Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE DII
Corso di Laurea Magistrale in Ingegneria dell'Energia Elettrica

Tesi di laurea magistrale

Advanced LiDAR Systems

Design of a LiDAR platform

Candidato:
Vittorio Giarola
Matricola 1152954

Relatore:
Prof. Arturo Lorenzoni

Correlatore:
Prof. Wei Wei

Anno Accademico 2017–2018

To my Family

Abstract

It's possible to say that one of the most interesting laser application is LiDAR. This system can measure the distance between a laser source and a target by hitting it with a pulsed light and then revealing the reflected ray with a suitable detector. Differences in laser return times and wavelengths can then be used to make digital 3D-representations of the target. This is the basic idea behind this complex and fascinating object. The willing of understanding better its way of functioning and exploiting modern applications in obstacles detection made me curious and at the same time excited, to work in this research field. The final aim of this work is to show that is possible to design and manufacture a LiDAR prototype that associate good performances and a low price; typically less than 250 dollars. Moreover, this work promises to develop an ecosystem able to show the LiDAR data in real time. So, in other words, an integrated environment.

Contents

List of Figures

List of Tables

Chapter 1

Project Development

Considering the fact that this project is strongly technology and product oriented we think that it should be better to follow a mind set based on project management. Considering that, it's important to walk on the well known trace of PM, putting a strong attention to the agile method. This consideration comes from the need of a flexible and constantly adjustable development strategy to deliver the project even considering time as a main driver of the process. These are the projects for which the goal is clearly defined but the solution is not. They are usually called Iterative and Adaptive life cycles. We hope that this short introduction has been sufficiently clear and able to explain the reason why we would like to proceed on that way.

1.1 Why Project Management

Considering the fact that we are actually developing a brand new product and we want to deliver something exploitable on the market, there's the need of a structure, a general way of thinking, which will help us to fight against misfortunes and at the same time manage unexpected events. One of the keys of the project is time, as said. Time is scarce and we need to optimize and refine its management. For all these reasons it's a good thing to set up a structured project thanks to which we can understand from the mistakes we commit. And then, for sure, it's even a good thing to learn something more about modern tools of project management, thinking about a possible future work experience.
To be honest, the first thing we have to do is to choose which kind of project management life cycle suits best our situation. Our attention fell on Adaptive models which are able to accommodate a high level of uncertainty and complexity. In that sense, they fill a void between the traditional Iterative and Extreme models: keeping in mind that solution discovery is still the focus of these models. Each iteration in the Adaptive models must address

not only task completion for newly defined functions and features but also further solution definition through function and feature discovery. It is the discovery part of the Adaptive PMLC[1] models that sets them apart from other PMLC models.[1] An Adaptive PMLC model consists of a number of phases that are repeated in cycles, with a feedback loop after each cycle is completed. Each cycle proceeds based on an incomplete and limited understanding of the solution. Each cycle learns from the preceding cycles and plans the next cycle in an attempt to converge on an acceptable solution. Adaptive PMLC model is missing both depth and breadth of the solution. Figure 1.1 depicts the Adaptive PMLC model for projects that meet the conditions of an incomplete solution due to missing features and functions. In the Adaptive PMLC model, as with other Agile approaches, the degree to which the solution is known might vary over a wide range from knowing a lot but not all. The less that is known about the solution, the more risk, uncertainty, and complexity will be present. To remove the uncertainty associated with these projects, the solution has to be discovered. That will happen through a continuous change process from cycle to cycle. That change process is designed to create a convergence to a complete solution. In the absence of that convergence, Adaptive projects are frequently cancelled and restarted in some other promising direction.

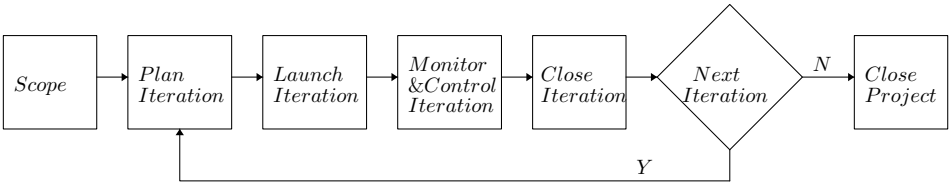


Figure 1.1: Adaptive PM process

1.2 The procedure

Abraham Lincoln was used to say: "Give me six hours to chop down a tree and I will spend the first four sharpening the axe." Since we like this mind set, we think it's important to put a big effort from the beginning of the project, trying to guess the possible evolution and what to expect from the process itself. Since the product is brand new, we want to experience something that probably we will struggle to govern.
Anyway, clearly speaking, defining the target of the project and roughly the time in which we have to deliver the final product it's probably the most important thing. Once we have established these things we can focus on the product itself with its requirements and performances. Only thanks

[1]PMLC states for project management life cycle.

to clear requirements we would be able to understand if we are on the right path, if we are doing well and if we are on time for the delivery. The following Fig. 1.2 is only a preliminary Requirement chart based on macroscopic and general requirements. In the next sections we will develop a new requirement chart listing the technical performances and features of the product. So if we are asked to summarize how to develop the project

