3D Indoor Mapping

A Real-Time Wireless LIDAR Scanner

Carlos Jorge Vieira da Silva

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

September 4, 2020

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Electrotécnica e de Computadores

Candidato: Carlos Jorge Vieira da Silva, Nº 1160628, 1160628@isep.ipp.pt
Orientação científica: Prof. Doutor Carlos Vinhais, cav@isep.ipp.pt

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

September 4, 2020

"You, the people have the power - the power to create machines.
The power to create happiness!
You, the people, have the power to make this life free and beautiful,
to make this life a wonderful adventure."

Transcript of Charlie Chaplin's Final Speech in The Great Dictator

Agradecimentos

Quero agradecer imenso ao meu orientador, Prof. Doutor Carlos A. Vinhais, por todas as horas de atenção, ajuda, paciência e disponibilidade, especialmente em tempo de pandemia. Por me ter inspirado e motivado a explorar ferramentas científicas que hoje penso fazerem parte do meu futuro, nomeadamente o LaTeX, o Python, o VTK e o OpenCV. E por partilhar comigo momentos de felicidade quando algum objetivo do projeto era cumprido.

Agradecer especialmente à minha mãe, por nunca ter deixado de estar presente na minha vida, sei que estás sempre comigo.

Um obrigado enorme à minha namorada, Susana Marreiros, por todo o apoio, preocupação e compreensão, como também pelo aconselhamento científico que me ajudou a melhorar o projeto.

Agradecer também à minha família, com nuance especial para o meu tio Manuel Vieira por ter facultado a placa de acrílico que suporta o projeto, e aos meus amigos e comunidade do ISEP, por todos os momentos de boa disposição e partilha de conhecimento.

A todos os mencionados e não mencionados, obrigado por de alguma forma terem contribuído para a minha formação como pessoa e estudante de Engenharia.

Acknowledgements

I would like to thank my advisor Carlos A. Vinhais, PhD, for all the hours of care, help, patience and availability, especially in times of pandemic. For having me inspired and motivated to explore scientific tools that today I think will be part of my future, namely LaTeX, Python, VTK and OpenCV. And for sharing with me the all moments of happiness when any objective of the project was fulfilled.

Special thanks to my mother, for being always present in the my life, I know that you are always with me.

A huge thank you to my girlfriend, Susana Marreiros, for all the support, concerns, understanding, and all the scientific advice that helped me to improve the project.

Also thank my family, with a special nuance to my uncle Manuel Vieira for providing the acrylic plate that supports the project, and to my friends and ISEP community, for all the moments of happiness and knowledge sharing.

To all those mentioned and not mentioned, thank you for somehow having contributed to me growing as a person and Engineering student.

Abstract

3D scanners are increasingly impacting life in today's world. With the increase of computational resources, areas such as 3D printing, Architecture, Robotics, Entertainment, among others, are making an increased use of this technology, in order to obtain considerably better results.

In Architecture, the survey of the construction site is repeatedly based on drawings. This practice is time-consuming and complex, as it is necessary to collect all the measurements of the construction site with the support of a measuring tape or LASER distance meter. Later it will still be necessary to recreate all measurements in a software. 3D scanning simplifies these processes, getting a survey of the site through cloud points, which can be used by software linked to Architecture.

One of the main reasons why 3D scanners are not yet used broad is the high cost per unit, being able to reach, several tens of thousands of euros. What is considered to be a high investment by a large part of companies and individuals.

Within the scope of this project, a low-cost wireless 3D scanner was developed, as well as a graphical user interface capable of controlling it, and visualize the resulting point cloud. The construction of the implemented 3D scanner is divided into two parts: a rotating platform with two degrees of freedom and an electronic power, control and communication board. The device was used to acquire information of a room, in a fully automated way, in which point clouds were obtained. Based on the work developed and the obtained results, it is possible to claim that a functional basis was created for the scanning of indoor structures, ready for future development, proving that it is possible to obtain alternatives to commercial methods using more limited financial resources.

Keywords

3D Scanner, Point Cloud, Microcontroller, LIDAR, Stepper Motor, Python, Wireless, Automation and Control, 3D LIDAR Scanning.

Resumo

Os *scanners* 3D estão a ter cada vez mais impacto no quotidiano do mundo atual. Com o aumento dos recursos computacionais, áreas como a Impressão 3D, a Arquitetura, a Robótica e o Entertenimento, entre outras, estão a fazer um uso crescente destas tecnologias de forma a obter melhores resultados.

Na Arquitetura, o levantamento de obra é recorrentemente feito apartir de desenhos. Esta prática é demorada e complexa, pois é necessário recolher todas as medidas da obra com o apoio de fita métrica ou medidor de distâncias LASER. Sendo posteriormente ainda necessário recriar todas as medidas em software. O *scanning* 3D simplifica estes processos, conseguindo um levantamento do local através de pontos, que poderão ser aproveitados por *software* ligado à Arquitetura, como por exemplo o AutoCAD.

Um dos principais motivos pelos quais os *scanners* 3D ainda não são utilizados de forma ainda mais ampla é o elevado custo por unidade, podendo atingir, as várias dezenas de milhares de euros. O que é considerado um alto investimento por grande parte das empresas e particulares.

No âmbito deste projeto, desenvolveu-se um *scanner* 3D, *low-cost* e de transmissão sem fios, como também uma interface gráfica capaz de controlar com o mesmo e possibilitar a visualização da resultante nuvem de pontos. A constituição do *scanner* implementado está dividida em duas partes: uma plataforma rotativa com dois graus de liberdade e uma placa eletrónica de alimentação, controlo e comunicação. Este dispositivo foi utilizado para adquirir informação tri-dimensional de uma sala, de forma totalmente automatizada, no qual se obtiveram nuvens de pontos. Com base no trabalho desenvolvido e nos resultados obtidos, é possível afirmar que foi criada uma base funcional para o varrimento de estruturas interiores, apta para desenvolvimento futuro, provando que é possível obter alternativas aos métodos comerciais utilizando recursos financeiros mais limitados.

Palavras-Chave

Scanner 3D, Nuvem de Pontos, Microcontrolador, LIDAR, Motor de Passo, Python, Sem-fios, Automação e Controlo, *Scanning 3D* com LIDAR

Contents

Acknowledgments vii

Abstract ix

Resumo xi

Contents xiii

List of Figures xvii

List of Tables xxi

List of Listings xxiii

Acronyms xxv

1. Introduction 1
- 1.1 Contextualization 1
- 1.2 Goals and Motivation 2
- 1.3 Timing 3
- 1.4 Contributions 3
- 1.5 Report Organization 4

2. 3D Scanners 5
- 2.1 Fundamentals 5
- 2.2 Contact Methods 6
- 2.3 Contactless Methods 7
  - 2.3.1 Stereoscopic Scanners 7
  - 2.3.2 Rotational Scanners 9
  - 2.3.3 Commercial Scanners 10

3. Automation and Control 11
- 3.1 Microcontrollers 11
  - 3.1.1 ESP32 Microcontroller 11
  - 3.1.2 Programming Environment 13
- 3.2 IMU 16

3.2.1 Working Principles . . . 16
3.2.2 BNO055 IMU . . . 18
3.2.3 I2C Protocol . . . 22
3.2.4 Developed IMU Sensor Library . . . 25
3.2.5 Experimental Results . . . 26
3.3 LIDAR . . . 29
3.3.1 Working Principles . . . 29
3.3.2 TFmini-S LIDAR . . . 31
3.3.3 USART Protocol . . . 37
3.3.4 Developed TFmini-S Sensor Library . . . 39
3.3.5 Experimental Results . . . 40
3.4 Motion Control . . . 43
3.4.1 Working Principles . . . 43
3.4.2 NEMA Stepper Motor . . . 44
3.4.3 DRV8825 Stepper Motor Driver . . . 45
3.4.4 Developed DRV8826 Stepper Motor Driver Library . . . 49

4. Hardware Implementation . . . 51
4.1 Breadboard . . . 51
4.2 Mechanical Structure . . . 52
4.3 Power Considerations . . . 53
4.4 Printed Circuit Board . . . 56
4.4.1 PCB Design using KiCad . . . 56
4.4.2 Printing the PCB . . . 57
4.5 Financial Analysis . . . 59

5. Software and User Interface . . . 61
5.1 Programming Environment . . . 61
5.2 Data Communication . . . 62
5.3 Functions and Threads . . . 64
5.4 Point Cloud Visualization . . . 64
5.5 Graphical User Interface . . . 65
5.6 3D Scanner Interface . . . 65
5.7 Executable Python File . . . 69

6. Point Clouds . . . 71
6.1 3D Scanning . . . 71
6.1.1 Data Acquisition . . . 74
6.1.2 Data Interpretation and Storage . . . 75
6.2 Experimental Results . . . 77
6.3 Data Post-Processing . . . 82
6.3.1 Point Cloud Visualization . . . 82
6.3.2 Area and Perimeter . . . 86

7. Conclusion . . . 91

Bibliography . . . 95

A. ESP32 Data 97

B. Micro Python 99

C. TFmini-S Tests 107

D. Board Design 109

List of Figures

1.1 Timing of Project Execution . . . 3

2.1 Taxonomy of Active Range Acquisition Methods . . . 5
2.2 DS10 Contact Scanner . . . 6
2.3 OpenCV Test: Chessboard . . . 8
2.4 OpenCV Test: Depth Image . . . 8
2.5 Sample of 360° LIDAR Data . . . 9
2.6 RPLidar A1 . . . 9
2.7 Leica RTC360 . . . 10

3.1 ESP32 Microcontroller . . . 12
3.2 Arduino IDE: General View . . . 14
3.3 Arduino IDE: Additional Boards Manager . . . 15
3.4 Arduino IDE: Install board . . . 15
3.5 Euler Angles . . . 17
3.6 BNO055 IMU . . . 18
3.7 BNO055 Architecture . . . 20
3.8 BNO055 Fusion Modes . . . 20
3.9 BNO055 I2C Timming Diagram . . . 22
3.10 BNO055 I2C Write Example . . . 23
3.11 BNO055 I2C Read Example . . . 24
3.12 Accelerometer . . . 27
3.13 Gyroscope . . . 27
3.14 Magnetometer . . . 28
3.15 Absolute Orientation . . . 28
3.16 Scanning LIDAR . . . 30
3.17 Detector Arrays . . . 30
3.18 Detector Arrays Modulation . . . 30
3.19 TFmini-S LIDAR . . . 32
3.20 TFmini-S Object Detection . . . 33
3.21 TFmini-S Middle Distance . . . 34
3.22 TFmini-S Accuracy . . . 35
3.23 TFmini-S Length of Effective Detection . . . 35
3.24 TFmini-S Light Spots . . . 35
3.25 TFmini-S Pinout . . . 36
3.26 TFmini-S USART Data Format . . . 37

3.27 TFmini-S USART Communication Frame Definitions . . . 38
3.28 TFmini-S Testing . . . 42
3.29 Stepper Motor Types . . . 43
3.30 Stepper Motor and Driver . . . 44
3.31 DRV8825 Basic Schematics . . . 45
3.32 DRV8825 Micro stepping Waveform . . . 47
3.33 DRV8825 Current Regulation . . . 48
3.34 DRV8825 Setup External Power Supply . . . 49

4.1 Breadboard Circuit . . . 51
4.2 Mechanical Structure . . . 52
4.3 AMS1117 Voltage Regulator . . . 54
4.4 PCB Boards . . . 58
4.5 Final Soldered Board . . . 58
4.6 Financial Analysis: Comparison by Absolute Cost . . . 60
4.7 Financial Analysis: Comparison by Relative Cost . . . 60

5.1 Server Client Model . . . 63
5.2 Sockets Protocol . . . 63
5.3 VTK Test: Conceptual overview of pipeline execution . . . 64
5.4 Qt Logo . . . 65
5.5 Graphical User Interface: General View . . . 66
5.6 Graphical User Interface : Communication . . . 67
5.7 Graphical User Interface: IMU . . . 68
5.8 Graphical User Interface: Motors Step Resolution . . . 68
5.9 Graphical User Interface: Start Scanning . . . 69
5.10 Software Installer . . . 70

6.1 3D Scanner Complete . . . 72
6.2 Simplified Flow Chart . . . 73
6.3 3D Scanning Results (Rendering Window) . . . 79
6.4 3D Scanning Results (Top View) . . . 80
6.5 3D Scanning Results (Inner View) . . . 81
6.6 MeshLab Environment . . . 82
6.7 ParaView Environment . . . 84
6.8 AutoCad Environment . . . 85
6.9 Riemann Area: Comparison through Rectangles . . . 88
6.10 Riemann Area: Comparison through Semi-Circles . . . 88
6.11 Scanned Blueprints . . . 89

B.1 MicroPython: List USB connections . . . 99
B.2 MicroPython: Check last used USB Ports . . . 100
B.3 MicroPython: Erase the flash memory . . . 100
B.4 MicroPython: Write in the flash memory . . . 100
B.5 MicroPython: Python prompt . . . 101
B.6 MicroPython: List directory . . . 101
B.7 MicroPython: Soft reboot . . . 101
B.8 MicroPython: General help . . . 102

B.9 MicroPython: Help on a module 102
B.10 MicroPython: Change operating frequency 103
B.11 MicroPython: Toggle pins 103
B.12 MicroPython: RSHELL tool 103
B.13 MicroPython: List available boards 104

D.1 KiCad Step 1: Design the Schematics 109
D.2 KiCad Step 2: Annotate the Schematics 110
D.3 KiCad Step 3: Verify the Schematics Electrical Connections 110
D.4 KiCad Step 4: Assign Footprints 111
D.5 KiCad Step 5: Generate a netlist 111
D.6 KiCad Step 6: Board size and organization 112
D.7 KiCad Step 7: Add tracks 112
D.8 KiCad Step 8: Verify the Layout Electrical Connections 113
D.9 Board 3D Top View 113
D.10 Board Layout View 115
D.11 Board 3D Views 116

List of Tables

3.1 ESP32 Specifications . . . 12
3.2 BNO055 Specifications . . . 19
3.3 Comparison of Distance Sensors Technology . . . 31
3.4 TFmini-S Specifications . . . 32
3.5 TFmini-S Minimum Side Length . . . 36
3.6 TFmini-S USART Configuration . . . 36
3.7 TFmini-S Pinout . . . 36
3.8 TFmini-S Abnormal Data . . . 37
3.9 TFmini-S Example: Set distance data to mm . . . 39
3.10 TFmini-S Linear Regression Results . . . 41
3.11 17HS4023 Specifications Table . . . 44
3.12 DRV8825 Specifications . . . 46
3.13 DRV8825 Steps Modes . . . 46

4.1 Weight of the Mechanical Structure . . . 52
4.2 Power Consumption at 3.3 V Powered Devices . . . 53
4.3 Power Consumption at 5.0 V Powered Devices . . . 53
4.4 Power Consumption at 9.0 V Powered Devices . . . 53
4.5 AMS1117 Specifications Table . . . 55
4.6 Total Cost of the Scanner . . . 59

5.1 Programming Machine Specifications . . . 61

6.1 Data Frame . . . 73
6.2 Final Results . . . 78
6.3 Predicted Results . . . 78
6.4 Results on the approximation of area and perimeter . . . 89

A.1 BNO055 Output Data Registers . . . 97
A.2 BNO055 Sensor Default Settings . . . 97
A.3 BNO055 Output Data Representation . . . 97
A.4 BNO055 Output Data . . . 97

C.1 Cardboard box test . . . 107
C.2 Black box test . . . 108
C.3 Mirror test . . . 108

List of Listing

3.1 Sample of BNO055 I2C Write from File: BNO055.cpp . . . 23
3.2 Sample of I2C Read from File: BNO055.cpp . . . 25
3.3 File: BNO055.h . . . 25
3.4 Sample of UART Read from File: LIDAR.cpp . . . 38
3.5 Sample of UART Write from File: LIDAR.cpp . . . 39
3.6 File: LIDAR.h . . . 40
3.7 File: DRV8825.h . . . 49
6.1 File: FilterParaview.py . . . 83
6.2 File: AutoCadPoints.scr . . . 83
6.3 File: AutoCadLines.scr . . . 83
B.1 File: GPIORead.py . . . 104
B.2 File: PwmAdc.py . . . 104
B.3 File: PinInterrupt.py . . . 105
B.4 File: Timer.py . . . 105
B.5 File: UART.py . . . 105
B.6 File: RtcDeepSleep.py . . . 106

Acronyms

API	Application Programming Interface
COM	Communication Port
DAC	Digital to Analog Converter
DoF	Degrees of Freedom
ESR	Equivalent Series Resistance
FoV	Field of View
GPIO	General Purpose Input/Output
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
LADAR	LASER Detection And Ranging
LIDAR	Light Detection And Ranging
LVTTL	Low Voltage Transistor to Transistor Logic
MCU	Microcontroller Unit
OS	Operating System
PCB	Printed Circuit Board
PPS	Points Per Second
PWM	Pulse Width Modulation
RADAR	Radio Detection And Ranging
ROM	Read-Only Memory
SCL	Serial Clock Line
SDA	Serial Data Line
SiP	System in a Package
SMD	Surface Mounted Device
SoC	System on a Chip
SRAM	Static Random Access Memory
SONAR	Sound Navigation And Ranging
SSID	Service Set Identifier
TCP	Transmission Control Protocol
ToF	Time of Flight
UART	Universal Asynchronous Receiver-Transmitter
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF	Unicode Transformation Format
VTK	Visualization ToolKit

Chapter 1

Introduction

1.1 Contextualization

Lately, there's been an increase in the importance of three-dimensional modelling of architectural scenes from point cloud data generated by 3D scanners in areas like Virtual and Augmented Reality, Navigation, and Virtual Heritage Conservation. While traditional user-assisted methods are still on regular use, automatic methods for 3D interior modelling have received much interest in recent years. The modelling of interiors poses two important challenges, that are not found in buildings exteriors. A complexity challenge, as building interiors may contain several interconnected rooms and hall-ways, and a scalability challenge, given the prevalence of small-scale objects, like books and plates [1]. LIght Detection And Ranging (LIDAR) technology provides dense accurate 3D point clouds over reflective objects, from which building models can be derived. Although available auxiliary data such as building plan maps can assist this process, they may not always be available or up to date [2]. The reflectance strength of the returned signal depends on distance, incidence angle and material properties. This technology can be mounted onto scanning platforms such as Airborne Systems (air vehicles), Mobile Systems (ground vehicles) and Terrestrial Systems (stationary platform) [3]. The application of 3D survey techniques in the environment of Cultural Heritage helps the recognition, interpretation and characterization state of conservation and degradation of architectural structures. It also plays a crucial role in the monitoring of the restoration phase, and educational use on virtual and augmented reality environments[4]. On a bigger scale, 3D city models have also received much interest and became popular among urban planners and the telecommunication industry. Analysis of propagation of noise and air pollution through cities are some other potential applications of 3D city models [5].

1.2 Goals and Motivation

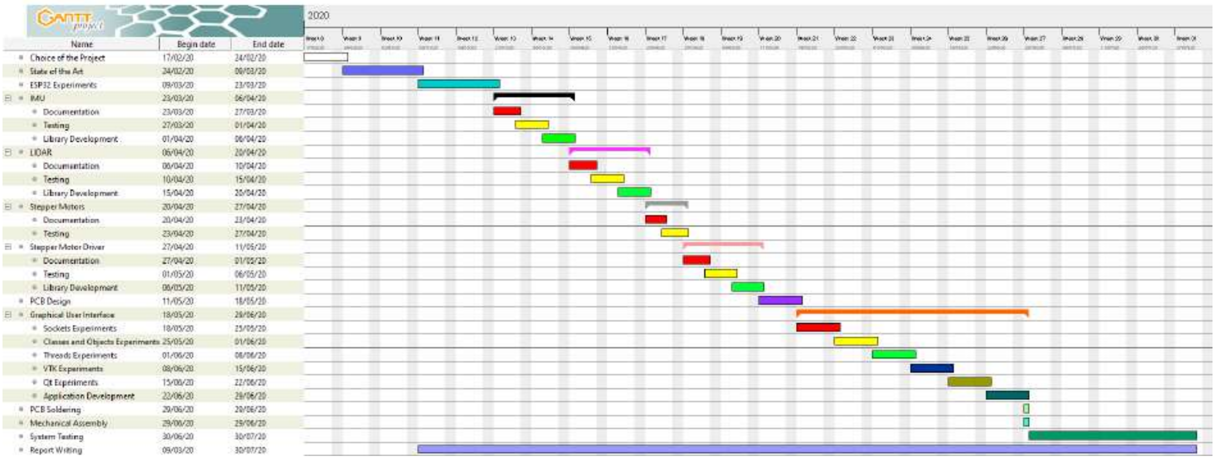
From the author's point of view, the most addictive thing you can give to an engineering student or an engineer is a project to do and things to learn in order to accomplish that goal. That's where all the motivation came from, the desire to learn new things and to build something that only an engineer can and have fun while at it. There was also some desire to create a PCB (Printed Circuit Board), to explore some tools available with Python, like OpenCV, VTK and Sockets, but also the ESP32, a cheap wireless-capable microcontroller. Since ISEP's DEE's "Jornadas Eletrotécnicas 2020", a curiosity about LIDAR and stereoscopic cameras started growing. The main proposal of this project came from Carlos A. Vinhais, PhD, within a range of options, which was accepted with great joy and expectations. The acquisition of 3D data and its visualization is still today's challenge due to its many limitations. On the market there are several devices ready for this purpose, however, they are quite expensive. The virtualization of the real world was already popular in videogames, virtual reality, entertainment and architecture. However, due to the lack of people's mobility, forced by the COVID-19 pandemic, virtual tourism and the virtualization of the real world grew an even larger interest in the current year. In "Chernobyl", a TV series based on April 1986's, explosion at the Chernobyl nuclear power plant in the Soviet Union (today's Ukraine), one of the challenges was to analyse the inner structure of the nuclear reactor that had exploded, but because of the radioactivity levels, no human could go inside of the reactor. This made the author realize that the 3D scanner should be wireless and have some kind of user interface.

Taking all of the above into consideration, the goal was set, and divided into two parts:

- Hardware
  - Capable of scanning indoors in 3D,
  - Efficient,
  - Mechanically simple,
  - Low-cost,
  - Transmit data wirelessly,
  - Light weight.
- Software
  - Control the hardware wirelessly,
  - Visualize 3D point clouds,
  - Export data ready for post-processing.

1.3 Timing

The timing of the project objectives along the semester is displayed in Fig. 1.1. The semester started at 17/02/2020 and the predicted completion is at 30/07/2020.



Name	Begin date	End date
Choice of the Project	17/02/20	24/02/20
State of the Art	24/02/20	09/03/20
ESP32 Experiments	09/03/20	23/03/20
IMU	23/03/20	06/04/20
Documentation	23/03/20	27/03/20
Testing	27/03/20	01/04/20
Library Development	01/04/20	06/04/20
LIDAR	06/04/20	20/04/20
Documentation	06/04/20	10/04/20
Testing	10/04/20	15/04/20
Library Development	15/04/20	20/04/20
Stepper Motors	20/04/20	27/04/20
Documentation	20/04/20	23/04/20
Testing	23/04/20	27/04/20
Stepper Motor Driver	27/04/20	11/05/20
Documentation	27/04/20	01/05/20
Testing	01/05/20	06/05/20
Library Development	06/05/20	11/05/20
PCB Design	11/05/20	18/05/20
Graphical User Interface	18/05/20	29/06/20
Sockets Experiments	18/05/20	25/05/20
Classes and Objects Experiments	25/05/20	01/06/20
Threads Experiments	01/06/20	08/06/20
VTK Experiments	08/06/20	15/06/20
Qt Experiments	15/06/20	22/06/20
Application Development	22/06/20	29/06/20
PCB Soldering	29/06/20	29/06/20
Mechanical Assembly	29/06/20	29/06/20
System Testing	30/06/20	30/07/20
Report Writing	09/03/20	30/07/20

Fig. 1.1: Timing of Project Execution.

1.4 Contributions

The 3D scanner and all the algorithms present here, constitute simple and economical methods to virtualize indoor rooms through point clouds, and its implementation was multidisciplinary:

- ***Creation of C++ Arduino based Libraries:*** To simplify and better organize the main code of the microcontroller, several libraries were developed, each one of them related to one of the sensors or actuators used.
  - *TFmini-S LIDAR Distance Sensor*,
  - *BNO055 IMU Sensor*,
  - *DRV8825 Stepper Motor Driver*.
- ***Main microcontroller code:*** This code makes use of the libraries listed before, gets the data from the respective sensors, sends that same data to the computer and controls the stepper motors.
- ***Printed Circuit Board:*** To contain all the electronics in an embedded solid board, making it easier to power them and to prevent malfunctions and loose wires during testing.

- ***Graphical User Interface:*** Contains all the user control interface and data handling. Sends the commands to the board and receives the data that will be displayed in a sub-window. A general template of this software was provided by Carlos A. Vinhais, PhD.

1.5 Report Organization

This report is divided in the following chapters:

- ***Chapter*** 2: In this chapter, the fundamentals of the different types of 3D scanners are addressed, focusing on their respective working principles. As well as references to 3D Scanners applications and market solutions.
- ***Chapter*** 3: In this chapter, all of the hardware related observations are studied and demonstrated. Namely, the microcontroller, the IMU, the LIDAR, the Stepper Motor and the Stepper Motor Driver.
- ***Chapter*** 4: In this chapter, the implementation of the hardware is explained. First on a breadboard, then the mechanical structure, the power considerations, the PCB and the project financial analysis.
- ***Chapter*** 5: In this chapter, all of the software related observations are studied and demonstrated. Namely, Sockets, Threads, Visualization ToolKit (VTK) and Qt User Interface. There is also an explanation on how to use the developed graphical user interface and how to make an executable Python file.
- ***Chapter*** 6: In this chapter, the scanner operation is explained and the different point cloud test results are displayed and compared. The post-processing of the point cloud files are also demonstrated, and includes the calculation of the area and perimeter, as well as the use of dedicated software to handle point cloud data.
- ***Chapter*** 7: In this chapter, the final conclusions about the developed project are presented. Some relevant aspects are discussed along with some final suggestions for future development.

Chapter 2

3D Scanners

2.1 Fundamentals

A 3D scanner is a device that analyses a real-world object or environment to obtain information about its three-dimensional shape, in an automatic or semi-automatic way, producing point clouds or depth maps [6]. These special scanners can be divided into two main groups: contact and contactless (also called range finders). The contact type is not practical for the pretended application so this work will focus on range finders. A more complete classification is illustrated in Fig. 2.1.

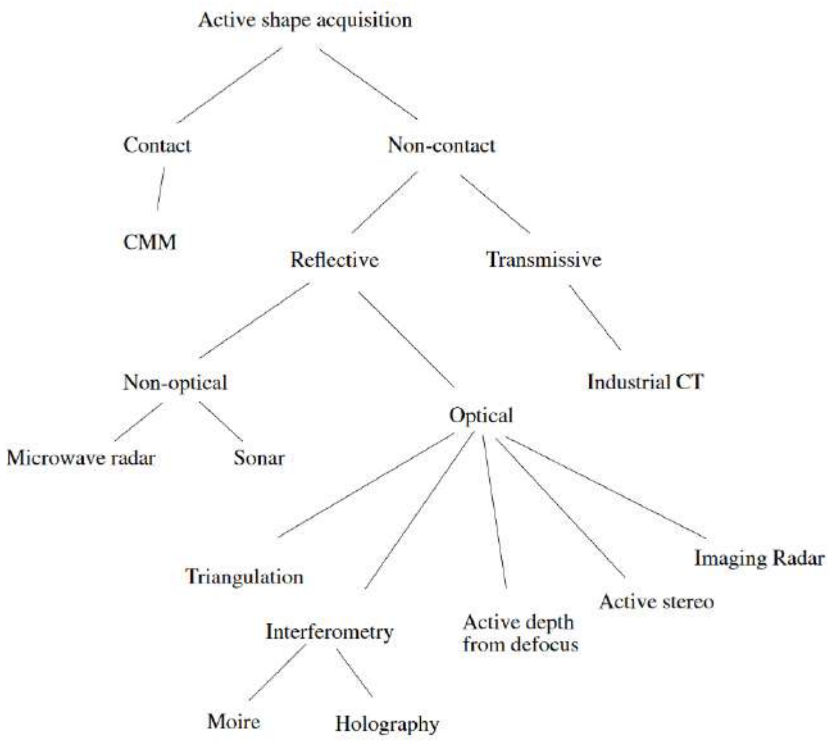


Fig. 2.1: Taxonomy of Active Range Acquisition Methods [7].

Point clouds are groups of certain points spread in space while depth maps represent information of distances measured from an observer (with an optical sensor or camera) to the scene. These can then be used to build 3D virtual models of the object. This activity has become more common each day due to a significant improvement in 3D technology, both in software and hardware, that allows high precision analysis about the shape and characteristics of the object surface [8]. The reconstruction process presents three stages [9]:

1. Sampling of the real world in point clouds or depth maps,
2. Alignment of several views within the same coordinate system, known as image registration,
3. Integration of the views for the generation of surface meshes, named merging.

2.2 Contact Methods

These make use of an analog probe with a known diameter that moves across an object contact surface and acquires points related to it through contact. An example of this type of system is the *DS10 contact scanner* [10], shown in Fig. 2.2, developed by *RENISHAW*.

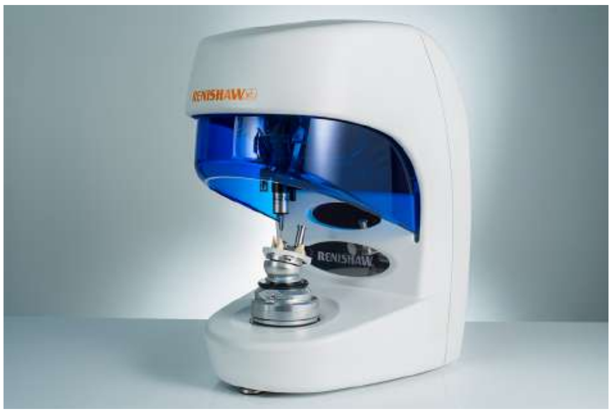


Fig. 2.2: DS10 Contact Scanner [10].

Some advantages related to this type of system are [11]:

- No need for surface treatment, to avoid reflection,
- Amount of data depends on the complexity of the surface,
- It is extremely accurate.

Meaning it is great for detailed solid surfaces, but also has some disadvantages [12, 13]:

- Needs contact with the object, which might damage it or change its shape,
- It is slow compared to contactless methods.

2.3 Contactless Methods

As the name suggests, this type of scanners don't touch the object, being less invasive and more flexible. As seen in Fig. 2.1, these systems are subdivided into transmissive and reflective. Inside the reflective category, there are the optical and non-optical variants. SONAR (Sound Navigation and Ranging) and RADAR (RAdio Detecting And Ranging) are some examples of non-optical methods. The working principle is determining the distance to an object by emitting a sound pulse or microwave energy, measuring the return time of that same pulse. Optical systems are divided into active and passive, having each one of them, several methods to acquire three-dimensional data of objects [14]. This project will focus on the active ones.

2.3.1 Stereoscopic Scanners

It is possible to obtain depth maps from stereoscopic cameras, creating RGBD (Red, Green, Blue and Depth) types of files. A typical system for the building of 3D models from stereo images is done in three steps. In the first step, a set of matched points (triangulation with pixels in the two views that are the images of the same point in the real world), are established between the two cameras images. The second step is about identifying matched points that are used to derive the relative locations, orientations and other parameters of the cameras, through a process called visual odometry. In the final step, the positions of the 3D points are calculated [15].

To test and prove this concept, that could prove to be a viable path for this project, several tests were made, involving *OpenCV* (Open Source Computer Vision Library) for Python. Two Logitech C270 cameras were used to perform these tests. The python code and algorithms, were adapted from the following GitHub code repository.

The first step was to calibrate the camera and to do that, the algorithm needs to take several pictures of a chessboard paper in different positions, to detect its corners and find the set of matched points as explained before. Around 30 pictures were taken per camera, which gave it stable values for the intrinsic parameters of the cameras. One example of these pictures is illustrated in Fig. 2.3.

Once the intrinsic parameters of the cameras are obtained, it is possible to create an RGBD (depth map) picture. Preliminary results are displayed in Fig. 2.4. However, the cameras need to stay in a parallel position throughout the whole process. The red lines on one of the cameras must match the same lines of the other camera in the real world, to obtain the best possible results. The brightest colours mean that the object (or point) is close to the centre point in between the cameras, and the darkest means the opposite.

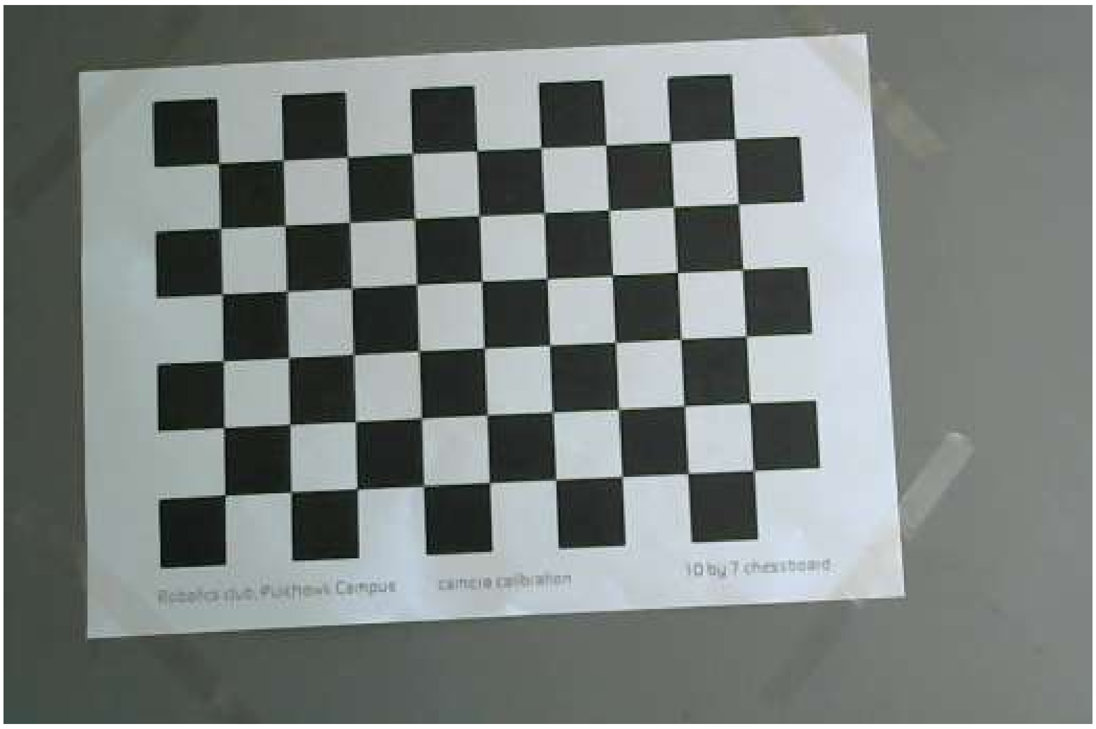
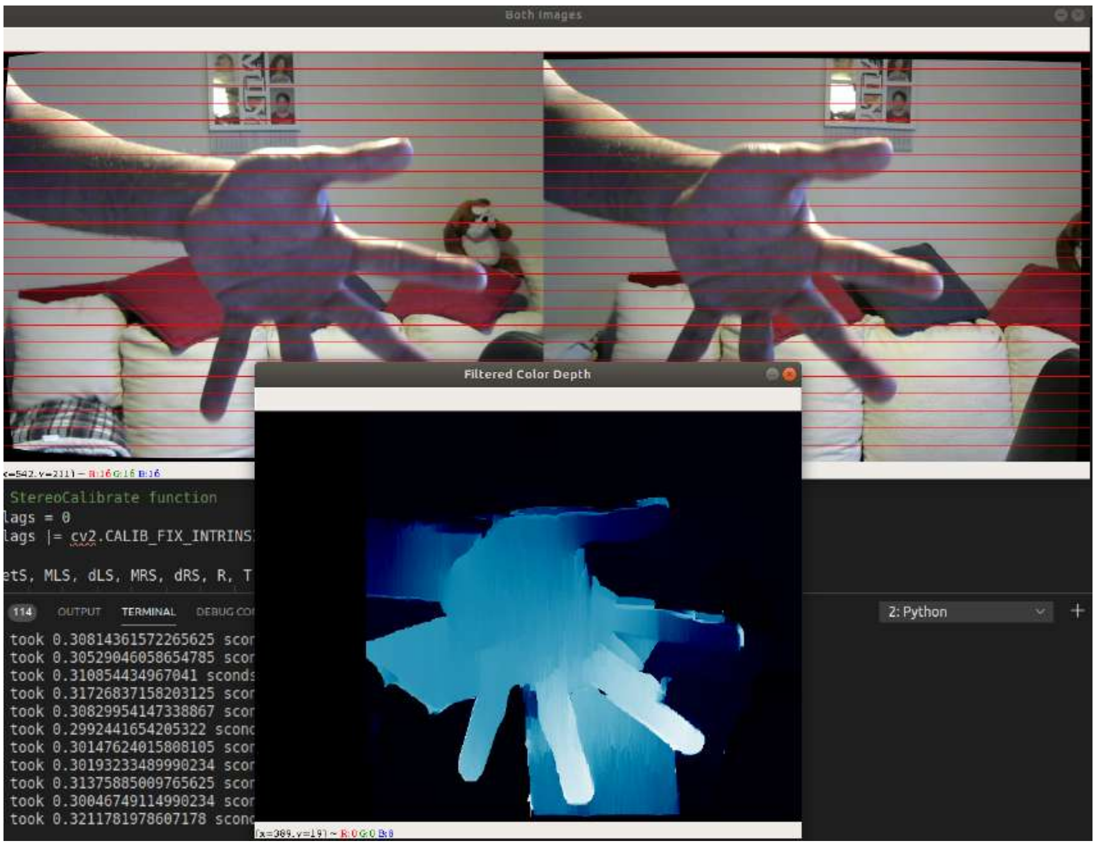


Fig. 2.3: OpenCV Test: Chessboard.



```
 StereoCalibrate function
lags = 0
lags |= cv2.CALIB_FIX_INTRINS

etS, MLS, dLS, MRS, dRS, R, T
```

```
 took 0.30814361572265625 scor
 took 0.30529046058654785 scor
 took 0.310854434967041 sconds
 took 0.31726837158203125 scor
 took 0.30829954147338867 scor
 took 0.2992441654205322 scond
 took 0.30147624015808105 scor
 took 0.30193233489990234 scor
 took 0.31375885009765625 scor
 took 0.30046749114990234 scor
 took 0.3211781978607178 scond
```

Fig. 2.4: OpenCV Test: Depth Image.

2.3.2 Rotational Scanners

Advances in LIDAR technology, in particular 360° LIDAR sensors, create new opportunities to augment and improve the sampling frequency and detection angle of 2D and 3D Scanners. The result is a dense point cloud containing data that represents a wider field-of-view of its surroundings, capable of scanning an entire room without additional sensors or manual intervention [16]. An example of the resulting point cloud is displayed in Fig. 2.5, that shows an ice ring with people skating on it.

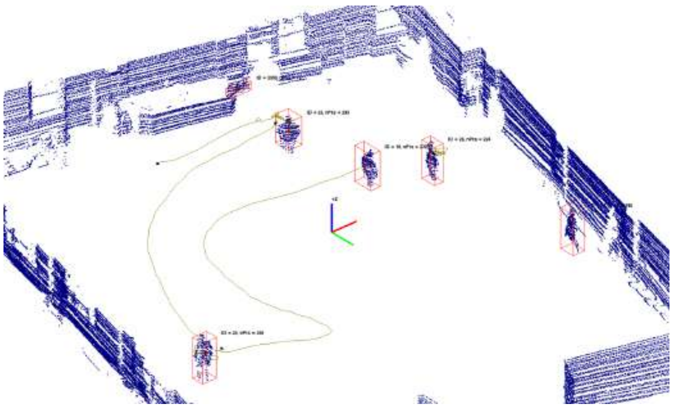


Fig. 2.5: Sample of 360° LIDAR Data [16].

That are many companies that make this type of scanners. One of them is *SLAMTEC*, located in Shanghai, China, creator of the scanner *RPLIDAR A1* [17], the first low-cost alternative of rotation LIDARs, in 2012. This scanner is showed in Fig. 2.6 and it is based on laser triangulation ranging principle, having a range of 12 meters in 360° with a sampling frequency up to 8000 Hz.

