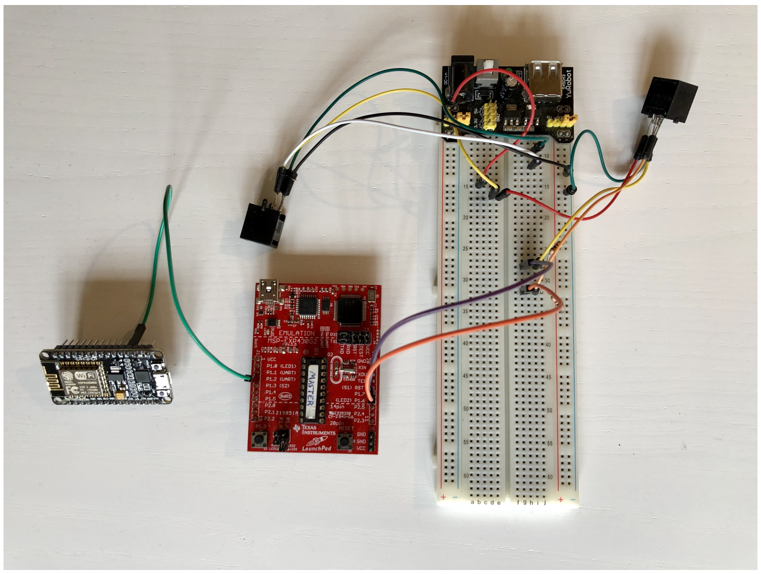
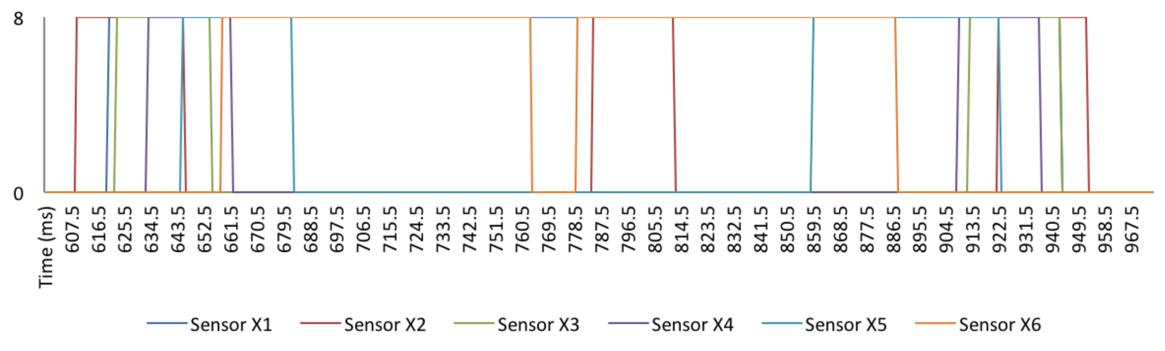


Figure 4.8: Prototype of the GC for testing.

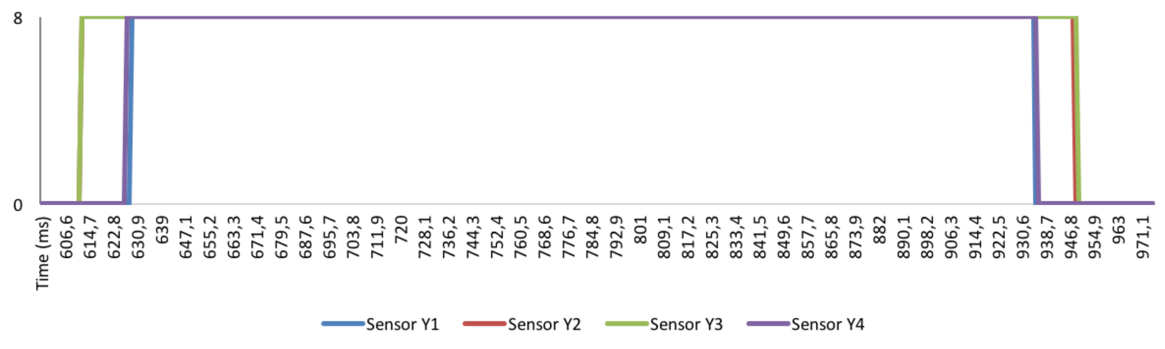
The first test was done by connecting all 10 IRM-Rx and IRM-Tx (aligning them with each other) to the GC and using the code shown in Listing B.3. By alternating pin 1.0 between “high” and “low”, in the beginning of the main loop, and using an oscilloscope, it was possible to determine how long each complete main loop took (poll all ten IRM-Rx). This is the critical part of the code as it determines the granularity/resolution of the sensor system. The results showed that each main loop took approximately 600 microseconds, meaning that each of the ten IRM-Rx I2C communications took 60 microseconds (the rest of the code isn’t significant for this measure, as each CPU clock cycle takes 62.5 nanoseconds). This value is well within the determined requirements for TOF. In terms of HD, only further testing can determine if it is quick enough to meet the desired requirements. During this test it was also determined that the value of the R3 and R4 “pull-up” resistors needed for the I2C bus to function correctly was 1kΩ. Also, this test was conducted using a AC/DC voltage transformer of 12V and 5A. The voltage measured at the furthest IRM-Tx was 6.8V, which is still within the requirements for the chosen low dropout linear voltage regulator to function correctly.

For the second test, a new code for the MSP430 and the ESP module was developed. The main idea of this program was to constantly read the values of all the IRM-Rx, send each reading via the serial connection to the ESP module and then send it to the computer where a Python program would save the data in an Microsoft Excel file. The same hardware setup as before was used but the ESP module was connected via USB to a computer and the IR-GRID was actually mounted on the trampoline. A GoPro camera was positioned below the trampoline in order to catch the entirety of the bed. This camera filmed the athlete while she/he was jumping in order to validate the data from the IRM-Rx sensors with the HD area where she/he landed after each jump. For each jump two line charts, similar to those shown in Figure 4.9, were obtained.

These particular charts show the IRM-Rx sensor values when an athlete landed on area A4 on both the X-axis (Figure 4.9a) and Y-axis (Figure 4.9b). Each color represents an IRM-Rx sensor where a value of “8” means that the corresponding IR beam is interrupted and a value of “0” means that the beam is uninterrupted (there are no intermediate values). According to the architecture described in Subsection 3.2.1, if an athlete lands on area A4, the first IR beam to be interrupted in the X-axis should be the one in IRM-Rx X2 and in the Y-axis should be the one in either IRM-Rx Y2 or Y3. After analyzing the charts, we can assume that this theory was indeed correct: (1) all the sensors start with a value of zero which means the athlete is in the air; (2) when the athlete lands on the trampoline bed the beams start to become interrupted by a certain order, depending on where the athlete landed on; (3)



(a) X-axis IRM-Rx sensors.

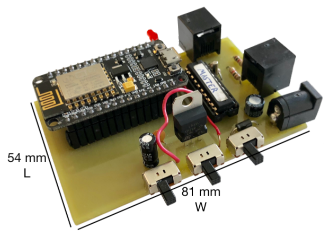


(b) Y-axis IRM-Rx sensors.

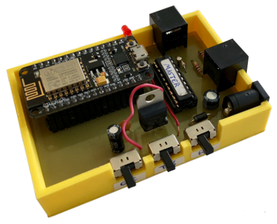
Figure 4.9: Line charts of the IRM-Rx sensor values during one jump (landing on area A4). Each sensor has a value of “8” when its IR beam is interrupted and “0” when its IR beam is uninterrupted.

while the athlete is on the bed, some of the IR beams alternate between uninterrupted and interrupted; and (4) when the athlete leaves the trampoline bed all the beams become uninterrupted in the inverse order from when they became interrupted initially. Another insight from these tests was that, when the athlete was on the trampoline bed, no IR beam became uninterrupted for more than 400 milliseconds. One important note is that these tests were done with a GC resolution of 900 microseconds due to the extra data being transfered via the serial connection from the MSP430 to the ESP module.

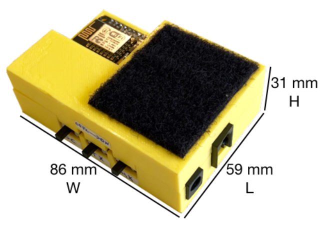
After passing all the tests, a PCB layout was developed, based on Figure 4.6. The main requirements for the PCB were that it needed to be as small as possible and the RJ11 connectors should be placed taking into account the configuration of the IR-GRID, shown in Figure D.1. The final PCB with all of the electrical components soldered to it is shown in Figure 4.10a, measuring 54 millimeters in length and 81 millimeters in width. In the case of the GC, the PCB was made in the Taguspark laboratory by the author of this thesis.



(a) GC PCB with all electrical components soldered to it.



(b) GC PCB inside the bottom part of the GC box.



(c) Full GC box encapsulating the GC PCB.

Figure 4.10: Hardware implementation of the GC and all of its components (PCB and box).

To encapsulate the GC PCB, a box was designed and 3D printed using PLA material. This box needed to have openings for the two RJ11 female connectors, for the three ON/OFF switches and for the LED. The final version of the box and how the PCB fits inside it is shown in Figure 4.10b and Figure 4.10c, measuring 86 millimeters in width, 59 millimeters in length and 31 millimeters in height.

Another important requirement for the system was to keep the cost of production as cheap as possible. Table A.3 (Appendix A) shows the GC BOM which was based on ordering the minimum quantities needed of each component from the cheapest supplier. The total cost of the materials for the GC is **9.73€**.

4.4 APS Design and Implementation

The main purpose of the APS is to wirelessly bridge the user with the trampoline sensor system. It serves as a Wi-Fi access point, creating a network to which both the GC and the UT connect to. The APS also contains a server that has a WebApp which stores the data from the IR-GRID and displays it in the UT.

This architecture (see Figure 3.1) was chosen over simply wirelessly connecting the UT to the GC and having the application and data stored locally in the UT for one main reason: most trampoline clubs don't have the money or conditions to buy a medium spec computer and have it permanently in the gym. This means that they will most likely use the athletes' or coaches' personal devices (laptop, tablet or smartphone) as the UT. It would be imprudent to have the data and the application stored in a personal device and being dependent on its owner to bring it every day to the gym. Having a separate device to store the data and the application gives a lot more flexibility to the system in terms of what the user can utilize as the UT. Another advantage is that this way it is a lot easier for the coach to take the data with her/him to analyze outside the gym. It is worth mentioning that a cloud based service to store the application was discarded as most clubs do not have good or any Internet connections.

The wireless technologies chosen for the system needed to meet the following requirements: reliable wireless communication; cheap hardware; good support and documentation online; easy way to create a good GUI; and data storage capabilities. Taking into consideration all of these requirements, the best options were Wi-Fi coupled with HTTP server-client based Internet protocols and WebApp software frameworks. This means that the GC, the UT and the APS needed to have hardware with Wi-Fi capabilities, the GC and the UT needed to be configured as HTTP clients and the APS needed to be configured as both a Wi-Fi network Access Point (AP) and a web server containing a WebApp. By using these technologies, the application is cross-platform, meaning that the UT can be almost any device that has Wi-Fi and an Internet Browser (desktop computer, laptop, tablet or smartphone).

The hardware chosen for the APS is the **Raspberry Pi (RPi) 3 Model B** [37][38]. This device was chosen because it is relatively cheap, it has Wi-Fi already integrated, it has a powerful CPU and it has a lot of support and documentation online.

4.4.1 APS Software Design

In terms of the software, the back-end programming language chosen was **Python** [36] and the web framework chosen was **Flask** [32][20]. The front-end languages used were **Hypertext Markup Language (HTML) 5** [58], **Cascading Style Sheets (CSS) 3** [57] and **JavaScript** [29].

Flask is a Python web microframework that was developed to be extended, which means that it only includes the essential core that all WebApps need. The developers can then choose which extensions they want to use for their specific application, such as database engines, web form validation systems and user authentication systems. The Flask core is dependent on two main packages, Jinja2 [6] for the HTML templates and Werkzeug [51] for the routing, debugging and Web Server Gateway Interface (WSGI) [35]. This last package is a specification that describes how a web server communicates with WebApps using the Python programming language.

When a client, such as a web browser (UT) or Wi-Fi enabled device (GC), sends a request, the web server uses the WSGI protocol to encapsulate that request into several Python objects and sends them to the Flask WebApp. The application will then look up the URL requested in the application's URL map to determine which function will handle it. This is done by defining *routes* (the association between an URL and a function to handle it) as *view functions*. These functions process the necessary data for the client's request and return a response that usually is an HTML file to be displayed in the client's browser. An example of a simple *view function* is shown in Listing 4.1.

Listing 4.1: Example of a Flask *view function*. The URL called 'url_address' is mapped to the *view function* called viewFunctionName. The *view function* returns the HTML file called 'html_file_name.html' to the client who sent the request.

```
@app.route('/url_address')
def viewFunctionName():
  #Other code to process data and other tasks
  return render_template('html_file_name.html')
```

As described in Chapter 3, the application needs to have the following main features: (1) store the TOF and HD data of the most recent jump performed; (2) store athlete profiles with relevant personal information, routine information and a history of all saved routines performed; (3) have an interface for creating and managing new athlete profiles; (4) display, filter and organize the history of all saved routines performed by a certain athlete; (5) controls to start and stop the display of jump data for a certain routine and a control to save the routine data; and (6) have a way to graphically and intuitively display all the TOF, HD and routine data for a specific athlete. The final application that was developed contains more features to improve the user experience, but only these six will be described here.

In order to store this data, a database system needed to be chosen. Based on the prior knowledge of the author of this thesis, a Structured Query Language (SQL) database based on the relational model was chosen. The database abstraction layer chosen for this application was SQLAlchemy [46], which was adapted for Flask in the extension Flask-SQLAlchemy [5]. In order to meet the requirements referenced above, the following relational database diagram, represented in Figure 4.11, was designed.

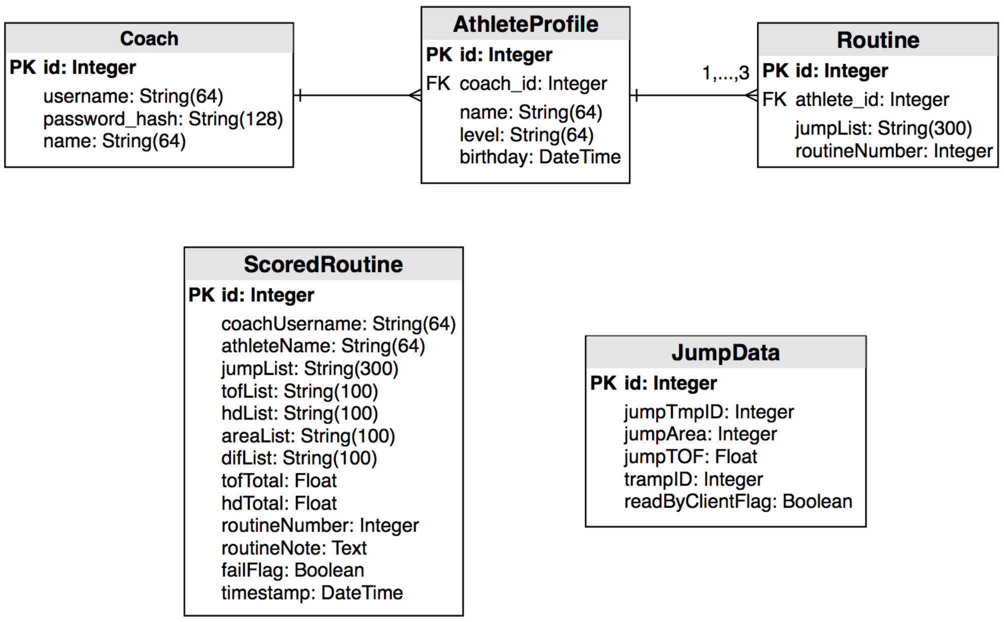


Figure 4.11: Diagram representation of the relational database in the APS WebApp.

The main idea behind the organization of this database is as follows:

1. There is a table for "Coach" that is characterized by a "username" (unique), a "password_hash" (unique) and a "name" to log into the application.

2. A “Coach” can have many “AthleteProfile”, where each is characterized by a “name”, a “level” and a “birthday”.

3. An “AthleteProfile” can have a maximum of three “Routine” (in average a trampoline athlete only has three different routines at any given time of her/his career), where each is characterized by a “jumpList”, which contains a *String* with the names of every skill in the routine, and a “routineNumber”.

