DEVELOPMENT OF A SYSTEM TO MONITOR SURFER'S BALANCE AND PLANTAR PRESSURE AND SURFBOARD'S MOVEMENT

Daniel Dezan de Bona

Master in Electronic and Computer Engineering
Automation and Systems
Department of Electrical Engineering
School of Engineering

2014

This report fulfills the requirements contained in form Discipline of Thesis/Dissertation, of 2º year, of Master in Engineering of Electric and Computer.

Candidate: Daniel Dezan de Bona, Nº 1110023, daniel.dezan@gmail.com
Supervisor: Maria Arcelina Marques, PhD, mmr@isep.ipp.pt
Co-Supervisor: Miguel Velhote Correia, PhD, mcorreia@fe.up.pt

Master in Electronic and Computer Engineering
Automation and Systems
Department of Electrical Engineering
School of Engineering
12 de March de 2014

This work is dedicated to my sister Fernanda Dezan de Bona *(in memoriam)* who I really miss…

Acknowledgments

I would like to thank everybody that somehow have contributed to the success of this project which, beyond of the professional development reported in this text, had impact on my personal life.

First of all, I would like to thank God to be a constant presence in my daily routine.

My parents Geraldo de Bona and Maria Albertina Dezan de Bona for being the most important professors I ever had and also my brother Eduardo de Bona to show me how important family is. Likewise, I would like to thank my wife Cynthia de Mello de Bona for the support and relief in hard times.

I would like a special thanks to Maria Arcelina Marques and Miguel Velhote Correia, my supervisors, for believing in this project and for the many hours spent in conversations about it. I also would like to thank Gustavo Ribeiro Alves, PhD, to the referrals in the beginning.

To the institutions that somehow supported me during the development: School of Engineering - Polytechnic of Porto; Biomechanics Laboratory - Porto University; Faculty of Engineering - Porto University; SRS Surfboards.

To my friends, that due to the distance, I left behind and for the friends that I have made in Portugal. Especially those for whom this project was a common commitment, namely Marcio Borgonovo dos Santos and Marcelo Peduzzi de Castro.

Finally, but not least, I would like to thank my co-workers from Federal Institute of Education, Science and Technology of Santa Catarina to trust in my work and support my absence during that time.

Resumo

O uso da tecnologia tem crescido nas últimas décadas nas mais diversas áreas, seja na indústria ou no dia-a-dia, e é cada vez mais evidente os benefícios que traz. No desporto não é diferente. Cada dia surgem novos desenvolvimentos objetivando a melhoria do desempenho dos praticantes de atividades físicas, possibilitando atingir resultados nunca antes pensados. Além disto, a utilização da tecnologia no desporto permite a obtenção de dados biomecânicos que podem ser utilizados tanto no treinamento quando na melhoria da qualidade de vida dos atletas auxiliando na prevenção de lesões, por exemplo. Deste modo, o presente projeto se aplica na área do desporto, nomeadamente, na modalidade do surfe, onde a ausência de trabalhos científicos ainda é elevada, aliando a tecnologia eletrônica ao desporto para quantificar informações até então desconhecidas.

Três fatores básicos de desempenho foram levantados, sendo eles: equilíbrio, posicionamento dos pés e movimentação da prancha de surfe. Estes fatores levaram ao desenvolvimento de um sistema capaz de medi-los dinamicamente através da medição das forças plantares e da rotação da prancha de surfe. Além da medição dos fatores, o sistema é capaz de armazenar os dados adquiridos localmente através de um cartão de memória, para posterior análise; e também enviá-los através de uma comunicação sem fio, permitindo a visualização do centro de pressões plantares; dos ângulos de rotação da prancha de surfe; e da ativação dos sensores; em tempo real. O dispositivo consiste em um sistema eletrônico embarcado composto por um microcontrolador ATMEGA1280; um circuito de aquisição e condicionamento de sinal analógico; uma central inercial; um módulo de comunicação sem fio RN131; e um conjunto de sensores de força Flexiforce. O *firmware* embarcado foi desenvolvido em linguagem C. O *software* Matlab foi utilizado para receção de dados e visualização em tempo real.

Os testes realizados demostraram que o funcionamento do sistema atende aos requisitos propostos, fornecendo informação acerca do equilíbrio, através do centro de pressões; do posicionamento dos pés, através da distribuição das pressões plantares; e do movimento da prancha nos eixos *pitch* e *roll*, através da central inercial. O erro médio de medição de

força verificado foi de -0.0012 ± 0.0064 N, enquanto a mínima distância alcançada na transmissão sem fios foi de 100 m. A potência medida do sistema foi de 330 mW.

Palavras-Chave

Centro de Pressões Plantares, CoP, instrumentação de prancha de surfe, avaliação no surfe, surfe, pitch, roll.

Abstract

The use of the technology has been increase on last decades in many different fields. On sports is not different. Everyday new developments came to help athletes improving their performance, allowing achieve results never thought. Beyond that, the use of technology allows the biomechanical data acquisition which can be used for training development or in injuries prevention, for instance. Thus, this project is developed in sport field, particularly in surf modality where the absence of scientific studies still present, combining electronic technology and sport to quantify information that remaining unknown.

Three basic factors known as influence surfer's performance were verified: balance; feet positioning; and surfboard's movement. These factors led to the development of a system that is capable to measure them dynamically trough measuring plantar forces and surfboard's rotation. Beyond measuring these factors, the system is able to store the data locally trough a mass storage device and send them through WiFi, allowing the visualization of the Center of Pressure; surfboard's rotation; and sensors' activation; in real time. The device is an electronic embedded system composed by a microcontroller ATMEGA1280; a analog conditioning and acquisition circuit; an inertial measurement unit; a WiFi module RN131; and two matrix of Flexiforce force sensors. The embedded firmware was developed in C Language. The software Matlab was used to receive and show the data in graphics in real time.

The tests have shown that the system works properly fulfilling the project requirements providing information about balance, through the center of pressure; feet positioning, through the plantar force distribution; and surfboard's movement on pitch and roll axes, through the inertial measurement unit. The force mean error of measurement of force verified was -0.002 ± 0.0064 N, meanwhile the shorter data transmission distance was 100 m. The measured power in operation was 330 mW.

Keywords

Center of Pressure, Centre of Pressure, CoP, surfboard instrumentation, remote surfing assessment, surfe, pitch, roll.

Index

Index of Figures

Index of Tables

Acronyms

ASP	–	Association of Surfing Professionals
GPS	–	Global System Positioning
LE	–	Lower Extremities
CoP	–	Center of Pressure
IMU	–	Inertial Measurement Unit
EEPROM	–	Electrically Erasable Programmable Read Only Memory
SRAM	–	Static Random Access Memory
RISC	–	Reduced Instruction Set Computing
JTAG	–	Joint Test Action Group
ADC	–	Analog to Digital Converter
UART	–	Universal Asynchronous Receiver/Transmitter
SPI	–	Serial Peripheral Interface
OPAMP	–	Operational Amplifier
PCB	–	Printed Circuit Board
SMD	–	Surface Mount Device
USB	–	Universal Serial Bus
I/O	–	Input / Output
RTC	–	Real Timer Counter/Clock
LSB/MSB	–	Least significant bit / Most significant bit

1. INTRODUCTION

Nowadays, electronic systems have been applied in many distinct areas beyond industrial applications such as home automation, car controlling, gadgets, medicine and even sports. The cost reduction of the electronic components and the popularization of technology for daily use also contribute for the development of innovations either for simple mobile phone applications or more complex systems that can control the electric consumption in a house, for example.

Following that trend, in sports the electronic systems have been used in different manners, e.g., for obtaining better accuracy in real-time acquisition during running competitions. However, they have also been applied in sports to assist in movement capture and analysis in order to understand the body dynamic behavior. Those applications can provide important information for athletes and coaches, figuring out the best technique that can be used to increase performance and help understanding the mechanisms of injuries which is a concern of professionals.

This project aims to develop an electronic system to be used in action sports using a board[1] by acquiring data from force sensors, accelerometer and gyroscope to provide information such as feet positioning, CoP displacement, and board's rotation. Surf was the sport chosen to be the application environment due to the familiarity that the candidate has with this sport, making it easier to understand the specific needs.

[1] Board is general in this context which can be applied as: surfboard; skateboard; snowboard; ski; etc.

1.1. BACKGROUND

The idea behind this project follows from a previous work [1] where some changes were identified as a needed improvement on the technology used, namely in the sensors applied in that situation to collect plantar pressure. In addition, no quantitative measurement was presented, neither relating feet positioning nor plantar pressures involved in wave riding. Following those suggestions and the state of the art analysis, this project also includes an inertial measurement unit allowing to determine the surfboard's movement altogether with the plantar pressure distribution.

1.2. OBJECTIVE

This project aims to develop a system which will allow to measure and collect biomechanical parameters of a surf practitioner while surfing. Therefore, the system has to provide local backup and wireless data transmitting so that the real-time information is collected and sent to the remote station where it can be processed. To achieve this goal, the project was elaborated with a microcontroller platform as the core of the system which is capable of acquiring, converting and processing the data. The goals of the project were pursued into five main topics:

- Find a sensor capable of measuring plantar force;
- Use of a inertial unit to acquire the surfboard movements;
- Develop an electronic platform to acquire data from force sensors and a Inertial Measurement Unit;
- Develop the embedded firmware;
- Test and validation of the system;
- Writing the report.

1.3. SCHEDULE

This project was developed under the Master in Electronic and Computer Engineering program of the School of Engineering of Polytechnic of Porto with a limited period of time of one year. In order to manage the time usage, a schedule was done according to the objective which is presented in APPENDIX A. The schedule includes a group of tasks like hardware development; research of the state of the art; tests; and report writing; for example.

1.4. REPORT ORGANIZATION

Chapter 2 presents the state of the art related to measuring biomechanical parameters during surf practice, apart from a description of the major parameters and features of surfing. Chapter 3 and 4 present project's development, divided into its main parts: sensors, hardware development and firmware. Chapter 6 presents the system validation tests and results. Finally, Chapter 7 presents the conclusions and identifies some options for future work.

2. SURF CHARACTERIZATION

2.1. INTRODUCTION

During the last decade, surfboard ridding has seen an increasing number of followers and suffered an increasing attention from the media. This fact takes the sport to another level, attracting much more investment for this sector – mainly in surf wear fashion - which was condemned on the past because of the prejudice about surfers and their lifestyle. Different from most Olympic sports, although its practice is equally older, surfing has a recent history as a professional sport, dating from the 80's, the beginning of the professional contests [1].

Apart from that, the sport of riding waves also follows the technology evolution. Different kinds of materials have been applied in different situations in order to help surfers to increase performance. Furthermore, the use of technology during sports practice minimizes the difference between the coach and athlete's perceptions on the athlete's performance [2] since by comparing the expected pattern with the movement performed, both athlete and coach are learning [3]. Due to that, surfing has been following the trend of using technology to increase the surfer's performance although the major commercial developments focus on improving equipment quality like surfboards and wetsuits, for

example. One of those improvements is the process and materials which are used to manufacture a surfboard where pre shape for the surfboard can be done by CNC machines [4] or the shaper could use an industrial method named thermoforming[2], either with carbon fiber or Kevlar, to produce it. Like surfboards manufacturing, another industry applying good effort to improve the quality of their products is fins industry. They are using carbon, for instance, instead plastic, providing an extremely hard and lightweight product which increases the drive, pivot and hold [5]. Also the manufacture of wetsuits with new materials and fabrication techniques appear every year, aiming to provide more flexibility, comfort and warmth to the surfer. Despite all these improvements, good equipment is not enough to increase the surfer performance. It is necessary hard training inside or outside the water and good physical conditions [6]. Most scientific studies about surfing are focused on surfer's injuries and physiological behavior [7-12] which seek to figure out how the surfer's body is affected by the sport technique and which are the body responses to that kind of stimulation, which means trying to understand the reaction of the body for some physical external stimuli. Apart from these aspects, there are some technical studies related to paddle movement [13, 14] where they conclude that paddling requires a high energy expenditure by upper body which is similar to that of competitive swimmers and surf lifesavers. They also conclude that the relation between heart rate and oxygen consumption reserve during arm paddling is not straightforward, suggesting the use of an individual relational equation. Furthermore, a qualitative description of some of the major surfing movements and manoeuvers identify time expenditure for different activities and surfer movements analyzing surfers during the practice, even in professional or recreational sessions [6, 15, 16].

As a professional sport, surfing has its own international association in which rules and standards are set for the contests. The Association of Surfing Professionals (ASP) is the responsible agency to managing the ASP Rule Book [17], among other activities like anti-doping and ranking control, for example. The ASP Rule Book is used as a reference for the contests' judges to score every surfer's wave according to the parameters listed.

[2] http://www.powerlightsurfboards.com.br/

2.2. SURFING CHARACTERIZATION

From the literature, it is seen that just a small part of the surfing activity is spent on wave riding [11-13, 16, 18], but it does not mean that a good physical and physiological condition is not necessary, since others factors, like wave conditions and psychological stress, may also be present.

Farley *et. al.* concerned about physiological demand of competition surfing, using a heart rate monitor, a global system positioning (GPS) and a time-motion analysis (using video) to acquire heart rate data, velocity and time spent in each category of movement. The result, for that particular situation, shows that the average covered distance per heat was 1,605 ± 313.5 m, with an average speed of 33.4 ± 6.5 km/h [11].

Otherwise, studies like Mendez-Villanueva *et. al.* e Loveless *et. al.*, try to evaluate how the surfers use their technique during the surf session analyzing paddle movement [13, 16]. Mendez-Villanueva *et. al.* also evaluated the profile of men's competitive surfing during an international competition. To figure out the time expend during surfing, four distinct movements' categories were analyzed as follows: paddling, wave ridding, stationary and miscellaneous. Each category had had the percentage of 51%, 42.5%, 3.8% and 2% of the total surfing time, respectively. In contrast Meir *et. al.* obtained percentages of 44% and 35% for paddling and stationary categories in recreational surfing and 5% for wave ridding [19]. These results show the gap existing in time spent on each category in different purpose practices. With 206 waves, the average length of riding wave was 11.6 seconds [16]. In contrast, Lowdon *et. al.* reported higher numbers, in between 20.0 and 23.7 seconds, both in international contests [20]. These differences are justified by the competitions taken place in different locations and surf conditions. Despite to be the major goal of surf, wave ridding occupies only a small period of time (~ 3.8% - 5%) of the total surfing time corresponds to this category of movement. Take into consideration a surfing session of one hour, it means that the surfer is able to ride 10.6 ± 3.1 waves per hour. Furthermore, the environment where surf is practiced involves different kind of conditions, for example different wave sizes, type of breaker and line-up situation [18]. These facts could explain the absence of quantitative studies in this sport, generating an underexplored research area.

According to the ASP Rule Book [17] , there are some rules and criteria to be followed for the contests to receive the ASP seal. This book is also used to define the judgment criteria

which must be adopted by every judge, creating a standard judgment. There are five major elements that contribute to increase wave score:

- Commitment and degree of difficulty;
- Innovative and progressive manoeuvers;
- Combination of major manoeuvers;
- Variety of maneuvers;
- Speed, power and flow.

The judges use these five elements to score each wave ridden by the surfer. The scoring range varies between zero and ten.

Although the ASP Rule Book do not specify criteria over ride distance and numbers of maneuvers, Peirão *et.al.* conclude that these two factors have indirect effect on the judgment result in the events analyzed. The study reaches a significant correlation between the scores assigned by judges and the judgment criteria, indicating that wave riding time and frequency of manoeuvers affect judgment indirectly [15].

2.3. MAJOR SURFING MANOEUVERS

During a surf session, conditions can change quickly since every wave is different. For a surfer, being able to reach a great wave riding in these dynamic conditions constitutes a challenge. Furthermore, being the set of manoeuvers quite large, surfers are always looking for evolution and performance improvement. During training sessions, surfers frequently stay 4-5 hours in the water [18] thus requiring high technical and skill abilities to support it [6, 21].

In 2007, a study was conducted to describe the physiological demands of surfing using qualitative analysis of major manoeuvers. According to Everline, the primary movements which the surfer must be capable of are paddling and take-off, where the surfer moves around to get the surf zone and also take the wave. It is necessary that the surfers have the skills to know the correct angle of take-off and velocity, which can change according to wave type. Stand up quickly on the correct instant and have the balance are the next step after the take-off [6]. Figure 1 shows stand up and take-off movements.

(a) – moving elbows back

(b) – hands position

(c) – raising the body

(d) – finishing take-off

Figure 1 – Take-off movement. Four different moments describe the action.

First the surfer needs paddle to reach the speed to follow the wave until the breaking zone. After that, he starts the take-off movement moving elbows back (Figure 1a), putting the hands on board (Figure 1b) and pushing the surfboard down and the body up (Figure 1c). On the last movement the surfer must control the surfboard to follow the wave flow (Figure 1d). After take-off the surfer is able to ride the wave.

According Everline, the surfer's position in relation to the surfboard is frequently half squatting with flexed knees after take-off. From this period onwards the movements depend of the leg strength and the balance, which allows the surfer control the surfboard direction and movement as depicted in Figure 2a. After take-off and positioning, the surfer executes the bottom-turn changing the natural path of the surfboard in direction of the beach, executing a turn on the base as depicted in Figure 2b. The bottom-turn is a critical manoeuver that allows the surfer follow the wave wall direction. Surfer's foot position will dictate the mechanics of the manoeuvers during and after the bottom-turn. The surfer changes the balance putting more weight to the surfboard's tail and flexing his knee [6].

(a) – positioning (b) – bottom turn and positioning

Figure 2 – Surfer positioning after take-off and bottom turn approach.

There are two foot stances in surf: regular foot and goofy foot. The difference between them resides on the foot that goes in front. Regular footers put the left foot in front while goofy footers put the right foot in front. For the regular ones, a left turn on the wave requires a dorsal flexion of the feet, rotation of hip and shoulders to the left. On the other hand, a right turn with a regular foot position involves plantar flexion and rotation of the hip and shoulders to the right.

Another factor that must be observed is the acceleration because controlling the acceleration the surfer is able to ride the wave and execute different maneuvers, among other elements listed. In order to increase de acceleration, the surfer can make fine turn up and down the wave using body movements or can move the surfboard with flexion/extension movements using the hip and knee. Moreover, if the surfer would like to reduce the velocity he can put more weight on the surfboard's tail or make strong curves. It is also important to observe that the surfer can ride a wave in different directions in relation of the wave's face. Basically there are two directions: left and right. Left wave occurs when the wave break from the left to the right in relation a one observer on the beach. On the other hand, right waves occur when the wave breaks from right to left.

Peirão *et.al.* analyzed two professional international championship events with 21 competitors from different countries, by studying variables included in the ASP judging criteria elements and major manoeuvers, the surfer position in relation to the wave, length of the ride and frequency of major manoeuvers. The major manoeuvers were described as: Carving; Re-entry; Floater; Cut-back; Three-sixty; Tube; and Aerial [15].

From both studies, we can verify that the execution of most manoeuvers is based on three controlling factors: (1) upper body movements, (2) trunk rotation, and (3) Center of Pressure (CoP) displacement. The control of these three factors altogether with the surfer's knowledge of the wave will provide the ideal conditions to execute those manoeuvers. Table 1 presents the set of manoeuvers listed and the description of the controlling factors.

According to Lierbermann, the feedback information of the movements is a major factor to allow systematic correction in sport performance, though the knowledge of the goal is also necessary to perceive the need to carry out corrections [3]. The most common manner to evaluate a surf athlete during the training riding is using a visual sense where the coaches observes the performance of the surfer from the beach, giving information to the surfer on the execution of the manoeuvers. Nowadays, video motion analysis has been used to record the surf session which will be after evaluated by coaches and surfers. Furthermore, different angles and more than one camera - including waterproof cameras - have been used to get the largest possible information to evaluate and understand the surfer movements. This method has been successful to date, however it is fully dependent of the coaches' knowledge and his perception of how the surfing is done. Thus, using a qualitative analysis, it is more difficult to measure the physical variables involved like force, angles and speed, for example.

Table 1 – Set of manouvers listed and the controlling factors to execute it.

	Factors			
Manoeuver	**Upper body**	**Trunk rotation**	**CoP displacement**	Obs
Carving	Body and arms opened in the beginning, moving arms down when reaches top wave.	Slight rotation.	Starts moving the weight forward to reach the top of the wave. Then, move the weight backward to make pressure on the back foot.	
Re-entry	Follow trunk rotation.	Strong rotation (almost 180°).	Starts moving the weight forward to reach the top of the wave. Then, move the weight backward to make pressure on the back foot to complete the turn. After it, move weight forward to take-off.	
Cut-back	Body and arms opened in the beginning, moving arms down when reaches top wave.	Slight rotation on the beginning. Strong rotation in the end when reaches the wave's lip.	Starts moving the weight forward to reach the top of the wave. Then, move the weight backward to make pressure on the back foot to complete the turn. After it, move weight forward to reach the wave's lip. Then, move the weight backward to make pressure on the back foot to complete the turn.	
Floater	Keep body position centered during riding to control the balance, moving down when speed get slow.	Slight rotation.	Most part of time the center of pressure is forward.	

	Factors			
Manoeuver	**Upper body**	**Trunk rotation**	**CoP displacement**	Obs
Three sixty	Move arms forward in direction of broken wave's lip in the beginning, turning completely with surfer in the end.	Strong rotation. Executes a completely 360° turn.	Starts moving the weight forward to reach the top of the wave. Then, move the weight backward to make pressure on the back foot to complete the turn. After it, move weight forward to takes-off.	
Tube	Controls the balance. However, one hand could grab the surfboard's rail to help in surfboard's control, or touch the water to increase grip and decrease the speed.	Once in balance, no trunk rotation is required.	Starts moving the weight forward to reach the most critical section. Then, control backward and forward to follow inside wave.	
Aerial Reverse	Move arms forward in direction of no broken wave's lip in the beginning. One hand could grab the surfboard's rail while in the air. Upper body spins around surfboard's normal axis.	Strong rotation. Executes a completely 360° turn.	Starts moving the weight forward to reach the top of the wave. When landing, controls weight distribution.	

2.4. COMMON INJURIES

Like other radical sports, surfing uses nature as the environment to practice, which provides uncontrolled situations for the surfer. Besides this, the complex movements involved on surfing activity can become a potential factor to surfers getting injured.

Injuries are the main cause of surfer absence from competition, the ASP website reports almost eleven injuries from 2011 to 2012 on professional men's ranking. Apart from that, during the 2013 eleven more injured surfers were registered, leaving them out at least one event (out of 10 events). The worst reported case is the absence from 8 events [22]. In these reported cases, most surfers developed the injuries during training sessions, while only a few happened during competitions.

The complex movement during surfing raises questions on the surfer's vulnerability to injuries. Although the most frequent type of injuries is lacerations, (41% - 46% of all surfing injuries [9]), there are several injuries related to overextending. Mainly at the professional level, musculoskeletal injuries are another type of common injury [6, 18]. Knee stress, bad posture and again overextending generate the right scenario for injury to occur. Considering a paddling time of 44-51% [12, 16], surfers can be more susceptible to injuries due to muscular fatigue, for example.

Nathanson *et al.* [7] reported that 62% of the surfing injuries occur during wave riding. Although, most of them could be connecting with excessive use of the upper body during paddling time, like shoulder dislocation and shoulder strain that represents 35% of the total upper body injuries. The author reaches that 37% of the total acute injuries occurs on the lower extremities (LE) where foot, knee, and ankle injuries were the most common. The survey reports significant percentage of the total LE injuries on foot (75%) were lacerations. On the other hand, knee injuries such as sprains, meniscal tears, and dislocations represent 70% of the knee injuries. Although they did not specify other kinds of foot injuries beyond lacerations [7], over use of plantar flexion and dorsal flexion could have been responsible for ankle strain related by ASP.

As a result of excessive body torque, 28% of upper body injuries occur while performing maneuvers which was confirmed by [6]. Furthermore, it is pointed out [7] that up to 13% of the upper body injuries occur on the trunk whereas 43% occur on the back, 35% on the

chest wall, the musculoskeletal strain injuries were prevalent although factures were reported.

2.5. EVOLUTION OF SURF EVALUATION

In 2009, Bona *et.al.* tried to understand the wave riding characteristics, from the surfer's point of view, taking into consideration three principal parameters: foot positioning; balance; and distribution of force on the surfboard's decks. An electronic system was developed to collect the data from strain gauge sensors on the board and transmit wireless to a remote central. The project was finished before water tests because the strain gauges sensors did not work properly after fiber resin reaches total cure. Overall, the laboratorial tests showed that it was possible to obtain the necessary data based on plantar pressure [1].

In 2011, two companies from Spain launched a surfboard with an inertial measurement unit, strain gauges, and GPS which they called as SurfSens Project – the first technological surfboard on the world [23] . The results were not disclosed and therefore the concept could not be verified and no others news were released since.

In 2013, a North American company started a campaign to raise funds to support the development of a new sport tracker known as Active Replay. The company already developed a mobile application to trace snowboards and skiers using smart phones resources. However, the new system promises an innovation allowing user to stick the tracker on the surfboards giving information about distance, speed, size of wave, height of air manoeuvers, number of turns, length of the ride, among others features. The device stores data that is uploaded to a server after use. The launch in the market is scheduled for March 2014 [24].

These projects aim to describe the behavior of the surfer and the surfboard during the wave ride, though none of them presented until the moment scientific results. Once more, this could reflect the difficult of quantitative evaluation of the surfing. However, they prove that there are some efforts in this area, either in academia or in the private sector.

2.6. VARIABLES TO MEASURE

The studies conducted by Everline and Peirão describe the major manoeuvers and basically, all of them are based on the same body movements [6, 15]. By controlling the balance and feet positioning, the surfer is able to change the surfboard's direction and velocity. By rotating the upper body, the surfer might control the surfboard's direction as well. By choosing feet positions, the body movements can be controlled. Combining all movements, the surfer is able to complete the set of major manoeuvers. Thus, there are three variables that must be measured:

- Balance;
- Feet position;
- Surfboard's position.

According to Winter, dynamic balance is the ability to perform a task maintaining a stable position [25] and it is also describe as the process of maintaining the body's center of gravity vertically over the base of support [26]. Furthermore, from Hrysomallis literature review, the CoP displacement is an indication of balance, where minimum CoP motion indicates good balance, which is normally measured by using force platforms [27]. Therefore, to measure the CoP on a surfboard, one should expect to have some sort of force platform. Although, the unstable support provided by surfboards and the type of platforms constructions available nowadays[3] a unidirectional measurement of force is proposed on this project using a distributed plantar pressure measurement system.

Using the same principle to measure plantar pressures, feet position determination is a consequence of plantar pressure measurement distribution. With the array of force sensors measuring the surfboard's reaction force on each foot, it is possible to determine the feet's positions.

On the other hand, to measure surfboard's angle, it is necessary to use a different kind of sensors that provide information about acceleration and rotation, converted into angles and

[3] The force platforms are composed by hard materials like aluminum and steel and measure the force on three directions in which: normal force and shear forces. Furthermore, the force platforms are commonly installed over hard support to minimize noise from undesired vibrations from the surrounding structure.

then, the surfboard's movement can be determined. Figure 3 shows the system components and their orientation.

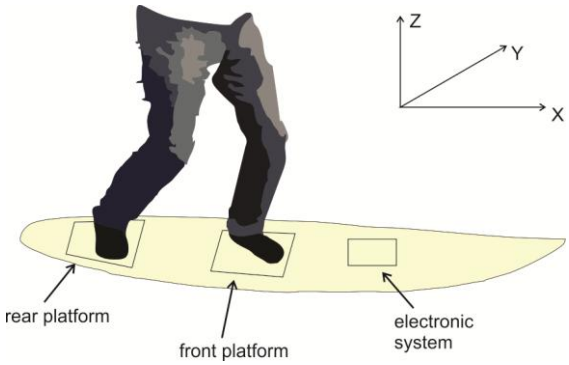


Figure 3 – System positioning over the surfboard and surfboard's orientation.

2.6.1. CENTER OF PRESSURE

To measure the plantar pressure, force sensors based mainly on capacitive and resistive technologies have been used. The sensing elements are normally arranged to form a matrix of sensors as shown in Figure 4 from Tekscan® Incorporated (USA) that uses force sensing resistors (FSR) as the sensing element.

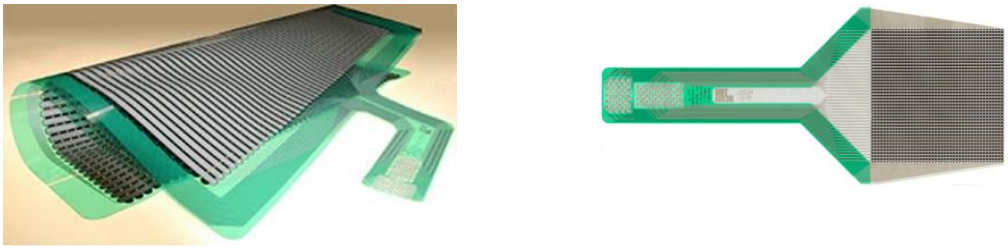


Figure 4 – Matrix of force sensor elements to measure pressure distribution by Tekscan®.

Depending on the matrix space resolution it is possible to have enough accuracy to define feet position precisely.

2.6.2. SURFBOARD ROTATION

The surfboard can perform rotations about any axis in three-dimensional (3D) space during wave riding. Furthermore, it is recognized that these rotations can be performed independently from each other. The surfboard's rotational axes are depicted in Figure 5. By convention for this project, the axes x, y and z are positioned longitudinally, transversely and normal to the surfboard, respectively. The rotation's direction is positive when in the clockwise direction.

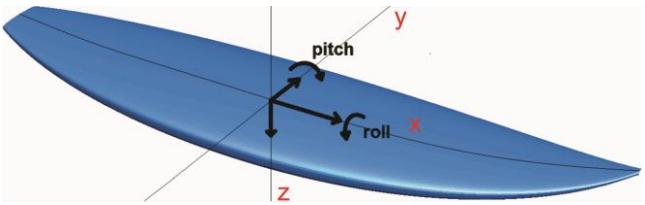


Figure 5 – Surfboard's pitch, roll rotation axis orientation.

