
Laser Scanner for 3D Indoor Mapping

Project Report
ED3-1-E18

Aalborg University
Electronics and Computer Engineering

Copyright © Aalborg University 2018

This report was typeset using L^AT_EX



Electronics and Computer Engineering
Aalborg University
<http://www.aau.dk>

AALBORG UNIVERSITY

STUDENT REPORT

Title:

Laser Scanner for 3D Indoor Mapping

Theme:

Microprocessor Based Systems

Project Period:

Fall Semester 2018

Project Group:

ED3-1-E18

Participant(s):

Devrat Singh
Jegors Kirilovskis
Matei Căciuleanu
Sergio Hernández

Supervisor(s):

D. M. Akbar Hussain
Daniel Ortiz-Arroyo

Copies: 1

Number of Pages: 87

Date of Completion:

December 18, 2018

Abstract:

In the modern world, activities within many sectors of economy are contingent on having geometric data of physical structures available in a digital format. In response to this real-world need, the present report puts forward a particular solution in the form of a 3D scanning system, which could be utilized for indoor mapping. The system has a microcontroller at its core and integrates a stepper and a servo motor with a LiDAR sensor. Data comprising distance measurements and angular positioning is processed and then displayed as a point cloud with the aid of MATLAB. The entire system is brought to a working condition and tested in a real environment. Results show that the developed piece of technology is able to successfully map its surroundings by producing a comprehensible point cloud.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the authors.

Table of Contents

Preface	xi
1 Introduction	1
1.1 Introduction	1
1.2 Initiating Problem	2
2 Problem Analysis	3
2.1 Six Ws	3
2.2 Structures of Interest	4
2.3 3D Data Acquisition	5
2.3.1 Laser Scanning	5
2.3.2 Photogrammetry	6
2.3.3 Integrated Approach	6
2.4 Applications of 3D Scanning	7
2.4.1 Virtual Reality and Game Design	7
2.4.2 Building Information Modeling (BIM)	7
2.4.3 Accident Scene Reconstruction	7
2.4.4 Historical Building Documentation	8
2.4.5 Architecture and Construction	8
2.4.6 Urban Modeling	9
2.4.7 Underground Railroad Survey	9
2.5 Possible Solutions	10
2.5.1 MLS	10
2.5.2 TLS	10
2.6 Stakeholder Analysis	11
2.6.1 Interested Parties	12
2.6.2 Actors	12
2.6.3 Drivers of Technology	12
2.7 Legislation and Standards	12
2.8 Risk and Safety	13
2.9 Problem Statement	14

2.10	Project Specifications	14
2.11	Project Delimitations	16
3	Methods and Materials	17
3.1	System Overview	17
3.2	Arduino Uno	18
3.3	LiDAR	19
3.3.1	Theory of Operation	20
3.3.2	Limitations	21
3.4	Stepper Motor	22
3.4.1	Motor Selection	22
3.4.2	Working Principle	23
3.4.3	NEMA 17 Stepper Motor SM-42BYG011-25	23
3.5	Stepper Motor Controller	23
3.5.1	Motivation	23
3.5.2	Basic Control Circuit	24
3.5.3	Excitation Modes	24
3.5.4	Microstepping	25
3.5.5	EasyDriver	26
3.6	Servo Motor	27
3.6.1	Motor Selection	27
3.6.2	Internal Structure of a Servo Motor	28
3.6.3	Working Principle	28
3.6.4	Servo Control	29
3.6.5	Metal Gear Servo	29
3.7	Spherical Coordinates	30
3.8	Power Estimation	31
3.9	Battery	32
3.9.1	Battery Selection	32
3.9.2	UL 2.4-12 Lead Acid Battery	32
3.10	Buck Converter	33
3.10.1	Selection	33
3.10.2	Working Principle	33
3.10.3	XL4015 LM2596 Step-Down Converter	34
3.11	MATLAB	34
3.12	Point Cloud Filtering	35
4	Implementation	37
4.1	Data Transfer and Plotting Test	37
4.2	LiDAR Experiment	39
4.3	Stepper Motor Experiment	42
4.4	Servo Motor Experiment	44

4.5	3D Modelling	46
4.6	Hardware Implementation	47
4.6.1	Coordinate Acquisition	47
4.6.2	Movement Description	48
4.6.3	Power Transmission During Rotation	48
4.7	Arduino Programming	49
4.8	MATLAB Programming	52
4.8.1	User Interface	52
4.8.2	Main Script	53
4.8.3	Displaying Prior Point Clouds	55
4.9	Schematic	55
4.10	Prototype	56
5	Testing	57
5.1	Motivation	57
5.2	Selection of Testing Area	57
5.3	Setup	58
5.4	Performed Scans	58
5.5	Scanning Time	62
5.6	Distance Measurement	63
6	Discussion	65
7	Conclusion	67
	Bibliography	68
A	Arduino Code	74
B	MATLAB Code	80
B.1	User Interface	80
B.2	Main Code	81
B.3	Functions	82
B.3.1	Serial Communication	82
B.3.2	Write Data	82
B.3.3	Read Data	83
B.3.4	Calculate Scan Time	83
B.3.5	Coordinate Conversion	83
B.3.6	Display Current Point Cloud	84
B.3.7	Load and Display a Specific Point Cloud	84
C	Obtained Point Clouds	86

List of Figures

2.1	Six Ws Diagram	3
2.2	Stakeholder Diagram	11
3.1	Diagram of General Idea	18
3.2	System Block Diagram	18
3.3	Arduino Uno Rev3 (ATmega328P)	19
3.4	LiDAR-Lite v3HP	20
3.5	LiDAR Beam (Diffuse Surface) [38]	21
3.6	LiDAR Beam (Specular Surface) [38]	21
3.7	NEMA 17 Stepper Motor SM-42BYG011-25	23
3.8	Bipolar Stepper Dual H-Bridge Drive Circuit	24
3.9	Full Stepping Current Wave Forms	25
3.10	Microstepping Current Wave Forms	26
3.11	EasyDriver Bipolar Stepper Motor Driver	26
3.12	Servo Control Loop	29
3.13	Servo Control Signal	29
3.14	14g Servo Motor with Metallic Gears	29
3.15	Spherical Coordinates of a Point P	30
3.16	Buck Converter Basic Circuit	33
3.17	XL4015 LM2596 Buck Converter	34
3.18	Filtered/Unfiltered LLSR Demonstration	35
3.19	Cube with Uniformly Distributed Noise	36
3.20	Cube After Applying <code>pcdenoise</code>	36
4.1	Point Cloud of the Test Sphere	39
4.2	Results of LiDAR Test for 20 cm	40
4.3	Results of LiDAR Test for 100 cm	41
4.4	Results of LiDAR Test for 350 cm	41
4.5	Oscilloscope Screenshot of Full Stepping Current and Step Signals	43
4.6	Oscilloscope Screenshot of Microstepping Current and Step Signals	44

4.7	Oscilloscope Screenshot of Servo Current and Control Signal for Idle Current	45
4.8	Oscilloscope Screenshot of Servo Current and Control Signal for Stall Current	45
4.9	Structure Design With Components	46
4.10	Spherical Coordinate Acquisition Using LiDAR	48
4.11	Point Data Sent Through Serial Communication	50
4.12	Arduino Code Flowchart	51
4.13	The 3D Scanner User Interface in MATLAB App Designer	52
4.14	Flowchart of MATLAB Code	54
4.15	Circuit Schematic for 3D LiDAR Scanner	55
4.16	Final Setup of the Developed Prototype	56
5.1	Scan 2 Point Cloud in MATLAB	59
5.2	Testing Area (Side A)	60
5.3	Testing Area (Side B)	60
5.4	Scan 1 Point Cloud (Side A)	60
5.5	Scan 1 Point Cloud (Side B)	60
5.6	Scan 1 Point Cloud (Side A), Enlarged	61
5.7	Scan 2 Point Cloud (Side A)	61
5.8	Scan 3 Point Cloud (Side A)	62
C.1	Scan 4 Point Cloud (Side A)	86
C.2	Scan 5 Point Cloud (Side A)	87

List of Tables

- 3.1 EasyDriver Microstepping Configurations 27
- 3.2 Power Estimation 31

- 5.1 Scanning Parameters 58
- 5.2 Estimated and Measured Scan Duration 63

Preface

The project entitled "Laser Scanner for 3D Indoor Mapping" was written by four students from the Electronics and Computer Engineering Bachelor's program at Aalborg University Esbjerg, for the P3 project in the third semester. Hereafter, every mention of "we" refers to the four co-authors listed below.

The completion of this project would not have been possible without the contribution of our supervisors, D. M. Akbar Hussain and Daniel Ortiz-Arroyo, who directed their efforts towards guiding us to the right path. We would like to express our gratitude for the time they took for providing their support and relevant advice.

Aalborg University, December 18, 2018



Devrat Singh
<dsingh17@student.aau.dk>



Jegors Kirilovskis
<jkiril17@student.aau.dk>



Matei Căciuleanu
<ccaciu17@student.aau.dk>



Sergio Hernández
<shern17@student.aau.dk>

Chapter 1

Introduction

1.1 Introduction

We live in a three-dimensional (3D) world where knowledge about the geometry of physical structures is important from both an industrial and a societal point of view. Architecture, construction, medicine and quality control are just a few examples of sectors that are heavily dependent on using highly precise measurements regarding tangible structures on a regular basis. With the advancement of technology, gaining access to this type of information became easier, as new solutions for gathering data and generating a 3D digital model out of it emerged, in the form of 3D scanning. In fact, the spectrum of applications of these new technologies became more diverse as well. Modern archaeology, for instance, relies on the use of these new solutions for heritage preservation and restoration. Yet another example is represented by the gaming industry, which is utilizing the technology to confer an impression of reality to the user. In the past decades, the other aforementioned sectors turned to the use of digital solutions as well, for they bring the benefits of increasing accuracy and decreasing costs [1, 2, 3, 4].

Collecting geometric data is thus a relevant requisite nowadays and the demand for solutions is high. Out of the smorgasbord of 3D scanning techniques, laser scanning presents itself as one of the most prominent ones. This report scrutinizes the aspects of 3D laser scanning by using different tools and narrows the focus down to a particular application that can be accomplished by following the identified technical specifications. It also delineates the process of developing a solution, presenting the materials and methods utilized, as well as the implementation procedure, which illustrates how individual parts are combined into an electromechanical system, i.e., a 3D laser scanner. A conclusion is drawn in the end, as a result of testing performed with the developed device.

1.2 Initiating Problem

On the basis of the aforementioned background, the following question initializes the problem:

How can we obtain a three-dimensional digital model of a physical structure?

Chapter 2

Problem Analysis

2.1 Six Ws

In order to get started in the process of analyzing the problem, six essential questions were taken into consideration. Figure 2.1 provides a summary of this preliminary examination.

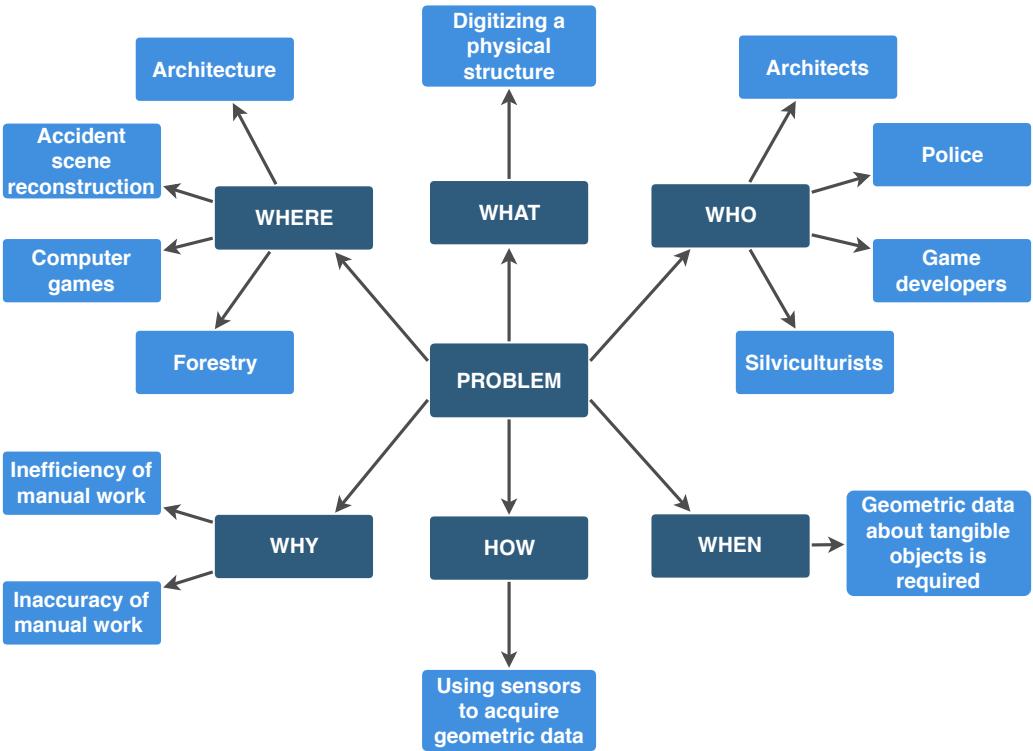


Figure 2.1: Six Ws Diagram

What is the problem?

In the professional sectors where the dimensions of the items of interest are relatively short, the usual way is to measure an object or space manually in order to analyze its physical structure. That entails spending human resources on a low-complexity task, therefore increasing costs and time elapsed for a certain project.

Where does the problem occur?

A large variety of environments, both natural and artificial, can be digitized with interesting applications in architecture, computer game development or forestry [5, 6]. For instance, in architecture, digitization can be useful for analyzing the stability and conservation state of old buildings or to optimize the process of redecorating a house [7].

When does the problem occur?

Every time it is needed to analyze or measure a three-dimensional space for any purpose, digitization can prove useful.

Who does the problem affect?

The spectrum of people affected includes professionals of different fields: architects, game developers or silviculturists.

Why does the problem occur?

Making a map based on manual drawings and measurements can be a hard and tedious task.

How can the problem be solved?

In order to get a digital representation of certain structures, the most logical way is to use sensors that can obtain the shape of a solid object in front of them. The sensor readings will give a number of points and, after a sufficient amount of readings are performed, while changing the position of the sensor if necessary, a point cloud can be obtained.

2.2 Structures of Interest

As it was seen in the foregoing sections, information about the geometry of tangible structures is valuable in many fields. However, making a complete analysis encompassing all the cases in which this information is used for a particular purpose would be a laborious task. For this reason, certain application sectors shall be

discarded from the very beginning of this project and thus the analysis will focus only on a smaller range of uses.

A very crude classification of the structures whose geometric data is needed in a digital format can be made by following a dimension criterion. On the one hand, there are the (relatively) small objects, category that includes artefacts or human body parts [8]. On the other hand, there are the much larger structures, such as terrain or buildings [9, 10]. Even though working with both categories entails interesting applications and deals with problems that affect different stakeholders, it was decided to carry the analysis solely on the latter, that is, transforming a large physical structure into digital format. Examples of applications concerning such structures will be mentioned in Section 2.4.

Now that it is known which objects fit the scope of this project, it is time to look into possible ways the required data could be gathered and transferred to the digital world.

One can argue that measurements of a structure could be taken manually - with a tape measure, for example - and then a digital model based on the measurements could be made with the aid of CAD software. While this is certainly possible, it is not a viable solution, since it is time consuming, as S. Klimoski stated in a 2006 report dedicated to modeling the interior of a theater in Minnesota with the objective of renovating it. He estimated that, if the work had been done manually, it would have taken 4 people around 12 weeks to perform the task. With the use of technology, however, the project required only 2 weeks and 2 workers for completion [10]. This brings into discussion a very important aspect: a piece of equipment able to take measurements of a given structure automatically by performing 3D data acquisition.

2.3 3D Data Acquisition

Three-dimensional data acquisition is the process of obtaining "point clouds or volumetric data by using established mechanisms or phenomena for interacting with the surface or volume of an object of interest" [11]. A device that performs 3D data acquisition is regarded as a 3D scanner. There are many technologies available on the market but the remainder of this section is devoted only to the main 3D scanning technologies suitable for acquiring data about some objects of interest, such as buildings or open spaces.

2.3.1 Laser Scanning

Laser scanners (or time-of-flight scanners) make use of an object's property of reflecting light to measure the distance between the device and the target. Precisely, these devices are based on "the propagation delay of electromagnetic waves" [11].

They are well suited for medium to long-range scanning and applications in which a high degree of precision is required - the estimated accuracy for high-end products is 2-5mm - and they bring the advantage of rapid surveying. The technology constitutes the "point cloud-based approach" towards 3D data acquisition, since the result is a set of discrete points that make up the scanned object [12]. There are different methods of laser scanning, depending on the platform on which the sensor is placed. Thus, we can distinguish between terrestrial laser scanning (TLS) and mobile laser scanning (MLS) [13].

2.3.2 Photogrammetry

Photogrammetry represents the "image-based approach" towards 3D data acquisition and it relies on the process of taking numerous photographs of the same object from different angles using one or more cameras. While the accuracy of the model is not extremely high (errors of a few decimeters occur) [12], the strength of this technology lies in the ability to capture features such as texture or color, thus making a digital object aesthetically pleasing and giving the impression of higher resemblance with its real counterpart [14]. The acquired data is actually of a two-dimensional nature but 3D reconstruction can be done by software in order to obtain a point cloud [15, 16]. In this case as well, the sensor - here, a camera - can be mounted on different platforms, such as drones (aerial photogrammetry) or tripods (in architectural photogrammetry, for example) [17].

2.3.3 Integrated Approach

The combination of the two technologies leads to the so-called "integrated approach", where a laser scanner and a digital camera work in tandem and the result comprises all the desired features: high accuracy alongside texture and colour. More often than not, professional high-end scanners utilize the integrated approach for its completeness, but they are still branded as "laser scanners" [12].

Conclusion

The aforementioned technologies are part of the 3D scanning industry because they provide features suitable for specific applications, with research clearly illustrating the success of each one of them [10, 12, 15, 16, 4]. Considering the information provided above, the integrated approach seems the best one if getting the highest scan quality is the primary objective. It should be reiterated, however, that the choice really depends on the application.

2.4 Applications of 3D Scanning

The point cloud obtained as a result of 3D scanning is a useful asset which can be modelled into different forms that allows it to be utilized for various applications. Therefore, the section below will cover the application of 3D laser scanning techniques such as terrestrial and mobile laser scanning, as well as its combinational use with photogrammetry in multiple fields.

2.4.1 Virtual Reality and Game Design

For games and virtual reality, the developers are trying to create environments that could contribute to an immersive experience for the user. To do so, these environments are designed to be highly realistic but creating such detailed realistic surroundings manually can be painstakingly long and time consuming. Consequently, the industry is utilizing 3D scanning technology to scan real physical sites to create 3D environments that are detailed and accurate to the minuscule scale, which can then be rendered into the gaming or virtual reality platform. This process of rendering the scanned image into the platform instead of creating it on a computer makes the whole procedure time efficient and saves a lot of effort and resources that could be diverted elsewhere in the creation stage [18].

2.4.2 Building Information Modeling (BIM)

The BIM process is a digital flow of information among all the stakeholders such as engineers, architects, and construction professionals involved in an infrastructure project. This allows them to have access to all the information that incorporates every aspect of an asset, such as building design using 3D models, schedules etc. Furthermore, BIM is a collaborative effort to effectively plan, design, construct and manage buildings and infrastructure in general. One of the advantages of BIM, i.e., to capture information of a construction site, is being made possible with the use of 3D scanners which are used to scan the site even before the process begins. This can provide important information like ground elevation and 3D models of pre-existing infrastructures, therefore accurately capturing reality and greatly streamlining the process of project preparation [19]. Additionally, information modeling is not just unique to buildings, it is being applied in other fields such as automotive manufacturing or aerospace engineering.

2.4.3 Accident Scene Reconstruction

Accident scene reconstruction is performed for in-depth analysis of the dynamics of a collision event [20]. On the accident scene, the experts need to collect data such as distance of the debris from the vehicles or tire marks. One of the standard practices is to take measurements manually, but this method is time consuming

and limited to few tens of measurements. Time is a big factor because when the traffic lanes are closed for examining the site, collateral accidents may occur. Furthermore, at the scene of the accident, the exact amount of data required for the analysis is not known and relevant details can be missed [21].

The emergence of 3D scanning techniques such as TLS have made the process of data collection safer and highly efficient, as it takes place without traffic interruptions. The process is faster, flexible and reduces the people required to gather data. In a standard setting, an investigator returns to the accident scene between 8 and 12 times [20]. However, with the utilization of 3D scanners, these visits could be made virtually through a reconstructed 3D digital model, which could also reveal crucial pieces of evidence not considered at the beginning of the investigation.

2.4.4 Historical Building Documentation

The deterioration of historical monuments along the passage of time is inevitable. Thus to preserve these buildings, restoration and documentation are needed. This is where 3D scanning techniques come into play: they prove as certainly convenient and efficient ways of documenting the heritage buildings as 3D digital models. The 3D models can be utilized by restoration companies as well as provide opportunity for archaeologists around the globe to study a heritage site virtually without even travelling to the actual location. Furthermore, 3D laser scanning is a no-contact technology which ensures that the monument does not get affected in the process. For documenting heritage sites, terrestrial laser scanners are used in conjunction with a camera. The combination results in a high-resolution detailed scan [4, 22].

2.4.5 Architecture and Construction

Many architectural projects involve renovation and reconstruction of old infrastructure and many times the “as-built” drawings of the structure are inaccurate or incomplete. So, if a project is planned according to an outdated drawing, it would result in the errors being “designed-in” along with the rest of the plan. These errors would be very difficult to spot until the latter half of the process, i.e.: the construction phase, at which it would be far more complicated and time consuming to resolve the problem. Consequently, TLS can be utilized to capture point clouds of the project site to make accurate and detailed 3D digital drawings of the infrastructure, which could be compared with the existing drawings in order to validate and correct them. Moreover, if the infrastructure in question is missing the “as-built” drawings from the start, then the scan information can be used as a template for creating new documentation as well as for planning the project [23].

The other situation in which TLS can be applied is construction validation, it is for scanning the on-going construction process to validate whether the work put in place is as planned. The scans could be taken on a regular basis and then aligned

with the original design to detect clashes. Subsequently, the issue can be resolved before it delays the entire construction process [23].

2.4.6 Urban Modeling

Obtaining the open-air map of the city is useful in many infrastructure fields. Generally, it shows the picture of the entire urban asset that city authorities will consider in order to plan future alterations of the city construction. MLS is the most popular tool for modeling the urban map. Its ability to scan the surrounding while moving is an advantage in terms of efficiency. A laser scanner is usually mounted on a automobile which is driven around the city or on a drone to get an aerial view. Surely, in an urban environment there are some areas where vehicle access is restricted and in that case the scanner is mounted on a backpack and the working principle is the same as in the vehicle case [24].