4. After each jump, the GC sends the corresponding HD and TOF data to the APS, where it stores them in the “JumpData” table, which is characterized by having a “jumpArea” and a “jumpTOF”. This table always only has the most recent jump performed (only one record). Each new jump overwrites the previously stored one.

5. After a routine is performed and all the TOF and HD data from each skill has been sent from the IR-GRID to the APS and finally to the UT, the user can save that data into a table called “ScoredRoutine”. This table is characterized by a “coachUsername”, an “athleteName” and other routine relevant data like “tofList”, “areaList” and “hdList”. To save a routine, this data has to be transferred from the UT back to the APS.

6. Each table’s primary key (PK) is called “id”, which is an *Integer* that uniquely identifies every record in each table.

There are also other attributes for each table that were not referenced as they are not very relevant for the main data organization objective of this specific application.

In terms of the *view functions*, there are seven main ones that are summed up in Listing B.5. The `login()` *view function* has to do with displaying the login page and validating the credentials that are submitted by the user to login. It returns the `‘login.html’` file. The `appPage()` *view function* is related with displaying main application, which has the controls to start/stop displaying the jump TOF and HD data to the user, save the routine data, create a new profile, ... It returns the `‘gui.html’` file. The `getScore()` *view function* has the objective of getting the most recent skill data from the “JumpData” table and sending it back to the client browser to be displayed in the GUI. This transaction is done using JavaScript, particularly Asynchronous JavaScript And XML (AJAX), which is a way to partially update web pages asynchronously, by exchanging data with a web server “behind the scenes”, i.e. without reloading the whole page. This *view function* returns a *Dictionary* data type that contains all the TOF and HD data for the most recent jump performed. The `postSensorJumpData()` has to do with sending the TOF and HD data of the last jump performed from the GC to the APS and storing it in the “JumpData” table. It returns a simple `String` so that the GC can confirm that the data was received by the APS. In the `saveRoutine()` *view function*, the TOF and HD data from the whole routine can be saved when the user presses the SAVE button in the main application page. This data is stored in the “ScoredRoutine” table along with a timestamp from the client browser. This transaction is also done via AJAX and it returns a simple JSON text for the client browser to know that the data was successfully received. The `profilePage()` *view function* is used for displaying the athlete’s profile page with the data that is stored in the “AthleteProfile” table. It returns the `‘profile.html’` file. Finally, the `profileHistoryPage()` *view function* is related with displaying, organizing and filtering all of the routine history of a specific athlete, which means displaying all of the records stored in the “ScoredRoutine” table of that athlete, under the current coach login session. It returns the `‘profileHistory.html’` file.

In terms of the server used, Flask comes with an in-house development web server for prototyping, which was capable enough for the desired application.

4.4.2 APS Testing, GUI and Final Implementation

One very important part of designing a system that collects data and will be used by "everyday non-engineering" users is the GUI. As important as gathering valid, accurate and precise data is the way the data is presented to the end user. In fact, the way the user can visualize and organize the data is very important in order to get practical and relevant insights that can help, in this case, the athletes and the coaches improve their performance. Understanding this, the GUI for this application was developed with the consulting of several international and Olympic level athletes and coaches from Portugal. Appendix C has screenshots of all the pages in the WebApp.

Figure C.1 shows the first page the user is greeted with, containing the login interface. If the user is not registered yet, she/he also has the option to do that. By pressing the "Register" button, the application takes the user to the register page, shown in Figure C.2. Here, the user must introduce a name, an unique username and a password.

After logging in, the user is taken to the main and most important page, shown in Figure C.3. In the top-right area of the page there is a bar graph that displays the TOF of each jump performed as well as the final total. Values in green represent jumps that have a greater TOF than the immediately previous one and values in red have the opposite meaning. In the bottom-middle area there is a trampoline representation that displays the location where the athlete landed on in each jump, with a small cross. Different colors on the crosses indicate different HD deductions: green means 0.0 deduction (i.e. no deduction), yellow 0.1, orange 0.2 and red 0.3. As explained before, there are only eleven different areas but due to the IR-GRID configuration, the system can actually differentiate between 24 "virtual" areas that are shown in Figure D.2. Given this extra capability, the GUI can also display the crosses on all 24 areas as it gives the user a better notion of where the athlete actually landed on. In the bottom-right corner, there is a table that contains all the TOF and HD score information, per jump, for the routine. It is important to remember that the total HD score shown in the table is actually 10.0 minus the sum of the HD deduction per jump, which is how it appears in a Trampoline competition. This decision was suggested by the consulting coaches, as they found this way (opposed to showing the HD deduction sum) easier for training. There is also a button for the coach to add relevant notes for the specific routine. The controls related with the athlete profile are in the top-left corner, where the user can select which athlete will perform which routine, create a new athlete profile and view the information of the selected athlete profile. When a athlete profile and routine is selected, the corresponding skill names appear on the score table. In the bottom-left corner are the main commands that are used to start/stop displaying the TOF and HD data of the jumps performed on the trampoline and to save the routine data that was performed. When the user presses the START button, the JavaScript code starts to poll the "JumpData" database table, every 300 milliseconds, in order to check whether or not a new jump has been performed. When a new record is added to the table, the application sends that data to be displayed in the GUI. After 10 different jumps, the JavaScript automatically stops polling the server. The user can also end this process prematurely by pressing the STOP butotn. When the SAVE button the pressed, the data on the score table is sent back to the server and saved in the "ScoredRoutine" database table.

To create a new athlete profile, the user presses the New Profile button and an overlay appears, as shown in Figure C.4. Here, the user can fill the athlete's name, level and routine skill names. Each routine column has buttons to clear, copy and paste the entire column text. This is important because many athletes have similar routines, differing in only one or two skills between them. There is also an autocomplete function that helps writing the names of the trampoline skills in the columns. By pressing the Create button, a new record on the "AthleteProfile" table is added.

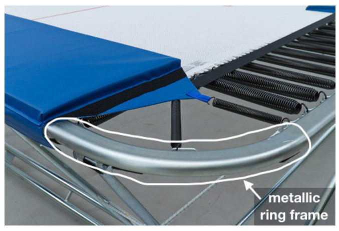
When the user presses the View Profile button, the profile information page, shown in Figure C.5, is loaded. This page shows a summary of the athlete's profile information. The top-right yellow button takes the user to the edit profile page, as shown in Figure C.6. Here the user can edit all the parameters of the athlete's profile, including deleting that specific profile.

The top-right green button in the profile page loads the profile history page, as shown in Figure C.7. This page is very important for the coaches and athletes that want to have a more in-depth perception of how they are performing. The right side of the page displays all the score tables of the saved routines, with their corresponding timestamp. Even failed routines (routines with less than tens jumps performed) can be saved and are represented with a red background. In the top-left part of the page are the filtering and ordering options. The user can filter the data by a specific routine number (All, R1, R2 or R3), and or by a time interval. Concerning the ordering of data, there are four options plus a special one. The four are: order by date (from most recent to oldest), by TOF (highest to lowest), by HD (highest to lowest) and by both, which sums the total TOF and HD of each routine and orders them from the highest to the lowest sum. The fifth special option is one called Analyze, that attempts to help coaches and athletes get valuable insights from their routine history. It uses the routines filtered from the above options and makes an average of each jump's TOF and HD scores, and calculates the mode of the area where each jump lands on. This way the user can find patterns/tendencies that the athlete has during the routine. For instance, one may notice that, on the last jump of the routine the athlete has a tendency to land on area A7, getting an HD deduction of 0.2. On the top right corner of each score table division there is a purple button that displays an overlay with the graphical representation of the scored routine, as shown in Figure C.8.

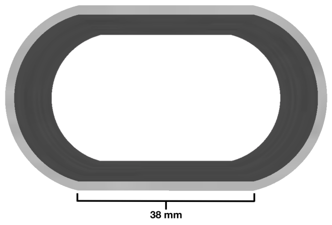
To encapsulate the RPi, a box was 3D printed using a design available online in PLA material. In terms of cost, as shown in Table A.4, the total is **36.62€**.

4.5 Final System Design and Implementation

One last important problem needed to be solved to finalize the design of the whole system: where and how to attach the IR-GRID to the trampoline in an universal (i.e. one fit for most Olympic sized trampolines) and non-invasive way? The only structure where the IR-GRID could be placed without changing the elastic/bouncing properties of the trampoline is the metallic ring frame that goes around the trampoline and holds the springs and bed in place (see Figure 4.12a). This frame has an oval shape with straight sides, as shown in Figure 4.12b. With this shape, the IR-GRID modules can be attached below the ring frame. This surface is flat which guaranties that the IRM-Rx and IRM-Tx are leveled and aligned, which is very important.



(a) Picture of the metallic ring frame around the trampoline that holds the springs and bed in place.



(b) 3D representation of the shape of the metallic ring frame (side section cut).

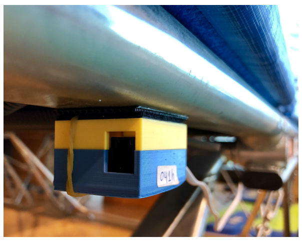

(c) Image of how the IR-GRID modules are attached to the metallic ring frame.

Figure 4.12: IR-GRID modules attachment to the metallic ring frame of the trampoline.

In terms of how to attach the IR-GRID modules, many options were considered. The main requirements were that modules needed to be fixed in all three dimensions, as vibrations could cause them to misalign, and to be able to attach on any brand of Olympic sized trampolines. The first option tested was to use 3D printed clamps that would wrap around the metallic ring frame and have latch to mechanically attach the IR-GRID modules. This solution was very good at keeping the modules strongly fixed and aligned with each other. However, even though all Olympic trampolines have a frame with the same shape, their sizes may vary, implying that it would be necessary to make custom clamps for each trampoline brand and model. Another problem with this design was that the clamps could only be positioned in some specific locations on the ring frame due to other fixtures, like the spring hook rings that are welded on to the frame, which made configuring the IR-GRID modules, as shown in Figure D.1, very difficult. The second solution considered was to use magnets on the IR-GRID modules. This solution was good in terms of where the modules could be positioned, not having any limitations due to other fixtures on the metallic ring frame. The downside was that the magnets create most of their attracting force in the vertical direction, meaning that the vibrations of the trampoline could cause the modules to twist and become misaligned. Besides, strong enough magnets can be very expensive. Finally, the chosen option was to use velcro, where one side would be glued to the metallic ring frame, on correct locations (Figure D.1), and the other side on the IR-GRID modules. This solution offers the best characteristics of both previous options: the modules can be freely positioned in any location of the metallic ring frame without any limitations; it is independent of the dimensions of the ring frame shape (universal between trampoline brands); and offers enough fixing strength across all three dimensions. The main downside of this solution is that the velcro must be glued on the ring frame by hand without any physical guide to help the user know precisely where to position it. Also, the IR-GRID modules are aligned and attached to the ring frame manually which can cause some difficulties during this process. After testing this attachment method, it was found that the manual alignment of the IR-GRID modules was not complicated, as it took about seven minutes to mount the entire system. This is aided by the wide diameter of the IR beam which does not mandate the IRM-Tx to be perfectly aligned with the IRM-Rx. There are many types and brands of velcro, but in average it is relatively cheap to buy. Figure 4.12c shows a picture of how the IR-GRID modules are attached to the metallic ring frame.

To finalize the system design, the last hardware needed were the **RJ11 cables** to connect all the IR-GRID modules with the power and I2C bus lines, the **RJ11 male connectors** and the **AC to DC 12V power supply**. Table A.5, shows the final cost for the hardware of the entire system at **178.76€**. Figure 4.13 shows an image of the complete hardware implementation of the designed system.

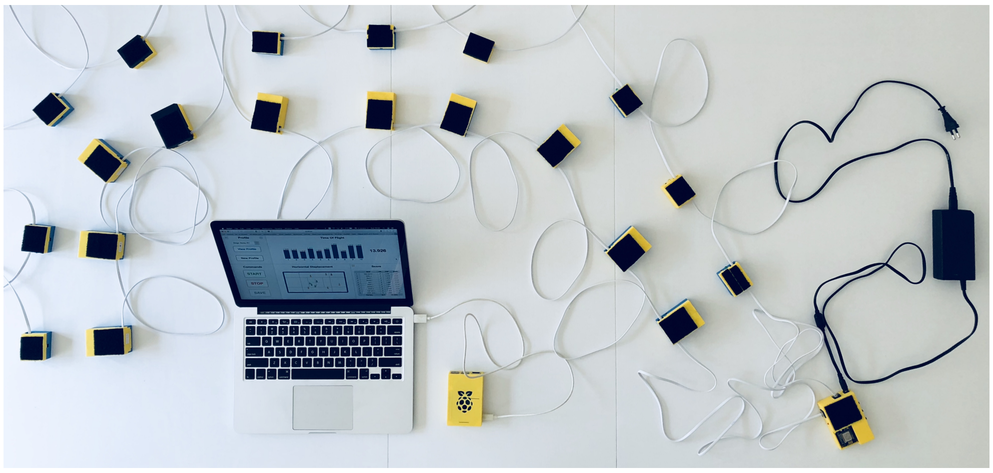


Figure 4.13: Picture of the hardware implementation for the complete system.

Although some simple experiments were performed in order to design the system, further testing needs to be conducted in order to better characterize how the system works and which are its limitations. In the next chapter, these experiments and results will be described in more detail.

5 Testing and Results

Contents

In this chapter, the main tests that were done with the complete system will be described. The main goal was, of course, to check whether the system was working correctly, in a “real world” environment, and if it met all the requirements proposed in Chapter 3. It was also important to characterize how the trampoline behaves when an athlete is jumping, which variables influence that behavior and how that limits the capabilities of the system. These experiments were mostly conducted to understand the system’s capability of measuring the Horizontal Displacement (HD), as the Time of Flight (TOF) was a easier problem to solve and validate. All the tests were conducted in the Sporting Clube de Portugal Trampoline Gymnasium using an Olympic size trampoline, with the help of national team level athletes and coaches, in order to get valuable feedback from them.

5.1 Trampoline Bed Behavior Characterization

The main variables that may influence the behavior of the trampoline and therefore the output of the system were the jumping height, the location where the athlete lands on the bed, the athlete’s weight and the trampoline model. Given the nature of the sport and the resources available, conducting the experiments in a controlled environment, where all variables can be accounted for, is virtually impossible. For instance, it is impossible for an athlete to jump at a determined height and land in an exact spot several times in a row. Given this, some assumptions and approximations needed to be made. Theoretically the jumping height is the variable that influences the most the way the trampoline bed deforms when an athlete is jumping on it, so two intervals that can be easily controlled by the athlete were considered: less than two meters and more than two meters, which will later be referred to as “low” and “high”, respectively. To do this, a visual reference point in the gym was used to aid the athlete at maintaining her/his height. In terms of the location where the athlete lands on, given that the trampoline is symmetrical, only areas A0, A1, A3, A4 and A5 were tested (see Figure D.1). Even though, the athlete weight should not influence too much the behavior of the trampoline bed, it was always the same athlete (male), with approximately 67 kilograms, to perform the experiments. The trampoline model used was also always the same, an Eurotramp Ultimate 4x4 Trampoline [15], which is the latest model to be used in the Olympic Games. In order to validate the data read by the Infrared Beam Grid (IR-GRID) sensors, during all tests, a fish-eye camera (GoPro) was positioned below the trampoline so that it could capture all of the bed areas, as shown in Figure 5.1.