In electronic systems developments it has been usual to use an electronic Inertial Measurement Unit (IMU) to measure rotation. The IMU has at least one accelerometer and one gyroscope to provide information about acceleration and rotation about one axis, however, it is common to find IMU's that integrate other components, e.g. barometer and compass. Generally, the signals from these devices are processed by a microcontroller or microprocessor to provide the information of rotation and direction.

In this project, an IMU with 5 degrees of freedom measures accelerations on each direction – X, Y and Z – and roll and pitch movements, i.e. rotation along X and Y. This data allows determining the surfboard's movement.

3. HARDWARE

The absence of studies characterizing surfing during its practice is noticeable. The characteristics of the environment where is practiced along with the complex movements, creates a tough scenario to seek a solution for its characterization. Although surfer performance has been evaluated over the years in professional competitions, it remains difficult to get a straightforward continuous feedback for surfer and coach, about the surfer´s performance evolution. In this chapter, a solution to collect data from sensors under surfer's feet and on the surfboard to provide information about surfing movements is presented.

The system is composed by five blocks as depicted in Figure 6 according to their feature. First of all, the Input block contains the system inputs from the force sensors and IMU and is characterized by converting physical quantities (acceleration, angular rotation and force), into electrical quantities (voltage and electrical resistance). On the other hand, the Output block contains a micro SD Card as the data storage device and stores all information processed to ensure no data is lost when real-time communication fails. The Processing block conditions the signals from the Input and processes them to obtain the specified variables. There is also a Communication block that provides either wireless connection (using WiFi) or cable connection (USB) to enable real-time operation and system debug.

Finally, in the Software block the real-time charting and local data backup is done using Matlab

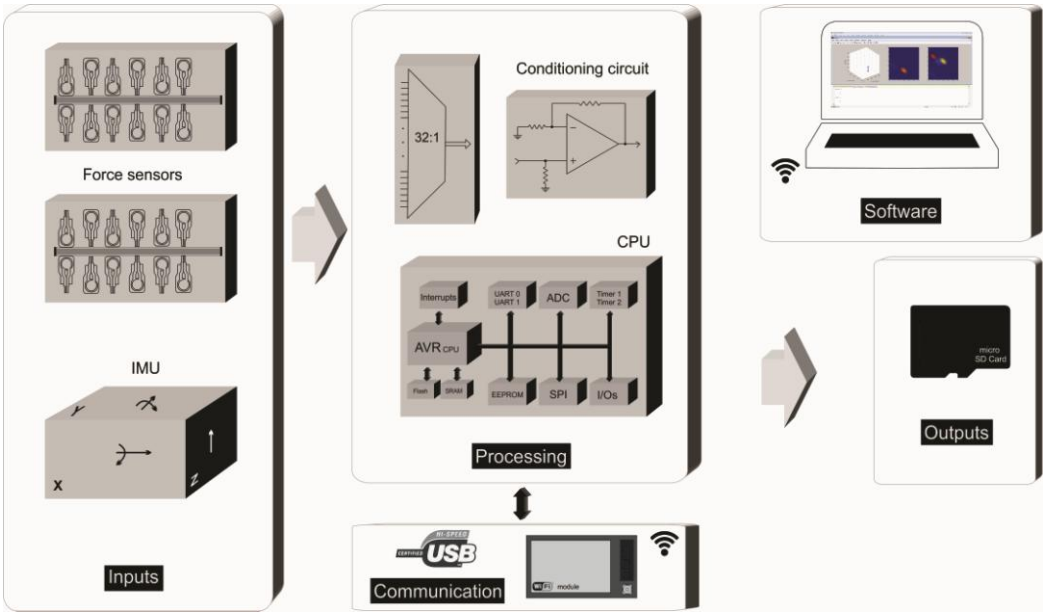


Figure 6 – System block diagram.

3.1. FORCE SENSORS ARRAY

In order to measure the feet pressure, it was necessary to define which technology would be used on this project. To figure out the possibilities for this implementation, a study on the available commercial force sensors was done of three well-know distributors[4][5][6], taking into consideration the application characteristics (mounting surface - the surfboard's deck) and the available budget. The search was restricted due to the following requirements:

- No Load Cells – Although load cells could measure the force over three directions, giving information about normal and shear force, the construction of a force platform over the surfboard would be extremely tough work. To do that, the load cells should have one side fixed on the base of the platform and the other side fixed on mobile

[4] http://www.digikey.com/ ou Digikey Corporation

[5] http://www.mouser.com/ ou Mouser Electronics

[6] http://www.farnel.com/ ou Farnell

part of the platform and then, to allow the foot positioning measurement, it would be necessary to create an array of small platforms. Furthermore, the materials used in the production, make the load cells heavy for this kind of application. Overall, the price of each load cell could be hundreds of dollars which would make the project not feasible.

- No liquid Pressure Sensors – Once upon starting the searching for force sensors, it is unavoidable not to look for pressure sensors which measure the force applied by a fluid over a known area that gives the information about the fluid pressure. Although a lot of options of pressure sensors appeared during the search, the use of a fluid to measure the force in this project requires a huge effort to create a matrix of dots where each one has a fluid passing through it and then the fluid pressure have to be measured by the sensors. Furthermore, the number of pressure sensors must be the same of the dots on the matrix, which also would make the project financially inappropriate.
- Individual sensors to form a matrix – As previously defined, it is necessary to implement a matrix of sensors to measure the feet position. Although there are some companies that produce that sort of matrices, the price is high. Furthermore, the companies typically do not sell only the matrices; they sell an entire solution which includes hardware and software. To reduce the cost of the electronics for the prototype, individual sensors to implement a matrix was the chosen solution.

These requirements have limited the research's result remaining a few options including: Force Sensing Resistor[7] (FSR®) and Flexiforce®[8] Sensor. Both sensors could be applied on this project, however the Flexiforce® was chosen. Two force ranges was selected for comparison in order to test which one best suit for the project: 4.4 and 110 N. The model A301 was chosen because of a shorter length of the terminal area, which allows better control of the element's positioning during assembly of the matrix. Figure 7 illustrates the sensor and the datasheet is shown in APPENDIX B.

[7] Trademark of Interlink Electronics.

[8] Trademark of Tekscan Incorporation.

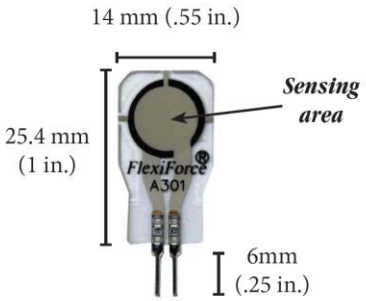


Figure 7 – Flexiforce A301 model. [28]

The Flexiforce sensor is a thin-film force sensor designed using a flexible printed circuit technology. Sensors assemblage is composed by two polyester layers for support and one layer of a pressure sensitive ink in between. Each support layers receives a conductive material to allow the electrical contact and an adhesive to glue the layers together. The pins are accessible for 2 male pins crimped through the layers.

The sensitive ink on the sensor is composed by a resistive material that changes the resistance when subject to a force variation. The resistance of the unloaded circuit is greater than 5 MΩ and decreases when force is applied. A typical sensor's response for Force *vs.* Resistance/Conductance of a 100 lb range is shown in Figure 8

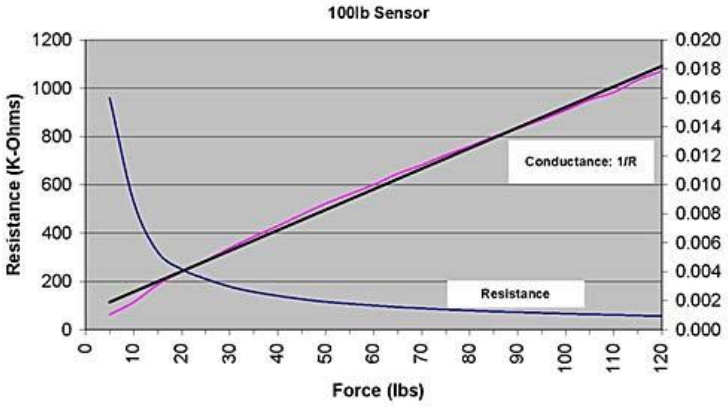


Figure 8 – Typical response for Flexiforce sensors. [28]

According to the sensor's specification, the sensitive area is 0.71 cm^2 and it is given by Equation 1, where '*r*' represents sensor's radius of 0.4765 cm.

$$A_{sensor} = \pi \times r^2 \quad (1)$$

Due to the sensor's range of 4.4N to 110N and area calculated, the maximum measureable pressure is 6.17 and 154.21 N/cm^2, respectively.

In what relates to the active measurement area for each foot, previous work [1] shows that it is acceptable a coverage area of 30 cm x 30 cm on each foot as a result of changing constantly feet positions. Then, considering the foot typically inside that area and the limited number of 24 sensors available, the sensing elements were equally distributed into two distinct areas according the scheme presented on Figure 9. The sensors areas are named rear platform and front platform for identification purposes. In this way, it is possible to cover an area of 17.12 cm^2. The sensors chosen distribution tried to ensure that at least one sensor would be activated under the forefoot and another under rearfoot.

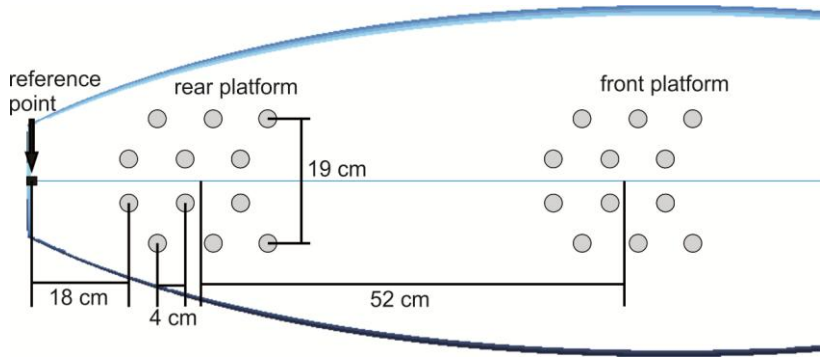


Figure 9 – Force Sensors distribution and platform identification.

Each sensor has 2 pins to connect it to the circuit which must receive 24 wires for each platform. Due to the thickness of the sensors – 0.203 mm, it is important that the cables do not physically interfere on the measurement, which means that the wires must be thin. Furthermore, the cable's current range must accept the conditioning circuit specification and must be cheap. Owing to these requirements, a 24-way flat cable of 0.89 mm thickness and 30.5 mm width was chosen. The cable assemblage is shown in Figure 10.

Figure 10 – Rear and Front platforms with cables assembled.

To connect the sensors to the conditioning circuit, a wire-to-board connector was used providing the electrical contact through soldering terminals.

3.1.1. SENSOR'S CALIBRATION

The sensor's calibration response for load applications was performed by using the protocol suggested by Tekscan [28]. In order to obtain the sensor's response and to quantify the measured error, two tests were performed for each sensor. The tests aim to measure the output voltage from the conditioning circuit and the data converted by using the calibration equations.

On the first test, the voltage output from the conditioning circuit was recorded to verify the sensor's linearity response. After that, the calibration equations were calculated and inserted on the firmware application.

On the second test, the conversion results were sent through USB for a terminal and the result was recorded to verify the measurement error after calibration.

Only the sensors of 4.4 N range were calibrated, a set of nine loads was used comprising eight loads of 50 g and one load of 28.6 g, plus a bar of 9.62 g used to ensure that the force were being applied on the sensitive area of the sensor. The total load used in calibration is shown in Table 2.

Table 2 – Force range used in sensor's calibration.

Load [g]	Force [N]
59.62	0.58
109.62	1.08
159.62	1.57
209.62	2.06
259.62	2.55
309.62	3.04
359.62	3.53
409.62	4.02
438.62	4.30

First of all, the loads were applied to the sensor and the voltage measured on the output of the conditioning circuit by using a digital oscilloscope. The results have shown that all sensors presented a coefficient of correlation (R^2) higher than 0.98 with one exception of Sensor 21 that presented 0.93, between voltage and force. The Table 3 shows the R^2 calculated.

Table 3 – Coefficient of correlation of 4.4 N force range sensors

Sensor	R^2	Sensor	R^2
3	0.995160	15	0.994777
4	0.990279	16	0.992899
5	0.988782	17	0.986208
6	0.996928	18	0.997648
7	0.989282	19	0.999469
8	0.999411	20	0.997267
9	0.989898	21	0.939296
10	0.996232	22	0.998948
11	0.998698	23	0.991836
12	0.991911	24	0.988847

To illustrate the first test response, Figure 11 depicts the data from Sensor 8 which present the highest R^2. Due to the linearity of the voltage response, the equation coefficients were

calculated for each sensor and placed in the ADC conversion function in the firmware as will be explained later.

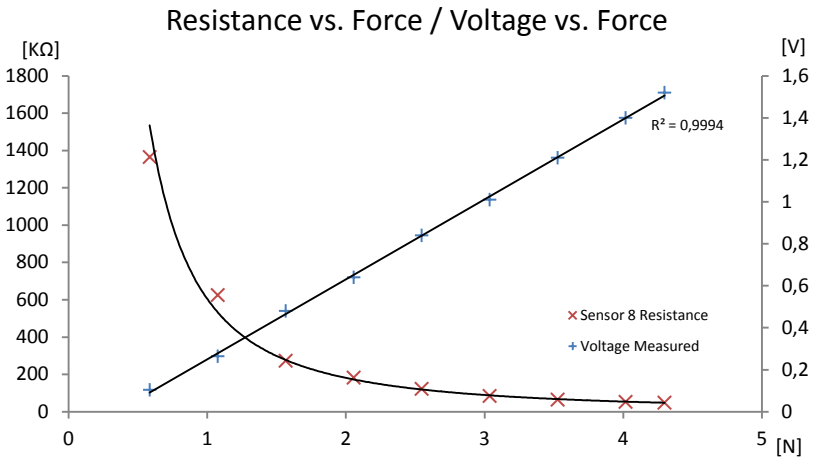


Figure 11 – Voltage and resistance response for Sensor 8.

Secondly, to ascertain the measurement error, the second test was performed by loading the sensors with the same force range and checking the output conversion on a terminal. The highest measurement error verified is -0.25 N for the range of 3.52 N on Sensor 18 which represents 7.18 % on that range. However, the mean error is -0.012 ± 0.064 N which is not significant in this application. The Table 4 shows the mean measurement error for each sensor calculated from the response of the whole force range.

Table 4 – Measurement error verified after calibration.

Sensor	Meas. Error [N]	Sensor	Meas. Error [N]
3	-0,00732	15	-0,00287
4	-0,03398	16	0,00380
5	-0,01843	17	-0,05843
6	-0,03620	18	-0,08509
7	-0,01398	19	-0,01732
8	0,02268	20	0,00491
9	0,02491	21	N/A
10	0,01157	22	-0,01398
11	0,01935	23	0,01935
12	-0,02732	24	-0,01843

The Figure 12 shows the measurement error associated with the force applied for Sensor 18 and Sensor 3.

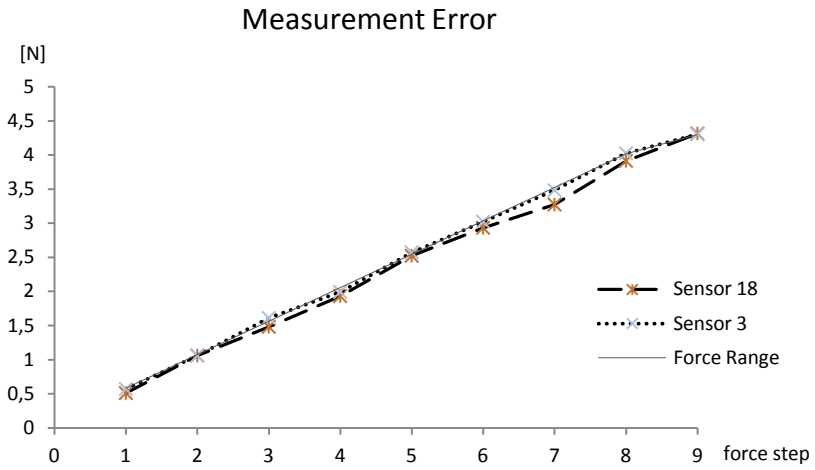


Figure 12 – Measurement error associated with force applied.

3.2. MICROCONTROLLER

The microcontroller can be considered the main electronic component of the system, since it's where processing takes place. Microcontrollers are largely used on embedded system due to the number of on-chip peripherals available and the range of commercial options. Furthermore, the use of microcontrollers allows fast prototyping because of the reduced number of external components to work, e.g. memory and peripherals, in comparison with processors. For this solution, an ATMEGA1280 from Atmel® Corporation was used. The previous knowledge of AVR®[9] 8-bit architecture and the compatibility with system requirements were the choice parameters.

The selected device is a high performance low power 8-bit microcontroller from Atmel Incorporation which is implemented with a RISC architecture that could provide up to 16 MIPS throughput at 16 MHz of operational frequency. The ATMEGA1280 has 128 KB of Flash Memory, 4 KB of EEPROM, which are both programmable through JTAG,

[9] AVR is a trademark of Atmel Incorporation.

furthermore has 8KB of SRAM Memory. Special features can be used to manage the power like six different sleep modes, or to detect Brownout, for instance.

There are different peripherals available like timers, analog to digital converters, serial communication and Real Time Counters (RTC), for example. The list of the main peripherals is shown in Table 5. In the context of this project, the used peripherals are an Analog to Digital Converter (ADC), 3 Timers, 2 Universal Asynchronous Receiver/Transmitter (UART) and a Serial Peripheral Interface (SPI).

Table 5 – Peripherals available in the AVR ATMEGA1280.

Peripheral	Features
Timer/Counter	Two 8-bit Timer/Counter with Prescaler and Compare Mode
	Four 16-bit Timer/Counter with Prescaler, Compare and Capture Mode
ADC	16 channel, 10-bit analog to digital converter
PWM	Four 8-bit PWM channel
	Six/Twelve PWM channels with Programmable Resolution from 2 to 16 bit
USART/UART	Four Programmable Serial USART/UART
SPI	Master/Slave Serial Peripheral Interface
I2C	2 wire serial interface
Watchdog timer	Programmable Watchdog Timer with separate on-chip oscillator

The circuit connections with the microcontroller are shown in APPENDIX C. The ADC is connected to the circuit from channels 9 to 14 where the connections labeled as XRATE, YRATE, XOUT, YOUT, and ZOUT represents de IMU signals from gyroscope and accelerometer, and the connection OUT_SENS is the output of the force sensors conditioning circuit. The signals from the accelerometer must have a parallel capacitor connected to each output to select the bandwidth limit. However, the use of a commercial IMU solution eliminates the need of external components because they are already implemented. Likewise, for the gyroscope the need for a RC circuit is provided by the IMU circuit. That will be explained afterwards on IMU description. To provide an internal voltage reference for the ADC, a capacitor of 100 nF is connected into pin AREF and an internal circuit sets the reference according to the programming code. Another connection to the microcontroller is the WiFi module. This module communicates using an UART

protocol. The pins WiFi_RX and WiFi_TX from the module are connected to the microcontroller UART1 on Port D (RXD1 and TXD1).

To generate a time reference, the Timer/Counter 2 is used in a RTC configuration. In that configuration, it is recommended the use of an external oscillator of 32.768 kHz because this timer is optimized for better accuracy when in RTC mode. Due to that, a crystal X2 is connected to the microcontroller through the pins TOSC1/TOSC2. Two capacitors of 6 pF are connected to the crystal and GND to warrantee the accuracy specified by the manufacturer.

The main frequency is generated by a 16 MHz crystal connected to the XTAL1 and XTAL2 pins on the microcontroller which is the maximum clock frequency allowed for this device. Two 22 pF capacitors are connected to the crystal and GND to avoid unwanted oscillations and noise.

A general external reset circuit is implemented using a RC configuration. The reset circuit is low active and the minimum pulse period to reset the microcontroller is 2.5 µs.

The control of the multiplexed conditioning circuit is done by eight pins from Port A and three pins from Port D. These pins control the multiplexer's output selection and multiplexer enable in order to sweep the inputs from the platform sensors. Moreover, there are five decoupling capacitors connected on the power input.

3.3. CONDITIONING CIRCUIT

The conditioning circuit is responsible for sweeping, receiving and adjusting the signals from the sensors platforms to be acquired before the ADC. The circuit is composed by an analog multiplexer cascade and an operational amplifier (OPAMP). Due to the need of using a matrix with 32 sensors, five 8:1 multiplexers were cascaded, where four of them (U1 to U4) represent the inputs and the last one (U5) combines them for one single output. Afterwards, the multiplexed output is connected to non-inverting OPAMP. Figure 13 shows one multiplexer and OPAMP circuit from the conditioning circuit.

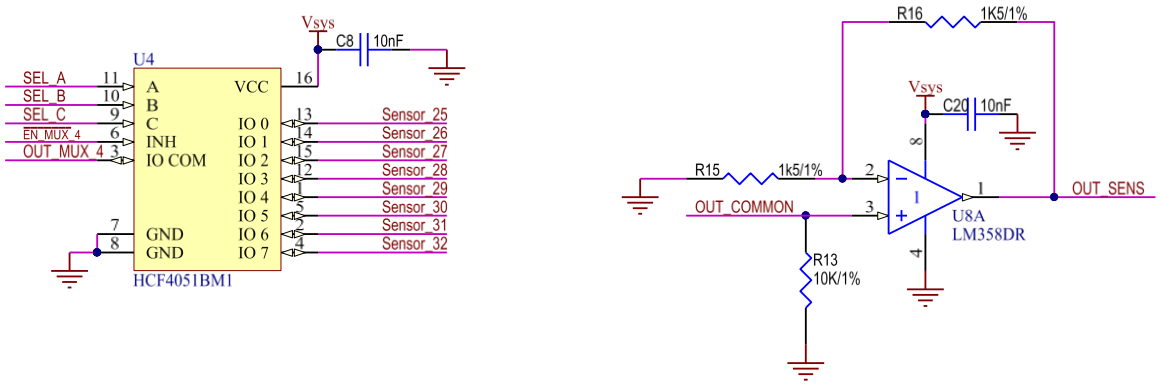


Figure 13 – Conditioning circuit implementation.

The multiplexer circuit uses a HEF4051B[10] 8 channel multiplexer/demultiplexer components. This component is actually bidirectional, however was configured as a multiplexer with eight inputs – IO0 to IO7 – and an output IOCOM. Furthermore, there are three signals to control the input addressing – A, B and C, and one active low signal to enable/disable the multiplexer – INH.

Due to the application in cascade topology, the main features to be considered on the multiplexer circuit are the serial equivalent resistance and propagation delays. When an input is selected, the sensor signal must pass over two multiplexers to be connected to the OPAMP circuit, associating multiplexer resistances to the sensor's resistance. Although the minimum supply voltage accepted is 3VDC, the component datasheet do not specify a typical resistance curve for that range. Take into consideration the typical curve presented for 5VDC, the resistance on conduction is typically 115 – 350 Ω. In worst case, total resistance associated by two multiplexers will be 700 Ω.

Due to the sensor resistance specified by the manufacture being larger than 5 MΩ when there is no load applied and the lowest sensor resistance measured of 24.8 kΩ in maximum load, the associated conduction resistance can be ignored.

[10] http://www.nxp.com/documents/data_sheet/HEF4051B.pdf

On the other hand the propagation delays must be considered during firmware development to ensure proper analog to digital conversion. However, similarly in the conduction resistance, the propagation delays are based on 5VDC supply voltage. Table 6 shows the delays where: HIGH to LOW and LOW to HIGH are the delay for transitions between high and low voltage levels; HIGH to OFF/OFF to HIGH and LOW to OFF/OFF to LOW are the time for signal stabilization after switching.

Table 6 – Dynamic characteristics for HEF4051B.

Parameter	Signal	Typical	Maximum
HIGH to LOW	Input to Output	15 ns	30 ns
HIGH to LOW	Address Selection	150 ns	300 ns
LOW to HIGH	Input to Output	15 ns	30 ns
LOW to HIGH	Address Selection	150 ns	300 ns
HIGH to OFF	Enable to Input/Output	120 ns	240 ns
OFF to HIGH	Enable to Input/Output	140 ns	280 ns
LOW to OFF	Enable to Input/Output	145 ns	290 ns
OFF to LOW	Enable to Input/Output	140 ns	280 ns

Due to the time delays, it is necessary to ensure that the signal acquisition is executed when the multiplexer has completed switched. Otherwise, the acquisition will get the transitions and the signal will do not correspond to the sensors variation. In Chapter 4 the embedded firmware managing properly the delays is described.

3.4. INERTIAL MEASUREMENT UNIT

The IMU gives information about the surfboard's rotation using one accelerometer and one gyroscope. Then, the movement of the object can be calculated according the acceleration and rotation parameters. It is important to note that the position must be calculated using both parameters to provide better results than individual calculation as is described below.

The IMU used (SENS-09268 from Sparkfun Electronics) has five degrees of freedom, corresponding to three axis of acceleration – X, Y and Z – and two rotation axis – pitch and roll. The sensing devices inside the IMU are an ADXL335 accelerometer and an

IDG500 gyroscope [29, 30]. Both devices are assembled to guaranty the same orientation for all axes and the necessary connection to acquire the electrical signals. As a prototyping kit, the signals are accessible for standard 2.54mm pitch block terminal connectors which makes easy to incorporate into a printed circuit board (PCB). However, there are dimensions requirements that must be considered during the PCB design. The SENS-09268 is shown in Figure 14 and the electrical circuit is on APPENDIX D.

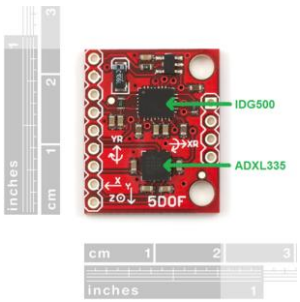


Figure 14 – Inertial Measurement Unit – SENS-09268 from Skparkfun Electronics.

The electrical signals provided by the IMU are analog signals in which the voltage varies according to the device movement. For the accelerometer the signals represent acceleration over each axis. Once on Earth, all bodies are subject to the gravity force – G force, which represents an acceleration of 9.8 $m.s^{-2}$ and must be considered on the accelerometer device measurements. On the other hand, the gyroscope signals represent a rotation around the axis and they are not affected by the G force. When a rotation about any axis is performed, the resultant Coriolis effect causes a vibration that is detected by the gyroscope system and then, the converted signal is proportional to the rotation.

Both devices are connected to the circuit through the analog to digital converter from the microcontroller. The signals are then read and the conversions are based on features presented on the datasheets. The most important features are described in Table 7.

Table 7 – Features from ADXL335 and IDG500.

ADXL335			
Parameter	**Condition**	**Typical**	**Unit**
Input Signal	Each axis		
Operational range		±3.6	g
Sensitivity	Each axis		
Xout, Yout, Zout	Vs = 3V	300	mV/g
Voltage in 0g			
Xout, Yout	Vs = 3V	1.5 – 1.65 (Max)	V
Zout	Vs = 3V	1.5 – 1.8 (Max)	V
Supply		1.8 - 3.6 (Max)	V
IDG500			
Operational range	X-OUT e Y-OUT	±500	º/s
Sensitivity	X-OUT e Y-OUT	2.0	mV/º/s
Voltage reference		1.35	V
Supply		2.7 – 3.6 (Max)	V

3.5. COMMUNICATION – WIFI AND USB

To allow the system to communicate with the user, two different ways have been provided: WiFi and USB interfaces. First of all, USB interface is used as a debug interface. Second, WiFi transfers data acquisition and receives data commands over the air allowing the application to work remotely.

Both ways were implemented using external converters connected to the microcontroller's UART channels, one FTDI232R [31] and one RN-131 [32] modules communicating with the microcontroller through UART0 and UART1 for USB and WiFi connections, respectively. Once the modules were connected, the communication protocols were implemented by the modules being transparent from the microcontroller's programmer view.

The connections between USB converter FTDI232R and microcontroller are shown in Figure 15. Due to the fact of not implementing the flux control, a few connections had to be done to provide a UART-USB conversion, in which: the signals labeled as FTD_RXD and FTD_TXD are the data connections with the microcontroller UART0; the signals D-

and D+ connect the FTDI to the USB connector; the signals LED_RX and LED_TX for LED connection to provide visual information about traffic data; and the power source Vsys and GND. Afterwards, the microcontroller should send the data for the FTDI232R through the FTD_TXD pin and should receive the data over the FTD_RXD. The USB converter data transfer baud rate could achieve 3 Mbps, however the maximum baud rate used was 1 Mbps, limited by the microcontroller.

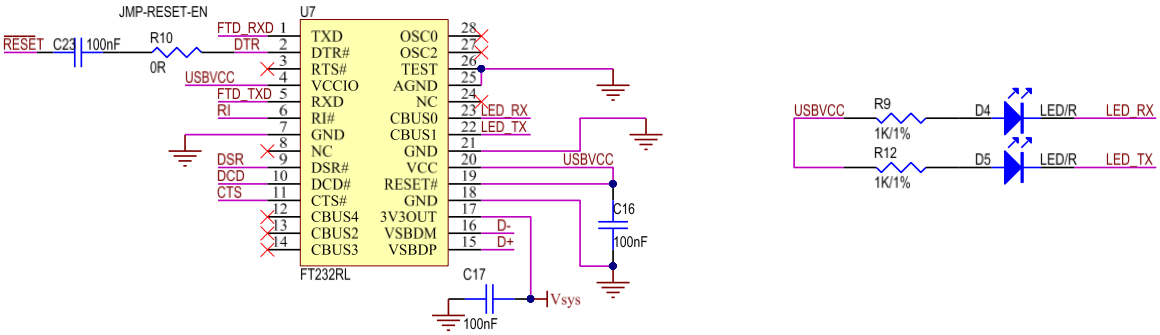


Figure 15 – FTDI232R schematic connection.