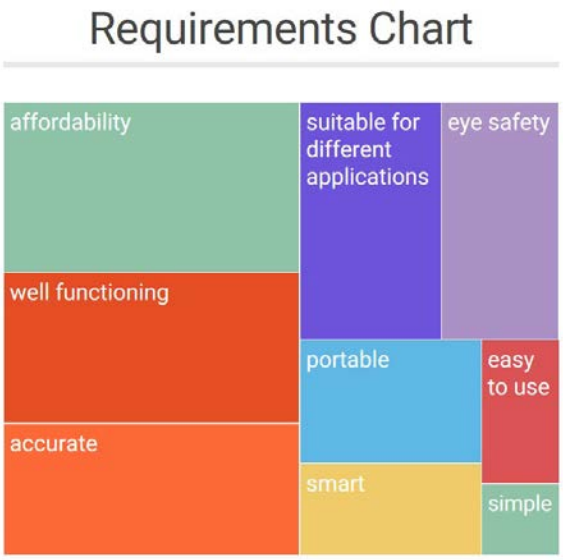


Figure 1.2: Table of requirements

pointing out all the steps and describing the macroscopic phases of if, we can easily reply explaining Fig. 1.3. In this picture are reported the milestones of the project. By the continuous check and monitoring of the status of the project, taking into account the successful engagement of the milestones we are able to understand how we are proceeding. In Fig. 1.4 is showed the timeline of the whole project. The fist step is related to the acquisition of the informations. At this stage is fundamental to study the LiDAR technology by itself and then make a research related to the prior art. Talking about this last topic, we will firstly focus our attention on the market products and secondly to the open sources projects. The open source community offers a huge variety of examples and related projects from which we can take some inspiration. Moreover, the DIY spirit fits perfectly our idea to keep low the cost of our platform. In the second stage it is expected to identify the components which are necessary to manufacture the LiDAR, at least in its first version. This version will be a learning trial version from which we expect to gather experience and understand the limits and the boundaries of the system. Once we have a precise idea of the equipment, it will be necessary to understand how to organize our work and necessarily

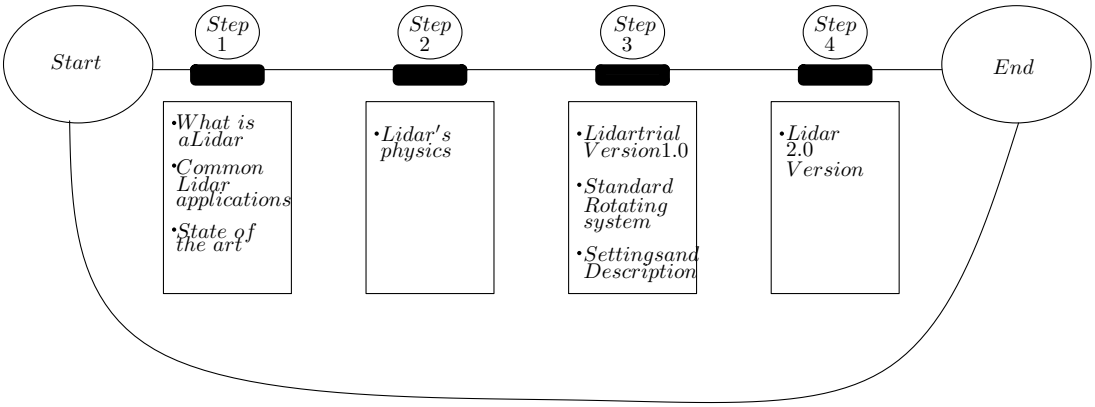


Figure 1.3: The process

undertake a training phase that will allow us to become at least confident in the use of software and hardware the project requires. It's extremely important to get familiar with the microcontroller architecture and IDE[2], how the motors work and how to control them, how to show the data in real time and write a java code that shows the results. Finally, it is important to have a precise idea of the integration process that is necessary to follow in order to put the parts together to deliver a robust and reliable LiDAR. Only then, the first version can be designed. In our case, the prototype is quite simple, just a turret. After writing the code and make the hardware work smoothly and checked the processing interface it's mandatory to test the accuracy and reliability of the equipment. The data analysis will provide a solid understanding on the validation of the process and the performances. Moreover, the key factor is to underline the limitations of the prototype, trying to figure out how to fix these problems and improve functionalities in the further version. Then the second version should be the final one: fixing the problems observed in the previous one and reaching the desired quality and accuracy standards.

[2]IDE stands for integrated development environment.