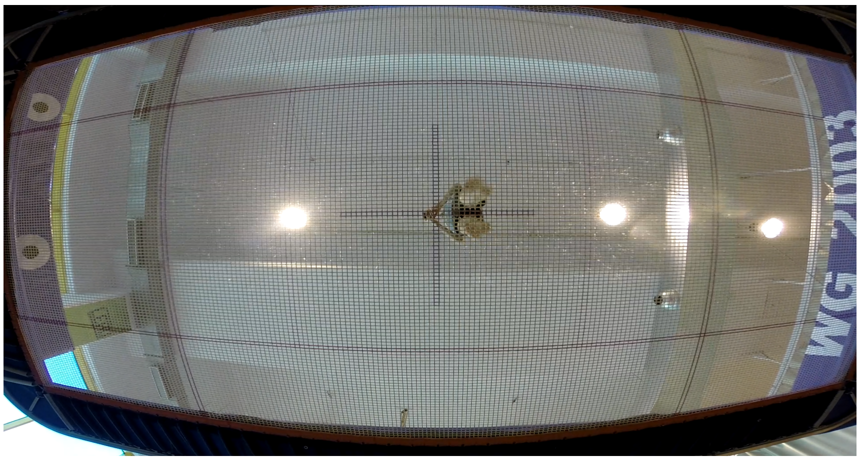
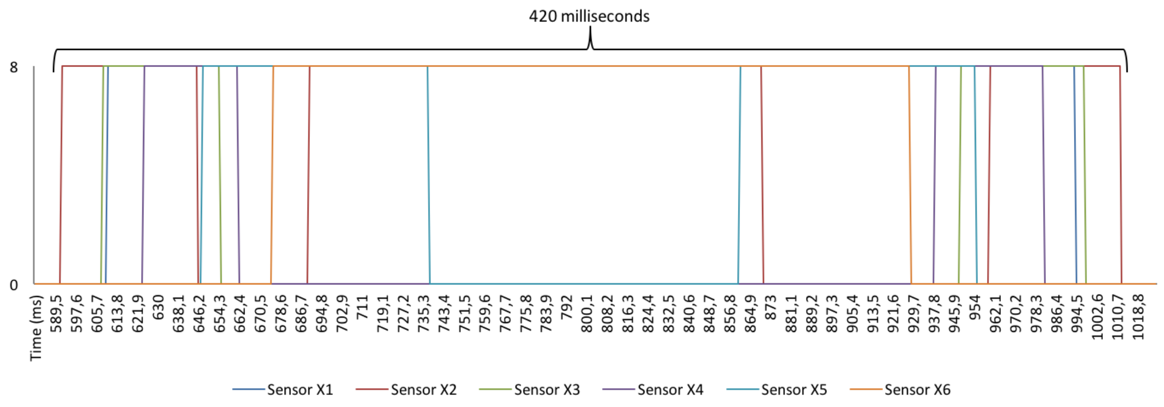


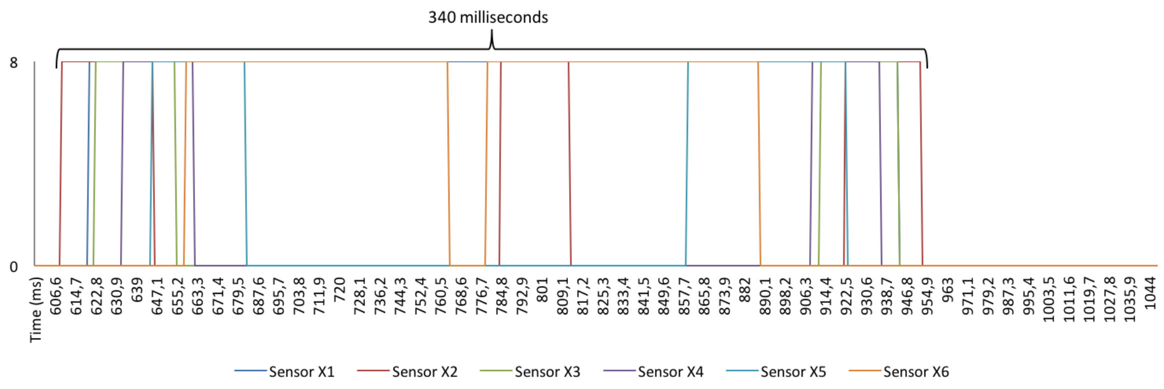
Figure 5.1: Picture of the GoPro point of view. The “fish eye” lens enables all trampoline bed areas to be visible.

This first phase of testing was to gather the Infrared Beam Module Receiver (IRM-Rx) sensor values, while an athlete was jumping, and cross-reference that data with the video from the GoPro camera. This experiment was already briefly explained in Subsection 4.3.3, but the main objective here was to characterize the behavior of the trampoline bed and which variables can influence it. The IR-GRID was mounted on the trampoline frame (see Figure D.1) and the IRM-Rx sensor values were read by the Grid Controller (GC) and sent to a computer via USB. In this test, the resolution of the system (i.e. the time interval to read all 10 IRM-Rx sensors) was 900 microseconds. The athlete was asked to jump ten times on areas A3, A4 and A5, at both the “high” and “low” heights. In areas A0 and A1, the athlete was only asked to jump at less than two meters, as it can be very dangerous to jump higher on those areas.

As described in Subsection 4.3.3, the data obtained clearly shows how the trampoline bed deforms as the athlete lands on it. This is evident by the order in which the infrared (IR) beams are interrupted, the first ones being those closest to the center of pressure of where the athlete lands on the trampoline bed. After analyzing the data from all 80 jumps, one can conclude that this behavior appears to be the same in all bed areas, in both the X and Y axis and at both jumping heights. Regarding the jumping height, it was observed that the higher an athlete jumps, the less time she/he actually spends on the trampoline bed. This is somewhat counter intuitive, as the bed actually deforms more (i.e. it goes lower) as an athlete jumps higher. A probable explanation is that due to the increase in the athlete’s falling speed, the bed actually deforms faster, which results in less time on the trampoline. Figure 5.2 illustrates these conclusions with the data from a jump where the athlete landed on area A4 and was jumping at “low” and “high” (Figure 5.2a and Figure 5.2b, respectively). This behavior was similarly observed in both the A3 and A5 areas.



(a) Athlete jumping at less than 2 meters of height.

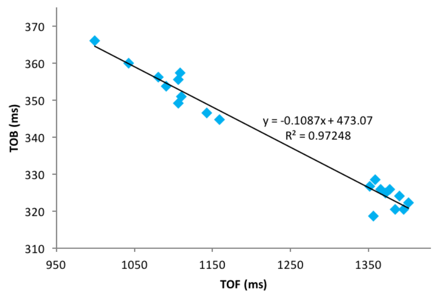


(b) Athlete jumping at more than 2 meters of height.

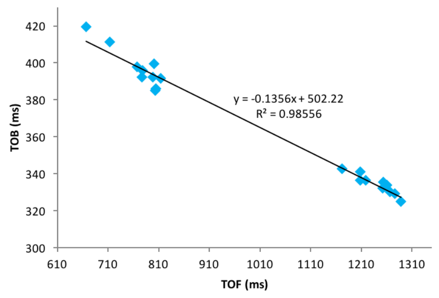
Figure 5.2: Chart of the X-axis IRM-Rx sensor values during one jump (landing on area A4). Each sensor has a value of “8” when its IR beam is interrupted and “0” when its IR beam is uninterrupted.

From Figure 5.2 one can see that, when the athlete was jumping low (with a TOF of 666.9 milliseconds), the Time on Bed (TOB) was 420.0 milliseconds, and when the athlete was jumping high (TOF of 1171.8 milliseconds), the TOB was 340.0 milliseconds. The relation between the TOF and the

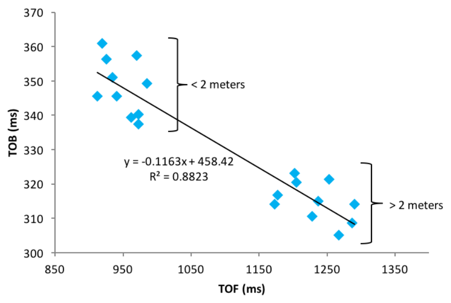
TOB can be observed more explicitly in Figure 5.3, where a linear relationship between the two can be inferred. In the charts for areas A4 and A5 (Figure 5.3b and Figure 5.3a respectively) this is very visible, as the coefficient of determination or R-squared value of the linear regression is 0.98556 and 0.97248, respectively. This is not so visible in the chart for area A3, where the R-squared of the linear regression had a value of 0.8823. This is probably due to the fact that area A3 is closer to the limit of the trampoline bed, where the springs exert non-symmetrical forces, differing a lot from when the athlete is closer to the center of the bed (areas A4 and A5).



(a) Area A5.



(b) Area A4.



(c) Area A3.

Figure 5.3: Charts showing the relationship between TOF and TOB values in areas A3, A4 and A5 at the heights of less and more than two meters. The data was organized in scatter charts and a linear regression was done in each one.

Even though knowing how the TOB varies with the TOF is relevant to characterize the trampoline bed's behavior, what is most important to understand for this application is that, as an athlete jumps higher, the TOB becomes lower, and therefore the time interval between consecutive IR beams becoming interrupted also becomes lower. This is critical, as the maximum resolution of the system is 0.6 milliseconds, which means that if two consecutive IR beams become interrupted in less than that time, the system cannot differentiate between those two events.

This effect can easily be seen in Figure 5.2, where the IR beams in both charts are interrupted approximately in the same order, but the time interval between those events is shorter when the athlete jumps higher (more than two meters). In average, the TOF when the athlete was jumping "low" in area A4 was 772.29 milliseconds, and the difference between the first and second IR beam becoming interrupted (in the X-axis) was 12.15 milliseconds. In contrast, when the athlete was jumping "high" in area A4, the average TOF was 1240.47 milliseconds, and the difference between the first and second IR beam becoming interrupted (in the X-axis) was in average 8.28 milliseconds. This means that an increase of 468.18 milliseconds in the TOF resulted in a decrease of 3.87 milliseconds in the time interval between the first and second X-axis IR beams becoming interrupted. An over-simplistic way to interpret these results is to assume that this relationship is linear. Thus, it would be easy to calculate that for a TOF of 2100 milliseconds (one of the highest single jump TOF that can be performed in a routine), the time interval between the first and second X-axis IR beams becoming interrupted would be 1.18 milliseconds, meaning that the system would always be able to measure the HD when an athlete landed in area A4. Unfortunately, this cannot be concluded as there are many other variables that influence the trampoline bed’s behavior. For example, the linearity assumed above was not formally proved and it probably varies depending on where the athlete lands on (like landing near the edges).

Another important characteristic in the trampoline bed’s behavior is that inside the same area, the difference in time intervals between when each IR beam is interrupted may also vary. This is most significant when an athlete lands closer to one of the trampoline area lines. For example, if an athlete lands on top of line LX1 (see Figure D.1), the time interval between when IR beams X1 and X2 become interrupted is almost zero. This makes it very hard for the system to correctly determine the correct HD score when the athlete lands very close to the lines. Given that there are no Fédération Internationale de Gymnastique (FIG) guidelines describing the resolution, accuracy and precision of an HD measuring machine, a 10 centimeter “gray area” to each side of the trampoline lines was defined (see Table 3.1). When the athlete lands on this region (their feet are already touching the lines), it is not demanded that the system correctly determine in which trampoline area the athlete actually is on. In the experiment described above, the athlete always landed with his center of pressure more that 10 centimeters away from each line. In order to better test the system’s capability to correctly determine the HD score when the athlete lands on the “gray area”, further experiments needed to be performed.

5.2 “Real World” Testing

After understanding the basic behavior of the trampoline bed and how it could affect the system’s performance, it was time to test it in a “real world” scenario. To do this, the entire system, including the IR-GRID, Access Point Server (APS) and a laptop computer as the User Terminal (UT), were setup in an Ultimate 4x4 Eurotramp Trampoline [15] at the Sporting Clube de Portugal gymnasium. The APS was positioned at a distance of approximately 16 meters from the IR-GRID. A GoPro camera was also positioned under the trampoline bed in order to capture all bed areas (Figure 5.1) and validate the HD results given by the system. This experiment was conducted during a real Trampoline Gymnastics training session, gathering a total of 80 jumps/skills, ranging in TOF from 1.247 to 1.814 seconds. Jumps form four different athletes were measured, who ranged in weight from 44 to 75 kilograms. Table 5.1 shows the results from this experiment. The system, in particular the UT, was used by two national team level coaches from Sporting Clube de Portugal.

Table 5.1: Results from “real world” testing of the system.

Location	Correct	Wrong	Inconclusive	Accuracy
More than 10 cm away from lines	60	0	0	100%
Less than 10 cm away from lines	16	2	2	80%

Each jump performed was divided into two groups: those where the athlete landed with his center of pressure more than 10 centimeters away from any area line (outside the “gray area”) and those where the athlete landed less than 10 centimeters of any area line (inside the “gray area”). As can be seen in Table 5.1, the athlete’s center of pressure was outside the “gray area” in 60 jumps, and the system correctly identified the HD area every time (accuracy of 100%). Also, the athlete landed 20 times inside the “gray area”, where in 16 of those jumps the HD area was correctly identified, in 2 jumps it was wrongly identified and in 2 other jumps the results were inconclusive (the human eye was not able to discern in which was the correct HD area).

After the experiment, the overall consensus from the coaches was that the system met all of the necessary requirements to be an useful tool to help athletes and coaches in a training scenario. These requirements mainly had to do with HD and TOF accuracy and precision, intuitive User Interface (UI) and routine data management. The only issue referred by the users was that when setting up the system for the very first time, the process was a bit complex and time-consuming. However, after the first time and having the velcro strips attached to the trampoline frame, it was relatively quick to mount and dismount the IR-GRID.

5.3 Limit Testing

In order to better understand the limits of the system when an athlete is jumping on the “gray area”, a limit test was performed. The same setup as Section 5.1 was used but this time with only three IR beams, X1, X2 and X3 (Figure D.1). Therefore, the system was able to have a resolution of 264 microseconds, instead of the 900 microseconds when using all 10 IR beams. The idea behind this experiment was to make the athlete jump as high as he could, landing as many times as possible on lines LX1 or LX2 and then measure the time interval between the moments when the first and second IR beams become interrupted in each jump. After the test, a total of 50 jumps were recorded with an average TOF of 1.208 seconds. Figure 5.4 shows the distribution of the time intervals obtained, ranging from 0 to 6.072 milliseconds, where 0 means that the beams were interrupted in the same 264 microsecond cycle.

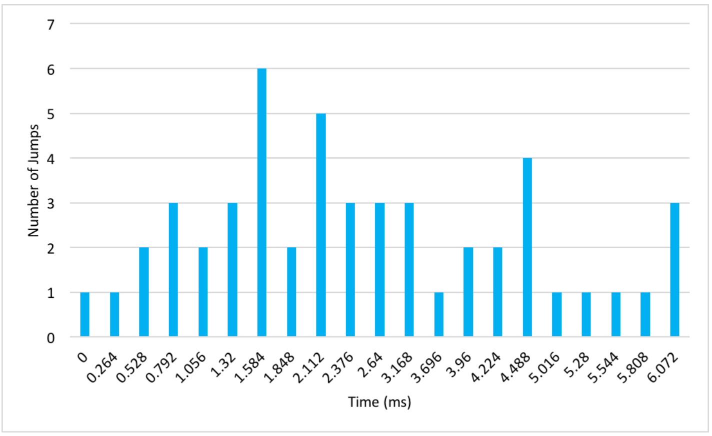


Figure 5.4: Chart with the distribution of the time interval between the moments when the first and second IR beams are interrupted. Athlete jumping on lines LX1 and LX2.

Taking into consideration that the system’s resolution for differentiating between IR beam interruption events is 600 microseconds, there are only four jumps out of 50 where the time interval would be too small for the system to detect. This means that even when an athlete lands with her/his center of pressure inside the “gray area”, at a 1.208 second TOF, the system should be able to correctly differentiate between HD areas 92% of the time. Of course there are many other variables that will affect these results, like different trampoline bed areas and jumping height, but it is interesting to know that the system has some resolution margin on top of the defined requirements.

6
Conclusions and Future Work

Contents

At the highest level of sports, the difference between ending in first or second place can be very small. This is particularly true for the sport of Trampoline Gymnastics, where jumping a little higher and more centered on the trampoline than your opponent can have a huge impact in the final score. This is why Sporting Technologies (STs) used in training can give coaches and athletes the edge they need to reach their objectives faster and more efficiently. By measuring and analyzing certain parameters, these technologies can help coaches and athletes identify hidden problems and develop new training methods, which would be impossible otherwise. In the case of Trampoline Gymnastics, two parameters were chosen to be measured: the Time of Flight (TOF) and the Horizontal Displacement (HD).

At the time that this work was started, there were no commercially available systems that could automatically measure these two components in a way that was non-invasive, and there was almost no prior work done in this area. An architecture was designed to meet the minimum requirements so that the system could be useful for the athletes and coaches during training. The system is mainly comprised of a infrared (IR) grid that is positioned under the trampoline bed and gathers data relating to how it deforms when an athlete is jumping. This data would be used to calculate the TOF and HD of each individual skill. As there were almost no scientific studies on modeling the behavior of a trampoline when an athlete is jumping on it, many tests were performed in order to validate what was theorized. The data gathered by the system is then sent to a database to be saved and visualized by the user. A User Interface (UI) was developed to control the entire system and to manage the stored data.