Like the USB converter, the WiFi module communicates with the microcontroller through the UART channel, however they are independent each other. The WiFi module is connected to the UART1 channel by pins WiFi_RX and WiFi_TX, and no flux control was implemented. The LED connections allow a visual feedback of the module status - STATUS, data transfer – DATA_TRAN – and module connection – CONNEC. The jumper JP5 and the pull up resistor R25, enable the system for work in ad hoc mode immediately after being powered. Figure 16 shows the RN-131 connections.

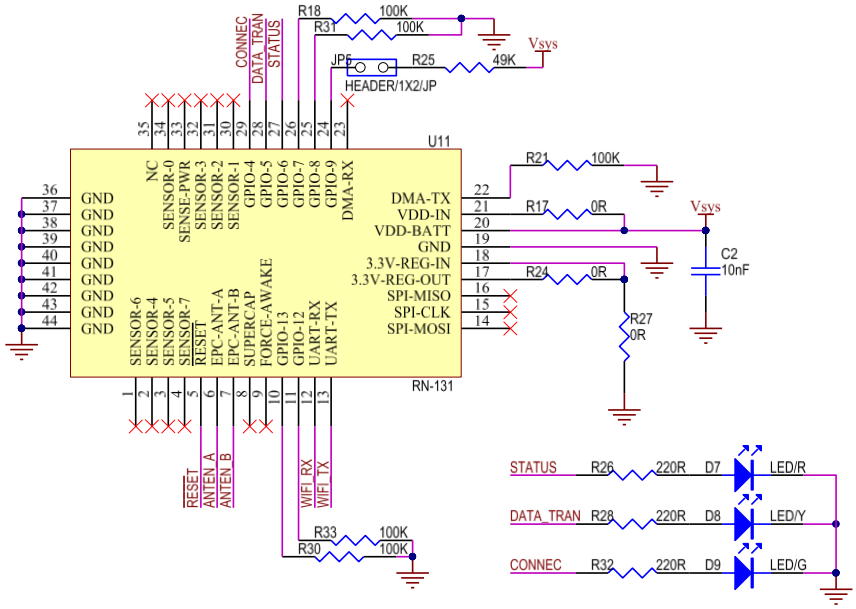


Figure 16 – RN-131 schematic connection.

The maximum transfer data rate over the WiFi connection for this module is 54 Mbps, however the UART connection presents an lower rate. The maximum baud rate described on the component datasheet was 1 Mbps for UART connection. On the other hand, the user manual presents different values were the maximum baud rate is 921.600 bps. That sort of value cannot be configured in the microcontroller, been impossible to reach the fastest baud rate on both devices. Due to that, the maximum rate reached was 115200 bps, remaining implemented.

To connect the module wireless, an on-chip antenna is integrated on the WiFi module. However, it is possible to configure the module to use external antenna through the U.FL connector allowing optimize system range. The on-chip antenna presents a gain of 2.0 dBi with maximum peak power of 237.7mW. The module is shown in Figure 17.

Figure 17 – RN-131 module with integrated on-chip antenna.[32]

3.6. MICRO SD CARD - MEMORY STORAGE

To store the data, a Micro SD card support was implemented using the Serial Peripheral Interface (SPI) peripheral available in the microcontroller. The SPI allows full-duplex communication between devices up to 4 MHz with 16 MHz main clock. The Micro SD card is connected to the microcontroller using four control pins plus two power pins. The controls pins are described in Table 8.

Table 8 – SPI connections describe.

Acronym	Description	Function
MOSI	Master Output, Slave Input	Send data from master to slave device
MISO	Master Input, Slave Output	Master receives data from slave device.
SCK	SPI Clock	Synchronization clock
CS	Chip Select	Select the current chip to receive/transmit data

As hardware implemented peripheral, the SPI protocol is controlled internally by the microcontroller and the peripherals registers and flags providing the information about the state of communication. To control the micro SD card using SPI interface, the micro SD card must be a slave and the microcontroller must be a master device. That configuration allows the microcontroller routines control the read/write process. Due to these requirements, MOSI, SCK and CS pins are output pins and MISO is input pin from the microcontroller view. The micro SD card socket connection is presented in Figure 18.

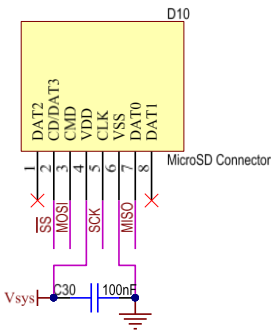


Figure 18 – Micro SD socket connections.

3.7. TIME CONSTRAINTS

Due to the tasks performed during the conversion and processing data, the time constraints is the result of the peripheral's clock frequency and/or the code program implemented.

The analog to digital conversion routine converts 24 force sensors (N_{Force}) plus 5 IMU (N_{IMU}) signals. Furthermore, force sensors conversion routine uses an average filter (AVG) to smooth the signal, filtering high oscillations. Due to the ADC characteristics[11] and clock frequency[12] (ADC_{CLK}), the minimum period for conversion (T_{Conv}) is 104µs, given by Equation 2.

$$T_{Conv} = 13 \times \frac{1}{ADC_{CLK}} \tag{2}$$

Thus, the total conversion time ($T_{TotalADC}$) is 21ms, considering times from force sensors, IMU, and filtering process, given by Equation 3.

$$\begin{gathered} T_{Force} = T_{Conv} \times AVG \times N_{Force} \\ T_{IMU} = T_{Conv} \times N_{IMU} \\ T_{TotalADC} = T_{Force} + T_{IMU} \end{gathered} \tag{3}$$

Due to an UART baud rate ($BRate_{UART}$) of 115200 bps and the total number of sent bytes (N_{Bytes}) of 231 (see section 4.3.5 – Data Communication), the UART time constraint (T_{UART}) is approximately 16 ms, given by Equation 4.

[11] Due to successive approximation circuitry, a normal conversion requires 13 clock cycles to be done.

[12] The ADC clock frequency is defined during peripheral initialization process for 125 KHz.

$$T_{UART} = \frac{1}{BRate_{UART}} \times 8 \times N_{Bytes} \quad (4)$$

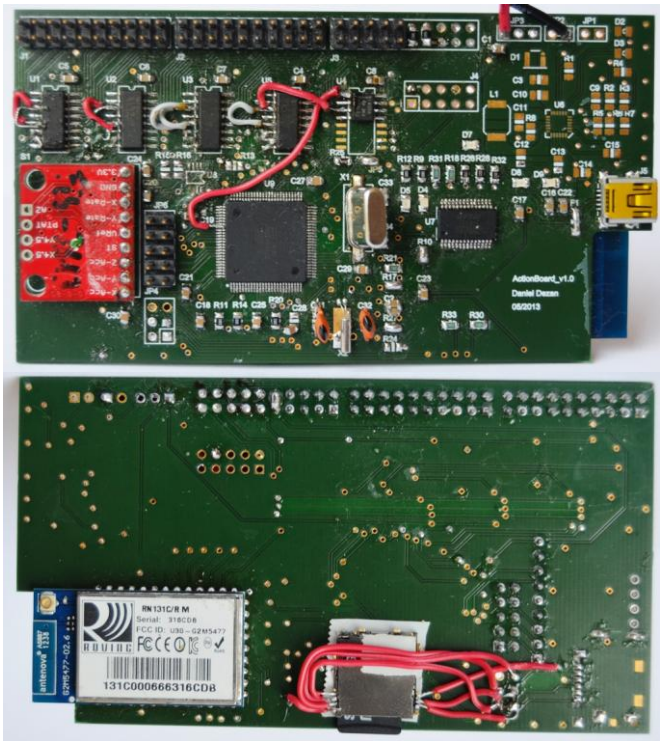
Apart from those constraints, the highest time constraint is given by the SD card and FAT32 routines. These routines take approximately 47.3 ms to perform a new data writing in the card. Although the number of bytes contributes to increase the time of data writing, the time limitation is given by the library implementation.

Due to the time constraints presented, the highest frequency that can be used avoiding data loss is 11.86 Hz. However, it is appropriate to use a frequency lower than that, so the acquisition frequency chosen was 10 Hz.

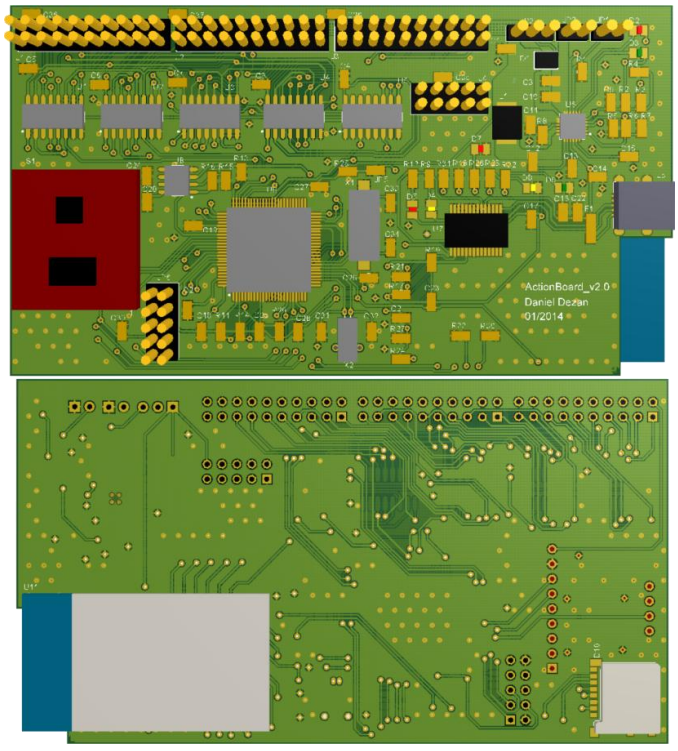
3.8. PRINTED CIRCUIT BOARD LAYOUT

A printed circuit board was developed from the schematic diagram presented in APPENDIX C, using the software Altium Design 13.1. The PCB prototype contains two layers to allowing laboratorial assembly and cost reduction. Surface mounting device (SMD) technology were used to optimize space usage, however the footprints selected should allow soldering without industrial machines which was a restriction to consider. Most of components were positioned on the top layer, although there are two components on the bottom layer. The total size of the PCB is 111x62mm and the layers are detailed is in APPENDIX E.

The Figure 19a shows the PCB produced using the first design version. On the other hand, The Figure 19b shows the 3D model of the second version of the PCB which contains an improvement in multiplexers positioning and in the micro SD Card footprint.



(a) – PCB produced using the first design.



(b) – 3D model of the second version.

Figure 19 – Printed circuit board.

4. FIRMWARE

Due to the interdependence between hardware and firmware, both are concurrently developed, however, general tests and validation procedures only can be done when the development process is concluded. Especially in firmware applications, the features are fully hardware dependent due to the low level of the implementation.

Based on the V-model development process[13], the firmware development can be described by using five major steps: requirements definition, design, code implementation, hardware integration and validation, as depicted in Figure 20.

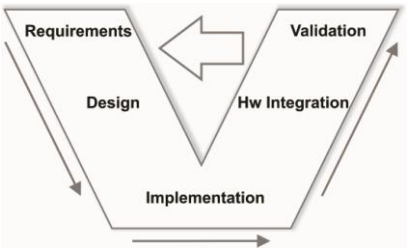


Figure 20 – Firmware development based on V-Model process.

[13] V-Model development process is one methodology for software/hardware development that demonstrates the relation between the phases of the development.

Based on the V-Model process presented the steps are described in the sequence.

4.1. REQUIREMENTS DEFINITION

The firmware is responsible for implementing a routine that controls the microcontroller operation in order to achieve the project goals and its requirements can be seen as tasks to do it. Therefore, the tasks to be performed in the firmware are:

- Controlling the multiplexer cascade circuit;
- Acquiring conditioning circuit and IMU voltages;
- Storing the data from conversions and calculations;
- Transmitting data trough wireless and USB to a remote computer;
- Receiving wireless and USB data from a remote computer;

Based on these requirements, the next steps define the general firmware structure.

4.2. DESIGN

The firmware design is described here by using flow diagram. The Figure 21 shows the general functionality of the firmware and describes the main flow.

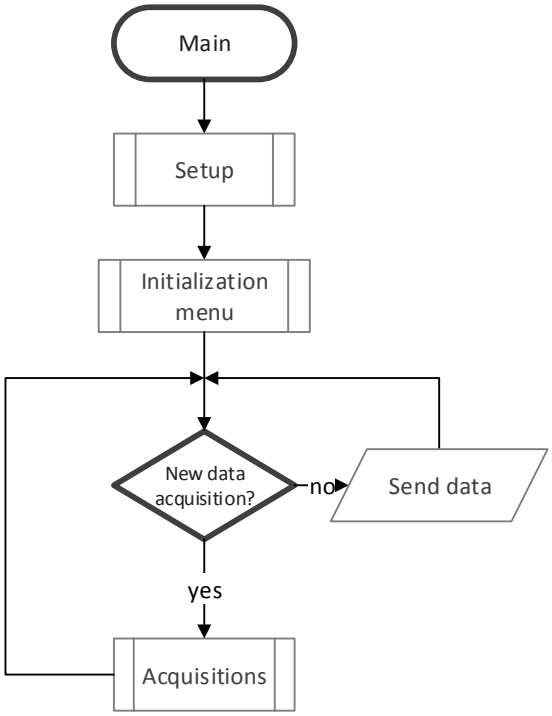


Figure 21– General firmware functionality.

Primarily, the system initialization routine sets up the peripherals and variables to allow proper operation. This setup is called only once during power-on. After the setup, the system enters in an initialization menu where the user can adjust a few parameters such as, date and time, e.g., and start the data acquisition. Then, the system enters in an infinity loop where the data acquisition is performed and the data frame is sent over WiFi.

Next topics will describe the functionality of each stage in the main flow.

4.2.1. SETUP

The setup routine initializes the system by setting the microcontroller's register properly according to the hardware architecture and system requirements. The routine must initialize the peripherals and some functions like multiplexer control. Figure 22 shows the sequence of initialization.

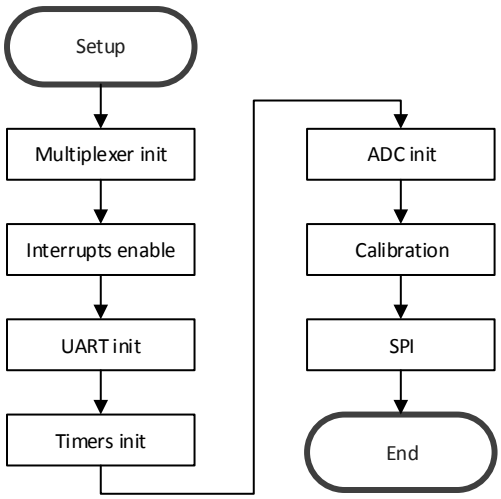


Figure 22 – Setup sequence for system initialization.

At first, the multiplexer is initialized by assigning proper I/O pins direction and signals state as described in Table 9. This configuration ensures that the multiplexers are disabled.

After that, the global interrupt enable command is executed due to every interrupt vector are dependent of the global interrupt state. Once the global interrupt is enabled, every interrupt can be controlled individually by its own register.

Table 9 – Multiplexer pinout configuration.

Pin	Description	Configuration	State
PA0	channel selection A	output	Low
PA1	channel selection B	output	Low
PA2	channel selection C	output	Low
PA3	mux enable 1	output	High
PA4	mux enable 2	output	High
PA5	mux enable 3	output	High
PA6	mux enable 4	output	High
PA7	mux enable 5	output	High
PD5	channel selection A_out	output	Low
PD6	channel selection B_out	output	Low
PD7	channel selection C_out	output	Low

The next setup step is the assignment of the UART peripheral registers. This configuration sets the serial baud rate, frame format and enable the transmitting/receiving data. Due to the usage of two UART channels, the configurations must be done for each one independently and can assume different values if it is necessary. Despite of being independent, the frame format used is the same and only the baud rate is change to allow faster communication in USB usage. Table 10 shows the register and the values assigned in both channels.

Table 10 – UART 0 and UART 1 registers configuration.

Register[(1)]	Description	Configuration	UART 0	UART 1
UBRRnH UBRRnL	Baud rate register High/Low defines the baud rate		1 Mpbs	115200 bps
UCSRnB	Control and status register to enable/disable Tx and Rx operation	TxEn RxEn	Enable	
UCSRnC	Control and status register to define frame format	Data size	8	8
		Stop bit	1	1
		Parity	ODD	

[(1)] n represents the UART channels

After UART setup, the Timers are initialized as well. Both Timer 1 and Timer 2 are used for control acquisition frequency and RTC, respectively. First of all, due to the time constraints presented, the Timer 1 is configured to tick new match every 0.1 s providing

the frequency of 10 Hz needed for sensors data acquisition. On the other hand, Timer 2 is configured to tick in counter overflow providing a period of 7.8 ms due to the 8-bit resolution and the frequency operation of 32.678 Hz provided by the external oscillator. The Table 11 shows the register configuration for both timers.

Table 11 – Timers 1 and Timer 2 registers configuration.

Timer 1			
Register	**Description**	**Configuration**	**State**
TCCR1B	Defines prescaler	CS10:2	÷ 64
TCCR1B	Sets mode of operation	WGM13:0	CTC[2]
TCCR1A	Sets normal mode of operation for pins output OC0A and OC0B	COM1A1:0 COM1B1:0	0
OCR1AH OCR1AL	Sets the compare value for timer counting compare		25000
TIMSK1	Enables timer output compares interrupt enable	OCIE1A	High

Timer 2			
Register	**Description**	**Configuration**	**State**
ASSR	Sets Asynchronous mode for external clock signal	EXCLK	Low
	Sets Asynchronous mode for external 32 KHz crystal	AS2	High
TCCR2B	Defines prescaler	CS20:2	No prescaling
TCCR2A	Sets mode of operation	WGM23:0	Normal[3]
TCCR2A	Sets normal mode of operation for pins output OC0A and OC0B	COM2A1:0 COM2B1:0	0
TIMSK2	Enables interrupt overflow	TOIE2	High

[1] n represents the timer number.
[2] Clear timer on compare match.
[3] Normal mode counts always up until overflow occurs.

Following this stage, the ADC is initialized setting parameters like prescaler, voltage reference and resolution, for instance. Due to the time constraints, the ADC is configured as shown in Table 12.

Table 12 – ADC registers configuration.

Register	Description	Configuration	State
ADCSRA	Turn on the ADC peripheral	ADEN	High
	Defines prescaler	ADPS2:0	÷ 128
	Enables interrupt operation after conversion	ADIE	High
	Start first conversion	ADSC	High
DIDR0 DIDR2	Digital interrupt registers		Disable
ADMUX	Voltage reference selection	REFS0:1	AVCC

The next step acquires data from the force sensors to be used as voltage offset calibration. Due to the matrix configuration and the multiplexer sweep, the no load sensor's voltage can vary between each other which makes necessary to set the offset that will be used as a reference.

Finally, the setup is finished by initializing the SPI by setting the mode of operation, frame direction, e.g., and enables the peripheral operation as shown in Table 13.

Table 13 – SPI registers configuration.

Register	Description	Configuration	State
SPCR	Enables SPI peripheral	SPE	High
	Selects master operation for microcontroller	MSTR	High
	Selects the initial clock rate	SPR1:0	÷ 64
SPSR	Defines double speed	SPI2X	Low

4.2.2. INITIALIZATION MENU

After setup the parameters, the system goes into in a menu mode where the user can select Debug mode to configure date and time on the RTC and manage the files in the micro SD Card. The user can also select the Acquisition mode where the system will start to run effectively acquiring data. To illustrate the mode menu, Figure 23 shows a state diagram.

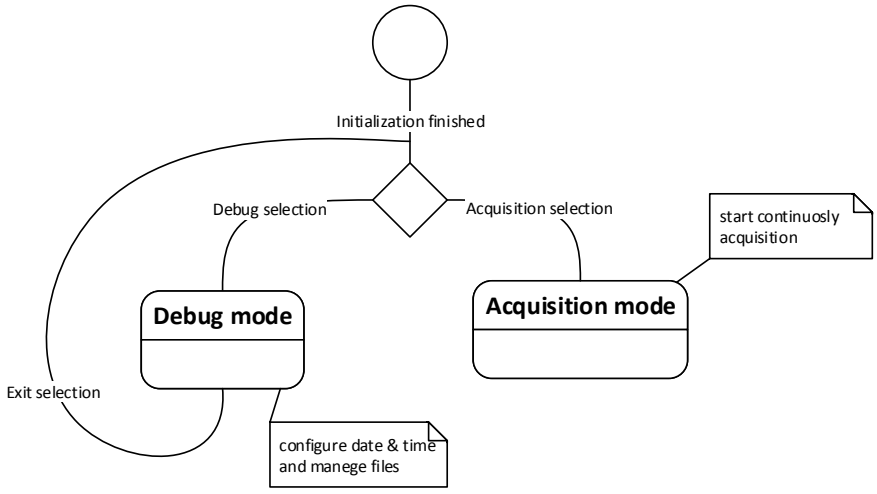


Figure 23 – Modes of operation from main menu.

Once Debug mode is selected, a new menu shows that mode options where the user can see the actual data and time apart from updating it. The user also can verify the list of files recorded into de micro SD Card. Finally, the file content can be read or deleted from the memory. On the other hand, once Acquisition mode is selected, the system will start to run continuously acquiring data from the sensors and converting it into the desirable variables.

4.2.3. MAIN OPERATION

When the Acquisition mode is selected, the system goes into a loop where the flag of Timer 1 tick is continuously verified to perform a new acquisition. Once the flag is activated the firmware executes four different functions where, at first, data from force sensors and IMU is acquired, then, it is processed to provide information about pitch and roll and CoP. Those functions will be described on the following topics.

Data stored into the variables is fitted into a data frame which will be sent for the micro SD card and WiFi. This frame generation updates the timestamp to ensure that its information is correct and equivalent to the current time. Finally, the data frame is sent by WiFi with the same format as was stored in the memory.

4.3. IMPLEMENTATION

The firmware was developed using C programming language in the Atmel Studio 6.1 and compiled using the native AVR toolchain available within the program. The binary code

was downloaded to the microcontroller using the AVR Dragon™ programmer from Atmel via JTAG which also allows the debugging process.

The code was divided into different files grouping the function according to the functionality. Next, will be described the main code programming contents into those files.

4.3.1. MAIN

The main function controls the system setup and operational modes. First of all, a setup function is called to configure peripherals and initialize variables. The peripherals are initialized in this function to ensure that all of them will start working before the main code starts to run, then Timers, UARTs, ADC, SPI and I/O pins are initialized according to system requirements. The multiplexer and IMU filter variables are also initialized. Furthermore, a force sensors calibration is performed.

4.3.1.1. MICRO SD CARD INITIALIZATION

After setup, an infinite loop contains the code to run the main routine. Due to the use of a SD card as a storage memory, the function starts initializing it, providing initialization messages to user and, if the SD card is detected and the initialization had been successful, the program will still be running. Otherwise, if some problem is detected, the user receives an error message and the system will stop running. The SD card initialization procedure is shown in Figure 24.

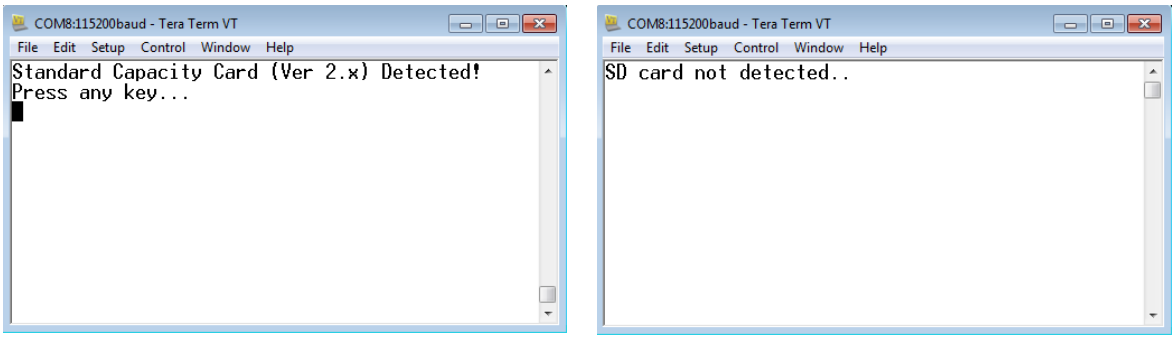


Figure 24 – SD card initialization messages.

4.3.1.2. OPERATION MODE SELECTION

Afterwards, there are two modes that control system operation, which are Debug and Acquisition modes. Using a menu that is accessed through a communication terminal, the user selects in which mode the system is going to operate by selecting option 1 to Debug mode and option 2 to Acquisition mode as depicted in Figure 25. If an invalid option is selected, the user receives an invalid option message and the menu is refreshed enabling a new selection.

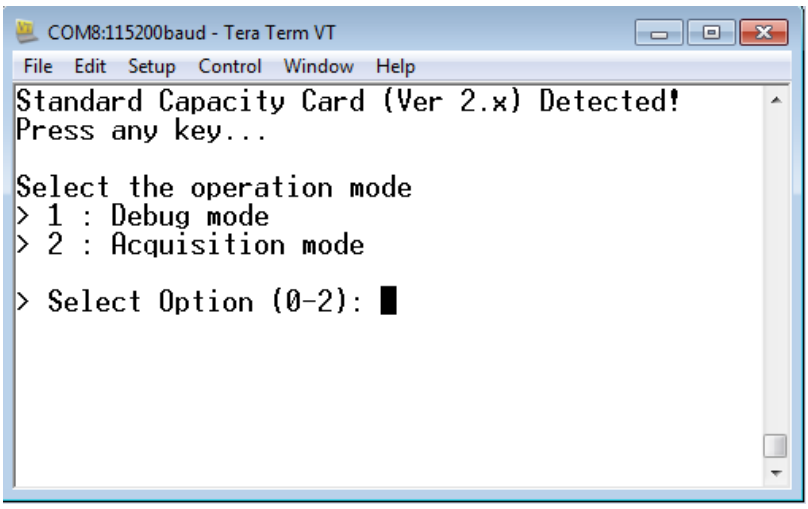


Figure 25 – Mode menu selection.

Once Debug mode is selected a new menu shows the mode options which the user can select to configure system parameters, Figure 26. The option zero exits from the menu and back to the menu mode selection. Option 1 shows the current date and time, meanwhile options 2 and 3 update its values. Furthermore, options 4 to 6 manage SD card file.

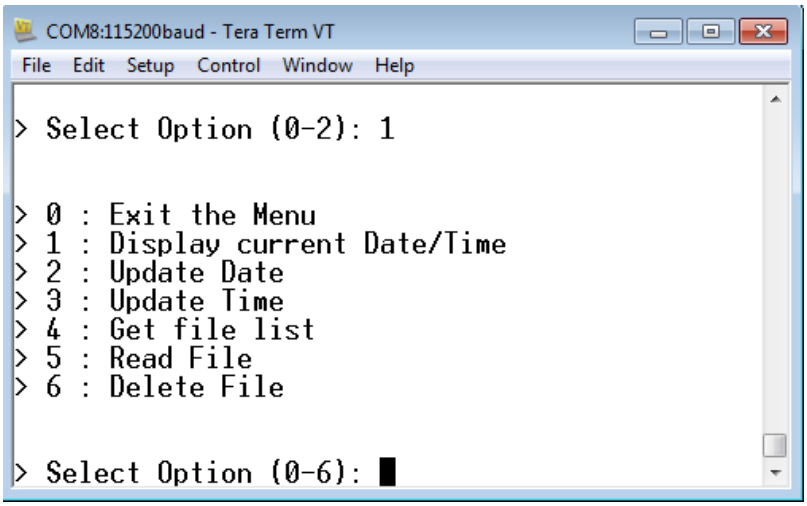


Figure 26 – Debug menu options.

To update date and time, functions *RTC_updateTime* and *RTC_updateDate* are called. These functions read the actual date and time and update it according to the data typed by user. There is also a type verification to certify that wrong characters are not inserted. Those routines are implemented in *RTC_routines* file. On the other hand, to manage SD card files, functions *findFiles*, *readFiles* and *deleteFiles* are used from *FAT32* file.

Once Acquisition mode is selected, the system starts to run and process data from sensors. To control the acquisition frequency, a flag is used from Timer 1. When the flag signals a new acquisition, functions that get data from sensors are called. Those functions and flag will be explained later. After that, the SD card's file name is created and data converted is stored in *dataString* array in ASCII format. Also the data is sent by WiFi using function *sendWiFiData* and stored in SD card using function *writeFile*.

4.3.2. INTERRUPTION

There are seven interrupt vectors implemented on this project firmware which controls ADC, Timers 1 and 2, and UART0 and UART1 peripherals.

The ADC vector interrupt is executed always when an ADC conversion ends. The interrupt routine reads the conversion result storage into two 8 bit registers – ADCL and ADCH – and converts them into a 16 bit integer variable named *iADCResult*. The interrupt routine returns when the ADCH register is read. Due to the 10 bit ADC resolution, the highest 6

bit of *iADCResult* variable has no affect on the result. The conversion code sequence is shown in Figure 27.

```
40  //ADC interrupt vector
41 ISR(ADC_vect)
42  {
43      uint8_t uiADCLow = ADCL;
44      iADCResult = ADCH<<2 | uiADCLow>>6; //10bits
45  }
```

Figure 27 – ADC read conversion registers sequence.

To control sensor acquisition frequency, a Timer 1 interrupt vector gives a precise period. The Timer 1 initialization selects prescaler of 64 and enables a Clear Time on Compare Match operation mode. The comparison value gives 10 Hz frequency operation for Timer 1 interrupt. In this mode, the interrupt occurs when the comparison between timer count and OCR1AH/OCR1AL registers matches. Then, the timer counter is cleared and then counting starts again. Inside the interrupt function there is a Boolean Flag *tmr1_flag* that signals a new match. Figure 28 shows Timer 1 interrupt routine.

```
54  //TMR1 match A interrupt vector
55 ISR(TIMER1_COMPA_vect)
56  {
57      tmr1_flag = !tmr1_flag; //timer 1 flag
58  }
```

Figure 28 – Timer 1 interrupt routine.