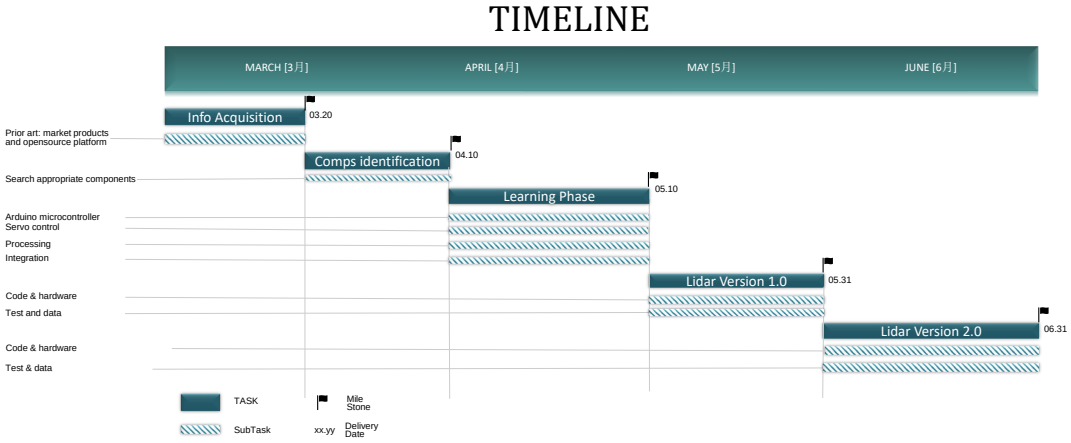


Figure 1.4: Timeline

Chapter 2

Introduction on LiDAR Systems

2.1 What is a LiDAR

From the origins of electronics we have always heard about RADAR. This devices is able to detect the position of objects which occupy the space thanks electromagnetic waves. Only in recent years the RADAR concept has been exploited in the optics field, by achieving an optical RADAR called LiDAR. In other words, RADAR systems emit radio waves and measure what bounces back, LiDAR uses light waves. It's a powerful data collection system that provides 3-D information for an area of interest or a project area. This device is versatile and suitable for different purposes. Especially in the last 5 or 10 years the scientific community and a lot of companies have been focused on this product trying to imagine possible applications in different fields of technology.
If we want to write on paper a more precise definition we can say that: LiDAR is a surveying method that measures distance to a target by illuminating it with a pulsed laser light, and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3D-representations of the target.[2]
To understand whether LiDAR might be appropriate for our agency's surveying project needs, we will need a thorough understanding of the accuracy and data requirements for the project at hand. By mapping out these critical components in advance we will be able to get a handle on the appropriate LiDAR solution, including the hardware, software, and ground control. Someone could ask us why? Because a thorough understanding of the type of project we are undertaking will help us determine the ground control of the project. For example, a corridor mapping project will be conducted differently from a county-wide collection. Or more, for example, a mobile-mapping collection is a whole different work from a terrestrial or airborne

collection. Lidar uses ultraviolet, visible, or near infrared light to image objects. It can target a wide range of materials, including non-metallic objects, rocks, rain, chemical compounds, aerosols, clouds and even single molecules.[3] But one thing we have to keep always in mind: LiDAR is pretty powerful, but it is not the solution to everything. It cannot "see" through walls or surfaces. It cannot peer through clouds; it is actually foiled by fog; and it cannot see if there is a dense smoke. Anyway, a narrow laser-beam can map physical features with very high resolutions. Wavelengths vary to suit the target: from about 10 micrometers to the UV (approximately 250 nm).[3] Typically light is reflected via backscattering, as opposed to pure reflection one might find with a mirror. Different types of scattering are used for different LiDAR applications: most commonly Rayleigh scattering.[4] The basic LiDAR scheme is the following (see Fig. 2.1): This scheme is pre-

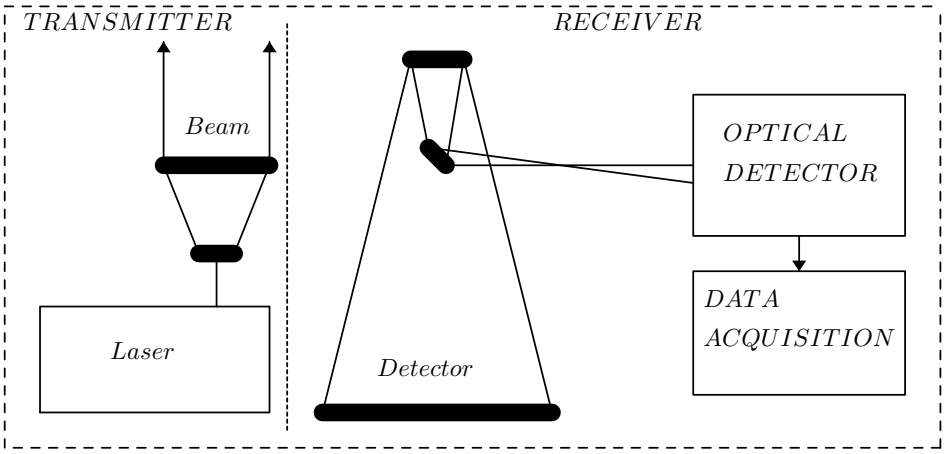


Figure 2.1: LiDAR, working principle

sented only to give a general idea of the concept behind LiDAR technology. We can recognise:

- *Laser*: 600–1000 nm lasers are most common for non-scientific applications. They are inexpensive, but since they can be focused and easily absorbed by the eye, the maximum power is limited by the need to make them eye-safe. Eye-safety is often a requirement for most applications. A common alternative, 1550 nm lasers, are eye-safe at much higher power levels since this wavelength is not focused by the eye. Laser settings include the laser repetition rate (which controls the data collection speed). Pulse length is generally an attribute of the laser cavity length, the number of passes required through the gain material, and pulsing speed. Better target resolution is achieved with shorter pulses, provided the LiDAR receiver detectors and electronics have sufficient bandwidth;[4]

- *Scanner and optics*: How fast images can be developed is also affected by the speed at which they are scanned. There are several options to scan the azimuth and elevation, including dual oscillating plane mirrors, a combination with a polygon mirror and a dual axis scanner. Optic choices affect the angular resolution and range that can be detected. A hole mirror or a beam splitter are options to collect a return signal;

- *Photo-detector and receiver electronics*: Two main photo-detector technologies are used in Lidar. The first is solid state photo-detectors, such as silicon avalanche photo-diodes, or photo-multipliers. The sensitivity of the receiver is another parameter that has to be balanced in a Lidar system design;

- *Position and navigation systems*: LiDAR sensors that are mounted on mobile platforms such as airplanes or satellites require instrumentation to determine the absolute position and orientation of the sensor. Such devices generally include a Global Positioning System receiver and an Inertial Measurement Unit (IMU).

Talking about traditional and conventional LiDAR is good to get familiar with some specific language which can help to understand better what just said:

- *Repetition Rate*: this is the rate at which the laser is pulsing, and it is measured in kilohertz (KHz). Laser emits extremely quick pulses, so it's important to check that all the equipments, such as trans-receiver and receiver are able to operate at the maximum laser pulsing speed;

- *Scan frequency*: while the laser is pulsing, the scanner is oscillating, or moving back and forth. The scan frequency will tell how fast the scanner is oscillating. A mobile system has a scanner that rotates continuously in a 360 degree fashion, but most airborne scanners move back and forth.

- *Scan angle*: this is measured in degrees and is the distance that the scanner moves from one end to the other. We will adjust the angle depending on the application and the accuracy of the desired data product;

- *Flying attitude*: it is not a surprise that the farther the platform is from the target, the lower the accuracy of the data and the less dense the points will be that define the target area. That's why for airborne systems, the flying attitude is so important. This aspect in our case it's less relevant since we are interested in short range applications;

- *Nominal point spacing (NPS)*: The rule is simple enough the more points that are hit in the collection, the better we will define the targets. The point sample spacing varies depending on the application. It's also good to keep in mind that LiDAR systems are random sampling systems. Although it's not possible to determine exactly where the points are going to hit on the target area. The only thing that is possible to do is that we can decide how many times the target areas are going to be hit, so for example we can choose a higher frequency of points to better define the targets.

- *Cross track resolution*: this is the spacing of the pulses from the LiDAR system in the scanning direction, or perpendicular to the direction that the platform is moving;

- *Swath*: this is the actual distance of the area of coverage for the LiDAR system. It can vary depending on the scan angle. Mobile LiDAR has a swath, too, but it is usually fixed and depends on the particular sensor. For these systems, though, you might not hear the word "swath" it may instead be referred to as the "area of coverage" and will vary depending on the repetition rate of the sensor;

- *Overlap*: it's the amount of redundant area that is covered between flight lines or swaths within an area of interest. Overlap is not a wasted effort, because sometimes it provides more accuracy.[3]

So, the accuracy of the system depends not only on laser's feature but even on the proprieties of the revelation and the system in general. Here we have to be honest and add some more informations. Actually, during the development of this work, we are basically interested in mechanical scanning systems. Even though mechanical rotational detection systems have some limits and disadvantages, they're quite easy to implement and develop. One of the major drawbacks is the fact that the scanning speed is a limit to the performances of the LiDAR and at the same time reduces the rotating components life, causing fatalities and premature failure. As with just about any technology that requires precision you cannot count on LiDAR collection without proper sensor calibration. All sensors should be calibrated routinely, and calibration is recommended after every installation into a platform. How you calibrate depends on the system and the LiDAR provider, but the results are similar. Terrestrial LiDAR scanners, however, have self calibrations that take place before every collection. In addition, all LiDAR systems (whether in the air or on the ground) are initially calibrated by the manufacturer in a lab and during field trials. Whenever a hardware component is repaired or replaced on a LiDAR sensor head, the system should be recalibrated. It is also important to remember that every further modification and adjustment of the system brings to a revision and check up status of the new configuration.