2.4.7 Underground Railroad Survey

At present, traditional surveying methods are being used in order to measure parameters concerning underground rail structures and compare them with initial measurement data of the given structure. In case of data discrepancy, consequences can be life-threatening, so a comprehensive analysis of the data is made by the city authorities. Performing this task manually could be also dangerous because of permanent train rides. Therefore, utilization of 3D scanning is becoming necessary. MLS system is suitable for performing the task because its mobility allows faster scanning of large areas. It is mounted on a special automatic slide rail car to scan the area in front of the vehicle [25].

Conclusion

The selection of an application field depends on the group's ability to develop, implement and test the technology required for the said application. Most applications, like accident scene reconstruction or historical building documentation, require expensive high-precision 3D scanning equipment and thus, within the available resources to the group, a proper implementation cannot be made for these applications. On the other hand, for architecture and construction, the scans need not be highly detailed and precise in terms of texture and colour for further procedures, contrary to virtual reality and game design. Thus, an integrated approach with a camera will have no applicable consequences. Also, the equipment available to the group would be adequate to obtain the required point cloud density for architectural analysis of 3D scanned surroundings. Therefore, any further analysis in this report would be based on the application of 3D scanning in architecture and construction.

2.5 Possible Solutions

In the following section, several possible solutions are listed and described for the chosen application: architecture and construction. It was already stated before that the main focus will be directed to the laser scanning methods and solutions. Additionally, it should also be noted that we will restrict ourselves to considering indoor scanning exclusively, since operating a system outside would entail dealing with various external agents, such as unfavorable atmospheric conditions and the large scale of the testing area.

2.5.1 MLS

Wheeled Vehicle

Wheel-based MLS entails having a laser scanning system integrated into the AGV (Automated Guided Vehicle) that will be able to move inside buildings and scan the surroundings in an efficient way. However, utilizing the vehicle would be inconvenient for several reasons. For example, the AGV may meet obstacles such as stairs and doors that will require a person to lift it and move it across the obstacle. Also, if the building has more than one floor, the vehicle will not be able to automatically move between storeys.

Handheld and Backpack

Another possible solution related to MLS scanning methods is simply a scanning system that can be handheld or mounted on a backpack. Arguably, these solutions are suitable for scanning any indoor area, irrespective of the size because the scanner would be able to create a 3D digital map of any place where the person carrying it may go. However, a mobile laser scanner involves implementing a method to keep track of the scanner's continuously changing location. Due to the fact that GPS cannot be used indoors, an algorithm based on the obtained data from different sensors should be used instead. Even if such algorithms have already been implemented (e.g: Simultaneous Localization and Mapping [26]), their complexity is beyond the knowledge base of the group. Additionally, scanning the surroundings with a handheld device entails the effort of the operator.

2.5.2 TLS

Unlike MLS, TLS does not have many variations on where it can be mounted. Since it is able to scan the area only motionlessly, there is no need to argue where it should be integrated - on a tripod, wall, table etc. In this case, the utilization of different scanners plays an important role in obtaining a 3D digital map for architectural purposes. Generally, LiDAR (acronym for Light Detection And Ranging)

is used for TLS in order to scan indoor sites, gather data points and send them for further processing. There are several LiDAR sensors that are suitable for this purpose, such as 2D and 3D LiDAR, which are capable of providing more than one point per reading. However, with regard to the resources available, these sensors are not a viable option due to their high cost. A second solution would be a 1D LiDAR sensor, which can provide the distance to a point in its direct line of sight. This solution may seem irrelevant to the project because the goal is to obtain a three-dimensional map. Nonetheless, it is possible to obtain a 3D point cloud from a 1D LiDAR sensor by rotating it using actuators, thus optimizing costs [27, 28].

Conclusion

After considering all the aforementioned possible solutions, the group has chosen to use TLS for scanning an indoor area. The solution to work with a 1D LiDAR sensor hereinafter is not the simplest because of its inability to scan 3D spaces while being motionless. However, working with a 1D LiDAR in the project offers the possibility of implementing actuators for the operation of the 3D scanner.

2.6 Stakeholder Analysis

One of the main factors when developing a new technology is identifying who might be interested or affected by it. If the technology does not have a solid base of people that find it useful, there is no point on investing any resources on it. For this reason, it is necessary to make a stakeholder analysis, distinguishing between entities who do not have a direct control over the technology (interested parties) and those whose decisions can affect its implementation (actors and the drivers of technology), as shown in Figure 2.2.

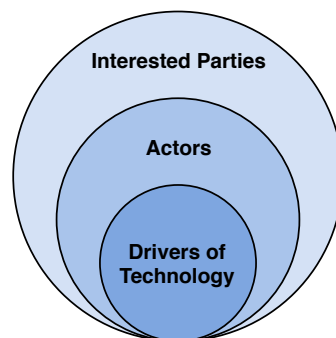


Figure 2.2: Stakeholder Diagram

2.6.1 Interested Parties

Although there is a variety of parties who might be interested in 3D scanning, the main ones, provided how the project is focused, are architects, interior designers and any construction-related professional [5] as well as the companies employing such people. The clients of these professionals can benefit from the implementation of this technology and therefore they can also be classified as interested parties.

2.6.2 Actors

As any electronics-based system, our technology needs companies that provide the necessary components and assemble the final product so it can be sold to the public in an optimal working condition. Thus, the decisions taken by hardware suppliers and manufacturing companies can drastically affect the characteristics and the availability of our solution. In addition, competitors that develop 3D scanners can also have an impact on the features of the device implemented in this project. Governments and standard agencies are important actors as well, since the technology may need to be slightly changed to adapt to a certain legislation in order to be able to sell the product in all the markets.

On a different level, it is inevitable to mention our supervisors and the university as important actors whose decisions could influence the project. Finally, this project, would not be possible without a team of developers. The group ED3-1-E18 have the final responsibility over the development and implementation of the 3D scanner.

2.6.3 Drivers of Technology

Although a single electronics manufacturing company cannot be considered a driver of technology, the manufacturing industry as a whole can be considered a technology carrier, since it has the power over the production of the components needed by the developed device. Another example of a driver of technology is represented by the investors that back up the 3D scanning industry financially.

2.7 Legislation and Standards

Even though no European or international legislation strictly concerning architectural 3D laser scanning has been found, some legal aspects, as well as international standards, that affect the design and use of 3D scanners, amongst other pieces of electrical equipment, can be identified.

Low Voltage Directive (LVD)

The scope of this EU regulation is to ensure the health and safety of users of electrical appliances placed on the Union market. The equipment that falls under its purview is the one that operates between the following voltage ranges: 50V - 1000V (for AC) and 75V - 1500V (for DC). All devices should be engineered according to the state of the art and their functionality should not damage the property where they are installed [29].

Waste Electrical & Electronic Equipment (WEEE) Directive

This piece of legislation aims to diminish the environmental impact of waste in the form of electrical devices, by facilitating their recycling. For this reason, it forbids the producers of electrical equipment to impede the reuse of WEEE by implementing specific features to their devices [30].

International Electrotechnical Commission (IEC) Standard 60825-1:2014

The 60825-1:2014 international standard concerns the laser products that operate at a wavelength of laser radiation which falls in the range 180nm - 1mm. These products can be either stand-alone or part of a more complex electrical, mechanical or optical system. The scope of the standard is to classify laser products by the potential radiation hazard they pose and to specify the protective attributes that minimize the risk of human injury. Emphasis is placed on adequate labelling of the laser products [31].

Conclusion

All these regulations must be respected in order to obtain a quality product that could be utilized in the EU.

2.8 Risk and Safety

The only risk that is worth being taken into consideration is laser radiation. Working with lasers should be a matter of great caution, for there is a risk of eye or skin damage being inflicted on operators. According to the IEC 60825-1 standard, the least hazardous laser class is Class 1; it is low-power and safe under all conditions of normal use, in the sense that there is no upper limit for time exposure. For other laser classes, protective eyewear is essential [31, 32].

Conclusion

To minimize the risk, a low-power laser should be used in the project, with Class 1 being recommended from a safety point of view.

2.9 Problem Statement

After analyzing the different applications of laser scanning in various contexts, it is time to make the initiating problem more specific so it is easier to find a concrete solution for it. Thus, the definitive problem is stated as:

Designing a stationary LiDAR-based 3D scanner to construct the digital model of a room

2.10 Project Specifications

This section will describe, according to the guidelines fleshed out throughout the problem analysis, the specifications and procedures that are necessary for the implementation of the practical phase of the project, i.e.: to build a 3D laser scanner that can scan the internal structure of a room.

Acquisition of Point Cloud Data

The output from a laser scanner is a point cloud data of the area being scanned. Accordingly, the 3D scanning apparatus designed by the project group should be capable of generating point cloud data of the infrastructure to be digitized.

LiDAR Sensor

LiDAR sensors are a crucial part of 3D laser scanners and a 1D LiDAR is the only viable option available. Accordingly, the group should be able to utilize a 1D LiDAR sensor in conjunction with other components for acquiring point cloud data.

Actuators

It is necessary to move the 1D LiDAR sensor to acquire the point cloud data from all its ambient infrastructure. Such motion can be achieved by using two actuators, one for 360° rotation of the sensor in the horizontal plane and the other for the 180° rotation in the vertical plane. Hence, the group should be able to implement the above-mentioned maneuvering through the use of appropriate actuators, so that the built device in question is fully capable of scanning a 3D space.

Battery

The laser scanner can be required to scan in situations where it is not possible to power it through the grid (power socket). Therefore, the power to the scanner should be supplied through a battery, which has adequate autonomy and is rechargeable. Battery autonomy is a crucial aspect as low battery conditions can derange the movement of motors which can affect the overall quality of scan.

Processing Scanner Data

The point cloud data captured through the scanner is of no use unless it has been exported to a appropriate software for further processing. The data obtained from the scanner is just a series of numbers indicating the position of a point, thus it first needs to be converted to a coordinate system. The acquired coordinates should be plotted by the software automatically to a medium that is comprehensible. If necessary, the software should also be able to remove the noise from the input data to improve the overall quality of the scan.

Microcontroller

As discussed in the previous points, a complete 3D laser scanning system would require actuators and a LiDAR sensor. Therefore a microcontroller (uC) is needed for controlling these input and output devices as well as for sending data to the computer.

Interface

The procedure to operate the 3D scanner should be as easy as possible which could be achieved by designing a user-friendly interface. Through the interface, the user should be able to start a scan and choose between different scanning parameters.

Operating Thresholds

The device should be programmed in such a way that the user can change the operating thresholds. The thresholds involve rotation speed, maximum and minimum vertical and horizontal angles of movement etc. The adjustment in these parameters would in turn change the output of the device.

Storage

The processed point cloud data from various scans of different locations should be stored, so that they can be accessed later.

2.11 Project Delimitations

Given the time allocated for the completion of the project is three months and a half, some clear boundaries have to be set before the development phase is started. This section is meant to indicate the aspects related to the final product that are intentionally left aside because they cannot be implemented in the specified time period.

System Appearance

The solution we aim to develop mainly concerns aspects related to the fields of electronics and computer engineering. Even though we desire our product to be aesthetically pleasing, little effort will be directed towards accomplishing an elaborate exterior design.

Data Visualization

Following the data acquisition process, a set of points will be plotted in the form of a point cloud. Apart from noise reduction, no further actions will be taken towards surface reconstruction, because of the well-known time constraints. A point cloud with a fairly high density conveys valuable information about the aspect and dimensions of a scanned space and should suffice to prove the functionality of the device.

Platform for Displaying Point Clouds

The control over the actuators and the sensors employed in the project is granted to a microcontroller, but the processing and visualization of the acquired data will be done using a computer. The motivation is that, unlike a computer, a microcontroller does not have an operating system and cannot run programs able to plot the required set of points.

Chapter 3

Methods and Materials

3.1 System Overview

In this chapter, the theoretical basis of the project will be discussed and the list of the utilized components will be laid out. For convenience, a crude overview of the practical part of the project should be provided. This overview will focus on augmenting the specifications discussed in Section 2.10 to describe the basic operating procedure of the developed laser scanning apparatus. The following steps elucidate the respective procedure:

1. To scan the surrounding area, the system is designed to maneuver the LiDAR sensor around its horizontal and vertical axis. A stepper motor rotates the sensor 360° in the horizontal plane and a servo motor rotates it in vertical plane within an angular range of 180° .
2. To procure the coordinates of a point, the system utilizes the spherical coordinate system. The indexes for the system corresponds to the distance given by LiDAR, rotation angle of the servo motor and rotation angle of the stepper motor respectively.
3. Once a coordinate is acquired, the readings of the sensor, as well as the angular data are transferred through serial communication from Arduino Uno to a laptop, then processed using MATLAB and stored in a file. The data is finally plotted in 3D space, thus constructing the point cloud of the scanned environment.

A graphical representation conveying the basic idea behind the practical part of the project is illustrated in Figure 3.1.

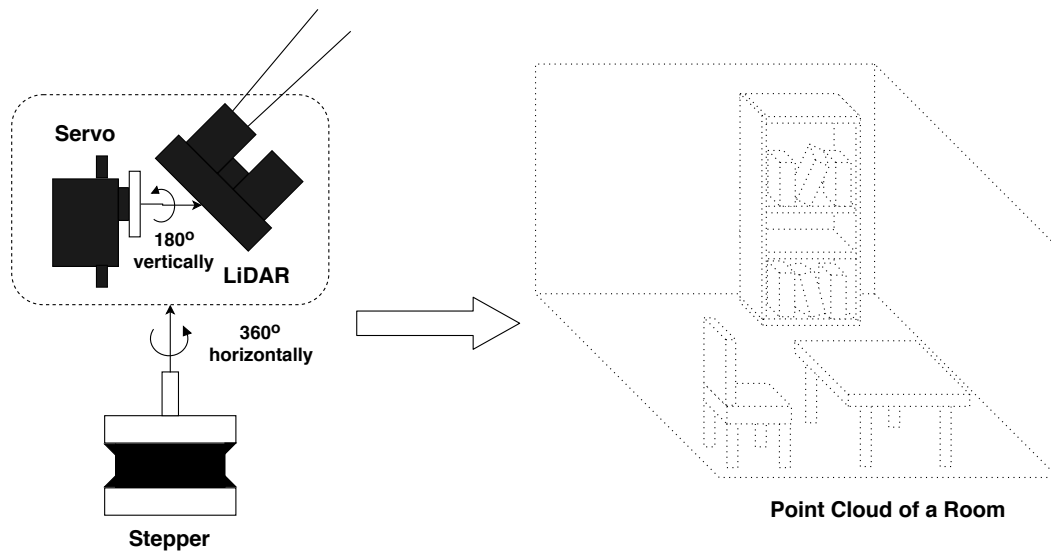


Figure 3.1: Diagram of General Idea

To show the relationship between the main pieces of hardware utilized, a block diagram was made, and it is shown in Figure 3.2. In the subsequent sections of the present chapter, all the individual components will be described in more detail.

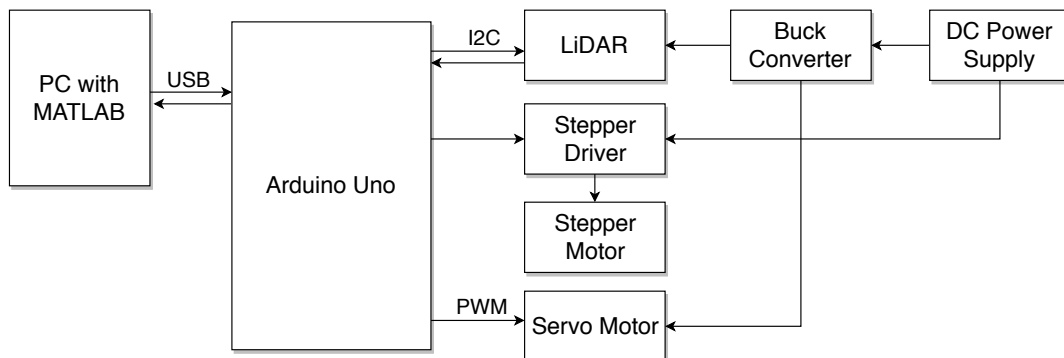


Figure 3.2: System Block Diagram

3.2 Arduino Uno

When it came to the choice of an appropriate microcontroller, the group opted for the use of an Arduino board, because of its appropriate specifications and prevalence in electronics projects. Moreover, the compatibility with many sensors, for which dedicated libraries have been developed (such as `Servo.h` and `LIDARLite.h`, for interfacing RC servo motors and LiDAR Lite v3/v3HP sensors, respectively), and the wide range of support available online corroborated to the hypothesis that

utilizing an Arduino would be a solid decision. A very popular and commonly used microcontroller in the Arduino family is Arduino Uno (Figure 3.3), which provides enough resources in terms of power and pin count to be at the core of the system developed in this project. The board operates at 5V and comes with an on-board voltage regulator that makes it possible to use a supply voltage in the range 7-12V. It has 20 GPIOs, out of which 6 have Pulse Width Modulation (PWM) output capabilities. The maximum current per I/O pin is 40mA and the upper limit for a 5V-GND line is 200mA [33].

At the core of Arduino Uno lies the ATmega328P, a CMOS 8-bit AVR RISC microcontroller with a clock frequency of 16MHz that uses the Harvard architecture, which means program and data are stored separately and transferred employing distinct buses. The program memory is represented by 32KB of Flash, while data is stored in 2KB of SRAM. Additionally, 1KB of EEPROM is also present on board. The device features 3 timers, namely Timer0 (8-bit), Timer1 (16-bit) and Timer2 (8-bit), which can be used for event management as well as for generating PWM outputs (on Arduino Uno, the `Servo.h` library uses Timer1 for this purpose). For serial communication, the interfaces available in the uC include UART (Universal Asynchronous Receiver Transceiver) and I2C (Inter-Integrated Circuit) [34].

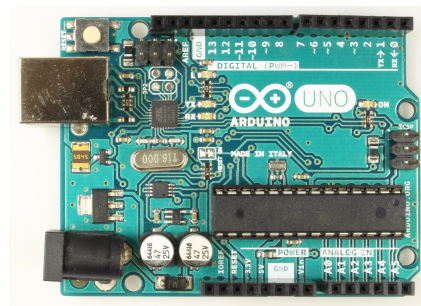


Figure 3.3: Arduino Uno Rev3 (ATmega328P)

3.3 LiDAR

The project requires a sensor to measure the distance to a point on the external surface of an object. There are various sensors that can determine this distance such as ultrasonic and infrared. When discussing these, the ultrasonic sensor provides a medium distance range up to 4m [35], but the spread of its transmitted beam makes it erroneous for our application, which requires the distance measurement of closely spaced points. The infrared sensor on the other hand provides a smaller beam that can capture much more distinct points over a region as compared to the ultrasonic sensor, but its range presents itself as a drawback, being up to 150cm [36], which is greatly deficient for our needs. All these constraints can be overcome

by a LiDAR sensor that brings higher range distance measurements along with a narrow beam which can acquire closely spaced points. As a result of this and the problem analysis, we decided to use LiDAR as the driving technology of the project, but the variety of different LiDAR sensors available on the market is wide, and there is always a compromise between price and features of the device. Thus, we were looking for something affordable but sophisticated enough to fulfill our goals.

LiDAR-Lite v3HP (shown in figure 3.4) seemed to be highly cost-efficient so we decided to implement it in our prototype. With its 5V operation voltage, low power consumption (about 0.5W) power and compelling specifications, it seemed the ideal solution for our purposes. Among its more interesting characteristics, we can highlight its 40m range that is suitable for our purpose of indoor mapping, its 5V/85mA operation (that makes it possible to power the device directly from a microcontroller for testing purposes), its connection versatility (it can be controlled by both PWM and I2C) and its 1kHz maximum frequency, allowing for a high point acquisition rate. Furthermore, it also boasts an accuracy of $\pm 2.5\text{cm}$ for distances greater than 2m and $\pm 5\text{cm}$ for less than 2m, providing a resolution of 1cm. In addition, the sensor is designated as a Class 1 laser which ensures the safety of the user and fulfills the regulations specified in Section 2.8.



Figure 3.4: LiDAR-Lite v3HP

3.3.1 Theory of Operation

LiDAR is a remote sensing method based on pulsed laser beams [37]. It allows the measuring of ranges with solid precision in a variety of environments and mediums (air, water, void). The LiDAR Lite v3HP works on the “time of flight” principle to calculate the distance. Accordingly, it measures the time taken by a transmitted signal to return to the device’s receiver, which is then translated into distance using the known speed of light (shown in Equation 3.1).

$$R = \frac{c \cdot t}{2} \quad (3.1)$$

where:

R = the distance between LiDAR and the target

c = speed of light in vacuum

t = time delay between transmission and reception

3.3.2 Limitations

Although the selected sensor has various interesting features, it has some obvious limitations too. The most concerning is its one dimensional scanning design. This entails a vital downside: the sensor can only deliver 1 point per measurement. Thus, the output we get after every cycle is the distance from the device to the obstacle instead of a set of points within a certain angle. This translates to LiDAR-Lite v3HP needing a higher time span for delivering the same amount of points compared to higher-end equivalents. Its frequency does not allow a very high point redundancy in a reasonable time span, meaning that the resulting point cloud is more susceptible to noises and misreadings. It must be stated that these issues are especially concerning in applications where the sensor is mounted on a moving object, such as a drone. Provided that in our system the sensor does not displace but only rotate around itself, these issues do not represent a major problem further than waiting more for getting the final point cloud.

On the other hand, some other limitations can represent a major inconvenience. Due to the physical principle of LiDAR (reflection of light in surfaces), specular surfaces such as mirrors that are smooth and reflect energy instead of dispersing it, may not be detected by the system (see Figure 3.6). When a diffuse surface like a wall is irradiated with a laser beam, it spreads the beam into lower energy beams in all directions (see Figure 3.5). A percentage of the reflected rays hits the sensor after a certain time. When the surface is specular, most of the energy is reflected away from the emitting point, so the sensor does not detect any returning beam and therefore the point is not saved [38].

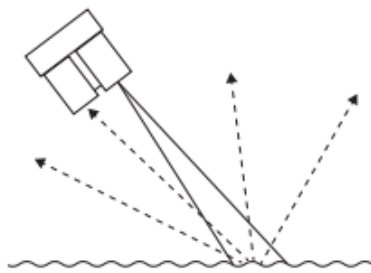


Figure 3.5: LiDAR Beam (Diffuse Surface) [38]

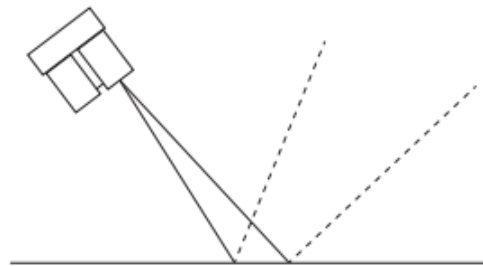


Figure 3.6: LiDAR Beam (Specular Surface) [38]

3.4 Stepper Motor

3.4.1 Motor Selection

The choice of using a specific type of DC motor to create a 360° rotation was taken after comparing the most popular options available on the market (brushed and brushless DC, RC servo, stepper), knowing that not all are suitable for a given application. The critical features were found to be the ability to complete full rotations, precise shaft positioning, robustness and low cost.