Another timer interrupt is implemented on Timer 2. This interrupt allows asynchronous mode operation which is associated with the RTC microcontroller feature. The Timer 2 is an 8 bit timer/counter that is configured to generate an interrupt when a counter overflow occurs and no prescaler is used providing 128 counts per second. The flowchart is shown in APPENDIX F. When an interrupt occurs in Timer 2 vector, the global RTC variables are updated, controlling the RTC operation. Furthermore, the function monitors the occurrence of leap years. Figure 29 shows a code extract for time update.

```
61  //TMR2 overflow interrupt vector
62  ISR(TIMER2_OVF_vect)
63  {
64  ...
74  /**
75  *///flag to verify that time/date is not been updated
76      if (!UpDateTimeAndDate)
77      {
78          if (ucMili == 128)
79          {
80              ucSec += 1; //update seconds
81              ucMili = 0;
82
83              if (ucSec == 60)
84              {
85                  ucMin += 1; //update minute
86                  ucSec    = 0;
87
88                  if (ucMin == 60)
89                  {
90                      ucHou += 1; //update hour
91                      ucMin    = 0;
92
93                      if (ucHou == 24)
94                      {
95                          ucHou    = 0;
96                          ucDay += 1; //update day
97                      }
98                  }
99              }
100         }
101         else
102             ucMili += 1; //update milliseconds
103 ...
162 }
```

Figure 29 – Timer 2 counter interrupt routine for RTC control.

To control data communication, an UART vector interrupt is used. Due to two UART channels implemented – UART0 and UART1, on this firmware description will be used only UART to represent both channels. There are two interrupt vectors for UART communication controlling receive and transmit separately. The data received/transmitted by UART channels are stored into UDR register which is a 16 bit where the upper byte is the receive byte (RXB) and the lower byte is the transmit byte (TXB).

The interrupt for UART receive occurs when the register UDR receives a new data in RXB. Due to a FIFO method used in communication process, the byte received is stored into the final position of local FIFO buffer – *bufferRx*. Then, the buffer pointers are updated and the interrupt ends. Figure 30 shows the UART interrupt code.

```
220  /* Receive USART 1 interrupt */
221 ISR(USART1_RX_vect)
222 {
223     BufferUSART1.bufferRx[BufferUSART1.endRxF] = UDR1;  //insert data in the buffer
224
225     if (BufferUSART1.endRxF < (BUFTX_LEN - 2))  //update buffer address
226     {
227         BufferUSART1.endRxF++;
228     }
229     else
230     {
231         BufferUSART1.endRxF = 0;
232     }
233     //when an overflow occurs, overwrite the oldest byte
234     if (BufferUSART1.endRxF == BufferUSART1.endRxI)
235     {
236         if (BufferUSART1.endRxI < (BUFRX_LEN - 2))
237         {
238             BufferUSART1.endRxI++;
239         }
240         else
241         {
242             BufferUSART1.endRxI = 0;
243         }
244     }
245 }
```

Figure 30 – UART receive interrupt code routine.

Likewise, the interrupt vector that controls UART transmit process uses a FIFO buffer – *bufferTx* – to provide the data to be transmitted. However, this interrupt only occurs when a new data is inserted into the FIFO buffer and the UDRIE bit is set in the UCSRB register enabling the interrupt. The interrupt is disabled when the FIFO buffer is empty. Figure 31 shows interrupt transmit code.

```
206  /* Trasmist USART 1 interrupt */
207 ISR(USART1_UDRE_vect)
208 {
209     if (BufferUSART1.endTxF != BufferUSART1.endTxI) { //verify new buffer data
210         UDR1 = (BufferUSART1.bufferTx[BufferUSART1.endTxI]); //send buffer data
211         while (!TXC1); //waits transmit flag indicate end of transmition
212         if (BufferUSART1.endTxI < (BUFTX_LEN-2)) { //update buffer address
213             BufferUSART1.endTxI++;
214         } else
215         BufferUSART1.endTxI = 0;
216     }else
217     UCSR1B &= ~(1<<UDRIE1); //disable interrupt
218 }
```

Figure 31 – UART transmit code routine.

4.3.3. SENSORS ACQUISITION

The function *initADC* initialize ADC peripheral feature, setting all parameters required for start acquisition properly as described in the firmware design topic. Furthermore, this function initializes the variables that describe force sensor position in the matrix.

The conversion result is configured as left adjusted which means that the 8 most significant bit are stored in ADCH meanwhile last 2 bit are stored in ADCL as show Figure 32.

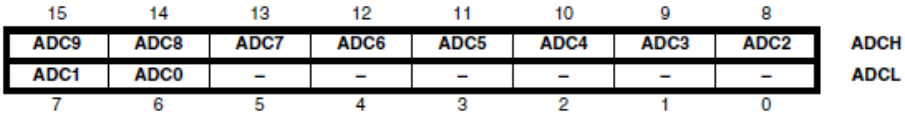


Figure 32 – ADC store result sequence.

Considering that the applied signals in the ADC are always positive and varies from GND to Vsys, the voltage reference is configured to Vsys using an external capacitor connected to AREF pin.

The acquisition routines include control over the multiplexer circuit to select the sensor according to the sweeping sequence defined. The multiplexer circuit selects one force sensor at a time, which is connected to the ADC through the channel 14. However, there are five remaining signals that are connected to the ADC which represents the IMU signals, from channels 9 to 13. The result of the force sensors conversion is a double type variable, meanwhile IMU result is an unsigned integer type. The variables are named as[14]:

- *dSensorValue[N_SENSORS];*
- *dResultantFrontX, dResultantFrontY, dResultantFront;*
- *dResultantRearX, dResultantRearY, dResultantRear;*
- *dResultantTotalX, dResultantTotalY, dResultantTotal;*
- *uiAccelXYZGiroXY[N_ACCEL].*

[14] where, N_SENSORS and N_ACCEL are the number of 24 force sensors and 5 IMU signals, respectively.

4.3.3.1. Selecting ADC channel

To select the ADC channel, a function named *ADCSelectChannel* receives the variable *uiChannel* as parameter. Then, the variable is used to assign the value into the register ADMUX. Since the channels are ADC9 to ADC14, only those 3 bits have to be changed[15]. Figure 33 shows the code to select ADC channel.

```
156 void ADCSelectChannel(uint8_t uiChannel)
157 {
158     //ADC input channel config (mux5:0 = 100110)
159     /*uiChannel &= 0x07;*/
160     //uint8_t uiChannelLow = uiChannel & 0x07;
161
162     char a = uiChannel&1;
163     char b = uiChannel&2;
164     char c = uiChannel&4;
165
166     //char channelHigh = (uiChannel & 0x20)>>5;
167     ADCSRB |= (1<<MUX5);
168
169     if(a) ADMUX |= 1<<MUX0; else ADMUX &= ~(1<<MUX0);
170     if(b) ADMUX |= 1<<MUX1; else ADMUX &= ~(1<<MUX1);
171     if(c) ADMUX |= 1<<MUX2; else ADMUX &= ~(1<<MUX2);
172
173     //ADMUX |= uiChannelLow;
174 }
```

Figure 33 – ADC channel selection.

4.3.3.2. Selecting multiplexer channel

To select the multiplexer channel, a function *selectMuxChannel* receives two parameters. First parameter is the channel number and second is a command to enable or disable the corresponding channel. Due to the 24 inputs the sweeping sequence uses three selection signals[16] which are SEL_A, SEL_B, and SEL_C for the first part and SEL_A_OUT, SEL_B_OUT, and SEL_C_OUT for the second part of the cascade circuit. By controlling the time constraints, the channel selection and multiplexer enable signals are synchronized to ensure proper signal stabilization. Figure 34 shows a part of the code that selects the first eight channels from the multiplexers.

---

[15] The complete input sequence selection to assign correctly ADC channel using bits MUX0 to MUX5 can be consulted on component datasheet on table 26-4.

[16] Due to a binary system, the number of bits to select 8 inputs is given by 2^n where n is the number of bits. For this case: $2^3 = 8$.

```
65      {
66              //Set selABC value
67              if(a) PORTA |= 1<<SEL_A; else PORTA &= ~(1<<SEL_A);
68              if(b) PORTA |= 1<<SEL_B; else PORTA &= ~(1<<SEL_B);
69              if(c) PORTA |= 1<<SEL_C; else PORTA &= ~(1<<SEL_C);
70              delay_us(10);//delay
71
72              if (cChannel<=7)
73              {
74                  //set selABC for mux 5 (input Y7)
75                  PORTD |= 1<<SEL_A_OUT;
76                  PORTD |= 1<<SEL_B_OUT;
77                  PORTD |= 1<<SEL_C_OUT;
78
79                  //delay to stabilize values on mux
80                  delay_us(1);//delay
81
82                  //disable mux output. active low
83                  PORTA |= (1<<EN_MUX_2);
84                  PORTA |= (1<<EN_MUX_3);
85                  delay_us(1);//delay
86
87                  //enable mux output. active low
88                  PORTA &= ~(1<<EN_MUX_1); //enable mux 1
89                  delay_us(1);//delay
90
91                  //enable 5th mux output. active low
92                  PORTA &= ~(1<<EN_MUX_5);
93
94                  //delay to stabilize signals
95                  delay_us(1);//delay
96              }
```

Figure 34 – Multiplexer input select code extract

4.3.3.3. Acquiring sensor's signal

To acquire data from force sensors and IMU, two different functions are implemented where *getForceSensors* gets data from force sensors and *getAccelGyro* gets data from IMU. The first one selects the ADC channel 14, converts the value, implements an average filter and converts the result into Newton[17], meanwhile, the second one sweeping between the ADC channels 9 to 13, converts the value and store it. Despite of being used in different manner, both functions present the same structure for ADC conversion and only the force sensors conversion will be explained in detail. The code flowchart is shown in Figure 35.

[17] Derived from International System Units, Newton is a unit to quantify force and can be represents either as N or $kg.m.s^{-2}$.

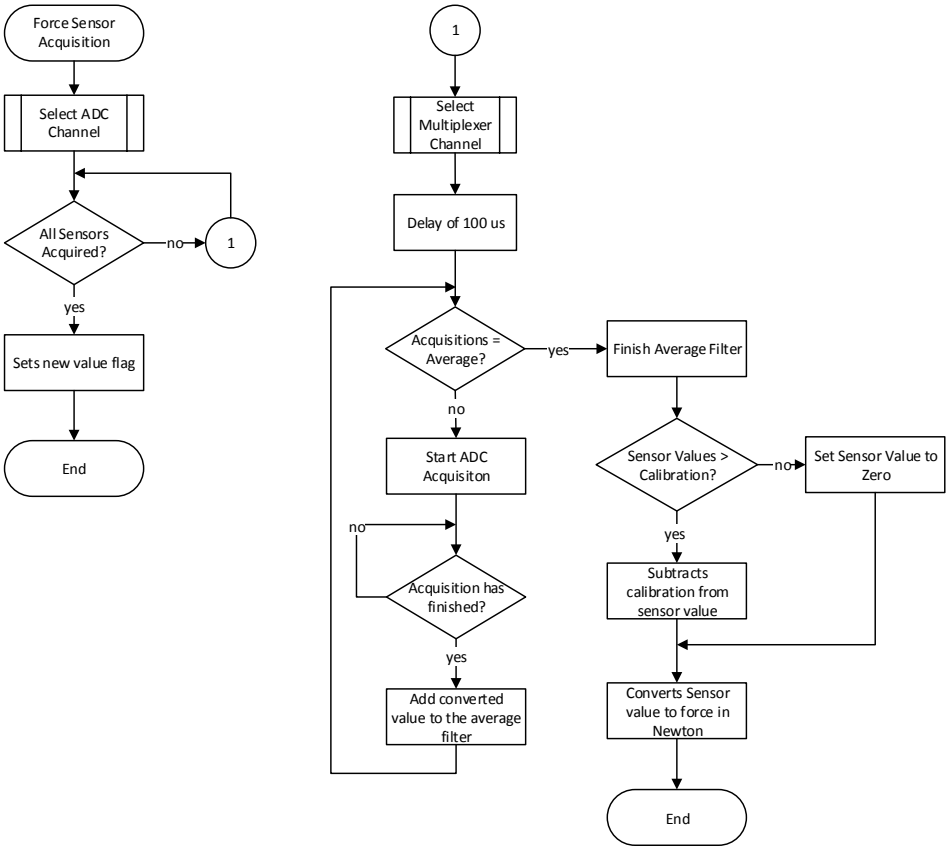


Figure 35 – ADC Force sensors acquisition flowchart.

This function starts selecting the ADC channel 14 and the conversion starts inside a *for loop* with the escape parameter N_SENSOR. Inside this loop, the function *selectMuxChannel* selects the multiplexer input channel[18] as shown in Figure 36.

```
197   void getForceSensors(void)
198   {
199   ...
211       ADCSelectChannel(6);                  //select channel 14 of adc converter
212       for (i=0;i<N_SENSORS;i++)             //multiplex N_SENSORS
213       {
214           //i=2;
215           selectMuxChannel(i,ENABLE);       //select ADC channel
216           delay_us(100);                    //Delay due to the multiplexer stabilization
217   ...
```

Figure 36 – Force sensor acquire extract function code.

[18] To avoid noise generated by switching, a delay of 100 µs is added in before the conversion starts.

After that, another *for loop* is executed to provide an average filter. The escape parameter is AVERAGE which is pre defined with value 4. Inside the loop, the ADSC bit is set high to start the next conversion and a *while loop* waits to the ADSC bit to be clear by hardware after conversion ends. At this moment, the new ADC data is available in the variable *iADCResult* and its value is added to *dSensorValue* array. After the average *for loop* ends, the sum of values stored in *dSensorValue* is divided by AVERAGE. Then, this result is compared with the respective pre stored offset. Figure 37 shows the acquisition and average extract code.

```
197 void getForceSensors(void)
198 {
199 ...
218         for (j=0;j<AVERAGE;j++)            //read ADC channel "AVERAGE" times to provide a average filter
219         {
220             ADCSRA |= 1<<ADSC;             //Start next conversion
221             while (ADCSRA&0x40);           //wait end of conversion
222             dSensorValue[i] += iADCResult;   //storage the sensor value adding the last value to do the average
223         }
224          dSensorValue[i] /= AVERAGE;
225
226         //update sensors value to zero if are residuals floating read
227         if (dSensorValue[i] > dOffSetCalibrate[i])
228             dSensorValue[i] -= dOffSetCalibrate[i];
229         else
230             dSensorValue[i] = 0;
231
232         dSensorValue[i] = (dSensorValue[i]*3.3)/1023; //convert adc value to voltage
233     }
234 ...
```

Figure 37 – ADC average calculation and off set adjustment.

After conversion and offset standardization, the values are converted to Newton assuming the linearity voltage versus force by using the equation from the calibration tests. An extract of the complete code found in APPENDIX G, is shown in Figure 38 for sensors 6 to 8.

```
197 void getForceSensors(void)
198 {
199 ...
235 ...
246     dForce[uipSensor[5]] = (2.5768*dSensorValue[uipSensor[5]]) + 0.2185; //sensor 6
247         if (dForce[uipSensor[5]] <= 0.2185) dForce[uipSensor[5]] = 0;
248     dForce[uipSensor[6]] = (2.9021*dSensorValue[uipSensor[6]]) + 0.3926; //sensor 7
249         if (dForce[uipSensor[6]] <= 0.3926) dForce[uipSensor[6]] = 0;
250     dForce[uipSensor[7]] = (2.6214*dSensorValue[uipSensor[7]]) + 0.3475; //sensor 8
251         if (dForce[uipSensor[7]] <= 0.3475) dForce[uipSensor[7]] = 0;
252 ...
286 }
```

Figure 38 – ADC result to force conversion.

On the IMU conversion, the conversion results are stored in the variable *uiAccelXYZGiroXY* as depicted in Figure 39.

```
217 void getAccelGyro(void)
218 {
219     uint8_t i;
220     for (i=1;i<=N_ACCEL;i++)        //multiplex N_SENSORS
221     {
222         ADCSelectChannel(i);        //select adc channel
223         ADCSRA |= 1<<ADSC;          //Start next conversion
224         while (ADCSRA&0x40);        //wait end of conversion
225         uiAccelXYZGiroXY[i] = iADCResult; //storage sensor's conversion
226     }
227     new_data_sensor = 1;
228 }
```

Figure 39 – IMU conversion code routine.

4.3.4. DATA STORAGE

All data processed from the force sensors, positions results and center of pressure are stored into a micro SD Card. To access the card, a SPI peripheral and a FAT32 file system library are used to allocate the data into a Windows known format.

First of all, the SPI is initialized during the initialization system into setup function. A function *spi_init* is called and the parameters are configured as shown in Table 14. Due to the SD card being a slave device, the microcontroller is configured as a master device to control SPI access setting high the MSTR bit. To configure SPI clock frequency, bit SPR0

and SPR1 select the main clock division factor and bit SPI2X enables double mode in master device. For initialization, the division factor is 64 and no double mode is used, which provides a SPI clock frequency of 250 kHz.

Table 14 – SPI initialization pinout assigned.

Pin	Description	Configuration
PB0	Chip select (SS)	output
PB1	Clock frequency (CLK)	output
PB2	Master Output / Slave Input (MOSI)	output
PB3	Master Input / Slave Output (MISO)	input

To read and write values through SPI peripheral, functions *SPI_receive* and *SPI_transmit* are implemented. The read/write register SPDR stores the content to be transmitted or received and it is single buffered in the transmit direction and double buffered in the receive direction.

After an SPI peripheral initialization, the SD card is also initialized, checking the file system and the storage capacity.

The files *SD_routines*, *FAT32* and *RTC_routines* are based on Dharmani libraries, which were adapted to this project context [33]. The functions into the *SD_routines* file controls the data transfer into microcontroller and SD card through the SPI bus. The functions into the *FAT32* file controls the file system read/write process. The functions into the *RTC_routines* file controls the RTC update and conversions.

4.3.5. DATA COMMUNICATION

The communication between the system and the external applications are done using two UART peripherals. Both of them are configured and can work independent of each other. The peripheral initialization function receives five parameters which configure the baud rate, double speed mode, data size, parity and number of stop bits as depicted in Figure 40.

```
/*
    Serial0 Init Function

*/
void initializeUART0 (long lBaud, char cAsyncDoubleSpeed, char DataSizeInBits,
                                        char cParityEVENorODD, char cStopBits)
{...
```

Figure 40 – UART initialization parameters.

The UART library has different functions to transmit data according to the data type required and two functions to receive data depending of the desired channel. The most important transmit function is called *serialIsrOutChar* which sends to FIFO TX buffer a char data type, updates the pointers and enables the transmitting interrupt. An extract of code for UART0 transmission is shown Figure 41.

```
void serialIsrOutChr(char data, char usart)
{
    if (usart == 0)
    {
        BufferUSART0.bufferTx[BufferUSART0.endTxF] = data;//insert new data into buffer

        if (BufferUSART0.endTxF < (BUFTX_LEN-2))    //update data buffer address
        {
            BufferUSART0.endTxF++;
        }
        else
        {
            BufferUSART0.endTxF = 0;
        }
        //when an overflow occurs, overwrite the oldest byte
        if (BufferUSART0.endTxF == BufferUSART0.endTxI)
        {
            if (BufferUSART0.endTxI < (BUFRX_LEN-2))
            {
                BufferUSART0.endTxI++;
            }
            else
            {
                BufferUSART0.endTxI = 0;
            }
        }

        UCSR0B |= 1<<UDRIE0;    //empty transmit buffer interrupt enable
```

Figure 41 – Transmit data through UART using FIFO buffer.

To process data received from receive interrupt, two functions named as *getSerialDataUSB* and *getSeralDataWiFi* return the data stored in the FIFO RX buffer from UART0 and

UART1, respectively. These functions verify if there is any remaining data to be read in buffer, returning it after updating the buffer's addresses. Figure 42 shows the code implemented.

```
403 char getSerialDataWIFI(void)
404 {
405     char cData = 0;
406 
407     if (BufferUSART1.endRxF != BufferUSART1.endRxI) { //checks if there is data on buffer
408         cData = (BufferUSART1.bufferRx[BufferUSART1.endRxI]); //read oldest data from buffer
409 
410         if (BufferUSART1.endRxI < (BUFTX_LEN-2)) {//update data buffer address
411             BufferUSART1.endRxI++;
412         } else
413             BufferUSART1.endRxI = 0;
414     }
415 
416     return cData; //return data read
417 }
```

Figure 42 – Serial WiFi data read implementation code.

Due to a demand for a standard data frame transmission, the function *sendData* creates a frame that is composed by # as a frame start character followed by data stored in *dataString* variable and a CR/LF command. The frame sequence is shown in Figure 43 and field description is show in APPENDIX H.

#	HH:MM:SS:MMM	24 SENSORS	PITCH	ROLL	ACCELX	ACCELY	ACCELZ	COPX	COPY	COP

Figure 43 – Data frame fields' sequence.

4.3.6. CENTER OF PRESSURE

The center of pressure calculation is based on the data from force sensors and the position of these sensors on the matrices (X_T, Y_T, CoP). Both matrices have the sensors equally spaced and the distance between them are fixed. The equation to calculate the CoP considers an individual force contribution (F_i) and distance from each sensor (X_i, Y_i) over the total resultant force (F_R). Take into consideration that the vector of force's direction is

normal to the matrix plane, Equation 5 provides the resultant force vector position (X_R, Y_R, F_R) on each platform.

$$\begin{cases} F_R = \sum F_i \\ X_R = \dfrac{\sum(F_i \times X_i)}{F_R} \\ Y_R = \dfrac{\sum(F_i \times Y_i)}{F_R} \end{cases} \quad (5)$$

However, to determine the body center of pressure a new calculation must be done considering those individual contributions. Equation 6 uses the values of X_R, Y_R and F_R from rear and front platforms.

$$\begin{cases} F_{Total} = F_{RFront} + F_{RRear} \\ X_{Total} = \dfrac{(F_{RFront} \times X_{RFront}) + (F_{RRear} \times X_{RRear})}{F_{Total}} \\ Y_{Total} = \dfrac{(F_{RFront} \times Y_{RFront}) + (F_{RRear} \times Y_{RRear})}{F_{Total}} \end{cases} \quad (6)$$

The code that implements equations system 5 and 6 is shown in APPENDIX I. A *for loop* is used to sweep all sensors. The array variable sensor provides the information about the position of each sensor.

4.3.7. ROTATION ESTIMATIVE

The surfboard's position is calculated based on the IMU data. The file *IMU_filter* contains the functions that are based on the Starlino Electronics and adapted for this project [34].

A function called *getEstimatedInclination* uses the data from ADC conversion to convert acceleration and gyro measurements into pitch and roll angles. The function starts converting data units [g] for acceleration, and [°/s] for gyro using sensitivity and offset

presented by manufacturers[19] as shown in Figure 44. Due to the IMU position being upside-down in relation to gravity force on the PCB, an inverter factor -1 is used on z axis. All accelerations are normalized to 1g.

```
105 void getEstimatedInclination(){
106     int w;
107     char signRzGyro;
108 ...
127     //convert to acceleration in g unit
128     dRxAcc = ((((double)(uiAccelXYZGiroXY[ACCX_AXIS])*3.3)/1023)-1.65)/0.33;
129     dRyAcc = ((((double)(uiAccelXYZGiroXY[ACCY_AXIS])*3.3)/1023)-1.65)/0.33;
130     dRzAcc = (-1)*(((((double)(uiAccelXYZGiroXY[ACCZ_AXIS])*3.3)/1023)-1.8)/0.33);
131     //convert to acceleration in º/s unit
132     dRxGyro = ((((double)(uiAccelXYZGiroXY[GYROX_AXIS])*3.3)/1023)-1.35)/0.002;
133     dRyGyro = ((((double)(uiAccelXYZGiroXY[GYROY_AXIS])*3.3)/1023)-1.35)/0.002);
134 ...
143     //normalize vector (convert to a vector with same direction and with length 1)
144     normalize3DVector(RwAcc);
145 ...
```

Figure 44 – Acceleration and gyro unit conversion.

The first estimation takes only the acceleration value as a reference and afterwards, every interaction uses the data from IMU. When previous estimated z angle is less than 0.1°, it means that there is no significant variation and the new gyroscope data is loaded with the previous estimated value. Otherwise, the estimation algorithm gets the angle projection on plane ZX and ZY based on last estimation values and then updates the angle using actual gyro data. Finally, the accelerometer acquired data and gyroscope calculated data are used to estimate pitch and roll. Figure 45 shows the principal code in the file to get angle estimation.

[19] The sensitivity for accelerometer is 330mV/g, meanwhile for gyroscope is 2mV/°/s

```
146 ...
149 if (firstSample){
150     for(w=0;w<=2;w++) RwEst[w] = RwAcc[w];    //initialize with accelerometer readings
151 }
152 else
153 {
154     //evaluate RwGyro vector
155     if(abs(RwEst[2]) < 0.1){
156         //Rz is too small and because it is used as reference for computing Axz, Ayz
157         //it's error fluctuations will amplify leading to bad results
158         //in this case skip the gyro data and just use previous estimate
159         for(w=0;w<=2;w++) RwGyro[w] = RwEst[w];
160     }
161     else
162     {
163     //get angles between projection of R on ZX/ZY plane and Z axis, based on last RwEst
164     for(w=0;w<=1;w++){
165         Awz[w] = atan2(RwEst[w],RwEst[2]) * 180 / M_PI; //get angle and convert to degrees
166     }
167
168     Awz[0] += (dRxGyro*Gyrodt);               //get updated angle according to gyro movement
169     Awz[1] += (dRyGyro*Gyrodt);               //get updated angle according to gyro movement
170
171     //estimate sign of RzGyro by looking in what quadrant the angle Axz is,
172     //RzGyro is positive if  Axz in range -90 ..90 => cos(Awz) >= 0
173     signRzGyro = ( cos(Awz[0] * M_PI / 180) >=0 ) ? 1 : -1;
174
175     //reverse calculation of RwGyro from Awz angles
176     for(w=0;w<=1;w++){
177         RwGyro[0] = sin(Awz[0] * M_PI / 180);
178         RwGyro[0] /= sqrt( 1 + squared(cos(Awz[0]*M_PI/180))*squared(tan(Awz[1]*M_PI/180)));
179         RwGyro[1] = sin(Awz[1] * M_PI / 180);
180         RwGyro[1] /= sqrt( 1 + squared(cos(Awz[1]*M_PI/180))*squared(tan(Awz[0]*M_PI/180)));
181     }
182     RwGyro[2] = signRzGyro * sqrt(1 - squared(RwGyro[0]) - squared(RwGyro[1]));
183 }
184
185 //combine Accelerometer and gyro readings
186 for(w=0;w<=2;w++) RwEst[w] = (RwAcc[w] + config.wGyro* RwGyro[w]) / (1 + config.wGyro);
187
188 ...
```

Figure 45 – Pitch and Roll estimation code.

4.3.8. MATLAB APPLICATION

A Matlab application was developed to allow real-time analysis and visualization during test procedure. The application opens a communication channel which allows reading the data frame and processing the data to show CoP and force sensors activation in a graphical interface. Furthermore, pitch and roll angles are displayed on command window. Figure 46 shows the graphical interface and pitch and roll angles during testing.

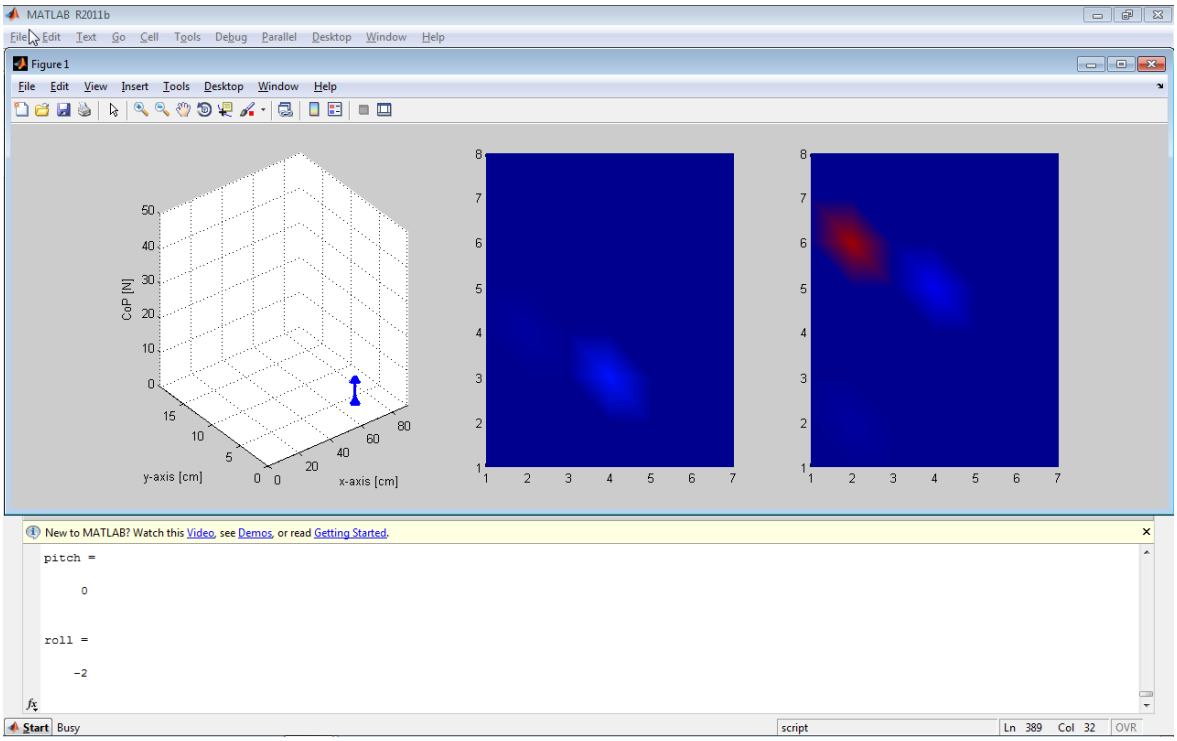


Figure 46 – Matlab application viewing CoP and position information.