After designing, developing and implementing the hardware and software needed, a functioning prototype of the system was created. In order to validate the systems capabilities and to see how it behaves in a "real world" use case, experiments were conducted in the Sporting Clube de Portugal trampoline gymnasium where national team and Olympic level athletes and coaches train. Most of the data gathered was used to help better understand how the trampoline bed behaves as an athlete jumps, in order to improve the systems performance. Several athletes and coaches from Sporting Clube de Portugal used the system and the recurrent opinion was that it could be a very helpful tool to improve their performance. The application's UI was also praised as it was very intuitive and displayed all the relevant data in a way that made it easy for coaches to gain important insights about an athlete's performance. By being able to store a history of the routines performed by each athlete, the coaches can find patterns in the athlete's jumping behavior that otherwise would be impossible. The fact that the interface between the user and the trampoline IR grid is wireless improved the usability of the system, as it avoids the use of fixed long cables. A big concern was put into developing an attachment method for the system that would be non-invasive for both the trampoline and the athlete and would be compatible with most Olympic format trampoline models. This was also achieved by utilizing velcro positioned in key places around the trampoline's metallic ring frame. In terms of TOF resolution and HD accuracy, the developed system met both requirements as it was able to detect TOF differences with 600 microseconds and during testing, it correctly detected the HD area of each jump 100% of the time, within 10 centimeters of each area line. Finally, the cost of making the hardware was kept as low as possible, and if commercially produced would likely be sold at an affordable price for the clubs.

On the down side, the users felt that setting up the system for the first time was a bit confusing and time consuming, as all the velcro strips needed to be attached to the trampoline frame by hand. Despite this, after some explanation and trials the users were able to mount or dismount the system in less than 10 minutes. Another difficulty felt during this project was trying to test the system in a controlled environment. Given the resources available and the nature of the sport, it was very difficult to replicate experiments several times while controlling all the involved variables.

As a Master's thesis, this project proved to be very multidisciplinary, as it involved hardware and software knowledge in many areas, such as: wireless and wired communications, embedded systems, sensor technologies, Web Application (WebApp) development and 3D printing. By developing a complete system from start to end, there were also many time and project management skills learned that can only be gained by actually doing a project like this. The development of this new system led to the submission of a national patent, which was also a valuable and interesting new learning experience.

6.1 Future Work

In order to improve the system's capabilities, more work needs to be done in terms of testing in controlled environments. For example, the trampoline bed reacts differently when the athlete is closer to the edges where the springs are. This work assumed that the bed behaved the same everywhere which is a good approximation but is not exactly true. Also, these experiments should be performed in conjunction with the Fédération Internationale de Gymnastique (FIG), in order to determine which are the exact TOF and HD requirements that these types of systems need to meet to be used in a competition scenario.

An example of a future improvement to the system is a different software implementation that could help improve the HD resolution dramatically. In this scenario the individual Infrared Beam Module Receiver (IRM-Rx) would locally measure the Time on Bed (TOB) during each jump and after, send those values to the Grid Controller (GC) to be analyzed and determine the correct HD and TOF values. Here the resolution of the system is only limited by the timer's capabilities in each IRM-Rx and not by the speed at which the GC can communicate with them. Instead of hundreds of microseconds it would be dozens of microseconds, i.e. one order of magnitude lower. This algorithm was actually implemented and some tests were performed. The problem was that due to the small differences between clock sources in the MSP430 microcontrollers, the results weren't valid. In order to solve this issue a new prototype needed to be developed, where each IRM-Rx has an external more reliable oscillator. Given the limitations in terms of time and resources this new version wasn't implemented.

Another interesting parameter that can be measured by this system, with little modifications, is the Synchro Score (SYN). In this case, two Infrared Beam Grid (IR-GRID) would be mounted on two trampolines, and connected to the same Access Point Server (APS). Both systems would then need to synchronize their clocks in order to measure the difference in time between the moments when each athlete lands on their corresponding trampoline bed.

In time and with more clubs using this system, more optimizations, in hardware and software, will surely be made to improve the system's performance and help more athletes and coaches achieve higher goals.

Bibliography

[1] ACRONAME (2017) SHARP INFRARED RANGER COMPARISON. https://acroname.com/articles/sharp-infrared-ranger-comparison

[2] Acrosport (russian) (2017) Innovative technologies / systems testing. http://www.acrosport.ru/catalogue/innovatsionnye_tekhnologii/sistemy_testirovaniya/sinkhronnyy_apparat_s_3_mya_datchikami_/

[3] Adafruit (2017) How PIRs Work. https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

[4] Analog Devices (2009) Decoupling Techniques. http://www.analog.com/media/en/training-seminars/tutorials/MT-101.pdf

[5] Armin Ronacher (2017) Flask-SQLAlchemy. http://flask-sqlalchemy.pocoo.org/2.3/

[6] Armin Ronacher (2017) Welcome to Jinja2. http://jinja.pocoo.org/docs/2.10/

[7] Atmel (2013) Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash

[8] Brock H (2010) Department of Computer Science Automated Classification of Trampoline Motions Based on Inertial Sensor Input. Master's thesis, Faculty of Natural Sciences and Technology I, Department of Computer Science, Saarland University

[9] Chi EH, Borriello G, Hunt G, Davies N (2005) Guest Editors' Introduction: Pervasive Computing in Sports Technologies. Pervasive Computing, IEEE (Volume:4 , Issue:3) pp 22–25

[10] Costa A, Batalha M, Umbelino V, Amaro J (2015) Sports System Monitoring Intensity of Trampoline Jump, Conference Paper at 2015 Conference on Engineering , At Covilhã - Portugal.

[11] David Hall (2010) United states patent, sensor, control and virtual reality system for a trampoline, patent no.: Us 8,206,266 b2

[12] Embedded (2002) Serial Protocols Compared. http://www.embedded.com/design/connectivity/4023975/Serial-Protocols-Compared

[13] Espressif (2018) ESP8266EX Datasheet. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[14] Eurotramp (2017) Eurotramp Trampoline bed. https://www.eurotramp.com/us-en/products/large-trampolines/premium/

[15] Eurotramp (2017) Eurotramp Trampoline frame. https://www.eurotramp.com/us-en/products/large-trampolines/ultimate/

[16] Eurotramp Trampoline - Kurt Hack (2017) United states patent publication trampoline no.:us2017/0157444a1

[17] Fédération Internationale De Gymnastique (2017) 2017 – 2020 code of points, trampoline gymnastics. http://www.fig-gymnastics.com/publicdir/rules/files/tra/TRA-CoP_2017-2020-e.pdf

[18] Fédération Internationale de Gymnastique (FIG) (2017) Competition Description. http://www.fig-gymnastics.com/site/page/view?id=435

[19] Google (2017) Google Cardboard. https://vr.google.com/cardboard/

[20] Grinberg M (2014) Flask Web Development - Developing Web Applications With Python. O'Reilly Media

[21] Hawk-Eye Innovations (2017) Hawk-Eye in Tennis. http://www.hawkeyeinnovations.co.uk/sports/tennis

[22] I2C-Bus (2017) I2C What's That. https://www.i2c-bus.org

[23] IBEX Electronic Product Design Specialists (2017) CAN vs RS485. http://www.electronic-products-design.com/geek-area/communications/can-bus/can-vs-rs485

[24] James L Levine, Susan A Luerich and Duane Scott Miller (2007) United states patent, infrared touch screen gated by touch force, patent no.: Us 8,130,202 b2

[25] Leif Bennett (2016) Choosing a Lens. http://alumnus.caltech.edu/~leif/infratag/lens_choice.html

[26] Maxim Integrated (2005) Serial Digital Data Networks. https://www.maximintegrated.com/en/app-notes/index.mvp/id/3438

[27] Metropolia Confluence (2017) Infrared Sensors. https://wiki.metropolia.fi/display/sensor/Infrared+sensors

[28] Microchip (2017) Microchip Home Page. http://www.microchip.com

[29] Mozilla, MDN and other contributors (2018) JavaScript reference. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

[30] National Instruments (2017) RS-232, RS-422, RS-485 Serial Communication General Concepts. http://www.ni.com/white-paper/11390/en/

[31] NXP Semiconductors (2014) I2C-bus specification and user manual. http://www.nxp.com/docs/en/user-guide/UM10204.pdf

[32] Pallets (2010) Welcome to Flask. http://flask.pocoo.org/docs/1.0/

[33] Pepperl+Fuchs (2017) Light Grids. http://www.pepperl-fuchs.com/global/en/classid_51.htm

[34] Pepperl+Fuchs (2017) Pepperl+Fuchs Home Page. www.pepperl-fuchs.com

[35] Python Software Foundation (2010) PEP 333 – Python Web Server Gateway Interface v1.0. https://www.python.org/dev/peps/pep-0333/#preface

[36] Python Software Foundation (2018) Python 2.7.15 documentation. https://docs.python.org/2/

[37] Raspberry Pi Foundation (2018) Raspberry Pi 3 Model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[38] Raspberry Pi Foundation (2018) Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/

[39] Rockwell Automation (2017) Ultrasonic Sensing. http://www.ab.com/en/epub/catalogs/12772/6543185/12041221/12041229/print.html

[40] SensorWiki (2017) Ultrasound. http://www.sensorwiki.org/doku.php/sensors/ultrasound

[41] Sharp (2017) GP2Y0A21YK datasheet. http://www.sharpsma.com/webfm_send/1208

[42] Shenzhen Anxinke Technology (2015) ESP-12E WiFi Module. http://www.kloppenborg.net/images/blog/esp8266/esp8266-esp12e-specs.pdf

[43] SparkFun (2017) IR Communication. https://learn.sparkfun.com/tutorials/ir-communication

[44] Sparkfun (2017) Sparkfun Home Page. https://www.sparkfun.com

[45] SportTechie - Sports Technology News (2015) Babolat Play Connected Tennis Racket is the Future of Tennis. http://www.sporttechie.com/2015/01/29/babolat-play-connected-tennis-racket-is-the-future-of-tennis/

[46] SQLAlchemy and other contributors (2018) SQLAlchemy 1.2 Documentation. http://docs.sqlalchemy.org/en/latest/

[47] Stack Exchange Inc (2017) USART, UART, RS232, USB, SPI, I2C, TTL, etc. what are all of these and how do they relate to each other? http://electronics.stackexchange.com/questions/37814/usart-uart-rs232-usb-spi-i2c-ttl-etc-what-are-all-of-these-and-how-do-th

[48] Texas Instruments (2013) MIXED SIGNAL MICROCONTROLLER. http://www.ti.com/lit/ds/symlink/msp430g2353.pdf

[49] Texas Instruments (2014) xx555 Precision Timers Datasheet. http://www.ti.com/lit/ds/symlink/ne555.pdf

[50] Texas Instruments (2016) MSP-EXP430G2 LaunchPad Development Kit User's Guide. http://www.ti.com/lit/ug/slau318g/slau318g.pdf

[51] The Werkzeug Team (2017) Documentation Overview. http://werkzeug.pocoo.org/docs/0.14/

[52] Trampoline Timing Systems (2017) AirTime Trampoline System. http://www.trampolinetimingsystems.com/products.htm?submenuheader=0

[53] Tseng Hsiang Lin (2012) United states patent publication trampoline with feedback system, pub. no.:us2012/0295763a1

[54] University of Ulster (2015) The Role of Technology in Sport. http://www.ulster.ac.uk/scienceinsociety/technologyinsport.html

[55] Vishay (2014) High Power Infrared Emitting Diode, 940 nm, GaAlAs, MQW

[56] Vishay (2016) IR Receiver Modules for Remote Control Systems. https://www.vishay.com/docs/82459/tsop48.pdf

[57] World Wide Web Consortium (W3C) and other contributors (2017) CSS Snapshot 2017. https://www.w3.org/TR/2017/NOTE-css-2017-20170131/

[58] World Wide Web Consortium (W3C) and other contributors (2017) HTML 5.2. https://www.w3.org/TR/html5/

[59] zeroday GitHub repository (2015) nodemcu-devkit-v1.0. https://github.com/nodemcu/nodemcu-devkit-v1.0

A

Hardware Bill of Materials (BOM)

Table A.1: IRM-Tx Hardware BOM

Component	Reference	Main Characteristics	Supplier	Quantity	Unit Price
IR LED	TSAL6100	• 10º half intensity • 100mA max DC current	TME	1	€0,132
Microcontroller	ATtiny85	• 8-bit timer/counter • 0-10Mhz CPU	Arrow	1	€1,010
Lens	Google Cardboard VR	• Asymmetric biconvex	Ebay	1	€0,500
Connector	RJ11 female	• 6P4C	Amazon	2	€0,068
Voltage Regulator	LD1117S50CTR	• 1V dropout voltage • 5V linear regulator	Farnell	1	€0,303
Transistor	FDN359AN	• $Vgs_{th} = 1.6V$	Farnell	1	€0,480
Switch	OS102011MA1QN	• ON-ON switch	Farnell	1	€0,247
IC Socket	SPC15494	• 8 contacts	Farnell	1	€0,217
Capacitor C1	MC0603F104Z2	• Ceramic • 0.1uF	Farnell	1	€0,014
Capacitor C2	MCGPR16V106M	• Elecrolitic • 10uF	Farnell	1	€0,045
Resistor R1	MC0063W06031100R	• 100Ω	Farnell	1	€0,001
Resistor R2	MC0063W060311M	• 1MΩ	Farnell	1	€0,001
Resistor R3	352022RJT	• 22Ω • 1Watt	Farnell	1	€0,125
PCB	-	• Double layer	PCB Way	1	€1,000
Box	-	• PLA • 3D printed	-	1	€0,990
				TOTAL	**€5,200**

Table A.2: IRM-Rx Hardware BOM

Component	Reference	Main Characteristics	Supplier	Quantity	Unit Price
IR Sensor	TSOP4838	• 38kHz carrier frequency	Allied Electronics	1	€0,309
Microcontroller	MSP430G2553	• USCI with full I2C • 0-16Mhz CPU	Farnell	1	€2,200
Connector	RJ11 female	• 6P4C	Amazon	2	€0,068
Voltage Regulator	LD1117S33TR	• 1V dropout voltage • 3.3V linear regulator	Farnell	1	€0,263
LED	MCL053GT	• Green	Farnell	1	€0,126
Switch	OS102011MA1QN	• ON-ON switch	Farnell	1	€0,247
IC Socket	2227-20-03-07	• 20 contacts	Farnell	1	€0,142
Capacitor C1	MC0603F104Z2	• Ceramic • 0.1uF	Farnell	1	€0,014
Capacitor C2	MCGPR16V106M	• Elecrolitic • 10uF	Farnell	1	€0,045
Resistor R1	ERJ3EKF60R4	• 60.4Ω	Farnell	1	€0,035
Resistor R2	MCWR06X102J	• 10kΩ	Farnell	1	€0,003
PCB	-	• Double layer	PCB Way	1	€1,000
Box	-	• PLA • 3D printed	-	1	€0,600
				TOTAL	**€5,119**

Table A.3: GC Hardware BOM

Component	Reference	Main Characteristics	Supplier	Quantity	Unit Price
Wi-Fi Module	NodeMCU V1.0	• ESP8266	Ebay	1	€2,950
Microcontroller	MSP430G2553	• USCI with full I2C • 0-16Mhz CPU	Farnell	1	€2,200
Connector	RJ11 female	• 6P4C	Amazon	2	€0,068
Barrel Connector	DCJ235-05-A-K1-K	• 2,35 mm	Farnell	1	€0,755
Voltage Regulator	LD1117S33TR	• 1V dropout voltage • 3.3V linear regulator	Farnell	1	€0,263
LED	L-1503SRD	• Red	Farnell	1	€0,066
Switch	OS102011MA1QN	• ON-ON switch	Farnell	3	€0,247
IC Socket	2227-20-03-07	• 20 contacts	Farnell	1	€0,142
Capacitor C1	MC0603F104Z2	• Ceramic • 0.1uF	Farnell	1	€0,014
Capacitor C2	MCGPR16V106M	• Elecrolitic • 10uF	Farnell	1	€0,045
Resistor R1	MCRE000037	• 1kΩ	Farnell	1	€0,020
Resistor R2	MCWR06X102J	• 10kΩ	Farnell	1	€0,003
Resistor R3	MCRE000037	• 1kΩ	Farnell	1	€0,020
Resistor R4	MCRE000037	• 1kΩ	Farnell	1	€0,020
PCB	-	• Single layer	Tagus	1	€1,000
Box	-	• PLA • 3D printed	-	1	€1,350
				TOTAL	**€9,724**