Brushed DC motors (BDC motors) incorporate a commutator ring and brushes that allow them to rotate. These components are very prone to damage caused by extended use, which makes the BDCs, in spite of their low price, a non-viable solution. Moreover, precise positioning requires the use of encoders to provide feedback and consequently the cost increases. Brushless DC motors (BLDC motors) do not rely on brushes, as their name suggests, which greatly prolongs their lifespan, but suffer from the same disadvantage of lack of position feedback [39]. RC servos (servo motors) are well-suited for precise positioning applications; notwithstanding, common servos are only able to rotate 180°, with the 360° versions encountered at higher price ranges. Usually, these motors do not send any position data back to the controller, but feedback versions can be acquired [40].

The last type of motor that will be discussed is the stepper motor. A stepper motor (alternatively, stepping motor or step motor) is an electronically commutated motor widely used in applications involving measurement and control because of its ability to adjust the position of its rotor by means of precise increments, that is, to move in very well-defined steps [41, 42]. Some characteristics of this class of actuators are: lack of brushes, full rotation, low price [43]. They offer open loop positioning, which means there is no position feedback given to the controller. To clarify, the position of the shaft is known at all times, even though there is no way to check the "actual" position [41, 42]. Despite this undesirable trait, accuracy should not pose an issue, provided the maximum torque rating of the motor is not exceeded, as steppers are designed to work without a feedback mechanism [44].

In the broadest classification of stepping motors, three types can be identified: variable reluctance (VR), permanent magnet (PM) and hybrid. They differ not only in construction and functionality but also in step angle and driving method. They all do have in common, however, the use of coils wound around the poles of the stator; the currents flowing through the coils create magnetic fields that control the rotation. Hybrid and PM motors can be further classified into unipolar and bipolar, depending on whether a half of a winding or a full winding is energized at a time, with the latter configuration providing more torque. In addition, bipolar motor drivers are widely available. Hybrids are extensively used in applications that call for high-resolution stepping, common step sizes varying between 0.9° and 3.6° [41, 42, 45]. Consequently, the selected motor was a bipolar hybrid stepper.

3.4.2 Working Principle

A stepper comprises a stator and a rotor. In a typical hybrid configuration, the rotor consists of a cylindrical permanent magnet covered at both the north and south poles by two misaligned toothed metallic end-caps. The stator has poles (not to be confused with the magnetic ones), usually 8, each with the same number of teeth. Two windings (also called phases) are located in the stator, covering 4 poles each. The phases are energized by running currents through them. This process generates magnetic fields that attract or repel the magnetized rotor caps. When current passes through one phase, with the other not being energized, the rotor moves one step, in the direction dictated by the current flow. Successive excitation of the windings in a specific sequence maintains the rotation. The most encountered step length is 1.8° , which translates to 200 steps per revolution [42, 46].

3.4.3 NEMA 17 Stepper Motor SM-42BYG011-25

The exact motor model utilized in the project is NEMA 17 SM-42BYG011-25, a 200-step 2-phased bipolar hybrid stepper motor, rated with 12V and 330mA per winding. The resolution is 1.8° . A picture of the component is shown in Figure 3.7.

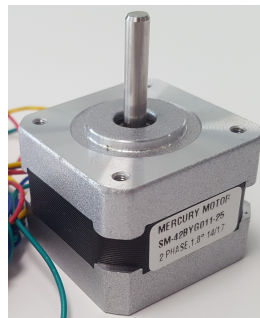


Figure 3.7: NEMA 17 Stepper Motor SM-42BYG011-25

3.5 Stepper Motor Controller

3.5.1 Motivation

A microcontroller cannot supply the voltage and current needed by a step motor and for that reason a driver that can provide the necessary power has to be employed. Upon receiving instructions from the uC, this device translates them to the required excitation sequence that turns the rotor [47].

3.5.2 Basic Control Circuit

Bipolar stepping motors, either PM or hybrid, are driven using a circuit that includes two identical H-bridges, because the current flow through each winding must be controlled independently [45]. Such a circuit is shown in Figure 3.8.

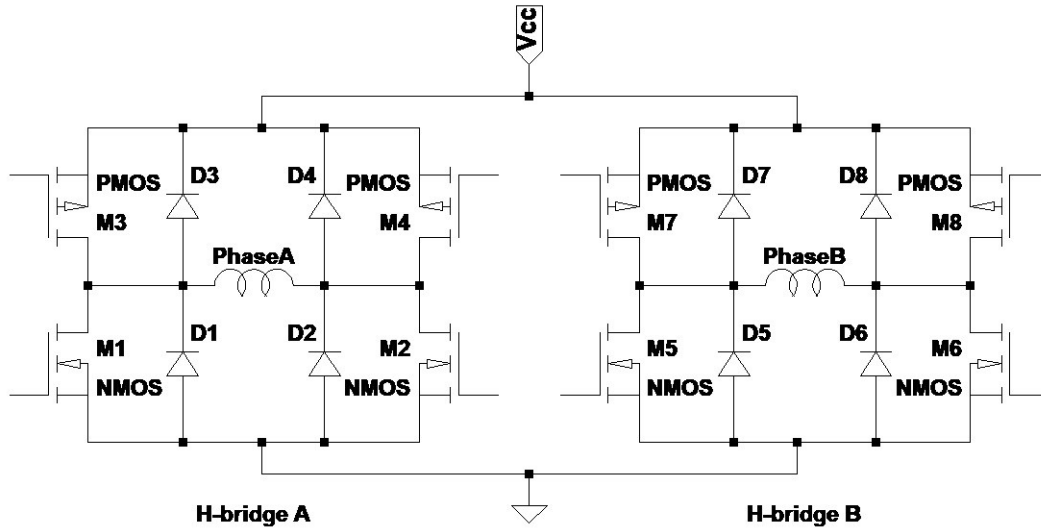


Figure 3.8: Bipolar Stepper Dual H-Bridge Drive Circuit

3.5.3 Excitation Modes

Energizing the motor windings in specific sequences controls the circular motion of the step motor. These are sometimes referred to as modes of operation or excitation modes. In an ideal scenario, step motors are driven with currents in the form of sinusoidal wave approximations: one phase using a sine signal, and the other using a cosine signal, such that the phase difference between the two currents is 90° . An excitation sequence can cause the rotor to move several steps and its duration is determined by the period of the current wave forms applied to the windings; one cycle of the current signal is equivalent to one excitation sequence [44, 48].

In the following discussion, it is helpful to think of the stator of a two-phased bipolar stepper as two distinct windings, a and b . Let the corresponding currents be labeled I_a and I_b . Additionally, let I_{\max} be the maximum current (specified by the manufacturer) that can flow through each phase of a given step motor. Any excitation sequence can be illustrated with the aid of graphs, in which I_a/I_{\max} and I_b/I_{\max} are plotted against time.

The most straightforward excitation mode is called full stepping and it implies that the rotor moves with its default step angle. In this case, a four-step sequence is utilized and thus each mechanical step corresponds to 90 electrical degrees [44,

48, 49]. Figure 3.9 shows two cycles for each of the current wave forms used to drive the motor in the described fashion. The chosen period is 2π seconds.

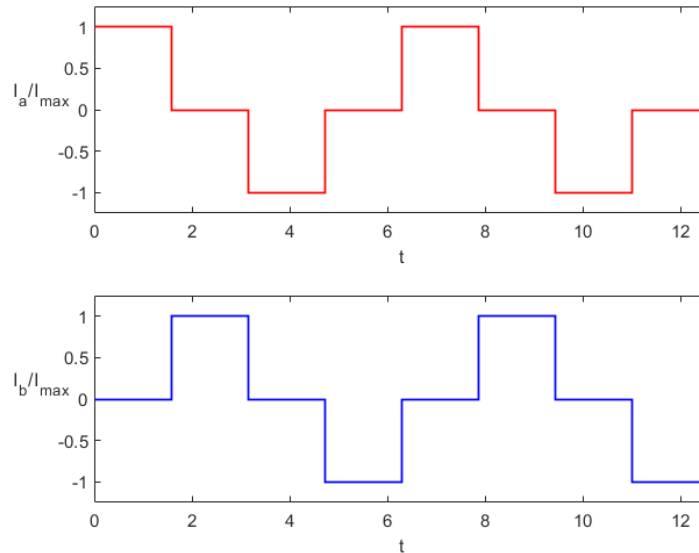


Figure 3.9: Full Stepping Current Wave Forms

3.5.4 Microstepping

The full stepping technique involves an important drawback. The movements of the shaft are spasmodic, especially when the angular speed is low, which creates vibrations and noise in the system. If this issue is to be avoided, the more advanced stepping algorithm referred to as "microstepping" must be used. This mode of operation increases the stepping resolution of the motor by dividing each original full step in many microsteps while minimizing the vibrations. The system performance is thus increased because the rotation of the motor is smoother and more precise [41, 48, 49].

Depending on the degree to which the resolution is increased, the number of steps in the excitation sequence grows as well, while the resemblance between the current wave forms through the windings and a sinusoidal signal becomes more conspicuous. Figure 3.10 illustrates two cycles (the period is 2π seconds) of the hypothetical wave forms for 1/8 microstepping. A mechanical microstep corresponds to 11.25 electrical degrees, so during an individual cycle the rotor moves 32 microsteps.

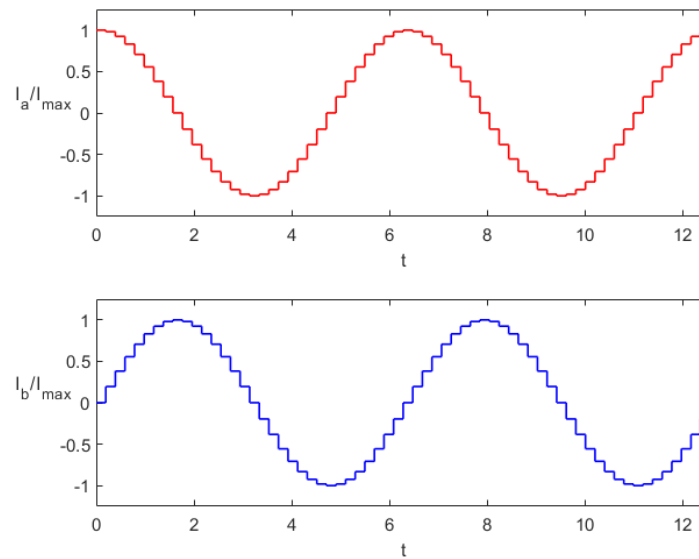


Figure 3.10: Microstepping Current Wave Forms

3.5.5 EasyDriver

The motor controller chosen for driving the bipolar stepper was EasyDriver v4.5 (Figure 3.11). It is an inexpensive driver that works with power supplies between 6V and 30V (it has an on-board 5V regulator) and can handle currents in the range of 150mA/phase - 700mA/phase, which means that it is compatible with the NEMA 17 SM-42BYG011-25 stepper. Apart from full and half stepping, it allows two microstepping modes that increase the stepping resolution by a factor of 4 and 8, respectively. Hence, with the selected 200-step motor, one can obtain either 400, 800 or 1600 steps per revolution. In terms of angular positioning, the corresponding step lengths are 0.9° , 0.45° or 0.225° . All these features, along with the small size, made EasyDriver a solid motor driver option.

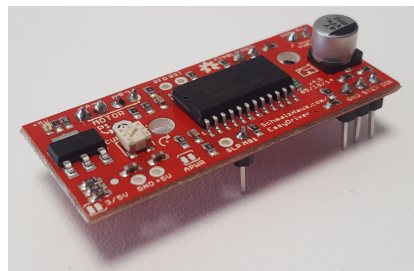


Figure 3.11: EasyDriver Bipolar Stepper Motor Driver

A bipolar stepper motor must be connected to the pins A+, A-, B+ and B- of the driver, which are the outputs of the two H-bridges, where each letter is associated with an inner motor coil. The upper limit for the peak current provided to the motor can be varied using a potentiometer. The rotation is controlled by sending a square wave to the STEP pin: the motor steps at every low-to-high (0-to-5V) transition. The spinning direction is dictated by the value written to the DIR pin. To set the length of each step, the pins MS1 and MS2 need to be configured. There are four possible configurations, shown in Table 3.1 [50, 51].

Table 3.1: EasyDriver Microstepping Configurations

MS2	MS1	Resolution	Step Angle (degrees)
0	0	Full Step	1.8
0	1	Half Step	0.9
1	0	Quarter Step	0.45
1	1	Eighth Step	0.225

3.6 Servo Motor

3.6.1 Motor Selection

In the previous section, we have discussed about the stepper motor, utilized for the 360° rotation in the horizontal plane. Similarly, a suitable actuator is necessary, that is responsible for the movement of the LiDAR sensor in the vertical plane. All the motors considered in Section 3.4, such as, brushed DC, brushless DC, stepper, and RC servo, can also be considered as possible choices for vertical plane movement. The brushed and brushless DC motors can be dismissed based on the same reasoning as in Section 3.4.

In order to get the accurate coordinates of the points on the scanned surface, the motor's movement needs to be precise, so that the scanned angle is the same as programmed. The stepper motors have a high precision movement, but they are designed for 360° rotation whereas the required angular coverage for the scanner, in the vertical plane, is 180° . Although the stepper motor can be programmed to move within the required range, it obligates the use of an encoder that keeps track of its position. The encoder's utility arises here due to the stepper not having pre-determined positions to which it can be moved. Accordingly, without an encoder it is not possible to rotate the stepper shaft to a specific position at the beginning of the scanning process, which is a key requisite for a motor needed for this purpose. On the other hand, servo motors are also capable of moving with high precision and generally have an angular range of 180° . They have known fixed positions for rotation which eliminates the requirement of an encoder. Furthermore, unlike

stepper motors, an RC servo motor does not necessitate the use of a motor driver. Thus, financially, simply based on components, a servo motor is much economical as compared to a stepper motor. Considering all these aspects, a 180° RC servo motor presents itself as the most suitable choice.

3.6.2 Internal Structure of a Servo Motor

A servo motor can position itself very accurately at any given input angle within its range of movement. It is a closed-loop servomechanism that utilizes position feedback to control the motion, speed, and final position of its shaft. The RC servo's anatomy incorporates components such as control circuit, feedback potentiometer, DC motor, and drive shaft. The control circuit is responsible for receiving the input signal and translating it into motor revolution in a way that the drive shaft reaches the desired position. The other key element of the closed loop mechanism in a servo motor is the feedback potentiometer, which is rotated along with the drive shaft through a gearbox. Each of its distinct resistance corresponds to a unique position of the shaft, which is sent as feedback to the control circuit. All the above-mentioned components are essential to the working of a servo motor but additional components such as encoders can be added based on the application [52, 53].

3.6.3 Working Principle

A pivotal element of the mechanism in servo motors is the comparison between the desired and actual position of the shaft, which dictates the direction of rotation for the internal DC motor. Firstly, the desired position is supplied as input, in the form of a PWM signal that is converted into a reference voltage by the controller circuit, such that, each reference voltage corresponds to a distinctive position of the drive shaft. Secondly, the controller reads the voltage of the feedback potentiometer representing the actual position. Now, the comparison between reference and potentiometer voltage yields an error signal according to the variation among them. This error signal serves as a control input to the H-bridge controlling the direction of rotation for the DC motor. The rotation is done in an effort to reduce the error. When the shaft reaches the desired position, the error becomes zero, but the process still continues if there is an input signal and frequently (40-50 times every second in RC servos) checks the change in the drive shaft position due to any external influence, ameliorating it if necessary [52, 54]. The servomechanism is presented in Figure 3.12.

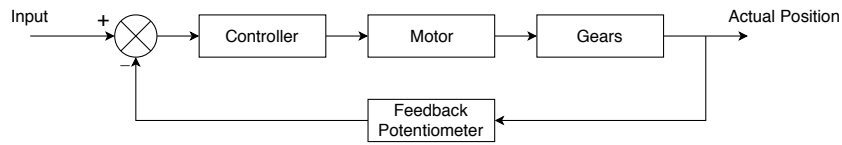


Figure 3.12: Servo Control Loop

3.6.4 Servo Control

As mentioned in the previous paragraph, the input to an RC servo motor is a PWM signal. Figure 3.13 shows the parameters that characterizes this signal: amplitude, pulse width, and period. Furthermore, it illustrates the standard time versus voltage chart for an analog RC servo motor. It can be comprehended from the figure that the pulse width of the PWM signal controls the position of the drive shaft, which changes according to pulse width variation [55, 54].

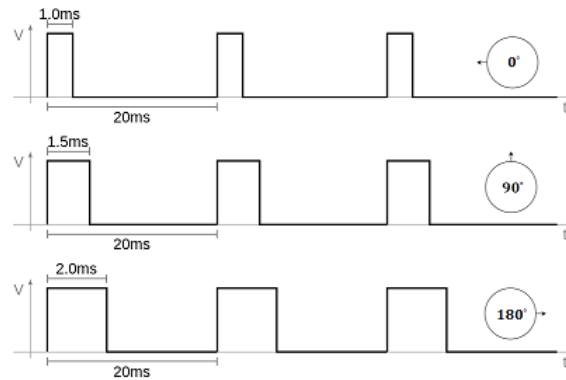


Figure 3.13: Servo Control Signal

3.6.5 Metal Gear Servo

The exact model utilized in the project is a metal gear analog servo motor with feedback, i.e., it can send the servo position as analog voltage to the microcontroller. The model is rated for 3-6V, and 2.5kg-cm torque. A picture of the component is shown in Figure 3.14.



Figure 3.14: 14g Servo Motor with Metallic Gears

3.7 Spherical Coordinates

The spherical coordinates represent the coordinate system in which a point in three-dimensional space can be described using three distinct values: a distance and two angles. Figure 3.15 illustrates the spherical coordinates of an arbitrary point P in space.

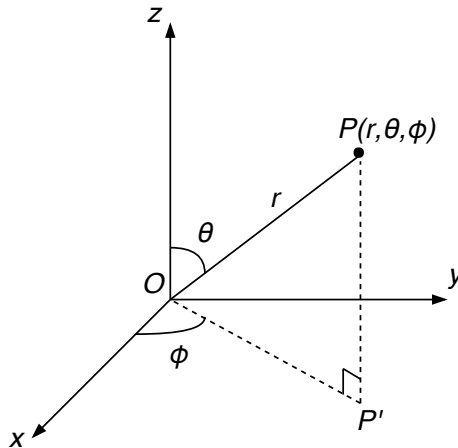


Figure 3.15: Spherical Coordinates of a Point P

The first coordinate is the distance $r = |OP|$ from the origin to the respective point and it is referred to as radius, because it constitutes the radius of the spherical surface on which the point is situated. The following coordinate is the azimuth angle, denoted by ϕ , which is the angle between the positive x -axis and the segment $|OP'|$, which unites the origin with the projection of the point P onto the xy -plane, P' . The third and last coordinate is represented by the inclination or polar angle, θ , and represents the angle measured between the positive z -axis and the line connecting the origin and the given point P . It is customary for the angular coordinates to be expressed in radians. To make sure all the points have a unique representation in the spherical coordinate system, the following restriction should be applied: $r \geq 0$, $\phi \in [0, 2\pi)$ and $\theta \in [0, \pi]$ [56, 57].

Spherical coordinates can be found using the chosen components for the project: two motors and one sensor. However, in order to not only get the coordinates of the points, but to also plot them to form a point cloud, a software tool is required. For that purpose, working with rectangular coordinates is more convenient.

The rectangular or Cartesian coordinates, which are the standard way of representing the points in three-dimensional space, are the projections of the vector \vec{OP} onto the x , y and z -axis, respectively. In other words, they can be found by sending straight lines from the point P perpendicularly onto the z -axis, and from the projection P' onto the x and y -axis. The distances between the origin and the

points of intersection between the axes and perpendicular lines yield the Cartesian coordinates. By using basic trigonometry, it is easy to convert from spherical coordinates to rectangular coordinates, as shown in Equations 3.2-3.4 [56, 57]:

$$x = r \cdot \sin(\theta) \cdot \cos(\phi) \quad (3.2)$$

$$y = r \cdot \sin(\theta) \cdot \sin(\phi) \quad (3.3)$$

$$z = r \cdot \cos(\theta) \quad (3.4)$$

3.8 Power Estimation

After all the necessary hardware components have been discussed, it is important to have a clear picture of the total power consumption of the scanner. Maximizing the usage of the external power source for the entire system is desired and therefore, the total power consumption should be known while choosing a suitable battery. For that reason, a theoretical power estimation of the 3D scanner was made, based on the technical specifications of the utilized hardware components.

The information about voltage and current consumption of the hardware is given in their datasheets. Components such as the stepper motor and the servo motor are considered notable power consumers. It should be noted that the datasheet of the utilized servo was not available and therefore the specifications regarding the voltage and current were taken from the datasheet of a similar analog RC servo motor [58]. The stepper motor driver and LiDAR Lite v3HP are included in the calculations as well, even though their power requirements are much lower compared to the actuators. Arduino Uno is left aside, since it would be powered by a laptop via USB. Table 3.2 shows the components of the 3D scanner and their voltage and current consumption, including the theoretically calculated power consumption.

Table 3.2: Power Estimation

Component	Voltage (V)	Current (mA)	Power (W)
LiDAR	5	85	0.425
Stepper Motor Driver	5	70	0.35
Stepper Motor	12	470	5.64
Servo Motor	5	700	3.5
Total			9.915

It should be stated that the values of voltages and currents shown in the table above correspond to maximum values in order to consider the worst case scenario in terms of power consumption. Therefore, the battery should be chosen accordingly. After calculation of the power for each component and then summing the individual contributions, a total of 9.915W was obtained as power estimation for

the 3D scanner. Even though Table 3.2 shows 12V supply for the stepper motor, it is not the maximum amount the given motor can handle. A stepper can run at a voltage higher than the rated one, which will cause the current in the windings to rise faster, boosting the overall performance of the actuator [41]. In this project however, where the torque requirements are not high, using the rated voltage will suffice. In the case of the servo motor, the current specified in the table is the stall current.

3.9 Battery

3.9.1 Battery Selection

In electric systems designed to operate in various environments, batteries are essential to provide power to the device. Given the outcome of the preceding estimation, a choice had to be made in terms of a DC supply that could guarantee the continuous working of the scanner for a reasonable amount of time. Judging by the data in Table 3.2, at least a 12-V battery is demanded by the stepper motor. The smaller voltage (5V) required by the rest of the components can be obtained by stepping down the battery voltage to the required value with a voltage regulator. Using the basic relation between power, voltage and current, and assuming no losses, the current the battery should supply is:

$$i = \frac{p}{v} = \frac{9.915}{12} = 826\text{mA}$$

If the working time for the scanner is assumed to be 2 hours, a suitable DC source should be able to constantly supply 826mA during this period, that is, it should have a capacity of 1652mAh. A further benefit will undoubtedly be added to the system if the battery is rechargeable.