After opening a communication channel, the data received is stored into a file in Matlab that is read afterwards and then separated and stored into local variables. The variables that correspond to the force sensors are stored into two matrices, which have the same distributions as that of the sensors in the surfboard. These matrices are plotted in 2D charts. The CoP data is plotted in 3D chart where X and Y axes represent the CoP position coordinate plane and Z axis represents the absolute force value. The corresponding code is shown in APPENDIX J.

5. TESTS

The tests performed aim to validate the system's hardware operation by analyzing the signals obtained and the system's operation when simulating basic movements present in surfing practice. The tests also aim to perform the last two steps presented in the V-Model for the firmware development which are: hardware integration and validation.

To verify the hardware integration, the electronic circuit was connected to a DSO-X 3012A InfiniiVision oscilloscope from Agilent Technologies to acquire the signals to be compared with predefined parameters such as delays and acquisition frequency, for example. Due to the importance for the data acquisition and the number of force sensors to read (24), the multiplexer signals are shown in detail; other features such as, data transmission and power consumption during transmission, for instance were also tested.

To simulate the movements from surfing activity, a test protocol was established using a regular footer surfer to perform three different setups over the surfboard, in which: Test 1 – Balance; Test 2 – Pitch rotation; and Test 3 – Roll rotation – plantar/dorsiflexion sideways displacement. Following the description in Chapter 2 of the surfing conditions and context, the surfboard can be considered an unstable platform where the surfer stands up over it controlling the balance to perform the manoeuvers as intended. Therefore, these conditions

were simulated by using a BOSU® Ball from BOSU® Fitness LLC, as depicted in Figure 47, which in turn, provides the necessary unstable support.

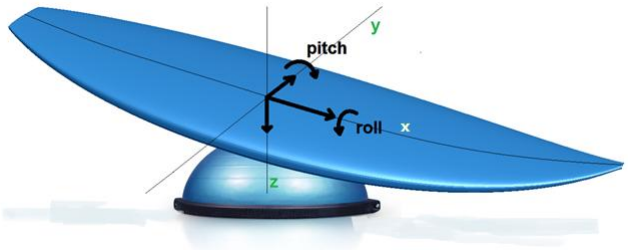


Figure 47 - Unstable setup to perform simulation tests and axis orientation.

For Test 1, the subject stays in balance keeping the surfboard as parallel as possible to the ground plane. For Test 2, the subject applies more weight on the back foot moving the surfboard's nose up until reaching - 30 degrees on pitch axis, keeping that position until the end of the trial. Finally, on Test 3 the subject keeps the surfboard parallel to the ground plane moving the surfboard from one side to the other along the roll axis, by controlling the plantar/dorsiflexion movement to reach ±15 degrees. Five 40s trials were performed at 10 Hz (or 400 samples). After acquisition, data was compiled and analyzed on Microsoft Excel®.

To allow a proper auto calibration routine the surfboard was unloaded and the subject waits for the command to start. On that moment, the surfboard stay over the BOSU® inclined about 17 degrees with surfboard's nose touching the ground. The calibration routine ends when the system starts to transmit data. After this step, a signal is given to the surfer and data acquisition starts.

5.1. HARDWARE INTEGRATION TESTS

To perform hardware tests, the setup depicted in Figure 48 was implemented. To ensure constant power supply during the tests, the system was powered with an external power supply model 1672 from BK Precision and the voltage and current verified by the Handheld Digital Multimeter U1251B from Agilent Technologies.

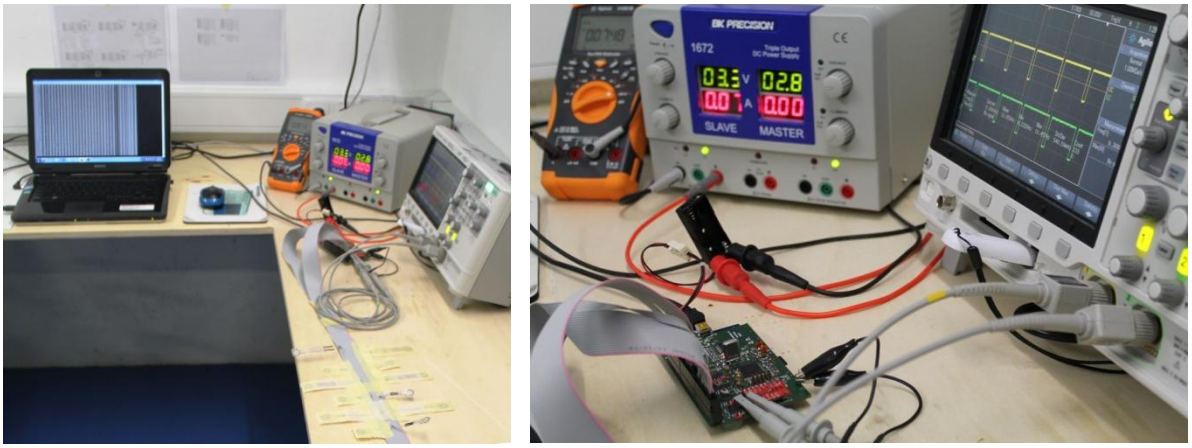


Figure 48 – Hardware test setup.

5.1.1. TIMING

The timing verification was divided into two tests, in which: ADC acquisition frequency; and multiplexer time constraints.

First of all, ADC acquisition frequency was verified by acquiring the enable signal from the multiplexer due to this signal being only activated when new data acquisition routine takes place. The multiplexer enable signal is active low. Figure 49 shows the oscilloscope screenshot with two enable signals, showing also the mean verified frequency of 10.052 Hz.

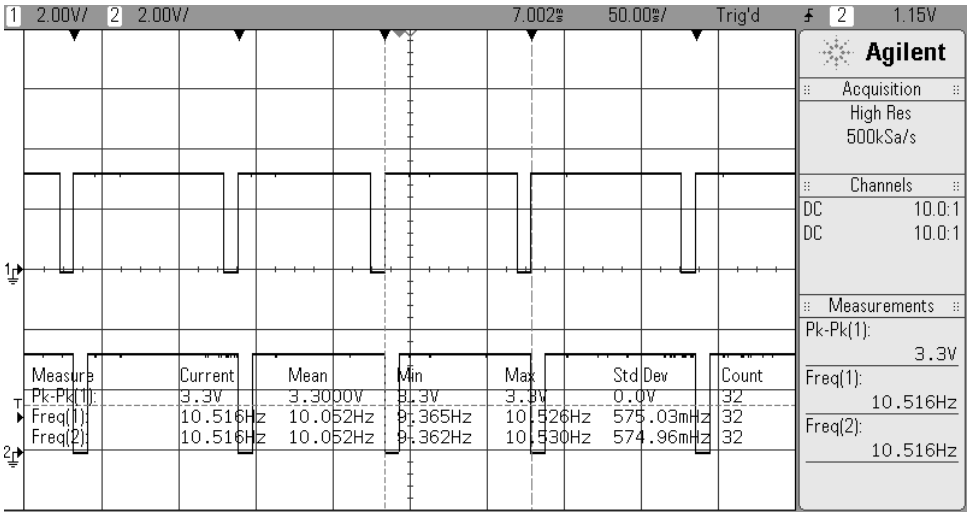


Figure 49 – ADC frequency verification

Thereafter, the tests for enable timing verification were performed by analyzing the time between enable/disable/enable instants[20]. The enable/disable analysis was performed acquiring data from multiplexers 1 and 2 during transition between multiplexer 1 enable and multiplexer 2 disable. The results have shown an average time interval of 9.935 μs which is depicted in Figure 50. The channel 1 on the oscilloscope represents the multiplexer 1 and channel 2 represents the multiplexer 2. Due to the duality between enable and disable time routines, the results for disable/enable are similar.

[20] On tests context is considered disable as one logic and enable as zero logic.

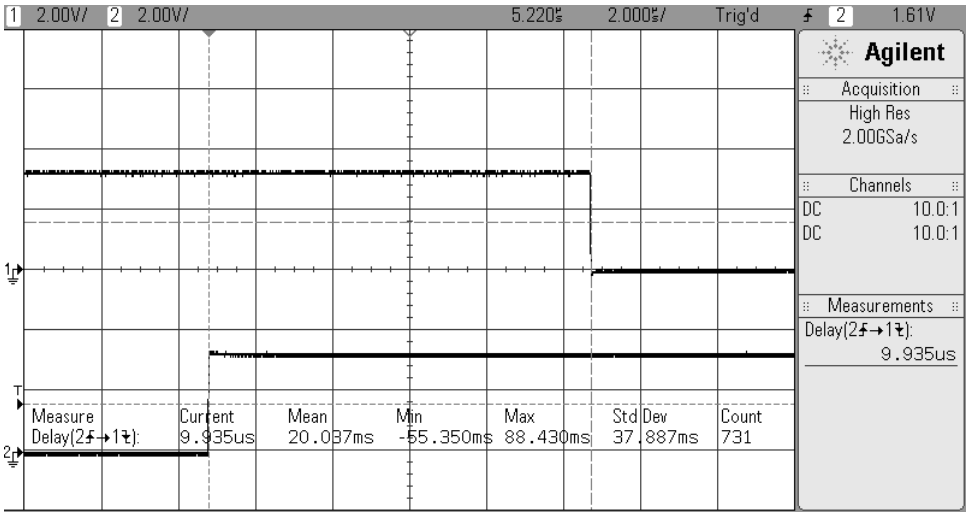


Figure 50 – Multiplexer enable timing verification.

5.1.2. MULTIPLEXER COUNTING SELECTION

The counting selection is responsible for selecting the multiplexer channel that will be read by the ADC in the microcontroller altogether with the enable signals. Due to the multiplexer cascade architecture the counting routine from signals SEL_A, SEL_B and SEL_C is performed three times for each ADC acquisition routine and is synchronized with the enable signals as shows in Figure 51.

Due to the use of 24 force sensors, the signal SEL_C_OUT has not been used, which explain its absence in the screenshot.

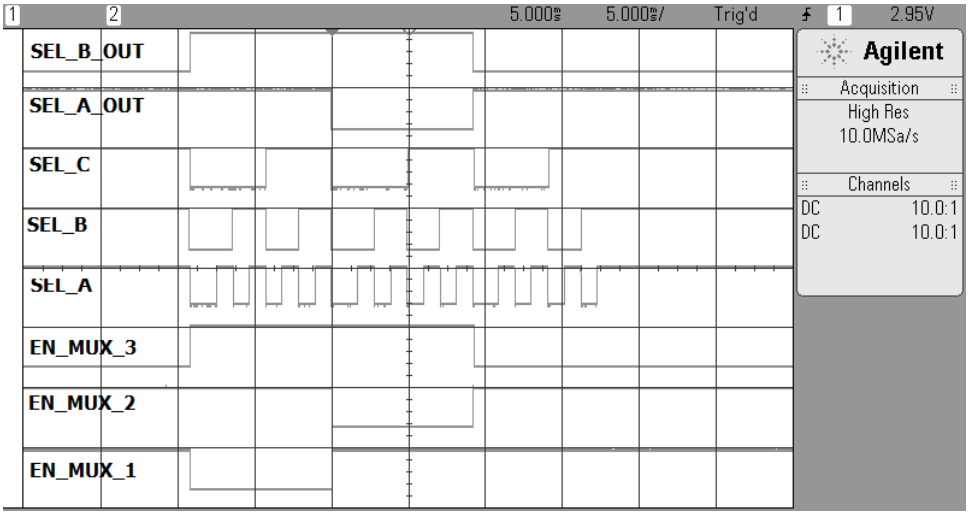


Figure 51 – Multiplexers counting channel selection.

5.1.3. SENSOR SIGNAL ACQUISITION

To verify the signals from sensor acquisition, the oscilloscope was connected to the conditioning circuit output to measure sensor's signal width and fall time. The sensor's signal width is defined as the time between changes in the level of the least significant bit (LSB) of the multiplexer select counter (SEL_A). On the other hand, the fall time is defined as the accommodation time[21] that occurs after changes in the LSB. To measure these parameters, one sensor was activated and the output is shown in Figure 52. The signal presented a width of 334.4 us and the rise and fall times of 3.19 us and 3.69 us, respectively.

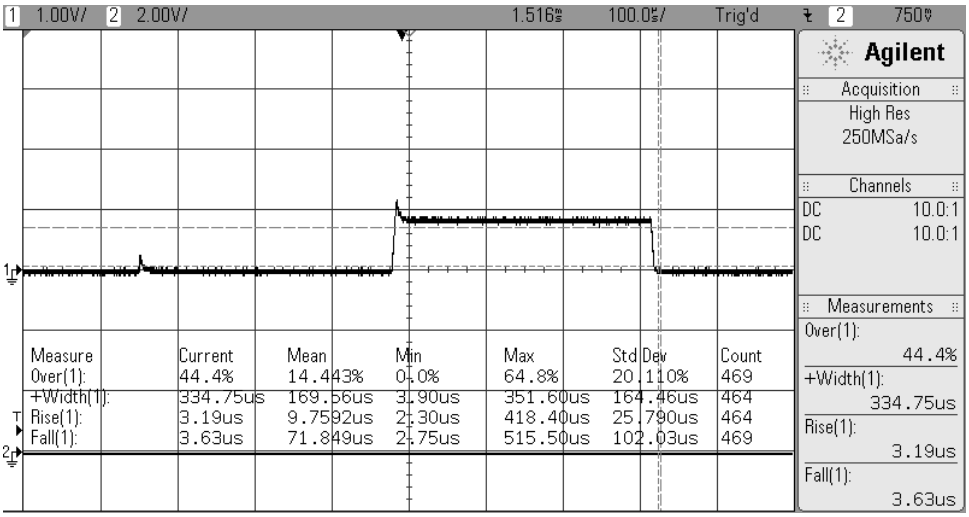


Figure 52 – Sensor signal width and fall times.

5.1.4. DATA TRANSFER DISTANCE

The data transfer distance was verified in open field by sending the data frame through WiFi for a computer, increase the distance until the signal gets lost. The test was performed three times where the position of the WiFi on-chip antenna in relation to the observer on the computer was changed as depicted in Figure 53. Due to the antenna specification the longest distance must be achieved on plane ZY [35].

[21] The accommodation time is considered in between 90% to 10% of the signal voltage.

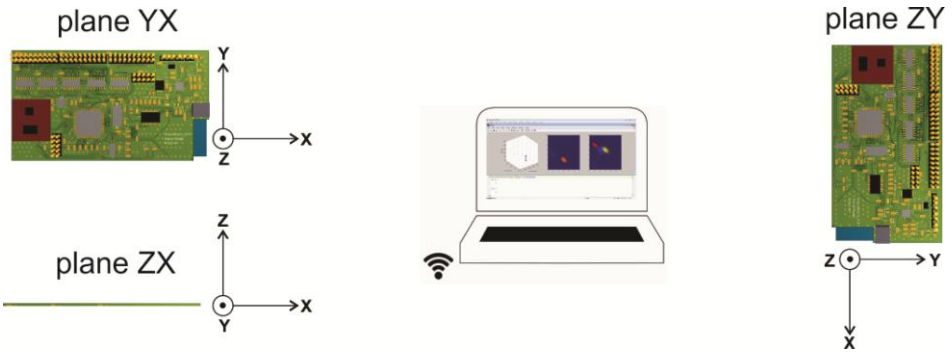


Figure 53 – WiFi range distance test.

The results have shown that the minimum covered distance was 100 meters on the plane ZX and the maximum distance was 230 meters in the plane ZY.

5.1.5. POWER CONSUMPTION

Due to the highest power consumption occurs when the data is transmitting by WiFi, the system was powered on in Acquisition mode while the current was verified on the multimeter. The total current verified was ~100 mA which gives a power consumption of 330 mW due to the supply voltage of 3.3 V.

5.2. PLANTAR PRESSURE

The plantar pressure measurements allow to determine the CoP and feet position as will be described next.

5.2.1. CoP

To maintain the balance when over the surfboard the surfers must respond to the changes in surfboard position by changing the CoP dynamically. The tests described earlier allow the determination of the CoP displacement and the obtained results are shown in Table 15. The first hundred samples were not considered due to the initial transient conditions.

Table 15 – CoP displacement and rotation along the simulation tests [1].

Tests	CoP X	CoP Y	Pitch Rotation	Roll Rotation
Test 1	63.72 ± 5.54	7.35 ± 1.36	-1.90 ± 1.31	-0.81 ± 0.97
Test 2	60.73 ± 9.23	10.32 ± 2.09	-30.03 ± 1.71	-3.54 ± 1.90
Test 3	67.82 ± 3.79	11.09 ± 3.38	-3.31 ± 2.65	1.91 ± 12.25

[1] The parameters corresponds to mean and standard deviation and the unit are presented in cm

During Test 1, a minimum deviation in rotation is observed keeping the CoP closest to the mean value in comparison with the other tests. On the other hand, Tests 2 and 3 presented higher deviations in Pitch and Roll, changing the CoP position as well. Furthermore, the Test 2 presented a CoP displacement in direction of the rear platform due to the pitch rotation performed by the surfer. Moreover, in Test 3 is observed the highest rotation deviation, mainly in Roll axis, which corroborates with the movements performed to move the surfboard sideways. Therewith, the test results have shown that the system is able to measure and quantify the CoP displacement. Figure 54 shows the CoP displacement over the surfboard drawing where is clearly noticeable the CoP displacement backward during Test 2 and the highest deviation during Test 3.

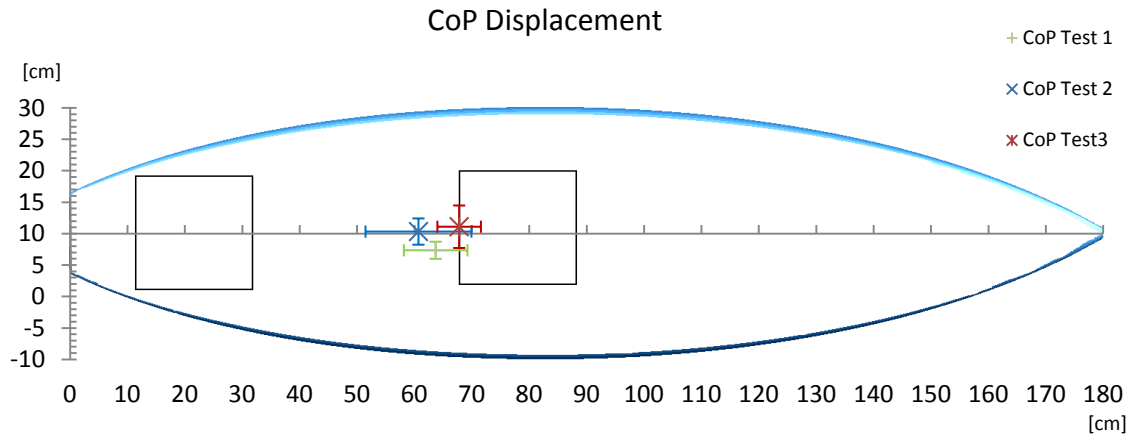


Figure 54 – CoP displacement over the surfboard during tests performed.

5.2.2. FEET POSITION

To determine the feet position, the plantar pressure measurement is done by reading the force sensors and, due to the knowledge of the surfer's foot stance, it is possible to establish the position.

Due to the data measured from the force sensors and the analysis of the sensors that presented an average value greater than zero, the foot position was establish for each trial. The Figure 55 shows the suggested position in one trial of each test.

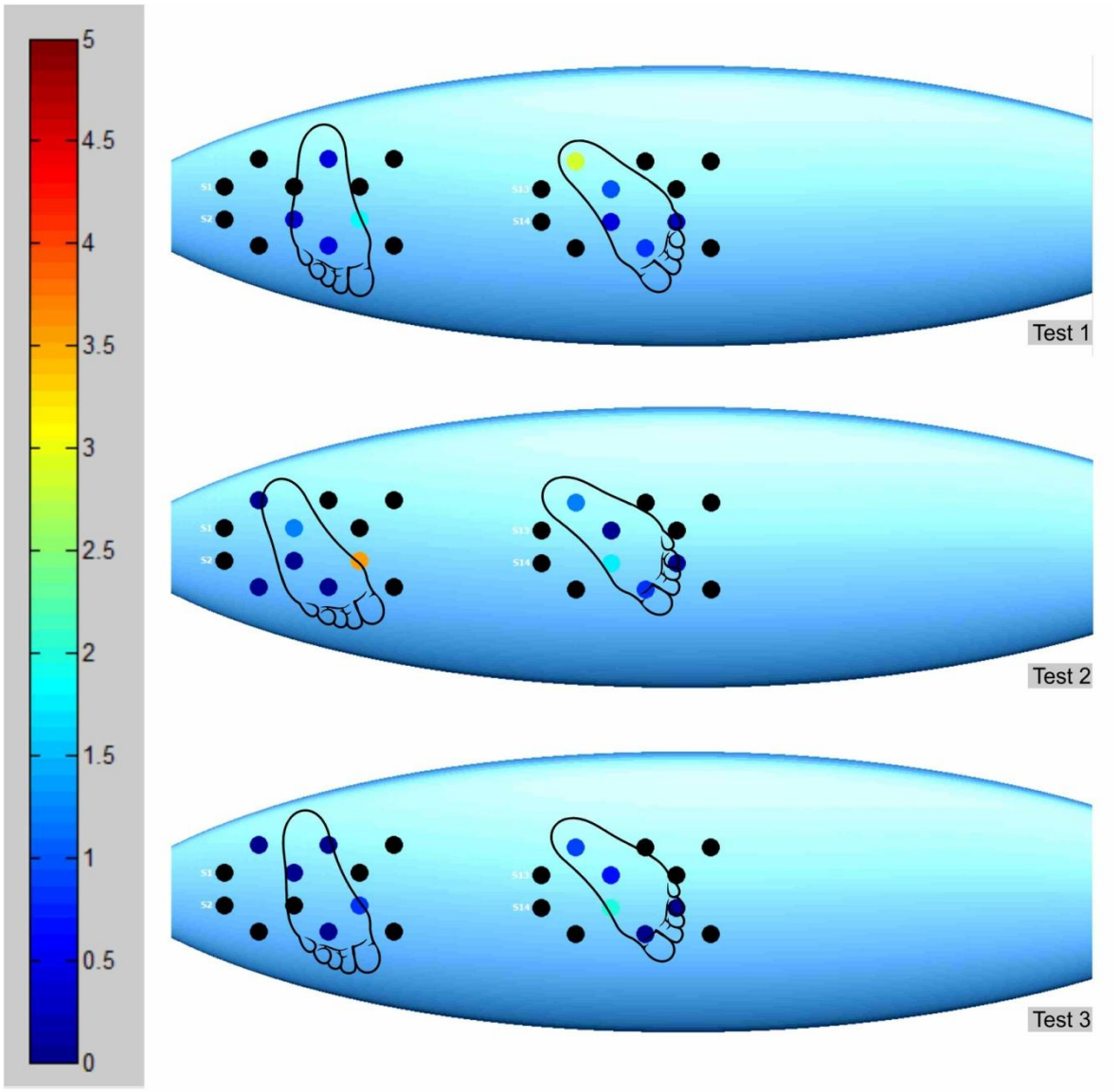


Figure 55 – Feet position from plantar pressure measurement.

It was observed during these tests, that due to the low spatial resolution, it was difficult to establish the feet position accurately, meaning that the spatial resolution is insufficient due to sensor discretization. However the results show that it is possible to use this method to establish that feature.

5.3. SURFBOARD MOVEMENT

The characterization of the surfboard movement over X and Y axes is obtained from the IMU data, where pitch and roll rotations and acceleration define sideways, forward and backward movements. The tests have shown a relation between CoP displacement and surfboard rotation for that specific setup; although the surfboard angles can vary without CoP changing due to external forces acting over the surfboard. The Table 15 shows the relation between CoP and rotation considering the displacement deviation. However, to analyze the movements described in the set of manoeuvers chosen, it might be useful to identify the transitions, for instance, when there are changes between surfboard's edges on the water.

The tests performed aimed to verify the capacity of measure the surfboard's rotation angles. Due to that Figure 56 shows both rotation axes during different trials where Test (2) represents the second trial on Test 2, likewise, Test (3) represents the first trial on Test 3.

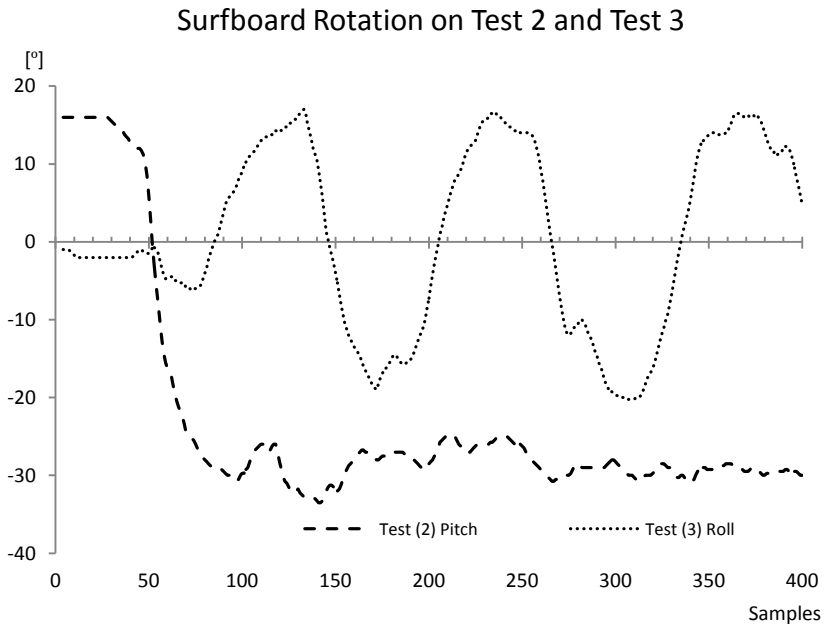


Figure 56 – Pitch and Roll along the Test 2 and Test 3.

The test results have shown that both pitch and roll can be measured for the system at the same time. Similarly, during Test 1, it was observed the stabilization of pitch and roll around zero degree is reached. It is also possible to verify the initial transients when the surfboard has its position changed by the surfer. Figure 57 shows the pitch and roll during the first trial of Test 1.

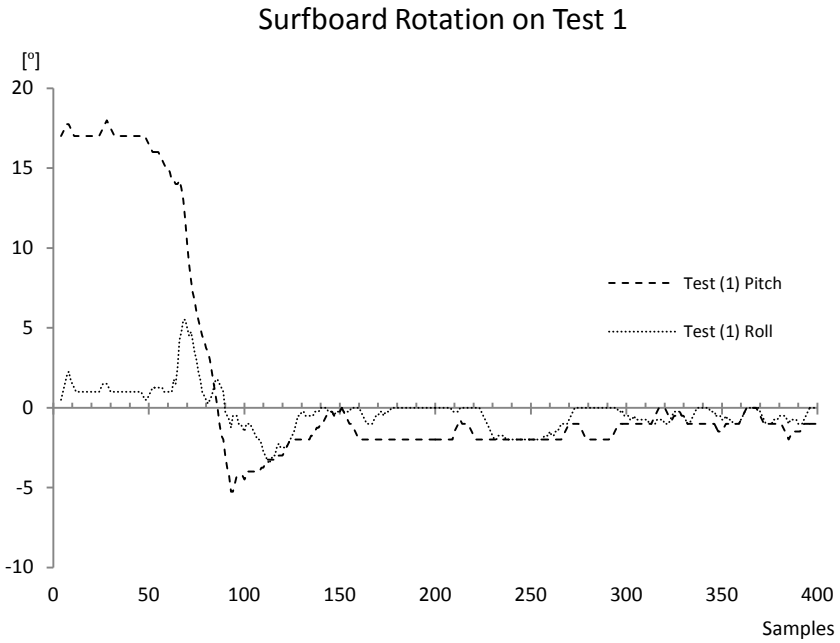


Figure 57 – Pitch and Roll along Test 1.

6. CONCLUSION

The use of technology is growing every day and in electronic field it is more visible due to the huge number of applications that are being developed either to make easier daily life or to help in medical diagnosis of diseases, for instance.

The literature review has shown an existing gap in surf analysis although there have been efforts in order to minimize the gap between the practicing and practice evaluation. Despite of been considered a relaxing activity, surfing requires great physical effort from the surfer mainly due to the environmental conditions and the movements complexity. Furthermore, the surfer is highly susceptible for occurring injuries.

These facts show the relevance of developing a system to collect data during any surf session to assist surfers and coaches in understanding the physical stimuli in which the surfer's body and the surfboard are subject, helping in designing of specific training practices, for example.

The development of a system to monitor surfer and surfboard’s movements was a challenge greatly due to the limited number of scientific work published, the scarce of quantitative data from the wave riding movements, which was only described qualitatively from a few authors.

The system proposed in this project aimed to fill this gap by developing an electronic system capable of acquiring dynamic data of the center of pressure and surfboard's position during surf practice. Although the evaluation in the water has not been possible due to time and budget limitations, the tests results have shown that is possible to determine the proposed variables. Furthermore, the use of a video analysis altogether with this system can contribute to achieve better results due to the visual feedback provided by the images.

The Flexiforce sensors used to acquire plantar pressure proved to be a good choice for this application due to the response linearity when subject to the force within its operational range. The sensors are also lightweight and thin which are relevant characteristics due to the low weight of the surfboards. However, the terminal connections have presented bad contact after some use which can be resolved by applying matrices off the shelf.

The IMU used also proved to be suitable for this application since tests have shown a correct response from the device. Despite of not be the most up-to-date device in the market, the IMU complied with the aim of measuring the pitch and roll axes, however the five degree of freedom have limited the application, since yaw axis was not measured.

For wireless communication, a WiFi module was used instead of technologies like Bluetooth, for example, since the distance covered is greater. The module applied proved to be flexible enough to allow the *adhoc* communication as was implemented and also to allow configuration to be used with routers if necessary. Despite of the maximum power consumption described by the manufacture of ~700 mW (when transmitting) which is far from ideal for remote application using batteries, as it was the case, the system power consumption verified was 330 mW

When testing the developed system, it was shown that the measurement error is small enough (-0.012 ± 0.064 N) to be disregarded in this application. Also, the linearity present in the sensor’s calibration showed that the system works properly.