Table A.4: APS Hardware BOM

Component	Reference	Main Characteristics	Supplier	Quantity	Unit Price
Raspberry Pi 3 Model B	RASPBERRYPI3-MODB-1GB	• BCM43438 WiFi on board • Quad Core 1.2GHz 64bit CPU	Farnell	1	€29,300
Micro SD card	047-1819	• 8 GB	Mauser	1	€4,44
Micro USB cable	047-1612	• Micro USB	Mauser	1	€1,53
Box	-	• PLA • 3D printed	-	1	€1,350
				TOTAL	**€36,620**

Table A.5: Complete System Hardware BOM

Component	Reference	Characteristics	Supplier	Quantity	Unit Price
IRM-Tx	-	• -	-	10	€5,200
IRM-Rx	-	• -	-	10	€5,119
GC	-	• -	-	1	€9,724
APS	-	• -	-	1	€36,620
Velcro	VELCRO Brand Heavy Duty	• 1 meter • 50cm^2 holds 7kg	Amazon	1	€4,000
Cable	015-0073	• Telefone cable • 15 meters	Mauser	1	€5,500
Connectors	015-0021	• RJ11 male • 6P4C	Mauser	40	€0,120
Power Supply	035-2830	• 12 Volt AC-DC • 5.0 A	Mauser	1	€14,920
				TOTAL	**€178,758**

B

Code and Flowcharts

Listing B.1: Infrared Beam Module Transmitter (IRM-Tx) software code implemented in the ATtiny85 microcontroller. Developed using the Arduino IDE.

```
void setup(){
  DDRB |= (1<<PB0); //Set pin PB0 as output
  TCNT0 = 0; //Timer0 Counter
  TCCR0A=0;  //Timer0 Control Register A
  TCCR0B=0;  //Timer0 Control Register B

  TCCR0A |=(1<<COM0A0); //Timer0 in toggle mode (associated with pin PB0)
  TCCR0A |=(1<<WGM01); //Start Timer0 in CTC mode
  TCCR0B |= (1 << CS00); //Use CPU clock with no prescalar
  OCR0A=105; //CTC Compare value
}
void loop(){}
```

Listing B.2: Infrared Beam Module Receiver (IRM-Rx) software code implemented in the MSP430G2553 microcontroller. Developed using the Code Composer Studio.

```
#include <msp430.h>

int main(void){

  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  //Calibrate CPU clock to 16MHz
  if (CALBC1_16MHZ==0xFF){  //Verifiy if constant is well defined
    while(1);
  }
  DCOCTL = 0;              // Select lowest DCOx and MODx settings
  BCSCTL1 = CALBC1_16MHZ;   // Set range
  DCOCTL = CALDCO_16MHZ;    // Set DCO step + modulation

  P2SEL &= (~BIT3);        // Set P2.3 SEL for GPIO
  P2DIR &= (~BIT3);        // Set P2.3 as input

  P1SEL |= BIT6 + BIT7;        // Assign I2C pins to USCI_B0
  P1SEL2|= BIT6 + BIT7;        // Assign I2C pins to USCI_B0

  UCB0CTL1 |= UCSWRST;         // Enable SW reset
  UCB0CTL0 = UCMODE_3 + UCSYNC;  // I2C Slave, synchronous mode
  UCB0I2COA = 0x40;            // Own Address is 040h. Each IRM-Rx will have a
      different address
  UCB0CTL1 &= ~UCSWRST;        // Clear SW reset, resume operation
  UCB0I2CIE |= UCSTTIE;        // Enable STT interrupt
  IE2 |= UCB0TXIE;             // Enable TX interrupt

  while(1){
    __bis_SR_register(CPUOFF + GIE);   // Enter LPM0 w/ interrupts
  }
}

// USCI_B0 Data ISR
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = USCIAB0TX_VECTOR
```

```
__interrupt void USCIAB0TX_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCIAB0TX_VECTOR))) USCIAB0TX_ISR (void)
#else
#error Compiler not supported!
#endif
{
  UCB0TXBUF = (P2IN & BIT3);  // TX data
  __bic_SR_register_on_exit(CPUOFF);  // Exit LPM0

}

// USCI_B0 State ISR
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = USCIAB0RX_VECTOR
__interrupt void USCIAB0RX_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCIAB0RX_VECTOR))) USCIAB0RX_ISR (void)
#else
#error Compiler not supported!
#endif
{
  UCB0STAT &= ~UCSTTIFG;  // Clear start condition int flag
}
```

Listing B.3: Grid Controller (GC) software code implemented in the MSP430G2553 microcontroller. Developed using the Code Composer Studio.

```c
#include "msp430g2553.h"

unsigned char RXData;
unsigned char takeOffFlag=0, x=0, y=0, numXSens=6, numYSens=4;
//These addresses correspond to the IRM-Rx shown in Figure D.1
//0x40-X1/0x41-X2/0x42-X3/0x43-X4/0x44-X5/0x45-X6
//0x46-Y1/0x47-Y2/0x48-Y3/0x49-Y4
unsigned char xAddrList[6] = {0x40, 0x41, 0x42, 0x43, 0x44, 0x45};
unsigned char yAddrList[4] = {0x46, 0x47, 0x48, 0x49};
//These area code values correspond to the areas shown in Figure D.2:
//0-A0/1-A11/2-A12/3-A13/4-A14/5-A2/6-A31/7-A41/8-A51/9-A52/10-A61/11-A71/12-A32/13-
    A42/14-A53/15-A54/16-A62/17-A72/18-A8/19-A91/20-A92/21-A93/22-A94/23-A10
unsigned char areaCodeList[6][4] =
    {{18,12,6,0},{19,13,7,1},{20,14,8,2},{21,15,8,3},{22,16,9,4},{23,17,9,5}};
unsigned int tofInClockCycles = 0;
int foundXFlag=0, foundYFlag=0, i=0;
unsigned char currentAddr = 0;

void pollForTakeOff();

int main(void){
  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  //Calibrate CPU clock to 16MHz
  if (CALBC1_16MHZ==0xFF){  //Verifiy if constant is well defined
    while(1);
  }
  DCOCTL = 0;          // Select lowest DCOx and MODx settings
  BCSCTL1 = CALBC1_16MHZ;   // Set range
  DCOCTL = CALDCO_16MHZ;    // Set DCO step + modulation
  //Source Timer clock from ACLK with VLO = 12khz
  BCSCTL3 |= LFXT1S_2;    // Source ACLK from VLO
  TA0CTL |= TASSEL_1 + MC_0;  //Use ACLK as source for timer0
  TA1CTL |= TASSEL_1 + MC_0;  //Use ACLK as source for timer1
  //Setup GPIO for LED
  P1DIR |= BIT0;        // P1.0 output
  //Setup I2C bus
  P1SEL |= BIT6 + BIT7;   // Assign I2C pins to USCI_B0
  P1SEL2|= BIT6 + BIT7;   // Assign I2C pins to USCI_B0
  UCB0CTL1 |= UCSWRST;        // Enable SW reset
  UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC; // I2C Master, synchronous mode
  UCB0CTL1 = UCSSEL_2 + UCSWRST;      // Use SMCLK, keep SW reset
  UCB0BR0 = 40;               // fSCL = SMCLK/40 = ~400kHz
  UCB0BR1 = 0;
  UCB0CTL1 &= ~UCSWRST;       // Clear SW reset, resume operation
  IE2 |= UCB0RXIE;            // Enable RX interrupt
  //Setup serial UART transmission to ESP-12E
  P2DIR |= 0xFF;             // All P2.x outputs
  P2OUT &= 0x00;             // All P2.x reset
  P1SEL |= BIT1 + BIT2 ;         // P1.1 = RXD, P1.2=TXD
  P1SEL2 |= BIT1 + BIT2 ;        // P1.1 = RXD, P1.2=TXD
```

```
P1OUT &= 0x00;
UCA0CTL1 |= UCSSEL_2;          // SMCLK
UCA0BR0 = 0x8A;            // 16MHz 115200
UCA0BR1 = 0x00;            // 16MHz 115200
UCA0MCTL = UCBRS2 + UCBRS0;      // Modulation UCBRSx = 5
UCA0CTL1 &= ~UCSWRST;          // **Initialize USCI state machine**

while (1){
//In the worst case, each loop is ~600us (this is the resolution)
takeOffFlag=0;
  if(!foundXFlag){
    for(x=0;x<numXSens;x++){
      UCB0I2CSA = xAddrList[x];     // define slave address
      WDTCTL = WDTPW + BIT3 + BIT2 + BIT1;//WDT of 43ms
      while (UCB0CTL1 & UCTXSTP);          // Ensure stop condition got sent
      UCB0CTL1 |= UCTXSTT;                 // I2C start condition
      while (UCB0CTL1 & UCTXSTT);          // Start condition sent?
      UCB0CTL1 |= UCTXSTP;                 // I2C stop condition
      __bis_SR_register(CPUOFF + GIE);     // Enter LPM0 w/ interrupts
      WDTCTL = WDTPW + BIT7 + BIT3 + BIT2 + BIT1; //WDT of 43ms
      if(RXData == 8){
        P1OUT |= BIT0;           // ON LED1.0
        foundXFlag=1;          //X-axis beam interrupted
        break;
      }
    }
  }
  if(!foundYFlag){
    for(y=0;y<numYSens;y++){
      UCB0I2CSA = yAddrList[y];     // define slave address
      WDTCTL = WDTPW + BIT3 + BIT2 + BIT1;//WDT of 43ms
      while (UCB0CTL1 & UCTXSTP);          // Ensure stop condition got sent
      UCB0CTL1 |= UCTXSTT;                 // I2C start condition
      while (UCB0CTL1 & UCTXSTT);          // Start condition sent?
      UCB0CTL1 |= UCTXSTP;                 // I2C stop condition
      __bis_SR_register(CPUOFF + GIE);     // Enter LPM0 w/ interrupts
      WDTCTL = WDTPW + BIT7 + BIT3 + BIT2 + BIT1; //WDT of 43ms
      if(RXData == 8){
        tofInClockCycles = TA0R;     //This will be the TOF. Timer0 was initialized
             in pollForTakeOff() function
        currentAddr = UCB0I2CSA;     //IRM-Rx whose IR beam was interrupted first
        P1OUT |= BIT0;           // ON LED1.0
        foundYFlag=1;          //Y-axis beam interrupted
        break;
      }
    }
  }
  if(foundXFlag && foundYFlag){
    while (!(IFG2&UCA0TXIFG));
    UCA0TXBUF = areaCodeList[x][y];     // TX area
    while (!(IFG2&UCA0TXIFG));
    UCA0TXBUF = (tofInClockCycles & 0xFF);  // TX lower
```

```
      while (!(IFG2&UCA0TXIFG));
      UCA0TXBUF = ((tofInClockCycles>>8) & 0xFF); // TX upper
      pollForTakeOff();           //Check when the atlete leaves the bed
      foundXFlag=0;
      foundYFlag=0;
    }
  }
}

void pollForTakeOff(){
  UCB0I2CSA = currentAddr;
  while(takeOffFlag == 0){
    TA0CTL |= TASSEL_1 + MC_0 + TACLR;    //Use ACLK as source for timer0. Timer0
        stopped.
    WDTCTL = WDTPW + BIT3 + BIT2 + BIT1;    //WDT of 43ms
    while (UCB0CTL1 & UCTXSTP);             // Ensure stop condition got sent
    UCB0CTL1 |= UCTXSTT;                    // I2C start condition
    while (UCB0CTL1 & UCTXSTT);             // Start condition sent?
    UCB0CTL1 |= UCTXSTP;                    // I2C stop condition
    __bis_SR_register(CPUOFF + GIE);        // Enter LPM0 w/ interrupts
    WDTCTL = WDTPW + BIT7 + BIT3 + BIT2 + BIT1; //WDT of 43ms
    if(RXData == 0){              //IR beam uninterrupted. Possibly takeoff?
      TA0CTL |= TASSEL_1 + MC_2;      //Start timer0 in continuous mode
      while(1){
        WDTCTL = WDTPW + BIT3 + BIT2 + BIT1;//WDT of 43ms
        while (UCB0CTL1 & UCTXSTP);         // Ensure stop condition got sent
        UCB0CTL1 |= UCTXSTT;                // I2C start condition
        while (UCB0CTL1 & UCTXSTT);         // Start condition sent?
        UCB0CTL1 |= UCTXSTP;                // I2C stop condition
        __bis_SR_register(CPUOFF + GIE);    // Enter LPM0 w/ interrupts
        WDTCTL = WDTPW + BIT7 + BIT3 + BIT2 + BIT1; //WDT of 43ms
        if(RXData==0 && TA0R>=4800){      //IR beam uninterrupted for 400ms?
          P1OUT &= ~BIT0;         // OFF LED1.0
          takeOffFlag=1;
          break;
        }else if(RXData==8){        //still on trampoline bed?
          takeOffFlag=0;
          break;
        }
      }
    }
  }
  return;
}

// USCI_B0 Data ISR
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = USCIAB0TX_VECTOR
__interrupt void USCIAB0TX_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCIAB0TX_VECTOR))) USCIAB0TX_ISR (void)
#else
```

```
#error Compiler not supported!
#endif
{
  RXData = UCB0RXBUF;                 // Get RX data
  __bic_SR_register_on_exit(CPUOFF);  // Exit LPM0
}
```