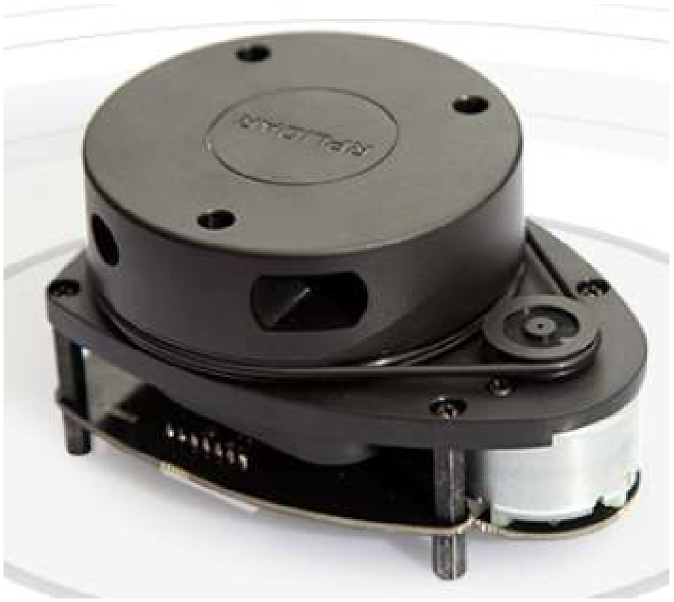
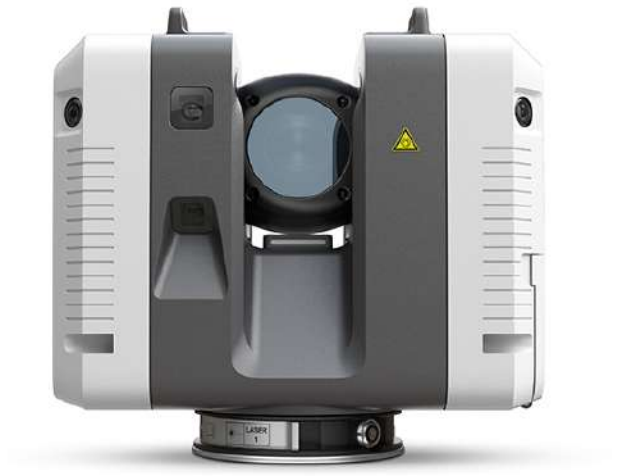


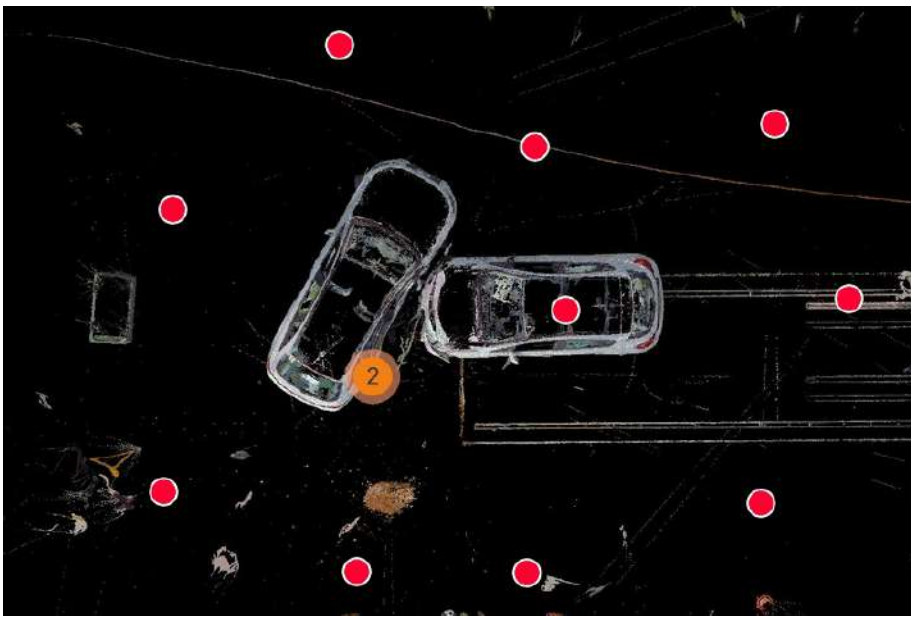
Fig. 2.6: RPLidar A1 [17].

2.3.3 Commercial Scanners

There are already commercially available 3D scanners like the *Leica RTC360 3D Laser Scanner*, produced by *Leica Geosystems*, illustrated in Fig. 2.7 a) and an example of the scanning results, displayed in Fig. 2.7 b) . It is commonly used in civil engineering and architecture, to help with the maintenance of infrastructures and construction site support, among others. The biggest obstacle in acquiring one of these scanners is the price, the *Leica RTC360 3D Laser Scanner* costs around 70 000€ and this is where the project is going to focus on. It will try to dissolve one of the problems about using this type of scanners.



(a) 3D Laser Scanner.



(b) Scanning Results.

Fig. 2.7: Leica RTC360 [18].

Chapter 3

Automation and Control

3.1 Microcontrollers

The microcontroller is the brain of the part of the project that is hardware related. The first thoughts when choosing the right microcontroller for this project are that it needs to be fast, have many available *GPIO's* (General Purpose Input Output pins), have the capability to communicate wirelessly, have a popular commercially available development board and to be as cheap as possible. The most popular development board is probably the Arduino UNO, that contains an ATmega328P microcontroller. However, it is not the fastest for the same price range. Another option would be one of the SMT32 boards, they are fast but unfortunately do not contain embedded wireless transmission capabilities such as *Wi-Fi*. The final verdict went to the ESP32 NodeMCU development board that contains an ESP32 microcontroller that is fast, has many available *GPIO's* and can transmit data via *Wi-Fi*, more specifically the ***ESP-WROOM-32*** microcontroller.

3.1.1 ESP32 Microcontroller

Made by Espressif and released on September 2016, the ESP32 [19] is a cheap SOC (System on a Chip) with many available *GPIO's* and built-in *Wi-Fi*. This microcontroller is crucial to the goal of making this project with wireless communication, making it possible to process the data in a server or dedicated *localhost* machine. The most important parameters are listed in Table 3.1, it has a high clock frequency for an MCU (MicroController Unit) and two available cores to make it even more efficient, plus a lower power co-processor to use in deep sleep mode. There only exists an SMD (Surface Mounted Device) form factor for this MCU and due to the impossibility of soldering SMD components such as the bare ESP32, this project makes use of a carrier board with the MCU already soldered to it. Being said, this is the main reason why this project is based on the carrier board ***ESP32 NodeMCU*** illustrated in Fig. 3.1.

Tab. 3.1: ESP32 Specifications.

Parameters	**Specification**
Architecture	32-bit
Number of Cores	2 + ultra low power co-processor
Clock Frequency	160MHz or 240MHz
Logic Level	3.3V
Wi-Fi	Yes
ROM	448 KiB
SRAM	520 KiB
GPIO Number	32
I2C	Yes
UART	Yes, 3
Price	3.48€
Supplier	LCSC.com
Price Module	9.99€
Supplier Module	Amazon.es

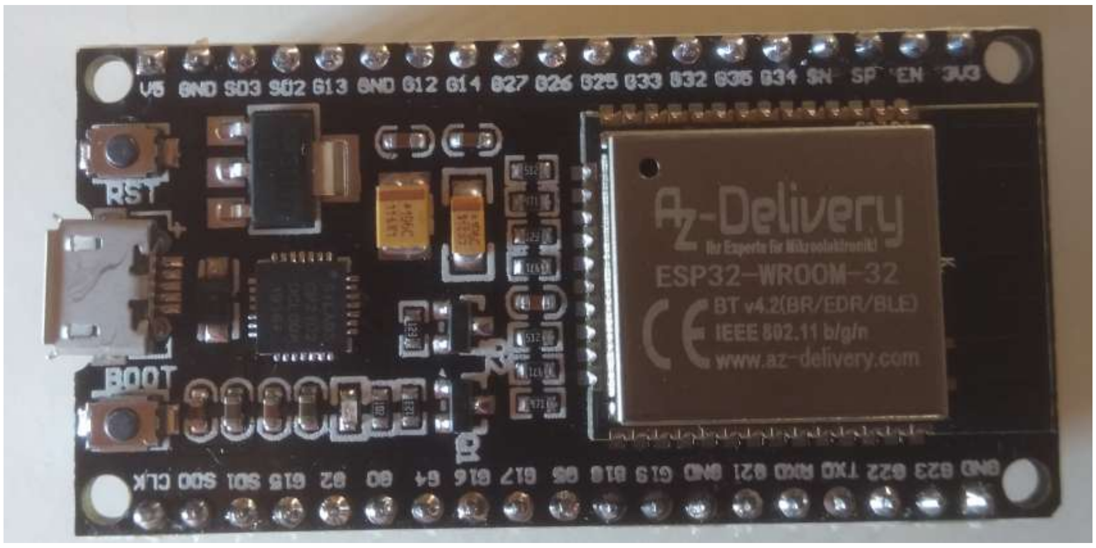

V5 GND SD3 SD2 G13 GND G12 G14 G27 G26 G25 G33 G32 G35 G34 SN SP EN 3V3
RST
BOOT
AZ-Delivery
ESP32-WROOM-32
BT v4.2(BR/EDR/BLE)
IEEE 802.11 b/g/n
www.az-delivery.com
CLK SD0 SD1 G15 G2 G0 G4 G16 G17 G5 G18 G19 GND G21 RXD TXD G22 G23 GND

Fig. 3.1: ESP32 Microcontroller.

3.1.2 Programming Environment

There are many programming languages and integrated development environments (IDE) for the ESP32, however only three of the most popular ones will be referred here:

- MicroPython,
- ESP-IDF (Official Espressif Framework),
- Arduino IDE.

MicroPython

Created by Damien George, MicroPython is a reimplementation of the Python programming language that targets microcontrollers and embedded systems [20]. The official 'Get Started' tutorial can be found at MicroPython Get Started. As a follow up from that tutorial, a more complete start-up guide on how to start using this language, is presented in Appendix B, composed of 14 steps:

1. **Install** all the necessary software requirements.
2. **Connect** the board to the machine.
3. **Check** which USB Port is in use.
4. **Erase** the flash memory of the ESP32 and load the new firmware.
5. **Connect** to the micro-controller and enter Python prompt.
6. **Check** the current directory of the File System.
7. **Use** the ***help*** command, to understand the available options.
8. **Check** the clock frequency of the MCU and change it.
9. **Interact** with the GPIO's.
10. **Install** python modules to the board, from module ***upip***.
11. **Insert** files into the board.
12. **Use** RSHELL.
13. **Check** the directory of the board and all the connected boards.
14. **Copy** files to the board.

MicroPython was greatly explored in this project, and it is fairly easy to use and to install. However it is not the best option when it comes to handling low-level data such as byte frames and bitwise operations.

ESP-IDF

The ESP-IDF, Espressif IoT Development Framework, is a framework that provides toolchain, API, components and workflows to develop applications for ESP32 using Windows, Linux and Mac OS operating systems [21]. The official 'Get Started' tutorial can be found at ESP-IDF Get Started. This tool was explored in this project, but abandoned after some experiments due to problems during installation, and the complexity of code writing and compiling.

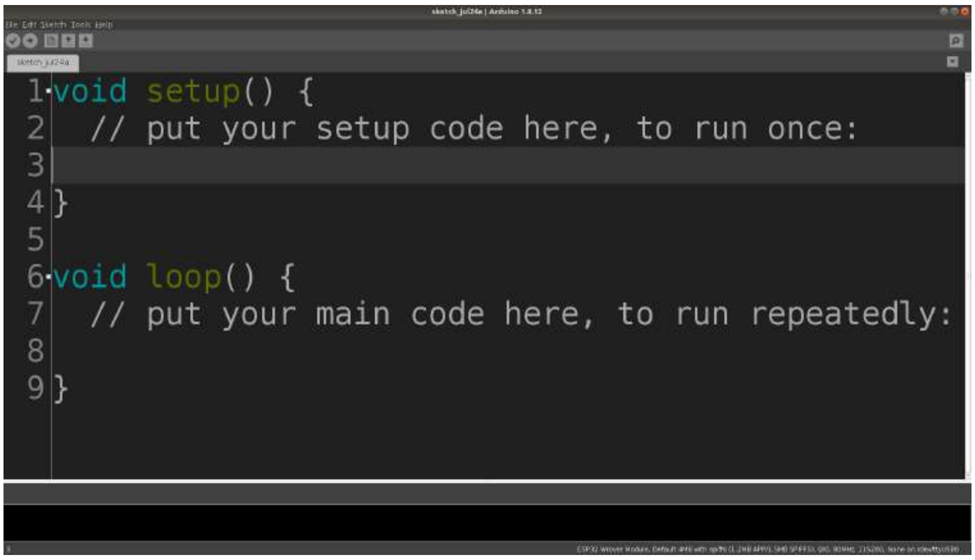
Arduino IDE

The Arduino IDE is an open-source Integrated Development Environment that allows the development of embedded systems. The Arduino language is merely a set of C/C++ functions that can be called from the code. The sketch undergoes minor changes (e.g. automatic generation of function prototypes) and then is passed directly to a C/C++ compiler (avr-g++). The official 'Get Started' tutorial can be found at Arduino IDE Get Started. As a follow up from that tutorial, the installation of the ESP32 tools on this IDE will be presented next. The version used was the 1.8.13 and it is the latest stable Arduino IDE software installed from Arduino IDE Download. In case MicroPython firmware is previously installed, the flash of the MCU must first be erased (the 'boot' button of the board might need to be pressed during this command).

```
$ esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```

This next list of 3 steps are only linked to the direct installation of the ESP32 on this IDE and not on how the software works in general.

1. The initial setup of the IDE is show in Fig. 3.2. **Go** to ***File > Preferences***.

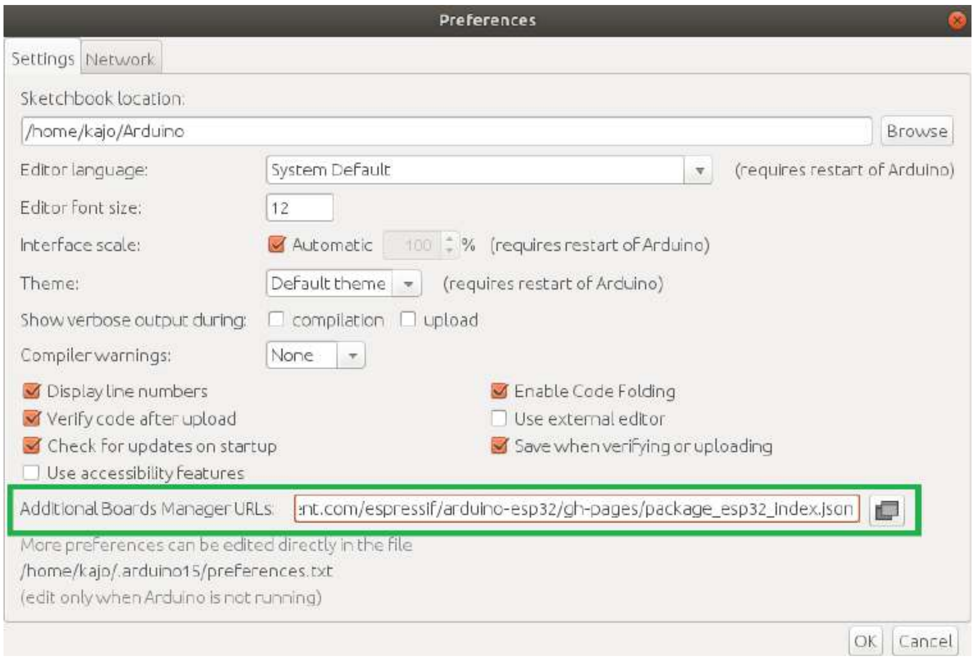


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

ESP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 115200, None on /dev/ttyUSB0

Fig. 3.2: Arduino IDE: General View.

2. Insert `https://dl.espressif.com/dl/package_esp32_index.json` into ***Additional Board Manager URLs*** field as shown in Fig. 3.3. Click *Ok* afterwards.



Preferences

Settings | Network

Sketchbook location:
/home/kajo/Arduino [Browse]

Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: ✓ Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None

✓ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
✓ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading

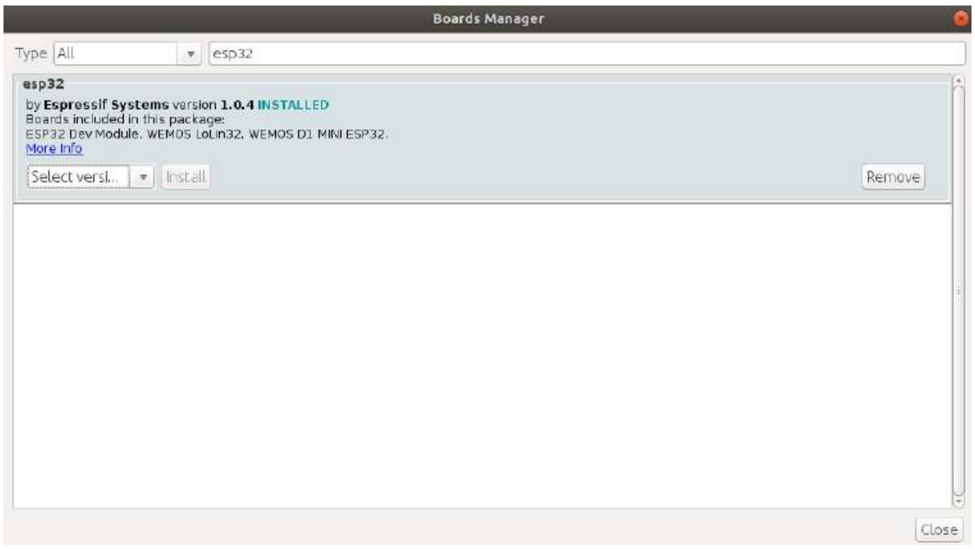
Additional Boards Manager URLs: :nt.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

More preferences can be edited directly in the file
/home/kajo/.arduino15/preferences.txt
(edit only when Arduino is not running)

[OK] [Cancel]

Fig. 3.3: Arduino IDE: Additional Boards Manager.

3. Go to ***Tools > Board > Boards Manager***. Insert *ESP32* in the search bar and install the *by Espressif Systems* option as seen in Fig. 3.4.



Boards Manager

Type All esp32

esp32
by **Espressif Systems** version **1.0.4** **INSTALLED**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

[Select versi...] [Install] [Remove]

[Close]

Fig. 3.4: Arduino IDE: Install board.

Example files can be found in ***File > Examples > ESP32*** . Note that after the upload of the sketch, the *RESET* button of the board must be pressed in order to run the new sketch.

3.2 IMU

3.2.1 Working Principles

Inertial Measurement Unit (IMU) sensors are widely used in many different movement applications, measuring components like velocity, orientation, and gravitational force. A degree of freedom (DOF) determines the number of independent parameters in a system. The number of DOF represents the measurement in x, y, and z-axis for each sensor. Typically, an IMU contains an accelerometer, a gyroscope and a magnetometer, or at least two of those sensors, having each one of them usually three DOFs to measure from three axes [22]. To prove these characteristics, several tests were made with the IMU BNO055 that contains all of those sensors.

Euler Angles

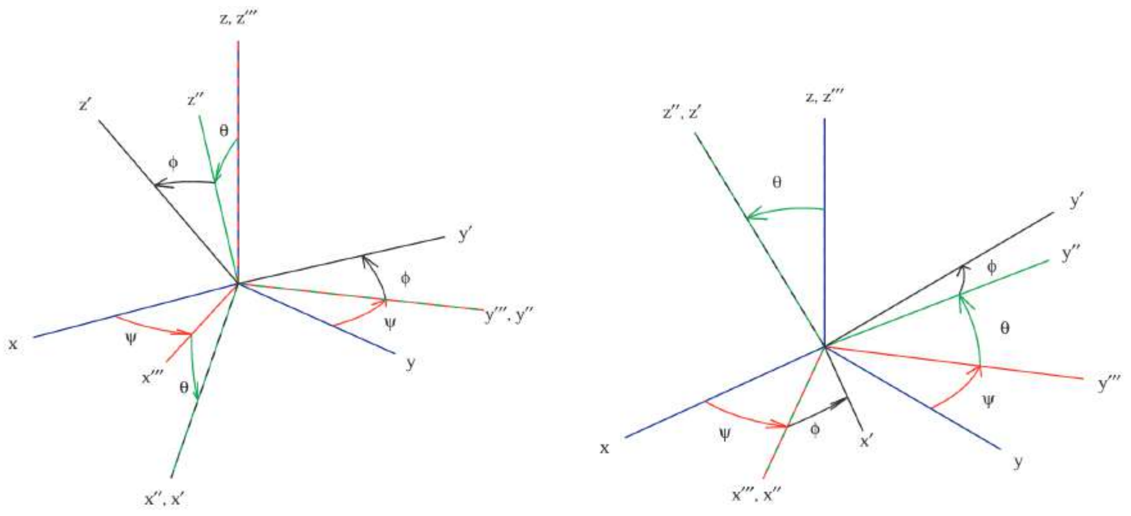
Euler Angles are a method to parameterize space orientation as a series of rotations about three mutually orthogonal axes in space. If we choose to consider a rotation as the action performed to obtain a given orientation, Euler angles can be used to parameterize the space of rotations. To describe a general rotation, three Euler angles (ϕ, θ, ψ) are required. The Cartesian coordinate systems axes (x, y, z) are the most commonly used and their rotations can be described accordingly as *roll*, *pitch* and *yaw* [23], although this method rises some problems.

ϕ	rotation about the X-axis (Roll),
θ	rotation about the Y-axis (Pitch),
ψ	rotation about the Z-axis (Yaw).

The conversion from a general rotation to Euler Angles is ambiguous since the same rotation can be obtained with different sets of Euler Angles, causing a problem called Gimbal Lock. Furthermore, the resulting rotation depends on the order in which the three rolls are performed, as seen in Fig. 3.5.

- Sequence (1,2,3)
  1. ψ / x / x'
  2. θ / y / y'
  3. ϕ / z / z'
- Sequence (3,1,3)
  1. $-\psi$ / -y / z'
  2. $\frac{\pi}{2} - \theta$ / -x / $-x'$
  3. ϕ / -z / $-y'$

This gives rise to further ambiguity but fits well with the fact that rotations in space do not generally commute. Some of the ambiguity in the conversion to Euler Angles can be eliminated by adopting a convention of which order the rolls should be performed [23].



(a) Sequence (1,2,3).　　(b) Sequence (3,1,3).

Fig. 3.5: Euler Angles [24].

Quaternions

To keep the focus on the main theme of this project, the subject quaternions will not be deeply explained. Instead, only some basic knowledge and notions will be presented here.

Quaternions were invented with the aim to generalize complex numbers to three dimensions, i.e. numbers of the form: $a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$, where a, b, c, $d \in \Re$ and $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ whereas a describes the size of the scaling, b number of degrees to be rotated and the last two numbers give the plane in which the vector should be rotated. Furthermore, a, b, c and d can also be described accordingly as q_0, q_1, q_2 and q_3. A quaternion is usually written as $[s, v]$, s $\in \Re$, $v \in \Re^3$. Here s is called the scalar part, and $v = (x, y, z)$ is the vector part [23].

Quaternions are a big advantage when computing rotations but it is hard for a human to visualize it, so sometimes its useful to convert it to Euler Angles at the end of the computing process. Quaternions can be converter to Euler Angles as follows [25]:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin 2(q_0q_2 - q_3q_1) \\ \arctan \frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}. \tag{3.1}$$

Where the X-axis points forward, Y-axis to the right and Z-axis downward with angles defined for clockwise/left hand rotation. This will be the method used to read the IMU. The data can be calculated by the IMU embedded microcontroller in quaternions and passed to the ESP32 using those same units. In a later process of the project, those quaternion units will be converted into Euler Angles, and easily visualized.

3.2.2 BNO055 IMU

Made by BOSCH, the BNO055 is an intelligent absolute orientation sensor, 9-axis Inertial Measurement Unit, System in Package (SiP) that contains a 3-Axis Accelerometer, a 3-Axis Gyroscope, a 3-Axis Magnetometer and a 32-bit cortex M0+ microcontroller running Bosch Sensortec sensor fusion software [26]. The key aspect about this IMU is that it can output fusion data from this three sensors. There only exists an SMD form factor for this IMU so, despite being an IC (Integrated Circuit), it is mostly know on its ”module” form factor because it allows users to test it on a breadboard or easily plug it to a PCB.

Being said, this is the main reason why this project is going to use the module version, which can be seen in Fig. 3.6.

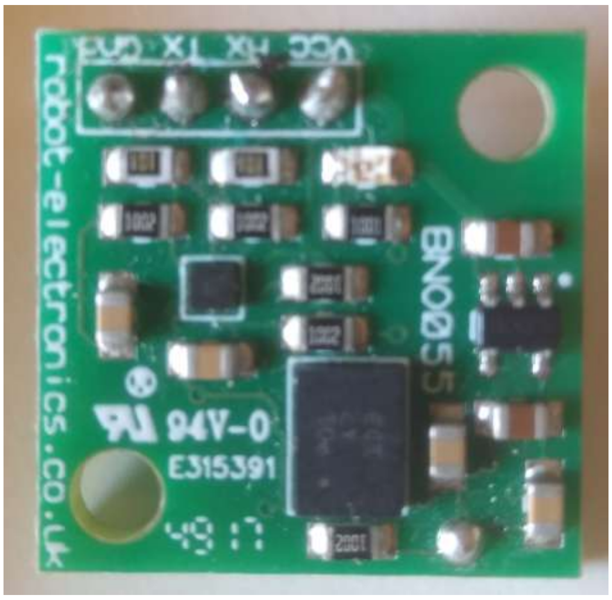


Fig. 3.6: BNO055 IMU.

BNO055 IMU Specifications

It has good sensors resolution, with a minimum of 13-bit resolution for the X and Y axis of the magnetometer and a maximum of 16-bit resolution for all gyroscope axes as well as configurable ranges to get the best possible results according to the purpose of use. Another big benefit is that it can output sensor fused data of many different types of data such as quaternion units, Euler angles, rotation vectors, linear acceleration, gravity vector and heading. It also has a low power consumption for a sensor of this type and logic levels that are compatible with the ESP32 microcontroller, combined with an affordable price in comparison to other IMU’s with data fusion algorithms. It’s possible to evaluate the most important parameters in Table 3.2.

Tab. 3.2: BNO055 Specifications.

Parameters	**Specification**
Sensors	14-bit Accelerometer
	16-bit Gyroscope
	(13/13/15)-bit (X, Y, Z) Magnetometer
Accelerometer features	Ranges: $\pm 2g$ / $\pm 4g$ / $\pm 8g$ / $\pm 16g$
	Low-pass filter bandwidths 1kHz - <8Hz
Gyroscope features	Ranges: $\pm 125°$/s to $\pm 2000°$/s
	Low-pass filter bandwitdhs 523Hz - 12Hz
Magnetometer features	Ranges: $\pm 1300\mu T$ (X-Axis, Y-Axis) / $\pm 2500\mu T$ (Z-Axis)
	Resolution: $\approx 0.3\mu T$
Output Fused Sensor Data	Quaternion
	Euler Angles
	Rotation Vector
	Linear Acceleration
	Gravity Vector
	Heading
Communication Interfaces	I2C
	UART
Voltage Supply	2.4 - 3.6V
Total supply current	12.3mA
Communication Level	LVTTL (3.3V)
Interface	I2C / UART
Price	8.91€
Supplier	LCSC.com
Price Module	19.90€
Supplier Module	botnroll.com

The basic building blocks of the BNO055 device can be seen in Fig. 3.7. It takes data from all the sensors so that it can later process data fusion algorithms to obtain more stable results and transmit them to a host processor.

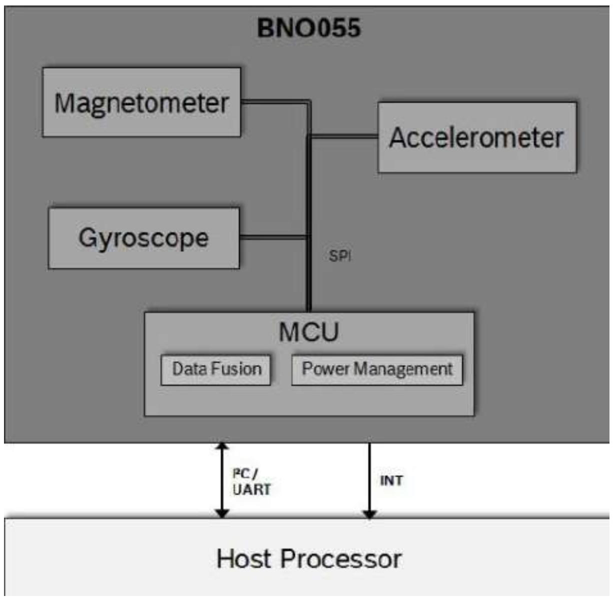
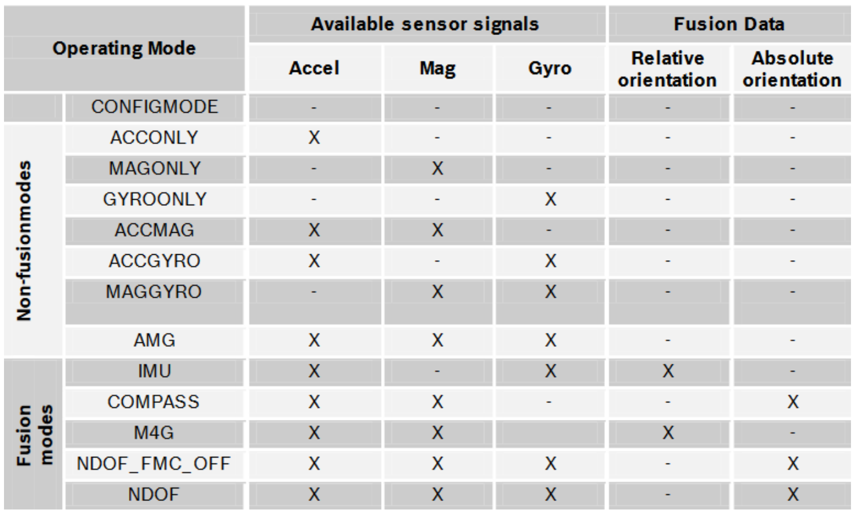


Fig. 3.7: BNO055 Architecture [26].

The BNO055 provides a variety of output signals, which can be chosen by selecting the appropriate operation mode. The different modes and the available sensor signals are listed in Fig. 3.8, the NDOF will be the one used. This fusion mode has 9 degrees of freedom where the fused absolute orientation data is calculated from the accelerometer, gyroscope and magnetometer. The advantages of combining this three components are a fast calculation, resulting in high output data rate, and high robustness from magnetic field distortions. The entire communication with the device is performed by reading from registers and writing to the registers in Tab. A.1 with signed data type [26].



Operating Mode		Available sensor signals			Fusion Data	
		Accel	Mag	Gyro	Relative orientation	Absolute orientation
	CONFIGMODE	-	-	-	-	-
Non-fusionmodes	ACCONLY	X	-	-	-	-
	MAGONLY	-	X	-	-	-
	GYROONLY	-	-	X	-	-
	ACCMAG	X	X	-	-	-
	ACCGYRO	X	-	X	-	-
	MAGGYRO	-	X	X	-	-
	AMG	X	X	X	-	-
Fusion modes	IMU	X	-	X	X	-
	COMPASS	X	X	-	-	X
	M4G	X	X		X	-
	NDOF_FMC_OFF	X	X	X	-	X
	NDOF	X	X	X	-	X

Fig. 3.8: BNO055 Fusion Modes [26].

The default operation mode after power-on is *CONFIGMODE* and the default sensor settings are displayed in Tab. A.2. The representation of the actual values are listed in Tab. A.3. During the initialization, this sensor always does a *Power On Self Test*, that examines the status of all the different components that it is composed of, and stores the result in a register. This might be useful in order to check for hardware related errors. The units used in the returned data can be seen in Tab. A.4.

IMU Calibration

Though the sensor fusion software runs the calibration algorithm of all the three sensors (accelerometer, gyroscope and magnetometer) in the background to remove the offsets, some preliminary steps had to be ensured for this automatic calibration to take place.

The accelerometer and the gyroscope are relatively less susceptible to external disturbances, as a result of which the offset is negligible. Whereas the magnetometer is susceptible to external magnetic fields and therefore to ensure proper heading accuracy, the calibration steps described below have to be taken.

The following simple steps had to be taken after every *Power on Reset* for proper calibration of the device.

Accelerometer Calibration:

- Place the device in 6 different stable positions for a period of few seconds to allow the accelerometer to calibrate.
- Make sure that there is slow movement between 2 stable positions.
- The 6 stable positions could be in any direction, but make sure that the device is laying at least once perpendicular to the x, y and z axis.

Gyroscope Calibration:

- Place the device in a single stable position for a period of few seconds to allow the gyroscope to calibrate.

Magnetometer Calibration:

- Make some random movements (for example: writing the number '8' on the air) until it is calibrated.

3.2.3 I2C Protocol

The default I2C address of the BNO055 device is 0101001b (0x29). The timming diagram of this type of communication is displayed in Fig. 3.9.

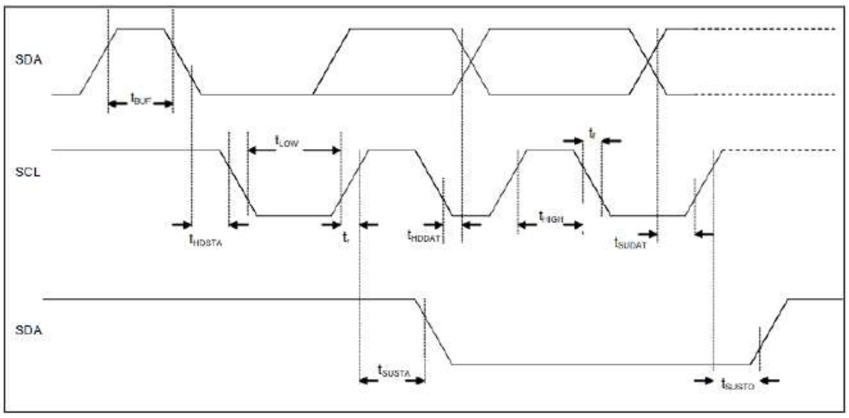


Fig. 3.9: BNO055 I2C Timming Diagram [26].

The I2C protocol works as follows [26]:

- START: Data transmission on the bus begins with a high to low transition on the SDA line while SCL is held high.
  - Start condition (S) indicated by I2C bus master.
  - Once the START signal is transferred by the master, the bus is considered busy.
- STOP: Each data transfer should be terminated by a Stop signal (P) generated by master.
  - The STOP condition is a low to HIGH transition on SDA line while SCL is held high.
- ACK: Each byte of data transferred must be acknowledged.
  - It is indicated by an acknowledge bit sent by the receiver.
  - The transmitter must release the SDA line (no pull down) during the acknowledge pulse while the receiver must then pull the SDA line low so that it remains stable low during the high period of the acknowledge clock cycle.

In the following diagrams these abbreviations are used:

S	Start,
P	Stop,
$ACKS$	Acknowledge by slave,
$ACKM$	Acknowledge by master,
$NACKM$	Not acknowledge by master,
RW	Read / Write.

I2C write access [26]:

It can be used to write a data byte in a 5 steps sequence.

1. The sequence begins with start condition generated by the master, followed by 7 bits slave address and a write bit (RW = 0).
2. The slave sends an acknowledge bit (ACK = 0) and releases the bus.
3. Then the master sends the one byte register address.
4. The slave again acknowledges the transmission and waits for the 8 bits of data which shall be written to the specified register address.
5. After the slave acknowledges the data byte, the master generates a stop signal and terminates the writing protocol.

An example of an I2C write access to the BNO055 (I2C address is 0x28) can be seen in Fig. 3.10 and the application on the BNO055 created library in List. 3.1:

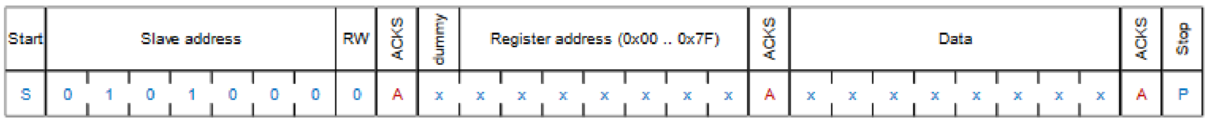


Fig. 3.10: BNO055 I2C Write Example [26].

Listing 3.1: Sample of BNO055 I2C Write from File: BNO055.cpp

```
// Register write algorithm:
// 1 - Begin transmission with the device
// 2 - Write the address of the register
// 3 - Write the value to the register
// 4 - End transmission with the device

bool BNO055::Register_Write(uint8_t reg, uint8_t value)
{
        Wire.beginTransmission(byte(_ADDRESS));
        Wire.write(byte(reg));
        Wire.write(byte(value));
        Wire.endTransmission();
        return true;
}
```

I2C read access [26]:

It can be used to read one or multiple data bytes in 6 steps sequence. A read sequence consists of a one-byte I2C write phase followed by the I2C read phase. The two parts of the transmission must be separated by a repeated start condition (Sr).

1. The I2C write phase addresses the slave and sends the register address to be read.
2. After slave acknowledges the transmission, the master generates again a start condition and sends the slave address together with a read bit (RW = 1).
3. Then the master releases the bus and waits for the data bytes to be read out from slave.
4. After each data byte the master has to generate an acknowledge bit (ACK = 0) to enable further data transfer.
5. A NACKM (ACK = 1) from the master stops the data being transferred from the slave.
6. The slave releases the bus so that the master can generate a STOP condition and terminate the transmission.

The register address is automatically incremented and, therefore, more than one byte can be sequentially read out. Once a new data read transmission starts, the start address will be set to the register address specified in the latest I2C write command. By default the start address is set at 0x00. In this way repetitive multi-bytes reads from the same starting address are possible.

An example of an I2C write access to the BNO055 (I2C address is 0x28) can be seen in Fig. 3.11 and the application on the BNO055 created library in List. 3.2:

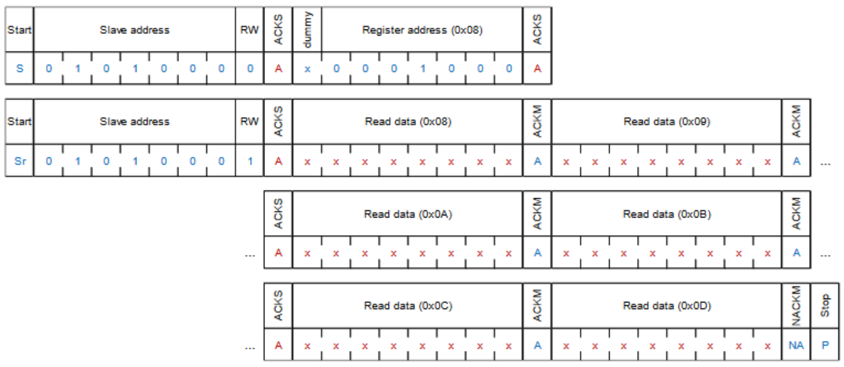


Fig. 3.11: BNO055 I2C Read Example [26].

Listing 3.2: Sample of I2C Read from File: BNO055.cpp

```
byte BNO055::SingleRegister_Read(uint8_t reg)
{
  byte value = 0;
  Wire.beginTransmission(byte(_ADDRESS));
  Wire.write(byte(reg));
  Wire.endTransmission();
  Wire.requestFrom(byte(_ADDRESS), (byte)1);
  value = Wire.read();
  return value;
}

byte* BNO055::MultiRegister_Read(uint8_t reg, byte *arr, uint8_t len)
{
  uint8_t i;
  Wire.beginTransmission(byte(_ADDRESS));
  Wire.write(byte(reg));
  Wire.endTransmission();
  Wire.requestFrom(byte(_ADDRESS), byte(len));
  for (i = 0; i < len; i++) arr[i] = Wire.read();

  return arr;
}
```

3.2.4 Developed IMU Sensor Library

During the development of this project, a C++ Library was developed in order to organize the code, making it more efficient and readable. A sample of the .h file is presented in List. 3.3, which includes the BNO055 object and its methods:

Listing 3.3: File: BNO055.h

```
#include "Arduino.h"
#include <Wire.h>

class BNO055 {
  public:
    //Constructor
    BNO055();