2.2 Lidar Applications

LiDAR originated in the early 1960s, shortly after the invention of the laser, and combined laser-focused imaging with the ability to calculate distances by measuring the time for a signal to return using appropriate sensors and data acquisition electronics. Its first applications came in meteorology, where the National Center for Atmospheric Research used it to measure clouds. The general public became aware of the accuracy and usefulness of LiDAR systems in 1971 during the Apollo 15 mission, when astronauts used a laser altimeter to map the surface of the moon. At the very beginning LiDAR system were predominantly used in high-resolution maps, with applications in geodesy, geomatics, archaeology, geography, geology, geomorphology, seismology, foretry, atmospheric physics, laser guidance, airborne laser swath mapping (ALSM), and laser altimetry. All those applications are associated with a high density and high power of the laser beam. Nowadays, conversely, we are seeking applications based on low laser beam power and eye compatible and safety. By considering that it's possible to exploit Lidar on short range, making possible 3D object scanning and object detection and avoiding collision systems. Actually, we are strongly interested in these last applications and solutions. In the following part we will discuss about the different general applications and then in the next section we will pay attention only to industrial and commercial applications in obstacle detection systems.

2.2.1 Airborn LiDAR

While attached to a plane during flight, creates a 3D point cloud model of the landscape. This is currently the most detailed and accurate method of creating digital elevation models, replacing photogrammetry. One major advantage in comparison with photogrammetry is the ability to filter out reflections from vegetation from the point cloud model to create a digital surface model which represents ground surfaces such as rivers, paths, cultural heritage sites, etc., which are concealed by trees. Within the category of airborne LiDAR, there is sometimes a distinction made between high-altitude and low-altitude applications, but the main difference is a reduction in both accuracy and point density of data acquired at higher altitudes. Airborne LiDAR can also be used to create bathymetric models in shallow water.[5] The main constituents of airborne LiDAR include digital elevation models (DEM) and digital surface models (DSM). The points and ground points are the vectors of discrete points while DEM and DSM are interpolated raster grids of discrete points. The process also involves capturing of digital aerial photographs. In order to interpret deep seated landslides for example, under the cover of vegetation, scarps, tension cracks or tipped trees air borne LiDAR is used. Air borne LiDAR digital elevation models can see through

the canopy of forest cover, perform detailed measurements of scarps, erosion and tilting of electric poles.[6] Airborne lidar data is processed using a toolbox called Toolbox for Lidar Data Filtering and Forest Studies (TIFFS) for lidar data filtering and terrain study software. The data is interpolated to digital terrain models using the software. The laser is directed at the region to be mapped and each point's height above the ground is calculated by subtracting the original z-coordinate from the corresponding digital terrain model elevation. Based on this height above the ground the non-vegetation data is obtained which may include objects such as buildings, electric power lines, flying birds etc. The rest of the points are treated as vegetation and used for modeling and mapping. Within each of these plots, lidar metrics are calculated by calculating statistics such as mean, standard deviation, skewness, percentiles, quadratic mean etc. ;

2.2.2 Terrestrial LiDAR

also terrestrial laser scanning happen on the Earth's surface and can be both stationary or mobile. Stationary terrestrial scanning is most common as a survey method, for example in conventional topography, monitoring, cultural heritage documentation and forensics.[7] The 3D point clouds acquired from these types of scanners can be matched with digital images taken of the scanned area from the scanner's location to create realistic looking 3D models in a relatively short time when compared to other technologies. Each point in the point cloud is given the colour of the pixel from the image taken located at the same angle as the laser beam that created the point. Mobile LiDAR (also mobile laser scanning) is when two or more scanners are attached to a moving vehicle to collect data along a path. These scanners are almost always paired with other kinds of equipment, including GNSS receivers and IMUs. One example application is surveying streets, where power lines, exact bridge heights, bordering trees, etc. all need to be taken into account. Instead of collecting each of these measurements individually in the field with a tachymeter, a 3D model from a point cloud can be created where all of the measurements needed can be made, depending on the quality of the data collected. This eliminates the problem of forgetting to take a measurement, so long as the model is available, reliable and has an appropriate level of accuracy. Terrestrial LiDAR mapping involves a process of occupancy grid map generation. The process involves an array of cells divided into grids which employs a process to store the height values when LiDAR data falls into the respective grid cell. A binary map is then created by applying a particular threshold to the cell values for further processing. The next step is to process the radial distance and z-coordinates from each scan to identify which 3D points correspond to each of the specified grid cell leading to the process of data formation.[8]