The movements’ simulation protocol allowed testing the system dynamically, but despite being a controlled environment, the unbalanced support provided by the BOSU Ball created an ideal scenario to simulate those movements.

The system developed overcomes the absence of quantitative data in this field, by collecting dynamic data from the plantar pressure and surfboard’s rotation to quantify

factors known to affect surfer's performance like feet position, CoP and surfboard's movement. Furthermore, since the system enables to identify biomechanical parameters that can be used to help in injuries preventions, could be a possible development of this system. Moreover, the system has become the first electronic embedded equipment exclusively applied on surf that had presented results that have shown these factors.

Finally, the project's development contributed to develop ideas on that specific application, providing a constant growth in partnerships and also in know hall about it.

6.1. FUTURE WORKS

The most important future work in this project is to integrate the electronic system into a waterproof case to allow the use during real situation. One possible solution consists in the use of a waterproof case for the PCB and a rubber or silicon sheet to seal the sensors area. However, other solutions can be tested like the use of flexible electronic circuits to allow the integration between sensors and electronics, removing the need of cables and reducing the equipment size.

To improve the system is also important, mainly in the establishment of the feet positioning, it is an advised to increase the number of sensors which can be done by using the sensor's matrix available in the market that can provide higher resolution.

Another future work is the implementation of a supply management circuit that uses the USB connection to charges the battery. The integrated circuit bq24278 is one option once it controls the battery charge besides provide control signals which the system can use to determine system capacity, for instance.

Afterwards, testing the system in a real situation, i.e. inside the water, during wave riding it is a need for future works, characterizing the surf movements allowing comparing different rides and evaluating them.

References

[1] Dezan de Bona, D., G. de Salvador Ferreira, and L. Schwarz. Sensoriamento remoto em pranchas de surfe utilizando tecnologia ZigBee. INDUSCON 2010: 9th IEEE/IAS International Conference on Industry Application, São Paulo, Brazil, 2010.

[2] Silva, A.S., et al., WIMU: WEARABLE INERTIAL MONITORING UNIT. BIODEVICES: International Conference on Biomedical Electronics and Devices, Rome, Italy, 2011.

[3] LIEBERMANN, Dario G., L.K.T., HUGHES, Mike D., BARTLETT, Roger M., McCLEMENT, Jim S., FRANKS, Ian M., Advances in the application of information technology to spor t performance. Journal of Sports Sciences, 2002: p. 14.

[4] Gesser, F.J., Possobon, F.R, Bonacorso, N.G., Silva, R., Desenvolvimento e construção de uma fresadora CNC de baixo custo destinada a confecção de pranchas de surf. 2007.

[5] Ltd, F.C.S.P. 2013 [cited 2013 15/05]; Available from: http://www.surffcs.com/.

[6] Everline, C., Shortboard Performance Surfing: A Qualitative Assessment of Maneuvers and a Sample Periodized Strength and Conditioning Program In and Out of the Water. Strength & Conditioning Journal, 2007. 29(3): p. 32-40.

[7] Nathanson, A., P. Haynes, and D. Galanis, Surfing injuries. The American Journal of Emergency Medicine, 2002. 20(3): p. 155-160.

[8] Nathanson, A., et al., Competitive Surfing Injuries A Prospective Study of Surfing-Related Injuries Among Contest Surfers. The American Journal of Sports Medicine, 2007. 35(1): p. 113-117.

[9] Lowdon, B., N. Pateman, and A. Pitman, Surfboard-riding injuries. The Medical Journal of Australia, 1983. 2(12): p. 613.

[10] Taylor, D.M., et al., Acute injury and chronic disability resulting from surfboard riding. Journal of science and medicine in sport / Sports Medicine Australia, 2004. 7(4): p. 429-437.

[11] Farley, O.R.L., N.K. Harris, and A.E. Kilding, Physiological Demands of Competitive Surfing. The Journal of Strength & Conditioning Research, 2012. 26(7): p. 1887-1896.

[12] Meir, R.A., B.J. Lowdon, and A.J. Davie, Heart rates and estimated energy expenditure during recreational surfing. Australian Journal of Science and Medicine in Sport, 1991. 23(3): p. 70-74.

[13] Loveless, D.J. and C. Minahan, Two reliable protocols for assessing maximal-paddling performance in surfboard riders. Journal of Sports Sciences, 2010. 28(7): p. 797-803.

[14] Mendez-Villanueva, A., et al., Inaccuracy of the HR Reserve vs. VO2 Reserve Relationship during Prone Arm-paddling Exercise in Surfboard Riders. Journal of Physiological Anthropology, 2010. 29(6): p. 189-195.

[15] Peirão, R. and S.G.d. Santos, Judging criteria in international professional surfing championships. Revista Brasileira de Cineantropometria & Desempenho Humano, 2012. 14(4): p. 439-449.

[16] Mendez-Villanueva, A., D. Bishop, and P. Hamer, Activity Profile of World-Class Professional Surfers During Competition: Acase Study. The Journal of Strength & Conditioning Research, 2006. 20(3): p. 477-482.

[17] Professionals, Association of Surfing, [cited 2013 10/11]; Available from: http://www.aspworldtour.com/.

<ASP rule book.pdf>. 2013.

[18] Mendez-Villanueva, A. and D. Bishop, Physiological aspects of surfboard riding performance. Sports Medicine, 2005. 35(1): p. 55-70.

[19] Meir, R.A., B.J. Lowdon, and A.J. Davie, Heart rates and estimated energy expenditure during recreational surfing. Aust J Sci Med Sport, 1991. 23: p. 70-4.

[20] Lowdon, B.J., et al., Manoeuvres Used and Judges' Scores in an International Surfing Contest: Summary Report. 1996: National Sports Research Centre.

[22] Mendez-Villanueva, A., I. Mujika, and D. Bishop, Variability of competitive performance assessment of elite surfboard riders. The Journal of Strength & Conditioning Research, 2010. 24(1): p. 135-139.

[23] Association of Surfing Professionals. ASP World Championship Tour Ranking. 2013 [cited 2014 03/01]; Available from: http://www.aspworldtour.com/rankings/asp-world-championship-tour-ranking/.

[24] Foundation, T.R.I. Pukas and Tecnalia work together to develop the world's first surfboard with integrated techology. 2011 22/02/2011 [cited 2011 10/03]; Available from: http://www.tecnalia.com/en/press-room/press-releases/tecnalia/pukas-tecnalia-surfboard-integrated-technology.htm.

[25] AlpineReplay. Trace - ActiveReplay. 2013 [cited 2013 10/10]; Available from: http://www.alpinereplay.com/trace/pre-order.

[26] Winter, D.A., A.E. Patla, and J.S. Frank, Assessment of balance control in humans. Med Prog Technol, 1990. 16(1-2): p. 31-51.

[27] Nashner, L.M., Practical biomechanics and physiology of balance. Handbook of balance function and testing, 1993: p. 261-279.

[28] Hrysomallis, C., Balance ability and athletic performance. Sports Medicine, 2011. 41(3): p. 221-232.

[29] Incorporation, T., FlexiForce Sensor User Manual. 2010. p. 15.

[30] Devices, A. ADXL335 Small, Low Power, 3-Axis ±3 g Accelerometer. 2010, [cited 2013 17/11]; Available from: http://www.analog.com/static/imported-files/data_sheets/ADXL335.pdf.

[31] Incorporation, I. IDG-500 Dual-Axis Gyro Product Specification. 2009 [cited 2013 03/04]; Available from: http://invensense.com/mems/gyro/documents/PS-IDG-0500-00-06.pdf.

[32] Limited, F.T.D.I. FT232R USB UART IC. 2010 [cited 2013 03/04]; Available from: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

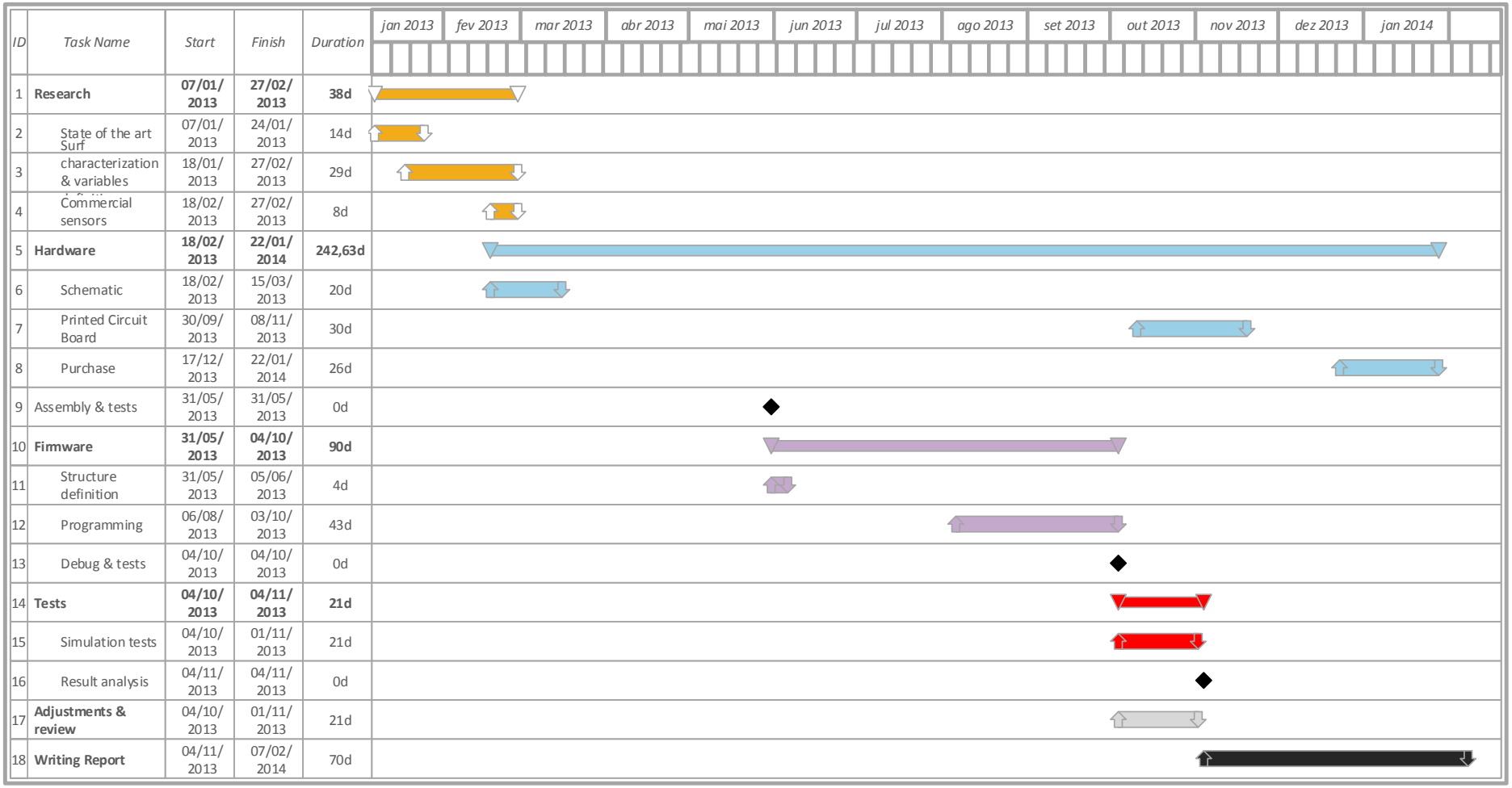
[33] Incorporation, R.N. RN-131G & RN-131C 802.11 b/g Wireless LAN Module. 2012 [cited 2013 05/04]; Available from: http://ww1.microchip.com/downloads/en/DeviceDoc/rn-131-ds-v3.2r.pdf.

[34] DharmaniTech. microSD ATmega32 Data-Logger. 2011 [cited 2013 01/11]; Available from: http://www.dharmanitech.com/.

[35] Project, S.E. A guide to using IMU (Accelerometer and Gyroscope Devices) in embedded Applications. 2009 [cited 2013 01/11]; Available from: http://www.starlino.com/imu_guide.html.

[36] Ltd., A. Rufa 2.4 GHz SMD Antenna. 2008 [cited 2013 04/04/2013]; Available from: http://ww1.microchip.com/downloads/en/DeviceDoc/Acc-Antanova-Chip-Ant-DS.pdf.

APPENDIX A. Project schedule



ID	Task Name	Start	Finish	Duration
1	**Research**	**07/01/2013**	**27/02/2013**	**38d**
2	State of the art Surf	07/01/2013	24/01/2013	14d
3	characterization & variables	18/01/2013	27/02/2013	29d
4	Commercial sensors	18/02/2013	27/02/2013	8d
5	**Hardware**	**18/02/2013**	**22/01/2014**	**242,63d**
6	Schematic	18/02/2013	15/03/2013	20d
7	Printed Circuit Board	30/09/2013	08/11/2013	30d
8	Purchase	17/12/2013	22/01/2014	26d
9	Assembly & tests	31/05/2013	31/05/2013	0d
10	**Fimware**	**31/05/2013**	**04/10/2013**	**90d**
11	Structure definition	31/05/2013	05/06/2013	4d
12	Programming	06/08/2013	03/10/2013	43d
13	Debug & tests	04/10/2013	04/10/2013	0d
14	**Tests**	**04/10/2013**	**04/11/2013**	**21d**
15	Simulation tests	04/10/2013	01/11/2013	21d
16	Result analysis	04/11/2013	04/11/2013	0d
17	**Adjustments & review**	04/10/2013	01/11/2013	21d
18	**Writing Report**	04/11/2013	07/02/2014	70d

APPENDIX B. Flexiforce Datasheet

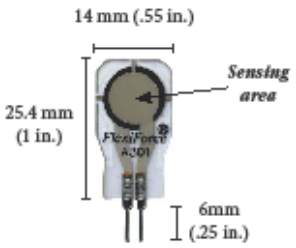
FlexiForce®

Standard Force & Load Sensors Model # A301

Physical Properties

Thickness	0.203 mm (0.008 in.)
Length	25.4 mm (1 in.)*
Width	14 mm (0.55 in.)
Sensing Area	9.53 mm (0.375 in.) diameter
Connector	2-pin Male Square Pin
Substrate	Polyester (ex: Mylar)
Pin Spacing	2.54 mm (0.1 in.)

✓ROHS Compliant



Actual size of sensor

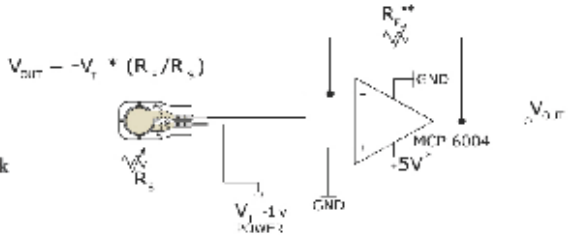
** Length does not include pins, please add approximately 6mm (.25 in.) for pin length for a total length of approximately 32mm (1.25 in).*

Standard Force Ranges (as tested with circuit shown below)

Force Range:
Low: 0 - 1 lb. (4.4 N)
Medium: 0 - 25 lb. (111 N)
High: 0 - 100 lb. (445 N)

Force Range Adjustments:
In order to measure higher forces, apply a lower drive voltage (-0.5 V, -0.10 V, etc.) and reduce the resistance of the feedback resistor (1kΩ min.) To measure lower forces, apply a higher drive voltage and increase the resistance of the feedback resistor.

Recommended Circuit



- * Supply Voltages should be constant
- ** Tolerance Resistance R_F is 1kΩ to 100kΩ
- Sensor Resistance R_S at no load is >5MΩ
- Max recommended current is 2.5mA

Typical Performance

Linearity (Error)	< ±3%
Repeatability	< ±2.5% of full scale
Hysteresis	< 4.5 % of full scale
Drift	< 5% per logarithmic time scale
Response Time	< 5 µsec
Operating Temperature	-40°F - 140°F (-40°C - 60°C)*

*Force reading change per degree of temperature change = ±0.2%/°F (0.36%/°C)

Evaluation Conditions

Line drawn from 0 to 50% load
Conditioned sensor, 80% of full force applied
Conditioned sensor, 80% of full force applied
Constant load
Impact load, output recorded on oscilloscope
Time required for the sensor to respond to an input force

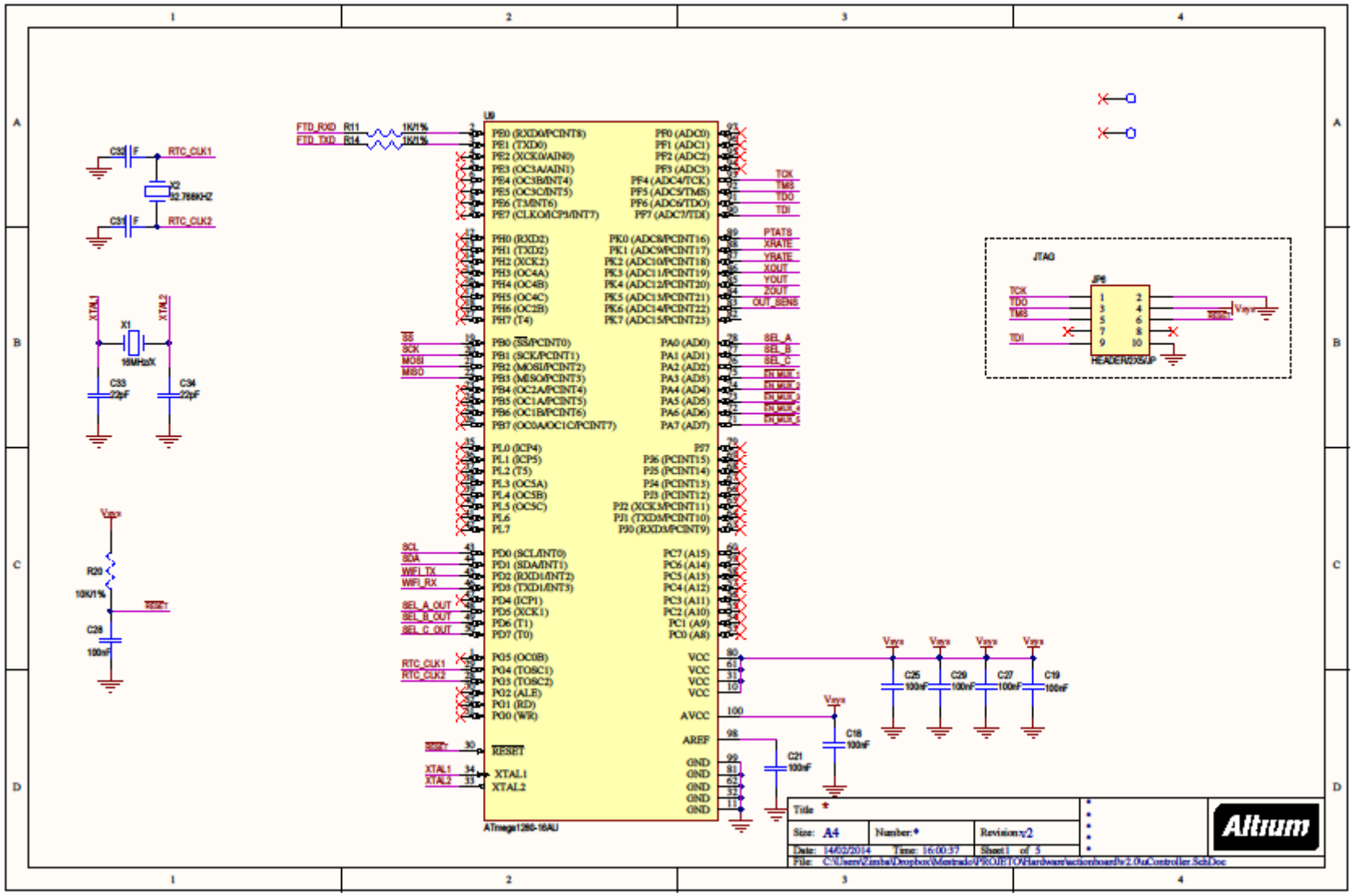
Tekscan, Inc. 307 West First Street South Boston, MA 02127-1309 USA tel: 617.464.4500/800.248.3669
fax: 617.464.4266 e-mail: marketing@tekscan.com URL: www.tekscan.com

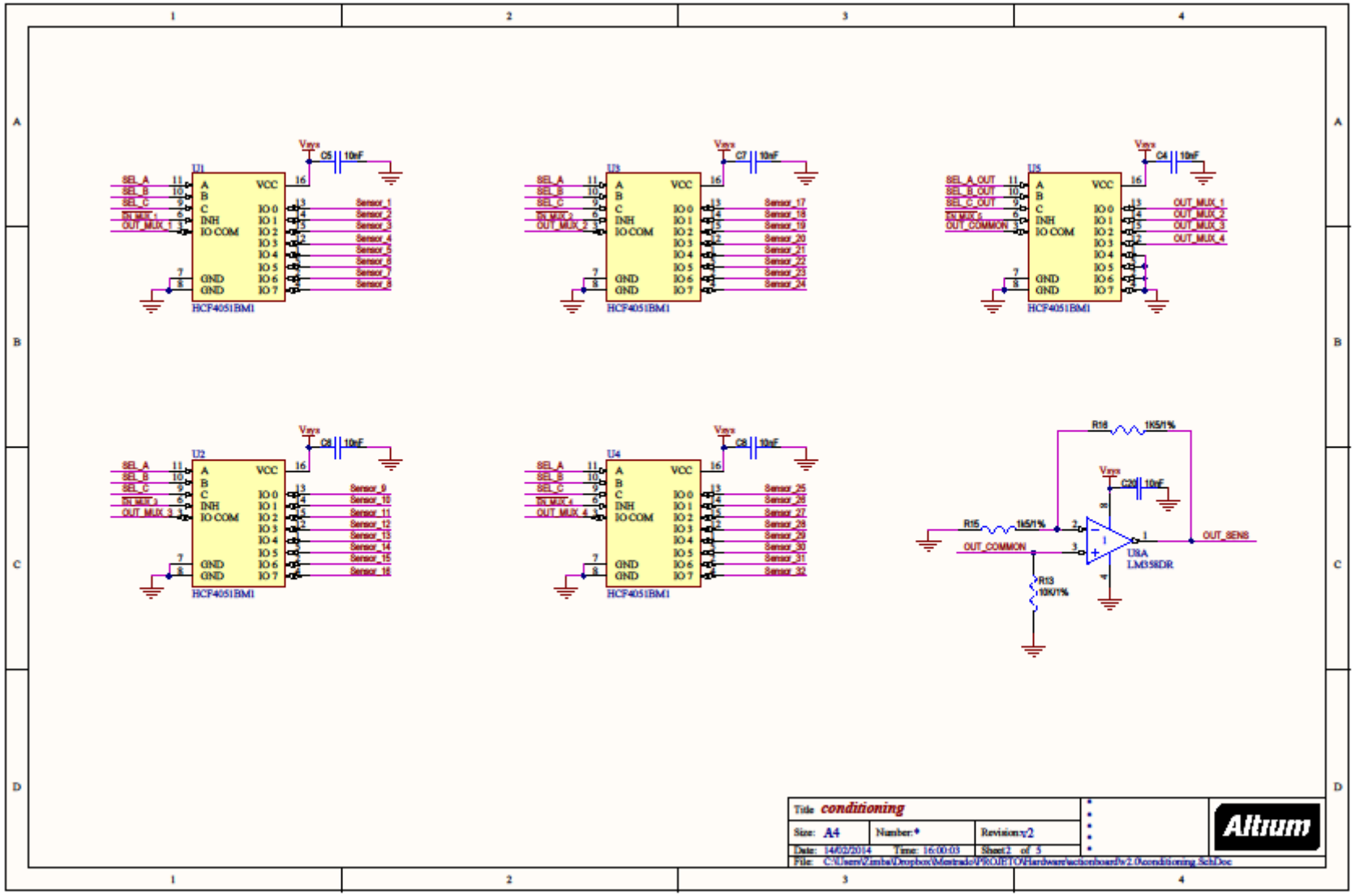
Tekscan

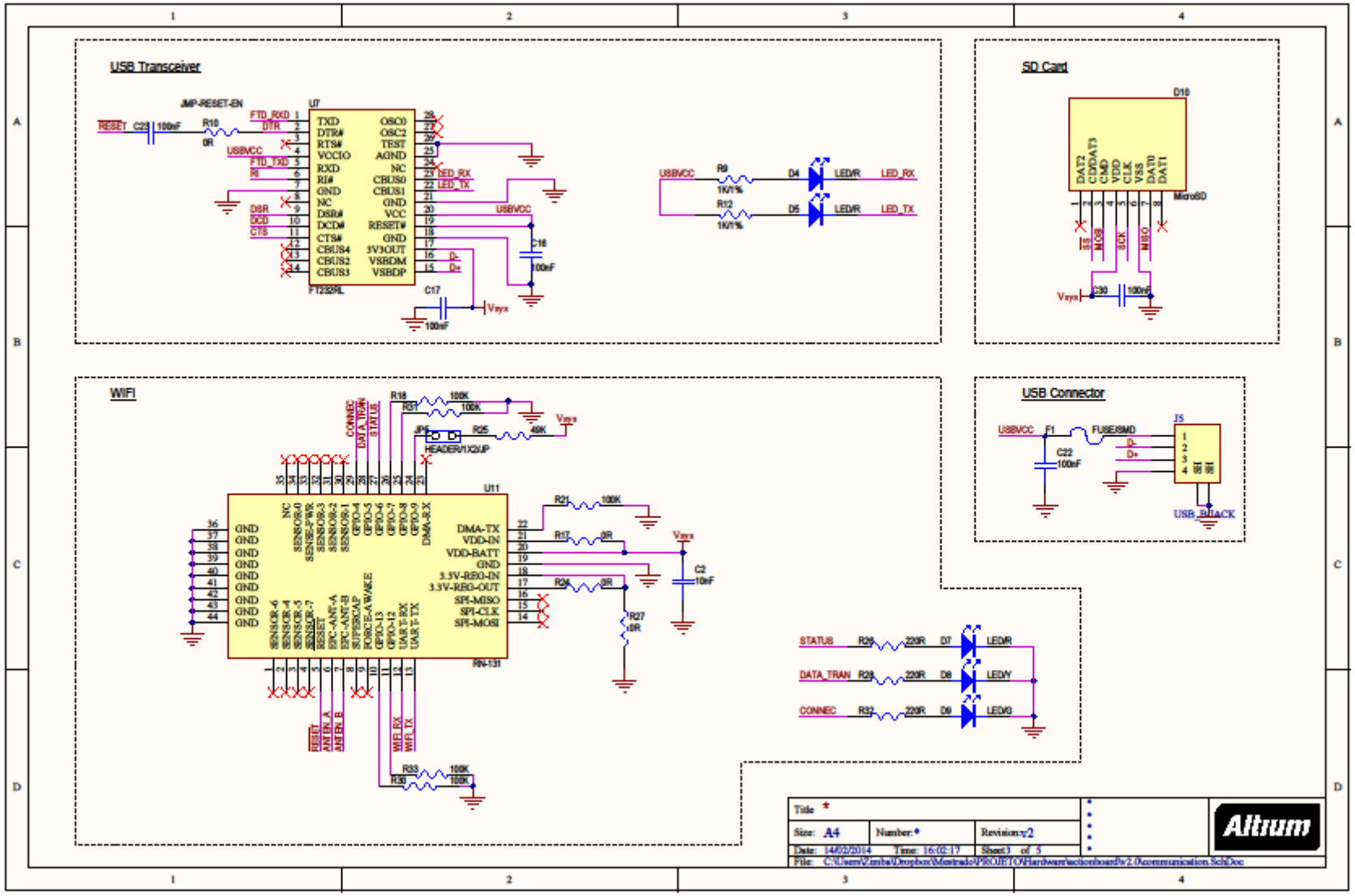
Rev E_9.30.13

APPENDIX C. Circuit schematic diagram

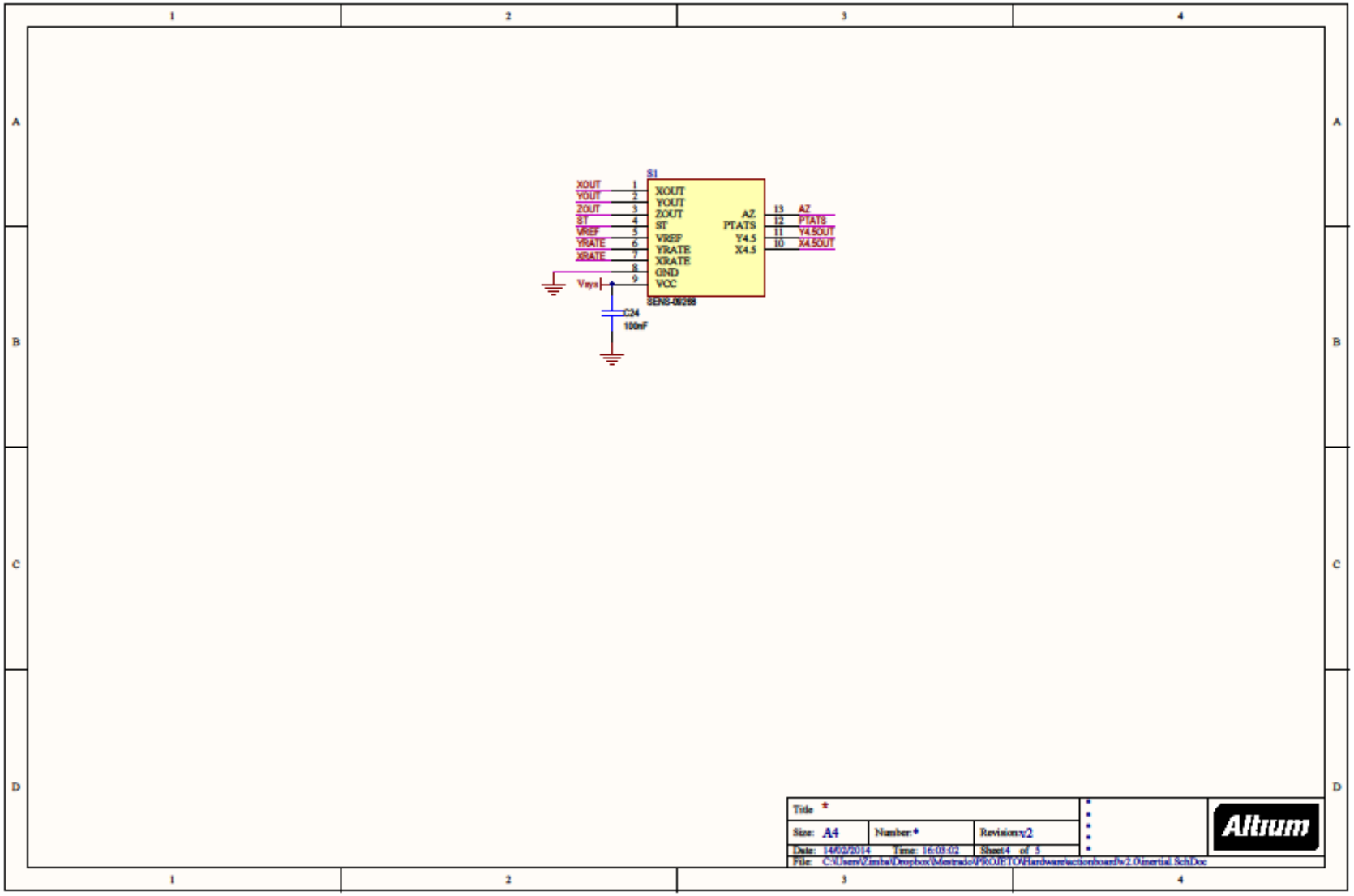
Table of Contents	
Page 1	Microcontroller connections and JTAG connector.
Page 2	Conditioning circuit with multiplexers and OPAMP.
Page 3	Communication devices and micro SD connector.
Page 4	SENS-09268 IMU connections.
Page 5	Connectors.