3.9.2 UL 2.4-12 Lead Acid Battery

The battery chosen for this project was the Lead Acid battery UL 2.4-12, which is rated 12V and has a capacity of 2400mAh. It provides the required voltage and its capacity exceeds the necessary value, which is, of course, an advantage. A simple calculation shows that employing this battery would, theoretically, keep the system running for almost 3 hours. The additional features of being rechargeable and inexpensive contributed to the decision of selecting this power supply [59].

3.10 Buck Converter

3.10.1 Selection

Two possibilities for decreasing the battery voltage to 5V were identified: linear regulators and step-down (buck) converters. The former option is simple, low-cost, yet power-inefficient when the difference between the input and output voltages is large, as in the present case. The latter solution, if more expensive and complex, guarantees power losses smaller than 10% and minimizes the amount of heat produced by the conversion [60]. As part of this project, to prolong the battery life as much as possible, a buck converter was selected.

3.10.2 Working Principle

The circuit of the simplest buck converter can be found in Figure 3.16. Conceptually, the working of this regulator can be easily explained. The input voltage is connected and disconnected alternatively from the rest of the circuit by the transistor controlled with a square wave signal. When the transistor is on, the diode is reverse biased and acts as an open circuit. The inductor starts charging and creates a voltage that opposes the increasing current, therefore creating a voltage drop. Additionally, during this period, the capacitor accumulates charge in its plates. After the transistor turns off, the input voltage source is removed from the circuit. The inductor cannot handle the instantaneous change in current and therefore it tries to maintain the flow by adjusting its voltage accordingly. The potential at the left terminal of the coil becomes negative, which forward biases the diode. Both the capacitor and inductor discharge during this time. By controlling the amount of time the input is connected to the circuit, i.e., the time the transistor is on, a desired output voltage, smaller than the input, can be obtained [61, 62].

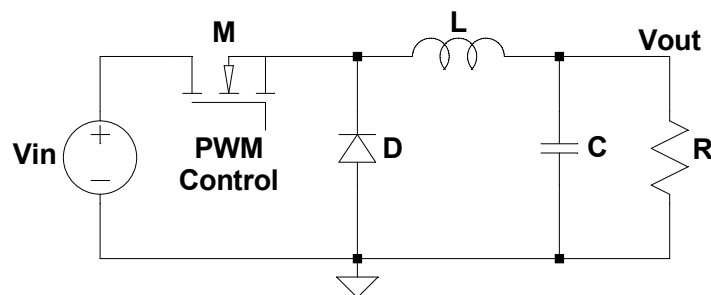


Figure 3.16: Buck Converter Basic Circuit

It can be deduced that, for a given "on" time of the transistor, a series of square wave pulses is present at the left terminal of the coil. The LC configuration shown

in the schematic filters this signal, producing a steady DC output across the resistive load [61].

3.10.3 XL4015 LM2596 Step-Down Converter

The selected switching regulator was XL4015 LM2596 (Figure 3.17). The input voltage range is 8V-36V and it can produce potential differences between 1.25V and 32V at the output. It is capable of handling loads consuming up to 5A.

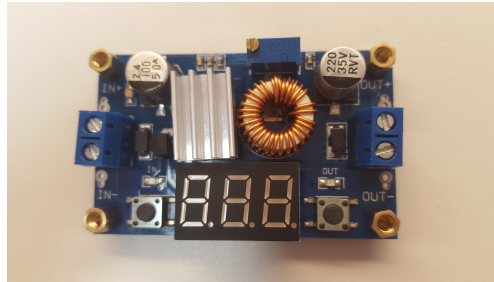


Figure 3.17: XL4015 LM2596 Buck Converter

The main advantage of the module is that it generally provides high efficiency. For an input voltage of 12V, an output voltage of 5V and a load current of around 826mA, which are the requirements imposed by the hardware employed in the project, the efficiency curve of the converter indicates around 91%. Thus, only 9% of the power supplied by the battery is dissipated as heat [63]. Because of that, the decrease in the battery discharge time is not significant and can, therefore, be ignored. Consequently, this step-down converter module represented a good choice for regulating the battery voltage to the desired output.

3.11 MATLAB

MATLAB is a programming language mainly designed for processing data represented by numeric arrays. It is extensively used in science and engineering due to its ability to unite complex mathematical calculation and data visualization, while still providing a multitude of easy-to-use functions, with the purpose of producing solutions to general and specific problems [64]. Because of its numerous features and highly developed help system, MATLAB was chosen to be a part of this project. Moreover, from an engineering point of view, MATLAB is a very useful tool that is beneficial when working in the industry, and it is essential from a learning perspective.

MATLAB brings various toolboxes which greatly extend the power of the software when tackling certain domain-specific tasks. In the present project, the Computer Vision Toolbox was used, which facilitated 3D point cloud processing.

3.12 Point Cloud Filtering

All the data obtained by the LiDAR Lite v3HP must be processed somehow: although it is reasonably accurate, we cannot suppose that all the resulted points are within an acceptable error range. A variety of obstacles and structures can make the sensor deliver readings that differ completely from the ones in its surrounding (electrical noise, glass, people passing in front of the sensor momentarily), making a potential surface reconstruction fail or converge to undesired values that might make the final result look unrecognizable. Thus, we need a method that, based on a large number of points, discards those that do not “fit”. Although it seems intuitive to make an estimation based on a linear least squares regression (LLSR) [65], as we do in two-dimensional data sets, this method fails when we try to interpolate data with significant errors, even if the number of “mistaken” points is small compared to the total sample. We can easily observe this with an example in Figure 3.18:

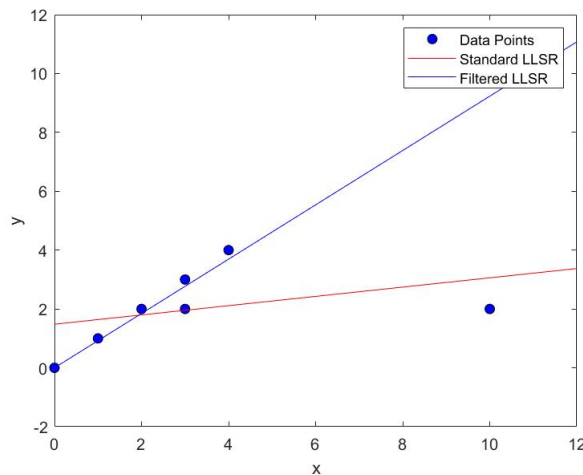


Figure 3.18: Filtered/Unfiltered LLSR Demonstration

In Figure 3.18 we can see how, even if most of the points comply approximately with a linear model (represented in blue), the method fails (as shown by the red line) with the addition of one point located far away from the “filtered” data set line. Thus, we need a model that takes into account a number of points that fit a linear model within a certain error range, discarding the ones out of that range.

One of the most popular ways of doing this properly is the algorithm called Random Sample Consensus (RANSAC). Unlike raw LLSR, RANSAC does not use all the available data from the very beginning. Instead, it takes the least amount of randomly chosen points (that we can call a) possible for fitting certain conjunction of points to any structure in the 2D plane ($a = 2$, in the case of a line) and checks

how many of other points in the data set can be estimated by the obtained structure within a certain error margin. If the number of points that can be fitted by the structure is larger than n (defining n as the number of points that are estimated to be within a reasonable error range, this number is calculated based on the expected precision of the system and the total number of points available) the estimation is considered valid and the “valid” points are fitted by a smoothing algorithm such as LLSR. If the number of points that can be fitted is smaller than n , the model is considered invalid, and the algorithm picks other set of a points randomly and starts the process again. If no estimation can fit a number of points higher than n , the estimation is made based on the model that can fit as many points as possible or the data set is considered invalid as a whole [66].

By extrapolating this method to a three-dimensional environment, we can efficiently remove the outlying points from a data set in 3D. However, the three-dimensional application of this method is much more complex and goes beyond the purpose of this project. The MATLAB function `pcdenoise` based on MLESAC (a variant of RANSAC) [67] is a solid example of its implementation. Figures 3.19 and 3.20 show a cube with 10000 added points as uniformly distributed noise before and after applying `pcdenoise` to it, respectively. We can clearly see how, thanks to `pcdenoise`, the outliers are completely removed from the figure, thus filtering the given point cloud.

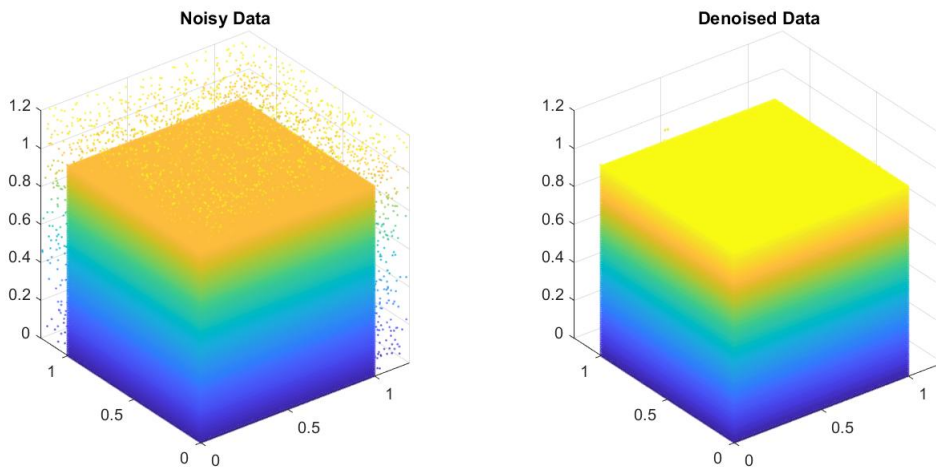


Figure 3.19: Cube with Uniformly Distributed Noise **Figure 3.20:** Cube After Applying `pcdenoise`

Chapter 4

Implementation

4.1 Data Transfer and Plotting Test

Processing the data gathered by LiDAR, along with the one extracted from the shaft positioning of the two motors, is crucial to produce the desired meaningful output. As this is performed on a computer that runs MATLAB, and the acquisition is done with a microcontroller, data should be transferred between the two. There is an Arduino function, `print()`, which sends serial data from the board to the serial monitor in the Arduino IDE with a set rate using UART and USB. To manipulate this data, a script in MATLAB¹ is needed to read it, store it, modify it, save it for further reference and create a visual output, i.e., the point cloud representing the objects whose volumetric data was previously acquired.

To illustrate the implementation of these features on the simplest level, a small test was conducted using only the Arduino Uno and a laptop, without employing any sensors. The relevancy of this test is indubitable, since the MATLAB code is independent on the way the data is acquired, meaning that the processing method remains unchanged, whether a sensor gathers the data or hypothetical values are used instead. Processing the data describing a sphere was found to be a good choice, as it requires a fairly large amount of data points, while being still easy to generate. To create such a surface in spherical coordinates, the radius has to be a positive constant, the azimuth angle should lie inside the interval $[0, 2\pi)$ and the polar angle should be restrained to the interval $[0, \pi]$.

The utilized Arduino code, in this case, provides the spherical coordinates of the points making up the hypothetical sphere of radius $r = 6$ and prints them on the serial monitor. In this simple program, incrementing each angle one degree

¹**N.B.** All the information regarding the MATLAB functions used in this project was taken from the MathWorks web site. The reference is given here and it is assumed the reader is aware that, whenever MATLAB programming is discussed henceforth in the present report, this source was utilized [68].

at a time using nested `for()` loops produced a 65160-by-3 matrix, accounting for 65160 points. A character (`*`) is printed in the last line to mark the end of the data stream that needs to be read.

On the receiving end, a MATLAB script establishes the connection to the uC using a serial port object. The serial port to which the board is connected, the baud rate and the terminator character (which must coincide with the parameters defined for Arduino) need to be specified. Once the connection is made, the read operation is carried out using the `fscanf()` function, which reads data from the device and takes as arguments the serial object, the format the data will be converted to and the dimensions of the matrix where the values will be stored. Data is read in column order, and thus, to keep values corresponding to different parameters separated, a 3-by-65160 matrix should be used for storage. Plotting the values requires a n-by-3 format however, so the matrix needs to be transposed after the serial communication is interrupted.

```

1 % Connect Matlab to Arduino using a serial port object
2 arduino_port = serial('COM12', 'BaudRate', 115200, 'Terminator', '*');
3 fopen(arduino_port); % Connect to Arduino
4 data = fscanf(arduino_port, '%lf', [3, 65160]); % Read data into a matrix
5 fclose(arduino_port); % Disconnect from Arduino
6 % Transpose data matrix
7 data = data.';

```

After the spherical coordinates are extracted from the data matrix, the coordinate conversion takes place, using the formulas given in Section 3.7. When the Cartesian coordinates are obtained, the individual vectors are concatenated to form yet another matrix, which is saved to a MAT-file, using the `save()` command, so that it can be later loaded in the workspace.

```

1 % Extract spherical coordinates from matrix
2 azimuth = data(:,1);
3 inclination = data(:,2);
4 radius = data(:,3);
5 % Convert to Cartesian
6 x = radius .* sin(inclination) .* cos(azimuth);
7 y = radius .* sin(inclination) .* sin(azimuth);
8 z = radius .* cos(inclination);
9 xyz = [x y z];
10 save('test_scan.mat', 'xyz'); % Save matrix to a file

```

The data can be plotted easily using commands from the Computer Vision Toolbox. The matrix containing the (x, y, z) -representation of the data points is stored in a point cloud object, which is then passed to the `pcshow()` function that creates the actual three-dimensional model.

```
1 p_cloud = pointCloud(xyz);           % Store data in a point cloud object
2 pcloudshow(p_cloud);                 % Display point cloud
```

The hypothetical sphere with radius $r = 6$ is shown in Figure 4.1.

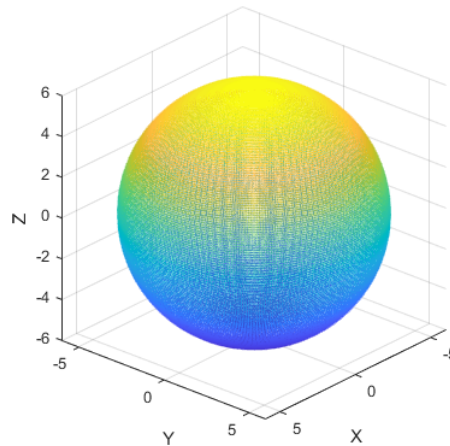


Figure 4.1: Point Cloud of the Test Sphere

Conclusion

Even though the described test illustrated the simplest procedure of data transfer and plotting using only the Arduino and a laptop, it laid the foundations for the MATLAB script required to process the data that would be provided by the motors and LiDAR.

4.2 LiDAR Experiment

Provided that all the point cloud data relies heavily on the distances measured by the LiDAR Lite v3HP, it was vital to perform different tests in order to assess its precision and behaviour in different situations. During all the experiments, the same setup was used: the LiDAR was connected to the Arduino using I2C, the uC was powered by a laptop via USB and the power to the sensor was provided by the 12V-battery, whose voltage has been stepped down by the buck converter. A $680\mu\text{F}$ capacitor was connected between the 5V and GND and two $4.7\text{k}\Omega$ resistors were placed between the 5V and SDA and SCL, respectively, as indicated in the datasheet. The code written to get the distance remained unchanged as well, with the function `Lidar.distance()`, defined in the `LIDARLite.h` Arduino library, being

invariably utilized. For the I2C communication, the `wire.h` library was included in the program.

Initially, the sensor was tested for a window. When placed perpendicular to the surface, the sensor was providing the distance corresponding to the object behind the glass. However, when the angle of incidence between the beam and the normal to the window was even slightly modified, the sensor read "0 cm". Then, the working of the device when being pointed towards a mirror was observed. In this case, regardless of the angle of incidence, the readings were absurd.

Three additional experiments were performed for diffuse reflective surfaces. LiDAR was placed in a fixed position with respect to a solid, opaque object (a matte wall) lying directly in front of it, i.e.: zero angle of incidence. The distance between sensor and obstacle was varied: 20, 100 and 350 cm respectively. These distances were chosen in order to allow accurate manual measurements with a measuring tape. The sensor was enabled for 1 minute, and the readings were printed on the Arduino serial monitor with a data rate of 115200 bits/s.

In the first experiment (Figure 4.2), the number of measurements was 15025, so the frequency of the sensor was calculated to be around 250Hz. The average distance measured was 15.6cm.

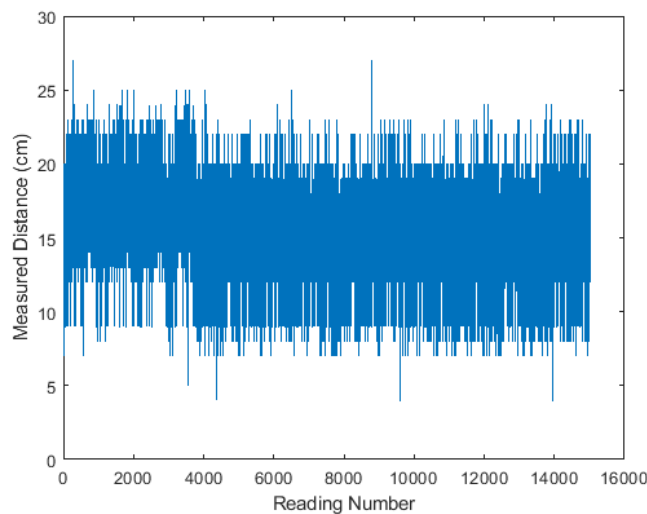


Figure 4.2: Results of LiDAR Test for 20 cm

In the second case, when the distance was increased to 100cm (Figure 4.3), the number of measurements also increased to 15513, resulting in a frequency of 258Hz. The average distance was calculated as being 100.8cm.

For the third experiment, the distance was 350cm and the results are shown in Figure 4.4. The number of readings was 15503, which translated to frequency of 258Hz. The mean value of the readings was 350.1cm.

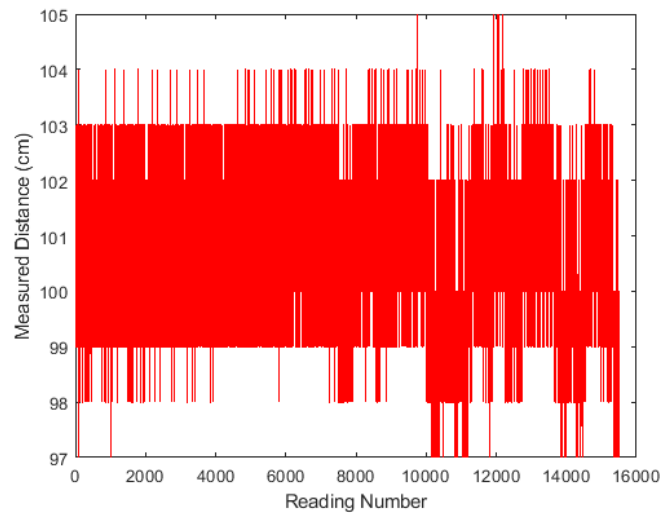


Figure 4.3: Results of LiDAR Test for 100 cm

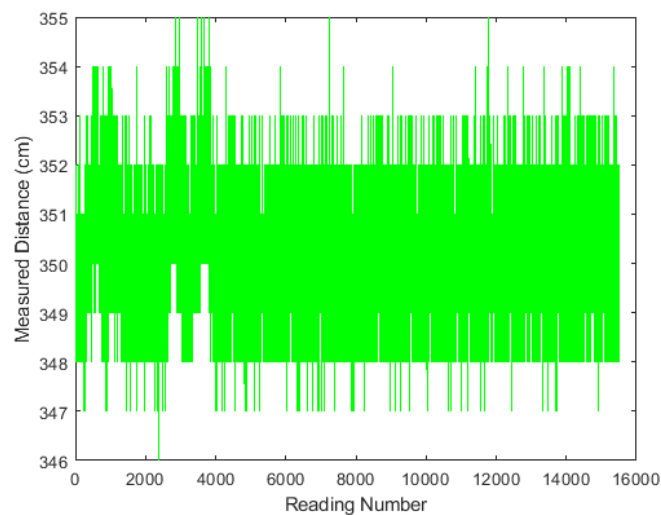


Figure 4.4: Results of LiDAR Test for 350 cm

The data obtained led to some interesting observations. Firstly, the amount of points obtained in the different experiments is not consistent. Although it can be considered equal in the last 2 experiments (circa 15500), the first one shows almost 500 measurements less. It must also be pointed out that although 500 points might seem like a large amount, it represents about 3% of the total point data, and that is why the frequency decrease, in relative terms, is not especially high.

If we look at the data itself, although all the elements of the setup were kept

completely steady, there is a variation on the readings that does not seem to follow any specific pattern. Besides, we can see that the values obtained do not correspond exactly to the distance measured manually between sensor and obstacle. In other words, the sensor is not accurate due to different reasons. As an example, the sensor provides a resolution of 1cm and only integer values are recorded. So, if the distance measured has decimals, the measurement simply oscillates between the value rounded down to the nearest integer, and the next higher integer value. In simple distance measurement, and if the amount of points is large enough, this can be easily solved by calculating the average distance, but in the case of point cloud representation it might pose a harder problem to go through, especially if the point redundancy (amount of data scanned for the same physical point) is low.

Concerning the imprecision in all of the readings, we can see a different behaviour depending on the magnitude of the distance measured. In Test 1 for example, the average distance scanned is about 4.4cm lower than the measured distance (a relative error of about 22%). In Test 2 the average absolute error is approximately 0.8cm (therefore, a 0.8% relative error). The last experiment indicates a 0.1cm and 0.02% absolute and relative errors respectively.

Conclusion

Based on the obtained results, it can be inferred that the accuracy of LiDAR Lite v3HP, within the distance boundaries set in the experiments, increases with distance, but even for shorter lengths the sensor provides results complying with the datasheet. However, in terms of frequency, the device did not perform as expected, giving an average frequency of approx. 255Hz, which is considerably less than 1kHz (as specified by the manufacturer). Considering our purpose, LiDAR works precisely and fast enough, as long as the data is treated and processed correctly in order to minimize errors and get a recognizable map of the scanned environment. It should be stated that, according to the performed tests, the 3D scanner that would be using LiDAR Lite v3HP would not be able to detect windows and mirrors.

4.3 Stepper Motor Experiment

In order to better understand the behavior of the stepping motor and compare it with the theoretical facts discussed in Section 3.5, an experiment was performed.

The utilized hardware consisted of Arduino, EasyDriver and the NEMA 17 SM-42BYG011-25 stepper motor, which was unloaded. A DC power supply set on 12V was used to provide power to the motor and driver, whereas the uC was powered through the laptop via USB. All the waveforms were plotted with the aid of the oscilloscope. The stepper was operated in two modes: full stepping and 1/8

microstepping. In both cases, the current-limiting potentiometer on the driver was set to allow a maximum current of 150mA (the lowest possible value, according to the datasheet) through the motor. In terms of software, a simple program was written, consisting of a `for()` loop to turn the shaft by sending a 500Hz pulse train of 50% duty cycle to the STEP pin of the driver. Minor modifications were made when adjusting the stepping resolution was required.