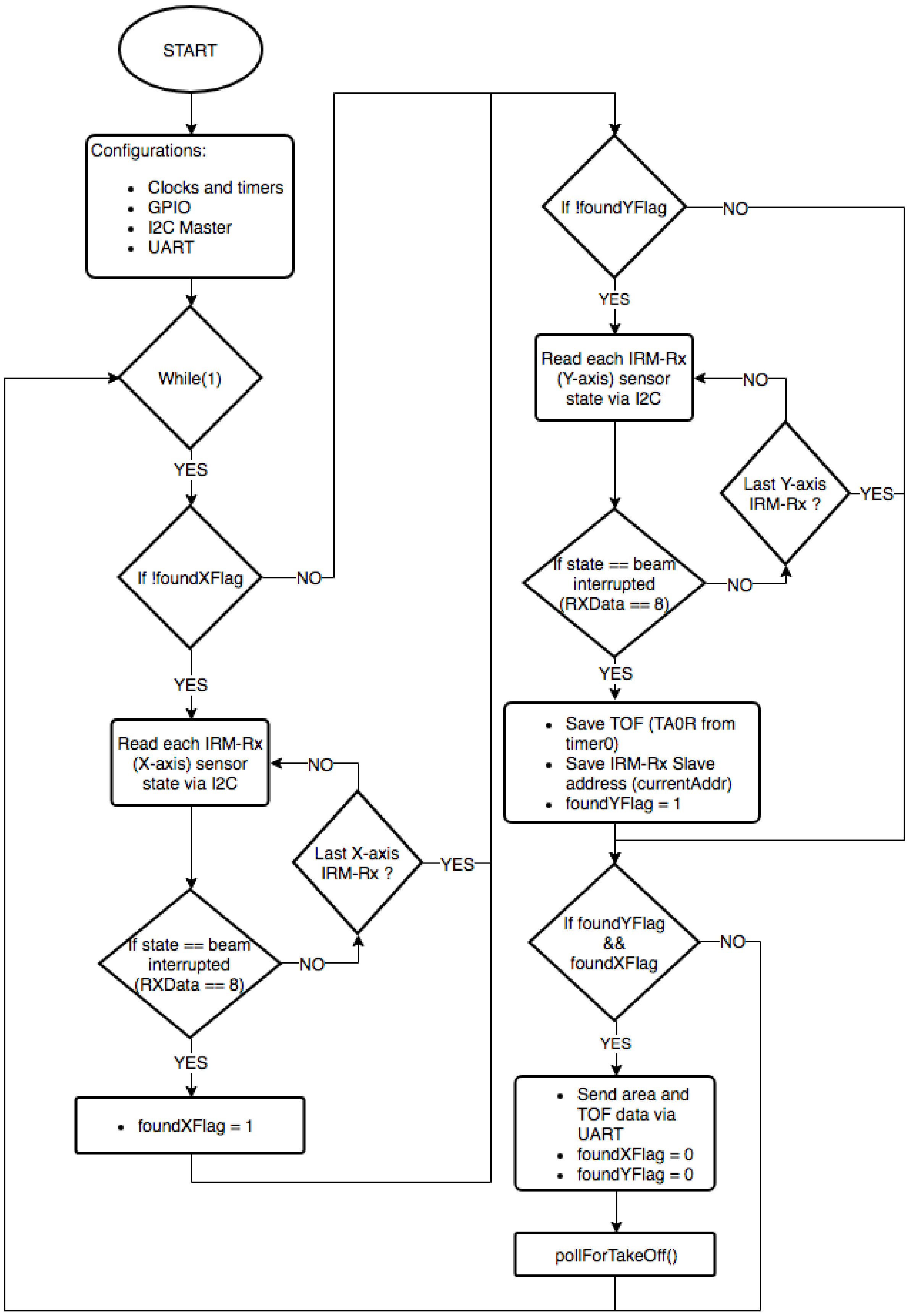
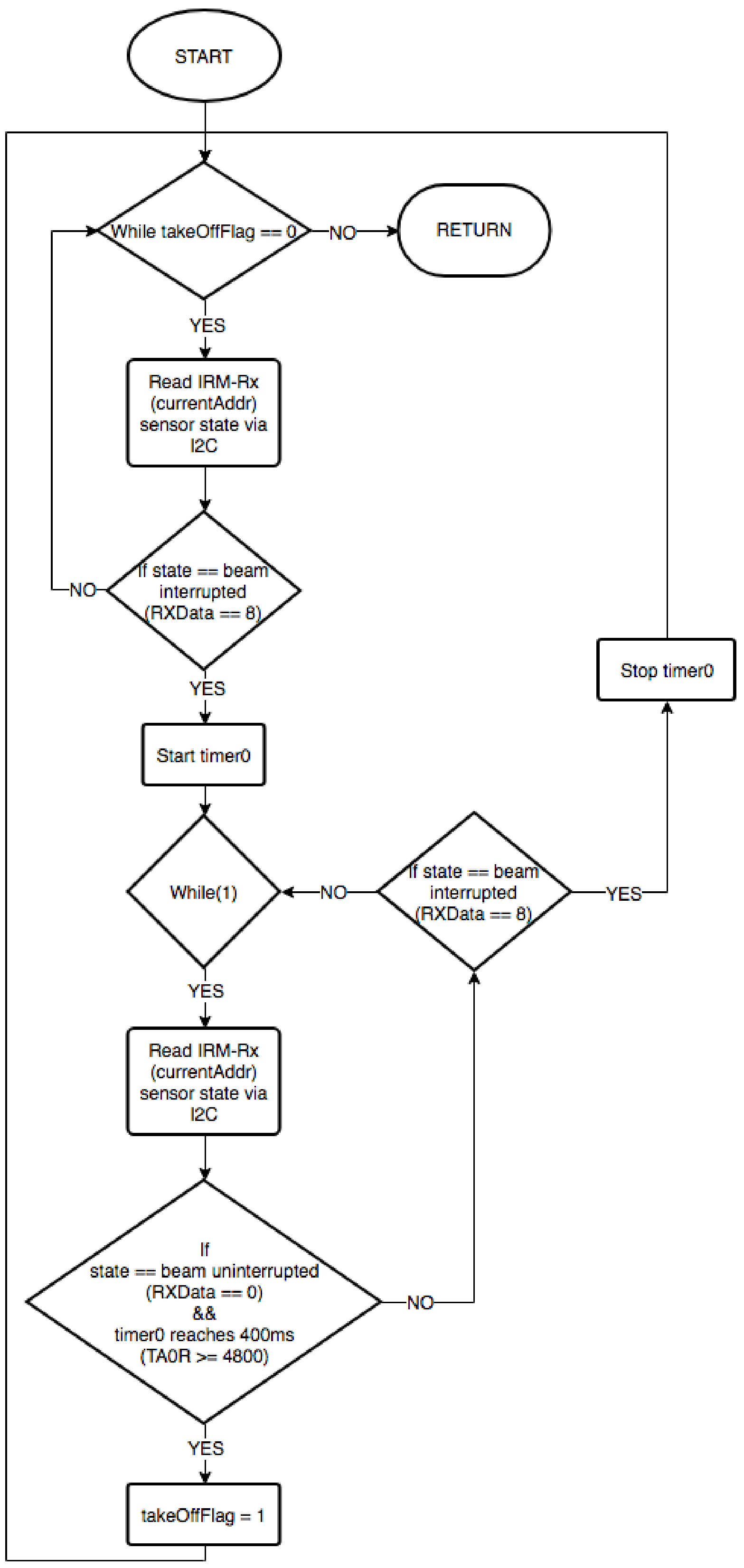

START
Configurations:
Clocks and timers
GPIO
I2C Master
UART
While(1)
YES
If !foundXFlag
NO
YES
Read each IRM-Rx (X-axis) sensor state via I2C
NO
Last X-axis IRM-Rx ?
YES
If state == beam interrupted (RXData == 8)
NO
YES
foundXFlag = 1
If !foundYFlag
NO
YES
Read each IRM-Rx (Y-axis) sensor state via I2C
NO
Last Y-axis IRM-Rx ?
YES
If state == beam interrupted (RXData == 8)
NO
YES
Save TOF (TA0R from timer0)
Save IRM-Rx Slave address (currentAddr)
foundYFlag = 1
If foundYFlag && foundXFlag
NO
YES
Send area and TOF data via UART
foundXFlag = 0
foundYFlag = 0
pollForTakeOff()

Figure B.1: IRM-Rx flowchart of the `main()` function in the software implemented in the MSP430G2553 microcontroller.



START
While takeOffFlag == 0
NO
RETURN
YES
Read IRM-Rx (currentAddr) sensor state via I2C
If state == beam interrupted (RXData == 8)
NO
YES
Start timer0
While(1)
NO
If state == beam interrupted (RXData == 8)
YES
Stop timer0
YES
Read IRM-Rx (currentAddr) sensor state via I2C
If state == beam uninterrupted (RXData == 0) && timer0 reaches 400ms (TA0R >= 4800)
NO
YES
takeOffFlag = 1

Figure B.2: IRM-Rx flowchart of the `pollForTakeOff()` function in the software implemented in the MSP430G2553 microcontroller.

Listing B.4: GC software code implemented in the ESP-12E microcontroller. Developed using the Arduino IDE.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "Pi_AP";                //APS Wi-Fi network name ID
const char* password = "betterjumppass";  //APS Wi-Fi network password
const char* host = "192.168.42.1";        //APS IP
const int trampID = 1;                     //GC ID (one GC per trampoline)
String message="";
unsigned int tofCycles=0, upper, lower, areaCode, jumpTmpID = 0;
float realTOF;

WiFiClient client;

void setup()
{
  Serial.begin(115200);       //open serial connection to receive data from GC

  WiFi.begin(ssid, password); //connect to APS Wi-Fi network
  while (WiFi.status() != WL_CONNECTED);
}

void loop()
{
  message="";
  //Receive jump data from GC
  while(Serial.available() == 0); //wait until there are bytes to be read
  areaCode = (int)Serial.read();  //get HD area code
  lower = (int)Serial.read();     //get lower 8 bits of TOF (in timer cycles)
  upper = (int)Serial.read();     //get upper 8 bits of TOF (in timer cycles)

  tofCycles=(upper);
  tofCycles=((tofCycles<<8)|lower);
  //Convert TOF in timer cycles to seconds
  realTOF = float(tofCycles);
  realTOF = realTOF/(12000);    //Timer0 at 12kHz
  //message to send to APS has this form: "areaCode/TOF(seconds)/jumpTmpID/trampID"
  message=message+areaCode+"/"+String(realTOF, 3)+"/"+jumpTmpID+"/"+String(trampID);
  //Create HTTP Client and send message to APS
  HTTPClient http;
  http.begin(host,5000,"/post_sensor"); // http://192.168.42.1:5000/post_sensor is
      the URL to send jump data
  http.POST(message);
  http.end();
  delay(0);
  jumpTmpID++;
  //Restart jumpTmpID when they reach 100
  if(jumpTmpID==100){
    jumpTmpID=0;
  }
}
```

Listing B.5: Summary of the seven most important Access Point Server (APS) Web Application (WebApp) *view functions*.

```
#Display the Login page
@auth.route('/login', methods=['GET','POST'])
def login():
  #  ...
  return render_template('auth/login.html', form=form)

#Display the main App page
@main.route('/', methods=['GET', 'POST'])
@login_required
def appPage():
  #  ...
  return render_template('main/gui.html', form=form, athleteList=athleteList)

#Called by the Browser Client to request (GET) the most recent skill
@main.route('/get_score', methods=['GET'])
@login_required
def getScore():
  #  ...
  return jsonify(routineDict)

#Called by the Grid Controller to POST data from the last skilled performed
@main.route('/post_sensor', methods=['POST'])
def postSensorJumpData():
  #  ...
  return "Received jump data\n"

#Called by Client Browser when the user presses on the SAVE button
@main.route('/save_routine', methods=['GET', 'POST'])
@login_required
def saveRoutine():
#  ...
return jsonify(result="OK")

#Display the Profile page
@main.route('/profile/<profileName>', methods=['GET'])
@login_required
def profilePage(profileName):
  #  ...
  return render_template('main/profile.html', profileName=profile.name, level=
      profile.level, profileRoutines=profileRoutines, birthDay=birthDay)