    //Methods
    void  begin(uint8_t i2c_address);
    bool  Register_Write(uint8_t reg, uint8_t value);
    byte  SingleRegister_Read(uint8_t reg);
    byte* MultiRegister_Read(uint8_t reg, byte *arr, uint8_t len);
    float RegisterPair(char regist);
    byte  CheckStatus(void);
    void  Config(void);
    void  Calibrate(int *sys, int *gyr, int *acc, int *mag);
    int   getTemp(void);
    void  getIDs(byte *chip_add, byte *acce_add, byte *magn_add, byte *gyro_add);
    float readACC(char axis);
    float readGYR(char axis);
    float readMAG(char axis);
    float readEUL(char axis);
    float readQUA(char axis);
    float readLIA(char axis);
    float readGRV(char axis);
    float readQUA_EUL(char axis);

  private:
    uint8_t _ADDRESS;
};
```

3.2.5 Experimental Results

An **accelerometer** is a device that measures inertial acceleration. To test the accelerometer, the device was put into a stable position and left untouched for 1 second as seen in Fig. 3.12. The offset on the X and Y axis will not be taking into consideration since having perfect alignment with those axes would be a very complex process and the sensor wasn't properly calibrated. On the Z-axis, the offset is much bigger than on the other ones, and this is due to the gravitational acceleration applied in it ($\boldsymbol{g} \approx 9.81$ m/s^2) plus some non-calibration offset. From all three axes, it would be expected (in ideal situations) to obtain a straight horizontal line, however, that is not the case considering small periods, so it's possible to conclude that they're very sensitive to small signals.

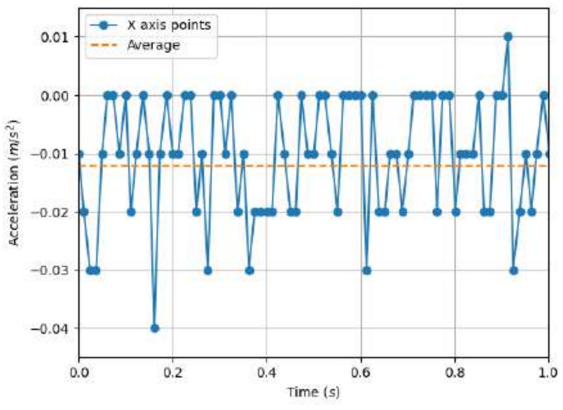
A **gyroscope** is a device that measures angular rotation. A similar testing process was taken to test the gyroscope, the device was put into a stable position and left untouched for 60 seconds as seen in Fig. 3.13. From all three axes, it would be expected (in ideal situations) to obtain a straight horizontal line equal to zero, however, that is not the case considering large periods, so it's possible to conclude that they're sensitive to drifting.

A **magnetometer** is a device that measures bearing magnetic direction. To test the magnetometer, the device was put in a stable position and left untouched for 20 seconds as seen in Fig. 3.14. During the testing period, a magnet was taken close and then far away from the sensor. This presence of the magnet affected all the three axes of the sensor, where the first peak corresponds to the approach of the magnet and the second corresponds to the magnet leaving.

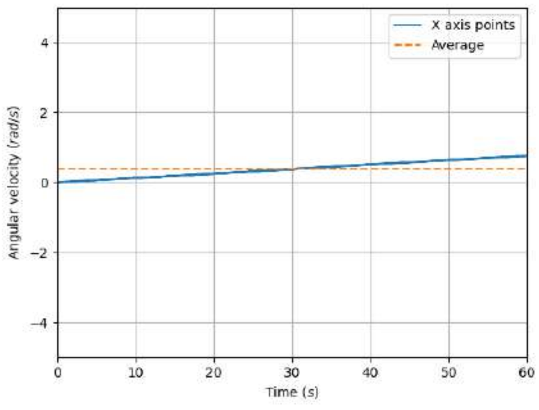
With the tests results it is possible to conclude that:

- Accelerometer: Measures inertial acceleration and is sensitive to high frequencies,
- Gyroscope: Measures angular rotation and is sensitive to drift,
- Magnetometer: Measures bearing magnetic direction and is sensitive to any disturbance of magnetic fields.

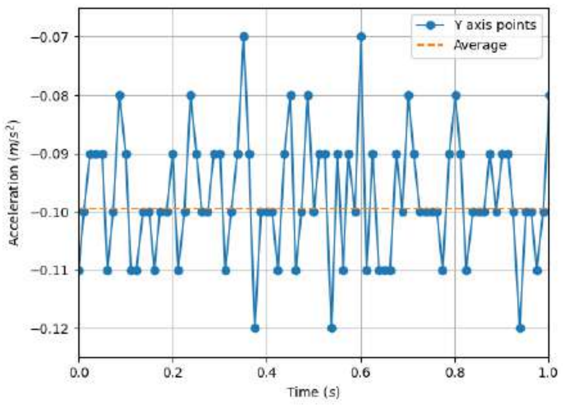
To analyse the performance of the IMU BNO055 fusion software returning quaternion units, several tests were also performed. The first test was to roll the IMU 360°. The results were approximately the same for both methods, varying between −90° and +90°. The second test was to apply a pitch movement on the IMU for 360°. The results were approximately the same for both methods, varying between −180° and +180°. It possible to see that once the angle reaches less than −180°, the IMU extrapolates to +180°. The last test was to apply a yaw movement on the IMU 360°. The results in this case were different, although with the same practical implications. The data in Euler Angles varies from 0° to 360°, extrapolating on the edges, while the conversion from quaternions to Euler Angles varies from −180° to +180° also extrapolating on the edges.



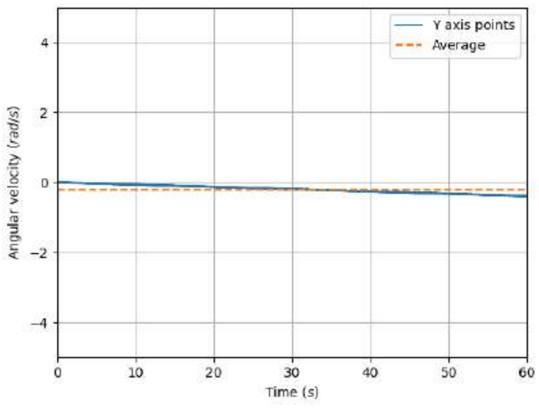
(a) X-Axis.



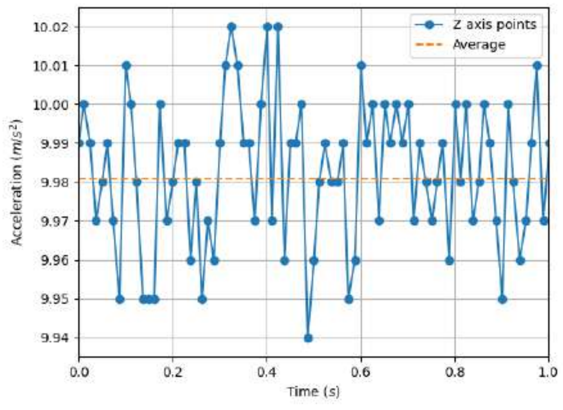
(a) X-Axis.



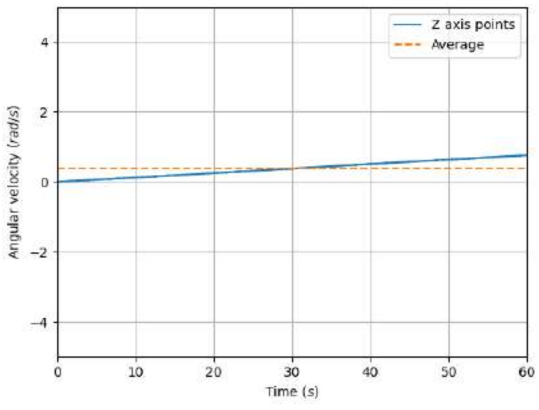
(b) Y-Axis.



(b) Y-Axis.



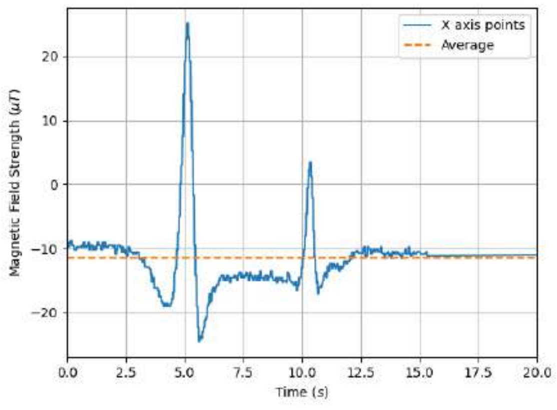
(c) Z-Axis.



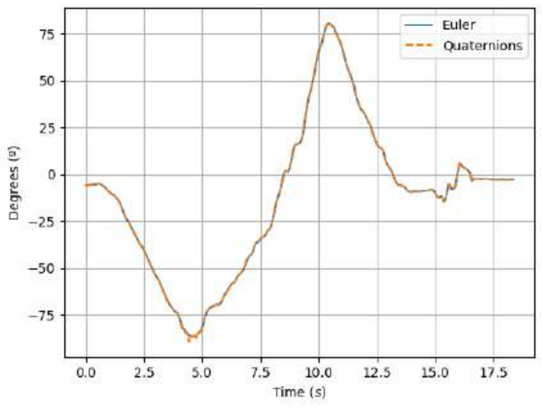
(c) Z-Axis.

Fig. 3.12: Accelerometer.

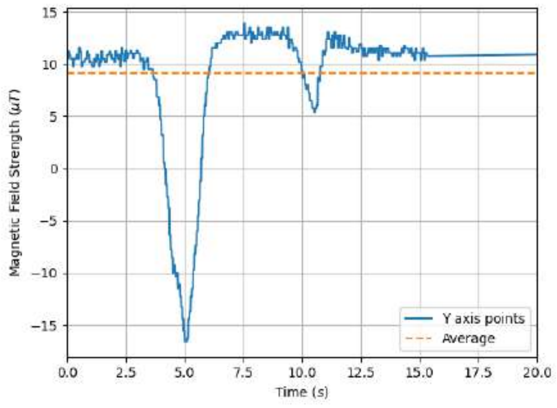
Fig. 3.13: Gyroscope.



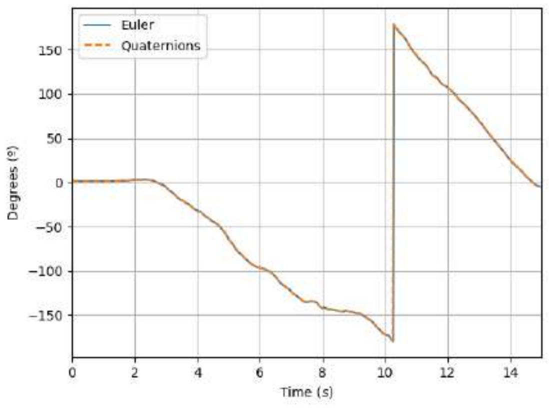
(a) X-Axis.



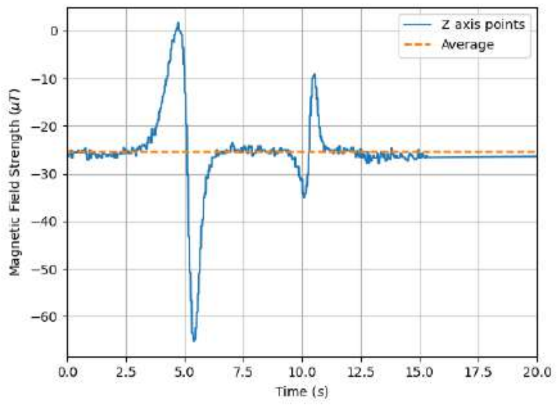
(a) X-Axis.



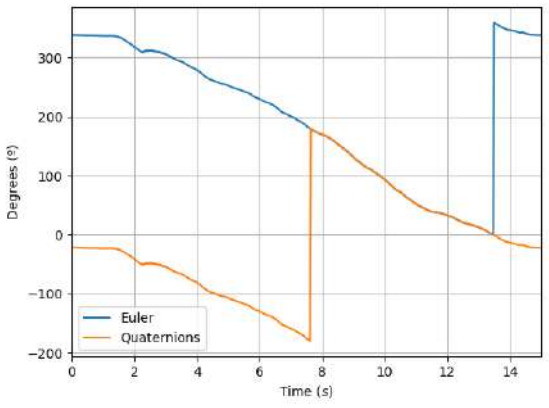
(b) Y-Axis.



(b) Y-Axis.



(c) Z-Axis.



(c) Z-Axis.

Fig. 3.14: Magnetometer.

Fig. 3.15: Absolute Orientation.

3.3 LIDAR

3.3.1 Working Principles

LIDAR (LIght Detection And Ranging), also known as LADAR (LAser Detection And Ranging) is an active remote sensing device that is typically used to obtain the range to one or more points of a target. This technology uses focused light (typically a laser) to illuminate the target and measure the time it takes for the emitted signal to return to the sensor [27]. In an ideal environment and with a perfect distance sensor, the distance can be calculated as follows:

$$r = \frac{c\,t}{2}, \tag{3.2}$$

where:

r	range from the sensor to a point on the target object (m),
c	speed of light (m/s),
t	is the laser time-of-flight (s).

LIDAR sensors may be divided into a few different categories based on how they illuminate and sense the scene in front of them [27]:

- **Scanning**, Fig. 3.16:
  - Illuminates the scene through sweeping of a laser beam and senses the return with only a single detector.
  - Sends out discrete laser pulses/flashes and then await the return pulse at the detector.
- **Detector Arrays**, Fig. 3.17 and Fig. 3.18:
  - Illuminates the entire scene at once and sense the return with an array of detectors.
  - Modulates a signal onto the laser and then tracks the phase shift in the returning signal to measure the ToF.
- **Spatial Light Modulators**:
  - Illuminates portions of the scene in a pattern and then senses the return with a single detector (along with compressed sensing algorithms).
  - Consists on encoding a pseudo-random number sequence onto the laser and then perform an autocorrelation with the sensed return value to determine laser ToF.

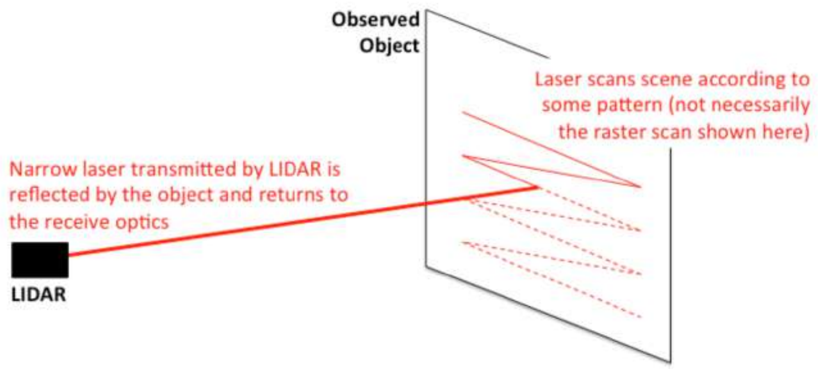

Observed
Object
Laser scans scene according to some pattern (not necessarily the raster scan shown here)
Narrow laser transmitted by LIDAR is reflected by the object and returns to the receive optics
LIDAR

Fig. 3.16: Scanning LIDAR [27].

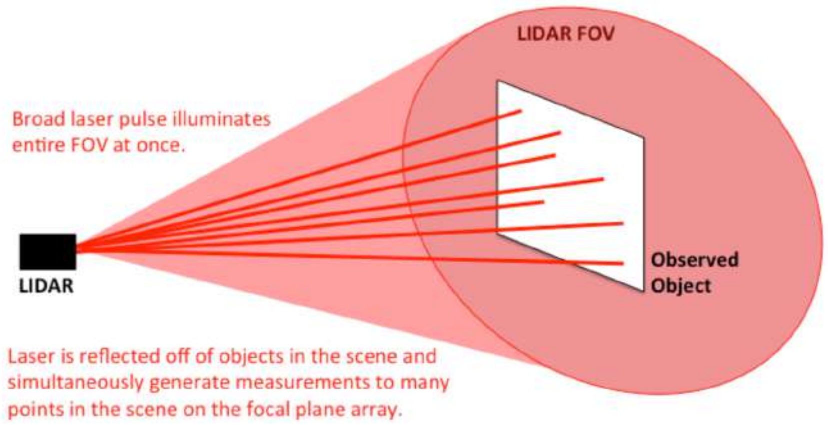

LIDAR FOV
Broad laser pulse illuminates entire FOV at once.
LIDAR
Observed
Object
Laser is reflected off of objects in the scene and simultaneously generate measurements to many points in the scene on the focal plane array.

Fig. 3.17: Detector Arrays [27].

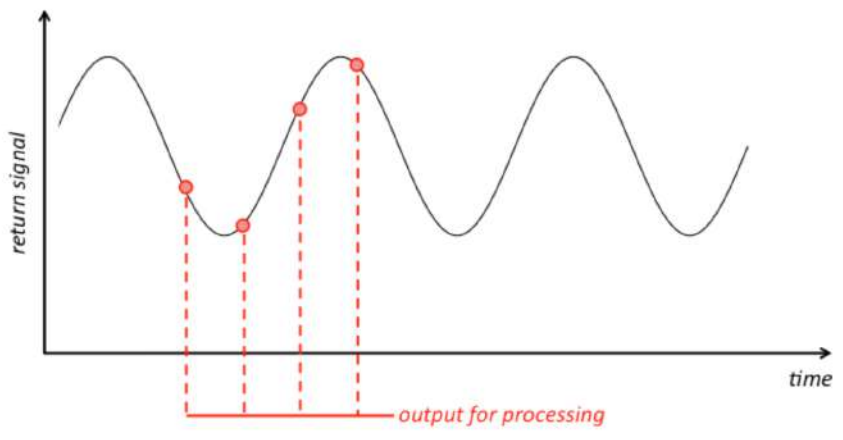


Fig. 3.18: Detector Arrays Modulation [27].

3.3.2 TFmini-S LIDAR

Choosing the best sensor

From Tab. 3.3, it is possible to compare the generality of the different types of small sized low-cost (under 100€) distance sensors that are commercially available in one of the biggest electronics online retail stores, SparkFun Eletronics.

Tab. 3.3: Comparison of Distance Sensors Technology.

Specification	**LIDAR**	**SONAR**	**Infrared**
Range (cm)	10 - 1200	0 - 765	0 - 150
Field of View (°)	2 - 27	≈ 15	0.3 - 42
Resolution (mm)	≈ 5	≈ 3	≈ 1
Price (€)	21.95 - 62	3.95 - 100	6.95 - 25.95

A good distance sensor for this application shall have a big range of measured distance, a small field of view, a small resolution and a low price. The infrared sensors have a small range, many indoor spaces have a radius bigger than 150 cm, so for this reason they shall not be used. The SONAR sensors have a good range but also a big field of view which is great for big object detection but not for small points in order to increase the resolution of the points scanned. Finally, the LIDAR sensors are the best option, they have a big range and many of them have a small field of view. The resolution is not as good as the other technologies but is compensated by the big range and small field of view. The price is medium.

TFmini-S

Made by Benewake, TFmini-S (Fig. 3.19) is a miniaturized, low-cost, single-point ranging product, based on the TOF (Time Of Flight) principle, as described by its datasheet. It has a wide range scan, which is enough for most indoor spaces and has a small FOV (Field of View) compared to other sensors such as the HCSR04 (15°) which is really important for the accuracy and resolution of this application. It has a possible high frequency frame rate which is great for fast scans. It also has a low power consumption for a sensor of this type and logic levels that are compatible with the ESP32 micro-controller. The accuracy isn't very good as shown in Fig. 3.22 however the affordable price and the other parameters makes it a good choice for this project. In Tab. 3.4 it's possible to evaluate the most important parameters.

Tab. 3.4: TFmini-S Specifications

Parameter	**Specification**
Range	0.1 - 12m
Accuracy	±6cm (0.1 - 6m)
	±1% (6 - 12m)
Measurement Units	cm / mm
Range Resolution	1 cm
Field of View	2°
Operating center wavelength	850 nm
Frame Rate	1 - 1000Hz
Supply Voltage	5V ± 0.1V
Average Current	140 mA
Peak Current	200 mA
Average Power	700 mW
Communication Level	LVTTL (3.3 V)
Interface	UART / I2C
Price	35.99€
Supplier	Amazon.es

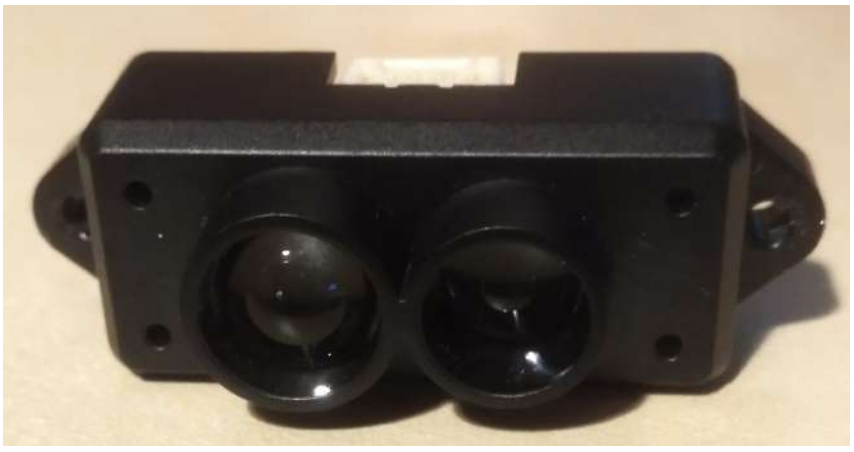


Fig. 3.19: TFmini-S LIDAR.

The ToF principle, that will be used internally by the sensor to calculate the distance to a certain point is represented by Eq. (3.3) and Fig. 3.20.

$$D = \frac{c}{2}\frac{1}{2\pi\, f}\Delta\phi, \tag{3.3}$$

where:

D	Distance (m),
c	Speed of light (m/s^2),
f	Frequency of the light signal (Hz),
$\Delta\phi$	Angle of phase shift (rad).

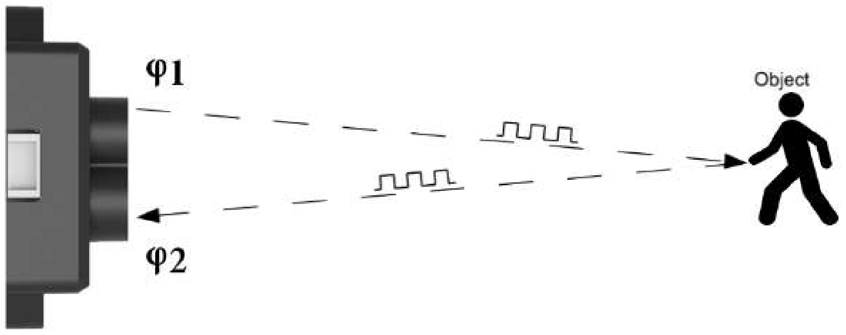


Fig. 3.20: TFmini-S Object Detection adapted from the datasheet.

Conditions with potential malfunction, described by the datasheet:

- Detecting an object with high reflectivity, such as a mirror or a smooth floor tile, may cause a system malfunction.
- The product will malfunction if there is any transparent object between it and the detecting object, such as glass or water.

If the light spot reaches two objects with different distances, as shown in Fig. 3.21, the output distance value will be a value between the actual distance values of the two objects. According to the specifications on Tab. 3.4 accuracy row, it is possible to write Eq. (3.4) and obtain the LIDAR accuracy graph in Fig. 3.22.

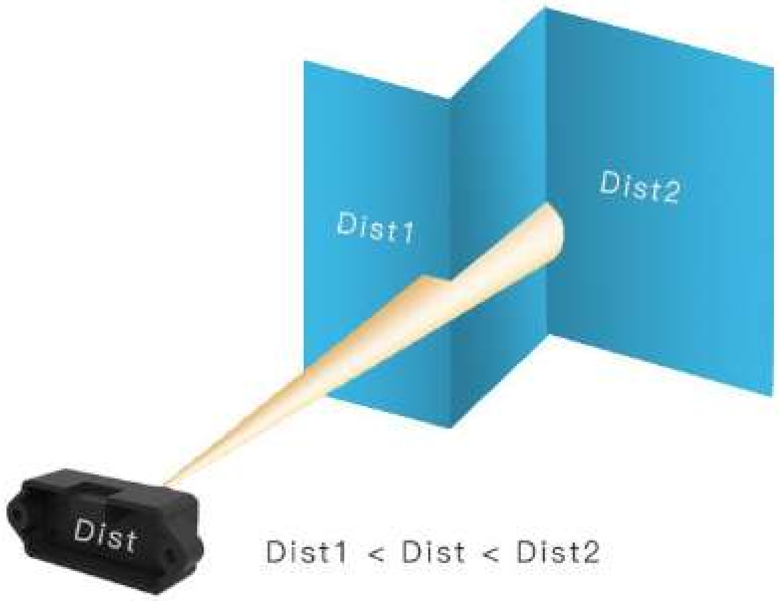


Fig. 3.21: TFmini-S Middle Distance, adapted from the datasheet.

$$
error\,(\mathrm{m}) = \begin{cases} 0.06, & \text{if} \quad 0.01 < d < 6 \\ 0.01d, & \text{otherwise} \end{cases} . \tag{3.4}
$$

The diameter of light spot depends on the FOV of TFmini-S (the term of FOV generally refers to the smaller value between the receiving angle and the transmitting angle), which is calculated by:

$$
d = D\,\tan\beta, \tag{3.5}
$$

where:

d	diameter of the lightspot (m),
D	detecting range (m),
β	value of the receiving angle (°).

Correspondence between the diameter of light spot and detecting range is given in Fig. 3.23 as a linear graph relation, in Fig. 3.24 as a representation of the laser beam, and in Tab. 3.5 with some already calculated values.

The datasheet of the sensor provides the data of Eq. (3.4) and Fig. 3.22, however, that information might be wrong or to be applied in different environments. For this reason, three tests were performed, to determine the truth of these values.

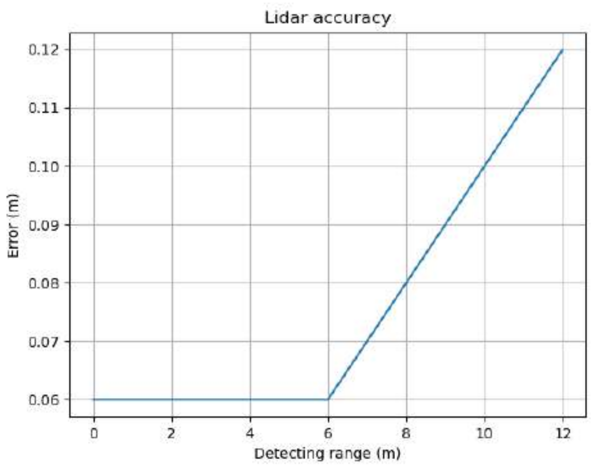


Fig. 3.22: TFmini-S Accuracy.

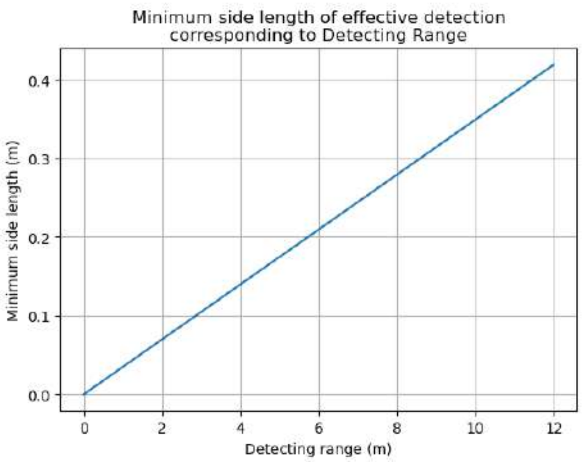


Fig. 3.23: TFmini-S Length of Effective Detection.

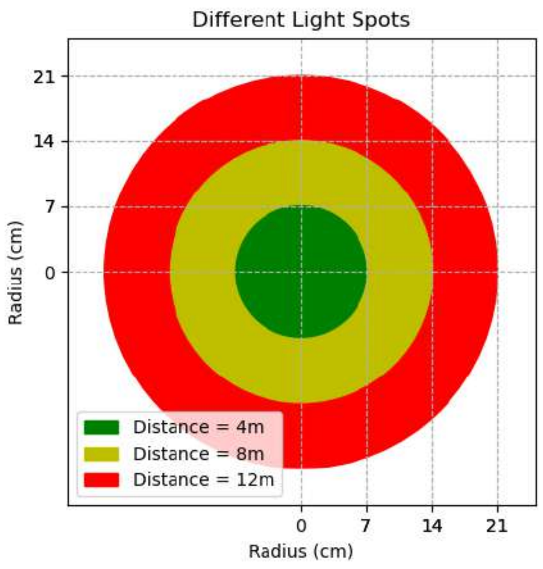


Fig. 3.24: TFmini-S Light Spots.

Tab. 3.5: TFmini-S Minimum Side Length.

Detecting Range	2 m	4 m	6 m	8 m	10 m	12 m
Minimum Side Length	7 cm	14 cm	21 cm	28 cm	35 cm	42 cm

Connections

As seen in Tab. 3.4, the TFmini-S needs a supply voltage of 5V and can communicate through LVTTL Logic Level (3.3V). It is possible to choose between USART (default) and I2C communication. For academic and learning reasons, the USART is going to be used, since the I2C communication was already used for the IMU although it would be possible to use both, but as a student, it is more enrichment to learn how to use both. The configurations used were the default ones and are available in Tab. 3.6 following the physical pinout in Fig. 3.25 and the legend of it in Tab. 3.7.

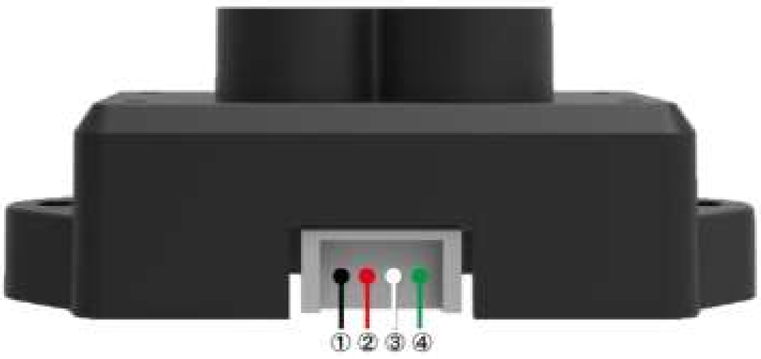


Fig. 3.25: TFmini-S Pinout, adapted from the datasheet.

Tab. 3.6: TFmini-S USART Configuration.

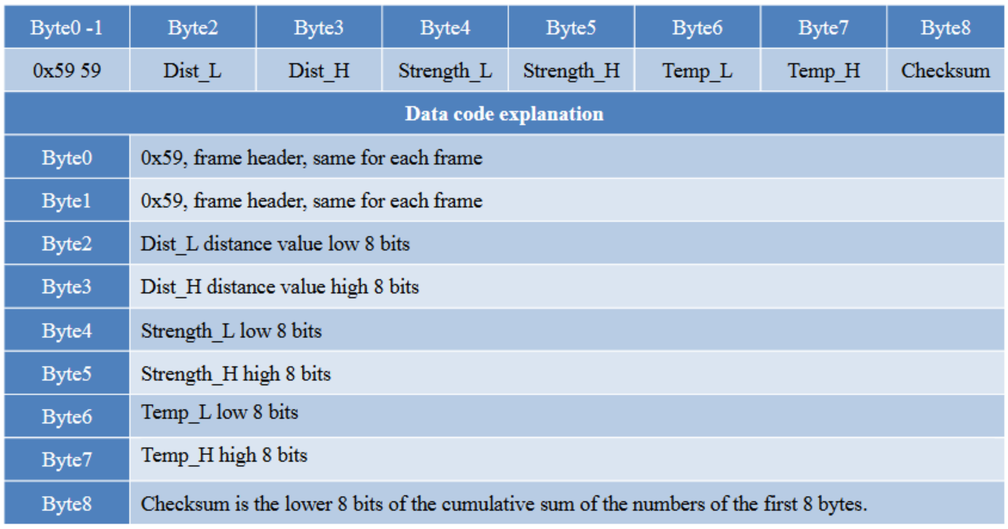
Communication Interface	**UART**
Baud rate	115200
Data bit	8
Stop bit	1
Parity Check	None

Tab. 3.7: TFmini-S Pinout.

Pin Number	**Color**	**Function**
1	Black	GND
2	Red	+5V
3	White	RXD
4	Green	TXD

3.3.3 USART Protocol

The format of the serial port is presented the data output in Fig. 3.26. It contains two frame headers with the value 0x59 followed by Low Byte, High Byte pairs of the distance, strength of the signal and temperature of the sensor, and a checksum of the data to prevent transmission errors.



Byte0 -1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x59 59	Dist_L	Dist_H	Strength_L	Strength_H	Temp_L	Temp_H	Checksum

Data code explanation	
Byte0	0x59, frame header, same for each frame
Byte1	0x59, frame header, same for each frame
Byte2	Dist_L distance value low 8 bits
Byte3	Dist_H distance value high 8 bits
Byte4	Strength_L low 8 bits
Byte5	Strength_H high 8 bits
Byte6	Temp_L low 8 bits
Byte7	Temp_H high 8 bits
Byte8	Checksum is the lower 8 bits of the cumulative sum of the numbers of the first 8 bytes.

Fig. 3.26: TFmini-S USART Data Format, adapted from the datasheet.

- **Dist** (Distance): Represents the output of the distance value detected, with the unit in centimetres by default (later changed to millimetres). This value is interpreted into the decimal value in the range of 0-1200 (cm) or 0-12000 (mm).

- **Strength**: Represents the signal strength with the default value in the range of 0-65535. After the distance mode is set, the longer the measurement distance is, the lower the signal strength will be; the lower the reflectivity is, the lower the signal strength will be.

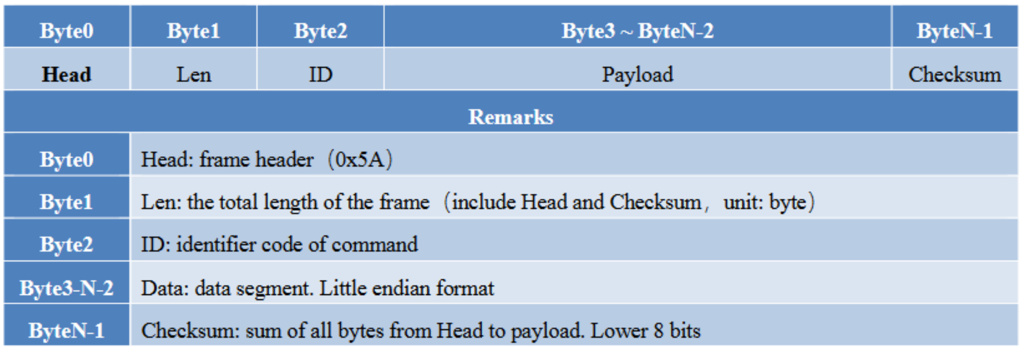
- **Temp** (Temperature): Represents the chip temperature.
  Its given by: $^{\circ}C = \frac{Temp}{8} - 256$

There are some cases where data is abnormal due to the environment where the sensor is working, or due to the characteristics of the object sensor is targeting. Those values and their meaning are showed in Tab. 3.8.

Tab. 3.8: TFmini-S Abnormal Data.

Dist	Strength	Meaning
-1	Other value	Strength ≤ 100
-2	-1	Signal strength saturation
-4	Other value	Ambient light saturation

The sensor has configurable parameters such as output units, frame rate, frequency rate and others. The structure of the byte oriented hexadecimal message to be sent to the sensor is presented in Fig. 3.27.



Byte0	Byte1	Byte2	Byte3 ~ ByteN-2	ByteN-1
Head	Len	ID	Payload	Checksum

Remarks	
Byte0	Head: frame header (0x5A)
Byte1	Len: the total length of the frame (include Head and Checksum, unit: byte)
Byte2	ID: identifier code of command
Byte3-N-2	Data: data segment. Little endian format
ByteN-1	Checksum: sum of all bytes from Head to payload. Lower 8 bits

Fig. 3.27: TFmini-S USART Communication Frame Definitions, adapted from the datasheet.

The function from the LIDAR created library that reads the data from the sensor can be examined in List. 3.4. First a byte array with the size of the frame is created. The next step is to clean the existing buffer because it might have values from other positions if the microcontroller didn't read those values at a higher speed rate than the one at which the sensor works. Afterwards, it confirms that the two first bytes are in fact the header of the frame and assigns the rest of the values to their respective variables.

Listing 3.4: Sample of UART Read from File: LIDAR.cpp

```
void LIDAR::readData(void)
{
  uint8_t frame[TFMINI_FRAME_SIZE];

  Serial2.flush(); // Erase older values accumulated in the serial buffer
  while(!Serial2.available());
    if(Serial2.read() == 0x59)
    {
      if(Serial2.read() == 0x59)
      {
        for(int i=0; i<=6; i++)
        {
          frame[i] = Serial2.read();
        }

        uint16_t distData     = (frame[1] << 8) + frame[0];
        uint16_t strengthData = (frame[3] << 8) + frame[2];
        uint16_t celsiusData  = (frame[5] << 8) + frame[4];

        //This value is interpreted into the decimal value in the range of 0-12000.
        //When the signal strength is lower than 100, the detection is unreliable,
        //TFmini-S will set distance value to -1
        _distance = int(distData);
        //Represents the signal strength with the default value in the rangeof 0-65535
        _strength = float(strengthData);
        //Represents the chip temperature of TFmini-S. Degree centigrade = Temp / 8 -256
        _celsius  = int(celsiusData)/8 - 256;
      }
    }
}
```

To better explain how to change the parameters of the sensor, an example will be presented.

- Change Output format to return distance in mm:

The datasheet provides the following command for this purpose:

0x5A 0x05 0x05 0x06 0x6A

We can confirm that the checksum value is correct by adding all the frames before it, as follows:

$$SU_{16} = 5A_{16} + 05_{16} + 05_{16} + 06_{16} = 6A_{16}. \tag{3.6}$$

The data to be sent to the TFmini-S will be delivered in the order explicit in Tab. 3.9.

Tab. 3.9: LIDAR Example: Set distance data to mm.

Byte 0	Head: 0x5A
Byte 1	Len: 0x05
Byte 2	ID: 0x05
Byte 3	Data: 0x06
Byte 4	Checksum: 0x6A

A practical example of the LIDAR created library that writes data into the sensor can be examined in List. 3.5. Inside the function that sets up the LIDAR sensor, the frames in Tab. 3.9 are sent, in order for it to return the distance values using the millimetre unit.

Listing 3.5: Sample of UART Write from File: LIDAR.cpp

```
const uint8_t OUT_MM[5]      = {0x5A, 0x05, 0x05, 0x06, 0x6A};
// Start communication with LIDAR sensor and set distance output to mm (default cm)
void LIDAR::begin(int baudrate)
{
  Serial2.begin(baudrate, SERIAL_8N1, RXD2, TXD2);
  for(int i=0; i<=4; i++) Serial2.write(OUT_MM[i]);
  delay(2000);
}
```

3.3.4 Developed TFmini-S Sensor Library

During the development of this project, a C++ Library was developed in order to organize the code, making it more efficient and readable. A sample of the .h file is presented in List. 3.6, which includes the LIDAR object and its methods:

Listing 3.6: File: LIDAR.h

```
/*
  LIDAR.h - Library for using TFmini-S Lidar distance sensor.
  Created by Carlos Silva, May, 2020.
*/

#include "Arduino.h"
#define LIDAR_BAUDRATE    115200

class LIDAR {
  public:
    //Constructor
    LIDAR();