2.2.3 Autonomous vehicles

From the official NTSB's (National Transportation Security Board - US) website we can actually see that an average of 38000 [12] people per year loose their life on the american roads due to car accidents and fatalities. To put that in perspective, that's more than a Boeing 737 falling out of the sky five times per week in a year. Could we accept something similar in the aviation? Probably the answer is no. So, this is the main reason why LiDAR technology has to be exploited in autonomous driving car; but safety is not the unique reason. The second theme is related to health and money: taking into account the average commuter in America, which is around 50 minutes per worker per day, you multiply by the number of daily workers, 120 million, you get 6 billion minutes per day lost by the american workers sat in traffic. If you compare this time to the the average life expectancy, around 70 years, you get 162 lifetimes waisted in the traffic jam per day. An other time, this is not social acceptable. And last but not least important, there's a reason related to diversity and inclusion: self driving car could help to improve the mobility of disable people.
Moreover, as per the latest World Health Organization reports, road traffic accidents account for 1.25 million deaths each year. In fact, according to an estimate, by the year 2030 road traffic accidents will be the 7^{th} major cause of death around the world. It is in this context that the UN's 2030 Agenda for Sustainable Development becomes even more important. A target has been set to reduce the number of traffic accident deaths by 50% by 2020. In light of these statistics, it is clear to see why the use of LiDAR technology is critical. Unlike other types of sensors or telematics currently available, LiDAR has improved capabilities when it comes to the detection of objects, even in cases where there is a complete absence of light. LiDAR's features, which include a blind spot monitoring system, adaptive cruise control, along with a collision avoidance system and pedestrian protection system, are not only better than the features of other sensors, but are also far more consistent and reliable. Analysts at Technavio predict that the global automotive LiDAR sensors market will see a CAGR of more than 34% by 2020.
May use LiDAR for obstacle detection and avoidance to navigate safely through environments, using rotating laser beams. Cost map or point cloud outputs from the LiDAR sensor provide the necessary data for robot software to determine where potential obstacles exist in the environment and where the robot is in relation to those potential obstacles. Examples of companies that produce LiDAR sensors commonly used in robotics or vehicle automation are Sick and Hokuyo. Examples of obstacle detection and avoidance products that leverage LiDAR sensors are the Autonomous Solution, Inc. Forecast 3D Laser System and Velodyne HDL-64E. This is actually the world in which we want to dive deeply. So in the next section we will look at the products now available on the market, painting the "affresco" of the

so called prior art. The very first generations of automotive adaptive cruise control systems used only LiDAR sensors and it is expected a more intense use of these technologies in the next future.

2.2.4 Urban Planning

in this growing phase of urbanization and industrialization there is an emergent need of proper town planning systems. The 3D city models in urban areas are essential for many applications, such as military operations, disaster management, mapping of buildings and their heights, simulation of new buildings, updating and keeping cadastral data, change detection and virtual reality. Urban, city, or town planning is the discipline of land use planning which explores several aspects of the built and social environments of municipalities and communities. The urban areas in the developing world are under constant pressure of a growing population.[9] Efficient urban information system is a vital pre-requisite for planned development. It is thought that LiDAR application to urbanization plans will bring to a more sustainable and efficient growth of a city. In traffic line design area, LiDAR technology offers a high precision DEM (Digital Elevation Model) for road and railway design, to facilitate route design and the calculation of earth work quantity. Besides, it is also of great application significance in route design of communication network, oil pipe, air pipe. In digital city project, a "digital city" system can be established and perfected by combining the high-precision DEM and GPS of LiDAR, and the data can be updated real time therewith.[11] It is expected that LiDAR systems will be massively used to plan the development of the cities of the future, to integrate new infrastructure in existing contexts, taking into account sustainability and real constraints due to environment, population and other key factors. In other words, the rational urban densification and optimization, as well as the development and construction of new buildings, streets, subway and tram lines and innovative means of transportation is strictly related to the implementation of this technology.

2.2.5 Energy

combining LiDAR technology and GIS (Geographic Information System) is possible to develop a method for predicting city-wide electricity gains from photovoltaic panels based on detailed 3D urban massing models combined with Daysim-based hourly irradiation simulations, typical meteorological year climactic data and hourly calculated rooftop temperatures. The resulting data can be combined with online mapping technologies and search engines as well as a financial module that provides building owners interested in installing a photovoltaic system on their rooftop with meaningful data regarding spatial placement and informations.[10] So, in this sense Li-

DAR could help the diffusion of RES (Renewable energy sources) in a more efficient and rational way. This is only a simple example of what is possible to conceive in the energy world thanks to LiDAR technology. An other important and relevant example is related to power transmission lines status monitoring. In electric power transmission area, it can make for the laying, maintenance and management of power grid. In the link of electric power line design, the terrains and geographic features within the entire line design area can be learned about based on LiDAR data. Especially in areas with thick trees, the area and amount of trees to be cut down can be estimated. In the link of power lines repair and maintenance, the height of line at any location can be measured and calculated based on the LiDAR data points on the electric power line and the elevation of corresponding exposed spot on the ground, which can facilitate repair and maintenance.[11] Not only Other relevant applications are missing for a time issues such as agriculture, archeology or biology. Who is writing doesn't want to explain in detail these applications.

2.3 State of the art LiDAR scanning Technology

So, as said, this section is dedicated to the prior art. In other words the understanding of the most important players on the LiDAR market at the moment, also called *incumbent*, and a focus on the prominent players or *new comers*.

- The world's fourth largest tires manufacturer, Continental AG is a leading German automotive manufacturing specialized in braking systems, automotive safety, tachographs etc. In 2016 was confirmed the acquisition of the Hi-Res 3D Flash LiDAR business from Advanced Scientific Concepts, Inc. (ASC) based in Santa Barbara, California. This technology has further enhanced the company's Advanced Driver Assistance Systems product portfolio with a future-orientated solution to add to the group of surrounding sensors needed to achieve highly and fully automated driving. Nowadays Continental AG is one of the most relevant players on the LiDAR market. A complete 3D model of the vehicle surroundings nearby or over 200 meters away and as close as few centimetres, is constructed in just 1.32 microseconds, 30 times per second. In addition, the distance to individual objects is also accurately measured. The sensor is completely solid state, so no rotating parts. The low complexity and high industrial feasibility mean that it's possible to efficiently install multiple sensors all around the vehicle, thereby enabling the complete generation, real-time, 360 degrees images of the vehicle surroundings. The technology, has already been deployed for space operations, provides a significantly more comprehensive and detailed 3D view of the entire vehicle surroundings –

both during the day and at night – and works reliably even in adverse weather conditions. Mass production will start at the end of 2019 and the product will be on the market in the early 2020, as scheduled. The price is not known yet, but rumors say around 7000 dollars (even though we don't know if this could be a realistic number).