#Display the Profile Routine History page
@main.route('/profile_history/<profileName>', methods=['GET', 'POST'])
@login_required
def profileHistoryPage(profileName):
  #  ...
  return render_template('main/profileHistory.html', profileName=profileName, srList
      =l, routineArray=routineArray, num=0, selNum=0, filterOption = 0, f="
      00-00-0000", t="00-00-0000",numRoutines=numRoutines,numBadRoutines=
      numBadRoutines,numGoodRoutines=numGoodRoutines,filterLabel=filterLabel)
```

C

WebApp Graphical User Interface (GUI) Screenshots

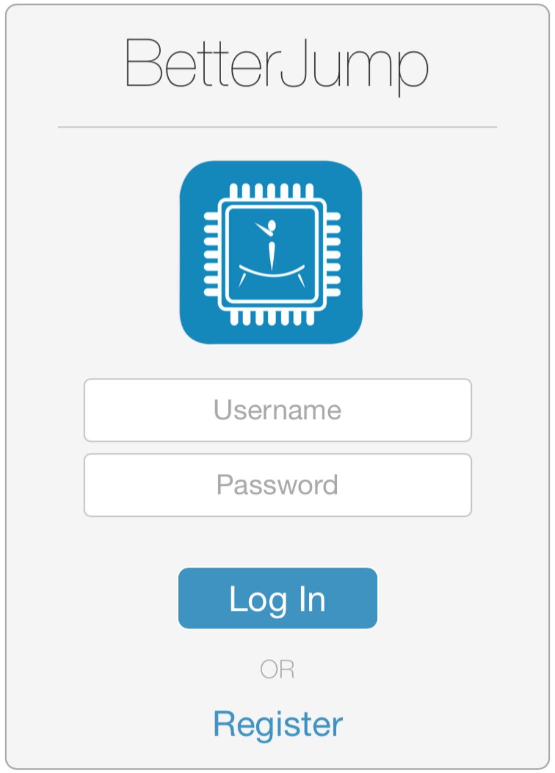


Figure C.1: Login page.

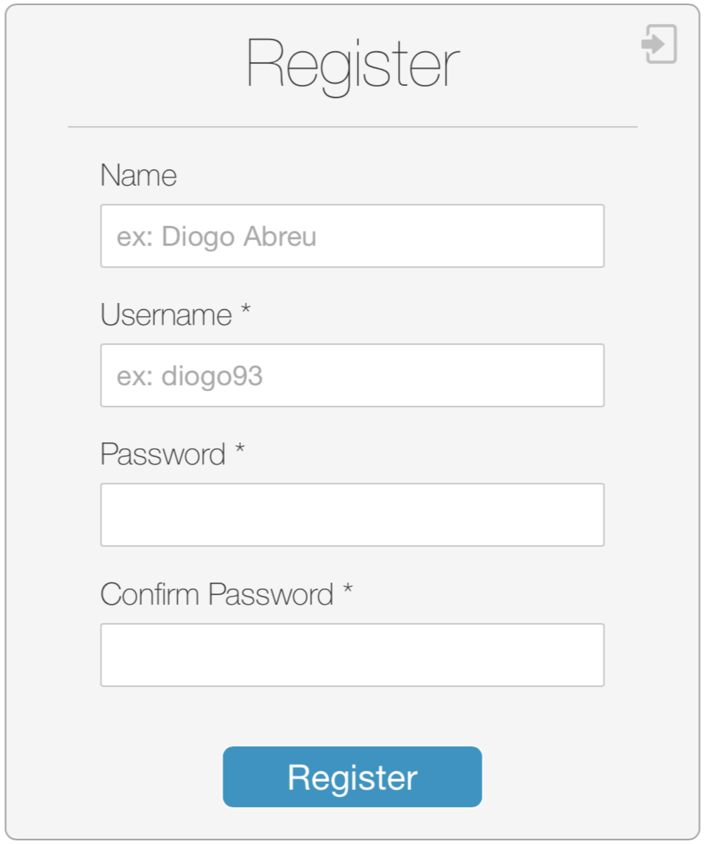
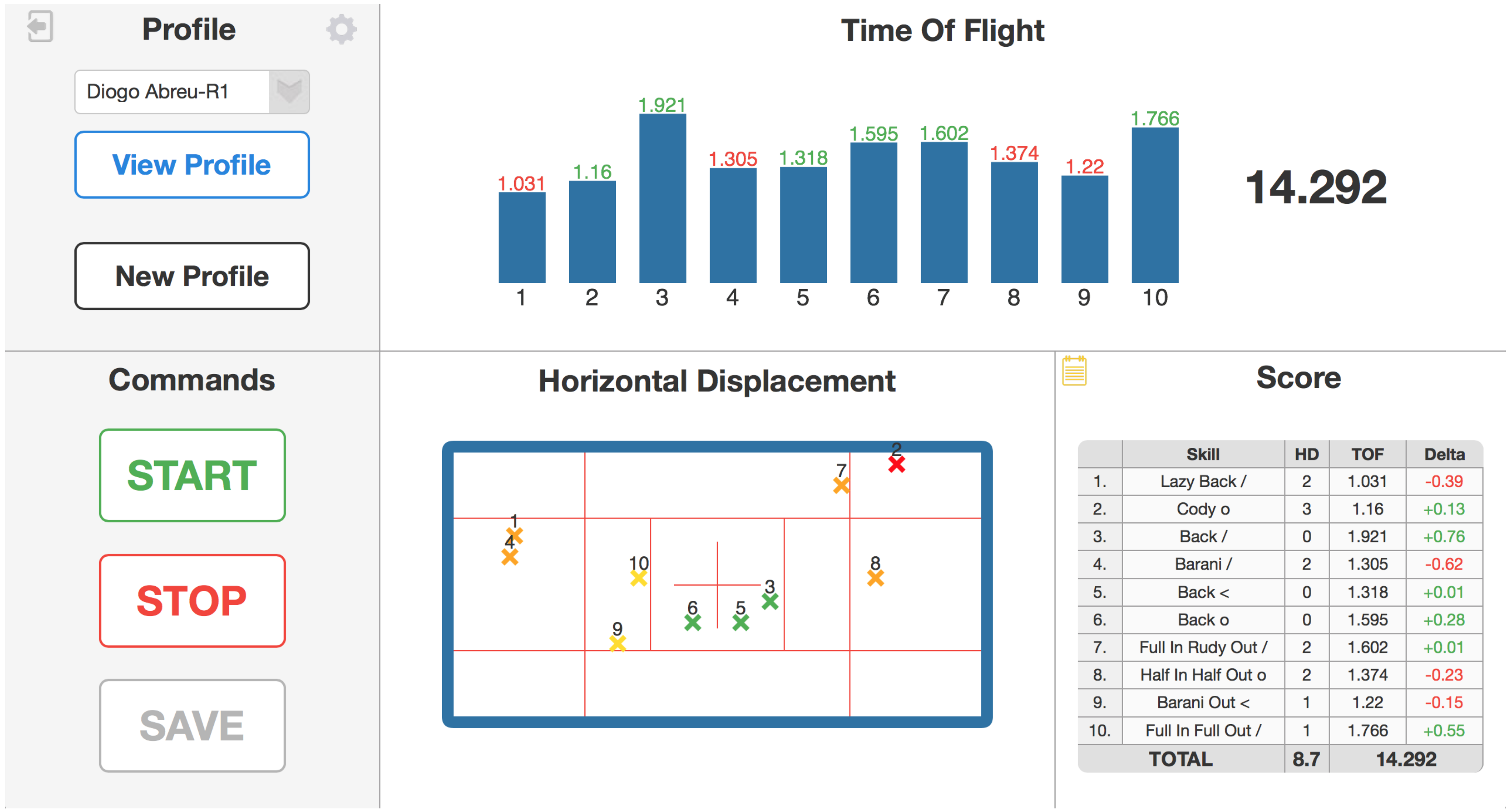
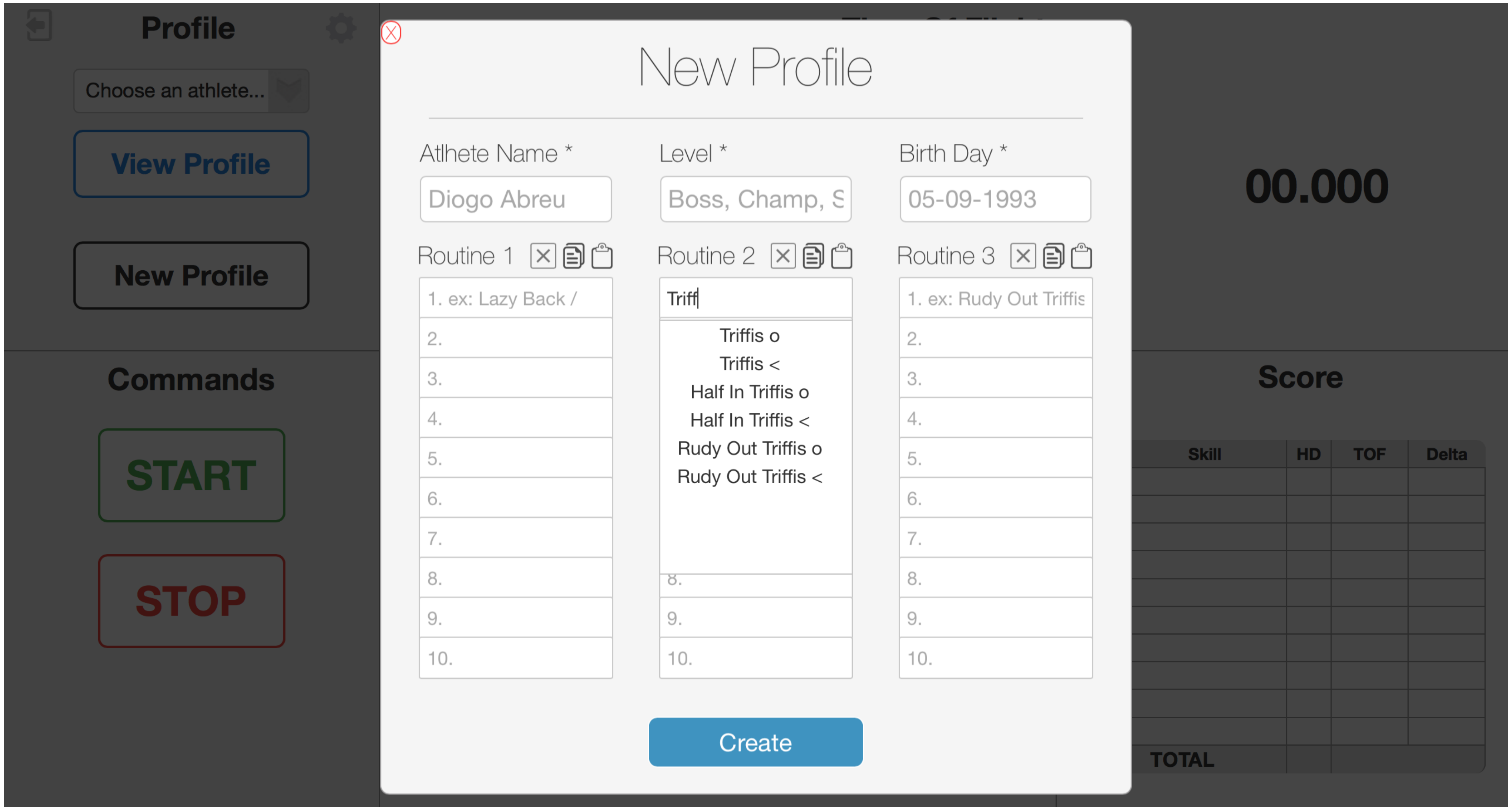


Figure C.2: Register page.



	Skill	HD	TOF	Delta
1.	Lazy Back /	2	1.031	-0.39
2.	Cody o	3	1.16	+0.13
3.	Back /	0	1.921	+0.76
4.	Barani /	2	1.305	-0.62
5.	Back <	0	1.318	+0.01
6.	Back o	0	1.595	+0.28
7.	Full In Rudy Out /	2	1.602	+0.01
8.	Half In Half Out o	2	1.374	-0.23
9.	Barani Out <	1	1.22	-0.15
10.	Full In Full Out /	1	1.766	+0.55
TOTAL		8.7	14.292	

Figure C.3: Main application page.



Profile

Choose an athlete...

View Profile

New Profile

Commands

START

STOP

New Profile

Atlhete Name *

Diogo Abreu

Level *

Boss, Champ, S

Birth Day *

05-09-1993

Routine 1

1. ex: Lazy Back /
2.
3.
4.
5.
6.
7.
8.
9.
10.

Routine 2

Triff

Triffis o
Triffis <
Half In Triffis o
Half In Triffis <
Rudy Out Triffis o
Rudy Out Triffis <

8.
9.
10.

Routine 3

1. ex: Rudy Out Triffis
2.
3.
4.
5.
6.
7.
8.
9.
10.

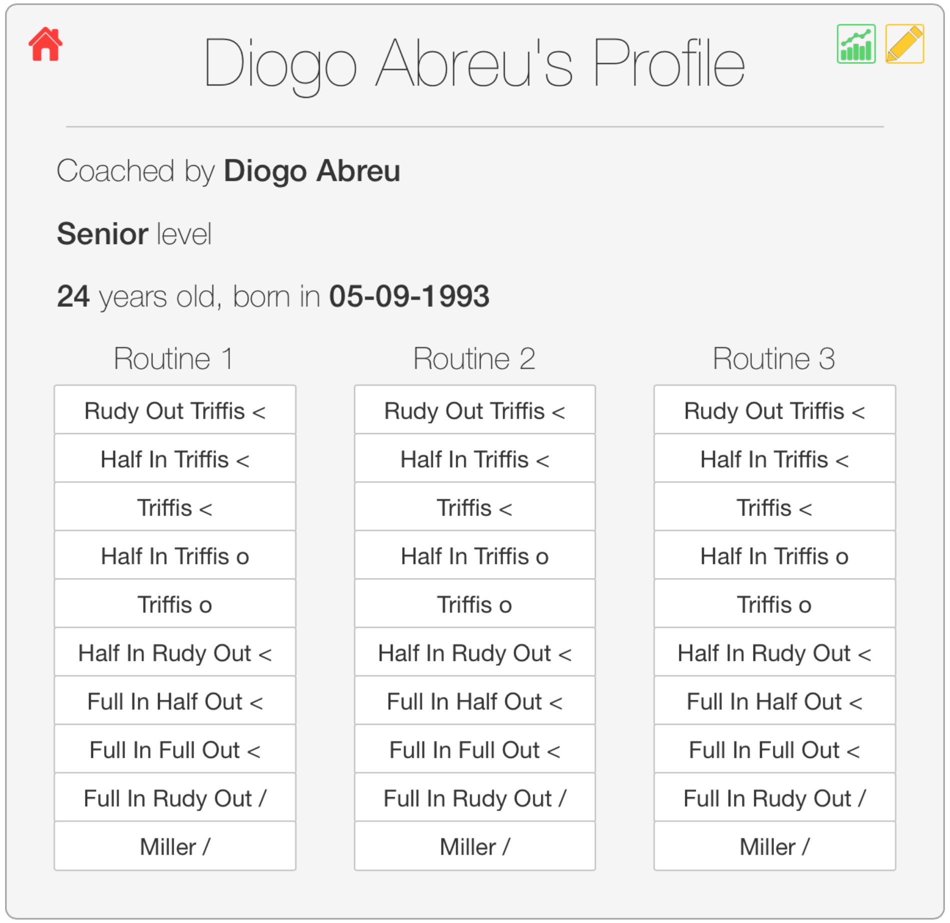
Create

00.000

Score

Skill	HD	TOF	Delta
TOTAL			

Figure C.4: Main application page with New Profile overlay.



Diogo Abreu's Profile

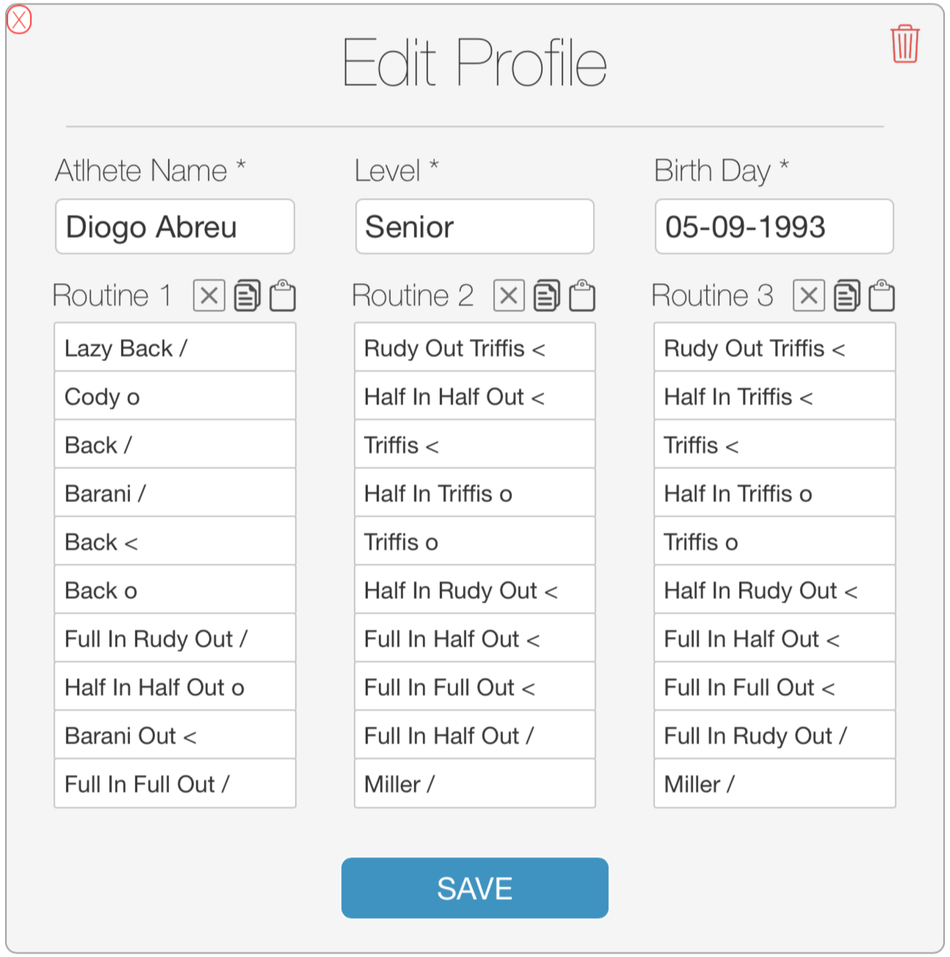
Coached by **Diogo Abreu**

Senior level

24 years old, born in **05-09-1993**

Routine 1	Routine 2	Routine 3
Rudy Out Triffis <	Rudy Out Triffis <	Rudy Out Triffis <
Half In Triffis <	Half In Triffis <	Half In Triffis <
Triffis <	Triffis <	Triffis <
Half In Triffis o	Half In Triffis o	Half In Triffis o
Triffis o	Triffis o	Triffis o
Half In Rudy Out <	Half In Rudy Out <	Half In Rudy Out <
Full In Half Out <	Full In Half Out <	Full In Half Out <
Full In Full Out <	Full In Full Out <	Full In Full Out <
Full In Rudy Out /	Full In Rudy Out /	Full In Rudy Out /
Miller /	Miller /	Miller /

Figure C.5: Profile information page.

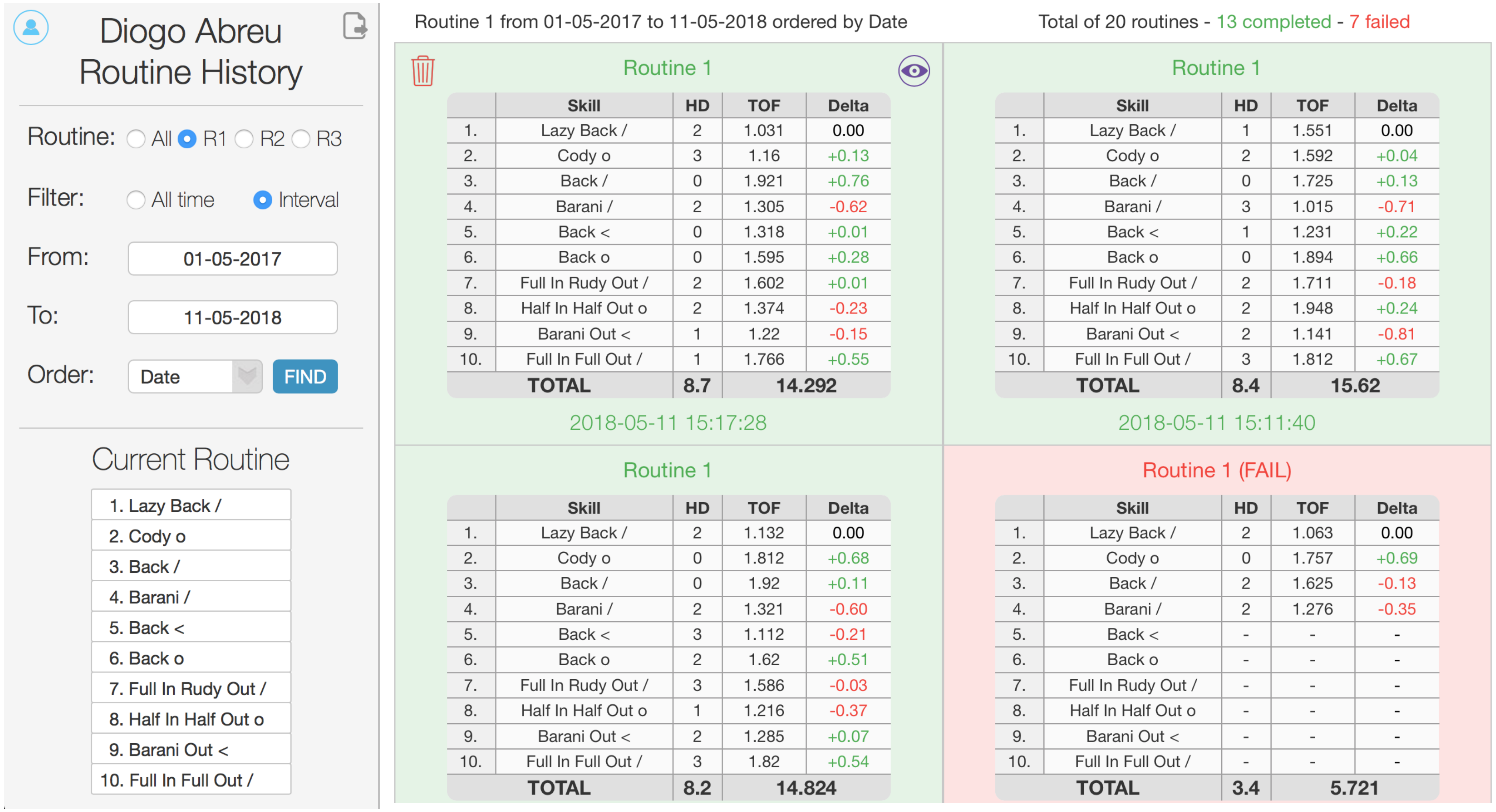


Edit Profile

Atlhete Name *	Level *	Birth Day *
Diogo Abreu	Senior	05-09-1993

Routine 1	Routine 2	Routine 3
Lazy Back /	Rudy Out Triffis <	Rudy Out Triffis <
Cody o	Half In Half Out <	Half In Triffis <
Back /	Triffis <	Triffis <
Barani /	Half In Triffis o	Half In Triffis o
Back <	Triffis o	Triffis o
Back o	Half In Rudy Out <	Half In Rudy Out <
Full In Rudy Out /	Full In Half Out <	Full In Half Out <
Half In Half Out o	Full In Full Out <	Full In Full Out <