    //Methods
    void   readData(void);
    void   begin(int baudrate=LIDAR_BAUDRATE);
    float  getDistance();
    float  getStrength();
    float  getCelsius();

  private:
    float _distance, _strength, _celsius;
};
```

3.3.5 Experimental Results

The datasheet of the sensor provides the data of Eq. (3.4) and Fig. 3.22, however, that information might be wrong or to be applied in different environments. For this reason, three tests were made, to determine the truth of these values. The tests consisted of measuring different objects in a straight line, taking fifty measurements each 10 cm. To make a full characterization of the sensor, the test should be done at least until the manufacturer provided maximum distance, however, the indoor interval of interest is only from 0 to 4 m. The first test targets a simple rectangular cardboard box, that doesn't have any special characteristics when it comes to light reflection. The second test aims a black box, where the reflective characteristics start to matter because black objects tend to absorb light. The last one focuses on a mirror, that has a high reflectivity value. Due to limitations on the movement of the mirror, this test only had a maximum distance of 1 m. To verify the accuracy of the measurements, a measuring tape will be used as a comparison to the real value of distance.

The data acquired from the tests is presented in Appendix C. Using that data it is possible to study the linearity of measurements by calculating the linear regression between the measured distance and the real distance value, which should ideally be a line with slope (m) equal to 1, zero intercept (b) equal to 0 and coefficient of determination (r^2) equal to 1. Looking at Tab. 3.10, its clear to conclude that the most stable and accurate object was the cardboard box, followed by the black box and lastly the mirror. The coefficient of determination is the closest to 1, the slope was also the closest to 1 and the zero intercept was the closest to 0. The opposite happens for the mirror, which has the worst values, especially for the coefficient of determination.

Tab. 3.10: TFmini-S Linear Regression Results.

Parameters	**Cardboard Box**	**Black Box**	**Mirror**
r^2	0.998	0.995	0.879
m	0.968	0.926	0.911
b	0.051	0.118	-0.077

Using the values from Tab. 3.10, it is possible to create an equation for each case. Those equations will be used to apply a correction on the obtained sensor data for different materials and therefore obtain a better approximation on the real value. The cardboard test results will be represented by Eq. (3.7), the black box by Eq. (3.8) and the mirror by Eq. (3.9):

$$x_c^* = 0.968x + 0.051, \tag{3.7}$$

$$x_b^* = 0.926x + 0.118, \tag{3.8}$$

$$x_m^* = 0.911x - 0.077, \tag{3.9}$$

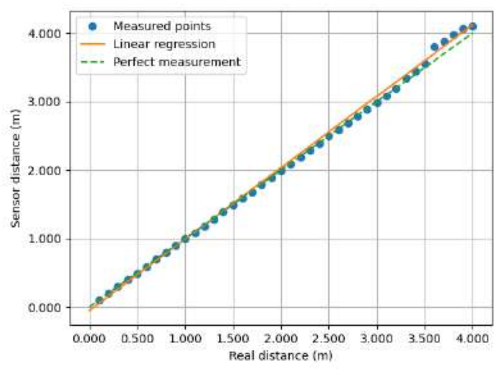
where:

x_c^*	Better approximation on the distance of the cardboard box,
x_b^*	Better approximation on the distance of the black box,
x_m^*	Better approximation on the distance of the mirror,
x	Sensor measured distance value.

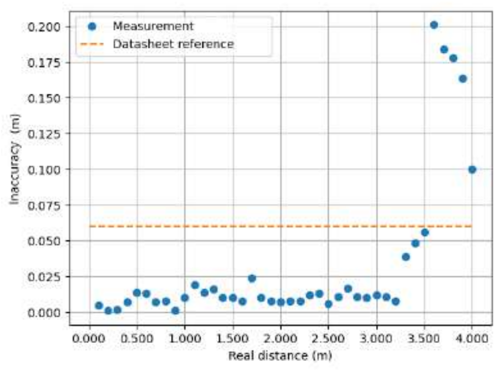
From Fig. 3.28 is possible to analyse the behaviour of the sensor according to the different objects. As for the cardboard, the measurements at the beginning were stable and with a inaccuracy, reaching a maximum of 2.4 cm, below the datasheet's 6.0 cm, however, after the distance of 3.300 m, the inaccuracy starts increasing exponentially, reaching a maximum of 20.1 cm at 3.600 m, well above the datasheet's reference value and then decreasing until the end of the test.

The black box measurements have a similar pattern to the ones registered for the cardboard box, but with an increased error, that would be expected by the characteristics of the material. Both the cardboard and black box tests displayed a systematic ”knee” in the [3.500, 4.00] m interval. This must be due to some manufacturing property of the sensor and it is the main causer of the r^2 disparity on those tests.

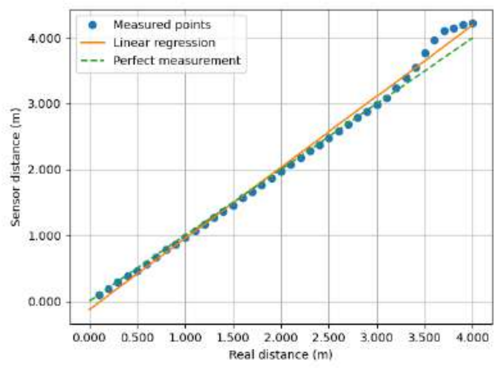
At the mirror, the results were very unstable and inaccurate compared with the ones before. Almost all the inaccuracy was above the datasheet's reference value, reaching a maximum of 38.8 cm before the 1.000 m distance.



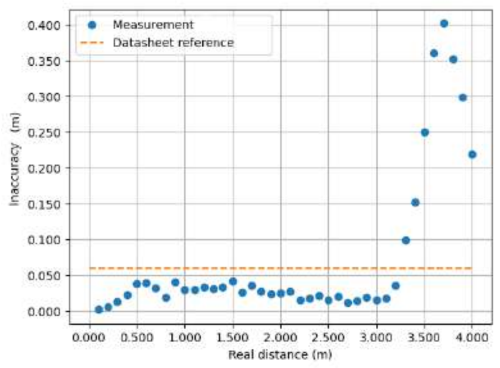
(a) Cardboard box distance test.



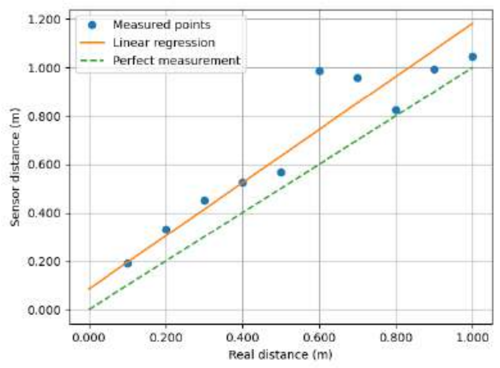
(b) Cardboard box test inaccuracy.



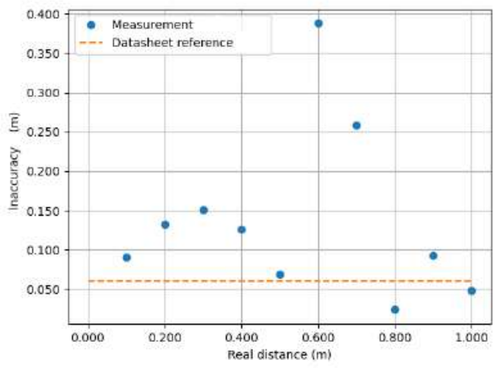
(c) Black box distance test.



(d) Black box test inaccuracy.



(e) Mirror distance test.



(f) Mirror test inaccuracy.

Fig. 3.28: TFmini-S Testing.

3.4 Motion Control

3.4.1 Working Principles

A stepper motor is an electromechanical device that converts electrical pulses into mechanical shaft rotation [28]. There are three basic types of stepping motors:

- Permanent Magnet: magnetized rotor,
- Variable Reluctance: toothed soft-iron rotors,
- Hybrid: Combines aspects of both permanent magnet and variable reluctance technology.

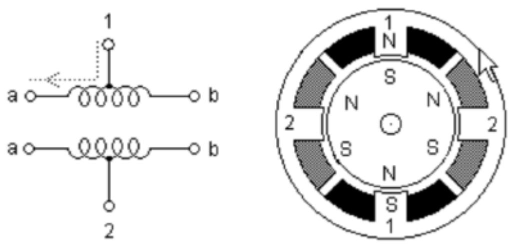
Although they will not be further explored in this document, to keep focus on the main theme. From an electrical and control system point perspective, permanent magnet and hybrid motors may be wound using either unipolar or bipolar windings.

Unipolar Motors

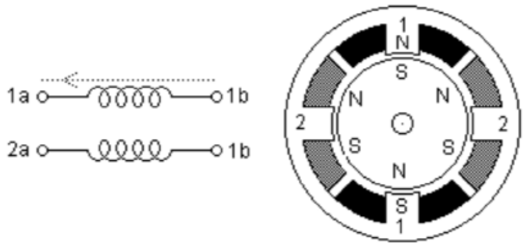
Unipolar stepping motors (Fig. 3.29 a)) are composed of two windings, each one with a center tap. The center taps are either brought outside the motor as two separate wires or connected to each other internally and brought outside the motor as one wire. As a result, unipolar motors have 5 or 6 wires. Regardless of the number of wires, unipolar motors are driven in the same way.

Bipolar Motors

Bipolar stepping motors (Fig. 3.29 b)) are composed of two windings and have four wires and no centre taps. The advantage of not having centre taps is that the current runs through an entire winding at a time instead of just half of the winding. As a result, bipolar motors produce more torque than unipolar motors of the same size.



(a) Unipolar Stepper Motor

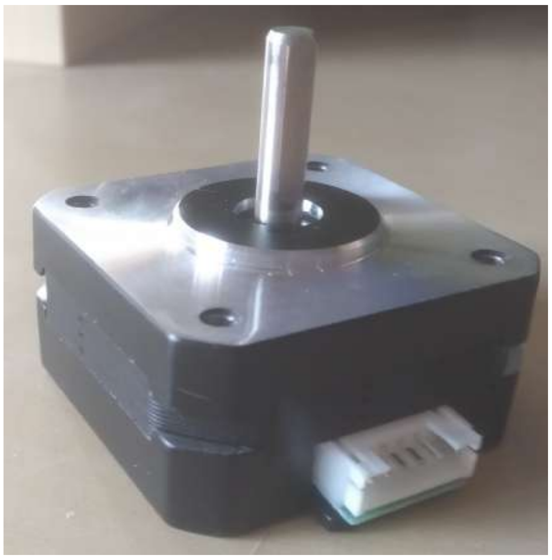


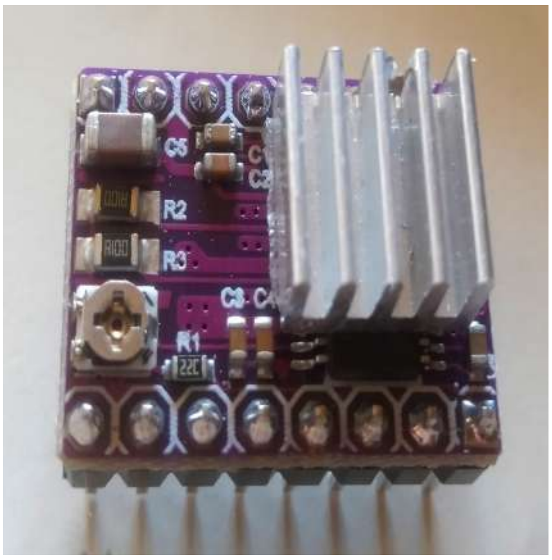
(b) Bipolar Stepper Motor

Fig. 3.29: Stepper Motor Types.

3.4.2 NEMA Stepper Motor

The precision that a stepper motor can offer will be crucial to this project, that's the main reason why this type of motor was chosen over other types of motors. It also needs to be as cheap as possible, light and small as possible without the need for big torque values, which differentiates the 17HS4023 from other NEMA 17 stepper motors. A NEMA 17 stepper motor is a stepper motor with a 1.7 x 1.7 inch (43.18 x 43.18 mm) faceplate, though it can have different lengths. This model is displayed in Fig. 3.30 a).


(a) NEMA 17 17HS4023 Stepper Motor



(b) DRV8825 Stepper Motor Driver

Fig. 3.30: Stepper Motor and Driver.

17HS4023 Specifications

The 17HS4023 is a NEMA 17 small cheap lightweight bipolar stepper motor with 4 wires. The main characteristics as for this project are listed in Tab. 3.11.

Tab. 3.11: 17HS4023 Specifications Table.

Parameters	**Specification**
Rated Voltage	12V
Rated Current	0.7A / Phase
Resistance	4.0 ±10% Ω/Phase
Inductance	3.2 ±20% mH/Phase
Holding Torque	14 N.cm
Step Angle	1.8°±5% / Step
Price	10.60€
Supplier	Amazon.es

A microcontroller such as the ESP32 can't drive directly a stepper motor due to its voltage and current limitations. So in order to solve this obstacle, a specific driver is used, that also simplifies and makes the process of driving the motor more efficient.

3.4.3 DRV8825 Stepper Motor Driver

Made by Texas Instruments, the DRV8825 is a bipolar stepper motor driver IC that contains N-channel power MOSFET's configured as full H-bridges. There only exists a SMD form factor for this driver so despite being an IC, it is mostly know in its "module" form factor because it allows users to test it on a breadboard or easily plug it to a PCB. A proof of this is that many 3D printer companies plug the module to their electronic boards and not the IC itself. Being said, this is the main reason why this project is going to use the module version, which can be seen in Fig. 3.30 b).

DRV8826 Stepper Motor Driver Specifications

A simple STEP/DIR interface allows easy interfacing to controller circuits. Internal shutdown functions are provided for over current, short circuit, under voltage lockout and over temperature. Fault conditions are indicated via the nFAULT pin. The internal micro stepping indexer is able to execute high-accuracy micro stepping without requiring the processor to control the current level. The current regulation is highly configurable, with three decay modes of operation. A low-power sleep mode is included which allows the system to save power when not driving the motor [29]. A basic schematic on how to control the driver is displayed in Fig. 3.31.

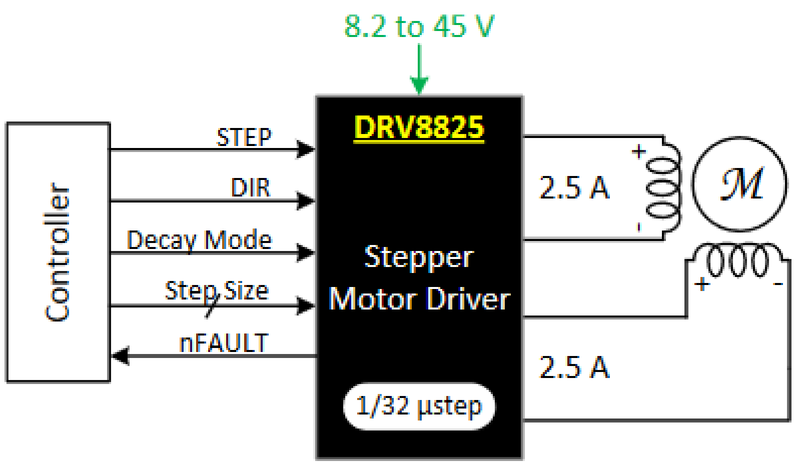


Fig. 3.31: DRV8825 Basic Schematics [29].

This stepper motor driver surpasses all the minimum requirements as for this project. It has a large amount of micro-stepping modes, its operating supply voltage and maximum drive current matches the specifications of the stepper motor used (17HS4023) and has compatible logic levels with the ESP32 microcontroller. In addition, it has a low current consumption as is affordable. These characteristics are displayed in detail in Tab. 3.12.

Tab. 3.12: DRV8825 Specifications.

Parameters	**Specification**
Microstepping Modes	Full Step
	Half Step
	Quarter Step
	1/8 Step
	1/16 Step
	1/32 Step
Operating Supply Voltage	8.2 - 45V
Maximum Drive Current	2.5A
Logic Level Input	2.2 - 5.25V
Step Frequency	0 - 250kHz
Current Consumption	5 - 8mA
Price	1.32€
Supplier	LCSC.com
Price of Module	2.40€
Supplier of Module	Amazon.es

Micro stepping is a method of dividing the full manufacturer stepper motor step size, in this case 1.8°, into smaller values, using power electronics or programming techniques to do so. This will increase the precision of the stepper motor. The MODE0, MODE1 and MODE2 pins are used to configure the stepping format as shown in Tab. 3.13.

Tab. 3.13: DRV8825 Steps Modes.

MODE2	MODE1	MODE0	STEP MODE
0	0	0	Full Step
0	0	1	Half Step
0	1	0	Quarter Step
0	1	1	1/8 Step
1	0	0	1/16 Step
1	0	1	1/32 Step
1	1	0	1/32 Step
1	1	1	1/32 Step

Current Regulation

The current through the motor windings is regulated by a fixed-frequency PWM current regulation, aslso known as current chopping. When an H-bridge is enabled, current rises through the winding at a rate dependent on the DC voltage and inductance of the winding. Once the current hits the current chopping threshold, the bridge disables the current until the beginning of the next PWM cycle. The current waveforms can be seen in Fig. 3.32. The measured current will be 0.7 times the current limit (since both coils are always on and limited to approximately 70% of the current limit setting in full-step mode).

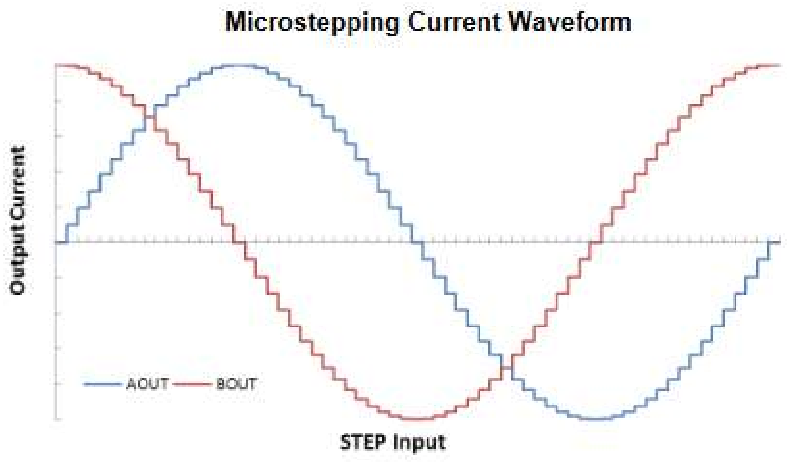


Fig. 3.32: DRV8825 Micro stepping Waveform [29].

The PWM chopping current is set by a comparator which compares the voltage across a current sense resistor connected to the xISEN pins, multiplied by a factor of 5, with a reference voltage. The reference voltage is input from the xVREF pins. The reference voltage is scaled by an internal DAC (Digital to Analog Converter) that allows fractional stepping of a bipolar stepper motor. The full-scale (100%) chopping current is calculated in Eq. (3.10) but since the carrier board current sense resistors are 0.100 Ω, it can be simplified to Eq. (3.11). V_{REF} can be set in a trimmer potentiometer placed on top of the board.

$$I_{CHOP} = \frac{V_{(REF)}}{5 \times R_{ISENSE}}, \tag{3.10}$$

$$I_{CHOP} = V_{REF} \times 2. \tag{3.11}$$

However, some considerations need to be taken. The 17HS4023 stepper motor is rated to a maximum of 0.7 A per phase and the driver is rated to a maximum of 2.5 A per phase. We will be limited to the smallest rated current, that is the one of the stepper motor, so the maximum reference voltage can only be 0.35 V in full-scale chopping current.

The working zones of the system using the board values for current sense resistors are illustrated in Fig. 3.33. It is also possible to notice that the current sense resistor is possibly wrongly dimensioned, not taking advantage of the full size of all the possible $V_{(ref)}$ values for precise current limitations. Based on Eq. (3.10), setting $V_{(ref)}$ and I_{CHOP} to their maximum values, 3.3 V and 2.5 A, respectively, we can conclude that the best R_{ISENSE} value is 0.264 Ω. The closest commercially available resistor value in LCSC.com online electronics store is 0.260 Ω.

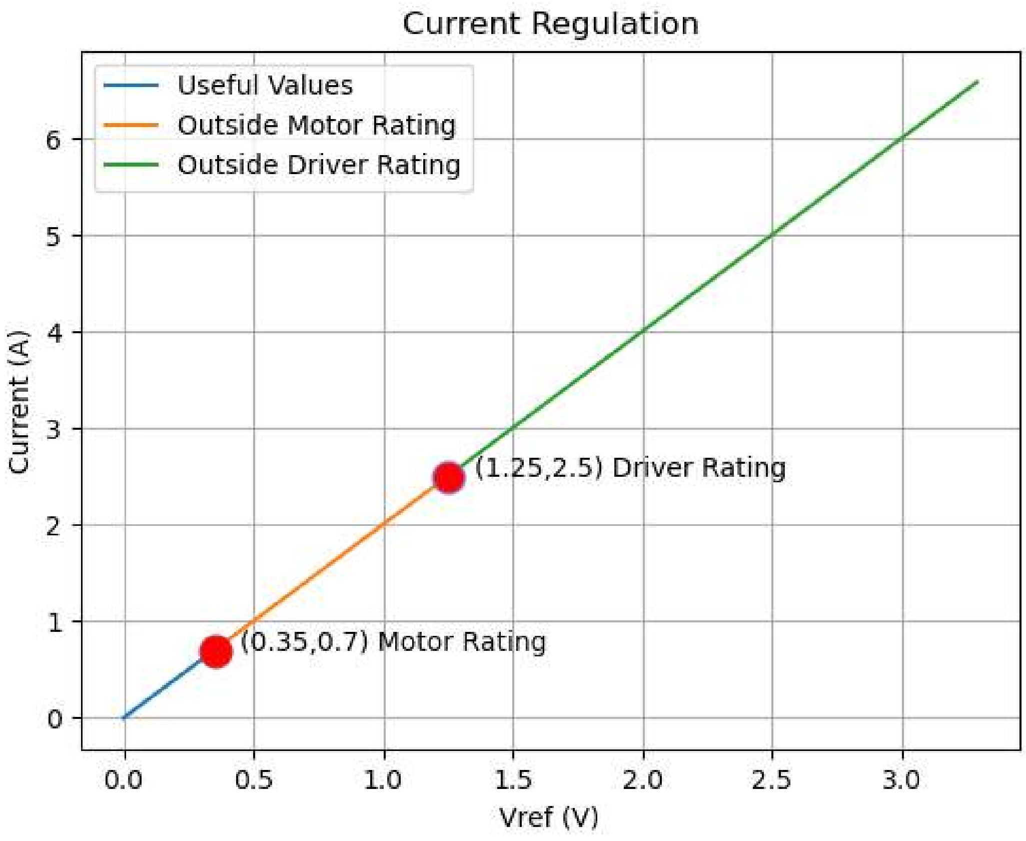


Fig. 3.33: DRV8825 Current Regulation.

This carrier board uses low-ESR ceramic capacitors, which makes it susceptible to destructive LC voltage spikes, especially when using power leads longer than a few inches. Under the right conditions, these spikes can exceed the 45 V maximum voltage rating for the DRV8825 and permanently damage the board, even when the motor supply voltage is as low as 12 V. One way to protect the driver from such spikes is to put a large (at least 47 uF) electrolytic capacitor across motor power (VMOT) and ground somewhere close to the board as shown in Fig. 3.34.

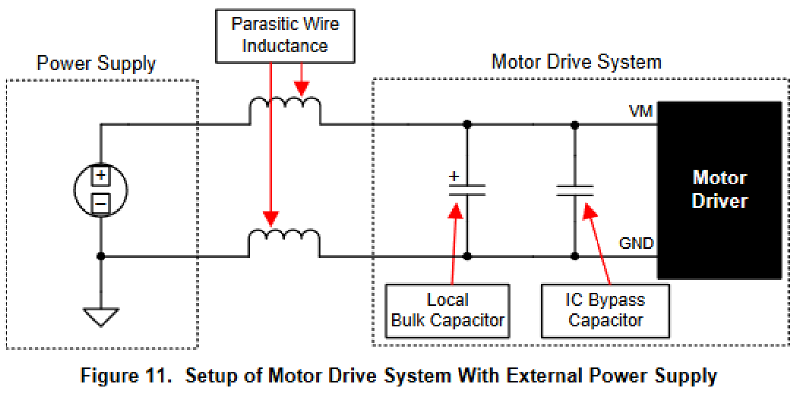

Parasitic Wire Inductance
Power Supply
Motor Drive System
VM
GND
Motor Driver
Local Bulk Capacitor
IC Bypass Capacitor
Figure 11. Setup of Motor Drive System With External Power Supply

Fig. 3.34: DRV8825 Setup External Power Supply [29].

3.4.4 Developed DRV8826 Stepper Motor Driver Library

During the development of this project, a C++ Library was developed in order to organize the code, making it more efficient and readable. A sample of the .h file is presented in Lst. 3.7, which includes the DRV8825 object and its methods:

Listing 3.7: File: DRV8825.h

```
/*
  DRV8825.h - Library for using DRV8825 stepper driver.
  Created by Carlos Silva, May, 2020.
*/

#include <Arduino.h>

class DRV8825
{
  public:
    DRV8825();
    void begin(int enablePin, int dirPin, int stepPin, int M0, int M1, int M2);
    void setMode(int steps);
    void takeStep();
    void changeDir();