On the oscilloscope, the two currents through the windings were plotted along with the step signal (see Figures 4.5 and 4.6). The current through phase A is shown in yellow and the current through phase B in cyan. In both pictures, the 90° phase shift between the two currents is apparent. The green square wave represents the step signal.

In full stepping (Figure 4.5), the winding currents are an unrefined version of the signals shown earlier in Figure 3.9. The stepping motor was noticeably vibrating and producing a humming sound. The total current consumption was around 90mA.

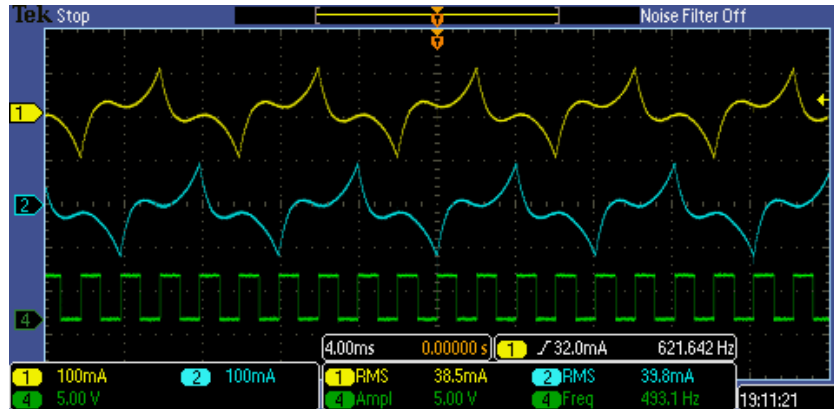


Figure 4.5: Oscilloscope Screenshot of Full Stepping Current and Step Signals

While microstepping with an angle of 0.225° , the currents approximate the ideal sinusoidal form, as it can be seen Figure 4.6 and as described in a prior graph (Figure 3.10). Microstepping visibly reduced the vibrations of the actuator and smoothed the shaft rotation, but the current drawn increased to 150mA.

Angular speed calculations were also carried out as part of the experiment. According to the EasyDriver datasheet, the motor steps at each low to high transition of the step signal. For the chosen frequency of 500Hz, it is straightforward to find that in full stepping (200steps/rev) the motor completes one full rotation in 0.4s and thus rotates at 150RPM. Reducing the stepping angle to 0.225° (1600steps/rev) caused, with 500Hz step signal, the time to complete one revolution to reach 3.2s, translating to a speed of 18.75RPM. A further decrease in angular speed is to be expected when the motor would be loaded.

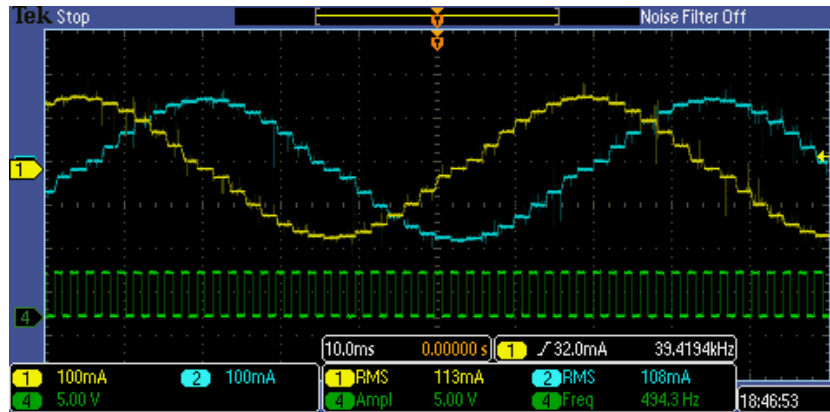


Figure 4.6: Oscilloscope Screenshot of Microstepping Current and Step Signals

Conclusion

The preceding experiment showed that, in practice, the stepping motor behaved as expected and brought evidence that supported the use of microstepping. Even though the measured current consumption was higher in this operating mode, the smaller step size, that would imply an increased point cloud density, and the lower vibration levels were found to be advantages indispensable for the 3D scanner.

4.4 Servo Motor Experiment

The experiment was performed with the objective of testing the servo motor that is to be employed in the 3D scanner. In the experiment, servo motor characteristics such as current consumption, frequency, and control signal waveform were analyzed and compared with the given information. Furthermore, the aim was also to determine the quiescent and stall current due to them being unspecified, to ensure that the battery supply is enough to drive the servo motor along with the rest of the components for a significant amount of time. Another purpose of the experiment was to ensure that the servo motor moves to the specified position.

The setup of the experiment included supplying the control signal to the motor using Arduino Uno (powered by a laptop) and then plotting the current supply waveform along with the PWM control signal on an oscilloscope. During the experiment, the servo shaft position was varied from 0 to 180° (employing the `Servo.h` library for the process), to determine the change in pulse width for the corresponding positions. Additionally, to find the current consumption of the servo motor, two cases were considered based on the external force applied on the motor.

In case 1, the current and input signal were analyzed when no external force was applied on the servo shaft and when the shaft was at the input position. Figure 4.7 for this case, shows a cyan line representing the quiescent or idle current

with the measured value of 60mA, and a yellow waveform (the input signal) with a frequency of 50Hz (20ms period).

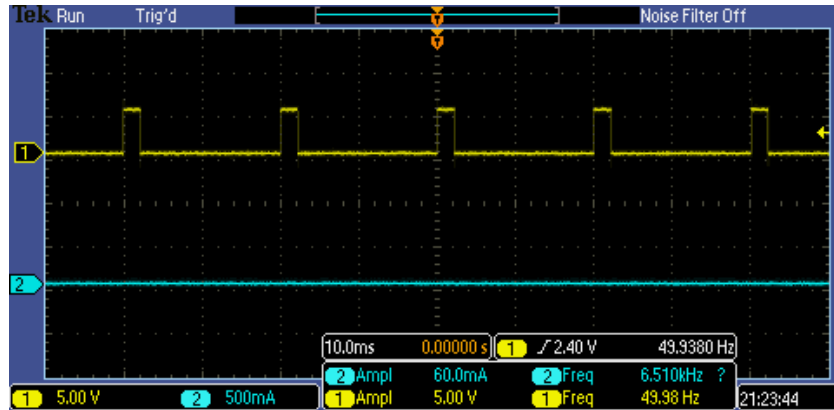


Figure 4.7: Oscilloscope Screenshot of Servo Current and Control Signal for Idle Current

In case 2, the same signals were plotted, but an external force was applied on the shaft of the motor. As it can be seen in Figure 4.8, each command pulse is followed by a pulse of current as the servo motor tries to push back to retain its position. This current is the stall current and was measured up to a maximum of 520mA, being directly proportional to the force applied.

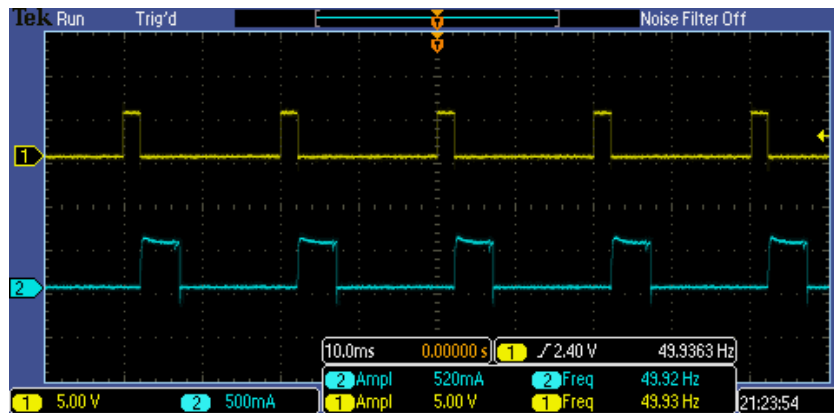


Figure 4.8: Oscilloscope Screenshot of Servo Current and Control Signal for Stall Current

Conclusion

As specified above, the quiescent current and the stall current for the servo motor operating at 5V are 60mA and 520mA respectively. The stall current conveys the maximum current the battery needs to supply for the servo, which is less than the current specified in power estimation for the scanner (see Section 3.8). Moreover,

the experiment also illustrated that, in practice, the servo motor attains the expected shaft position when given the control signal with presumed pulse widths.

4.5 3D Modelling

As mentioned in previous sections, the system to be developed utilizes a 1D LiDAR sensor for obtaining the distances to the neighboring surfaces and thus should be maneuvered using motors to scan these areas. Although, the servo and the stepper motor are responsible for rotating the LiDAR, a structure was still required to hold the LiDAR in place in conjunction with the motors. Moreover, due to the significance of precise measurements from the laser scanner, the supporting structure should be capable of being mounted on the stepper motor and move along with it for acquiring accurate point clouds. In addition, the structure should be lightweight so that there is no substantial increase in the torque needed to spin the LiDAR.

3D printing presented itself as a solution that would satisfy the mentioned conditions and it allowed us to customize the design for the laser scanner. Mainly, two parts were designed using AutoCAD: first, a "U"-shaped mount that can be attached to the shaft of the stepper motor, so that it rotates along with its each step, for a precise azimuth angle. Furthermore, it also supports the servo motor on one of its parallel vertical surface, which is then connected to the second part, i.e., simply a base that holds the LiDAR. This arrangement allows the movement of LiDAR simultaneously with the servo motor, ensuring an accurate inclination angle. Figure 4.9 illustrates this simple design in conjunction with the components.

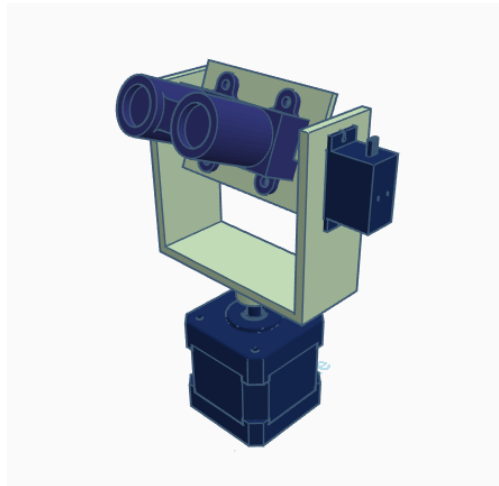


Figure 4.9: Structure Design With Components

4.6 Hardware Implementation

The working of the 3D scanner is the consonance between its hardware and software elements. Accordingly, this section will focus on the implementation of the scanner's hardware element which comprises the components analyzed in Chapter 3. The aim of the system is to create a point cloud, which is a set of points in three-dimensional space.

4.6.1 Coordinate Acquisition

Each point scanned by the developed device should have a unique location at which it could be placed when plotting the point cloud. Thus, the scanner should follow a coordinate system to express the position of its scanned points. This system is the spherical coordinate system which defines the position of its points in three-dimensional space by their radius, i.e., the distance from the origin, the inclination angle and the azimuth angle (see section 3.7). As a result, the 3D scanner is simply an amalgamation of components that generate the required data to represent the position in spherical coordinates.

Due to the line of sight nature of LiDAR Lite v3HP, the 3D scanning apparatus developed utilizes a stepper and a servo motor to move the LiDAR in the horizontal and vertical plane, respectively. Rotating the LiDAR in both planes grants a scan coverage that is equivalent to the volume of a sphere whose radius is equal to the maximum distance measured. Moreover, in order to get the first index, i.e. the radius, the scanner utilizes the LiDAR sensor whose location along with the whole setup can be considered at the origin of the three-dimensional space in which the point cloud is plotted. The distance to the point within the line of sight of LiDAR serves as the radial coordinate with respect to the origin. The next index, the inclination angle, is the polar angle measured from the zenith direction whose orientation is perpendicular to the base of the scanning apparatus. The servo motor is responsible for moving the distance sensor vertically. Hence, the angle at which LiDAR it is positioned by the servo, that is, the servo angle, accounts for the inclination index of the said point. The last index, the azimuth angle, is the angle that is traversed in the horizontal plane. The rotation of LiDAR in this plane is due to the stepper motor and thus the angular rotation of the stepper would provide the third and final coordinate.

An example could further illustrate and summarize the coordinate acquisition: if at a particular instance during the scan, the servo angle is θ , the angular coverage of the stepper until that instance is ϕ , and the distance to the point on an external surface lying in the optical axis of the LiDAR is r , then the spherical coordinates for the said point are expressed as (r, θ, ϕ) . Figure 4.10 illustrates this example.

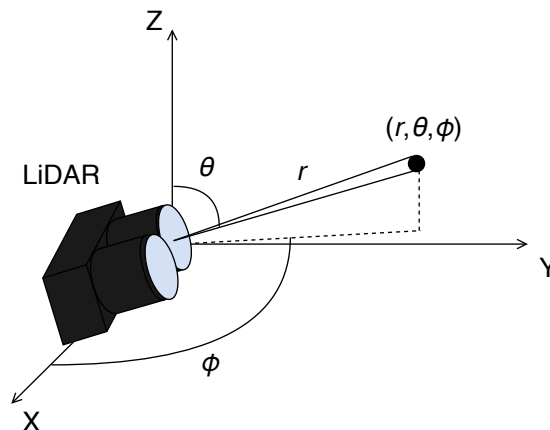


Figure 4.10: Spherical Coordinate Acquisition Using LiDAR

4.6.2 Movement Description

The previous subsection discussed the acquisition of coordinates for a single point, but to create a point cloud that is analogous to the real physical surroundings, the 3D scanner should gather a large amount of such points over entire regions, covering the external surfaces of the entities exposed to its range. Accordingly, the LiDAR is maneuvered by the motors in a fashion that grants it a sphere of activity. Nonetheless, to get a considerable amount of points, at least one trio of coordinates must be registered at every step or at regular intervals of this three-dimensional motion. To achieve such mobility, the servo motor and the stepper motor move in correlation with each other, such that, with every step of the servo motor (attached to LiDAR) the stepper motor completes one full 360° rotation. That means the motion of servo motor shifts, with each step angle, the height of the horizontal plane in which the distance readings are being registered, and the stepper motor rotation ensures the attainment of all the points encircling the setup in that plane. In addition, the total number of full rotations of the stepper motor would be equal to the number of steps the servo motor will move.

4.6.3 Power Transmission During Rotation

The fact that the structure holding the servo motor and LiDAR should rotate along with the stepper motor possesses an issue as the cables transmitting power and electric signals connected to the rotating structure would tangle around and disconnect from the stationary microcontroller and battery. To solve the problem, the structure can be attached to the stepper motor using two methods.

First, using a slip ring, which is an electromechanical component that allows the connection between a stationary and a rotating structure [69], would be a good solution, but employing it would necessitate its placement at the pivot point, i.e.,

the shaft of the stepper motor. As a result, the rotating structure cannot be directly mounted on the motor shaft and would require the use of gears to transmit torque from the motor to it. Moreover, the gears along with a new holding structure need to be 3D printed for custom requirements and using them can result in misalignment of rotation angle between the motor and the rotating part, henceforth effecting the azimuth coordinate.

The second method involves setting the structure directly on top of the motor shaft, eliminating the utility of gears. For the issue with entanglement of wires, the microcontroller would be programmed to rotate the stepper motor alternatively in clockwise and counterclockwise direction. Alternating the direction of rotation would cause the wires to first entwine around the shaft but, changing the direction in the next rotation would impel the wires to untangle, thereby returning to their initial state. The process transpires until the scan is complete. The latter method is implemented for the 3D laser scanner, as it is much simpler due to its implementation through software and adds no extra components as compared to the first method.

4.7 Arduino Programming

This section will focus on how the hardware components and the data transfer is controlled by the Arduino for performing a 3D scan. The following steps chronologically explain the process:

1. The scans are based on the user input received from the interface designed in MATLAB (interface discussed in Section 4.8). Once the interface has registered the input, it sends those values to Arduino. Consequently, the uC is programmed to assign those input values to their proper variables which are utilized for the remaining process to control the scanner. The following list specifies the variables received from the user:
 - **Servo start position:** The angle with respect to the positive z-axis, at which the servo should position itself before the scan starts.
 - **Servo stop position:** The angle, with respect to the positive z-axis, at which the servo should stop and cause the scan to cease.
 - **Servo step angle:** The angle by which the servo shaft should rotate when it changes its position
 - **Stepper step division:** The amount by which the steps of the stepper motor should be divided, i.e., the microstepping ratio.
 - **LiDAR readings per stepper step:** The number of distance readings LiDAR should perform for each step of the stepper.

2. Now that the input has been received, an initialization procedure is done that measures the time taken by the scanner to complete 2 servo steps (2 full stepper rotations). The time measured for this procedure is then sent to MATLAB where the total time is estimated.
3. Once the scan time has been sent to MATLAB, the servo is moved back to the servo start position, after which the actual scan starts, with the servo position being incremented for every full stepper rotation. Furthermore, for each step the stepper motor rotates, LiDAR acquires the distance to the point in its line of sight. When the distance is obtained, it is then printed on the serial monitor along with the number of steps moved by stepper motor and the servo position for that point. These three entities are printed on the serial monitor in the form of a row vector with the first column containing the stepper motor steps, second column for the servo position, and the third column showing the LiDAR distance. After the completion of the scan, the total output would be an n-by-3 matrix where n indicates the total number of points. By printing on the serial monitor, the point data is being transferred through serial communication to MATLAB, where it can be read from the serial port. Figure 4.11 shows an example of data transferred to MATLAB.

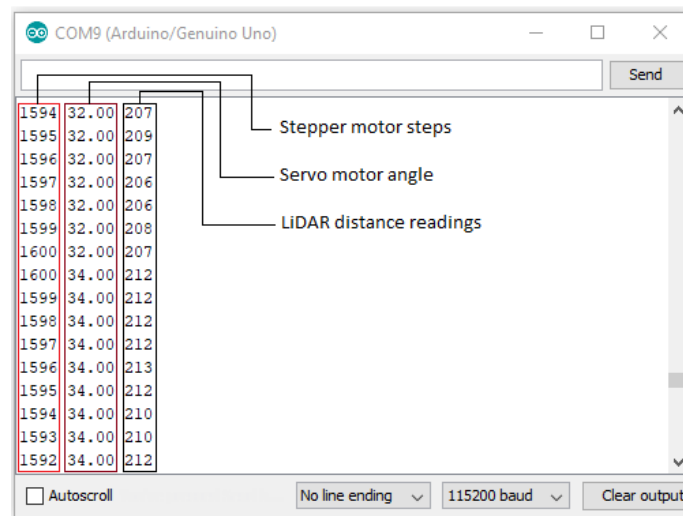


Figure 4.11: Point Data Sent Through Serial Communication

4. A condition is checked every time the servo position is incremented: it is verified whether the current servo position is greater than the servo stop position. If the condition is false, the scanning process is continued. However, if the condition is true, then it signifies that the scan is complete and all the points for that particular region have been registered and could then be plotted into a point cloud using MATLAB.

Arduino Code Flowchart

The chart shown in Figure 4.12 illustrates the overall code flow for a scan. The full Arduino code represented by the flowchart can be found in Appendix A.

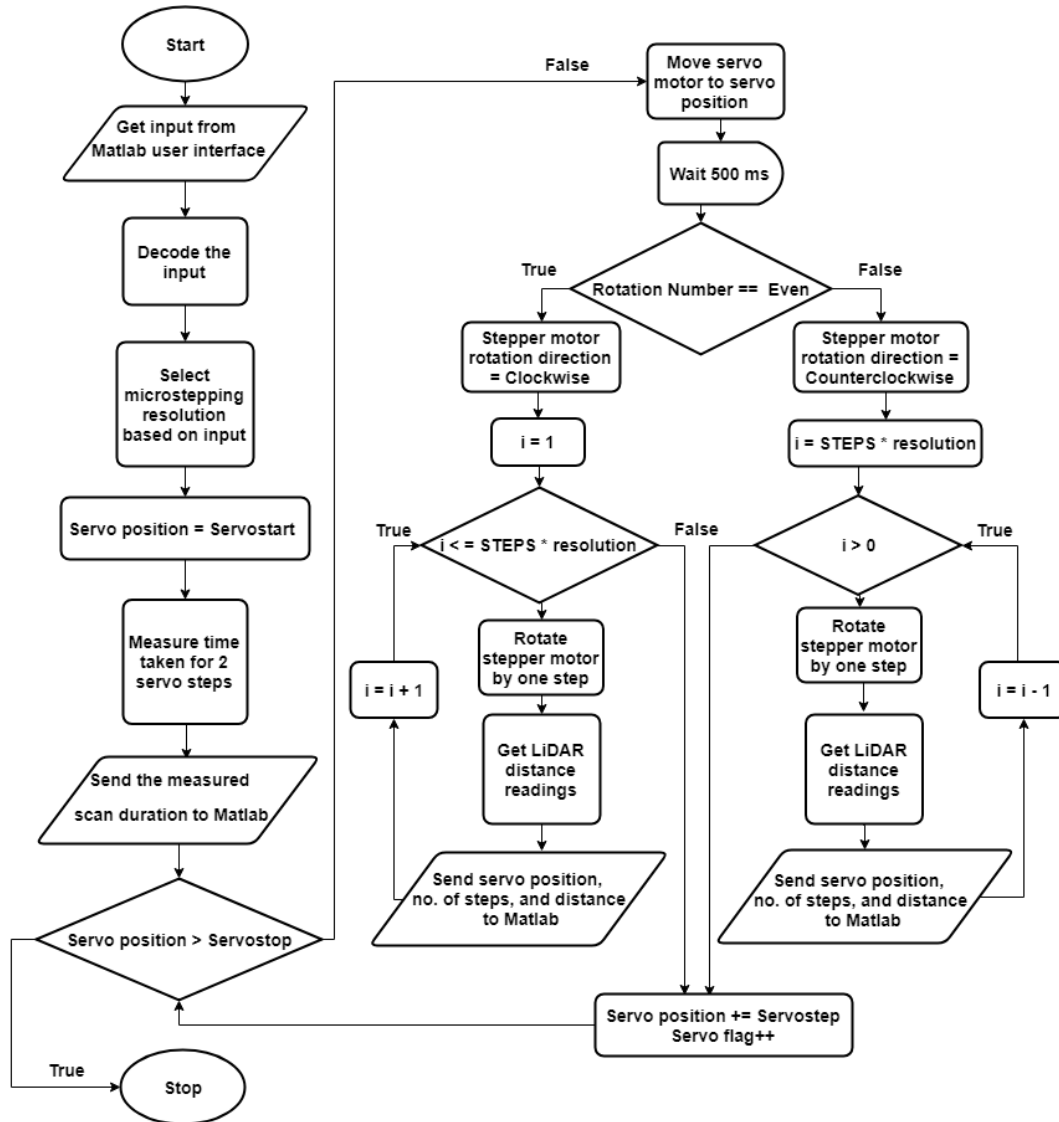


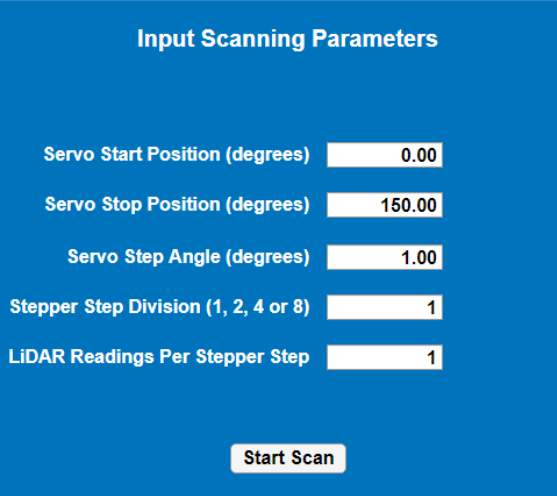
Figure 4.12: Arduino Code Flowchart

4.8 MATLAB Programming

In Section 4.1, the very basis of data transfer and plotting was illustrated without any apparent connection to the 3D scanner which represents the practical part of this project. The present section is meant to describe in a detailed manner the software component of the developed system which deals with data processing.