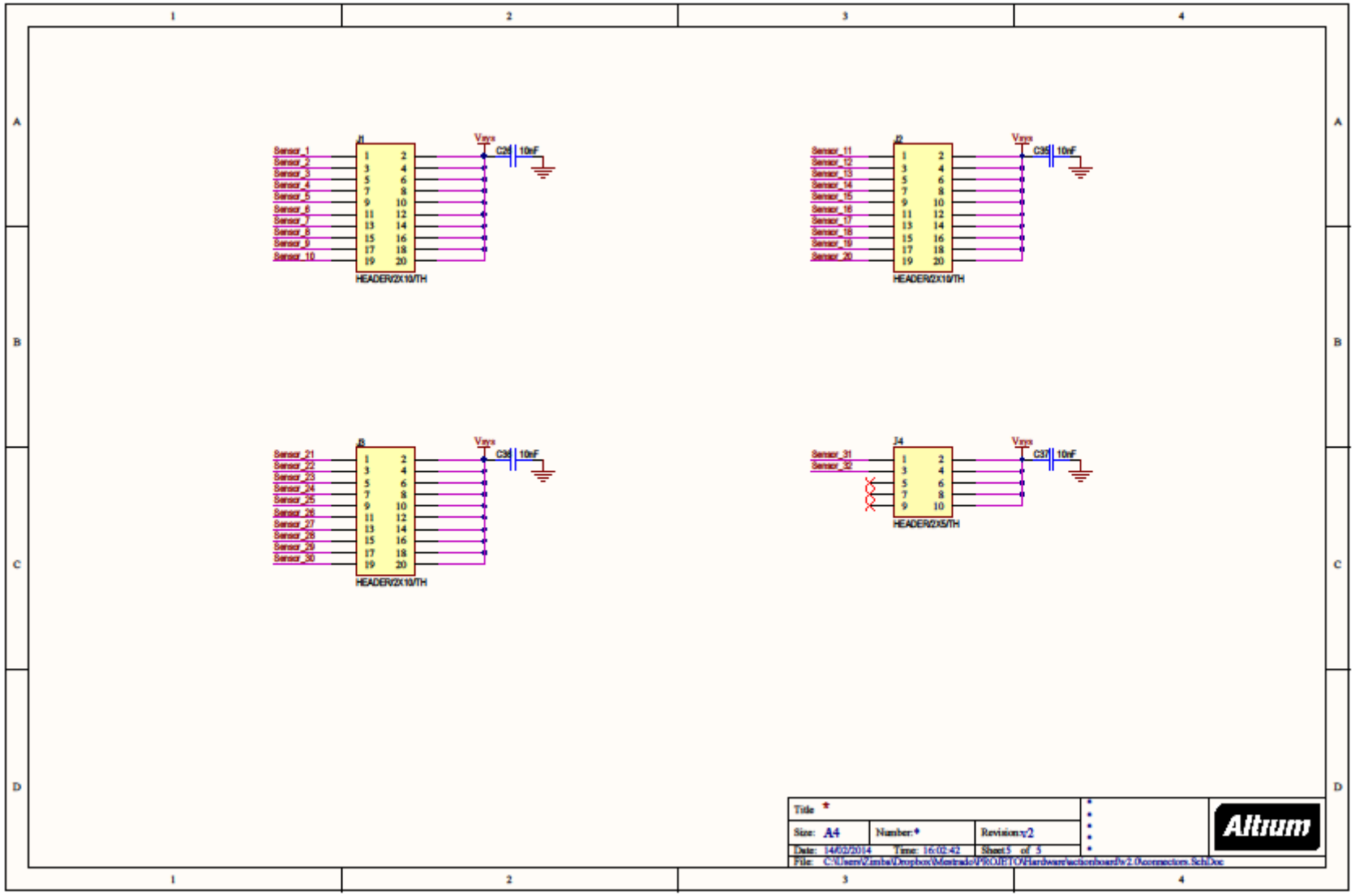

U9
ATmega1280-16AU
FTD_RXD
FTD_TXD
R11
R14
1K/1%
RTC_CLK1
RTC_CLK2
X2
32.768KHZ
C32
C31
XTAL1
XTAL2
X1
16MHz/X
C33
C34
22pF
Vsys
R20
10K/1%
RESET
C28
100nF
SS
SCK
MOSI
MISO
SCL
SDA
WIFI_TX
WIFI_RX
SEL_A_OUT
SEL_B_OUT
SEL_C_OUT
TCK
TMS
TDO
TDI
PTATS
XRATE
YRATE
XOUT
YOUT
ZOUT
OUT_SENS
SEL_A
SEL_B
SEL_C
EN_MUX_1
EN_MUX_2
EN_MUX_3
EN_MUX_4
EN_MUX_5
JTAG
JP6
HEADER2X5/JP
C25
C29
C27
C19
C18
C21
100nF
Title *
Size: A4
Number: *
Revision: v2
Date: 14/02/2014
Time: 16:00:37
Sheet 1 of 5
File: C:\Users\Z\mba\Dropbox\Mestrado\PROJETO\Hardware\actionboard\v2.0\uController.SchDoc
Altium


U1
HCF4051BM1
C5 10nF
SEL_A
SEL_B
SEL_C
IN_MUX_1
OUT_MUX_1
Sensor_1
Sensor_2
Sensor_3
Sensor_4
Sensor_5
Sensor_6
Sensor_7
Sensor_8
U3
HCF4051BM1
C7 10nF
IN_MUX_2
OUT_MUX_2
Sensor_17
Sensor_18
Sensor_19
Sensor_20
Sensor_21
Sensor_22
Sensor_23
Sensor_24
U5
HCF4051BM1
C4 10nF
SEL_A_OUT
SEL_B_OUT
SEL_C_OUT
IN_MUX_5
OUT_COMMON
OUT_MUX_1
OUT_MUX_2
OUT_MUX_3
OUT_MUX_4
U2
HCF4051BM1
C6 10nF
IN_MUX_3
OUT_MUX_3
Sensor_9
Sensor_10
Sensor_11
Sensor_12
Sensor_13
Sensor_14
Sensor_15
Sensor_16
U4
HCF4051BM1
C8 10nF
IN_MUX_4
OUT_MUX_4
Sensor_25
Sensor_26
Sensor_27
Sensor_28
Sensor_29
Sensor_30
Sensor_31
Sensor_32
R18 1K5/1%
C20 10nF
R15 1k5/1%
OUT_COMMON
OUT_SENS
U8A
LM358DR
R13
10K/1%
Title conditioning
Size: A4
Number:*
Revision:v2
Date: 14/02/2014
Time: 16:00:03
Sheet 2 of 5
File: C:\Users\Zimba\Dropbox\Mestrado\PROJETO\Hardware\actionboardv2.0\conditioning.SchDoc
Altium

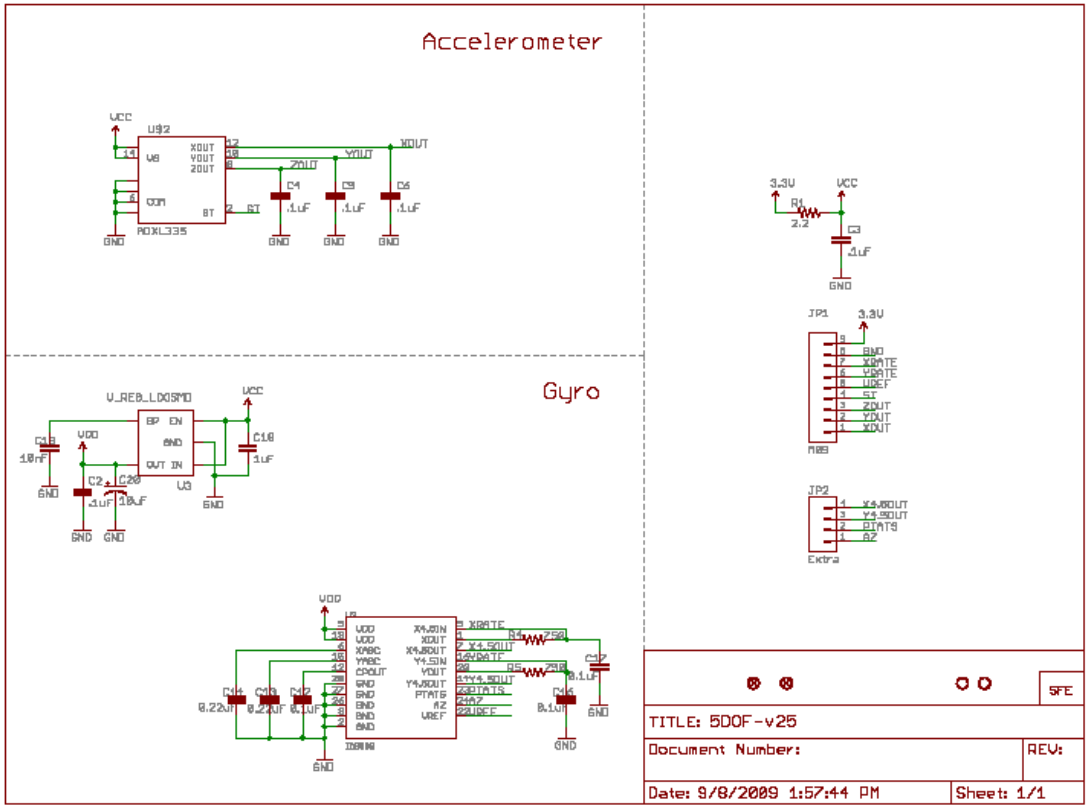

USB Transceiver
JMP-RESET-EN
U7
FT232RL
SD Card
D10
MicroSD
WIFI
U11
RN-131
USB Connector
J5
USB_BTACK
Title *
Size: A4
Number: *
Revision: v2
Date: 14/02/2014
Time: 16:02:17
Sheet 3 of 5
Altium

91


S1
XOUT 1
YOUT 2
ZOUT 3
ST 4
VREF 5
YRATE 6
XRATE 7
8
9
Vsys
XOUT
YOUT
ZOUT
ST
VREF
YRATE
XRATE
GND
VCC
AZ 13
PTATS 12
Y4.5OUT 11
X4.5OUT 10
AZ
PTATS
Y4.5
X4.5
SEN8-00268
C24
100nF
Title *
Size: A4
Number: *
Revision: v2
Date: 14/02/2014
Time: 16:03:02
Sheet 4 of 5
File: C:\Users\Z.imba\Dropbox\Mestrado\PROJETO\Hardware\actionboard\v2.0\inertial.SchDoc
Altium

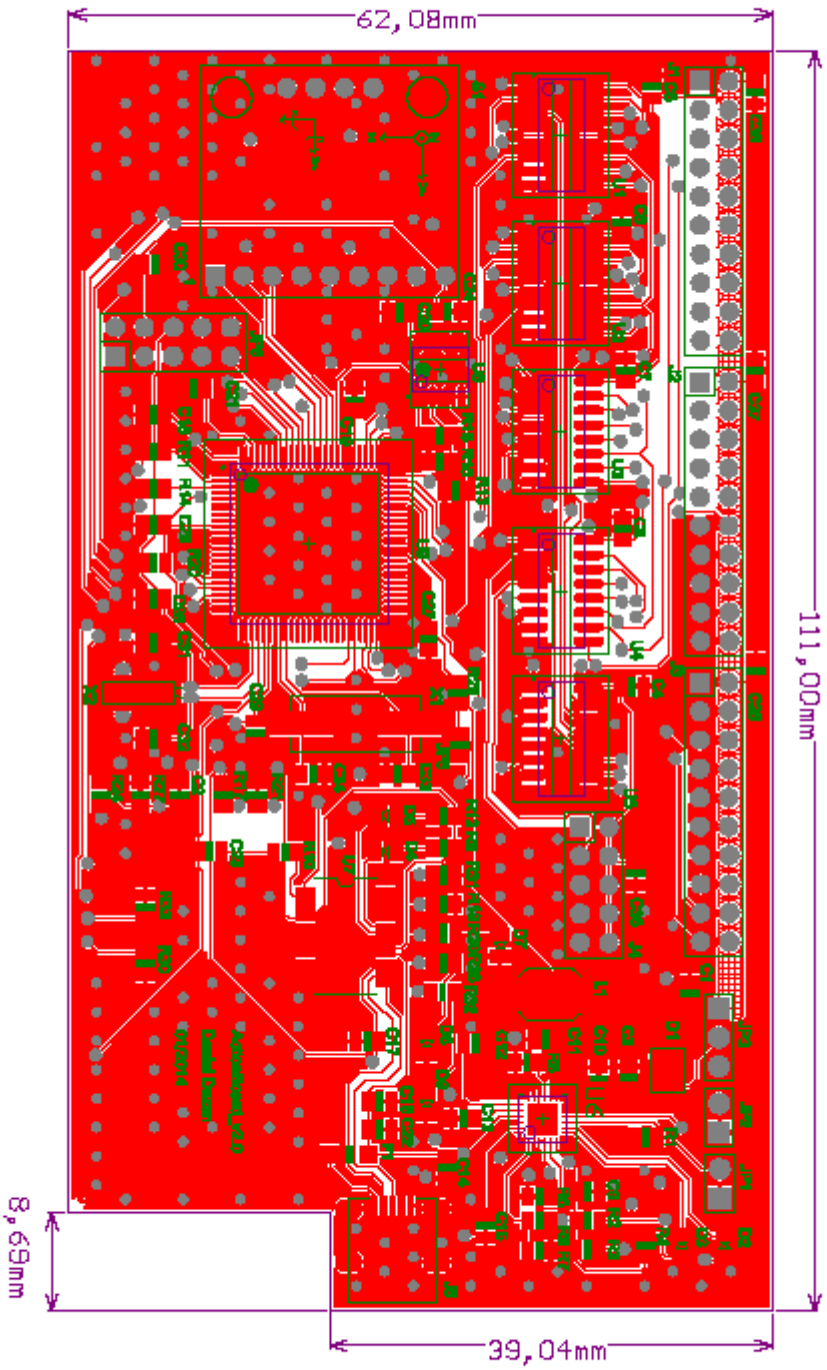

J1
Vmcu
C26 10nF
Sensor_1
Sensor_2
Sensor_3
Sensor_4
Sensor_5
Sensor_6
Sensor_7
Sensor_8
Sensor_9
Sensor_10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
HEADER2X10/TH
J2
Vmcu
C35 10nF
Sensor_11
Sensor_12
Sensor_13
Sensor_14
Sensor_15
Sensor_16
Sensor_17
Sensor_18
Sensor_19
Sensor_20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
HEADER2X10/TH
J3
Vmcu
C36 10nF
Sensor_21
Sensor_22
Sensor_23
Sensor_24
Sensor_25
Sensor_26
Sensor_27
Sensor_28
Sensor_29
Sensor_30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
HEADER2X10/TH
J4
Vmcu
C37 10nF
Sensor_31
Sensor_32
1 2 3 4 5 6 7 8 9 10
HEADER2X5/TH
Title *
Size: A4
Number: *
Revision: v2
Date: 14/02/2014
Time: 16:02:42
Sheet 5 of 5
File: C:\Users\Zimba\Dropbox\Mestrado\PROJETO\Hardware\actionboard\v2.0\connectors.SchDoc
Altium

APPENDIX D. SENS-09268 IMU Schematic diagram



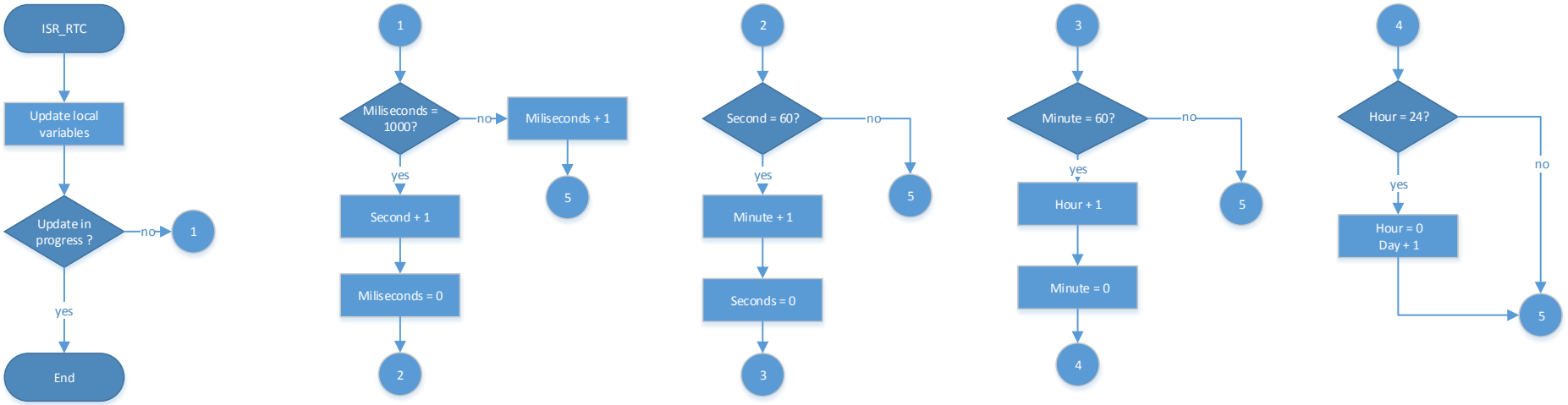
APPENDIX E. Printed Circuit Board

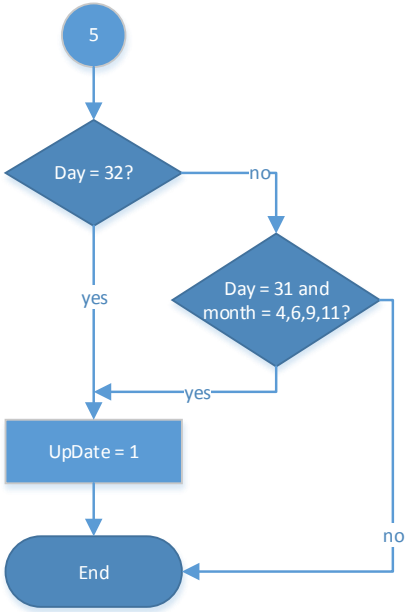
PCB Top Layer and Bottom Layer scaled in 2:1.

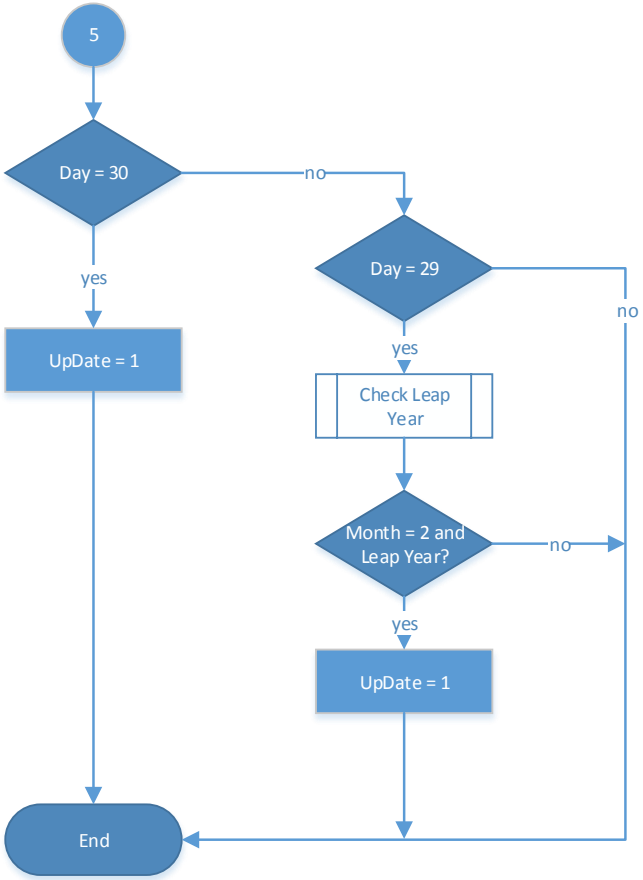

62,08mm
111,00mm
8,69mm
39,04mm

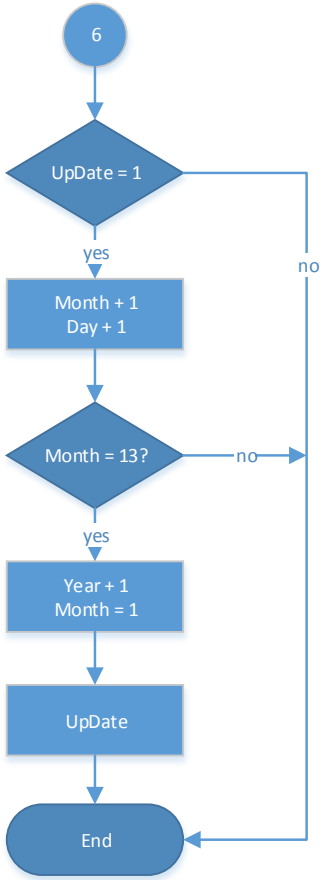
111,00mm
62,08mm
39,04mm
8,69mm

APPENDIX F. RTC flowchart



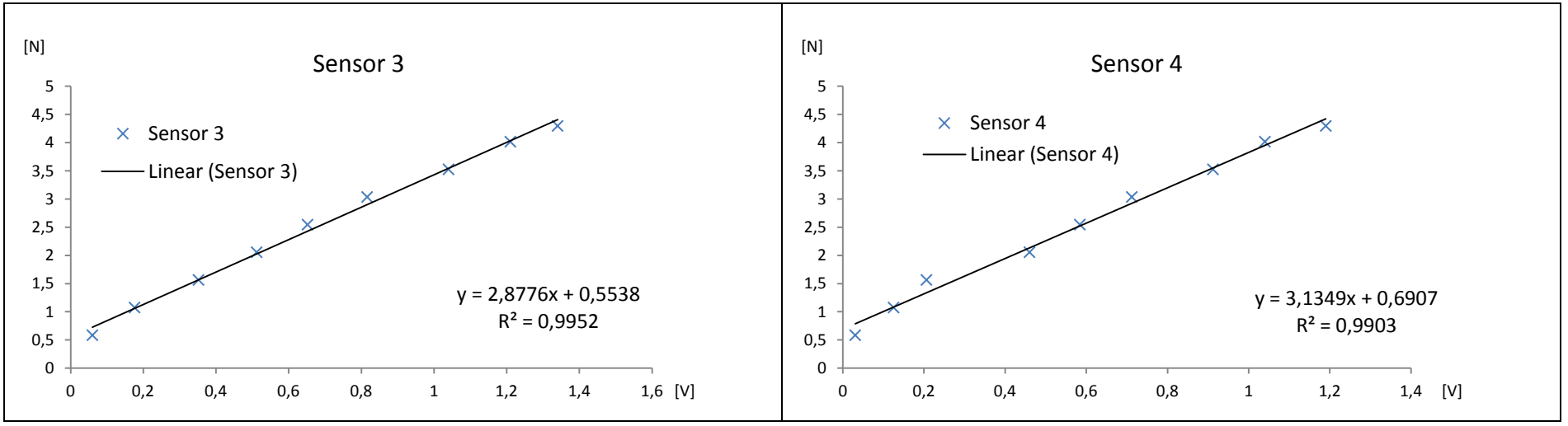

5
Day = 32?
no
yes
Day = 31 and
month = 4,6,9,11?
yes
UpDate = 1
End
no

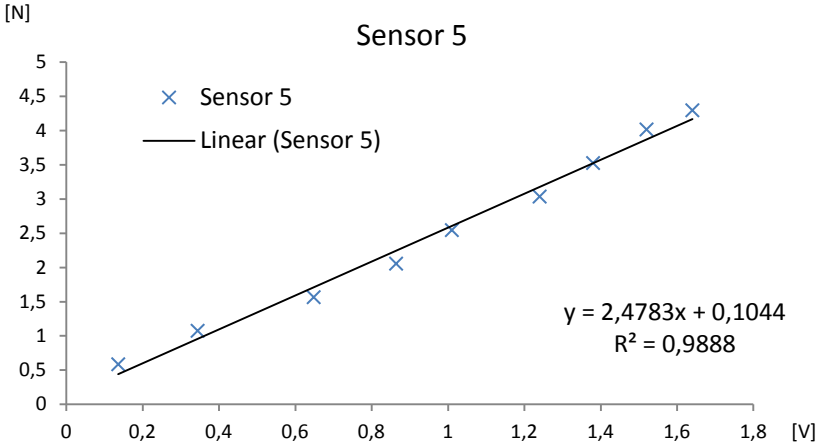

5
Day = 30
no
yes
Day = 29
yes
UpDate = 1
Check Leap
Year
Month = 2 and
Leap Year?
no
yes
UpDate = 1
End
no

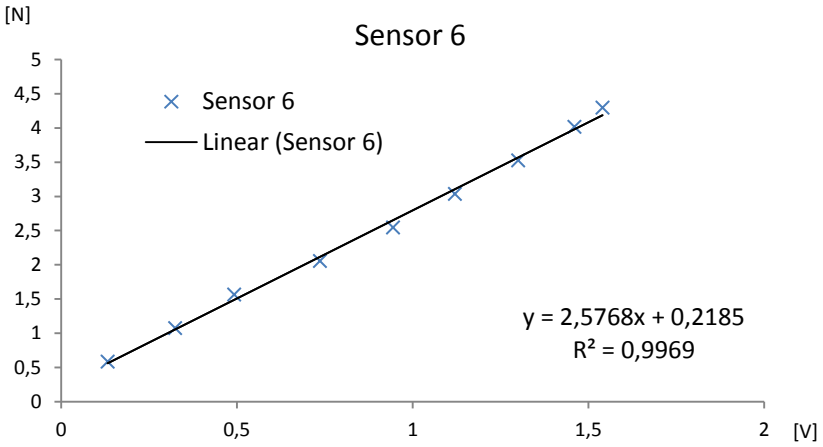

6
UpDate = 1
yes
no
Month + 1
Day + 1
Month = 13?
no
yes
Year + 1
Month = 1
UpDate
End

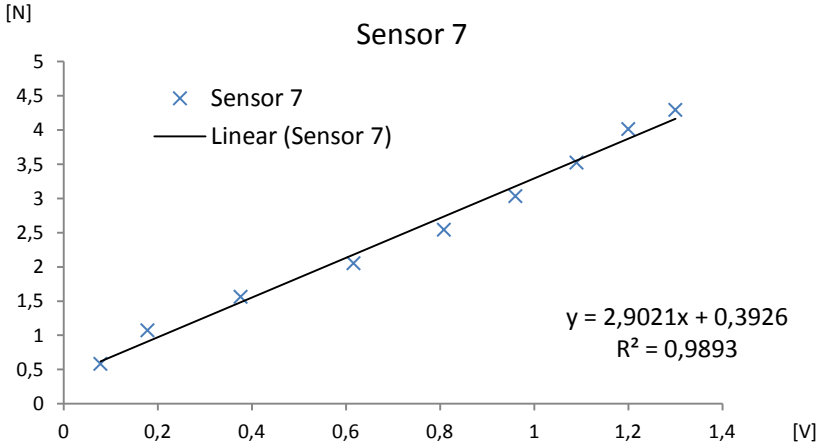
APPENDIX G. Calibration equations and charts

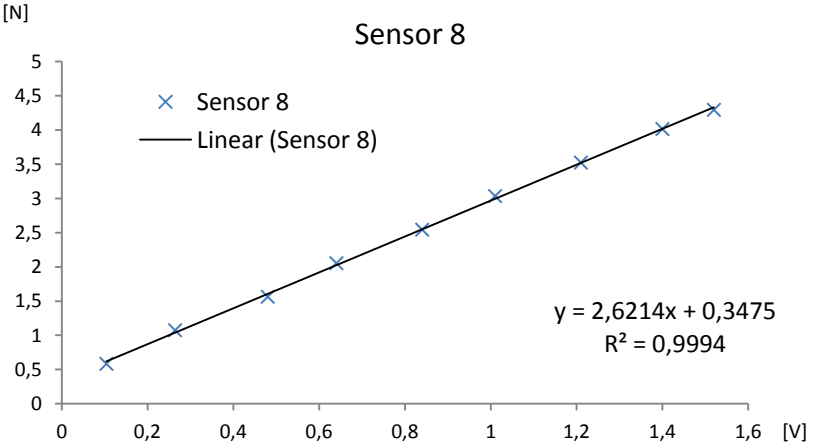
Following are presented the data used to on each sensor's calibration. The charts represent Force *versus* Voltage data from the sensor measured on the output of the conditioning circuit, likewise the coefficient of correlation (R^2) and the calibration equation. As previously described, it is noticeable that the Sensor 21 presented the lowest R^2 due to the shape of the curve.