  private:
    int _enablePin;
    int _dirPin;
    bool _dir;
    int _stepPin;
    int _M0;
    int _M1;
    int _M2;
};
```

Chapter 4

Hardware Implementation

4.1 Breadboard

During the initial phase of the development of this project and for initial testing purposes, a breadboard was used in order to test the components and develop the programming libraries for each one of them. It wouldn't be wise to jump immediately to the manufacturing of a PCB without completing this first tests, to make sure everything is working accordingly. It contains the components and sensors described in the previous Chapter, namely, the ESP32 microcontroller, the BNO005 Inertial Measurement Unit and the DVR8825 Stepper Motor Drivers. A single 100uF capacitor connects to the 9V power rail and to a 5V/3.3V Power Supply. It also had wires that would connect to the stepper motors, to the TFmini-S LIDAR and to a 12V Power Supply that was regulated to 9V. The Fig. 4.1 displays that same system (the stepper motors and the TFmini-S were already in the final structure).

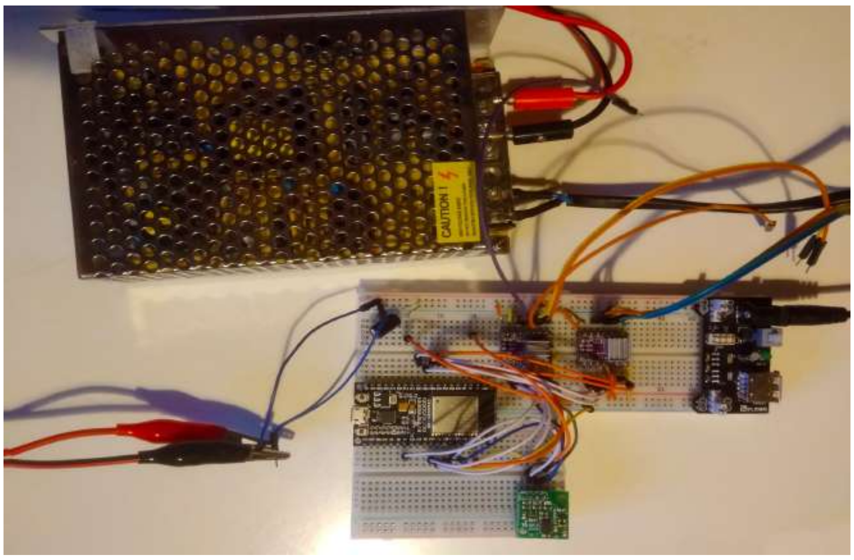


Fig. 4.1: Breadboard circuit used for initial testing purposes.

4.2 Mechanical Structure

The structure used in this project was adapted from MaketBot Thingiverse, its licensed under the Creative Commons - Attribution license and can be seen in Fig. 4.2. The changes made are related to the increase of holes size. Its dimensions are 132x132x140 (width, length, height) in mm and the total weight is around 359.6 g, calculated in Tab. 4.1. The structure rotates 180 ° on the represented rotations, having support for the TFmini-S and for the stepper motors. One of the motors goes on the side to generate an elevation on the distance sensor, and the other goes on the base to provide an azimuth movement. It also contains two limit switches that will be used to calibrate the orientation of the sensor. Finally, this structure is screwed into an acrylic base, to provide a stable position, less sensitive to the vibration of the stepper motors and to group it with the eletronic board.

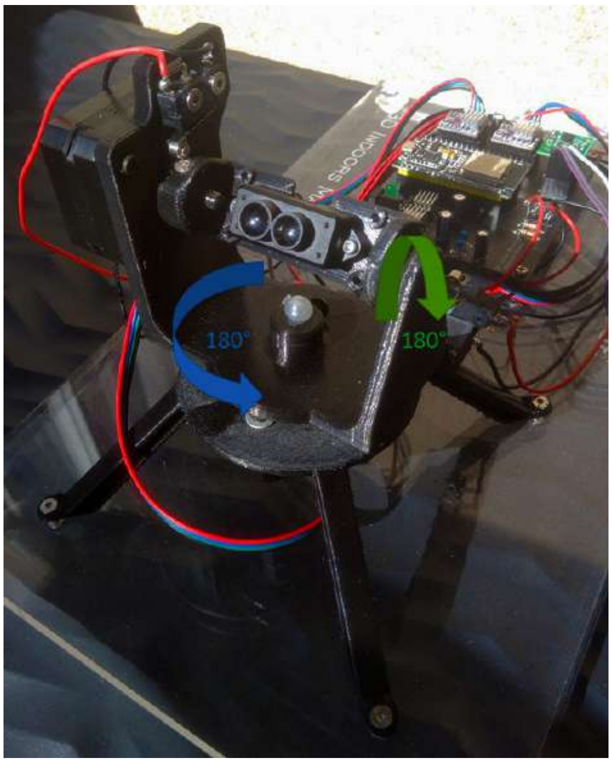


Fig. 4.2: Mechanical Structure.

Tab. 4.1: Weight of the Mechanical Structure.

Part	***Weight*** (g)
Micro Switches	2.8x2.0 = 5.6
TFmini-S	18.0
3D Structure	72.0
Stepper Motors	132.0x2.0 = 264.0
Total = 359.6 g	

4.3 Power Considerations

In order to choose the correct power supply and voltage regulators, a theoretical average current and voltage analysis of the all circuit needs to me made. Otherwise the circuit could have have power related problems like lack of power or components getting too hot and probably damaged also. This might be also useful to debug hardware related problems. The current consumption of each individual component separated by voltage levels is shown in Tab. 4.2, Tab. 4.3 and Tab. 4.3.

Tab. 4.2: Power Consumption at 3.3 V Powered Devices

Part	***Current Consumption*** (mA)
ESP32	125
BNO055	12
DRV8825	8x2 = 16
Total = 153 mA	

Tab. 4.3: Power Consumption at 5.0 V Powered Devices

Part	***Current Consumption*** (mA)
AMS1117-3.3V	153
TFmini-S	140
LED	11
Total = 304 mA	

Tab. 4.4: Power Consumption at 9.0 V Powered Devices

Part	***Current Consumption*** (mA)
AMS1117-5.0V	304
Stepper Motors (max.)	700x2 = 1400
Total: 1704 mA	

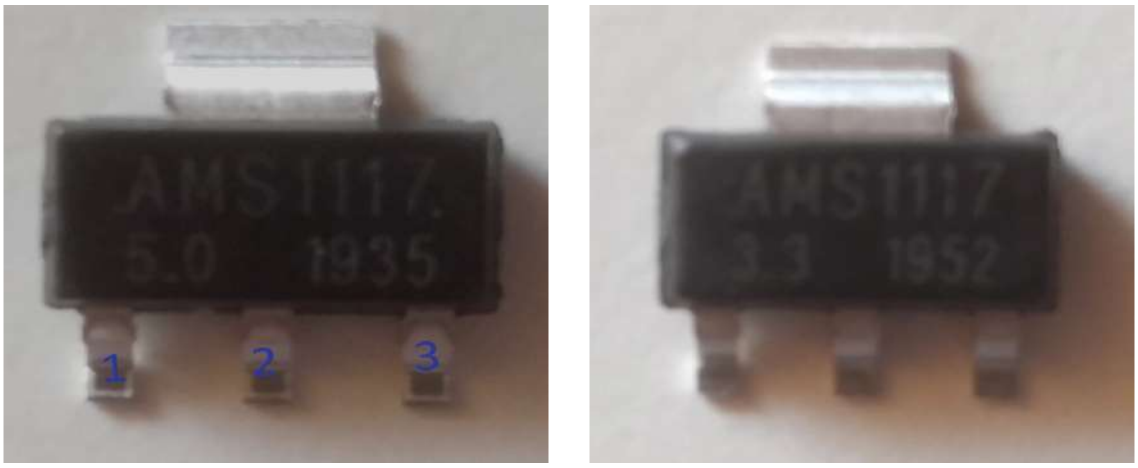
The power consumption at 9V and theoretical current of 1.704 A, is:

$$P_T = V_I I_T = (9)(1.7043) \approx 15.34\,\mathrm{W}. \tag{4.1}$$

The most import conclusions about Tab. 4.2, Tab. 4.3, Tab. 4.3 is that the 3.3 V voltage regulator needs to be able to handle at least 153 mA, the 5.0 V voltage regulator needs to be able to handle at least 304 mA and the external 9 V power supply needs to be able to handle at least 1.704 A or 15.34 W. Most of the components listed have current spikes especially during initialization, this will also be taken into consideration.

AMS1117 Voltage Regulator

The AMS1117, made by Advanced Monolithic Systems is a low cost, low drop-out voltage regulator that can be found with many different voltage level values, however, as for this project, the only interest is in the 5 V and 3.3 V versions. These can be seen in Fig. 4.3, the pinouts are the same for both of them.



(a) AMS1117 5.0V (b) AMS1117 3.3V

Fig. 4.3: AMS1117 Voltage Regulator.

In Fig. 4.3,

$Pin\,1$	Ground (GND)
$Pin\,2$	Output Voltage (V_{out})
$Pin\,3$	Input Voltage (V_{in})

There are some main considerations to take about voltage regulators. The first one is the dropout voltage, that is minimum voltage between V_{IN} and V_{OUT} in order for it to work properly. The second is the current limit, that is the maximum current that the regulator can handle. The third is the minimum load current, that is the minimum current that the regulator must be supplying to the load in order for it to work properly. And the final one is the maximum junction temperature. The regulator works on a power dissipation principle, by other words, it can get really hot, and that is maximum temperature that the regulator can be operating at. The detailed specification of these values are displayed in Tab. 4.5. Its possible to conclude from Eq. (4.6) and Eq. (4.7) that they're both working on their respective working zones, above their dropout voltage, above their minimum current and below their maximum current, and their temperature is below the maximum. However, just as good measure, heat sinks will be used on this regulators. They will help to dissipate heat better onto the air, keeping the regulators cooler, increasing their life span and making them more robust to rises of current.

Tab. 4.5: AMS Specifications Table.

Parameter	***Value***
Dropout Voltage V_{DO}	1.1 V
Current Limit I_{MAX}	1100 mA
Minimum Load Current I_{MIN}	5 mA
Total Thermal Resistance (junction-to-ambient) R_T	45 °C/W
Maximum Junction Temperature T_J	125 °C

The voltage drop (V_D) of the AMS1117 is given by:

$$V_D = V_{IN} - V_{OUT},\ V_D > V_{DO}. \tag{4.2}$$

The output current (I_{OUT}) of the AMS1117 satisfies the condition:

$$I_{MIN} < I_{OUT} < I_{MAX}. \tag{4.3}$$

The power dissipation (P_D) of the AMS1117 can be computed as follows:

$$P_D = (V_D)(I_{OUT}). \tag{4.4}$$

The maximum junction temperature (T_J) with ambient temperature (T_A) considered as 25 °C is:

$$T_J = T_A + P_D R_T. \tag{4.5}$$

For the AMS1117-3.3V, the above quantities are:

$$\begin{cases} V_D = 5 - 3.3 = 1.7\,\text{V},\ 1.7 > 1.1 \\ I_{OUT} = 0.153\,\text{A},\ 0.005 < 0.1533 < 1.100 \\ P_D = (1.7)(0.153) \approx 0.26\,\text{W} \\ T_J = 25 + (0.26)(45) \approx 38.00\,^{\circ}\text{C} \end{cases}. \tag{4.6}$$

For the AMS1117-5.0V, the same quantities are:

$$\begin{cases} V_D = 9 - 5 = 4\,\text{V},\ 4 > 1.1 \\ I_{OUT} = 0.304\,\text{A},\ 0.005 < 0.3043 < 1100 \\ P_D = (9 - 5)(0.304) \approx 1.22\,\text{W} \\ T_J = 25 + (1.22)(45) \approx 79.90\,^{\circ}\text{C} \end{cases}. \tag{4.7}$$

4.4 Printed Circuit Board

During the development of this project, the need to design a PCB increased every time a new component was added to the breadboard. The ”spaghetti” of wires was causing some bad connections and trying to correct them would sometimes cause even more problems. The solution found to get rid of most breadboard wiring, improve organization, solidify connections and use a single power supply, was the use a compact custom PCB for the project, where most wires are replaced by traces on a two sided board and the others are easily connected using pin sockets and headers with the respective pinout markings, making it easy to be taken out and in again. Other advantages are the improvement of the overall aesthetics, the easier transportation and better implementation for real use (less susceptible to movement changes and contact by others).

The first option is the use of a modular board, that will make use of the modules already shown, together with all the other necessary components. The modules, motors and switches will be plugged into the board using some pin sockets and headers. The second option is the use of a totally embedded board, that will make use of the individual IC’s. Unfortunately, this option is not possible due to the lack of conditions, materials and tools to solder SMD components. Maybe in the future this will be possible.

There are many different software’s to design PCBs, some of the most popular are Eagle, Altium and KiCad. The one choose for this project was KiCad because its free and open-source, and also because the author already has some experience with it.

4.4.1 PCB Design using KiCad

KiCad is a multi-platform open source software suite for PCB electronics design [30]. It is user-friendly, making it a great software for both experienced and non-experienced PCB designers and can be divided into three major components:

- KiCad - The main application and project manager.
- Eeschema - The schematic capture editor.
- Pcbnew - The PCB layout program.

KiCad holds the main organization of the whole project, all the general tools and option are available there. Eeschema is used to design the circuit in a schematic format and to the select the circuit components and how they are connected. Pcbnew is the final major tool, where the physical design of the PCB is made, such as board and tracks dimensions. The complete step to step guideline used is available in Appendix D.

To design and print a PCB using this software, the following 12 steps had to be taken:

1. **Select the components**.
2. **Annotate the schematic**.
3. **Verify the electrical connections**.
4. **Assign footprints**.
5. **Generate the netlist**.
6. **Define the board size**.
7. **Organize the components**.
8. **Add tracks**.
9. **Verify the layout**.
10. **Check the 3D View**.
11. **Generate the gerber files**.
12. **Send the gerber files to a manufacturer**.

4.4.2 Printing the PCB

Different manufacturers accept different types of files to be sent to them, in order to start producing the board. However, most of them accept what are so called ”gerber files” that are a group of files used by PCB industry software, to describe the printed circuit board specifications such as: copper layers, solder mask, legend and drill data and size. The board manufacturer will have some limitations related many aspects of the board, one of them being the minimum track width. The manufacturer chosen to produce the PCBs was JLCPCB. The most common color of PCBs is green, however, for aesthetic reasons, a black version was also produced by them and both are displayed in Fig. 4.4. The final process related to the board, was to solder all the necessary components on the board and plug-in the modules. The final result can be seen in Fig. 4.5.

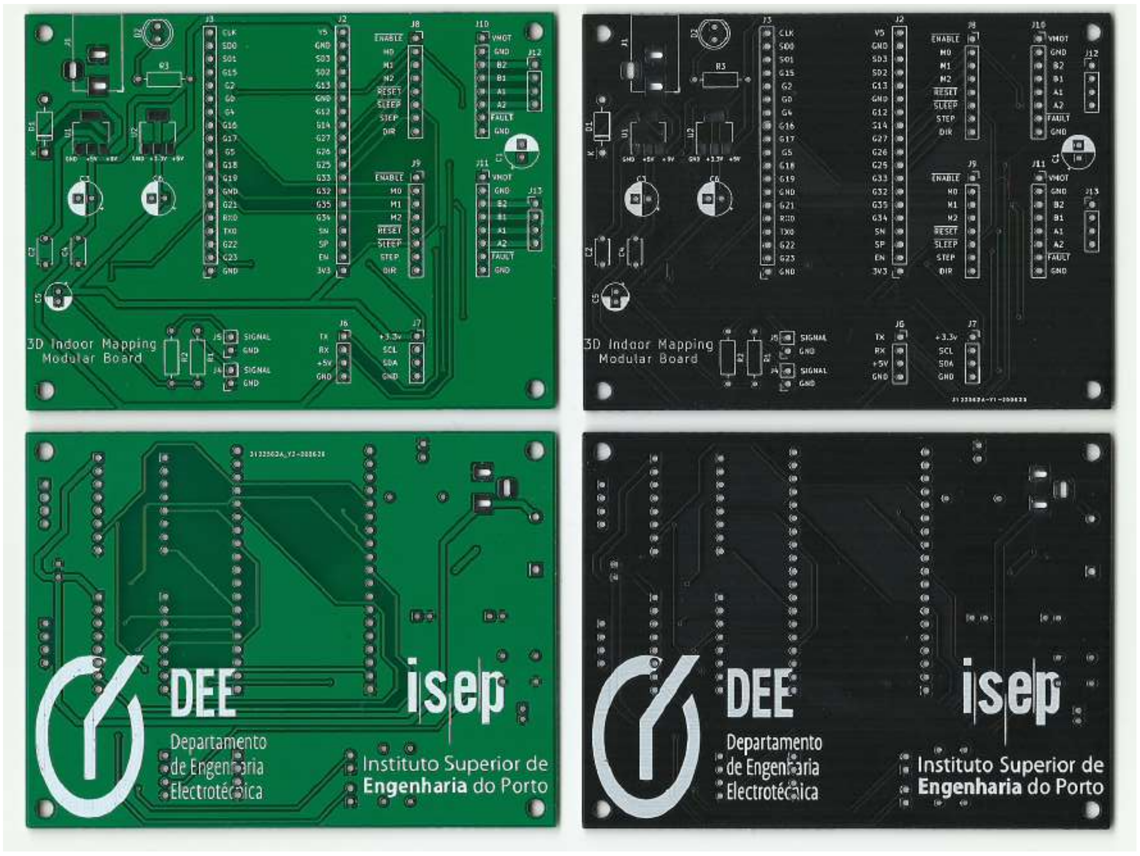


Fig. 4.4: PCB Boards (left - green version; right - black version).

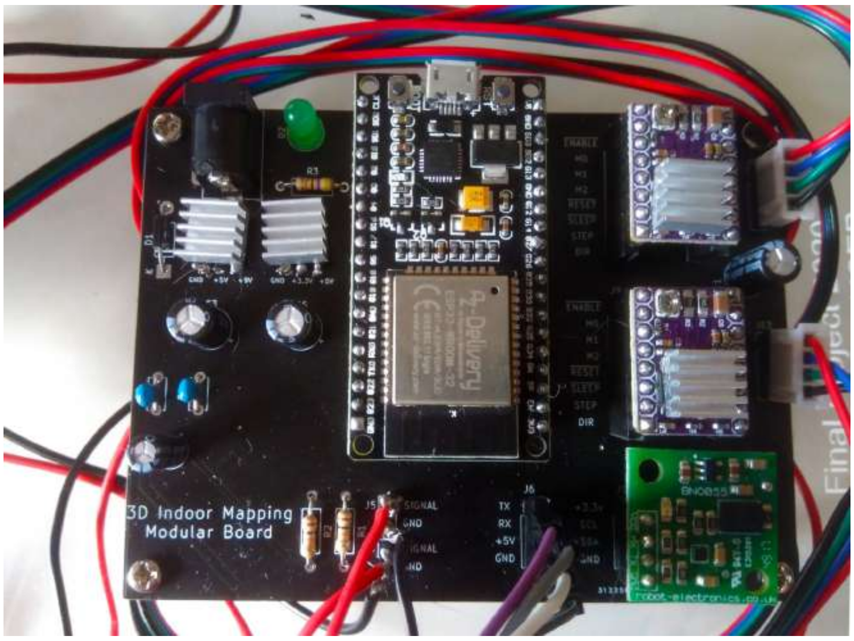


Fig. 4.5: Final Soldered Board.

4.5 Financial Analysis

The budget of the whole project is presented in Tab. 4.6 and does not include costs with transportation, minimum supplier required quantities, costs related to a 3D printer besides the filament or costs related to the soldering of components to the board. On Fig. 4.6 its possible to evaluate the absolute cost of each hardware implementation sector and in Fig. 4.7 its possible to see how each sector cost influences the total budget. Both those figures where grouped by sectors: ”LIDAR” contains the TFmini-s, ”Steppers” contains the NEMA 17 17HS4023 and the DRV8825’s, ”IMU” contains the BNO055, ”Power” contains the power supply, ”uController” contains the ESP32, ”PCB” contains all the components related to the PCB that have not been mentioned before and ”Filament” contains the filament for the 3D printer. Looking to the charts it is clear to conclude that the TFmini-S was the most expensive component of the all project but it is also one of the most crucial ones. One of the challenges for a future development might be the reduction of price by reducing the size of some sector or even find other alternatives to even delete them.

Tab. 4.6: Total Cost of the Scanner

Part	***Supplier***	***Cost (Uni.) €***	***Qt.***	***Cost (Tot.) €***
Electrolytic Capacitor 100uF	LCSC.com	0.05	3	0.15
Heat sink		0.05	2	0.10
Ceramic Capacitor 0.1uF		0.04	2	0.08
Power Jack		0.05	1	0.05
AMS1117-5.0V		0.04	1	0.04
AMS1117-3.3V		0.04	1	0.04
Green LED 5mm		0.03	1	0.03
Diode 1N4001		0.02	1	0.02
Electrolytic Capacitor 1uF		0.01	1	0.01
Resistor 470Ω		0.01	1	0.01
Resistor 10kΩ		0.01	2	0.02
TFmini-S	Amazon.es	35.99	1	35.99
NEMA 17 17HS4023		10.60	2	21.20
Power Supply 9 V 3 A | 27 W		11.99	1	11.99
ESP32		9.99	1	9.99
DRV8825		2.40	2	4.80
3D Filament (g)		0.02	72	1.44
BNO055 Module	botnroll.com	19.90	1	19.90
Mini Microswitch with Pulley	Aquario.pt	0.94	2	1.88
PCB	JLCPCB.com	0.35	1	0.35
Total Cost = 108.08 €				

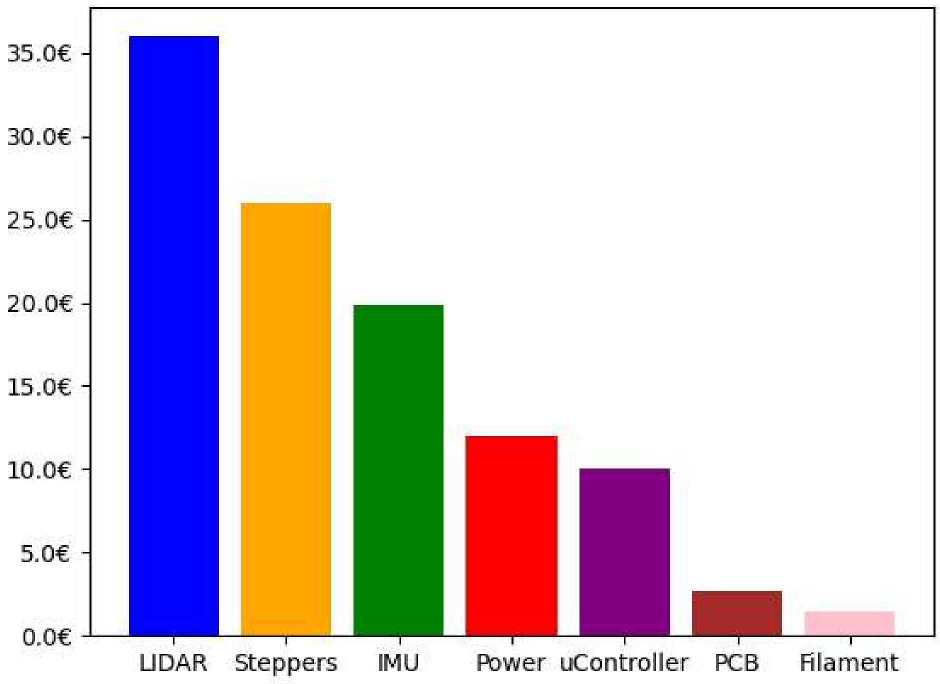


Fig. 4.6: Financial Analysis: Comparison by Absolute Cost.

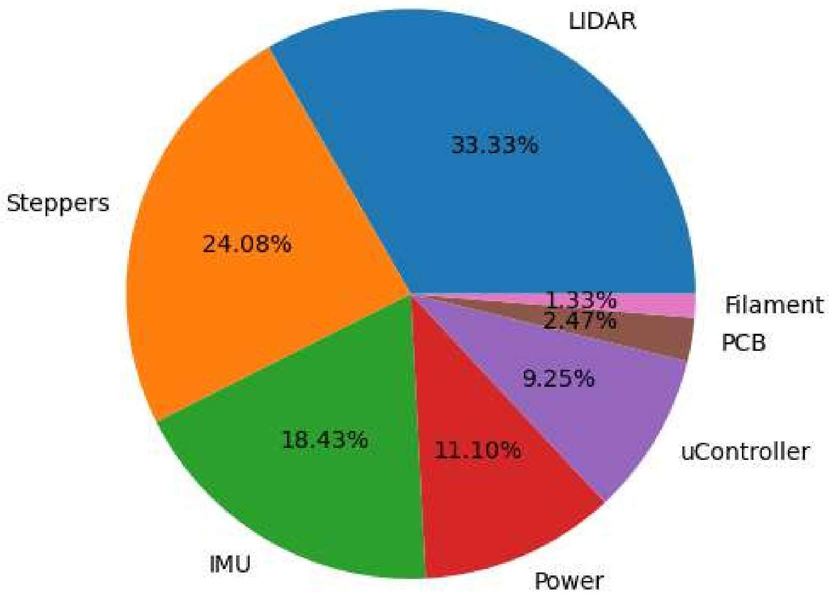


Fig. 4.7: Financial Analysis: Comparison by Relative Cost.

Chapter 5

Software and User Interface

5.1 Programming Environment

The final results might be influenced by the capabilities of the machine running the software and the tools used to develop the application. The elected OS was the Linux ***Ubuntu*** distribution because it is open-source, popular, with a lot of online support and it is mostly stable. The programming language chosen was Python, version 3.8 (the last stable version when the project started to be developed, around February) because it is a high-level script interpreted language that allows quick prototyping of software and also has a great online support with many available free libraries. The machine main specifications and the programming tools used to develop the application are described in Tab. 5.1.

Tab. 5.1: Programming Machine Specifications.

Operating System (OS)	**Ubuntu 18.04.4 LTS**
OS Type	64-bit
Processor	AMD FX-6300 (6 Cores)
Memory	8 GiB
Graphics Card	AMD R7 260x (2 GB)
Programming Language	Python 3.8
Integrated Development Environment (IDE)	Visual Studio Code

5.2 Data Communication

Client Server Model

Network programming uses the Client-Server Model (displayed in Fig. 5.1). A Client is a program or process that initiates the communication, and a Server is a program or process who waits for the communication to start. There are some cases where a program can be both a client and a server [31]. In the simplest case, a Client program sends the request to the Server and the Server sends the response. Network programming allows Interprocess Communication. It means it involves writing computer programs that communicate with other program across a computer network and makes use of the Client-Server model, so it is also Client-Server Programming.

Sockets

Network programming makes use of sockets for interprocess communication, where a socket acts as the endpoint of the interprocess communication. The biggest advantage about using them, is that the connection stays always open until it is explicitly closed. The data transmission is displayed in Fig. 5.2. Python already contains a standard library named ***socket*** that will simplify this process and will be used to communicate wirelessly to the microcontroller in the most efficient way. The algorithm to create a socket server using this library is as follows:

1. **Import** the standard socket library that the Python 3 installation already contains.

2. **Define the server parameters** such as PORT number and data format. The choice of the PORT number can be chosen randomly as long as it doesn't conflict with any registered or used PORTs. The format chosen to send the data was UTF-8 because it also includes Portuguese characters.

3. **Create the Server**. The constant ***Socket.AF_INET*** defines the IPV4 and the constant ***Socket.SOCK_STREAM*** defines TCP. This way it will use the TCP/IP protocol to send and receive data. Next it will bind the address given by the PORT and IP and start listening to clients.

4. Every time a client connects to this server, the server will **accept the connection** and save it, together with the IP of the client. It will read the message until a maximum of 2048 bytes and decode it as UTF-8.

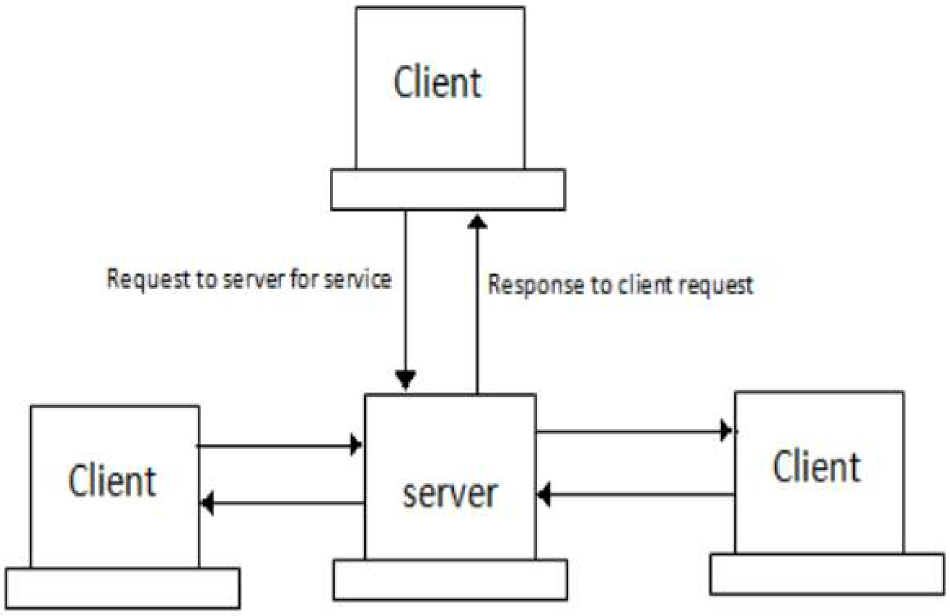


Fig. 5.1: Server Client Model [31].

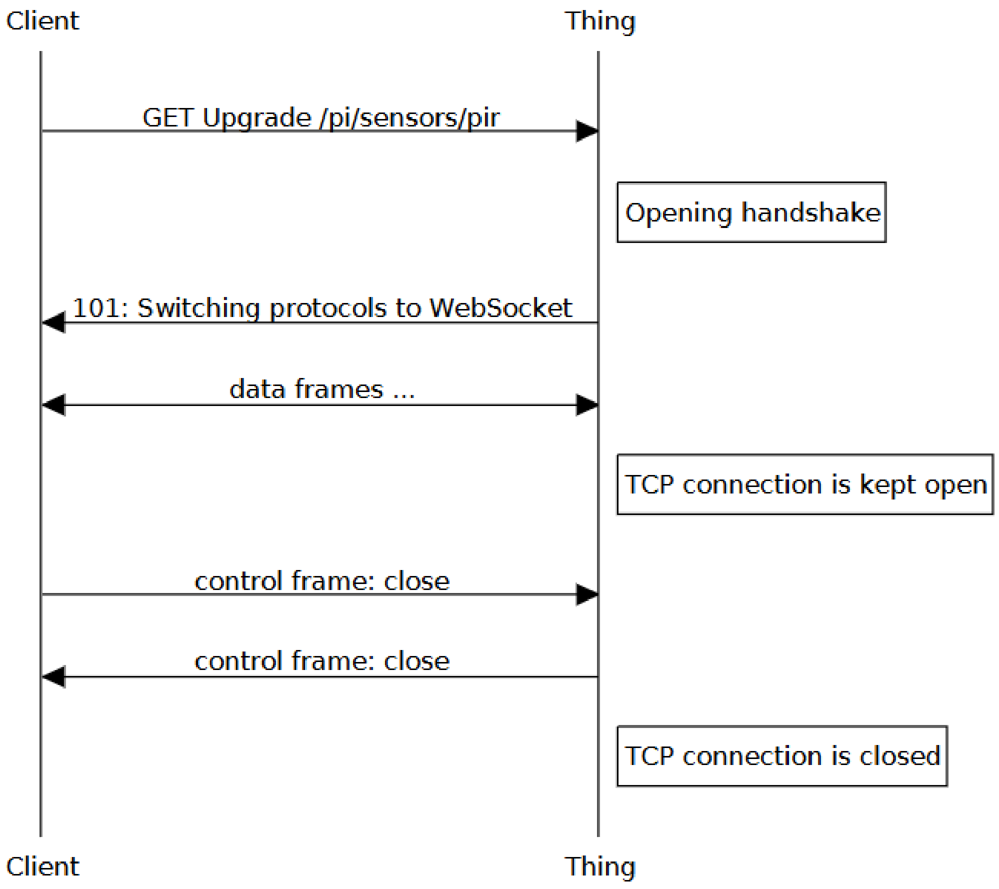


Fig. 5.2: Sockets Protocol [32].

5.3 Functions and Threads

”The term thread is shorthand for thread of control, and a thread of control is, at its simplest, a section of code executed independently of other threads of control within a single program” [33]. Every program has at least one thread, that is the main code. This is mostly useful for multi-tasking software such as the one used in this project. The threads can be daemon or non daemon. While a non-daemon thread blocks the main program from closing if the thread is still running, a daemon thread runs without blocking the main program from exiting. Python already contains a standard library called named ***threading*** that will simplify this process and improve the speed and efficiency of the GUI, computing several functions in an apparent parallel way. The thread is created by attributing a certain function to it and executing its start method.

5.4 Point Cloud Visualization

The Visualization Toolkit (VTK) is an open source toolkit for scientific data processing, visualization, and data analysis [34]. It was useful in this project to visualize the data that comes from the microcontroller, in 3D. The direction of the VTK visualization pipeline is represented by Fig. 5.3 in a simplified way. The simplification of the setup algorithm to create the cloud point is as follows:

1. Define the source - provides the initial data input.
2. Set the mapper - converts the data into virtual objects.
3. Create the actor - adjusts the visual properties of the object.
4. Apply the renderer - renders the objects to be displayed in a window.
5. Define the render window - contains the window where the objects will be visualized.
6. Set the interactor - contains the controls over the window, like camera rotation.

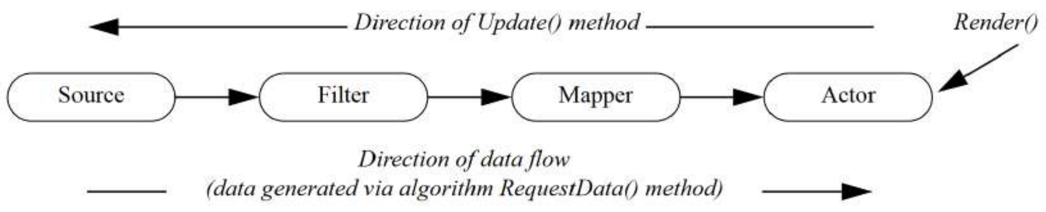

Direction of Update() method
Render()
Source
Filter
Mapper
Actor
Direction of data flow
(data generated via algorithm RequestData() method)

Fig. 5.3: VTK Test: Conceptual overview of pipeline execution [35].

5.5 Graphical User Interface

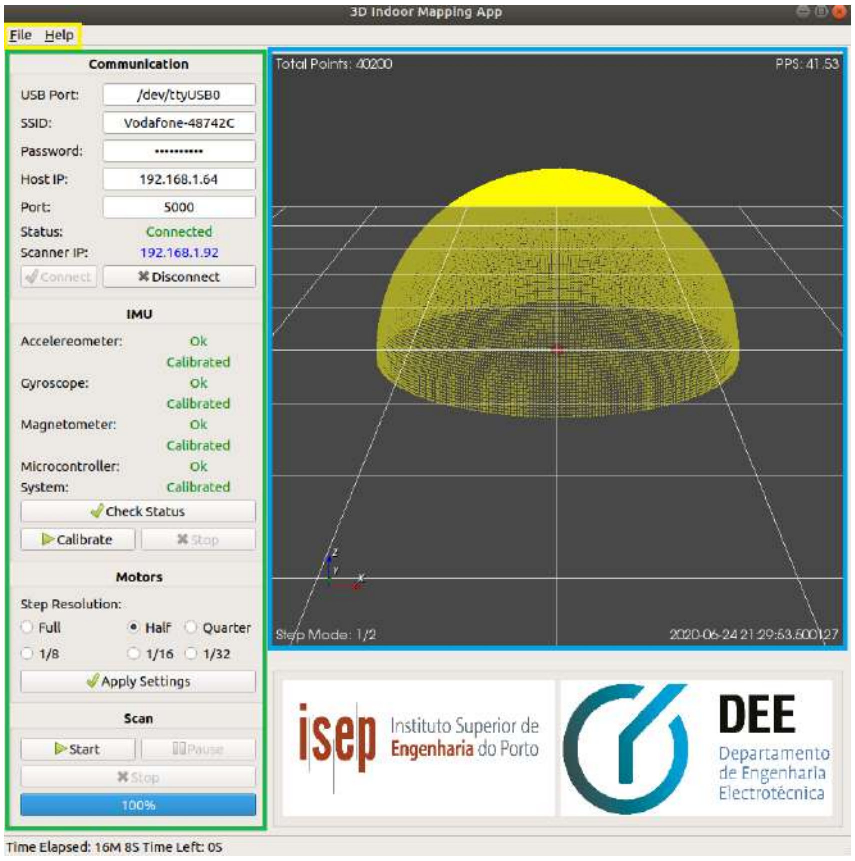
Qt is a free open-source graphical user interface framework. It introduces the concept of signals and slots (where a widget activates a signal which is connected to slot) [36]. It offers support to use embedded VTK window visualization and many different design tools such as Qt Designer and Qt Creator, that improves design time efficiency. However, the code is not as efficient as the one of a manual design, so for this reason, these tools weren't used. The use of a GUI (Graphical User Interface) will improve user-friendliness, allowing non specialized users to be able to use the application, as well as the improvement in the overall aesthetics and accessibility to functions. The company's logo is shown in Fig. 5.4.

Fig. 5.4: Qt Logo.

5.6 3D Scanner Interface

The automation and visualization was one of the priorities of this project. The main idea was that a user could make use of the equipment without having to understand it to the fullest. Visualizing the data gives it a feel for interactivity and also helps debugging data acquisition problems. This also makes it faster and easier to repeat tests for a long period of time. And for this reasons, an application was developed, that could handle some user parameters and export the data but also display the data in real time using wireless communication so that the scanner could be controlled by distance or even in the ”cloud”. This application is a combination on the use of sockets, threads, VTK and Qt. The software is divided into 3 major areas that can be seen in Fig. 5.5, divided in colours:

- Green - Contains all the configurable parameters of the software.
- Blue - Contains the VTK window, where the data will be visualized.
- Orange - Contains all the output options, after the scan is completed.



Communication

USB Port: /dev/ttyUSB0
SSID: Vodafone-48742C
Password: ••••••••••
Host IP: 192.168.1.64
Port: 5000
Status: Connected
Scanner IP: 192.168.1.92

IMU

Accelereometer: Ok Calibrated
Gyroscope: Ok Calibrated
Magnetometer: Ok Calibrated
Microcontroller: Ok
System: Calibrated

Motors

Step Resolution: Full [✓ Half] Quarter 1/8 1/16 1/32

Total Points: 40200
PPS: 41.53
Step Mode: 1/2
2020-06-24 21:29:53.600127

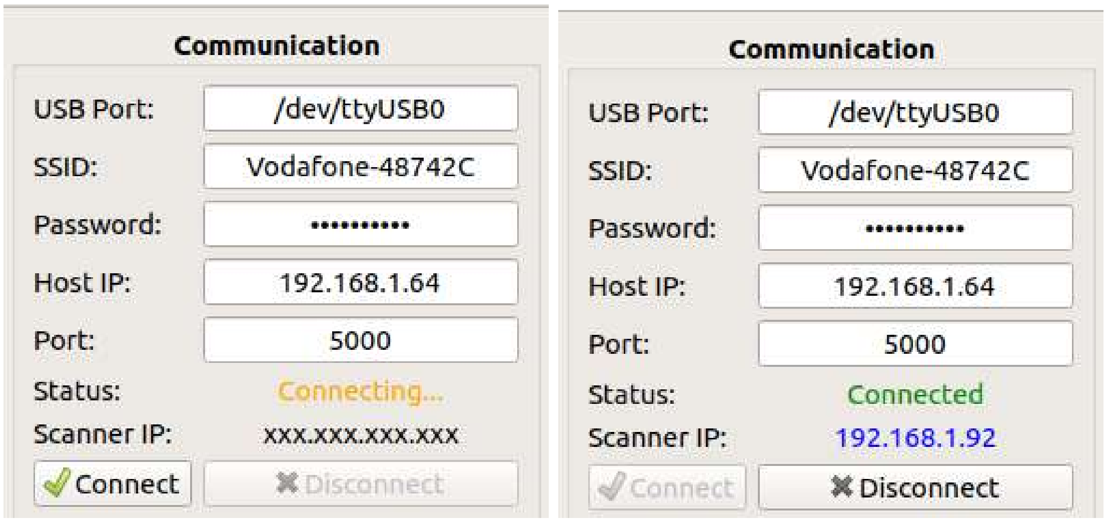
Time Elapsed: 16M 8S Time Left: 0S

Fig. 5.5: Graphical User Interface: General View.

Once the scan ends, the application will look similar to Fig. 5.5. The point cloud can then be exported to a .vtk, .xyz file or other formats, so that it can be used by other applications. Besides the point cloud, the user can also take notice of other parameters related to the scan. The left top corner of the VTK window contains information about the amount of points already scanned. The right top corner contains information about the current speed of the scan in PPS (Points Per Second). The left bottom corner displays the current step-mode resolution of the motors. And the right bottom corner displays information about the current time. Throughout the scan, the user can also keep track of the progress bar, time elapsed and time left in the left bottom corner of the interface. To use this software, the following steps need to be made:

1. The user needs to **connect the power supply** to the board and **wait for the scanner to calibrate itself** by adjusting the orientation of the sensor, then connects a micro USB cable from the computer to the board. Note that the board needs to be always powered throughout the all process because if it looses power, the microcontroller will restart and all the data will be lost, so at least one of this cables must be connected to the board for power considerations.

2. Although the transmission of data is made via Wi-Fi, the scanner needs to initially know which network to connect to and its respective password (if it has a password) and that information is transmitted by USB. Note that the desktop and the scanner need to be connected to the same network. After completing those parameters, the user is ready to press the ***Connect*** button to initialize USB communication and transmit that data to the microcontroller. After pressing it, the status will change to ***Connecting...*** as seen in Fig. 5.6 a). This process takes around 10 seconds until the status changes to ***Connected*** like in Fig. 5.6 b). If this doesn't happen, there will be an error message displayed in the bottom left corner explaining where the error occurred. After this step is concluded, the user no longer needs the USB cable because all the other parameters will be already transmitted by Wi-Fi.



Communication

USB Port:	/dev/ttyUSB0
SSID:	Vodafone-48742C
Password:	••••••••••
Host IP:	192.168.1.64
Port:	5000
Status:	Connecting...
Scanner IP:	xxx.xxx.xxx.xxx

(a) Connecting View

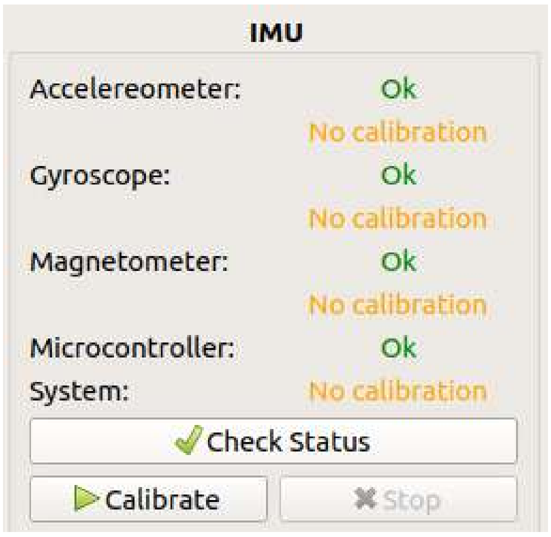
Communication

USB Port:	/dev/ttyUSB0
SSID:	Vodafone-48742C
Password:	••••••••••
Host IP:	192.168.1.64
Port:	5000
Status:	Connected
Scanner IP:	192.168.1.92

(b) Connected View

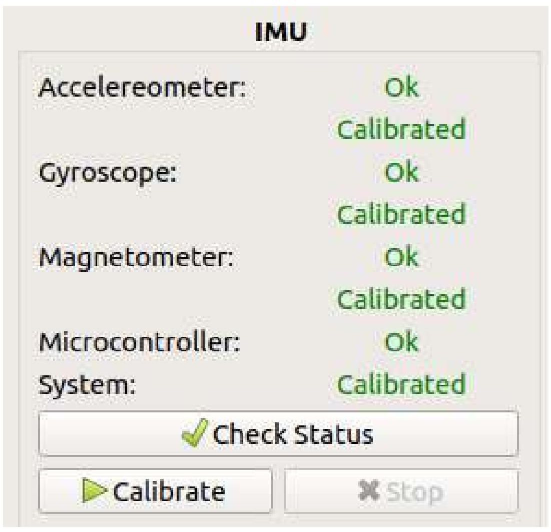
Fig. 5.6: Graphical User Interface : Communication.

3. Due to disturbances of the environment, the IMU needs to be calibrated every time it is powered. The first thing to do it is to check the status of all the components of the IMU by pressing the ***Check Status*** button. Once that is done, the message ***Ok*** shall be displayed in the menu for all the components as the example shows in Fig. 5.7 a). If it displays ***ERROR!***, it means that the IMU might be damaged. However, if that is not the case, then the user should proceed to calibration by pressing the ***Calibrate*** button. The values of calibration are immediately displayed in the bottom left corner. The values of calibration vary from 0 to 3. The value 0 means that its not calibrated and the value 3 means that it is fully calibrated. ***System*** is the average value of the calibration of all the components. See in Sec. 3.2.2 how to calibrate it. Once all the components are fully calibrated, press ***Stop*** to update the results and the final result will look like Fig. 5.7 b).


IMU

Accelereometer: Ok
No calibration
Gyroscope: Ok
No calibration
Magnetometer: Ok
No calibration
Microcontroller: Ok
System: No calibration

(a) Check Status

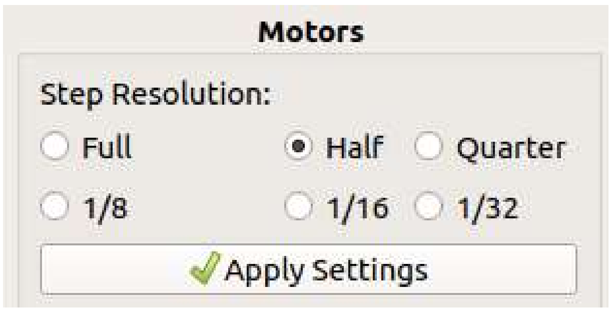

IMU

Accelereometer: Ok
Calibrated
Gyroscope: Ok
Calibrated
Magnetometer: Ok
Calibrated
Microcontroller: Ok
System: Calibrated

(b) IMU Calibrated

Fig. 5.7: Graphical User Interface: IMU.

4. The scanner has many available scanning resolutions defined by the microstepping mode of the stepper motors drivers. The smaller the step resolution, the more points will the scanner acquire, resulting in a better quality point cloud. The disadvantage of this is that it will take more time to do the scan. Each "level" of microstepping rises the number of points by 4x (Increase by a factor of 2 in two motors) which means 4x more resolution and theoretically 4x more time. To finalize this step press ***Apply Settings***. The conclusion of this step unlocks the ***Scan*** button.

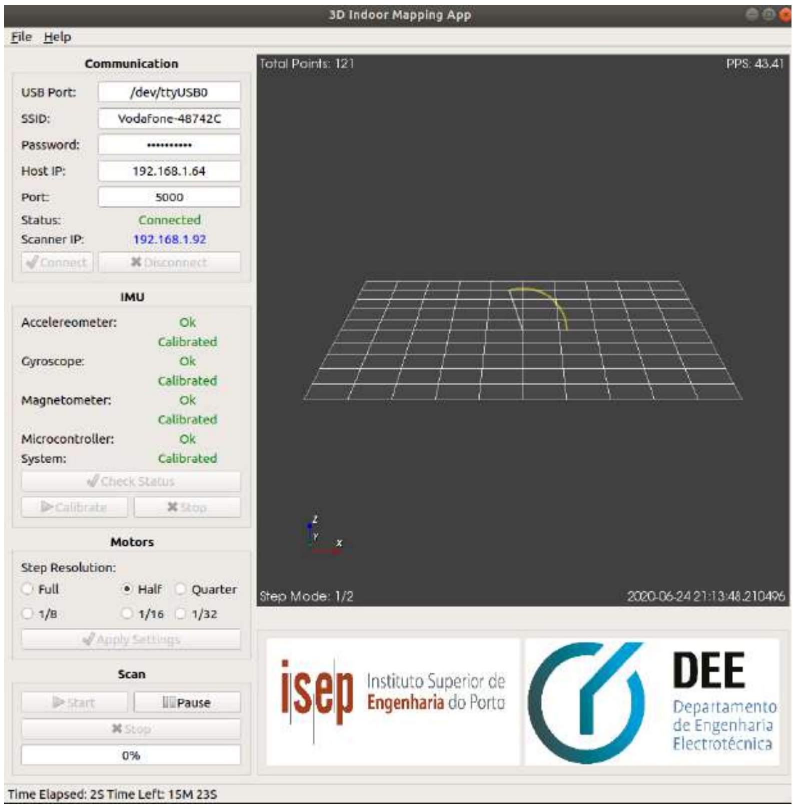

Motors

Step Resolution:
Full [✓ Half] Quarter
1/8 1/16 1/32

Fig. 5.8: Graphical User Interface: Motors Step Resolution.

5. The equipment is now ready to start scanning, press ***Scan*** to start the process. The cloud point will start to be displayed in the VTK window as seen in Fig. 5.9. The colour of the points is a representation of the total distance of the scanner to that exact point on a colour scale from red to blue, where red is the closest and blue is the furthest. The progress bar is updated throughout the scan so that the user might have a notion of the total progress. The bottom left corner displays the time elapsed since the beginning of the scan and a prediction of the time left to conclude

the scan, based on the speed of the scan given by PPS (points per second). The ***Pause*** button is unlocked so that the user might pause it for some reason or even stop it completely and restart another scan. The window is refreshed in real time every time a point is sent from the scanner. The white line represents the laser pointing and interconnects the position of the scanner to that last acquired point. The window corners contain important information about the scan, and three of them are updated in real time during the scan.



USB Port: /dev/ttyUSB0
SSID: Vodafone-48742C
Password: ••••••••••
Host IP: 192.168.1.64
Port: 5000
Status: Connected
Scanner IP: 192.168.1.92

Accelereometer: Ok Calibrated
Gyroscope: Ok Calibrated
Magnetometer: Ok Calibrated
Microcontroller: Ok
System: Calibrated

Step Resolution: Full [✓ Half] Quarter 1/8 1/16 1/32

0%

Total Points: 121
PPS: 43.41
Step Mode: 1/2
2020-06-24 21:13:48.210496

Time Elapsed: 2S Time Left: 15M 23S

Fig. 5.9: Graphical User Interface: Start Scanning.

5.7 Executable Python File

Trying to run the same code in different machines and operating systems might lead to different results. Python might not be installed or installed with a different version, and the same applies for the libraries used. It is not either user friendly having to run the code in a terminal or IDE for a common user. There are some tools that are useful in this situation, mainly ***PyInstaller*** (Linux and Windows) and ***NSIS*** (Windows).

The following steps were taken to convert the script into an executable file. Steps 1 and 2, are about installing the requirements and create the executable file and can be done using Linux OS or Windows OS but steps 3 and 4 are about the creation of an installer for the software and are only available for Windows OS.

1. **Install** PyInstaller in the command line:

   ```
   $ pip3 install pyinstaller
   ```

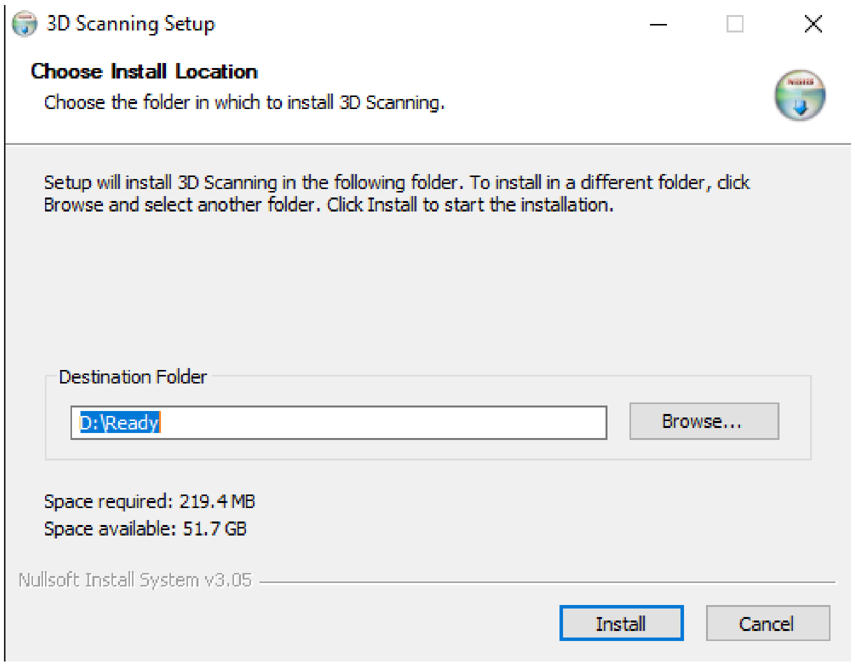
2. **Run** the following command to create the executable file. This will also define if the executable will run in Linux OS or Windows OS, depending on what OS is used in this step. The option [–onefile] defines that the result will be only one file, being it the executable file. The option [–windowed] disables the appearance of the console when running the software. The option [–icon] allows the application to have a customized icon. And finally the last argument is the main file of the code. All the dependencies must be in the same folder where this command is running.

   ```
   $ pyinstaller --onefile --windowed --icon=app.ico app.py
   ```

   The executable will be put into ***dist*** folder, and if it has dependencies it needs to be put into the original folder of the code. All the other newly created folders and files can be deleted. The following steps can only be done in Windows OS. The size of the file of this case is about 219.4 MB.

3. **Install** the NSIS software from NSIS Download.

4. Compress the project folder into a .ZIP folder, open NSIS and select ***Installer based on .ZIP file***. After that, the installer file will be generated and ready to use. The Fig. 5.10 is an example of the execution of that same file.


3D Scanning Setup

Choose Install Location

Choose the folder in which to install 3D Scanning.

Setup will install 3D Scanning in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

D:\Ready

Space required: 219.4 MB
Space available: 51.7 GB

Nullsoft Install System v3.05

Fig. 5.10: Software Installer.

Chapter 6

Point Clouds

6.1 3D Scanning

The scanner is finally complete and ready to start the final testing phase. It has a steady mechanical structure, a dedicated electronics board, a user-friendly GUI and all isolated elements are working properly. The final construction can be seen in Fig. 6.1 and contains the identification about the author and the context of the project. A simplified flow chart of the operations that occur in the server side and in the scanner side can be analysed in Fig. 6.2. A more detailed analysis will be adopted in the following sections. The main processes of the operating scanner follow a simple data pipeline:

1. Acquisition

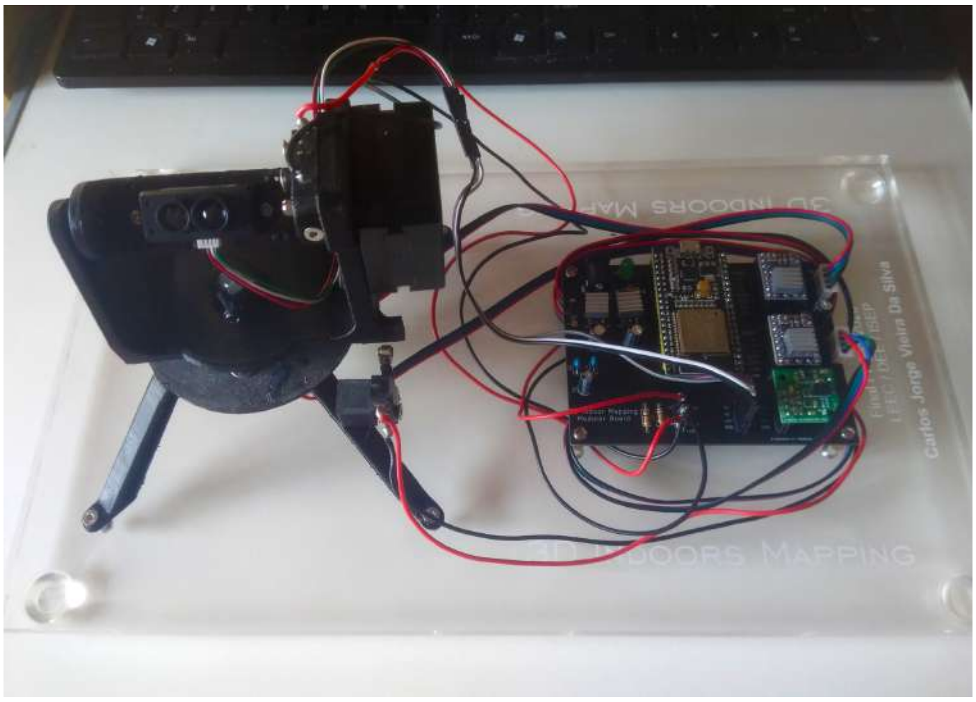
   The first step on the pipeline is to acquire the data that is provided by the TFmini-S that returns the distance to a point, its signal strength and the temperature of the sensor. And the data provided by the BNO055 that returns the rotation of the all structure in quaternion units. The information about the orientation of the sensor is calculated by the microcontroller, and registered the time stamp for each individual point.

2. Transmission, Processing and Visualization

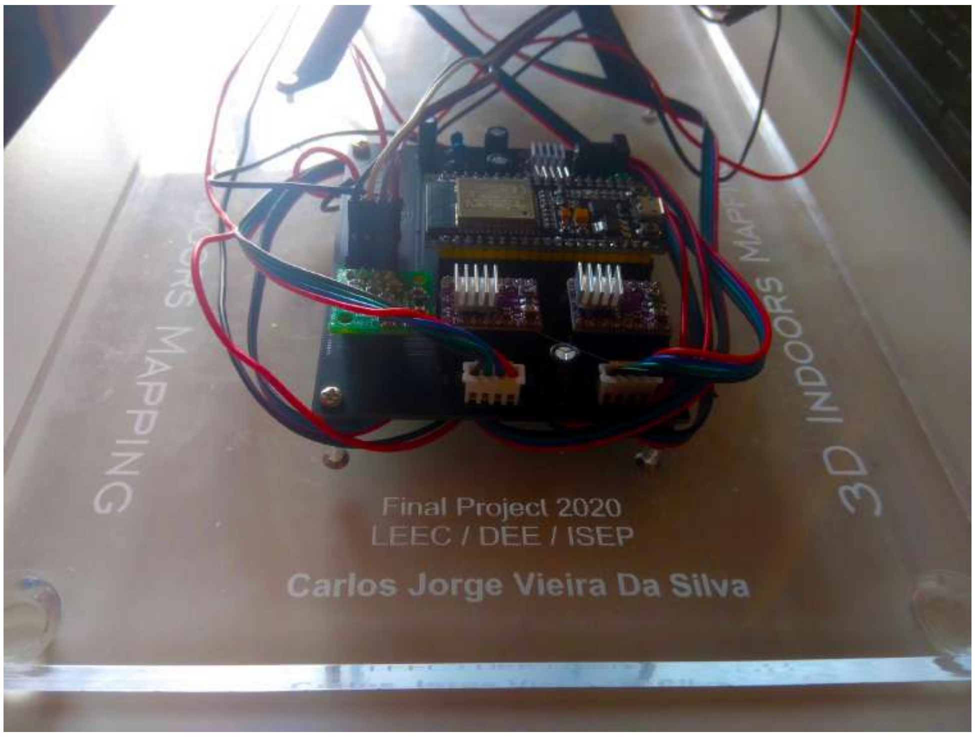
   That same data is then transmitted by Wi-Fi to a server connected in the same network where it will be initially processed. The data is then visualized by a user, in real-time, to keep track of the data, confirming that it is probably correct and that the system is working properly. The transmitted data frame is presented in Tab. 6.1.

3. Post-Processing

   The final step will be to save the data in the local storage of the server and apply modifications using dedicated software for this operations.



(a) Top View



(b) Side View

Fig. 6.1: 3D Scanner Complete.

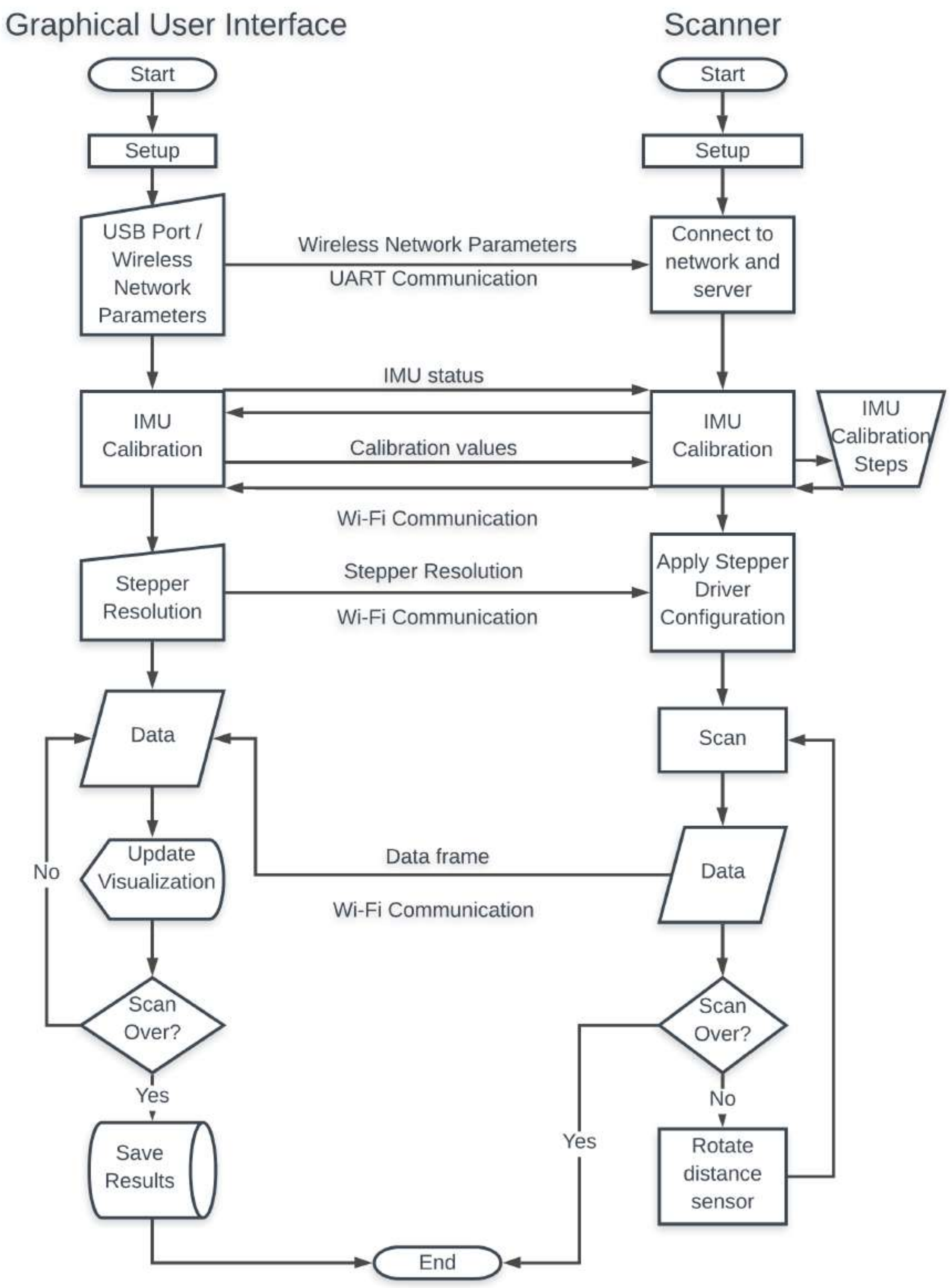


Fig. 6.2: Simplified Flow Chart.

Tab. 6.1: Data Frame.

Time Stamp (ms)	θ (rad)	ϕ (rad)	Distance (mm)	Signal Strength	Lidar Temperature (°C)	$q0$	$q1$	$q2$	$q3$

6.1.1 Data Acquisition

The developed libraries for the TFmini-S, BNO055 and DRV8825, presented in Sec. 3.2.4, Sec. 3.3.4 and Sec. 3.4.4 will converge to a main Arduino code where the acquisition of data and control of the motors is done, as well as the transmission of the data frame. An explanation of each step of the main process is presented:

1. Setup

   The serial communication is configured and initialized, followed by the initial configuration of the stepper motors drivers, the LIDAR sensor and the IMU. In order to perform the initial "homing" or calibration on the orientation the distance sensor, a timer was set on a 25 ms period and the pins connected to the limit switches were defined as interrupt pins by rising edge. The timer period defines the frequency at which the stepper motor takes steps, every time a step is made, the timer is restarted to make sure that the next step doesn't start before the defined period. This period is large due to the lack of torque on the motors because they are small and was calculated experimentally on a trial and error method until the steps were stable. Once the limit switches are pressed by the structure of the scanner, a signal will be send to the MCU and the motion on the switch axis will be stopped.

2. Connect to network and server

   The scanner will get from the GUI, the network and server parameters by the serial communication and then try to connect to the network first and then to the server and then its own IP information to the interface. If it cannot connect during a period of 5 seconds or if it looses the connection to the server or network, the microcontroller will restart. From this point all the communications will be done via Wi-Fi.

3. IMU Calibration

   Firstly checks the status of the IMU components and returns its value. Then it makes a continuous reading of the calibration values and returns them to the interface until the user stops it.

4. Apply Stepper Driver Configuration

   Changes the step mode value on the configuration parameters of the scanner. It also adjusts the timer period that controls the steps, the smaller the step mode, the smallest the timer period, by the same proportion so that it doesn't loose time efficiency.

5. Scan and Data Transmission

   The scan starts with the base stepper motor initial relative angle, $\theta = 0$ and the side stepper motor initial relative angle, $\phi = 0$. The angle interval between steps is defined by:

$$d\theta = d\phi = (\alpha)(n)\left(\frac{\pi}{180}\right), \tag{6.1}$$

   where:

α	Angle per step of the motor (1.8°),
n	Step Mode ($n \in [1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$),
$d\theta$	Yaw angle interval, in radians,
$d\phi$	Pitch angle interval, in radians.

   If the data has a weak signal or if there's an error in the reading, the microcontroller will attempt to read again until a maximum of 5 times and if the error persists, it will set the distance variable to zero. Then it will read the quaternions values from IMU sensor, pack the data frame and send it to the server. Then it will wait for an acknowledgement message (”ACK”) confirming that the server is now ready to receive another data frame, however if the message contains the command to cancel the scan or if the transmission with the network or server is lost during any part of the process, the MCU is restarted. If not, the process of the scan will continue until it completes 180° on each stepper motor.

6.1.2 Data Interpretation and Storage

The main python code was developed ”on top” of the GUI template, which was contributed by Carlos A. Vinhais, PhD. It is composed by 3 different files, the main file, the VTK window file and the GUI file. The main code loads the other two files, in order to provide a better organization of the code. An explanation about the GUI and its use was already made in Sec. 5.6. An explanation of each step of the main process is presented:

1. Setup

   Creates and displays the graphical user interface, along with all the necessary libraries and it is now ready for the user interface.

2. USB Port / Wireless Network Parameters

   Once this parameters are submitted by the user, a socket server will be initialized and the serial communication if the board will start.

3. IMU Calibration

   When the user presses the check status button, the message ”IS” (IMU Status) is send to the board which then responds for every individual component that composes the IMU, and that status is updated on the GUI. After the user presses the calibrate button, the message ”IC” (IMU Calibrate) is send to the board, and it starts receiving the calibration values and displaying them on the status bar. The calibration status is then updated on the GUI.

4. Stepper Resolution

   The message ”M” (Motor) is sent, along with the value of chosen step mode denominator. The microcontroller will then update the configuration of the stepper driver. This value will also be available in the bottom left corner of the visualization window.

5. Data reception and visualization

   A new thread starts so that the GUI continues to be responsive, the message ”S” (Scan) is sent and the process begins. A file *.txt* is also created and it will contain all the received values from the scanner, working as a backup file in case any problems occurs and the scan stops, as well as a full data file for future post-process analysis. The received data is then parsed to compute the Cartesian coordinates(x, y, z) of the corresponding scanned point:

   $$\begin{cases} x = r\cos(\phi)\cos(\theta) \\ y = r\cos(\phi)\sin(\theta) \\ z = r\sin(\phi) \end{cases}, \tag{6.2}$$

   where r, θ, ϕ are the spherical coordinates of the scanned point, with respect to the scanner position.

   The point containing these 3D coordinates is then added into the point cloud and the window is refreshed containing the new point. A line is also plotted from the origin to the most recently acquired point, representing the actual LIDAR direction which is useful for debugging purposes. The PPS value is then updated each 20 points as well as the current time. The amount of total points is updated with every point. The process ends when the full rotation of the distance sensor is over.

6. Save results

   Once the scan is complete, it is now ready to save the results of the acquired data. A *.vtk* is saved that contains all the information about the raw points, file that can be easily read again using this library functions. The next file to be saved is a *.xyz* that has the information about the coordinates of the points as well as the color code in RGB, all that separated by the *space* character and points separated by lines. The colour applied to which point will be according to the distance from the scanner, on the colour gradient scale *Red* - *Green* - *Blue*. The statistics of the scan will be saved on a *.txt* that will contain: The time and date of the beginning and end of the test, the total elapsed time, the number of total points, the average PPS, the precision using the redundancy of the points in the north pole in absolute and relative values, the maximum height and the values of the area and perimeter that are calculated using two different methods, with their respective error values. AutoCad script files *.scr* are also generated in this step, so that the point cloud can be imported by this software. One of the files plots the raw points and the other plots a line that goes through all points, in the same order they were scanned.

6.2 Experimental Results

Three tests were made, varying the step size of the motors from full step size mode to quarter step size mode. The quantitative results are expressed in Tab. 6.2. The total amount of points is, as predicted, bigger by a factor of 4 plus the redundancy on the north pole by a factor of 2, when decreasing the step size of the motor by a factor of 2, because it will be applied in both motors, resulting in a smaller angle interval, which increases the number of points scanned. The time elapsed during the scan is proportional to the number of points scanned while the speed averages 35.65 PPS. It is also possible to conclude that the farthest point is somewhere in the interval [3.325, 3.404] m by doing an intersection of the intervals of the different results for the maximum point distance and that the highest point somewhere in the interval [1.837, 1.890] m by also applying an intersection on the intervals of the different results for the maximum point height. The other step sizes weren't fully tested because they take too long to make a full scan and due to several limitations, it was not possible to reserve an empty room during the full period of those scans. Keeping the average PPS, the next step sizes would have presented the results expressed in Tab. 6.3. The predicted elapsed times are just too long to conduct this tests at home.

Tab. 6.2: Final Results.

Parameter	**Full Step**	**Half Step**	**Quarter Step**
Step Angle	1.8°	0.9°	0.45°
Total Points	10 100	40 200	160 400
Time Elapsed	5M 4S	17M 26S	1H 15M 41S
Average PPS	33.21	38.43	35.32
Precision (%)	99.976	99.986	99.982
Maximum Point Distance (m)	$3.385 \pm 1.77\%$	$3.344 \pm 1.79\%$	$3.368 \pm 1.78\%$
Maximum Point Height (m)	$1.858 \pm 1.72\%$	$1.875 \pm 2.02\%$	$1.865 \pm 2.25\%$

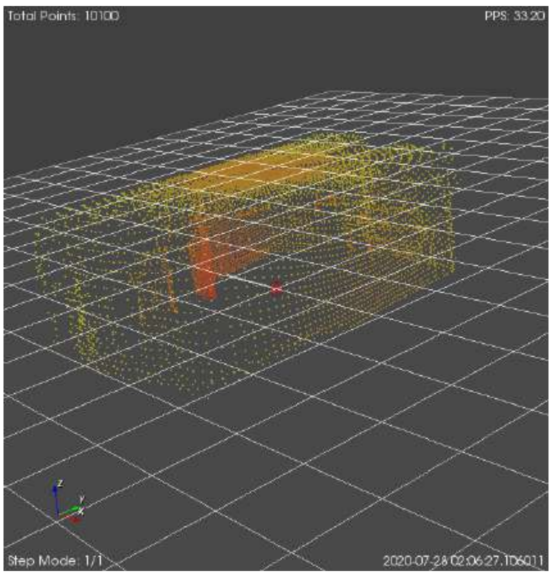
Tab. 6.3: Predicted Results.

Parameter	**Eighth Step**	**Sixteenth Step**	**Thirty-second Step**
Step Angle	0.225°	0.1125°	0.05625°
Total Points	640 800	2 561 600	10 243 200
Time Elapsed	04H 59M 35S	19H 57M 34S	3D 7H 48M 47S

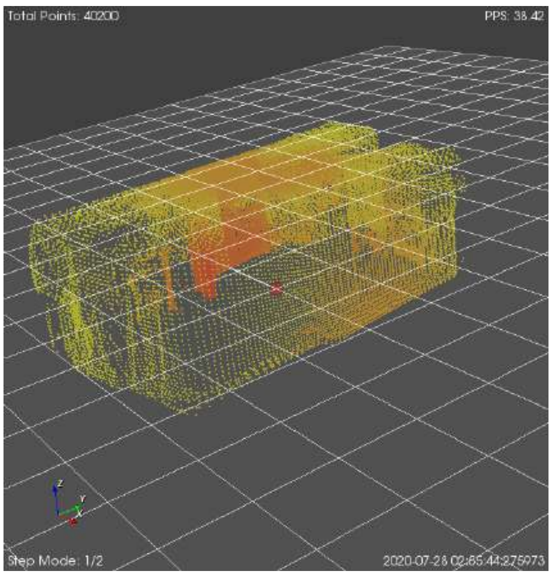
The qualitative results are displayed from Fig. 6.3 to Fig. 6.5. Note that no post-processing was applied. Fig. 6.3 displays a screenshot of the graphical user interface VTK window, that contains the total amount of points on the top left corner, the average speed on the top right corner, the used step mode on the bottom left corner and the date and time at which the test finished. Different colours represent different distances relative to the scanner, where the reddish are close to the scanner and the yellowish are further away. The structure that we are looking at, is the structure of a room in 3D dimensions, viewed from the outside. The different tests are very distinguishable due to the big difference in the amount of points and point density. For a better visualization, the cloud was then saved and imported by MeshLab software, which allows for visualization of different clouds in the same environment, providing a better and stable comparison view.

In Fig. 6.4, the view to the top of the room from the outside is illustrated. It is possible to verify that there is an accumulation of points right on the top of the location where the scanner was placed, converging on the north pole of it. The radius of that accumulation of points is bigger with the increase on the number of scanned points.

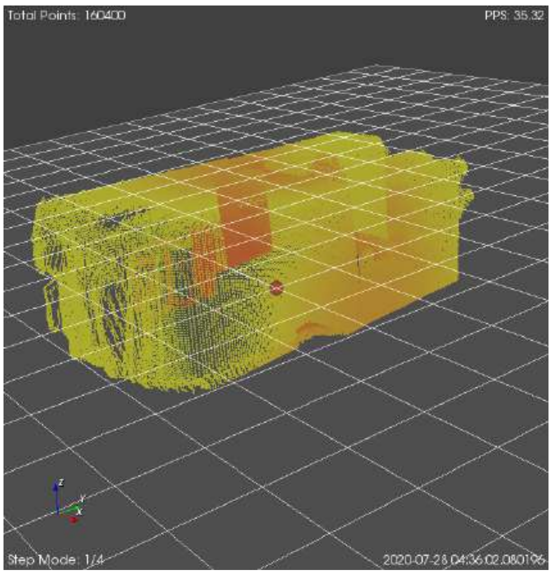
From Fig. 6.5, the view is now on the inside of the room where it is obvious to conclude that the bigger the point density, the better is the approximation on the mapping of the real environment. There is a hole on the ceiling of the room because of the lamp that was in the middle of the way between the scanner and the projected part of the ceiling. The most defined object with complex shapes is living room cupboard where there is a glass that caused some unstable results on that exact area.



(a) Full Step Mode

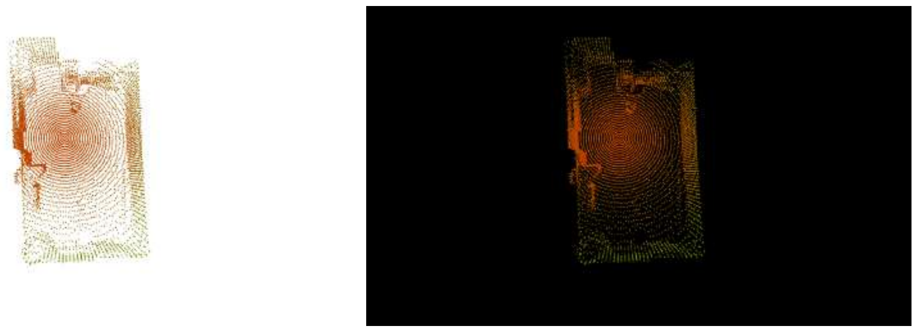


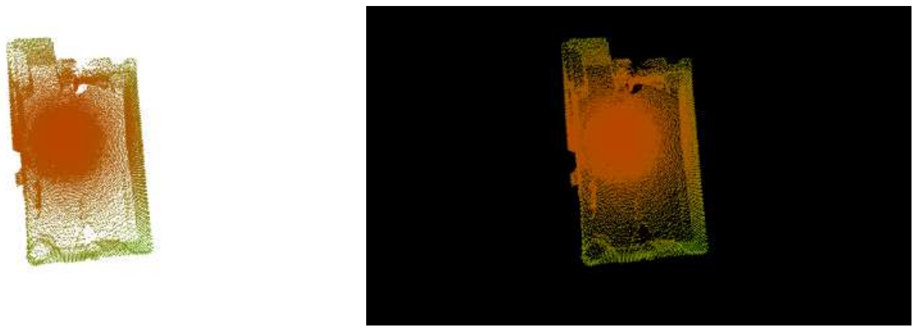
(b) Half Step Mode



(c) Quarter Step Mode

Fig. 6.3: 3D Scanning results, as shown in the rendering window of the GUI, using different stepping modes.


(a) Full Step Mode


(b) Half Step Mode

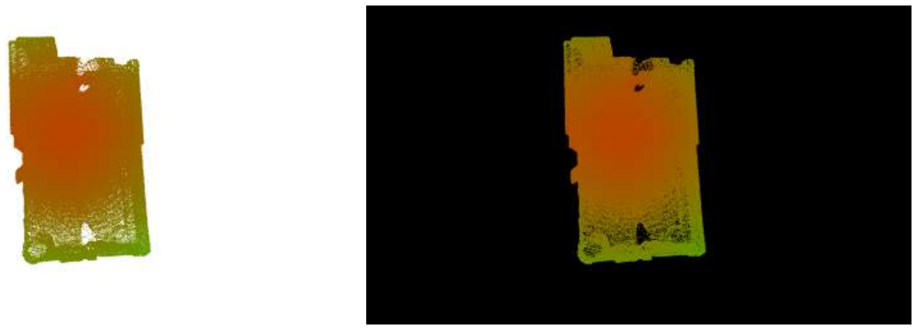
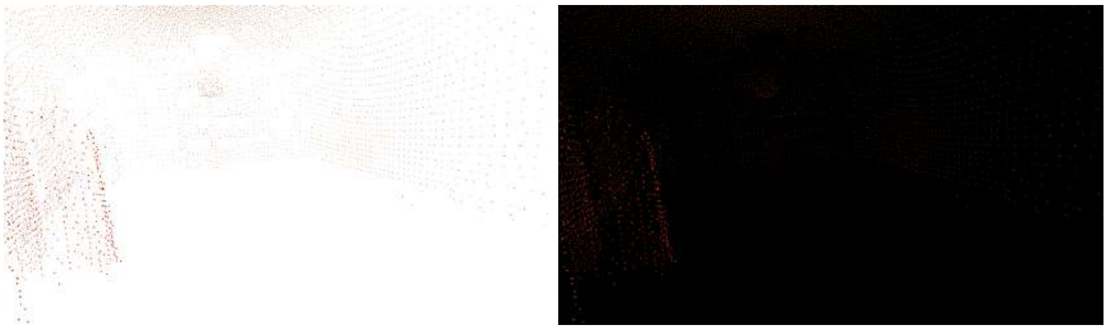
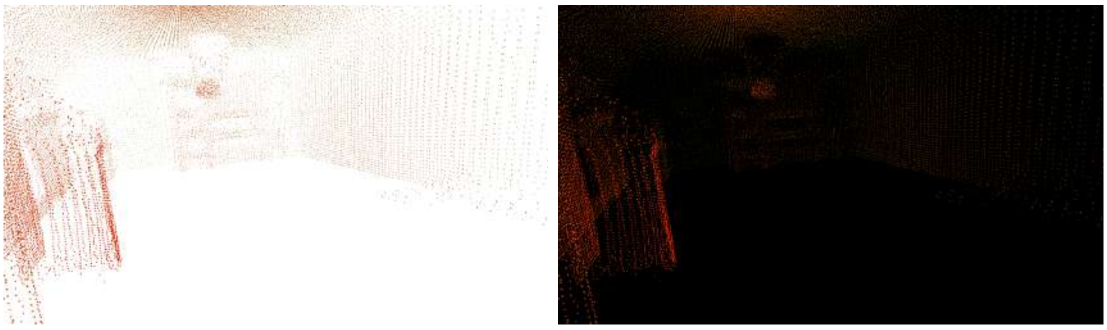
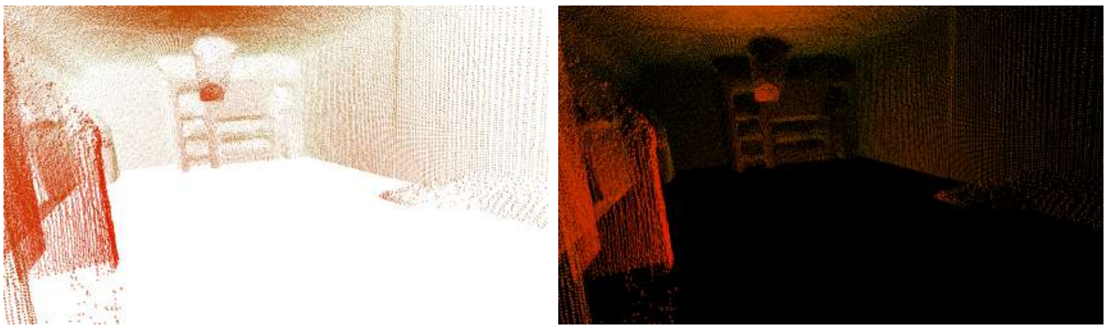

(c) Quarter Step Mode

Fig. 6.4: 3D Scanning results (top view), using different stepping modes. Point clouds were displayed with different background color, for visualization.


(a) Full Step Mode


(b) Half Step Mode


(c) Quarter Step Mode

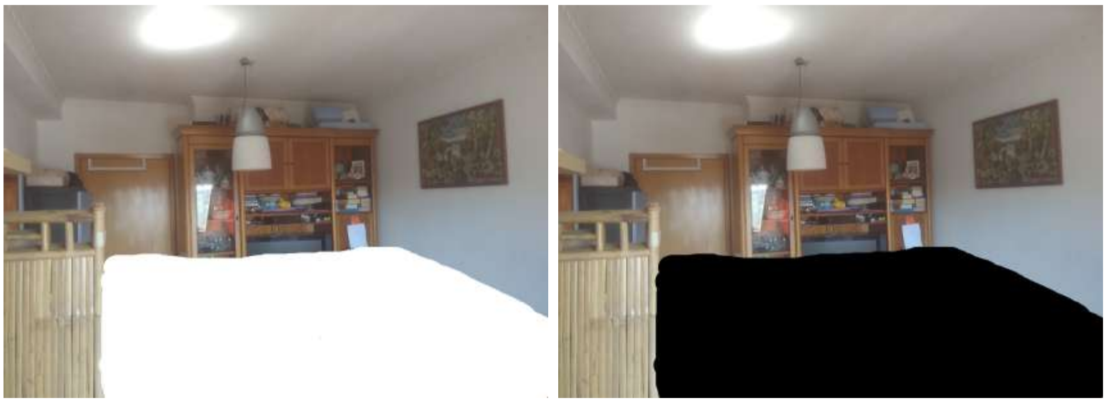

(d) Photo of the room

Fig. 6.5: 3D Scanning results (inner view), using different stepping modes. Point clouds were displayed with different background color, for visualization.

6.3 Data Post-Processing

6.3.1 Point Cloud Visualization

The process of mapping a room through a point cloud doesn't end after the scan is complete. The data needs to be processed *a posteriori* in order to filter the best results, change it, obtain other informations, convert it to other formats or for other purposes. Three popular tools will be presented here, used in different contexts:

- MeshLab - Processing of meshes
- Paraview - Scientific visualization
- AutoCad - Architectural analysis

In all softwares, the reference unit will be equal to 1 mm.

MeshLab

MeshLab is an open source system for processing and editing 3D meshes. It has several tools that allow the user to clean, edit, render, convert meshes, and others. However, it does not accept .vtk files, but accepts .xyz, a type of file that is also exported by the graphical user interface, containing information about the points location and also the colour of each of them. Later it can export that point cloud to many other formats like .obj. This software allows the loading of many point clouds in the same environment which was useful to obtain Fig. 6.5 and Fig. 6.4. The environment of this software is displayed in Fig. 6.6.

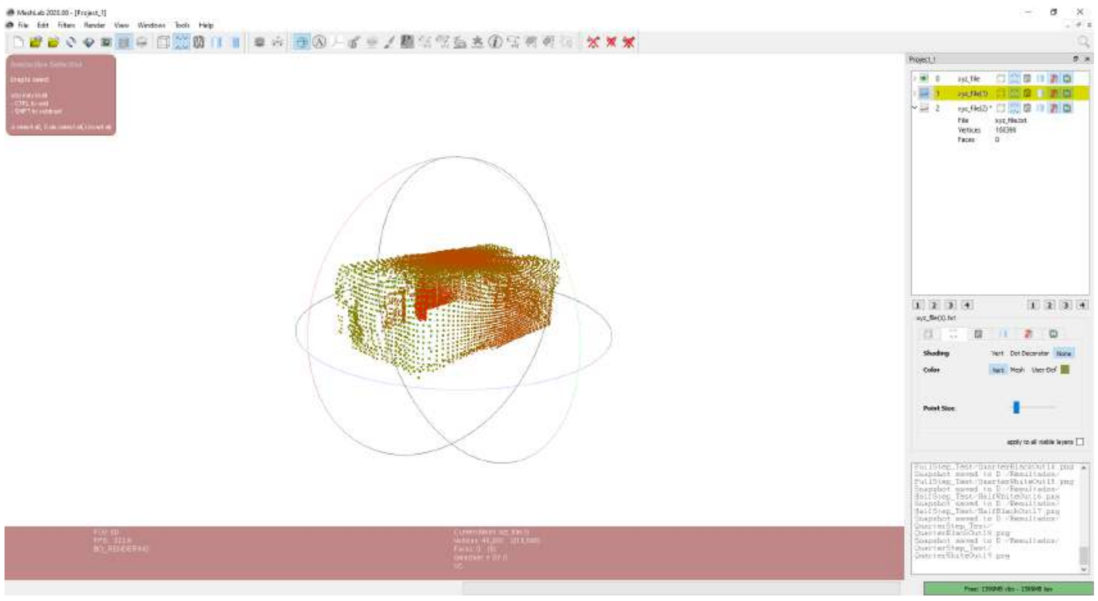


Fig. 6.6: MeshLab Environment.

ParaView

ParaView is an open-source software, and it is used in data analysis and visualization. It is useful in analysing the data with qualitative and quantitative techniques and can import .vtk files. Some point cloud characteristics are better visualized using a line that goes through all the points in the same order they were scanned. They might prove useful when debugging for imperfections and scan errors. To convert the point cloud into a polyline, the List. 6.1 must be introduced into the programmable filter option. An example of environment of this software is displayed in Fig. 6.7.

Listing 6.1: File: FilterParaview.py