4.8.1 User Interface

A simple user interface was created using the MATLAB App Designer environment (Figure 4.13). The user is offered the possibility to decide upon different scanning parameters, which are the very ones presented in Section 4.7.



Input Scanning Parameters	
Servo Start Position (degrees)	0.00
Servo Stop Position (degrees)	150.00
Servo Step Angle (degrees)	1.00
Stepper Step Division (1, 2, 4 or 8)	1
LIDAR Readings Per Stepper Step	1

Start Scan

Figure 4.13: The 3D Scanner User Interface in MATLAB App Designer

As shown in the figure, the interface consists of five numeric fields where the user can type the desired values, as well as a button which triggers the scanning process. It should be stated that the code for displaying the fields and buttons on the interface was auto-generated by App Designer, but their functions needed to be programmed (in other words, what happens when a value is written to a field or the button pressed). The values introduced in the numeric fields are retained in distinct variables which are then saved in a common MAT-file. In order for the values to be recognized at the receiving end, and assigned to the right variables in the microcontroller code, each number is multiplied by 10 and a different digit is added to the results. In case of the parameters that can be input as floating-point values (the ones related to the servo), the numbers are multiplied with the minimum power of 10 such that they become integers, and then they are "encoded" as well using the described method. The "Start Scan" button simply runs the main MATLAB script.

Point Cloud Density

The user input discussed in Section 4.8 has a direct impact on the total number of points to be acquired during a scan. This in turn affects the point cloud density for a scanned region. Equation 4.1 below shows the relation between the user input and the number of points.

$$p = \left(\left[\frac{stop - start}{step} \right] + 1 \right) \cdot 200 \cdot m \cdot n \quad (4.1)$$

where:

p = number of points in the resulting point cloud

m = microstepping ratio

$stop$ = servo stop angle

$start$ = servo start angle

$step$ = angle traversed by the servo in one step

n = number of LiDAR readings per stepper step

4.8.2 Main Script

The main MATLAB script is based on the snippets included in Section 4.1. Initially, the file where the user input was saved is loaded into the workspace. Afterwards, the script establishes the serial communication with Arduino, as previously shown, with the command `fopen()` resetting the uC. It sends the input variables as strings using the `fprintf()` function and, after Arduino receives the input, it initializes the scan. Starting from this point, read operations are performed. The read data consists of: the duration of the test scan, i.e., the time it takes the stepper shaft to complete two full revolutions (which is immediately used, along with the relevant input parameters, to estimate the duration of the entire scan), and the matrix containing all the raw data required to construct the point cloud.

When the communication is interrupted, the spherical coordinates are extracted from the transposed data matrix and then the spherical-Cartesian coordinate conversion takes place. The result is an n -by-3 matrix containing a collection of data points in rectangular coordinates, to which the name `xyzPoints` was assigned. Saving this matrix to a different MAT-file every time a scan is performed is crucial if the data needs to be consulted in the future. This can be accomplished by creating a file name using the output of the function `now`, which returns the date and time corresponding to the moment when it is called, and `sprintf`, which formats the data given as argument into a string. Displaying the point cloud is done using the same code as in Section 4.1, but before plotting the data points, the function `pcdenoise()` is utilized to remove the outliers.

A flowchart explaining the MATLAB code is shown in Figure 4.14 and may be of help in the process of better understanding the main script. If consulting the actual code shall be needed, please see Appendix B.

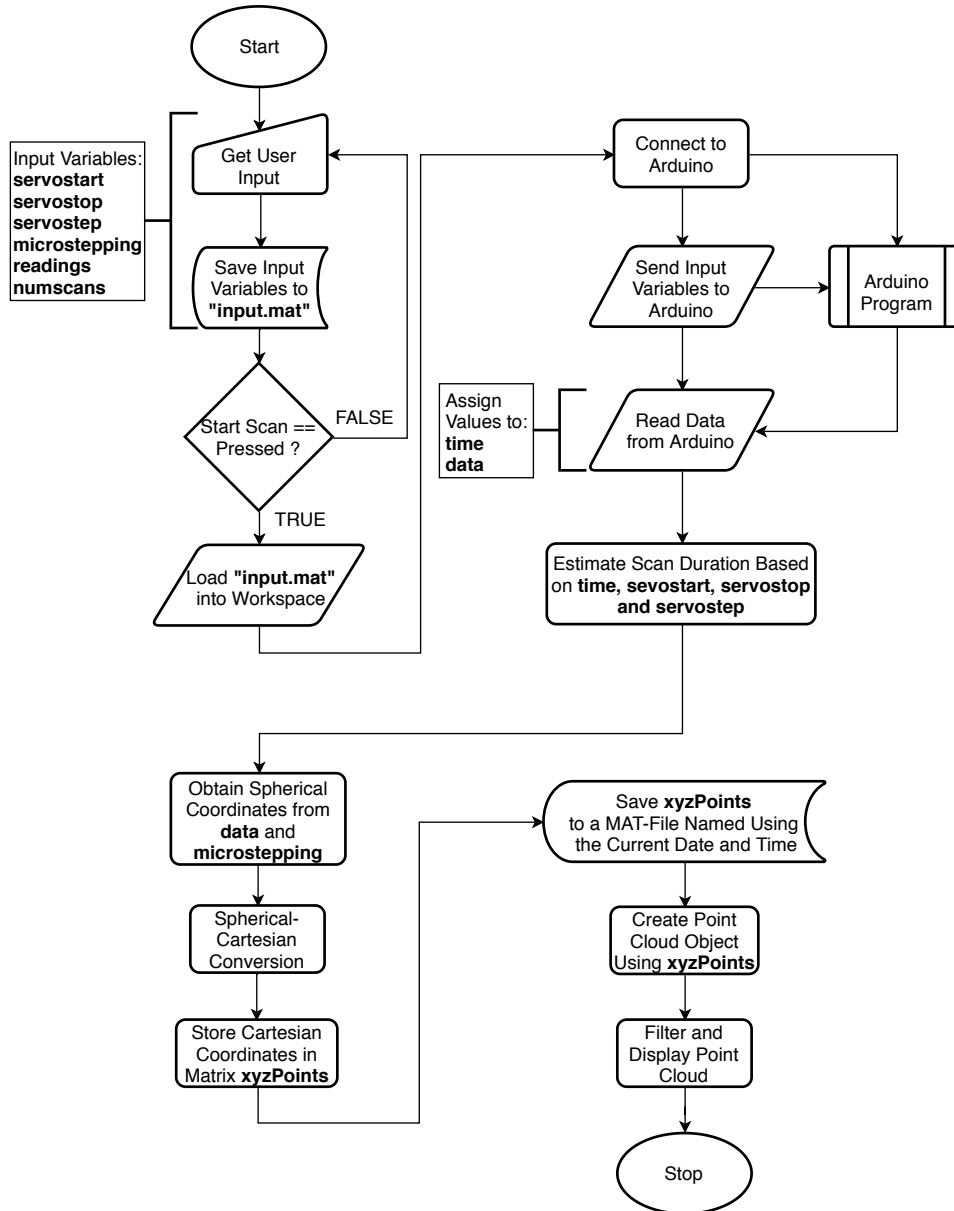


Figure 4.14: Flowchart of MATLAB Code

4.8.3 Displaying Prior Point Clouds

In case a previous scan is desired, a separate, very short script, shall be used. The script loads an *xyzPoints* matrix to the workspace and creates a point cloud object using the loaded data. After filtering is performed, the point cloud is displayed. The name of the file created at the time of taking the respective scan needs to be added in the code.

4.9 Schematic

The schematic shown in Figure 4.15 illustrates the utilized circuit with all the components that constitute the 3D laser scanner.

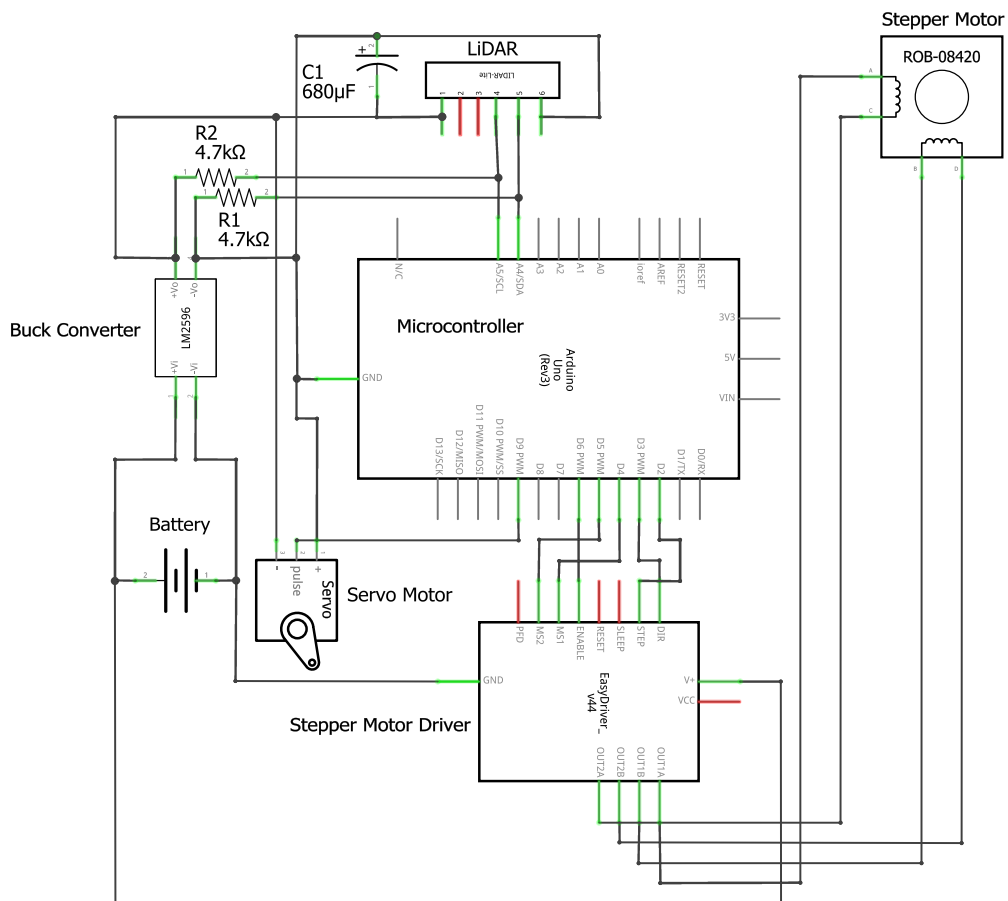


Figure 4.15: Circuit Schematic for 3D LiDAR Scanner

4.10 Prototype

The final result of the implementation of the hardware component of the project is the developed 3D LiDAR scanner shown in Figure 4.16.

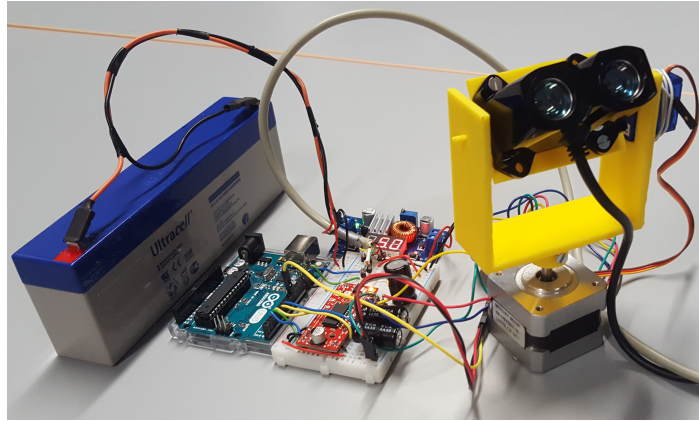


Figure 4.16: Final Setup of the Developed Prototype

Chapter 5

Testing

5.1 Motivation

Throughout this report we have explored the applications of 3D laser scanning and examined the components and the theory of operation of our own 3D laser scanner. The only aspect remaining is the proof of its applicability. Therefore, this section will concentrate on the procedure and results of the testing session carried out in order to validate and assess the developed device.

5.2 Selection of Testing Area

The testing location plays a crucial role and to determine an appropriate place we scanned a variety of locations using our apparatus. These locations differed in their dimensions and aspects such as windows, furnishings etc. The analysis of the scans from each location revealed various factors affecting the final results. First, if any one of the dimensions of the scanned space is significantly greater than the rest of the dimensions, then the distribution of points in the resulting point cloud is non-uniform over the surfaces and makes the further assessment problematic. Second, the presence of specular surfaces affects the quality of the scan. When the scanner sweeps over a window, for example, it scans beyond it and captures unwanted regions outside the target zone for that scan. Surfaces like mirrors, on the other hand, reflect the transmitted beam away and are not registered. Third, the scanned surroundings should contain heterogeneous entities which could later be used to draw comparison between the real physical characteristics and the point cloud. Accordingly, based on the aforementioned factors, a final place was chosen for further testing: a rectangular room with nearly equal length and width. The room contains a single window which would be covered during the scan and cites as an example of a residential space comprising diverse entities such as sofa, bed, book

shelf, chairs etc. The pictures of the final testing location are shown in Figures 5.2 and 5.3 and will be utilized in the forthcoming sections as reference for drawing conclusions. Consequently, all the testing was conducted in this location.

5.3 Setup

The setup of the experiment includes positioning the scanning apparatus at the approximate center of the room with the scanner located at a height of 82cm. This was done so that all the surfaces ambient to the scanner are equidistant, ensuring uniform surface point distribution. Throughout the experiment, the system was powered using a 12V-2.4Ah battery. As for the contents of the testing session, the experiment consisted of five different scans with varied parameters. For each of the five scans, the position of the scanner remains unchanged along with the servo motor range, defined by servo start and servo stop being 35° and 125° respectively. This angular range was decided in order to avoid scanning the monotonous surfaces of the roof and floor of the chamber. The other parameters, namely microstepping ratio, servo step and LiDAR readings, were altered for each scan to examine the effects of these thresholds. Table 5.1 shows the settings for each parameter for all five scans.

Table 5.1: Scanning Parameters

3D Scanner Parameters	Scan Number				
	1	2	3	4	5
Servo Angle Starting Position (degrees)	35	35	35	35	35
Servo Angle Ending Position (degrees)	125	125	125	125	125
Servo Step Angle (degrees)	0.25	1	1	0.25	1
Microstepping Ratio	8	8	1	1	8
LiDAR Readings per Stepper Step	1	3	1	1	1

5.4 Performed Scans

Once the scan is complete, MATLAB automatically plots the point data onto three-dimensional space for the user to see the results. The MATLAB plot corresponding to Scan 2 can be seen in Figure 5.1. From the picture, it is clear that, although the plotting process is automatic, it is difficult to discern the elements of the picture using MATLAB. Furthermore, MATLAB lacks proper tools for navigating through a point cloud, due to the fact that the software's plots are not specifically designed for 3D scanning purposes. Correspondingly, a different software named MeshLab, which is more suited for handling point clouds, was utilized for analyzing the collected data points. However, to work with MeshLab, the point data type needs

to be changed and loaded to the software manually. All the subsequent pictures are taken within MeshLab and present the results of the scans performed.

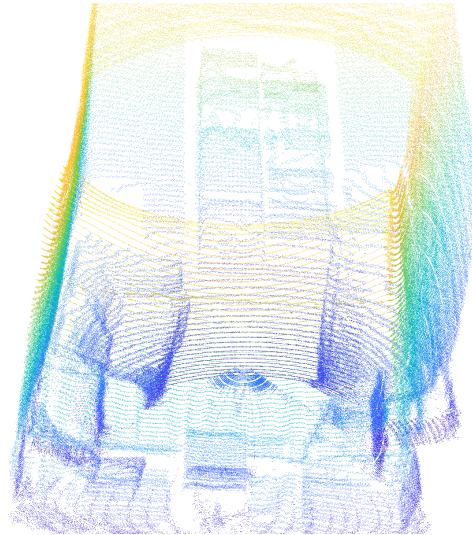


Figure 5.1: Scan 2 Point Cloud in MATLAB

Resemblance

The point clouds in Figures 5.4 and 5.5 (Scan 1) illustrate the side-to-side comparison between the digital representation of the testing area and the actual location shown in Figures 5.2 and 5.3. It can be observed that the geometry of the objects, as well as their position in the room, is preserved in the scan. This resemblance of the constructed point cloud with the real image proves the system's ability to construct a digital model from physical surroundings.

Scan 1

For the sake of clarity, Figure 5.4 has been enlarged and can be seen in Figure 5.6. The point cloud for this scan is a result of 538016 points, obtained after filtering the 577600 points acquired throughout the room, and has the highest density from all the scans performed during the session. Due to the high density, the cluster of points relatively resembled the surfaces, making it easier to discern the layout of the scan. However, with the increased number of points, the time taken to acquire the coordinates also increased and thus this scan took the longest to complete, almost 71 minutes. Furthermore, the power consumption was calculated based on the current measured with an ammeter while the device was performing the scan. The instrument indicated a value of 0.233mA. Therefore, the total power consumed was 2.796W.



Figure 5.2: Testing Area (Side A)



Figure 5.3: Testing Area (Side B)

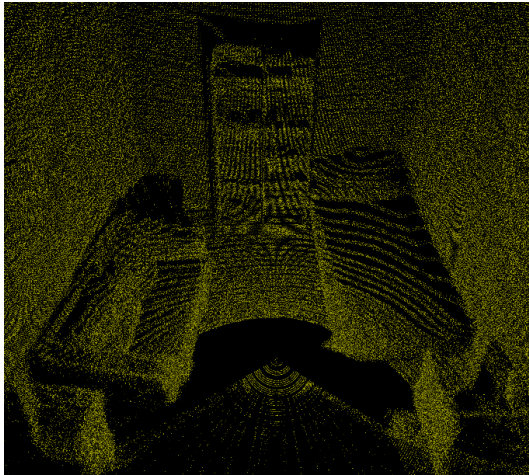


Figure 5.4: Scan 1 Point Cloud (Side A)

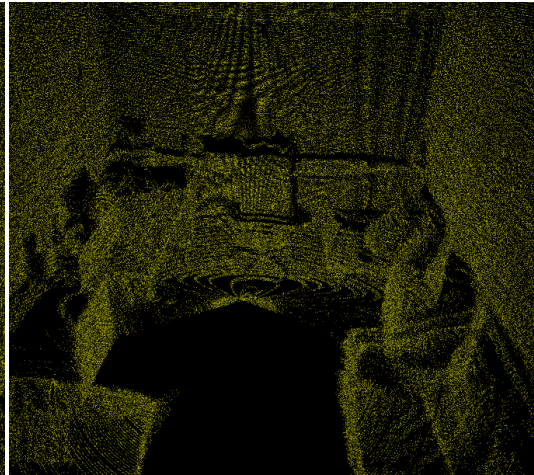


Figure 5.5: Scan 1 Point Cloud (Side B)

Scan 2

The parameters defining the servo step angle and the number of LiDAR readings were changed for the second scan and the picture in Figure 5.7 shows the resulting filtered point cloud with 423364 points (out of 436800 acquired points). Although the visual difference between the point cloud of Scan 1 and Scan 2 is not very noticeable, there is a discrepancy of 114652 points between the two. The reduction in the number of points caused the scan to complete in around 36 minutes, which is substantially less than the time taken for the first scan.

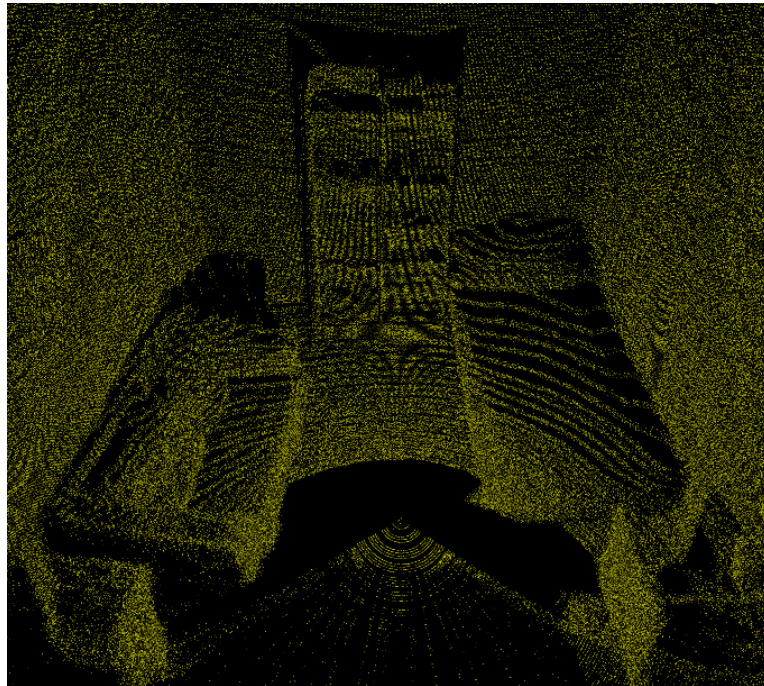


Figure 5.6: Scan 1 Point Cloud (Side A), Enlarged

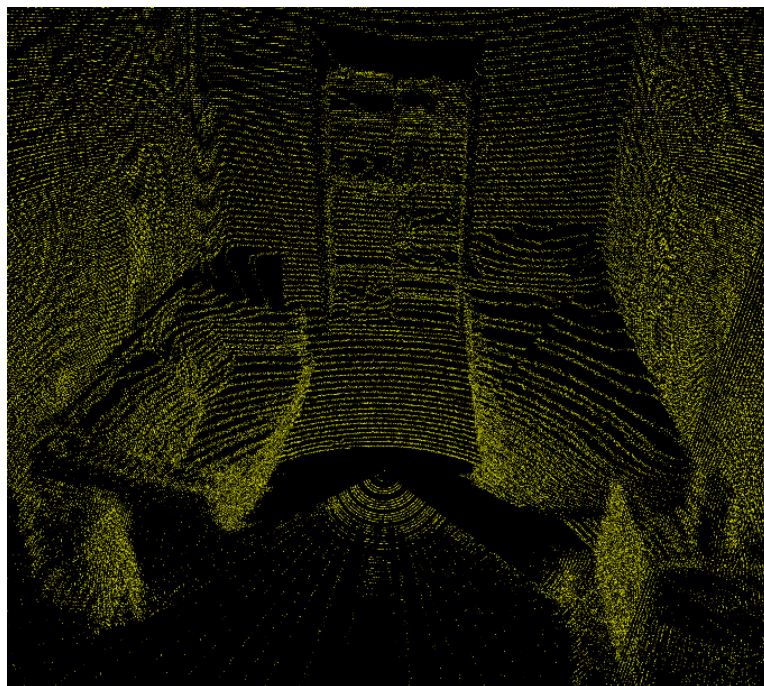


Figure 5.7: Scan 2 Point Cloud (Side A)

Scan 3

The third scan took the least amount of time and has the lowest density with a point count of 16708 points, filtered from an initial data set of 18200. A picture of the output is shown in Figure 5.8. For this picture, the point size had to be increased for the purpose of the report, as the point cloud obtained was indistinguishable using the default size. It is obvious that these points are very far apart as compared to the other two scans. This is due to the increased servo step and decreased microstepping ratio as well as the number of LiDAR readings. Furthermore, the scanned entities are distinctly amorphous, in contrast to the first two scans.