Barani Out <	Full In Half Out /	Full In Rudy Out /
Full In Full Out /	Miller /	Miller /

Figure C.6: Edit profile information page.



Diogo Abreu
Routine History

Routine: All [✓ R1] R2 R3

Filter: All time [✓ Interval]

From: 01-05-2017

To: 11-05-2018

Order: Date

FIND

Current Routine

1. Lazy Back /
2. Cody o
3. Back /
4. Barani /
5. Back <
6. Back o
7. Full In Rudy Out /
8. Half In Half Out o
9. Barani Out <
10. Full In Full Out /

Routine 1 from 01-05-2017 to 11-05-2018 ordered by Date

Total of 20 routines - 13 completed - 7 failed

Routine 1

	Skill	HD	TOF	Delta
1.	Lazy Back /	2	1.031	0.00
2.	Cody o	3	1.16	+0.13
3.	Back /	0	1.921	+0.76
4.	Barani /	2	1.305	-0.62
5.	Back <	0	1.318	+0.01
6.	Back o	0	1.595	+0.28
7.	Full In Rudy Out /	2	1.602	+0.01
8.	Half In Half Out o	2	1.374	-0.23
9.	Barani Out <	1	1.22	-0.15
10.	Full In Full Out /	1	1.766	+0.55
	TOTAL	8.7	14.292	

2018-05-11 15:17:28

Routine 1

	Skill	HD	TOF	Delta
1.	Lazy Back /	1	1.551	0.00
2.	Cody o	2	1.592	+0.04
3.	Back /	0	1.725	+0.13
4.	Barani /	3	1.015	-0.71
5.	Back <	1	1.231	+0.22
6.	Back o	0	1.894	+0.66
7.	Full In Rudy Out /	2	1.711	-0.18
8.	Half In Half Out o	2	1.948	+0.24
9.	Barani Out <	2	1.141	-0.81
10.	Full In Full Out /	3	1.812	+0.67
	TOTAL	8.4	15.62	

2018-05-11 15:11:40

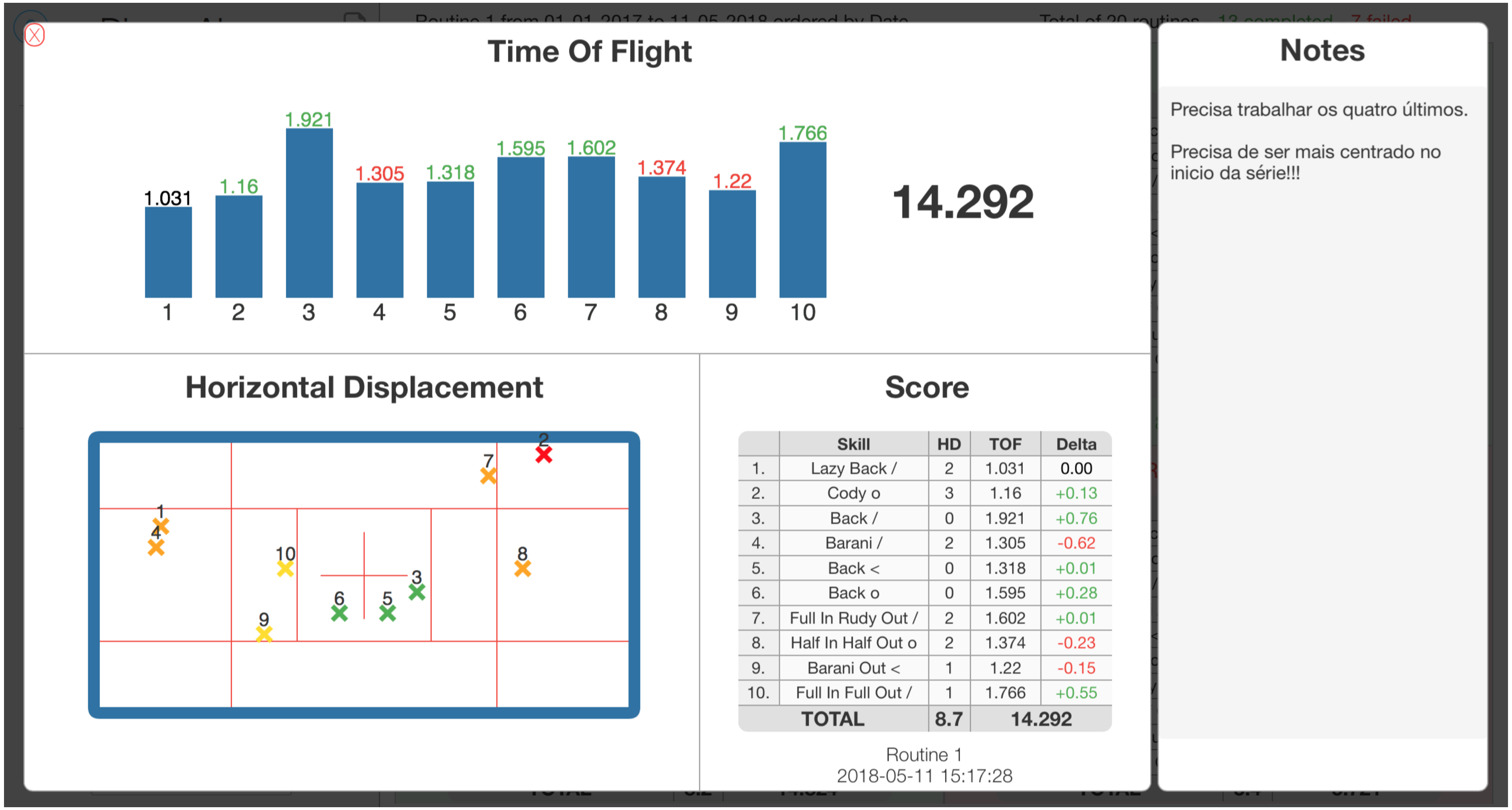
Routine 1

	Skill	HD	TOF	Delta
1.	Lazy Back /	2	1.132	0.00
2.	Cody o	0	1.812	+0.68
3.	Back /	0	1.92	+0.11
4.	Barani /	2	1.321	-0.60
5.	Back <	3	1.112	-0.21
6.	Back o	2	1.62	+0.51
7.	Full In Rudy Out /	3	1.586	-0.03
8.	Half In Half Out o	1	1.216	-0.37
9.	Barani Out <	2	1.285	+0.07
10.	Full In Full Out /	3	1.82	+0.54
	TOTAL	8.2	14.824	

Routine 1 (FAIL)

	Skill	HD	TOF	Delta
1.	Lazy Back /	2	1.063	0.00
2.	Cody o	0	1.757	+0.69
3.	Back /	2	1.625	-0.13
4.	Barani /	2	1.276	-0.35
5.	Back <	-	-	-
6.	Back o	-	-	-
7.	Full In Rudy Out /	-	-	-
8.	Half In Half Out o	-	-	-
9.	Barani Out <	-	-	-
10.	Full In Full Out /	-	-	-
	TOTAL	3.4	5.721	

Figure C.7: Profile routine history page.



Score

	Skill	HD	TOF	Delta
1.	Lazy Back /	2	1.031	0.00
2.	Cody o	3	1.16	+0.13
3.	Back /	0	1.921	+0.76
4.	Barani /	2	1.305	-0.62
5.	Back <	0	1.318	+0.01
6.	Back o	0	1.595	+0.28
7.	Full In Rudy Out /	2	1.602	+0.01
8.	Half In Half Out o	2	1.374	-0.23
9.	Barani Out <	1	1.22	-0.15
10.	Full In Full Out /	1	1.766	+0.55
TOTAL		8.7	14.292	

Routine 1
2018-05-11 15:17:28

Notes

Precisa trabalhar os quatro últimos.

Precisa de ser mais centrado no inicio da série!!!

Figure C.8: Profile routine history page with graphical representation of routine overlay.

D

Infrared Beam Grid (IR-GRID) Configuration

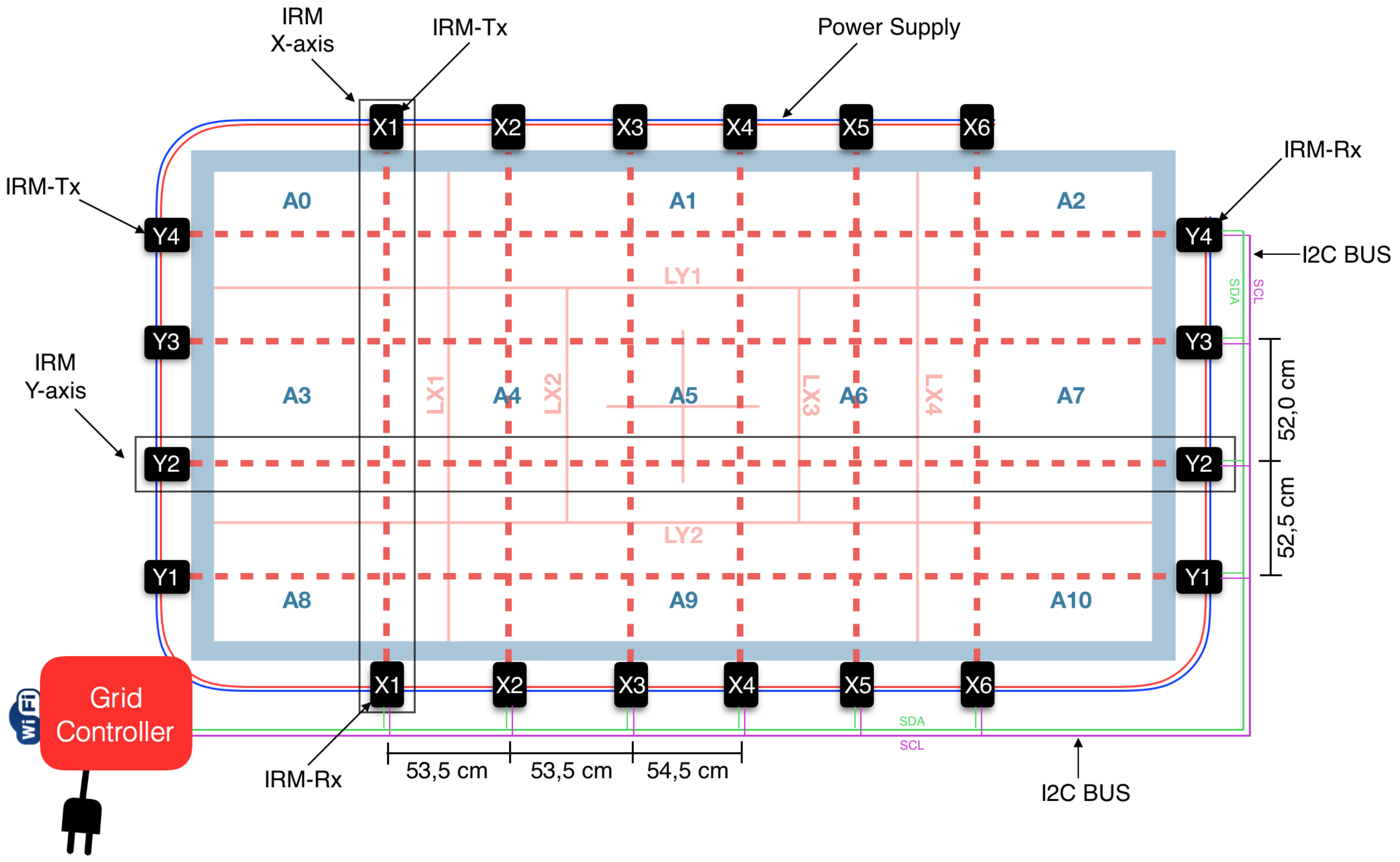


Figure D.1: Infrared Beam Grid (IR-GRID) configuration.

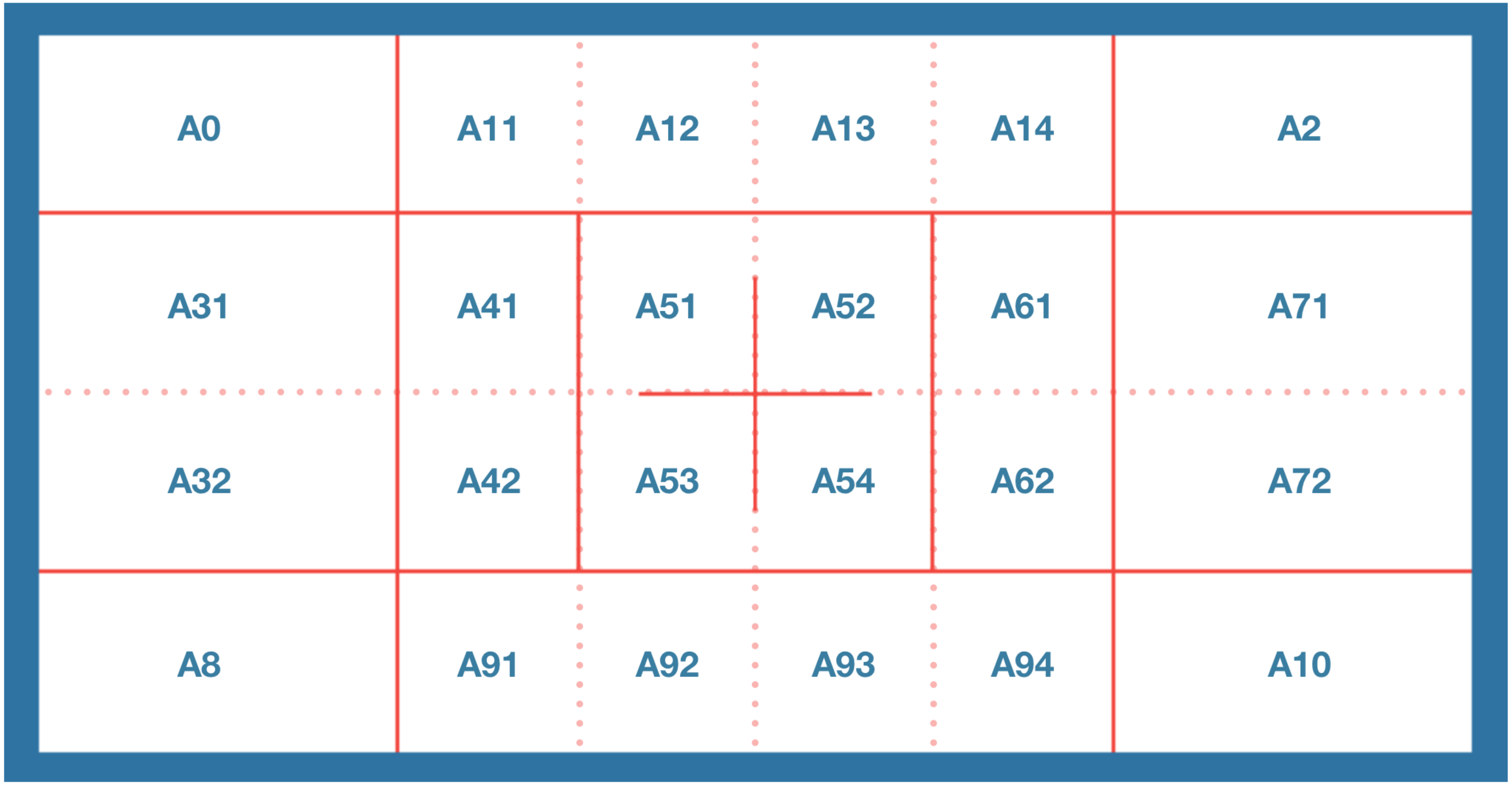


Figure D.2: All “virtual” areas that the IR-GRID configuration can differentiate. These areas are coded into integers, as shown in Listing B.3.