```python
output.Points = inputs[0].Points
numPoints = len(output.Points)

pointIds = vtk.vtkIdList()
pointIds.SetNumberOfIds(numPoints)
for i in range(numPoints):
    pointIds.SetId(i, i)

output.Allocate(1, 1)
output.InsertNextCell(vtk.VTK_POLY_LINE, pointIds)
```

AutoCad

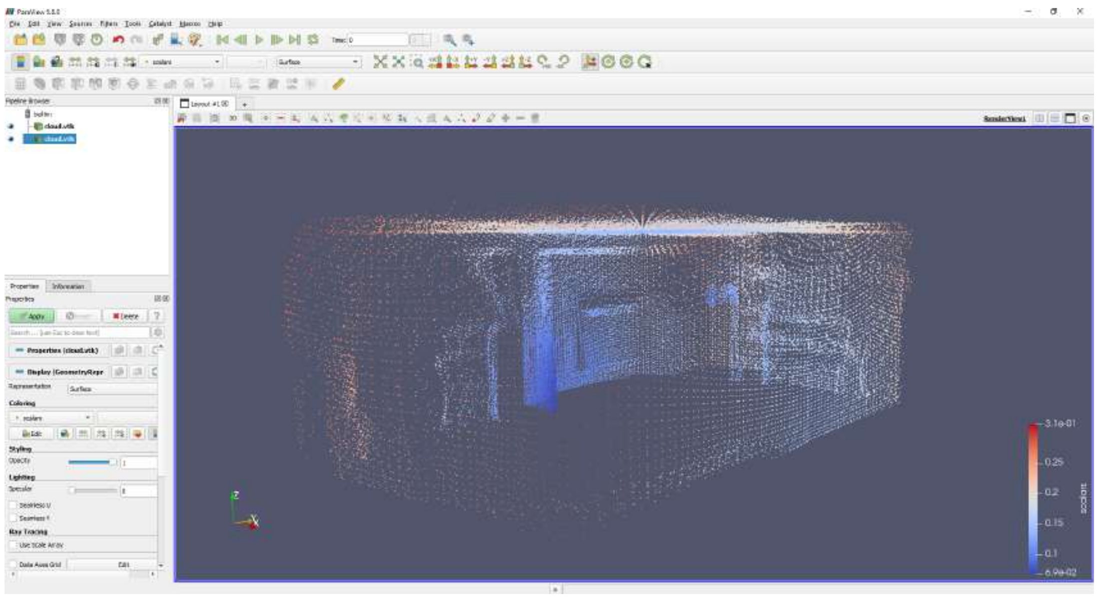
AutoCad was created by AutoDesk Inc. and is a CAD (Computer Aided Design) software to develop projects in many areas, however we're only going to focus on the architectural application. This software allows the reading and interpretation of drawings, plants, 3D models and performs structural calculations for architects. A 3D point cloud can be imported into this software using a script file that can be interpreted by it. These script files follow a simple structure, the first line contains the command to be executed by AutoCad while the other lines define the points to be read, in a 1 point per line logic, defined by X,Y,Z coordinates. Examples on the structure of the script files are display in List. 6.2 and List. 6.3. The first one is an example about how to plot points and the second exemplifies how to create a line that goes through all the points in the same order they were scanned. An example of environment of this software is displayed in Fig. 6.8.

Listing 6.2: File: AutoCadPoints.scr

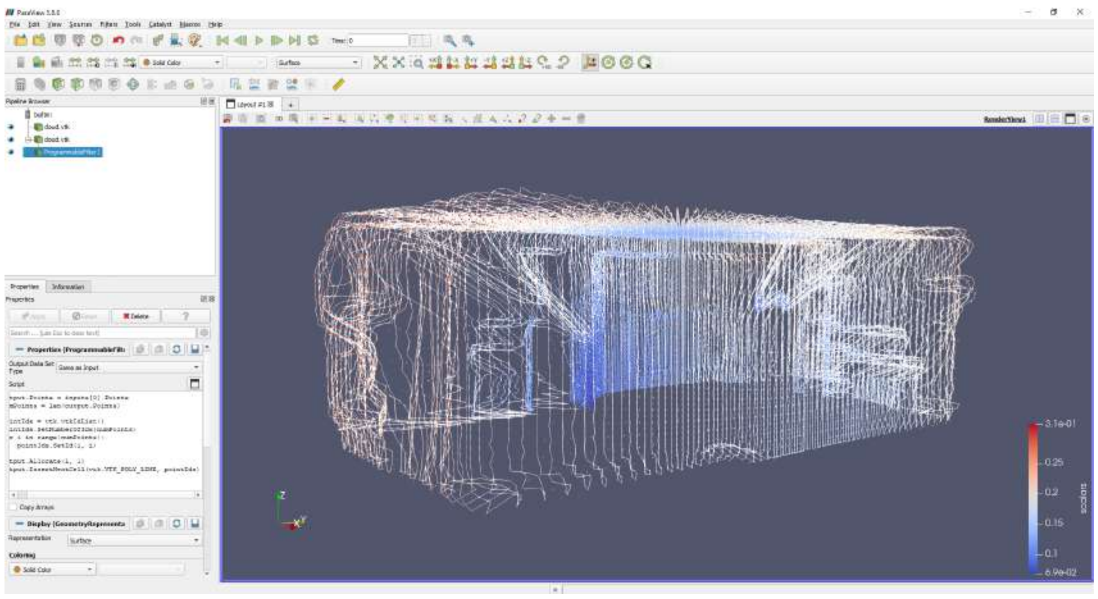
```
_MULTIPLE _POINT
1906.0,0.0,0.0
1905.93994140625,0.0,14.970000267028809
1904.77001953125,0.0,29.920000076293945
1903.469970703125,0.0,44.86000061035156
1902.06005859375,0.0,59.77000045776367
1899.530029296875,0.0,74.62999725341797
1898.8900146484375,0.0,89.55000305175781
1897.1300048828125,0.0,104.41000366210938
1895.25,0.0,119.23999786376953
```

Listing 6.3: File: AutoCadLines.scr

```
._3DPOLY
1906.0,0.0,0.0
1905.93994140625,0.0,14.970000267028809
1904.77001953125,0.0,29.920000076293945
1903.469970703125,0.0,44.86000061035156
1902.06005859375,0.0,59.77000045776367
1899.530029296875,0.0,74.62999725341797
1898.8900146484375,0.0,89.55000305175781
1897.1300048828125,0.0,104.41000366210938
1895.25,0.0,119.23999786376953
```



(a) Point Cloud



Output Data Set Type: Same as Input

Script

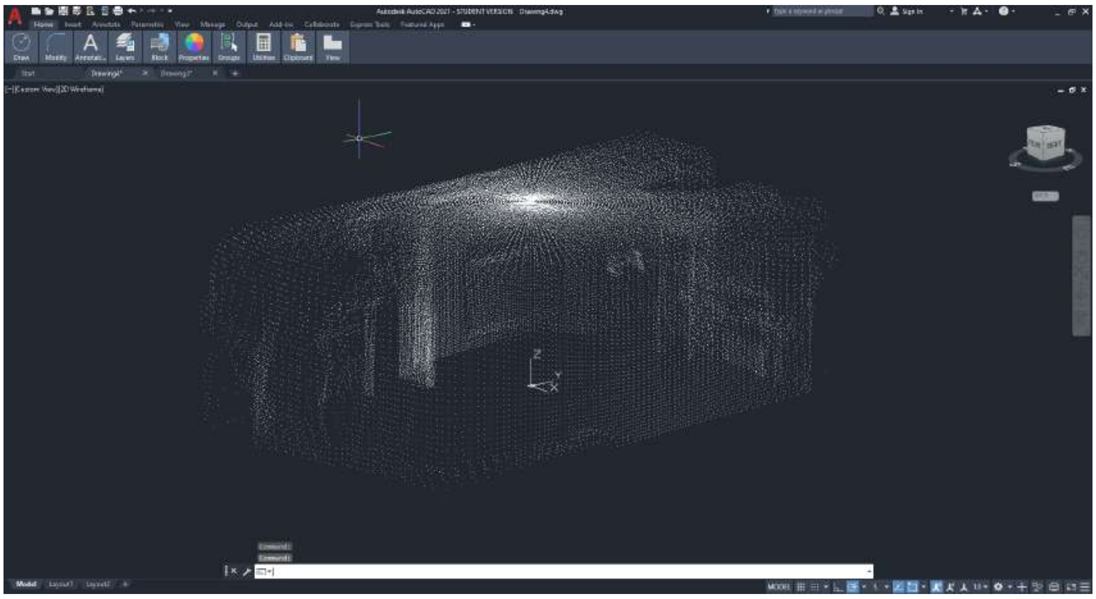
```
tput.Points = inputs[0].Points
mPoints = len(output.Points)

intIds = vtk.vtkIdList()
intIds.SetNumberOfIds(numPoints)
r i in range(numPoints):
  pointIds.SetId(i, i)