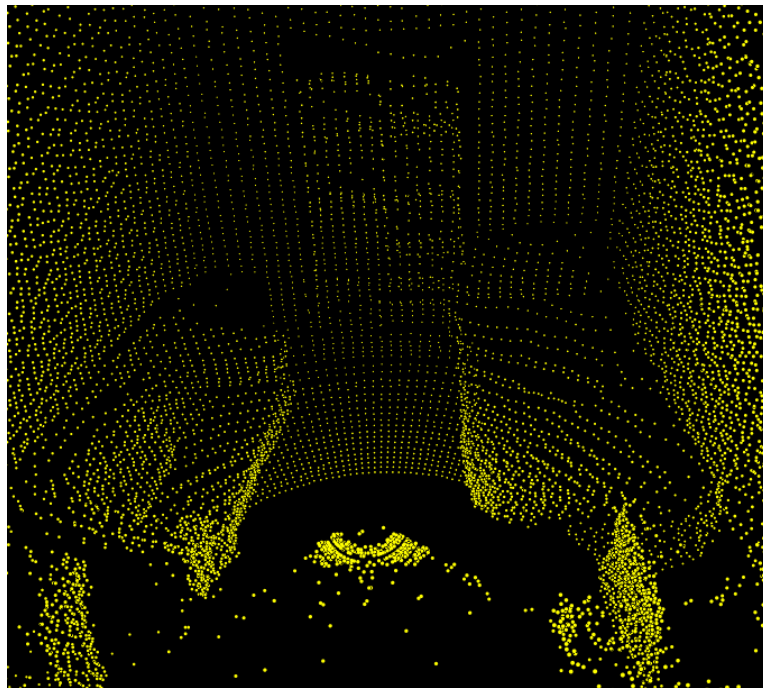


Figure 5.8: Scan 3 Point Cloud (Side A)

The point clouds from the fourth and fifth scan closely resemble the results from the third and second scan respectively, hence these remaining scans will not be discussed any further in this chapter. Nevertheless, the pictures of their generated point clouds are included in Appendix C.

5.5 Scanning Time

After comparing physically recognizable changes between the products of 5 different scanning modes, the time taken to perform the scans should also be considered and analyzed. Table 5.2 shows the estimated time (theoretically calculated) and the practical (measured) time for each individual scan.

Table 5.2: Estimated and Measured Scan Duration

Scanning Time	Testing Session Number				
	1	2	3	4	5
Theoretical (min)	60.17	33.12	2.72	10.22	15.16
Practical (min)	70.51	35.58	2.92	10.56	16.11

At first sight, the time has varied radically among all the scans. For example, in Scan 1, which delivers the highest amount of points, the time spent on scanning is equal to 70.51 minutes. However, Scan 3, situated at the opposite end with respect to density, took only 2.92 minutes. From this point of view, it can be inferred that the scanning time directly depends on the point cloud density, which itself is affected by the selected scanning parameters (see Table 5.1). So, if the user desires a dense 3D digital point cloud, suitable parameters should be selected on the interface, but the time taken for scanning will be longer accordingly.

From the table, we can also see that in all the scans, the practical time is undoubtedly bigger than the theoretical one. The actual time can be considered precise because it was measured internally (using `tic toc` in MATLAB). In earlier experiments (see Section 4.2), it was observed that LiDAR's frequency is changing depending on the distance to the object the sensor is facing – the longer the distance to the object, the higher the frequency of the sensor. Therefore, the scanning time could be affected by the change in distance between LiDAR and the scanned entities.

5.6 Distance Measurement

While looking at the point cloud, it is hard to see if there are some changes in size and distance between certain objects inside the scanned room, compared to the actual environment. However, the goal is making the 3D scanner precise in all the aspects, so comparing the point cloud in terms of size is also a necessary part of testing. Thus, the distance between 2 opposite walls (the ones adjacent to the sofa and the bed) was measured as they are parallel to each other. For obtaining the distance in the actual room, we used a simple tape measure that indicated 280.5cm. In the case of the point clouds, assuming that the size and distances are the same in all obtained scans, the distance was measured only on the point cloud of Scan 1. MATLAB did not offer this possibility, so MeshLab was utilized once again. Through this software, the distance can be simply found by selecting 2 points manually. The result obtained was 274.78cm, which gives an absolute error of 5.72cm. Even though the relative error is arguably small (2.04%), there are several factors that could affect this imprecision in measurement. Firstly, the LiDAR datasheet warns that the readings under 2m can lead to errors of up

to 5cm. So, the sensor is measuring the distance until each wall, therefore, the LiDAR distance is surely smaller than 200cm, which puts the error of 5.72cm close to the indicated limit. A second reason might be the inaccurate measurement in MeshLab. As stated before, this software allows to measure the distance between 2 points, but this manual selection can lead to errors. The problem could arise when a line drawn between 2 selected points is not perpendicular to the corresponding walls.

Conclusion

The aim of the testing session was to prove the functionality of the 3D LiDAR scanner developed by the project group and to derive the conclusion regarding the project. On these grounds, the experiments have been successful: the congruence between the actual pictures and the resulting point clouds proves the competence of the 3D scanner for digitization of tangible structures. From the scans, it suffices to assess that the time consumed for completing a full scan is proportional to the number of points acquired during the scan. Although the time difference and the barely noticeable discrepancy between the first two scans present the second scan as a better mode and dismiss the requirement for a very dense point cloud, the density specification is still based on the application. For instance, the points in the first point cloud are more uniformly distributed over the surfaces as compared to the distinct circular patterns in the second scan. This uniform distribution covers larger extents of the scanned surface, providing more data for surface reconstruction and other later procedures. The third scan was fast, but provided very limited information. The testing session justifies the use of the obtained point clouds for taking distance measurements - although not very accurately - of scanned structures. As for the power consumption during operation, it was considerably less than the one estimated in Section 3.8. This was due to the estimation being done for the worst case scenario.

Chapter 6

Discussion

After all the research and testing performed, we can conclude that the goals stated in the problem analysis were achieved: our system is capable of obtaining the digital image of a closed room. But this does not mean that the system is free of flaws or improvable features. The upgrade of the LiDAR sensor to 2D or 3D, for example, could improve the general performance of the system greatly, as it would allow the sensor to obtain more than 1 point at a time. The point clouds obtained would be sharper and the data rate would increase, therefore the total elapsed time would be reduced. In addition, the general precision of the system would be optimized, making it more reliable for distance measuring within the point cloud.

Although the general performance should not be affected by it, some of the working principles of the system could also be changed in order to make it more orthodox. It is the case of the rotating platform where the LiDAR sensor stands: the usage of a slip ring would enable its continuous rotation in the xy -plane. The group tried to include such a component, but as explained in Subsection 4.6.3 its implementation could not be performed. For improved user-friendliness and safety, the system could also benefit from a 3D-printed casing and a proper tripod that does not interfere with the LiDAR sensor. This would make the overall aesthetics better (less prototype-like), avoid accidents involving liquids to spoil the device and allow the user to fit the height of the sensor to their specific needs. We cannot forget about the range problem: point cloud density reduces quadratically with distance. A possible solution to this could be making a fast study of the size of the scanned room and optimize the amount of scanned points to it.

Another feature that could be implemented in the future would be the inclusion of an SD-card for data transfer. For this project, we decided to send the points from the microcontroller directly to the PC, so data writing would not impact the elapsed time of the scans. A faster and more sophisticated system would make this extra elapsed time tolerable, so all the data could be written in an external memory for its later processing. Therefore, the system would have a backup of the scanned

data and it would be possible to pause and resume the scan at any moment, as the device would know the coordinates of the last scanned point. The inclusion of an uninterruptible power supply would also ensure that no external agent such as battery life can bring the scan to a premature end, and would give the user the choice of using both the power grid or the included battery.

In the field of accuracy, the current system supposes that the stepper and the servo motor move flawlessly, that is, there is no error in their angular positioning. This assumption, especially considering the quality of the used components, is rather naive, and therefore it would be necessary to use an encoder as part of a feedback loop which ensures that the motors move exactly as they are supposed to. In fact, it should be noted that the servo motor utilized in the project is equipped with analog feedback, but due to its inaccuracy and its undesirable effects on the scan duration, it was decided to elude the usage of this feature. Moreover, instead of a servo, a second stepper motor could be employed, which would provide a higher resolution, resulting in a denser point cloud. We must also consider that the obtained digital image in the form of point cloud can be inconvenient for certain environments. On the one hand, there is the perspective problem: we cannot obtain 360° data of the scanned objects, especially if they are behind a certain obstacle. Solving this issue would involve taking readings from different angles and logging the position where those readings were made from. Furthermore, the surface reconstruction topic is rather complex: the understanding of the scientific works found requires a solid knowledge in computational geometry that the group lacks, so the obtained results for the morphology of our data are far from ideal.

Chapter 7

Conclusion

The present report detailed a particular TLS method of digitizing the interior of buildings using a microcontroller-based 3D scanner whose main characteristic is the concurrent operation of a 1D LiDAR, a stepper motor and a servo motor. A comparison of the project specifications (Section 2.10) and the achieved results can serve as a way to evaluate the success of the project.

The developed device utilizes actuators to guide the distance sensor in such a fashion that permits the detection of its surroundings. Sufficient data is provided by the hardware components to generate a graphical representation of the mapped environment - in the form of a point cloud - in MATLAB. The program responsible for data processing and point cloud visualization was designed to allow taking successive scans without overwriting any prior data. Operating thresholds for the apparatus have been implemented through a MATLAB user interface, where desired values of the scanning parameters can be input. A button press initiates a new scan, with no further human assistance needed until the output is produced.

The system was tested in an appropriate environment and the results (see Chapter 5) proved the compatibility between hardware and software and demonstrated that the functionality is the intended one. The 3D data acquisition was realized through the rotation of LiDAR in two planes, a technique which guaranteed the procurement of the spherical coordinates of the points located within the range of the sensor. With a certain combination of parameters, a dense point cloud clearly showing the scanned objects was obtained. Notwithstanding that further improvements are undoubtedly required, it can be inferred that the current state of the equipment is satisfactory and fulfills, therefore, the objective of the project.

Bibliography

- [1] Dr. George W. Calfas Carey L. Baxter Stephen E. Jankiewicz. *Terrestrial Laser Scanning in Archaeology and Cultural Heritage Management*. 2017. URL: <https://www.rdmag.com/article/2017/08/terrestrial-laser-scanning-archaeology-and-cultural-heritage-management>.
- [2] Abid Haleem and Mohd. Javaid. "3D scanning applications in medical field: A literature-based review". In: *Clinical Epidemiology and Global Health* (May 2018). ISSN: 2213-3984. DOI: 10.1016/J.CEGH.2018.05.006. URL: <https://www.sciencedirect.com/science/article/pii/S2213398418300952>.
- [3] John Hitch. *Ten Ways Industry Uses 3D Scanning* | *New Equipment Digest*. 2017. URL: <https://www.newequipment.com/technology-innovations/ten-ways-industry-uses-3d-scanning>.
- [4] T Bilis et al. "THE USE OF 3D SCANNING AND PHOTOGRAMMETRY TECHNIQUES IN THE CASE STUDY OF THE ROMAN THEATRE OF NIKOPO-LIS. SURVEYING, VIRTUAL RECONSTRUCTION AND RESTORATION STUDY". In: (). DOI: 10.5194/isprs-archives-XLII-2-W3-97-2017. URL: <https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W3/97/2017/isprs-archives-XLII-2-W3-97-2017.pdf>.
- [5] *Architecture & Interior Design* | *3D Laser Scanning Services* | *3D Modelling Services* | *Singapore* | *Asia* | *Aries Geomatics*. URL: <http://www.ariesgeomatics.com/architecture-interior-design/>.
- [6] Anita Kristiansen. *The terrestrial laser scanning revolution in forest ecology* | *Publishing blog* | *Royal Society*. URL: <https://blogs.royalsociety.org/publishing/laser-scanning-revolution/>.
- [7] David M Barber, Ross W A Dallas, and Jon P Mills. "Laser Scanning for Architectural Conservation". In: *Journal of Architectural Conservation* 12.1 (2006), pp. 35–52. DOI: 10.1080/13556207.2006.10784959. URL: <https://doi.org/10.1080/13556207.2006.10784959>.
- [8] *3D Scanner Applications* | *Portable 3D Scanning Solutions* | *Artec 3D*. URL: <https://www.artec3d.com/applications>.

- [9] Farzin Amzajerdian et al. "Lidar systems for precision navigation and safe landing on planetary bodies". In: (2011). ISSN: 0277786X. DOI: 10.1117/12.904062. URL: <https://ntrs.nasa.gov/search.jsp?R=20110012163>.
- [10] Sam Klimoski. *Using 3D Terrestrial Laser Scanning to Map/Model the Interior of an Abandoned Theater for Renovation Purposes*. Tech. rep. 2006. URL: <http://www.gis.smumn.edu>.
- [11] Z.M. Bi and Lihui Wang. "Advances in 3D data acquisition and processing for industrial applications". In: *Robotics and Computer-Integrated Manufacturing* 26.5 (Oct. 2010), pp. 403–413. ISSN: 0736-5845. DOI: 10.1016/J.RCIM.2010.03.003. URL: <https://www.sciencedirect.com/science/article/pii/S073658451000013X>.
- [12] Yinghui Xiao, Qingming Zhan, and Qiancong Pang. "3D Data Acquisition by Terrestrial Laser Scanning for Protection of Historical Buildings". In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing (2007)*, pp. 5966–5969. DOI: 10.1109/WICOM.2007.1464. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4341239>.
- [13] Lishan Zhong et al. "Segmentation of Individual Trees From TLS and MLS Data". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.2 (Feb. 2017), pp. 774–787. ISSN: 1939-1404. DOI: 10.1109/JSTARS.2016.2565519. URL: <http://ieeexplore.ieee.org/document/7484258/>.
- [14] *Laser Scanning vs. Photogrammetry - Lanmar Services*. URL: <http://lanmarservices.com/2014/11/07/laser-scanning-vs-photogrammetry/>.
- [15] M Faltýnová et al. "BUILDING FACADE DOCUMENTATION USING LASER SCANNING AND PHOTOGRAMMETRY AND DATA IMPLEMENTATION INTO BIM". In: (). DOI: 10.5194/isprsarchives-XLI-B3-215-2016. URL: <http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/571>.
- [16] Shaoxing Hu et al. *3D RECONSTRUCTION FROM IMAGE SEQUENCE TAKEN WITH A HANDHELD CAMERA*. Tech. rep. URL: http://www.isprs.org/proceedings/XXXVII/congress/4_pdf/99.pdf.
- [17] T Schenk. *Introduction to Photogrammetry*. Tech. rep. Ohio, 2005. URL: <http://www.mat.uc.pt/~gil/downloads/IntroPhoto.pdf>.
- [18] *Visual Revolution of The Vanishing of Ethan Carter*. URL: <http://www.theastronauts.com/2014/03/visual-revolution-vanishing-ethan-carter/>.
- [19] *Top 11 Benefits of BIM (Building Information Modeling) [Updated]*. URL: <https://www.autodesk.com/redshift/building-information-modeling-top-11-benefits-of-bim/>.

- [20] Vassilios Pagounis et al. "3D Laser Scanning for Road Safety and Accident Reconstruction 3D Laser Scanning for Road Safety and Accident Reconstruction". In: *Engineering Surevys for Constrution Works* June (2006), pp. 1–15.
- [21] *Laser Scanning for Accident Reconstruction - SPAR 3D*. URL: <https://www.spar3d.com/news/hardware/laser-scanning-for-accident-reconstruction/>.
- [22] Ong Chee Wei et al. *3D DOCUMENTATION AND PRESERVATION OF HISTORICAL MONUMENT USING TERRESTRIAL LASER SCANNING*. Tech. rep. 1. 2010, pp. 73–90. URL: <https://core.ac.uk/download/pdf/11800777.pdf>.
- [23] Cathi Hayes and Eric Richie. *When to Use Laser Scanning in Building Construction A Guide for General Contractors*. Tech. rep. URL: http://constructrealityxyz.com/test/ebook/LGS_AU_When%20to%20Use%20Laser%20Scanning.pdf.
- [24] Antero Kukko et al. "Multiplatform Mobile Laser Scanning: Usability and Performance". In: *Sensors* 12.9 (Aug. 2012), pp. 11712–11733. ISSN: 1424-8220. DOI: 10.3390/s120911712. URL: <http://www.mdpi.com/1424-8220/12/9/11712>.
- [25] V Budarova et al. "Mobile 3D laser scanning technology application in the surveying of urban underground rail transit". In: (). DOI: 10.1088/1755-1315/46/1/012057. URL: <http://iopscience.iop.org/article/10.1088/1755-1315/46/1/012057/pdf>.
- [26] D. J. Spero and R. A. Jarvis. "A New Solution to the Simultaneous Localization and Map Building Problem". In: *Robotics and ...* 17.3 (2005), pp. 229–241.
- [27] Juan Li et al. *FUSION OF LIDAR 3D POINTS CLOUD WITH 2D DIGITAL CAMERA IMAGE*. Tech. rep. URL: <http://secs.oakland.edu/~li4/research/student/JuanLi2015.pdf>.
- [28] Arun T S and Krishna S. "Autonomous 2D Mapping of an Unknown Environment using Single 1D LIDAR and ROS". In: *International Journal of Engineering Research and V7* (2018). DOI: 10.17577/IJERTV7IS030022.
- [29] European Commission. "Low Voltage Directive: Electrical equipment designed for use within certain voltage limits". In: (2014).
- [30] European Commission. "DIRECTIVE 2012/19/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 4 July 2012 on waste electrical and electronic equipment (WEEE) (recast) (Text with EEA relevance)". In: *Official Journal of the European Union* (2012). URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32012L0019&from=EN>.
- [31] *IEC 60825-1:2014 | IEC Webstore*. URL: <https://webstore.iec.ch/publication/3587>.

- [32] Jean-Paul Mosnier. *Laser Safety*. 2017. URL: https://www.dcu.ie/sites/default/files/science_and_health/laser_safety.pdf.
- [33] Arduino. *Arduino Uno Rev3*. 2018. URL: <https://store.arduino.cc/arduino-uno-rev3>.
- [34] Atmel. *ATmega328 / P*. 2016.
- [35] *Ultrasonic Ranging Module HC-SR04*. Tech. rep. URL: www.Electfreaks.com.
- [36] *GP2Y0A02YK0F*. Tech. rep.
- [37] National Oceanic US Department of Commerce and Atmospheric Administration. "What is LIDAR". In: (). URL: <https://oceanservice.noaa.gov/facts/lidar.html>.
- [38] Garmin. *LIDAR-Lite v3HP Operation Manual and Technical Specifications*. 2018. URL: https://cdn.sparkfun.com/assets/9/a/6/a/d/LIDAR_Lite_v3HP_Operation_Manual_and_Technical_Specifications.pdf.
- [39] Bill Schweber and Mouser Electronics. *Considerations in Choosing Motors for Robotics*. Tech. rep. URL: www.mouser.com.
- [40] Bill Earl. *Analog Feedback Servos*. 2018. URL: <https://learn.adafruit.com/analog-feedback-servos>.
- [41] Dr. Douglas W. Jones. *Stepping Motors Fundamentals | Microchip Technology Inc*. Tech. rep. University of Iowa, 2004, p. 22. URL: <https://homepage.divms.uiowa.edu/~jones/step/an907a.pdf>.
- [42] Paul Acarnley. "Stepping Motors: a guide to theory and practice". In: (2002). ISSN: 00135127. DOI: 10.1049/PBCE063E. URL: <http://digital-library.theiet.org/content/books/ce/pbce063e>.
- [43] Matthew Burris. *Choosing Between Stepper Motors or Servo Motors*. 2018. URL: <https://www.lifewire.com/stepper-motor-vs-servo-motors-selecting-a-motor-818841>.
- [44] Donald Labriola. *Why open-loop steppers lose steps, and how to solve the problem | Machine Design*. 2005. URL: <https://www.machinedesign.com/motorsdrives/why-open-loop-steppers-lose-steps-and-how-solve-problem>.
- [45] George Leger. *Unipolar Stepper Motor vs Bipolar Stepper Motors*. 2012. URL: <https://www.circuitspecialists.com/blog/unipolar-stepper-motor-vs-bipolar-stepper-motors/>.
- [46] William H Yeadon and Alan W Yeadon. *HANDBOOK OF SMALL ELECTRIC MOTORS*. McGraw-Hill, 2001. ISBN: 007072332X. URL: http://s1.nonlinear.ir/epublish/book/Handbook_of_Small_Electric_Motors_007072332X.pdf.