Figure 2.2: 3D flash LIDAR by Continental AG [13]

- Cosworth, the leading designer and manufacturer of high performance engines and end-to-end automotive technologies, demonstrated a prototype of solid state LiDAR sensor technology as part of the official opening of its new state-of-the-art manufacturing center and North American Headquarters in Shelby Township, last June.[14] Media and invited guests witnessed a demonstration of a solid state LiDAR technology. Having no moving parts, the technology is designed and built to provide stability that will meet the stringent automotive standards for reliability. At the grand opening of North American manufacturing center they demonstrated the stability, precision, distance and field of view that are necessary to make autonomous or assisted driving safe. With a minimum of 3,200 lasers, a clear 120-degree filed of view and 200 meter range, it is believed that this solution could be a game changer for automotive sensor technology. Right now we have no more information related to the product, large scale industrial production and especially the price the LiDAR will be sold on the market.[14] Anyway, since the author of this work consider that manufacturer and product interesting it has been reported here. More information in the following months.

Description	Value
Wavelenght	830 nm
Maximum pulse energy	5 nJ
Pulse Duration	4 ns
Pulse repetition frequency	2.16 MHz
Beam divergence (FWHM, full angle)	0.4 mrad
Mirror rotation	30 Hz
Base rotation	2.5 mHz

Table 2.1: BLK360 Laser Specs

- In this section is presented one of the LiDAR based products which I believe is extremely interesting since its aim is similar to the one that we have developed (later in this work we will talk about that extensively). In detail we are talking about LEICA BLK360 which is used to digitalize interiors and recreate the respective 3D representation.
  The Leica BLK360 captures the world around you with full-colour panoramic images overlaid on a high-accuracy point cloud. Simple to use with just the single push of one button, the BLK360 is the smallest and lightest of its kind. Anyone who can operate an iPad can now capture the world around them with high resolution 3D panoramic images. Using the ReCap Pro mobile app, the BLK360 streams image and point cloud data to iPad. The app filters and registers scan data in real time. After capture, ReCap Pro enables point cloud data transfer to a number of CAD, BIM, VR and AR applications. The integration of BLK360 and Autodesk software dramatically streamlines the reality capture process thereby opening this technology to non-surveying individuals. In few points:
  - Allows you to scan in high, standard and fast resolutions;
  - Weighs 1kg / Size 165 mm tall x 100 mm diameter;
  - Less than 3 minutes for full-dome scan (in standard resolution) and 150 MP spherical image generation;
  - 360,000 laser scan pts/se
  - HDR and thermal imaging

  Table 2.1 reports BLK360 Laser propriety.

The laser incorporated in the product produces an invisible beam which emerges from the rotating mirror. The laser product described in this section is classified as laser class 1 in accordance with IEC prescriptions. This

Figure 2.3: Leica BLK360

products is safe under reasonably foreseeable conditions of operation. The price for this product is quite prohibitive, around 16000$. Leica BLK360 can offer amazing performances and results but the price it's a great obstacle to its mainstream diffusion.

So, this is a brief presentation of the products I believe are more promising and interesting on the market. As I have said before the one we are more interested in is Leica BLK360, because as specifications and characteristics of the products is the closest to the LiDAR platform we have developed. In fact we focused our attention on interior scanning and 3D environment reconstruction. For sure our platform's performance will be not comparable with an industrial product which costs several thousands of dollars. Just to be clear our budget is around 250$. Anyway, we consider the low budget a positive aspect, in fact we want to show that is possible to get a good result (high density point cloud) even though a less powerful hardware is used. In this case it will be necessary to optimize and deploy at its best the hardware we have in our hands. One of the biggest differences between our version and Leica's version will be for sure the acquisition speed: actually, it is said that the commercial product is able to get a 360 degrees representation of the environment roughly in 3 minutes. Probably, our prototype could scan the environment for sure at a slower speed, maybe 6 times slower. We believe, though, is still ok, considering that time is not the principal driver for our project.

2.4 LiDAR Project Scope

In this section a deeper focus on the description of the LiDAR platform's scope is given to the reader. A partial portrait or canvas has been presented in §Chap. 1, §Sec. 1.2. Right now, we want to underline that the prototype under the magnifying glass will provide:

- high resolution & high density Point Cloud;
- to each point will be assigned a colour, considering the distance from the sensor;
- a navigation system based on Java which will allow the user to see in real time the scanning result and translate and rotate the Point Cloud.