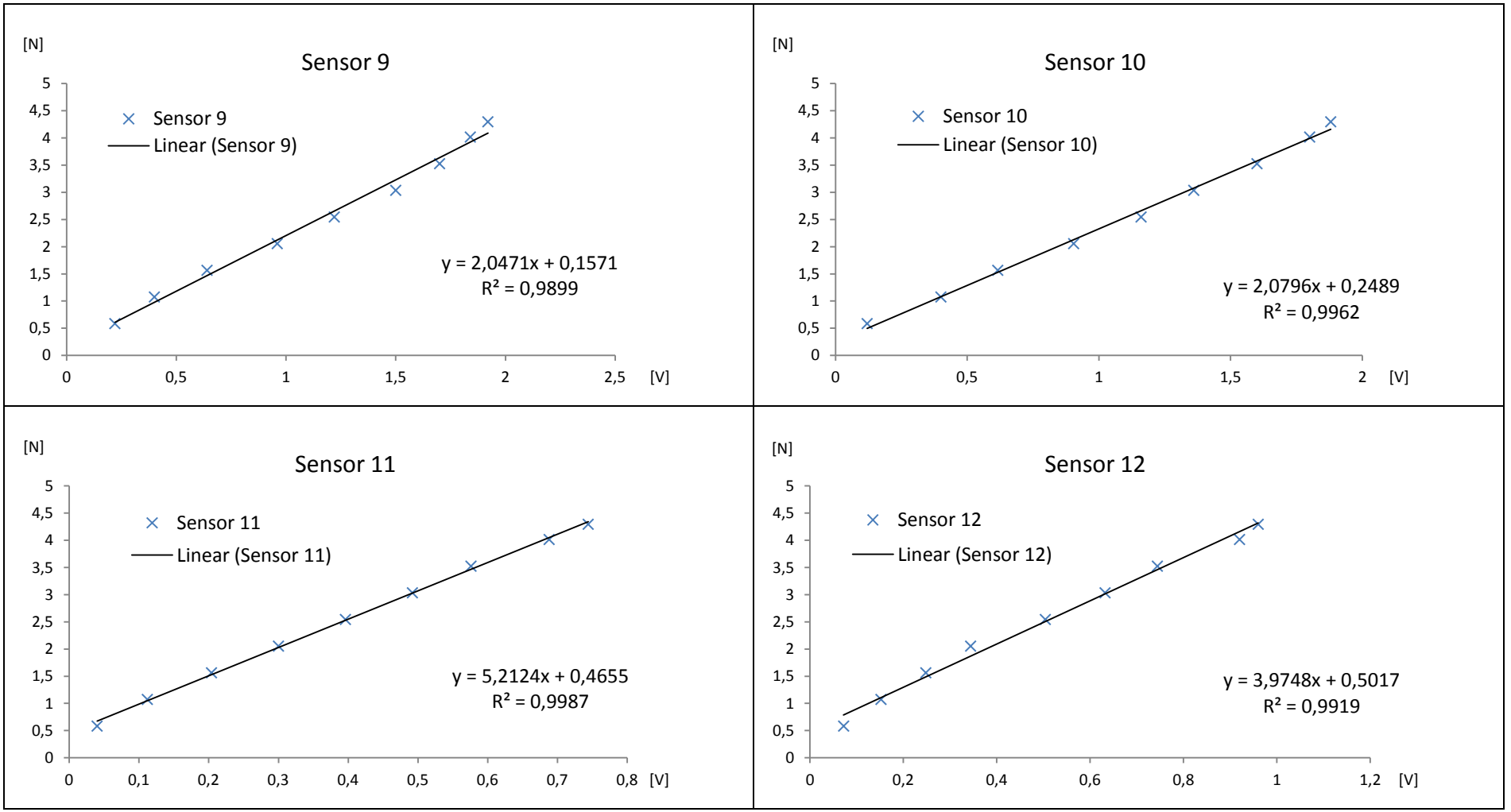


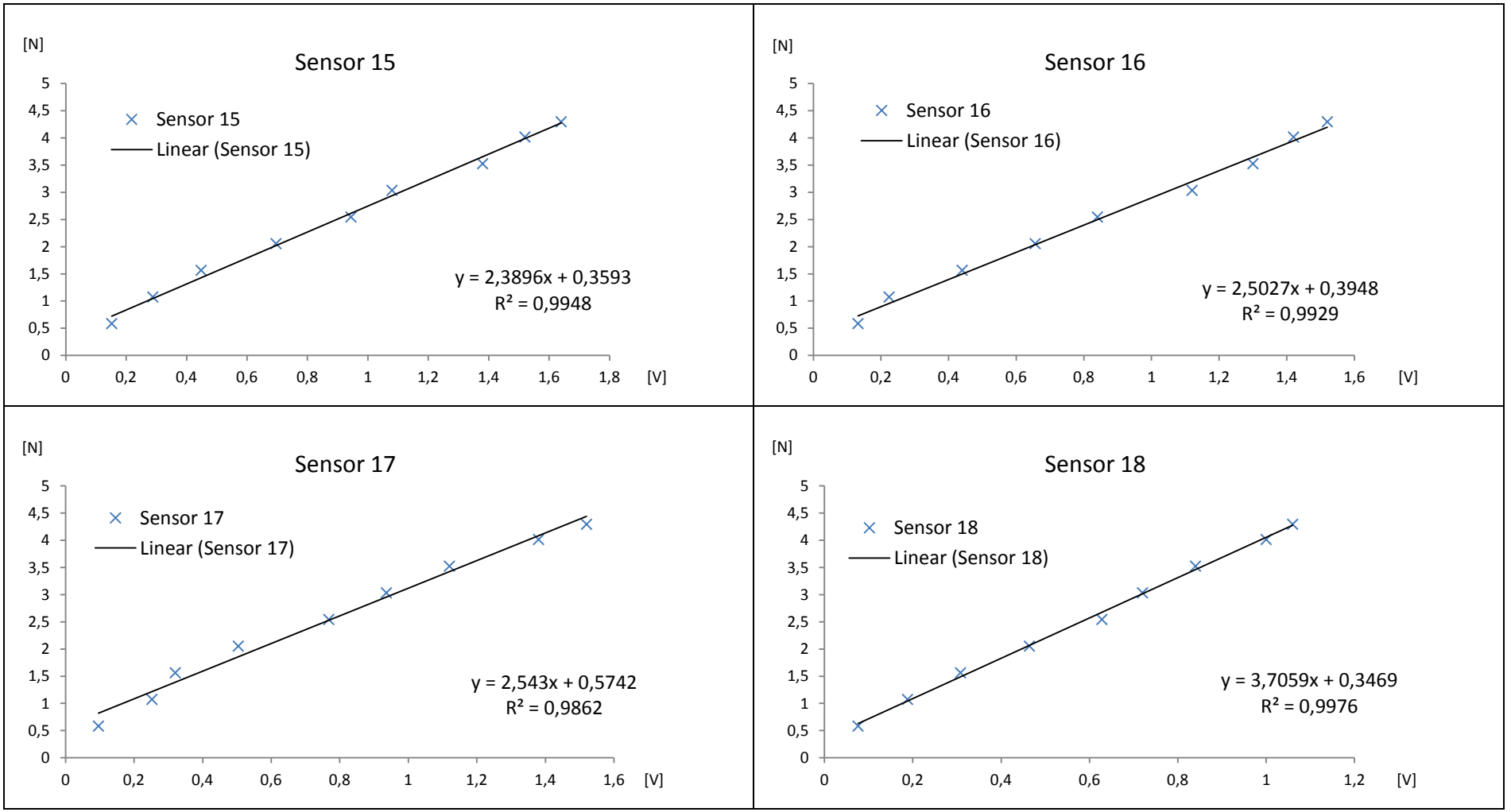

[N]
Sensor 5
Sensor 5
Linear (Sensor 5)
y = 2,4783x + 0,1044
R² = 0,9888
[V]

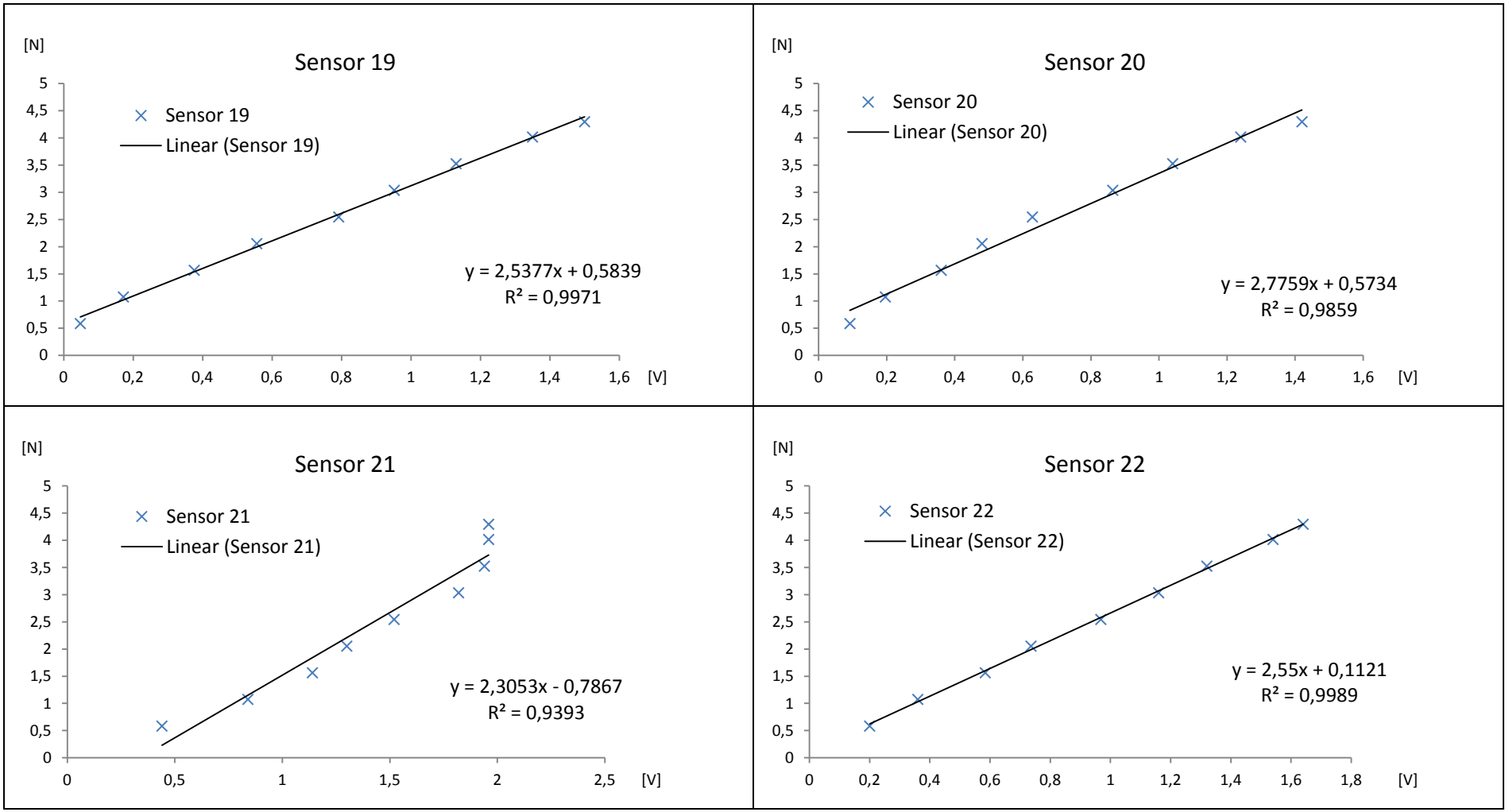

[N]
Sensor 6
Sensor 6
Linear (Sensor 6)
y = 2,5768x + 0,2185
R² = 0,9969
[V]

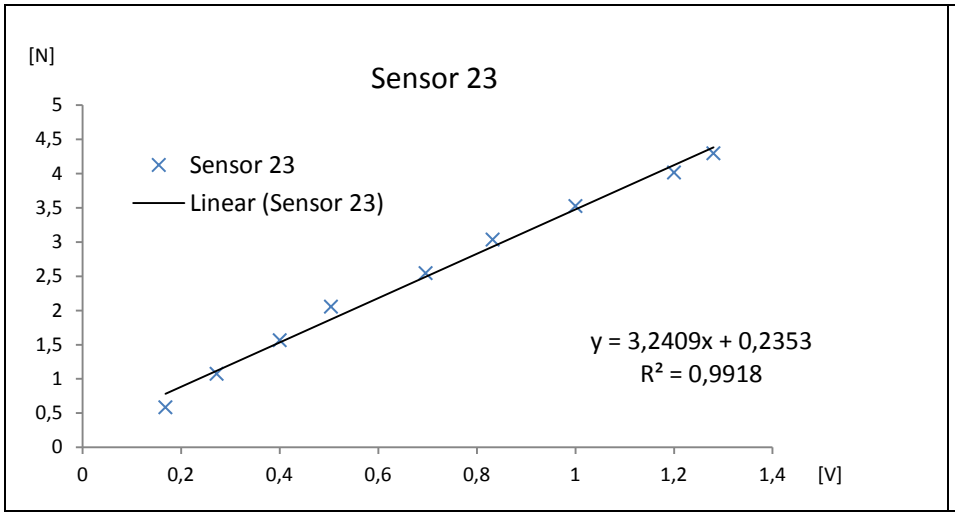

[N]
Sensor 7
Sensor 7
Linear (Sensor 7)
y = 2,9021x + 0,3926
R² = 0,9893
[V]

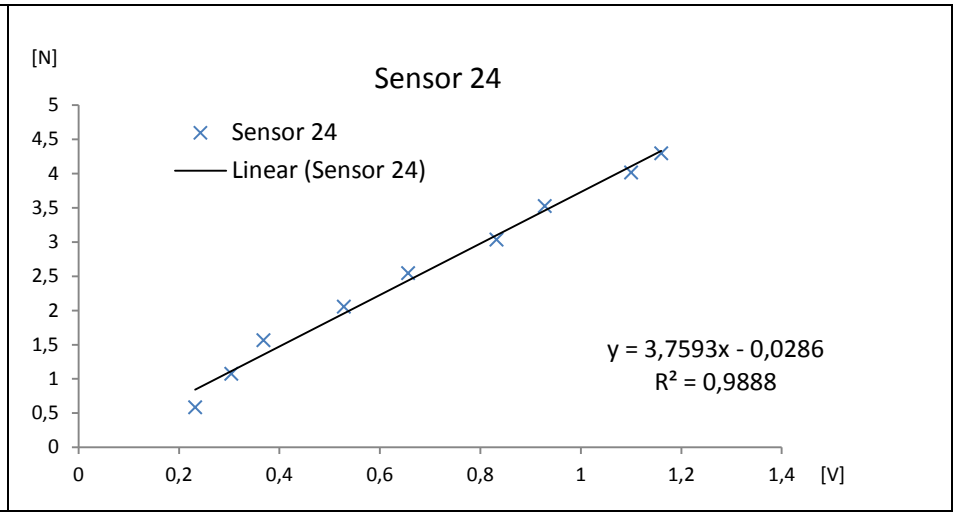

[N]
Sensor 8
Sensor 8
Linear (Sensor 8)
y = 2,6214x + 0,3475
R² = 0,9994
[V]


[N]
Sensor 9
Sensor 9
Linear (Sensor 9)
y = 2,0471x + 0,1571
R² = 0,9899
[V]
[N]
Sensor 10
Sensor 10
Linear (Sensor 10)
y = 2,0796x + 0,2489
R² = 0,9962
[V]
[N]
Sensor 11
Sensor 11
Linear (Sensor 11)
y = 5,2124x + 0,4655
R² = 0,9987
[V]
[N]
Sensor 12
Sensor 12
Linear (Sensor 12)
y = 3,9748x + 0,5017
R² = 0,9919
[V]


[N]
Sensor 15
Sensor 15
Linear (Sensor 15)
y = 2,3896x + 0,3593
R² = 0,9948
[V]
[N]
Sensor 16
Sensor 16
Linear (Sensor 16)
y = 2,5027x + 0,3948
R² = 0,9929
[V]
[N]
Sensor 17
Sensor 17
Linear (Sensor 17)
y = 2,543x + 0,5742
R² = 0,9862
[V]
[N]
Sensor 18
Sensor 18
Linear (Sensor 18)
y = 3,7059x + 0,3469
R² = 0,9976
[V]


Sensor 19
[N]
Sensor 19
Linear (Sensor 19)
y = 2,5377x + 0,5839
R² = 0,9971
[V]
Sensor 20
[N]
Sensor 20
Linear (Sensor 20)
y = 2,7759x + 0,5734
R² = 0,9859
[V]
Sensor 21
[N]
Sensor 21
Linear (Sensor 21)
y = 2,3053x - 0,7867
R² = 0,9393
[V]
Sensor 22
[N]
Sensor 22
Linear (Sensor 22)
y = 2,55x + 0,1121
R² = 0,9989
[V]


[N]
Sensor 23
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
0,2
0,4
0,6
0,8
1
1,2
1,4
[V]
Sensor 23
Linear (Sensor 23)
y = 3,2409x + 0,2353
R² = 0,9918


[N]
Sensor 24
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
0,2
0,4
0,6
0,8
1
1,2
1,4
[V]
Sensor 24
Linear (Sensor 24)
y = 3,7593x - 0,0286
R² = 0,9888

APPENDIX H. Data frame field description

The frame is composed by 34 fields where each one is described below.

Field	Description
#	Indicates the frame begins.
HH:MM:SS:MMM	Time information composed by Hour, Minute, Second and Millisecond
24 SENSORS	Composed by 24 separated field with information about force sensors in Newton
PITCH	Pitch angle information in degree.
ROLL	Roll angle information in degree.
ACCELX	Instantaneous acceleration about x axis.
ACCELY	Instantaneous acceleration about y axis.
ACCELZ	Instantaneous acceleration about z axis.
COPX	Center of Pressure position about x axis
COPY	Center of Pressure position about y axis
COP	Center of Pressure absolute force value.

APPENDIX I. Center of Pressure firmware code

```
319 void getCoP(void)
320 {
321     int i = 0;
322     int j = 0;
323 
324         |
325     for (i=0;i<N_SENSORS;i++)
326     {
327         j = uipSensor[i];
328         //calculate the contribution of each sensor for the CoP vector position
329         if (i<12) //rear contribution
330         {
331             dResultantRearX += dForce[j]*(double)(sensor[i][0]);
332             dResultantRearY += dForce[j]*(double)(sensor[i][1]);
333             dResultantRear  += dForce[j]; //calculate individual matrix resultant force
334         }
335         else //front contribution
336         {
337             dFrontX += dForce[j]*(double)((sensor[i-12][0])+52);
338             dResultantFrontY += dForce[j]*(double)(sensor[i-12][1]);
339             dResultantFront  += dForce[j]; //calculate individual matrix resultant force
340         }
341     }
342 
```

```
343
344     //calculate total resultant force
345     dResultantTotal = dResultantRear + dResultantFront;
346
347     if (dResultantRear > 0) {
348         dResultantRearX  /= dResultantRear;
349         dResultantRearY  /= dResultantRear;
350     } else {
351         dResultantRearX  = 0;
352         dResultantRearY  = 0;
353     }
354
355     if (dResultantFront > 0){
356         dFrontX /= dResultantFront;
357         dResultantFrontY /= dResultantFront;
358     } else {
359         dFrontX = 0;
360         dResultantFrontY = 0;
361     }
363
364     if (dResultantTotal > 0){
365         //calculate CoP positioning from individual contribution of both matrices
366         dResultantTotalX = ((dFrontX*dResultantFront) + (dResultantRearX*dResultantRear))/dResultantTotal;
367         dResultantTotalY = ((dResultantFrontY*dResultantFront) + (dResultantRearY*dResultantRear))/dResultantTotal;
368
369         dResultantRear = 0;
370         dResultantFront = 0;
371         dResultantRearX  = 0;
372         dResultantRearY  = 0;
373         dFrontX = 0;
374         dResultantFrontY = 0;
375     }
376     else{
377         dResultantTotalX  = 45;
378         dResultantTotalY  = 10;
379         dResultantTotal = 0;
380     }
381
382
383
384 }
```

APPENDIX J. Matlab code

```
       clear all; %clean workspace & variables
       clc;

       %--- system initialization ------------------------------------------------%
fclose(instrfind);                                      %close serial channel if
open
s=serial('COM8','BaudRate',115200);                     %open serial channel
%WiFi=tcpip('169.254.1.1',2000);                        %open WiFi channel
%fopen(WiFi);                                           %open file WiFi
fopen(s);                                               %open serial file
file = fopen('C:\Users\Zimba\Desktop\Teste_BOSU_4_calibrado\balance1.txt', 'w');
%open local file to record frame data
colormap(jet);                                          %color map pattern for
charts
%-------------------------------------------------------------------------------%
% ----auxiliar variables ---------------------------------------------------%
linhas               = 0;      %number of lines
n_linhas             = 400;    %number of frames to receive
n_divisoes           = 100;    %aux to average
n_linhas_anterior    = 0;      %aux to record previous number of lines
n_medias             = 1;
Gx=0; Gy=0;                    %aux to position plot

%-------------------------------------------------------------------------------%

% --- primary  primárias- ----------------------------------------------------%
%store the average value for both platforms
sensor_media_frente = cell(1,n_divisoes);
sensor_media_tras = cell(1,n_divisoes);
%store the sum of the values for front platform
sensor_frente = zeros(7,7);
%store the sum of the values for rear platform
sensor_tras = zeros(7,7);

sensor_media_total_frente = zeros(7,7);
sensor_media_total_tras = zeros(7,7);
%matrix to store sensor value according to its position
MD_frente = zeros(7,7);
MD_tras = zeros(7,7);
%positioning variables
pitch                = 0;
roll                 = 0;
%flag to determinate which charts to be plot
plot_CoP = 1;
plot_decks =1;
plot_posicao = 0;
plot_media_parcial = 0;
%variables to store de CoP result
xr_total = 0;
yr_total = 0;
resultante_total = 0;
%accelxanterior = 0;
%vel = 0;
%-------------------------------------------------------------------------------%

keyboard %wait for the command to run (return)

while 1 %main loop
```

```
    frame = fscanf(s);   %read frame from file
    tamanho_pacote = size(frame); %define frame size
    %only execute the next code if data frame is complete
    if (tamanho_pacote(2)>=200)
        if(linhas<n_linhas)             %check numbe of received lines
            if(frame~=0)
                inicioSTR = frame(1,1);  %store the frame start char
                if(inicioSTR=='#')
                    dado = idstrip(frame);  %strip number data from the file

                    linhas = linhas + 1;

                    % --- divide the frame received into local variables  -----%

                    %sensor 1
                        MD_tras(4,1) = dado(1,3);
                     %   sensor_tras(6,3) = sensor_tras(6,3)+ MD_tras(6,3);
                    %sensor 2
                        MD_tras(3,1) = dado(1,4);
                     %   sensor_tras(6,4) = sensor_tras(6,4)+ MD_tras(6,4);
                    %sensor 3
                        MD_tras(5,2) = dado(1,6);
                     %   sensor_tras(5,2) = sensor_tras(5,2)+MD_tras(5,2);
                    %sensor 4
                        MD_tras(2,2) = dado(1,5);
                     %   sensor_tras(5,5) = sensor_tras(5,5)+ MD_tras(5,5);
                    %sensor 5
                        MD_tras(4,3) = dado(1,18);
                     %   sensor_tras(4,3) = sensor_tras(4,3)+ MD_tras(4,3);
                    %sensor 6
                        MD_tras(3,3) = dado(1,7);
                     %   sensor_tras(4,4) = sensor_tras(4,4)+ MD_tras(4,4);
                    %sensor 7
                        MD_tras(5,4) = dado(1,16);
                     %   sensor_tras(3,2) = sensor_tras(3,2)+ MD_tras(3,2);
                    %sensor 8
                        MD_tras(2,4) = dado(1,9);
                     %   sensor_tras(3,5) = sensor_tras(3,5)+ MD_tras(3,5);
                    %sensor 9
                        MD_tras(4,5) = dado(1,17);
                     %   sensor_tras(2,3) = sensor_tras(2,3)+ MD_tras(2,3);
                    %sensor 10
                        MD_tras(3,5) = dado(1,8);
                     %   sensor_tras(2,4) = sensor_tras(2,4)+ MD_tras(2,4);
                    %sensor 11
                        MD_tras(5,6) = dado(1,15);
                     %   sensor_tras(1,2) = sensor_tras(1,2)+ MD_tras(1,2);
                    %sensor 12
                        MD_tras(2,6) = dado(1,10);
                     %   sensor_tras(1,5) = sensor_tras(1,5)+ MD_tras(1,5);

                    %sensor 13
                        MD_frente(4,1) = dado(1,25);
                     %   sensor_frente(6,3) = sensor_frente(6,3)+ MD_frente(6,3);
                    %sensor 14
                        MD_frente(3,1) = dado(1,24);
                     %   sensor_frente(6,4) = sensor_frente(6,4)+ MD_frente(6,4);
                    %sensor 15
                        MD_frente(5,2) = dado(1,23);
                     %   sensor_frente(5,2) = sensor_frente(5,2)+ MD_frente(5,2);
                    %sensor 16
                        MD_frente(2,2) = dado(1,26);
                    %    sensor_frente(5,5) = sensor_frente(5,5)+ MD_frente(5,5);
                    %sensor 17
                        MD_frente(4,3) = dado(1,21);
                     %   sensor_frente(4,3) = sensor_frente(4,3)+ MD_frente(4,3);
                    %sensor 18
                        MD_frente(3,3) = dado(1,14);
```

```
                    %   sensor_frente(4,4) = sensor_frente(4,4)+ MD_frente(4,4);
                   %sensor 19
                       MD_frente(5,4) = dado(1,20);
                    %   sensor_frente(3,2) = sensor_frente(3,2)+ MD_frente(3,2);
                   %sensor 20
                       MD_frente(2,4) = dado(1,11);
                   % sensor_frente(3,5) = sensor_frente(3,5)+MD_frente(3,5);
                   %sensor 21
                       MD_frente(4,5) = dado(1,19);
                    %   sensor_frente(2,3) = sensor_frente(2,3)+ MD_frente(2,3);
                   %sensor 22
                       MD_frente(3,5) = dado(1,12);
                    %   sensor_frente(2,4) = sensor_frente(2,4)+ MD_frente(2,4);
                   %sensor 23
                       MD_frente(5,6) = dado(1,22);
                    %   sensor_frente(1,2) = sensor_frente(1,2)+ MD_frente(1,2);
                   %sensor 24
                       MD_frente(2,6) = dado(1,13);
                    %   sensor_frente(1,5) = sensor_frente(1,5)+ MD_frente(1,5);

%                  %data sensor interpolation
%                       ND_tras = interp2(MD_tras,4);
%                       ND_frente = interp2(MD_frente,4);
                   %-------------------------------------------------------------%
                   % --- position ------------------------------------------ ----%
                   %pitch - eixo y
                   pitch = dado(27) % [°]
                   %roll eixo x
                   roll = dado(28) % [°]
                   %aceleração no eixo x
                   accelX = dado(29); %[m/s2]
                   %aceleração no eixo y
                   accelY = dado(30); %[m/s2]
                   %aceleração no eixo z
                   accelZ = dado(31); %[m/s2]
                   %-------------------------------------------------------------%
                   % --- CoP ----------------------------------------------------%
                   xr_total = dado(32);
                   yr_total = dado(33);
                   resultante_total = dado(34);

                   %Centro de Pressão
                   P1 = [xr_total, yr_total, 0];
                   P2 = [xr_total, yr_total, resultante_total];
                   pts = [P1; P2];
                %-------------------------------------------------------------%
                % --- data plot ----------------------------------------------%
                   %plot 3D for CoP
                   if(plot_CoP)
                       figure(1);
                       subplot(1,3,1);
                       plot3(pts(:,2),pts(:,1),pts(:,3),'-
^','MarkerFaceColor','b','Linewidth',2);
                       axis([0 19 18 90 0 60]);
                       view(54,-32);
                       xlabel('x-axis [cm]');
                       ylabel('y-axis [cm]');
                       zlabel('CoP [N]')
                       grid;
                   end
                   %plot 2D for each platform
                   if(plot_decks)
                       subplot(1,3,2);
                       pcolor(MD_tras)
                       %title('rear foot');
                       caxis([0 15]);
                      % shading('interp');
                       subplot(1,3,3);
```

```
                        pcolor(MD_frente)
                        %title('front foot');
                        caxis([0 15]);
                       % shading('interp');
                        %drawnow;
                    end
                    %plot 2D XY for position
                    if(plot_posicao)
                        subplot(1,5,4);
                        Gx = [Gx, roll];
                        plot(Gx);
                        ylim([-180, 180])
                        title('roll');
                               subplot(1,5,5);
                        Gy = [Gy, pitch];
                        plot(Gy);
                        ylim([-180, 180])
                        title('pitch');
                    end
                %-------------------------------------------------------------------%
                end
            end

            %save frame data into a local file
            n = size(frame);
            for i = 1:(n(2)+1)
                if(i<n(2)+1)
                    if(frame(1,i) == ' ')
                        Z(1,i) = ',';
                    else
                        Z(1,i) = frame(1,i);
                    end
                else
                    xr = num2str(xr_total);
                    yr = num2str(yr_total);
                    r = num2str(resultante_total);
                    a = horzcat(xr, ',', yr, ',', r,',');
                    Z = strcat(a,Z);
                    %write data into a .txt file
                    fprintf(file , '%s\r\n', Z);
                    Z=0;
                end
            end
        else
            break; %stops when the total number of lines is achieved
        end
    end
end
```