- [47] PoLabs. *Stepper motor driver - explanation - PoBlog*. 2015. URL: <https://blog.poscope.com/stepper-motor-driver/>.
- [48] Marc McComb. "Micro-stepping for Stepper Motors". In: (2008).
- [49] Zaber Technologies Inc. "Microstepping Tutorial". In: (2014). DOI: 10.1007/s00248-010-9783-6.
- [50] Brian Schmalz. *Easy Driver stepper motor driver*. URL: <http://www.schmalzhaus.com/EasyDriver/>.
- [51] Allegro MicroSystems. *Microstepping Driver with Translator*. 2013. URL: <https://cdn.sparkfun.com/datasheets/Robotics/A3967-Datasheet.pdf>.
- [52] *How RC Servos Works*. URL: http://www.pcbheaven.com/wikipages/How_RC_Servos_Works/.
- [53] Jameco Electronics. *How Do Servo Motors Work? | Servo Motor Controller*. 2018. URL: <https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html>.
- [54] *Introduction to Servomotor Programming*. Tech. rep. URL: www.basicxandrobotics.com.
- [55] *How to Build a Robot Tutorials - Society of Robots*. URL: http://www.societyofrobots.com/actuators_servos.shtml.
- [56] Paul Dawkins. *Calculus III - Spherical Coordinates | Paul's Online Notes*. 2018. URL: <http://tutorial.math.lamar.edu/Classes/CalcIII/SphericalCoords.aspx>.
- [57] C. Henry Edwards and David E. Penney. *Calculus: Early Trancendentals*. 7th. 2008.
- [58] *FS90 9g Mini Servo*. URL: <https://www.addicore.com/FS90-Mini-Servo-p/113.htm>.
- [59] *UL2.4-12 Dimensions F1 Terminal UL 2.4-12 12V 2.4AH*. Tech. rep. URL: www.ultracell.co.uk.
- [60] Bill Schweber. *Understanding Linear Regulator Advantages | DigiKey*. 2017. URL: <https://www.digikey.com/en/articles/techzone/2017/sep/understanding-the-advantages-and-disadvantages-of-linear-regulators>.
- [61] *Switching Regulator Fundamentals | Texas Instruments*. Tech. rep. 2016. URL: www.ti.com.
- [62] Eric Coates. *Buck Converters*. 2018. URL: <http://www.learnabout-electronics.org/PSU/psu31.php>.
- [63] XLSEMI. *Datasheet 5A 180KHz 36V Buck DC to DC Converter XL4015*. Tech. rep. URL: http://gotronik.pl/img/xl4015_datasheet.pdf.

- [64] Edward B. Magrab et al. *An Engineer's Guide to MATLAB*. 2011. ISBN: 978-0-13-199110-1.
- [65] Xin Yan and Xiao Gang Su. *Linear Regression Analysis*. WORLD SCIENTIFIC, June 2009. ISBN: 978-981-283-410-2. DOI: 10.1142/6986. URL: <https://www.worldscientific.com/worldscibooks/10.1142/6986>.
- [66] Martin A Fischler and Robert C Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* 24 (1981), p. 15. ISSN: 00010782. DOI: 10.1145/358669.358692.
- [67] Radu Bogdan Rusu et al. "Towards 3D Point cloud based object maps for household environments". In: *Robotics and Autonomous Systems* 56.11 (Nov. 2008), pp. 927–941. ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2008.08.005. URL: <https://www.sciencedirect.com/science/article/pii/S0921889008001140>.
- [68] *MATLAB Documentation - MathWorks Nordic*. URL: <https://se.mathworks.com/help/matlab/index.html>.
- [69] *Slip ring principle*. URL: <http://www.trolexengineering.co.uk/what-is-a-slip-ring.html>.

Appendix A

Arduino Code

```
1 /**Libraries**/  
2  
3 #include <LIDARLite.h> // Include Lidar library  
4 #include <Wire.h>      // Include I2C library  
5 #include <Servo.h>     // Include Servo library to control servo motor  
6  
7 /**Constants and Global Variables**/  
8  
9 /*LiDAR*/  
10  
11 LIDARLite Lidar;      // Create LiDAR object  
12 int readings;        // No. of LiDAR readings per stepper step  
13  
14 /*Servo*/  
15  
16 #define SERVOCTL 9    // Servo control pin  
17  
18 Servo RCServo;       // Create servo object  
19 float servopos;      // Servo current position  
20 float servostart;    // Servo start position  
21 float servostop;     // Servo end position  
22 float servostep;     // Servo step angle (angle by which the servo moves)  
23 int servoflag = 0;   // Keeps track of the number of steps the servo takes  
24  
25 /*Stepper*/  
26  
27 #define STP 2         // Step pin  
28 #define DIR 3        // Stepping direction pin  
29 #define MS1 4        // Microstep selector 1 pin  
30 #define MS2 5        // Microstep selector 2 pin  
31 #define EN 6         // Enable pin
```

```

32
33 #define STEPS 200      // Default number of steps
34
35 int resolution;      // Microstepping resolution
36
37 /***Functions***/
38
39 void getInput();      // Get input parameters from user
40 unsigned long getScanTime(); // Get the duration of two full stepper
    rotations
41
42 void servoRotate();   // Rotate servo
43
44 void stepperRotate(); // Rotate stepper
45 void stepperCW();     // Rotate stepper in clockwise direction
46 void stepperCCW();   // Rotate stepper in counter clockwise direction
47 void microstepSelect(); // Select stepping resolution
48
49 void printData(int, int); // Print the data on the serial monitor
50
51 void setup() {
52     Serial.begin(115200); // Set data transfer rate
53
54     getInput();          // Get the input from MATLAB interface
55
56     Lidar.begin(0, true);
57     Lidar.configure(0);
58     Serial.print('*');
59
60     //Setting stepper motor driver pins as output
61     pinMode(STP, OUTPUT);
62     pinMode(DIR, OUTPUT);
63     pinMode(MS1, OUTPUT);
64     pinMode(MS2, OUTPUT);
65     pinMode(EN, OUTPUT);
66     digitalWrite(EN, LOW); // Enable driver
67     microstepSelect();
68
69     RCServo.attach(SERVOCTL); // Attach the servo to SERVOCTL
70     servopos = servostart;    // Set the starting servo position to
    servostart
71     servoflag = 0;
72     RCServo.write(servopos); // Move the servo to start
73     delay(1000);
74
75     unsigned long scanTime = getScanTime();
76     Serial.print('*');

```

```
77 Serial.print(scanTime); // Print scantime on the serial
78 Serial.print('*');
79 delay(1000);
80 }
81 /* Main loop. Call the function that rotates the servo and
82 * stop the scanner when the servo reaches the end point
83 * specified in the interface.
84 */
85 void loop() {
86   servoRotate();
87   if(servopos > servostop) {
88     Serial.print('*'); // Print the terminator character
89     while(1);
90   }
91 }
92 /* Get the input form MATLAB. The input parameters are sent as strings.
93 * Input is read and "decoded" in order to perform the correct
94 * assignments of values.
95 */
96 void getInput() {
97   String input;
98   unsigned long param = 0;
99   bool f1, f2, f3, f4, f5;
100  f1 = 0;
101  f2 = 0;
102  f3 = 0;
103  f4 = 0;
104  f5 = 0;
105  while (1) {
106    if (Serial.available() > 0) {
107      input = Serial.readString();
108      param = input.toInt();
109      switch (param % 10) {
110        case 1:
111          servostart = param / 10 / 100.0;
112          f1 = 1;
113          digitalWrite(led1, HIGH);
114          break;
115        case 2:
116          servostop = param / 10 / 100.0;
117          f2 = 1;
118          digitalWrite(led2, HIGH);
119          break;
120        case 3:
121          servostep = param / 10 / 100.0;
122          f3 = 1;
123          digitalWrite(led3, HIGH);
```



```

124         break;
125     case 4:
126         resolution = param / 10;
127         f4 = 1;
128         digitalWrite(led4, HIGH);
129         break;
130     case 5:
131         readings = param / 10;
132         f5 = 1;
133         digitalWrite(led5, HIGH);
134         break;
135     }
136 }
137 if (f1 && f2 && f3 && f4 && f5) break;
138 }
139 }
140 /* Measure the time taken to scan for two servo steps
141 */
142 unsigned long getScanTime() {
143     unsigned long time1 = millis(); // Start timer
144     servoRotate();                 // Move servo two steps
145     servoRotate();
146     servopos = servostart;         // Set the servo position back to start
147     unsigned long time2 = millis() - time1;
148     return time2;                 // Return time
149 }
150 /* Rotate the servo with the specified step and, for each
151 * servo step, rotate the stepper. Keep track of the servo
152 * position and the number of steps the servo has moved.
153 */
154 void servoRotate() {
155     RCServo.write(servopos);
156     delay(500);                   // Delay to let the servo reach the
        position
157     stepperRotate();
158     servopos += servostep;
159     servoflag += 1;
160 }
161 /* Rotate the stepper 360 degrees. Based on whether
162 * the number stored in servoflag is even or odd, the
163 * rotation is either clockwise or counterclockwise.
164 */
165 void stepperRotate() {
166     if (servoflag % 2 == 0) {
167         stepperCCW();
168     } else {
169         stepperCW();

```

```

170 }
171 }
172 /* Rotate the stepper clockwise. A loop causes the movement by
173 * creating as many low-to-high transitions as the number of
174 * microsteps selected. For every step, LiDAR takes the number of
175 * distance readings specified by the user.
176 */
177 void stepperCW() {
178     int r;
179     digitalWrite(DIR, LOW);
180     for(int i = 1; i <= STEPS * resolution; i ++) {
181         digitalWrite(STP,LOW);
182         delay(1);
183         digitalWrite(STP,HIGH);
184         delay(1);
185         for(int j = 0; j < readings; j ++) {
186             r = getLidarData(i);
187             printData(i, r);
188         }
189     }
190 }
191 /* Rotate the stepper counterclockwise. The function is
192 * the same as stepperCW(), but the direction is changed.
193 */
194 void stepperCCW() {
195     int r;
196     digitalWrite(DIR, HIGH);
197     for(int i = STEPS * resolution; i > 0; i --) {
198         digitalWrite(STP,LOW);
199         delay(1);
200         digitalWrite(STP,HIGH);
201         delay(1);
202         for(int j = 0; j < readings; j ++) {
203             r = Lidar.distance();
204             printData(i, r);
205         }
206     }
207 }
208 /* This function sets the logic for MS1 and MS2 in order to create the
209 * desired microstepping resolution.
210 */
211 void microstepSelect() {
212     switch(resolution) {
213         case 1:
214             digitalWrite(MS1, LOW);    // logic to full
215             digitalWrite(MS2, LOW);    // stepping resolution
216             break;

```

```
217     case 2:
218         digitalWrite(MS1, HIGH); // Set logic to half
219         digitalWrite(MS2, LOW);  // stepping resolution
220         break;
221     case 4:
222         digitalWrite(MS1, LOW);  // Set logic to 1/4
223         digitalWrite(MS2, HIGH); // microstepping resolution
224         break;
225     case 8:
226         digitalWrite(MS1, HIGH); // Set logic to 1/8
227         digitalWrite(MS2, HIGH); // microstepping resolution
228         break;
229     }
230 }
231 /* This function prints the stepper motor steps in column 1,
232 * servo position in column 2 and LiDAR readings in column 3
233 * of the serial monitor. The serial monitor is then read using
234 * MATLAB which inputs the data into a 960000x3(max) matrix.
235 */
236 void printData(int stepp, int radius) {
237     Serial.print(stepp); // Print each step of stepper motor
238     Serial.print(" ");
239     Serial.print(servopos); // Print the servopos in degrees
240     Serial.print(" ");
241     Serial.print(radius); // Print the distance per each stepper step
242     Serial.print("\n");
243 }
```

Appendix B

MATLAB Code

B.1 User Interface

```
1 % The interface was created in MATLAB App Designer. This MATLAB feature
2 % provided the graphical components (e.g.: buttons) with which the user
3 % can interact, but the "result" of the interaction had to be programmed.
4 % Only this code is shown below.
5 % The interface consists of 5 numeric fields, where the user has to
6 % select the following parameters:
7 % angle at which the servo should start : servostart
8 % angle where the servo should stop : servostop
9 % angle by which the servo moves : servostep
10 % number to divide the default stepper step by : microstepping
11 % number of LiDAR readings per stepper step : readings
12 % These values need to be sent to Arduino and for
13 % them to be recognized at the receiving end, they are multiplied with
14 % an adequate power of 10, added with a distinctive digit and then
15 % converted to character arrays.
16 % All the variable are saved to a common .mat file that can be loaded
17 % in the main script
18
19 servostart = app.ServoStartPositiondegreesEditField.Value;
20 servostart = num2str(servostart * 1000 + 1);
21 input_file = 'input.mat';
22 save(input_file, 'servostart', '-append');
23 %%
24 servostop = app.ServoStopPositiondegreesEditField.Value;
25 servostop = num2str(servostop * 1000 + 2);
26 input_file = 'input.mat';
27 save(input_file, 'servostop', '-append');
28 %%
```

```

29 servostep = app.ServoStepAngledegreesEditField.Value;
30 servostep = num2str(servostep * 1000 + 3);
31 input_file = 'input.mat';
32 save(input_file, 'servostep', '-append');
33 %%
34 microstepping = app.StepperStepDivision124or8EditField.Value;
35 microstepping = num2str(microstepping * 10 + 4);
36 input_file = 'input.mat';
37 save(input_file, 'microstepping', '-append');
38 %%
39 readings = app.LiDARReadingsPerStepperStepEditField.Value;
40 readings = num2str(readings * 10 + 5);
41 input_file = 'input.mat';
42 save(input_file, 'readings', '-append');
43
44 % Additionally, there is a button which starts the scan when pressed
45 run('full_code.m');

```

B.2 Main Code

```

1 clc          % Clear command window
2 clear       % Clear workspace
3 close all   % Delete all figures
4
5 % Load the variables that store the user input into the workspace
6 load('input.mat');
7 % Connect to Arduino. Send the user input. Get the raw data
8 data = serialComm(servostart,servostop,servostep,microstepping,readings);
9 % Convert the microstepping resolution from a character array to a
10 % number
11 msres = floor(str2num(microstepping) / 10);
12 % Spherical-Cartesian coordinate conversion
13 xyzPoints = sphCart(data,msres);
14 % Get the current date and time, format them as a string and use it to
15 % create a file name that contains the date and time the scan
16 % finished
17 scan_data = sprintf('Scan_%s.mat', datestr(now,'mm-dd-yyyy HH-MM'));
18 % Save the matrix containing the xyz-coordinates in a .mat file that
19 % uses the previously created file name
20 save(scan_data, 'xyzPoints');
21 % List the variables stored in the created file
22 disp('Contents of the file:')
23 whos('-file',scan_data)
24 % Display the point cloud

```

```
25 displayPtCloud(xyzPoints);
```

B.3 Functions

B.3.1 Serial Communication

```
1 % Function that establishes and interrupts the serial communication
2 % with Arduino and returns the useful raw data. It calls three other
3 % functions utilized to transfer data to/from the uC, and calculate
4 % the scan duration
5 function data = serialComm(start,stop,step,micro,readings)
6     % Remove from memory any serial port object conncted to the port
7     % used by Arduino
8     delete(instrfind('Port','COM12'))
9     % Create serial port object and specify the properties as follows:
10    % Baud Rate: 115200 bit/s
11    % Terminator Character: '*'
12    % Input Buffer Size; 14MB
13    % Port Timeout: 2 hours
14    arduino_port = serial('COM12','BaudRate',115200);
15    arduino_port.Terminator = '*';
16    arduino_port.InputBufferSize = 14000000;
17    arduino_port.Timeout = 7200;
18    % Connect the serial port object to Arduino
19    fopen(arduino_port)
20    % Write the input data to Arduino
21    writeData(arduino_port,start,stop,step,micro,readings);
22    % Read data from Arduino
23    data = readData(arduino_port,start,stop,step);
24    % Disconnect from Arduino and remove the serial object
25    % from memory and workspace
26    fclose(arduino_port)
27    delete(arduino_port)
28    clear arduino_port
29 end
```

B.3.2 Write Data

```
1 % Function that sends the input parameters formatted as strings to Arduino.
2 % The MATLAB execution is paused to give time to the uC to read the input
3 % data
4 function writeData(arduino_port,start,stop,step,micro,readings)
5     pause(2);
```

```

6   fprintf(arduino_port, '%s', start);
7   pause(2);
8   fprintf(arduino_port, '%s', stop);
9   pause(2);
10  fprintf(arduino_port, '%s', step);
11  pause(2);
12  fprintf(arduino_port, '%s', micro);
13  pause(2);
14  fprintf(arduino_port, '%s', readings);
15 end

```

B.3.3 Read Data

```

1  % Function that reads data form Arduino and returns only the matrix
2  % containig data from the sesnors and actuators. Every call of fscanff
3  % reads at most the specified number of values until the terminator
4  % character is reached
5  function data = readData(arduino_port, start, stop, step)
6      residual = fscanf(arduino_port, '%c', 20000)
7      testscan = fscanf(arduino_port, '%f', [3, 1600]);
8      time = fscanf(arduino_port, '%i', 10);
9      scantime = getScanTime(time, start, stop, step)
10     data = fscanf(arduino_port, '%f', [3, 960000]);
11 end

```

B.3.4 Calculate Scan Time

```

1  % Function that calculates the scan duration (in moinutes) based on the
2  % time read from the uC follwoing input parameters: servo start angle,
3  % servostop angle and the angle by which the servo moves.
4  % The variables needed in the calculation are first converted to numbers
5  function scantime = getScanTime(time, servostart, servostop, servostep)
6      servostart = floor(str2num(servostart) / 10) / 100;
7      servostop = floor(str2num(servostop) / 10) / 100;
8      servostep = floor(str2num(servostep) / 10) / 100;
9      time = time / 2 * floor((servostop - servostart) / servostep + 1);
10     scantime = time / 60000;
11 end

```

B.3.5 Coordinate Conversion

```

1  % Function that extracts the spherical coordinates from the raw

```

```

2 % data and outputs a matrix containing the Cartesian coordinates
3 % of the scanned points. Each coordinate represents one column:
4 % col1 : x; col2 : y; col3 : z.
5 function xyzPoints = sphCart(data,msresolution)
6     % Transpose the data matrix, so that each parameter is on one column
7     A = data.';
8     % Extract spherical coordinates for the data matrix and convert the
9     % angles to radians
10    azimuth = A(:,1).*(2*pi/(200*msresolution));
11    inclination = A(:,2).*(pi/180);
12    radius = A(:,3);
13    % Use the formulas for coordinate conversion
14    x = radius .* sin(inclination) .* cos(azimuth);
15    y = radius .* sin(inclination) .* sin(azimuth);
16    z = radius .* cos(inclination);
17    % Concatenate the arrays into a matrix
18    xyzPoints = [-x y z];
19 end

```

B.3.6 Display Current Point Cloud

```

1 % Function which creates a point cloud object out of the matrix of
2 % Cartesian coordinates and plots it
3 function displayPtCloud(xyzPoints)
4     % Create a point cloud object with the specified xyz-coordinates
5     ptCloud = pointCloud(xyzPoints);
6     % Filter the point cloud
7     ptCloudFilt = pcdenoise(ptCloud);
8     % Create new figure window
9     figure('Name','Point Cloud','NumberTitle','off');
10    % Display the denoised point cloud without axes names
11    pcshow(ptCloudFilt);
12    axis off;
13 end

```

B.3.7 Load and Display a Specific Point Cloud

```

1 clc          % Clear command window
2 clear        % Clear workspace
3 close all    % Delete all figures
4
5 % Load data from a prior scan (file name should be modified accordingly)
6 prevscan = 'Scan_01-01-2018 00-00.mat';

```



```
7 load(prevscan);
8 % Create a point cloud object with the specified xyz-coordinates
9 ptCloud = pointCloud(xyzPoints);
10 % Filter the point cloud
11 ptCloudFilt = pcdenoise(ptCloud);
12 % Create new figure window
13 figure('Name',prevscan,'NumberTitle','off');
14 % Display the denoised point cloud without axes names
15 pcshow(ptCloudFilt);
16 axis off
```

Appendix C

Obtained Point Clouds

The Figures C.1 and C.2 illustrate the resulting point clouds from Scan 4 and Scan 5 respectively.

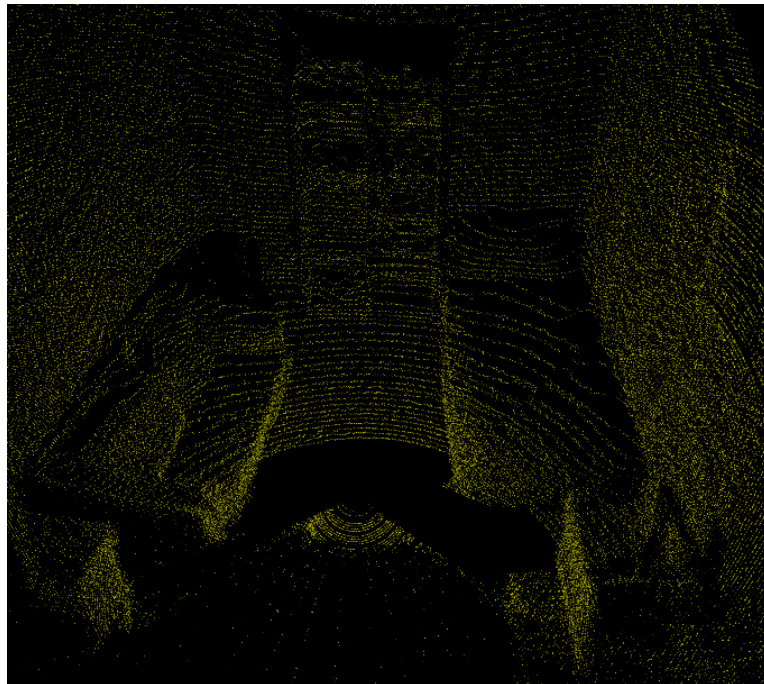


Figure C.1: Scan 4 Point Cloud (Side A)

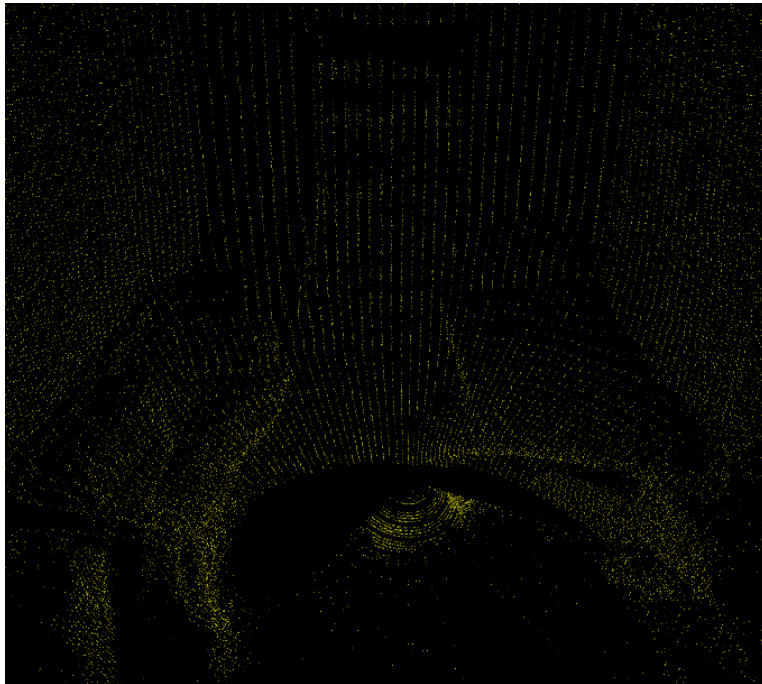


Figure C.2: Scan 5 Point Cloud (Side A)