tput.Allocate(1, 1)
tput.InsertNextCell(vtk.VTK_POLY_LINE, pointIds)
```

Copy Arrays

Representation: Surface

(b) Line Cloud

Fig. 6.7: ParaView Environment.


(a) Point Cloud

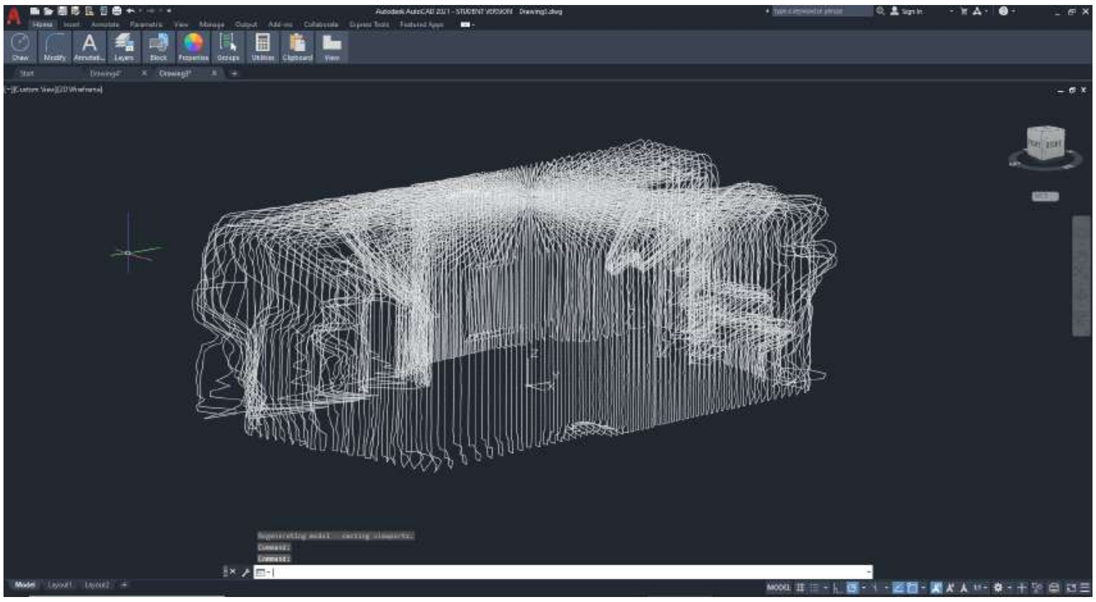

(b) Line Cloud

Fig. 6.8: AutoCad Environment.

6.3.2 Area and Perimeter

For certain applications, it might be useful to obtain a blueprint of the 3D scanned room. We will considerer as our reference, the lowest XY plane of the cloud point, where Z=0. This will not be equivalent to a real blue print because the real value of Z will depend on the height where the scanner its placed, and also on the height of the actual scanner. With the obtained blueprint, it is also possible to calculate the value of the respective area:

$$\begin{cases} A = \int_a^b f(x)\,dx \approx \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x \\ \Delta x = \frac{b-a}{n}, n \in \mathbb{N} \end{cases} . \tag{6.3}$$

An example of this process is displayed in Fig. 6.9, defined by the function $f(x) = sin(x)$ with $x \in [0, \pi]$, where a comparison between different values of n, results in different levels of approximation. This method will be adapted into polar coordinates.

Two different methods were used in order to approximate the values of the area, A, and perimeter, P, one using semi-circles and the other using triangles. The first one is exemplified in Fig. 6.10, defined by $f(x) = (3 + 2\cos(x))(1 - 0.2\sin(6x))$ with $x \in [0, 2\pi]$, where a comparison between different values of $\Delta\beta$, results in different levels of approximation. The following symbol definitions will be used from Eq. (6.4) to Eq. (6.7):

α	Angle per step of the motors in full step mode ($\frac{\pi}{100}$),
N	Stepping mode ($N \in [1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$),
β	Angle of the motors ($\beta = \alpha\, N$),
r	Distance to the point.

On this first case, the total area of the blue print will be approximated by the sum of the areas of all the semi-circles with radius (r_i) equal to the mid point between two closest points and arc ($\Delta\beta$) equal to the difference between the relative angles of those points, until 2π. The perimeter will be equal to the sum of all the arcs.

$$A \approx \sum_{i=1}^{\frac{2\pi}{\beta}} \pi\, r_i^2\, \Delta\beta \approx \pi\, \Delta\beta \sum_{i=1}^{\frac{2\pi}{\beta}} r_i^2, \tag{6.4}$$

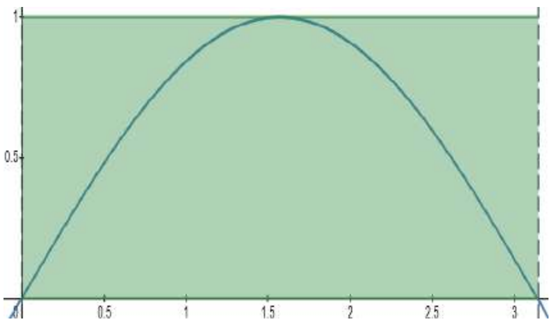
$$P \approx \sum_{i=1}^{\frac{2\pi}{\beta}} r_i\, \Delta\beta \approx \Delta\beta \sum_{i=1}^{\frac{2\pi}{\beta}} r_i. \tag{6.5}$$

On the second case, the total area of the blue print will be approximated by the sum of the areas of all the triangles with height (r_i) equal to the distance between the centre and the mid point between two closest points and base ($r_i \tan(\Delta\beta)$), until 2π. The perimeter will be equal to the sum of all the bases.

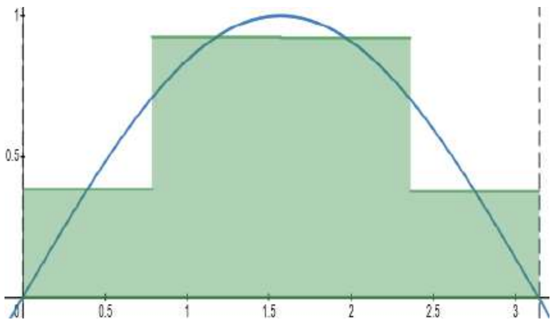
$$A \approx \sum_{i=1}^{\frac{2\pi}{\beta}} \frac{1}{2} r_i \left(r_i \tan(\Delta\beta)\right) \approx \frac{1}{2} \tan(\Delta\beta) \sum_{i=1}^{\frac{2\pi}{\beta}} r_i^2, \tag{6.6}$$

$$P \approx \sum_{i=1}^{\frac{2\pi}{\beta}} r_i \, tan(\Delta\beta) \approx tan(\Delta\beta) \sum_{i=1}^{\frac{2\pi}{\Delta\beta}} r_i. \tag{6.7}$$

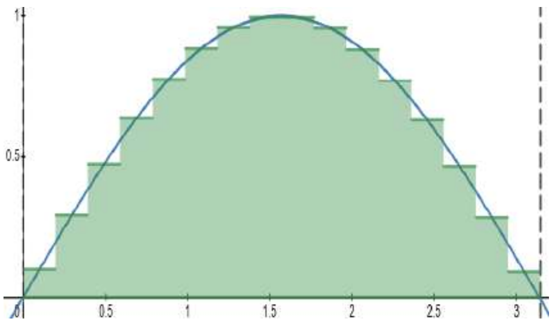
Let's considerer the method of the semi-circles as method A, and the triangles method as method B. These methods were tested using three different step modes: full step, half step and quarter step. Looking again at Fig. 6.10, its possible to conclude that the smaller the step size (or angle of rotation), the better will the approximation on the real value of the area be. The expected results follow the same logic. A visual analysis on the resulting blue prints that are present in Fig. 6.11, allowing us to conclude that the smaller the step size, the smoothest will the resulting blueprint be, and therefore the better the approximation on the real values of area and perimeter. Comparing the results of the different methods displayed in Tab. 6.4, it's noticeable that both methods have similar results and that they tend to decrease with the decrease of step size and more likely closer to the real value. Note that the values of the error are related to the error on the distance returned by the TFmini-S sensor and not on the real value. The real values weren't able to be obtained due to the difficulty of measuring complex shapes like the ones present in a furnished room. Ideally this test would have occurred in an empty room with a regular shape and known area value.



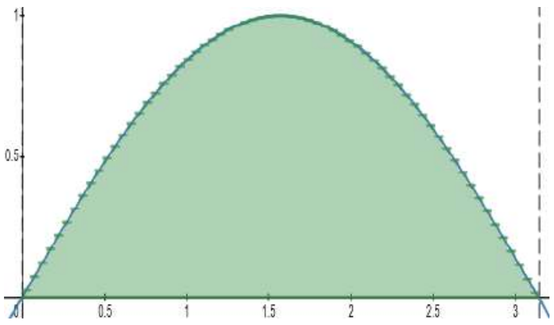
(a) Riemann n=1



(b) Riemann n=4

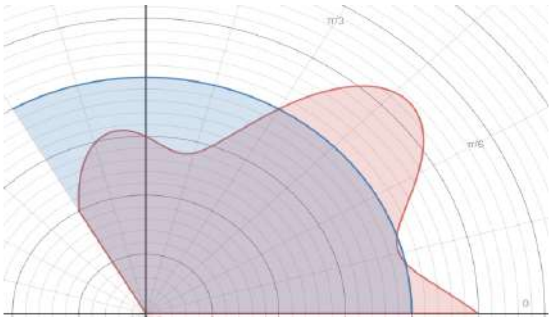


(c) Riemann n=16

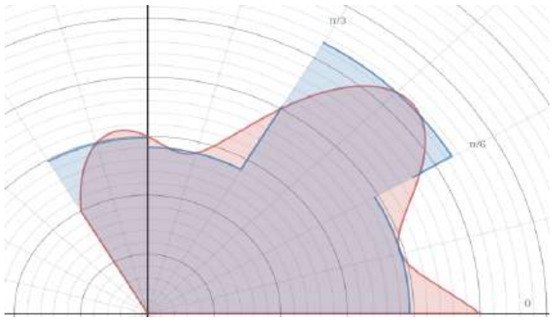


(d) Riemann n=64

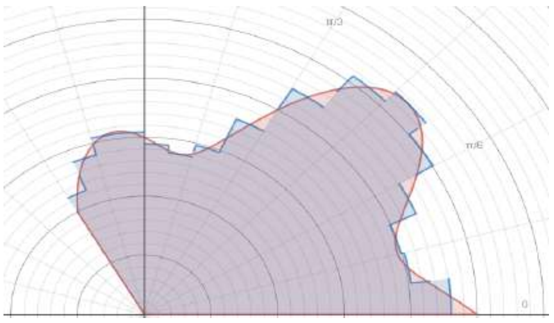
Fig. 6.9: Riemann Area: through Rectangles.



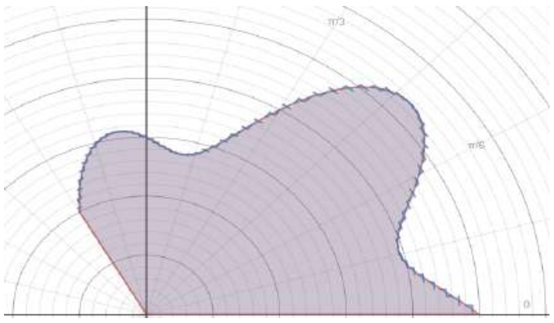
(a) Riemann $\beta = \frac{2\pi}{3}$



(b) Riemann $\beta = \frac{\pi}{6}$

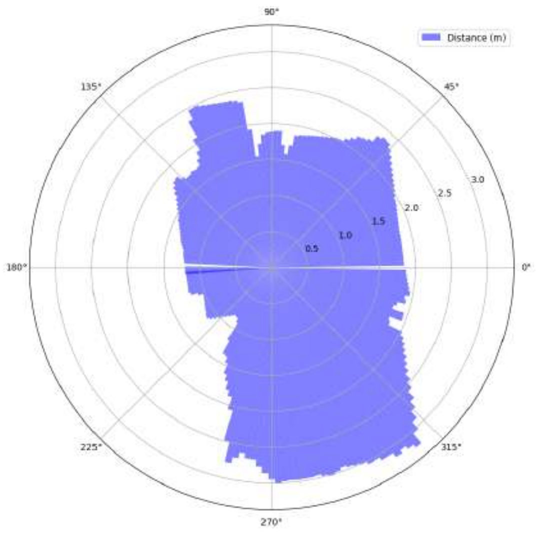


(c) Riemann $\beta = \frac{\pi}{24}$

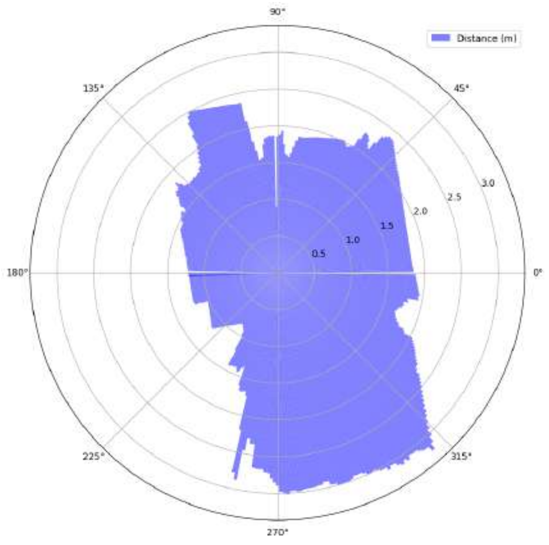


(d) Riemann $\beta = \frac{\pi}{96}$

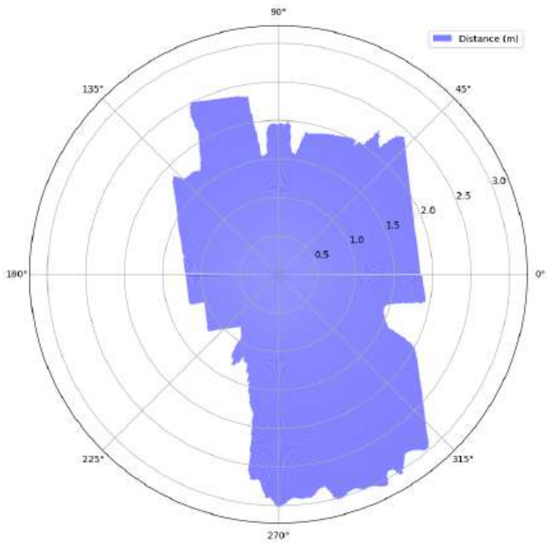
Fig. 6.10: Riemann Area: Comparison through Semi-Circles.



(a) Full Step Mode



(b) Half Step Mode



(c) Quarter Step Mode

Fig. 6.11: Scanned Blueprints.

Tab. 6.4: Results on the approximation of area and perimeter (Method A - Semi-Circles Approximation; Method B - Triangles Approximation).

Step Mode	**Area** (m^2) **Method A**	**Area** (m^2) **Method B**	**Perimeter** (m) **Method A**	**Perimeter** (m) **Method b**
1	$12.883 \pm 5.72\%$	$12.888 \pm 5.72\%$	$12.093 \pm 3.10\%$	$12.097 \pm 3.10\%$
$\frac{1}{2}$	$12.689 \pm 5.77\%$	$12.690 \pm 5.77\%$	$12.009 \pm 3.14\%$	$12.009 \pm 3.14\%$
$\frac{1}{4}$	$12.175 \pm 5.90\%$	$12.176 \pm 5.90\%$	$11.781 \pm 3.20\%$	$11.781 \pm 3.20\%$

Chapter 7

Conclusion

Indoor mapping using 3D scanners have become largely important in the last years, which is noticeable by the rising on the number systems and techniques to acquire three-dimensional data. One of the most solicited areas for this methods is Architecture where these scanners greatly improve scene analysis and calculation. One major problem is the high cost per unit which is considered a large investment that many individuals or companies cannot afford.

An economical real-time wireless 3D Scanner was developed in this project, that is composed by a mechanical structure, an electronic board and a graphical user interface that can detect its surroundings up to a distance of 12 m. This scanner was used to obtain three-dimensional information of a room, using different detail resolutions. With the use of the graphical interface, the process was easily automatized and able to be utilized by a non-technical user and with the wireless communication the placement of the scanner didn't suffer any cable limitations. The real-time visualization allowed quick detection of testing errors in different steps of the scan. As an alternative to a power supply, a set of batteries could also have been used but it would be more expensive. The disadvantages of this scanner in comparison with the commercial ones are that it is slower, especially when scanning with high resolution levels and the structure is made with less resistance materials.

For future development, the GUI shall be able of select any configuration of the sensors, as well as communication method (cable or wireless), detect the lost of communications and allow the connections of many scanners to the same application. The user must also be allowed to select the range of the scan. The mechanical structure can be greatly improved with a new design and better materials, to decrease vibration and friction between parts and allow 360° scans. The range of detection, resolution and frequency can also be improved by using a more expensive LIDAR sensor. Although the scanning rate can also be improved using the current scanner, by sending less data frames by increasing their size containing the values of many points in one single data frame.

Creating the scanner and its interface was really fun, educational and interesting. The project had some ”ups and downs” like any other but looking at the final results gives a feeling of pride. The most exciting moments were the first scanner movement tests, the first socket based creation on VTK points and the first scan. The amount of information that the author learned throughout the development process was also motivational and for sure, useful in the future:

- How 3D Scanners works.
- How to write a report using LaTeX.
- How stereoscopic cameras work.
- How an Inertial Measurement Unit and its components work, like the accelerometer, the gyroscope and the magnetometer, and how to use each one of them.
- How LIDAR sensors works and how to use one of them.
- How two of the most commonly used communication protocols work (I2C and USART).
- How to use VTK in Python language.

And improved:

- The capability of studying and understand components by analysing the datasheet.
- PCB design skills.
- Python programming skills.
- Graphical User Interface design skills.

The hardest challenges when developing this project had to do with the choice of the type of 3D Scanner to develop and the mechanical structure to use. Other challenges had to do with the use of a new report writing tool called LaTeX, that is really different from conventional report writing tools but proven to be more effective when mastered. And also with the language choice. English (not the native language) was chosen in order to develop English writing skills that are becoming more useful each day and so that a wider audience might be able to read this report. Some other challenges were due to the many consequences of the current year pandemic (COVID-19). The availability of an empty room during the testing period would also have allowed the acquisition of more data using different step resolutions.

Bibliography

[1] V. Sanchez and A. Zakhor, “Planar 3d modeling of building interiors from point cloud data,” in *2012 19th IEEE International Conference on Image Processing*, pp. 1777–1780, IEEE, 2012.

[2] A. Sampath and J. Shan, “Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds,” *IEEE Transactions on geoscience and remote sensing*, vol. 48, no. 3, pp. 1554–1567, 2009.

[3] G. Vosselman, “Advanced point cloud processing,” in *Photogrammetric week*, vol. 9, pp. 137–146, University of Stuttgart Stuttgart, Germany, 2009.

[4] F. Chiabrando, G. Sammartano, and A. Spanò, “Historical buildings models and their handling via 3d survey: from points clouds to user-oriented hbim.,” *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 41, 2016.

[5] G. Vosselman, S. Dijkman, *et al.*, “3d building model reconstruction from point clouds and ground plans,” *International archives of photogrammetry remote sensing and spatial information sciences*, vol. 34, no. 3/W4, pp. 37–44, 2001.

[6] L. Nishida, “Estudo de técnicas de processamento de imagens aplicadas a um active range finder com projeção de luz estruturada na forma de listras paralelas verticais,” *Laboratório de Técnicas inteligentes, Universidade de São Paulo.*, 2007.

[7] B. Curless, *New methods for surface reconstruction from range images*. PhD thesis, Stanford University Stanford, CA, 1997.

[8] F. Blais *et al.*, “Review of 20 years of range sensor development,” *Journal of electronic imaging*, vol. 13, no. 1, pp. 231–243, 2004.

[9] L. Albuquerque, M. José, and J. Motta, “Implementation of 3d shape reconstruction from range images for object digital modeling,” vol. 2, pp. 81–88, 01 2006.

[10] REMISHAW, “Ds10 contact scanner.” https://www.renishaw.com/en/ds10-contact-scanner--32394.

[11] L. Arcifa, D. Calì, A. Patanè, F. Stanco, D. Tanasi, and L. Truppia, “Laserscanning e 3d modelling nell’archeologia urbana: lo scavo della chiesa di sant’agata al carcere a catania (italia),” *Virtual Archaeology Review*, vol. 1, no. 2, pp. 51–55, 2010.

[12] M. J. Thali, M. Braun, and R. Dirnhofer, “Optical 3d surface digitizing in forensic medicine: 3d documentation of skin and bone injuries,” *Forensic science international*, vol. 137, no. 2-3, pp. 203–208, 2003.

[13] D. M. Orozco, “3D-Object Modelling for Computer Games,” *Presentation at Computer Seminar, ULM University*, 2010.

[14] B. CURLESS, *New Methods for Surface Reconstruction from Range Images. 1997, 189f.* PhD thesis, Tese de Doutorado em Engenharia Elétrica-Departamento de Engenharia Elétrica.

[15] R. I. Hartley, R. Gupta, and T. Chang, “Stereo from uncalibrated cameras.,” in *CVPR*, vol. 92, pp. 761–764, 1992.

[16] J. Shackleton, B. VanVoorst, and J. Hesch, “Tracking people with a 360-degree lidar,” in *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 420–426, IEEE, 2010.

[17] SLAMTEC, “Rplidar a1.” `https://www.slamtec.com/en/Lidar/A1`.

[18] L. Geosystems, “Leica rtc360 3d laser scanner.” `https://leica-geosystems.com/products/laser-scanners/scanners/leica-rtc360`.

[19] Espressif, *ESP32 Series*, 04 2020. Rev. 3.4.

[20] N. H. Tollervey, *Programming with MicroPython: Embedded Programming with Microcontrollers and Python.* ” O’Reilly Media, Inc.”, 2017.

[21] Espressif, “Esp-idf programming guide.” `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/about.html`.

[22] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, “Reviews on various inertial measurement unit (imu) sensor applications,” *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[23] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*, vol. 2. Citeseer, 1998.

[24] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[25] J. L. B. Claraco, “A tutorial on SE(3) transformation parameterizations and on-manifold optimization,” pp. 13–21, 2013.

[26] Bosch, *BNO055, Intelligent 9-axis absolute orientation sensor*, 11 2014. Rev. 1.2.

[27] J. A. Christian and S. Cryan, “A survey of lidar technology and its use in spacecraft relative navigation,” in *AIAA Guidance, Navigation, and Control (GNC) Conference*, p. 4641, 2013.

[28] M. Y. Stoychitch, “Generate stepper motor linear speed profile in real time,” *IOP Conference Series: Materials Science and Engineering*, vol. 294, pp. 12–55, jan 2018.

[29] Texas Instruments, *DRV8825 Stepper Motor Controller IC*, 04 2010. Rev. July 2014.

[30] KiCad, "About kicad." https://kicad-pcb.org/.

[31] A. A. Sawant and B. Meshram, "Network programming in java using socket," *InternationalJournal of Engineering Research and Applications (IJERA)*, vol. 3, no. 1, pp. 1299–1305, 2013.

[32] D. D. Guinard and V. M. Trifa, *Building the web of things*, vol. 3. Manning Publications Shelter Island, 2016.

[33] S. Oaks and H. Wong, *Java threads*. O'Reilly Media, Inc., 1999.

[34] M. D. Hanwell, K. M. Martin, A. Chaudhary, and L. S. Avila, "The visualization toolkit (vtk): Rewriting the rendering code for modern graphics cards," *SoftwareX*, vol. 1, pp. 9–12, 2015.

[35] L. Avila and I. Kitware, *The VTK User's Guide*. Kitware, 2010.

[36] I. Mezei, "Cross-platform gui for educational microcomputer designed in qt," in *2017 IEEE East-West Design & Test Symposium (EWDTS)*, pp. 1–4, IEEE, 2017.

Appendix A

ESP32 Data

Tab. A.1: BNO055 Output Data Registers.

Parameter	**Bytes**
ACC_DATA_X	2
ACC_DATA_Y	2
ACC_DATA_Z	2
GYR_DATA_X	2
GYR_DATA_Y	2
GYR_DATA_Z	2
MAG_DATA_X	2
MAG_DATA_Y	2
MAG_DATA_Z	2
EUL_HEADING	2
EUL_ROLL	2
EUL_PITCH	2
QUA_DATA_W	2
QUA_DATA_Y	2
QUA_DATA_Z	2
LIA_DATA_X	2
LIA_DATA_Y	2
LIA_DATA_Z	2
GRV_DATA_X	2
GRV_DATA_Y	2
GRV_DATA_Z	2
TEMP	1

Tab. A.2: BNO055 Sensor Default Settings.

Sensor	**Range**	**Resolution**
Accelerometer	4G	62.5 Hz
Gyroscope	2000 dps	32 Hz
Magnetometer	NA	10 Hz

Tab. A.3: BNO055 Output Data Representation (Quat. = Quaternion).

Unit	**Representation**
m/s^2	1 m/s^2 = 100 LSB
1 dps	1 dps = 16 LSB
1 μT	1 μT = 16 LSB
1 Degree	1 degrees = 16 LSB
Quat. (unit less)	1 Quat. = 2^{14} LSB
°C	1 °C = 1 LSB

Tab. A.4: BNO055 Output Data (Quat. = Quaternion).

Data	**Units**
Acceleration	m/s^2
Linear Acceleration	
Gravity Vector	
Magnetic Field Strength	μT
Angular Rate	Dps
Euler Angles	Degrees
Quat.	Quat. Units
Temperature	°C

Appendix B

Micro Python

Micro Python Installation

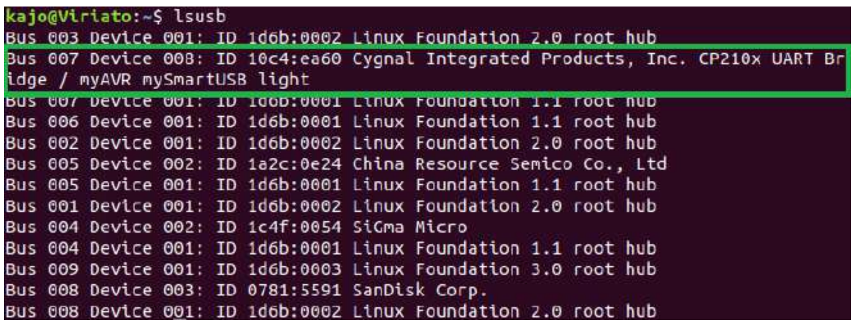
1. **Install** all the necessary software requirements on the machine terminal command prompt.

```
$ sudo apt-get install python3-pip
$ sudo pip3 install esptools
$ sudo pip3 install pyserial
$ sudo apt-get install picocom
$ sudo pip3 install adafruit-ampy
```

2. **Connect** the board to the machine.

3. **Check** which USB Port you are using

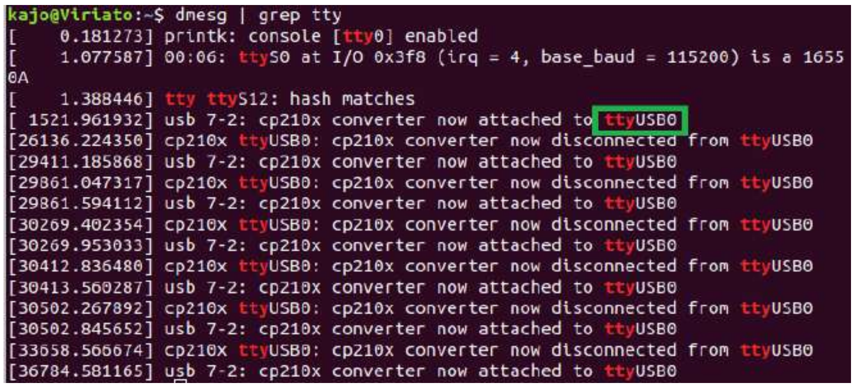
```
$ lsusb
```



```
kajo@Viriato:~$ lsusb
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 008: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Br
idge / myAVR mySmartUSB light
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 002: ID 1a2c:0e24 China Resource Semico Co., Ltd
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 002: ID 1c4f:0054 SiGma Micro
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 009 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 008 Device 003: ID 0781:5591 SanDisk Corp.
Bus 008 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Fig. B.1: MicroPython: List USB connections.

```
$ dmesg | grep tty
```

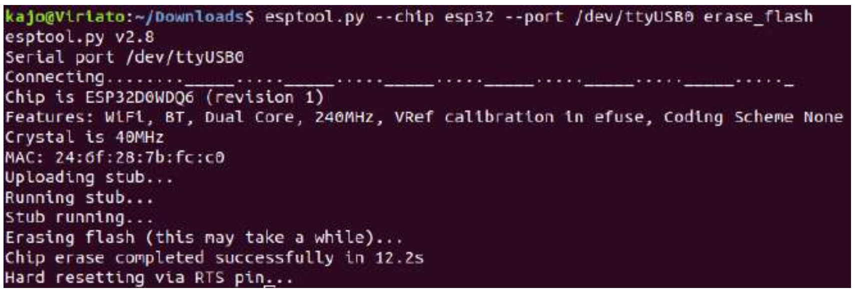


```
kajo@Viriato:~$ dmesg | grep tty
[    0.181273] printk: console [tty0] enabled
[    1.077587] 00:06: ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 1655
0A
[    1.388446] tty ttyS12: hash matches
[ 1521.961932] usb 7-2: cp210x converter now attached to ttyUSB0
[26136.224350] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[29411.185868] usb 7-2: cp210x converter now attached to ttyUSB0
[29861.047317] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[29861.594112] usb 7-2: cp210x converter now attached to ttyUSB0
[30269.402354] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[30269.953033] usb 7-2: cp210x converter now attached to ttyUSB0
[30412.836480] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[30413.560287] usb 7-2: cp210x converter now attached to ttyUSB0
[30502.267892] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[30502.845652] usb 7-2: cp210x converter now attached to ttyUSB0
[33658.566674] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0
[36784.581165] usb 7-2: cp210x converter now attached to ttyUSB0
```

Fig. B.2: MicroPython: Check last used USB Ports.

4. **Erase** the flash memory of the ESP32 and load the new firmware (depending on your board you might need to press the 'boot' button on both of this commands).

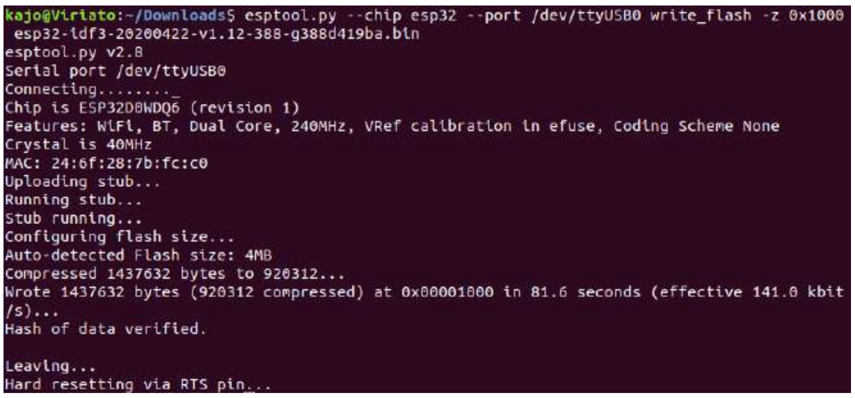
```
$ esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```



```
kajo@Viriato:~/Downloads$ esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
esptool.py v2.8
Serial port /dev/ttyUSB0
Connecting........_____....._____....._____....._____....._____....._
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:6f:28:7b:fc:c0
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 12.2s
Hard resetting via RTS pin...
```

Fig. B.3: MicroPython: Erase the flash memory.

```
$ esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z
0x1000 esp32-idf3-20200422-v1.12-388-g388d419ba.bin
```

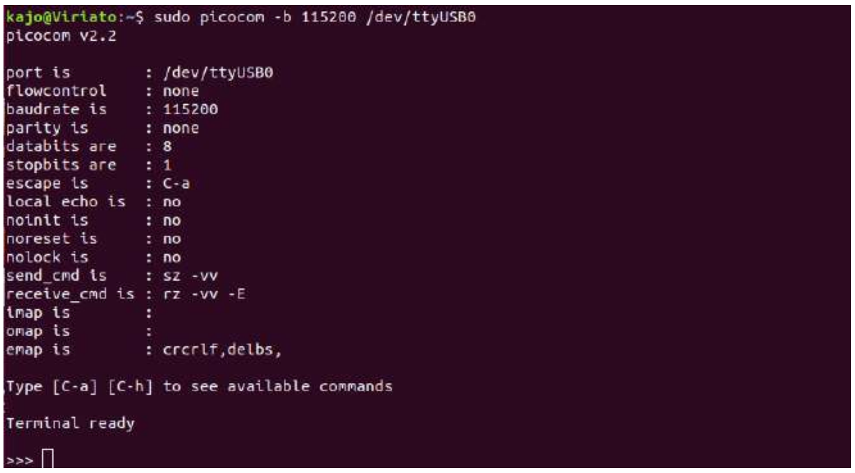


```
kajo@Viriato:~/Downloads$ esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000
 esp32-idf3-20200422-v1.12-388-g388d419ba.bin
esptool.py v2.8
Serial port /dev/ttyUSB0
Connecting........_
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:6f:28:7b:fc:c0
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1437632 bytes to 920312...
Wrote 1437632 bytes (920312 compressed) at 0x00001000 in 81.6 seconds (effective 141.0 kbit
/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Fig. B.4: MicroPython: Write in the flash memory.

5. **Connect** to the micro-controller and enter Python prompt.

```
$ sudo picocom -b 115200 /dev/ttyUSB0
```

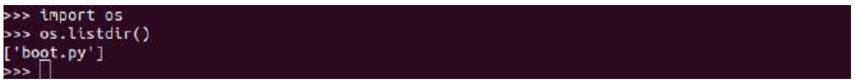


```
kajo@Viriato:~$ sudo picocom -b 115200 /dev/ttyUSB0
picocom v2.2

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Type [C-a] [C-h] to see available commands

Terminal ready

>>> 
```

Fig. B.5: MicroPython: Python prompt.

6. **Check** the current directory of the File System. It will contain the file *boot.py* that is the first file that it will automatically executed. If it also contains a **main.py** file, then that will be the second file to be executed on boot.

```
>>> import os
>>> os.listdir()
```

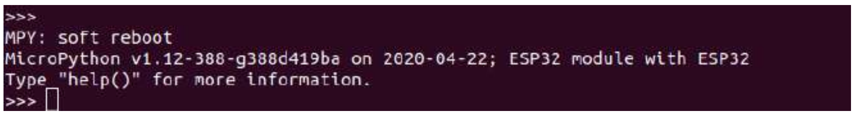


```
>>> import os
>>> os.listdir()
['boot.py']
>>> 
```

Fig. B.6: MicroPython: List directory.

Note that you can use the following shortcuts:

- CTRL - C to break
- CTRL - D for soft reboot
- CTRL - A and CTRL - X to exit

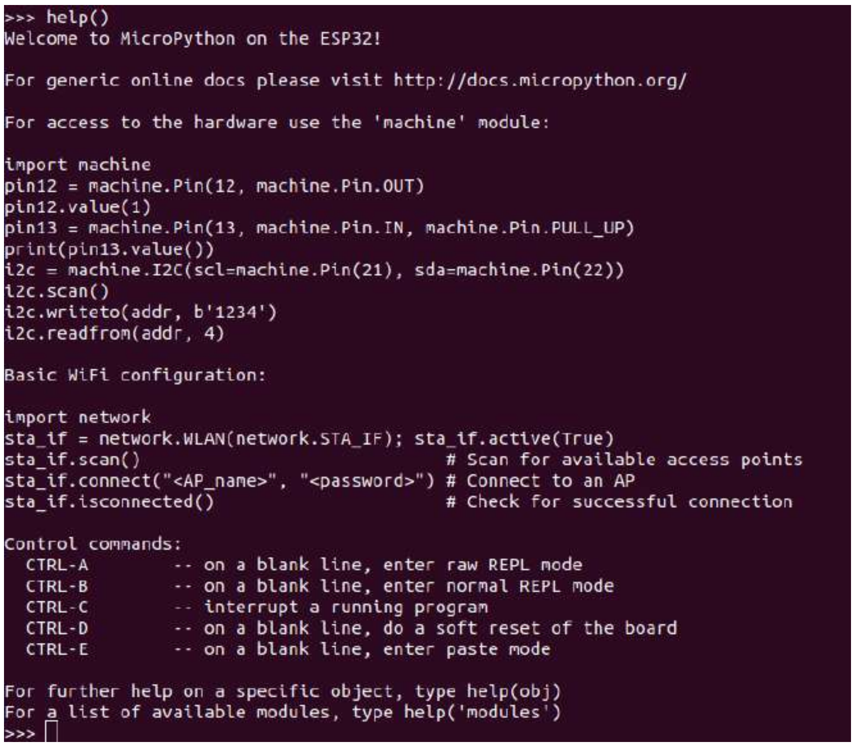


```
>>>
MPY: soft reboot
MicroPython v1.12-388-g388d419ba on 2020-04-22; ESP32 module with ESP32
Type "help()" for more information.
>>> 
```

Fig. B.7: MicroPython: Soft reboot.

7. An useful command is ***help***, to understand what the options are and how the libraries work.

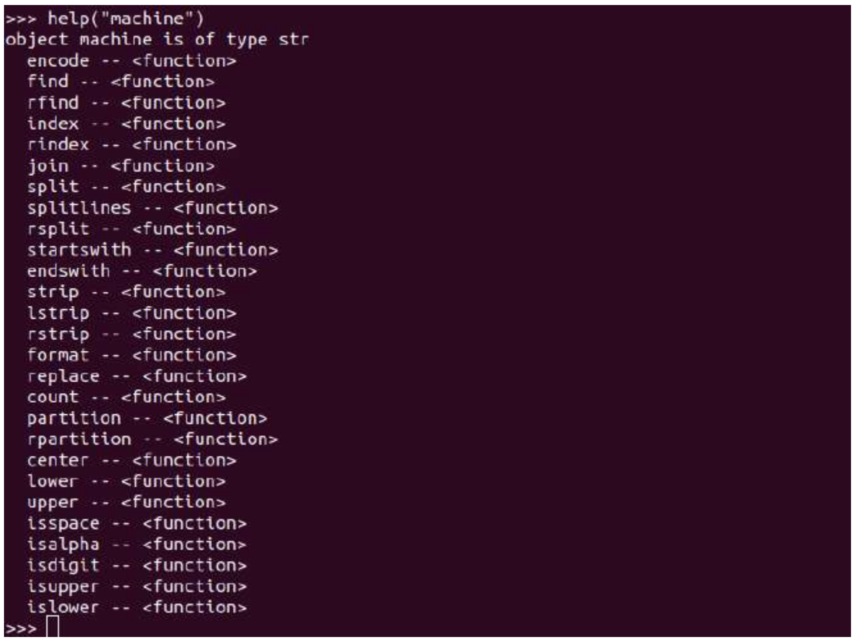
```
>>> help()
```



```
>>> help()
Welcome to MicroPython on the ESP32!

For generic online docs please visit http://docs.micropython.org/

For access to the hardware use the 'machine' module:

import machine
pin12 = machine.Pin(12, machine.Pin.OUT)
pin12.value(1)
pin13 = machine.Pin(13, machine.Pin.IN, machine.Pin.PULL_UP)
print(pin13.value())
i2c = machine.I2C(scl=machine.Pin(21), sda=machine.Pin(22))
i2c.scan()
i2c.writeto(addr, b'1234')
i2c.readfrom(addr, 4)

Basic WiFi configuration:

import network
sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
sta_if.scan()                             # Scan for available access points
sta_if.connect("<AP_name>", "<password>") # Connect to an AP
sta_if.isconnected()                      # Check for successful connection

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL-D        -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
>>> 
```

Fig. B.8: MicroPython: General help.

```
>>> help("modules")
```

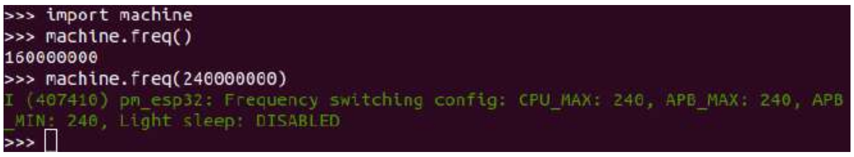


```
>>> help("machine")
object machine is of type str
  encode -- <function>
  find -- <function>
  rfind -- <function>
  index -- <function>
  rindex -- <function>
  join -- <function>
  split -- <function>
  splitlines -- <function>
  rsplit -- <function>
  startswith -- <function>
  endswith -- <function>
  strip -- <function>
  lstrip -- <function>
  rstrip -- <function>
  format -- <function>
  replace -- <function>
  count -- <function>
  partition -- <function>
  rpartition -- <function>
  center -- <function>
  lower -- <function>
  upper -- <function>
  isspace -- <function>
  isalpha -- <function>
  isdigit -- <function>
  isupper -- <function>
  islower -- <function>
>>> 
```

Fig. B.9: MicroPython: Help on a module.

8. It is also possible to check the clock frequency of the MCU and change it.

```
>>> import machine
>>> machine.freq()
```

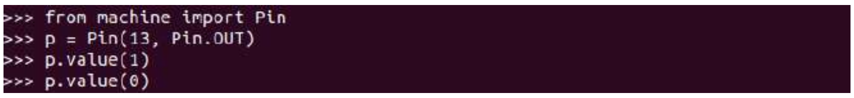


```
>>> import machine
>>> machine.freq()
160000000
>>> machine.freq(240000000)
I (407410) pm_esp32: Frequency switching config: CPU_MAX: 240, APB_MAX: 240, APB
_MIN: 240, Light sleep: DISABLED
>>> 
```

Fig. B.10: MicroPython: Change operating frequency.

9. Here's an example of GPIO interaction, where *GPIO13* is turned on and off.

```
>>> from machine import Pin
>>> p = Pin(13,Pin.OUT)
>>> p.value(1)
>>> p.value(0)
```



```
>>> from machine import Pin
>>> p = Pin(13, Pin.OUT)
>>> p.value(1)
>>> p.value(0)
```

Fig. B.11: MicroPython: Toggle pins.

10. To install python modules to the board, the method ***install*** from module ***upip*** shall be used.

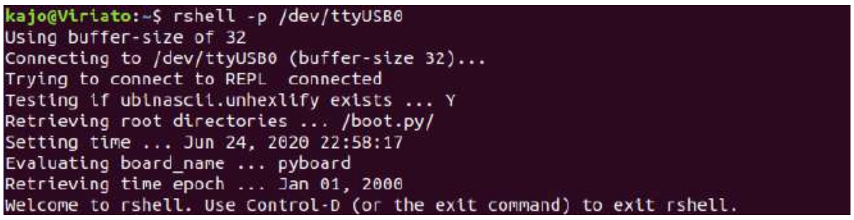
```
>>> upip.install('pickle')
```

11. Back to the command prompt, here's how to insert files into the board.

```
$ ampy --port /dev/ttyUSB0 put boot.py
```

12. Another usefull tool to interact with the ESP32 is RSHELL, available for free in GitHub.

```
$ sudo python3 setup.py install
$ rshell -p /dev/ttyUSB0
```

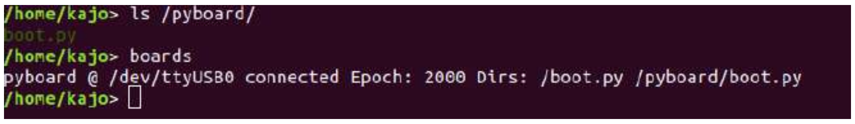


```
kajo@Viriato:~$ rshell -p /dev/ttyUSB0
Using buffer-size of 32
Connecting to /dev/ttyUSB0 (buffer-size 32)...
Trying to connect to REPL  connected
Testing if ubinascii.unhexlify exists ... Y
Retrieving root directories ... /boot.py/
Setting time ... Jun 24, 2020 22:58:17
Evaluating board_name ... pyboard
Retrieving time epoch ... Jan 01, 2000
Welcome to rshell. Use Control-D (or the exit command) to exit rshell.
```

Fig. B.12: MicroPython: RSHELL tool.

13. The following example describes how to check the directory of the board and how to check all the connected boards.

```
> ls /pyboard/
> boards
```



```
/home/kajo> ls /pyboard/
boot.py
/home/kajo> boards
pyboard @ /dev/ttyUSB0 connected Epoch: 2000 Dirs: /boot.py /pyboard/boot.py
/home/kajo>
```

Fig. B.13: MicroPython: List available boards.

14. It is also possible to copy files to the board by using the following command.

```
> cp *.py /pyboard/
```

MicroPython Code Examples

Several tests were made, that are presented next, like how to:

- Listing B.1: How to read GPIO's.
- Listing B.2: How to use PWM and ADC.
- Listing B.3: How to use a Pin Interrupt.
- Listing B.4: How to use a Timer.
- Listing B.5: How to communicate via UART.
- Listing B.6: How to use the RTC and DeepSleep Mode.

Listing B.1: File: GPIORead.py

```python
from machine import Pin
from time import sleep_ms

button = Pin(4, Pin.IN, Pin.PULL_UP)

while(True):
    print(button.value())
    sleep_ms(100)
```

Listing B.2: File: PwmAdc.py

```python
from machine import Pin, PWM, ADC

pwm_led = PWM(Pin(12))

pwm_led.duty(0) #0-1023; 0-100%
pwm_led.freq(1000)
print(pwm_led.freq())
print(pwm_led.duty())

while True:
    pot = ADC(0)
    analog = pot.read()
    pwm_led.duty=(analog)
```

Listing B.3: File: PinInterrupt.py

```python
# Libraries
from machine import Pin

# Pin definitions
led = Pin(5, Pin.OUT)
button = Pin(4, Pin.IN, Pin.PULL_UP)

def debounce(pin):
    prev = None
    for _ in range(32):
        current_value = pin.value()
        if prev != None and prev != current_value:
            return None
        prev = current_value
    return prev

def button_callback(pin):
    d = debounce(pin)

    if d == None:
        return
    elif not d:
        led.value(not led.value())

button.irq(trigger=Pin.IRQ_FALLING, handler=button_callback)
#Pin.IRQ_FALLING
#Pin.IRQ_RISING
#Pin.IRQ_FALLING | Pin.IRQ_RISING
```

Listing B.4: File: Timer.py

```python
# Libraries
from machine import Pin, Timer

# Pin definitions
led = Pin(5, Pin.OUT)
led.value(0)

timer = Timer(-1) # -1 -> Virtual Timer
#1000 ms, [2 options: ONE_SHOT or PERIODIC]
timer.init(period=1000, mode=Timer.PERIODIC, callback=lambda t:led.value(not led.value()))
#timer.deinit() # stop timer
```

Listing B.5: File: UART.py

```python
# Libraries
from machine import Pin, UART

# Start UART communication at UART0 at 1152000bps
uart = UART(0, 115200)
uart.init(115200, bits=8, parity=None, stop=1)
led = Pin(16, Pin.OUT)

# Bytes type
ch = b''

# Read, handle and send information
while True:
    # Checks if there is information in the buffer
    if uart.any() > 0:
        ch = uart.readline()        # Read buffer
        if ch = b'on':
            uart.write('Led On!')
            led.off()
        elif ch == b'off':
            uart.write('Led Off!')
            led.on()
        else:
            uart.write('Recebi: ')
            uart.write(ch)
```

Listing B.6: File: RtcDeepSleep.py

```python
from machine import RTC, Pin, Timer

# rtc = RTC()
# year, month, day, day of the week, hor , minute, second, milisecond
# Day of the week is calculated automatically by the other parameters
# the 0 is just a placeholder
#rtc.datetime((2020, 2, 19, 0, 14, 0, 0, 0))

led = Pin(5, Pin.OUT)
datetime_button = Pin(4, Pin.IN, Pin.PULL_UP)
deepsleep_button = Pin(14, Pin.IN, Pin.PULL_UP)

rtc = RTC()
rtc.irq(trigger = RTC.ALARM0, wake=machine.DEEPSLEEP)

def debounce(pin):
    prev = None
    for _ in range(32):
        current_value = pin.value()
        if prev != None and prev != current_value:
            return None
        prev = current_value
    return prev

def set_time_callback(pin):
    d = debounce(pin)

    if d == None:
        return
    elif not d:
        rtc.alarm(RTC.ALARM0, 5000)
        machine.deepsleep()

datetime_button.irq(trigger=Pin.IRQ_FALLING, handler=set_time_callback)
deepsleep_button.irq(trigger=Pin.IRQ_FALLING, handler=deepsleep_callback)

print("\n" + str(rtc.datetime()))

time = Timer(-1)
timer.init(period=1000, mode=Timer.PERIODIC, callback=lambda t: led.value(not led.value))
```

Appendix C

TFmini-S Tests

Cardboard Box Test

Tab. C.1: Cardboard box test.

Distance real (m)	**Distance measured** (m)	**Precision (%)**	**Accuracy (%)**
0.100	0.095	99.897	95.040
0.200	0.201	98.935	99.510
0.300	0.302	99.982	99.393
0.400	0.393	99.969	98.240
0.500	0.486	99.980	97.136
0.600	0.587	99.972	97.893
0.700	0.693	99.974	98.952
0.800	0.792	99.959	98.990
0.900	0.899	99.959	99.884
1.000	0.990	99.989	98.968
1.100	1.081	99.831	98.276
1.200	1.186	99.912	98.837
1.300	1.284	99.984	98.738
1.400	1.390	99.986	99.300
1.500	1.490	99.986	99.300
1.600	1.592	99.985	99.510
1.700	1.676	99.981	98.611
1.800	1.790	99.874	99.443
1.900	1.892	99.997	99.581
2.000	1.993	99.997	99.646
2.100	2.092	99.995	99.595
2.200	2.192	99.996	99.658
2.300	2.288	99.995	99.457
2.400	2.387	99.994	99.450
2.500	2.494	99.994	99.777
2.600	2.589	99.994	99.593
2.700	2.683	99.958	99.380
2.800	2.789	99.994	99.606
2.900	2.890	99.995	99.644
3.000	2.988	99.994	99.593
3.100	3.089	99.993	99.656
3.200	3.192	99.992	99.737
3.300	3.339	99.992	98.812
3.400	3.447	99.992	98.604
3.500	3.556	99.993	98.387
3.600	3.801	99.992	94.421
3.700	3.884	99.956	95.037
3.800	3.978	99.992	95.313
3.900	4.064	99.991	95.794
4.000	4.100	99.993	97.488

Black Box Test

Tab. C.2: Black box test.

Distance real (m)	**Distance measured** (m)	**Precision (%)**	**Accuracy (%)**
0.100	0.102	99.866	98.500
0.200	0.194	99.954	97.100
0.300	0.287	99.963	95.807
0.400	0.377	99.969	94.200
0.500	0.462	99.626	92.480
0.600	0.561	99.646	93.527
0.700	0.668	99.982	95.480
0.800	0.781	99.977	97.682
0.900	0.859	99.985	95.418
1.000	0.970	99.987	96.970
1.100	1.070	99.991	97.245
1.200	1.166	99.993	97.148
1.300	1.269	99.991	97.595
1.400	1.366	99.994	97.581
1.500	1.458	99.994	97.212
1.600	1.574	99.994	98.364
1.700	1.664	99.993	97.889
1.800	1.772	99.994	98.428
1.900	1.876	99.994	98.722
2.000	1.975	99.993	98.764
2.100	2.072	99.992	98.676
2.200	2.185	99.993	99.339
2.300	2.282	99.992	99.922
2.400	2.378	99.993	99.069
2.500	2.484	99.991	99.352
2.600	2.580	99.993	99.245
2.700	2.688	99.990	99.561
2.800	2.786	99.988	99.497
2.900	2.881	99.987	99.347
3.000	2.984	99.986	99.461
3.100	3.082	99.987	99.408
3.200	3.236	99.900	98.860
3.300	3.399	99.984	96.988
3.400	3.553	99.984	95.492
3.500	3.750	99.985	92.866
3.600	3.961	99.984	89.983
3.700	4.102	99.989	89.126
3.800	4.153	99.988	90.703
3.900	4.199	99.989	92.344
4.000	4.219	99.986	94.525

Mirror Test

Tab. C.3: Mirror test.

Distance real (m)	**Distance measured** (m)	**Precision (%)**	**Accuracy (%)**
0.100	0.191	99.739	8.760
0.200	0.333	99.939	33.350
0.300	0.451	99.974	49.800
0.400	0.526	99.922	68.565
0.500	0.569	99.800	86.112
0.600	0.988	99.935	35.343
0.700	0.958	99.923	63.200
0.800	0.824	99.979	96.943
0.900	0.993	99.940	89.668
1.000	1.048	99.857	95.152

Appendix D

Board Design

1. **Select the components**, make the necessary connections and add a disconnected marker to the pins that are not connected anywhere. Some comments and border lines were put between different parts of the circuit to make it more "readable". In the end of this process, the schematic should look like Fig. D.1.

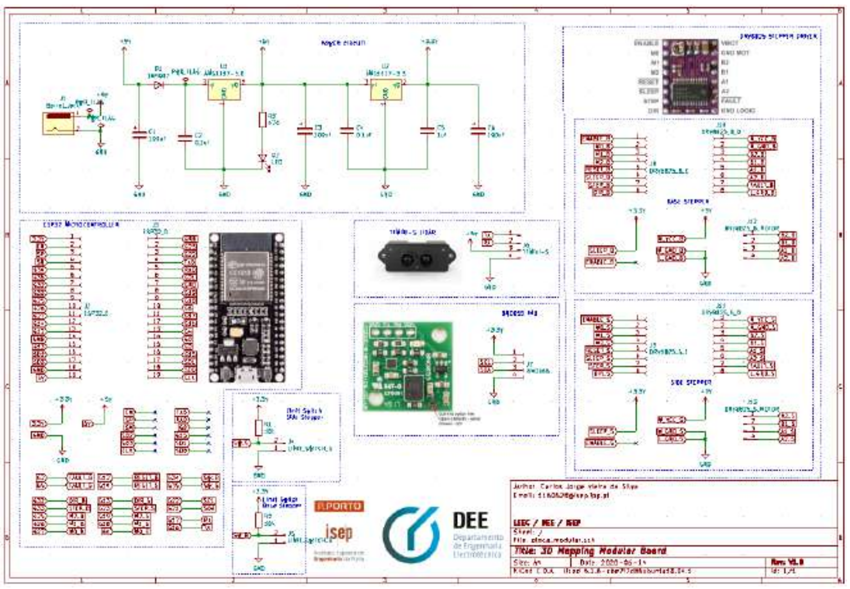
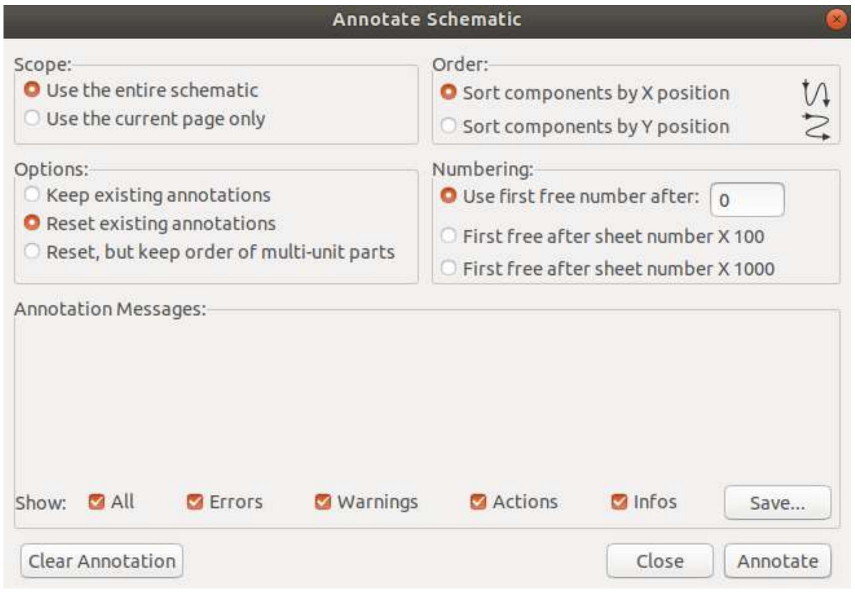


Fig. D.1: KiCad Step 1: Design the Schematics.

2. **Annotate the schematic**, components need to have labels (e.g.R_1 and R_2, for two different resistors) so that they can be differentiated from others with the similar characteristics. On the top menu, there is an icon with a pen writing on a paper. Press it and a similar window to Fig. D.2 will open. Select your desirable options for the annotation and the green message "Annotation Complete" shall appear.


Annotate Schematic

Scope:
[✓ Use the entire schematic] Use the current page only

Order:
[✓ Sort components by X position] Sort components by Y position

Options:
Keep existing annotations [✓ Reset existing annotations] Reset, but keep order of multi-unit parts

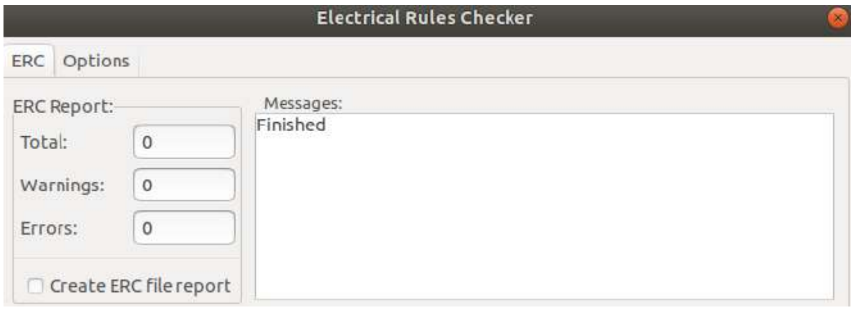
Numbering:
[✓ Use first free number after: 0] First free after sheet number X 100 First free after sheet number X 1000

Annotation Messages:

Show: ✓ All ✓ Errors ✓ Warnings ✓ Actions ✓ Infos

Fig. D.2: KiCad Step 2: Annotate the Schematics.

3. **Verify the electrical connections** to make sure no mistakes were made during the first steps. On the top menu, there is an icon with a ladybug, that shall be located next to the icon of the step before. Press it and run the test, if no mistakes were made, then the total number of warnings and errors shall be equal to zero, and the window will be similar to Fig. D.3.


Electrical Rules Checker

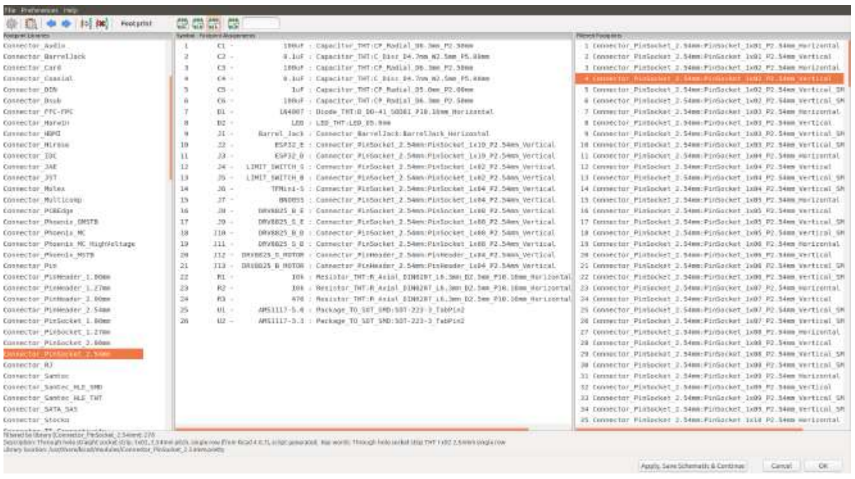
ERC Report:
Total: 0
Warnings: 0
Errors: 0

Messages:
Finished

Create ERC file report

Fig. D.3: KiCad Step 3: Verify the Schematics Electrical Connections.

4. **Assign footprints** to the schematic symbols. Footprints are the 2D arrangement of the components, used to physically attach and electrically connect a component to the board. On the top menu, there is an icon that contains two icons (one is the icon of the symbol editor and the other is the icon of the footprint editor), that shall be located on the right of the icon of the step before. Press it and make the desired associations between components and their footprints. In the end, the window will be similar to Fig. D.4.



```
Connector_Audio
Connector_BarrelJack
Connector_Card
Connector_Coaxial
Connector_DIN
Connector_Dsub
Connector_FFC-FPC
Connector_Harwin
Connector_HDMI
Connector_Hirose
Connector_IDC
Connector_JAE
Connector_JST
Connector_Molex
Connector_Multicomp
Connector_PCBEdge
Connector_Phoenix_GMSTB
Connector_Phoenix_MC
Connector_Phoenix_MC_HighVoltage
Connector_Phoenix_MSTB
Connector_Pin
Connector_PinHeader_1.00mm
Connector_PinHeader_1.27mm
Connector_PinHeader_2.00mm
Connector_PinHeader_2.54mm
Connector_PinSocket_1.00mm
Connector_PinSocket_1.27mm
Connector_PinSocket_2.00mm
Connector_PinSocket_2.54mm
Connector_RJ
Connector_Samtec
Connector_Samtec_HLE_SMD
Connector_Samtec_HLE_THT
Connector_SATA_SAS
Connector_Stocko
```

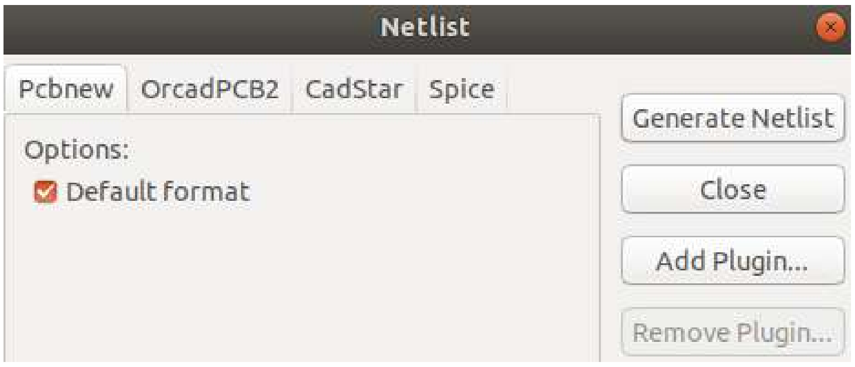
```
 1   C1 -              100uF : Capacitor_THT:CP_Radial_D6.3mm_P2.50mm
 2   C2 -              0.1uF : Capacitor_THT:C_Disc_D4.7mm_W2.5mm_P5.00mm
 3   C3 -              100uF : Capacitor_THT:CP_Radial_D6.3mm_P2.50mm
 4   C4 -              0.1uF : Capacitor_THT:C_Disc_D4.7mm_W2.5mm_P5.00mm
 5   C5 -                1uF : Capacitor_THT:CP_Radial_D5.0mm_P2.00mm
 6   C6 -              100uF : Capacitor_THT:CP_Radial_D6.3mm_P2.50mm
 7   D1 -             1N4007 : Diode_THT:D_DO-41_SOD81_P10.16mm_Horizontal
 8   D2 -                LED : LED_THT:LED_D5.0mm
 9   J1 -        Barrel_Jack : Connector_BarrelJack:BarrelJack_Horizontal
10   J2 -            ESP32_E : Connector_PinSocket_2.54mm:PinSocket_1x19_P2.54mm_Vertical
11   J3 -            ESP32_D : Connector_PinSocket_2.54mm:PinSocket_1x19_P2.54mm_Vertical
12   J4 -     LIMIT_SWITCH_S : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical
13   J5 -     LIMIT_SWITCH_B : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical
14   J6 -          TFMini-S  : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical
15   J7 -            BNO055  : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical
16   J8 -          DRV8825_B_E : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
17   J9 -          DRV8825_S_E : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
18   J10 -         DRV8825_B_D : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
19   J11 -         DRV8825_S_D : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
20   J12 -   DRV8825_S_MOTOR : Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical
21   J13 -   DRV8825_B_MOTOR : Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical
22   R1 -                10k : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal
23   R2 -                10k : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal
24   R3 -                470 : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal
25   U1 -        AMS1117-5.0 : Package_TO_SOT_SMD:SOT-223-3_TabPin2
26   U2 -        AMS1117-3.3 : Package_TO_SOT_SMD:SOT-223-3_TabPin2
```

```
 1 Connector_PinSocket_2.54mm:PinSocket_1x01_P2.54mm_Horizontal
 2 Connector_PinSocket_2.54mm:PinSocket_1x01_P2.54mm_Vertical
 3 Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Horizontal
 4 Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical
 5 Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical_SM
 6 Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical_SM
 7 Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm_Horizontal
 8 Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm_Vertical
 9 Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm_Vertical_SM
10 Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm_Vertical_SM
11 Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Horizontal
12 Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical
13 Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical_SM
14 Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical_SM
15 Connector_PinSocket_2.54mm:PinSocket_1x05_P2.54mm_Horizontal
16 Connector_PinSocket_2.54mm:PinSocket_1x05_P2.54mm_Vertical
17 Connector_PinSocket_2.54mm:PinSocket_1x05_P2.54mm_Vertical_SM
18 Connector_PinSocket_2.54mm:PinSocket_1x05_P2.54mm_Vertical_SM
19 Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Horizontal
20 Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical
21 Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical_SM
22 Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical_SM
23 Connector_PinSocket_2.54mm:PinSocket_1x07_P2.54mm_Horizontal
24 Connector_PinSocket_2.54mm:PinSocket_1x07_P2.54mm_Vertical
25 Connector_PinSocket_2.54mm:PinSocket_1x07_P2.54mm_Vertical_SM
26 Connector_PinSocket_2.54mm:PinSocket_1x07_P2.54mm_Vertical_SM
27 Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Horizontal
28 Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
29 Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical_SM
30 Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical_SM
31 Connector_PinSocket_2.54mm:PinSocket_1x09_P2.54mm_Horizontal
32 Connector_PinSocket_2.54mm:PinSocket_1x09_P2.54mm_Vertical
33 Connector_PinSocket_2.54mm:PinSocket_1x09_P2.54mm_Vertical_SM
34 Connector_PinSocket_2.54mm:PinSocket_1x09_P2.54mm_Vertical_SM
35 Connector_PinSocket_2.54mm:PinSocket_1x10_P2.54mm_Horizontal
```

```
Filtered by library (Connector_PinSocket_2.54mm): 270
Description: Through hole straight socket strip, 1x01, 2.54mm pitch, single row [illegible] Key words: Through hole socket strip THT 1x01 2.54mm single row
Library location: [illegible]
```

Fig. D.4: KiCad Part 4: Assign Footprints.

5. **Generate a netlist.** A netlist is a file that contains the information about the components, such as their identifier labels and footprints and also the electrical connections between the components on the circuit board and their respective footprints. On the top menu, there is an icon with ”NET” written on it, that shall be located on the right of the icon of the step before. Press it and a similar window to Fig. D.5 will open. Press ”Generate Netlist” and the respective file will be created. After this, open the Pcbnew editor to conclude the last steps.



Options:
✓ Default format

Fig. D.5: KiCad Step 5: Generate a netlist.

6. **Define the board size and organize the components.** If the project has size limitations, it might be better to start with defining the board size. To define the limits of the board, the ”Edge Cuts” layer must be used, and them defined by a polygonal line. It is also useful to write some comments on the board and to draw some icons. In the end, the window will be similar to Fig. D.6.

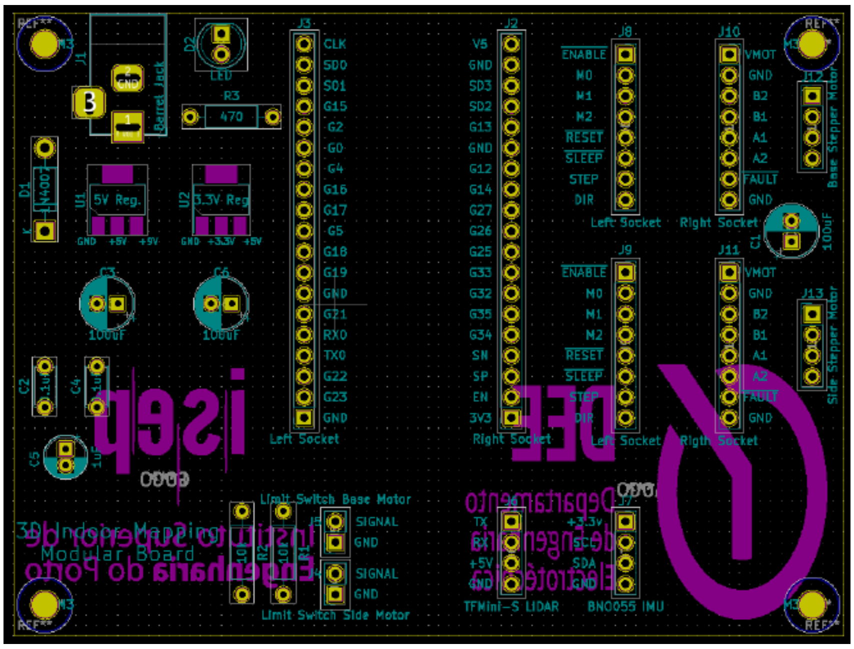


Fig. D.6: KiCad Step 6: Board size and organization.

7. **Add tracks.** To make the physical connections of the board, some copper tracks must be added. Bigger current levels need wider tracks so that they don't overheat or be damaged. The strategy used was to connect the closest parts together first, then the power connections, then the rest of the connections and finally fill the rest of the board with GND zones. In the end, the window will be similar to Fig. D.7.

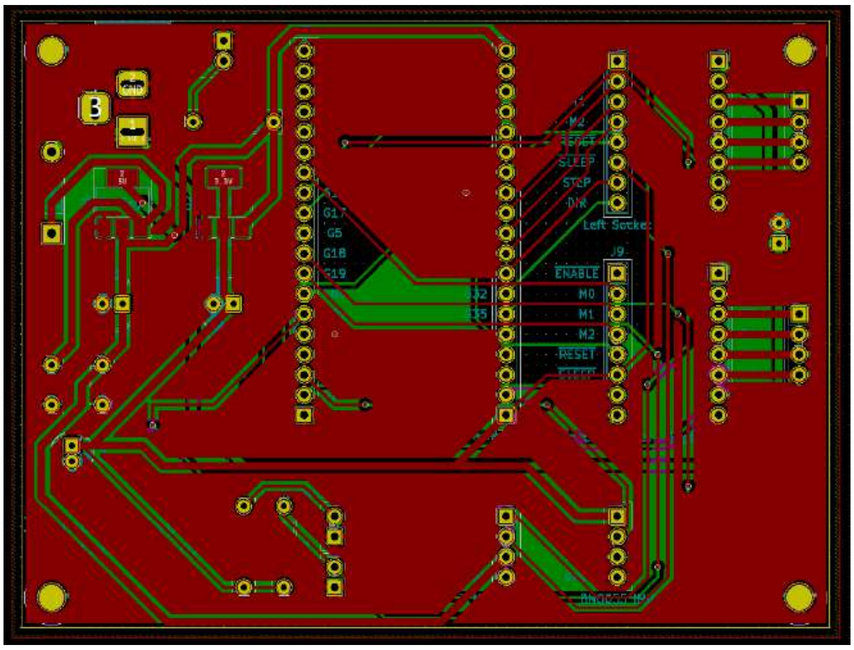
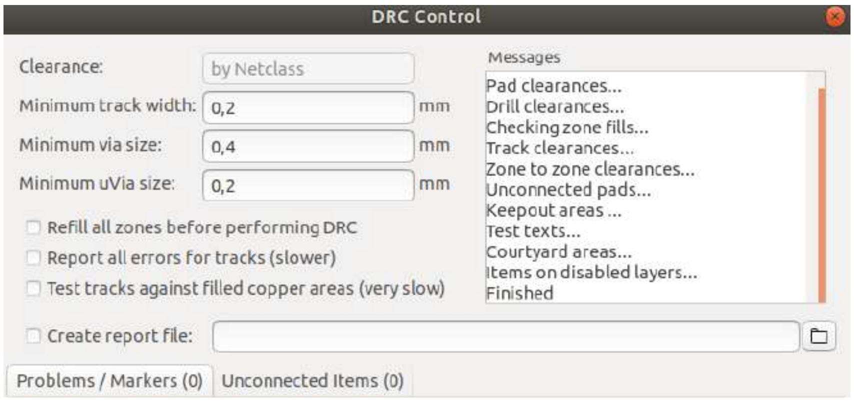


Fig. D.7: KiCad Step 7: Add tracks.

8. **Verify the layout** electrical connections to make sure no mistakes were made during this last process. On the top menu, there is an icon with a ladybug, press it and run the test, if no mistakes were made, then the total number of warnings and errors shall be equal to zero, and the window will be similar to Fig. D.8.


DRC Control

Clearance: by Netclass
Minimum track width: 0,2 mm
Minimum via size: 0,4 mm
Minimum uVia size: 0,2 mm

☐ Refill all zones before performing DRC
☐ Report all errors for tracks (slower)
☐ Test tracks against filled copper areas (very slow)

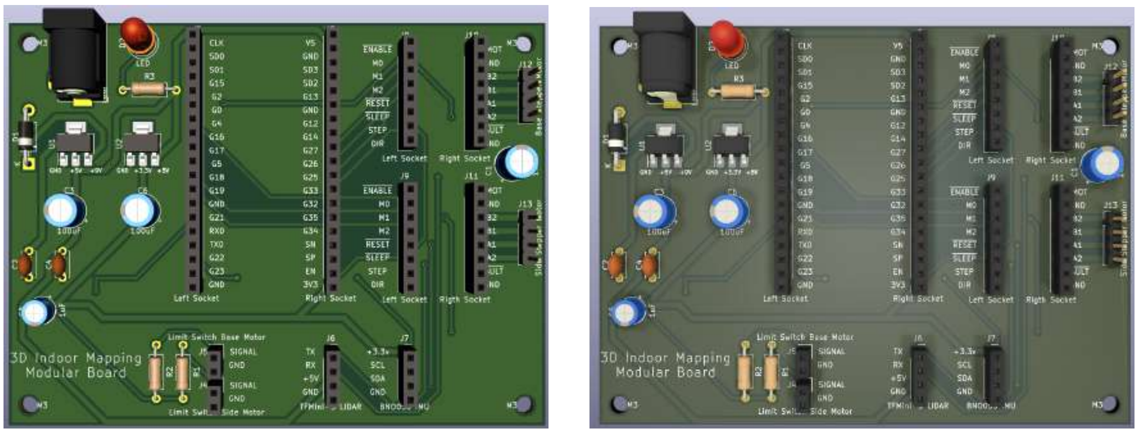
☐ Create report file:

Messages
```
Pad clearances...
Drill clearances...
Checking zone fills...
Track clearances...
Zone to zone clearances...
Unconnected pads...
Keepout areas ...
Test texts...
Courtyard areas...
Items on disabled layers...
Finished
```

Problems / Markers (0) Unconnected Items (0)

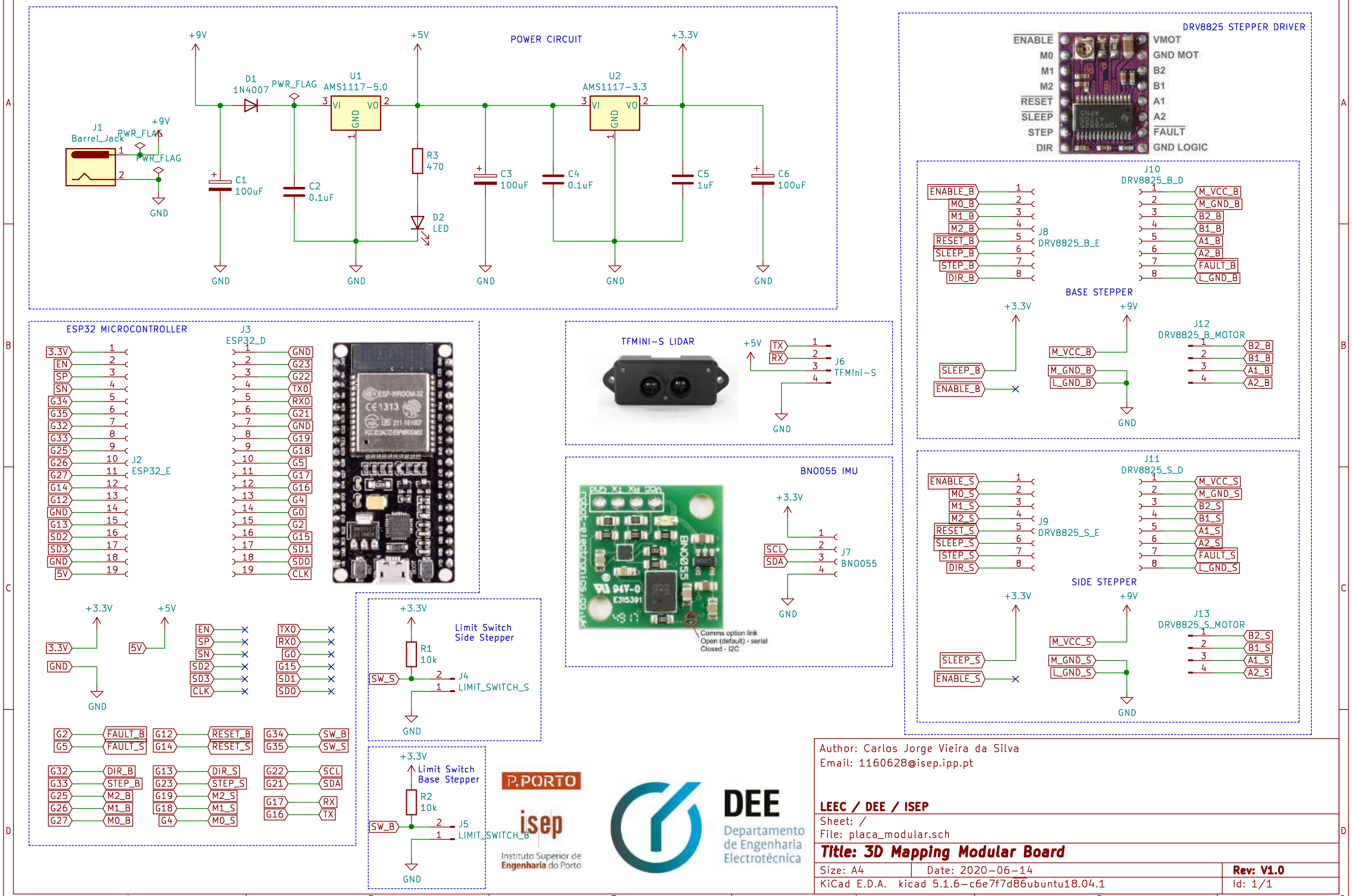
Fig. D.8: KiCad Step 8: Verify the Layout Electrical Connections.

9. **Check the 3D View** of the board. This might be useful to get an idea on how the board will look like once its completed and to maybe correct some design details. The board 3D view will be similar to Fig. D.9. GND is the most common connection of the all board, so this way is a lot more easier to make those connections than to make them manually. It is also gives a more appealing look to the board and protects it better the board against short circuits because most of it is grounded.



(a) Raytracing Off (b) Raytracing On

Fig. D.9: Board 3D Top View.



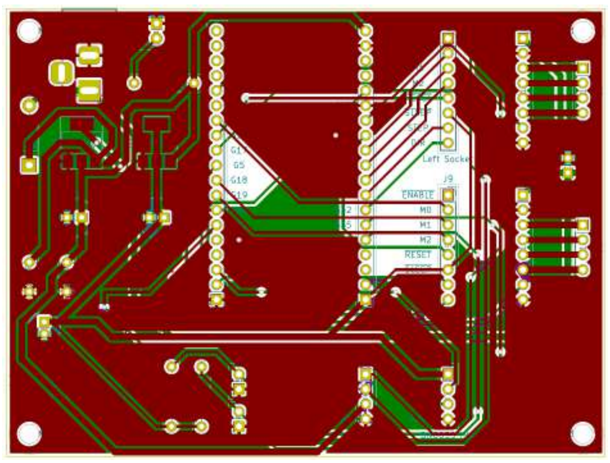
Author: Carlos Jorge Vieira da Silva
Email: 1160628@isep.ipp.pt

LEEC / DEE / ISEP

Sheet: /
File: placa_modular.sch

Title: 3D Mapping Modular Board

Size: A4 Date: 2020-06-14 **Rev: V1.0**
KiCad E.D.A. kicad 5.1.6-c6e7f7d86ubuntu18.04.1 Id: 1/1


(a) All layers.

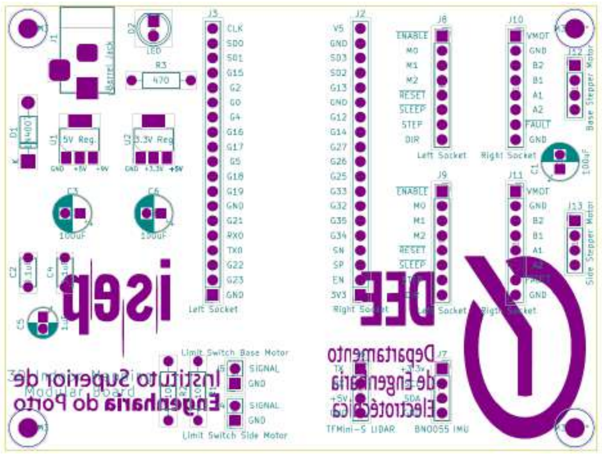
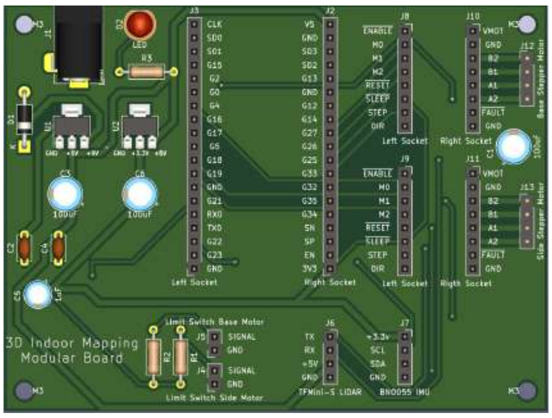
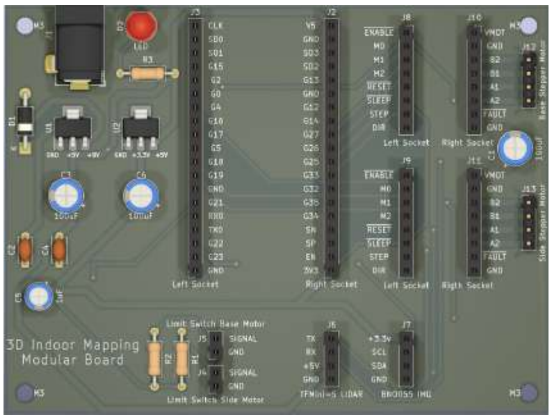

(b) No copper layers.

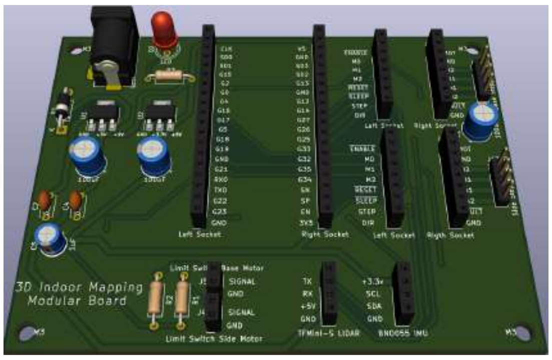
Fig. D.10: Board Layout View.



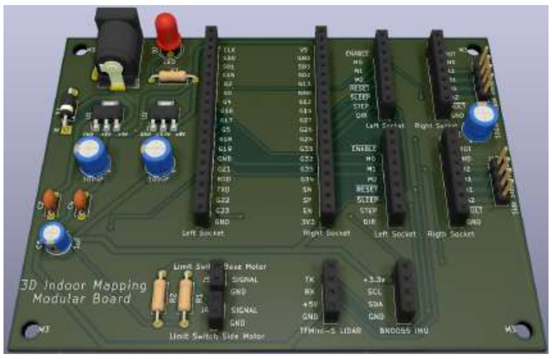
(a) Top View / Raytracing Off



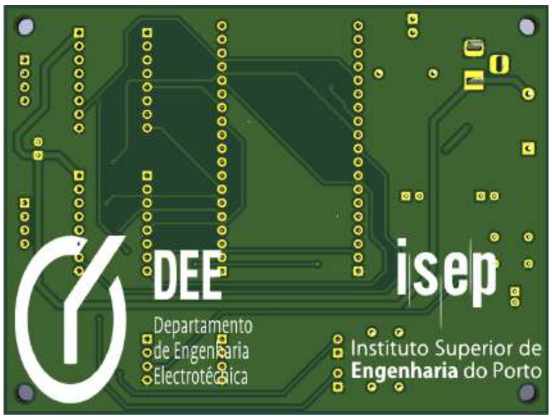
(b) Top View / Raytracing On



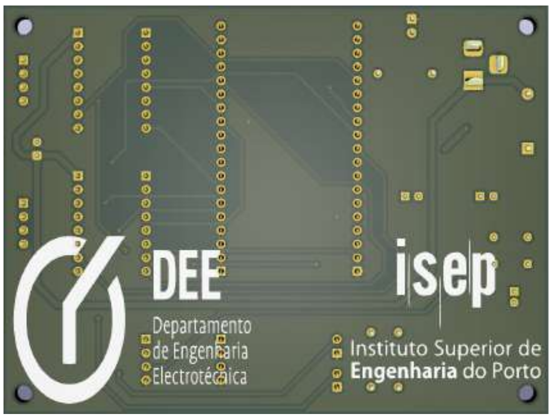
(c) Middle View / Raytracing Off



(d) Middle View / Raytracing On



(e) Back View / Raytracing Off



(f) Back View / Raytracing On

Fig. D.11: Board 3D Views.