As we have already mentioned, at the moment the scanning speed it's not a big issue. At first our attention will be totally focused on getting a high density and high resolution Point Cloud. Only, then we will change our focus on getting a higher scanning speed. The development of the final version is a complex problem, achievable only through incremental steps. First of all, it's necessary to get familiar with the concept of LiDAR systems, how do they work, their specifications, the safety rules needed to operate these systems. Then we will move towards the first prototype which is a simple LiDAR platform, more similar to a turret able to roughly scan the 3D space which surrounds us, without any pretension to be accurate or detailed. The aim of this first step is to get and accumulate experience, show to be familiar with MCU control and writing algorithm able to execute what is intended. In this first operative phase is necessary to understand that even a small project is constitute by different sub-projects and it's highly recommended to schedule precisely the different parts and then assemble meticulously the whole system solving in a smart way the integration problems. Only then it will be possible to extensively develop the more advance and final version. Starting from the limits and the problems emerged during phase 1, it will be possible to study and conceive engineering solutions able to push forward the performances of the machine and eventually get a better, more realistic and industrial exploitable result.
Last but not the least during the whole process in essential to test small parts of the platform in the Lab in order to validate components specifications, new possible solutions or the real boundaries of the platform itself. So the engineering development is strictly linked to the test phase and the pragmatic approach to problems.
In the next Chapter will be present the basic knowledge, necessary to understand how the following prototypes will work. We will pay attention to details and protocols, as for example how the data transmission between sensor and MCU works and other important topics. Eventually, then, it will be possible to present how the LiDAR platforms has been made.

Chapter 3

Fundamentals on LiDAR

As we have said in the previous sections, especially the realization of this first platform is essential to get confident with the equipment, and in particular study more deeply the interaction between hardware and software. The language necessary to control the MCU or better the micro controller that we'll use constitutes a completely new barrier. So, for this first period it's necessary to study and practice what's written on books, web manuals and tutorials.
In this Chapter will be reported and explained basic principles and how things work; in particular:

- Sensor;
- Micro-Controller (Arduino);
- Servo Motors;
- Processing v3;

3.1 The Sensor

When space and weight requirements are tight, the LiDAR-Lite v3 soars. It's the ideal compact, high-performance optical distant measurement sensor solution for drone, robot or unmanned vehicle applications. Using a single chip signal processing solution along with minimal hardware, this highly configurable sensor can be used as the basic building block for applications where small size, light weight, low power consumption and high performance are important factors. Featuring all of the core features of the popular LiDAR-Lite v2, this easy-to-use 40 meter laser-based sensor uses about 130 milliamps during an acquisition. It is user-configurable so you can adjust between accuracy, operating range and measurement time. It's reliable, powerful ranging and it's a versatile proximity sensor. This is LIDAR-Lite v3, a compact optical distance measurement sensor from Garmin. Each

Specifications	Measurements
Size (l x w x h)	20 x 48 x 40 mm
Weight	22 g
Operating Temperature	-20 to 60 C

Table 3.1: Phisical Propriety Garmin Lidar Lite v3

Specifications	Measurements
Power	5 Vdc nominal 4.5 Vdc min., 5.5 Vdc max.
Current	105 mA idle 135 mA continuous operations

Table 3.2: Electrical Propriety Garmin Lidar Lite v3

LiDAR-Lite v3 features an edge-emitting, 905nm (1.3 watts), single-stripe laser transmitter, 4 m Radian x 2 m Radian beam divergence, and an optical aperture of 12.5mm. The third version of the LiDAR-Lite still operates at 5V DC with a current consumption rate of <100mA at continuous operation. On top of everything else, it can be interfaced via I2C or PWM with the included 200mm accessory cable.
Note: CLASS 1 LASER PRODUCT CLASSIFIED EN/IEC 60825-1 2014. This product is in conformity with performance standards for laser products under 21 CFR 1040, except with respect to those characteristics authorized by Variance Number FDA-2016-V-2943 effective September 27, 2016.

3.1.1 Technology and Operations

This device measures distance by calculating the time delay between the transmission of a Near Infra-red laser and its reception after reflecting off a target. This translates into distance using the known speed of light. The unique signal processing approach transmits a coded signature and looks for that signature in the return, which allows for highly reflective detection with eye-safe laser power levels. Proprietary signal processing techniques are used to achieve high sensitivity, speed and accuracy in a small, low power and low-cost system.
To take a measurement, this device first performs a receiver bias correction routine, correcting for changing ambient light levels and allowing maximum sensitivity. Then the device sends a reference signal directly from the trans-

mitter to the receiver. It stores the transmit signature, sets the time delay for "zero" distance, and recalculates this delay periodically after several measurements. Next, the device initiates a measurements by performing a series of acquisitions. Each acquisition is a transmission of the main laser signal while recording the return signal at the receiver. If there is a signal match, the result is stored in memory as a correlation record. The next acquisition is summed with the previous result. When an object at a certain distance reflects the laser signal back to the device, these repeated acquisitions cause a peak to emerge, out of the noise, at the corresponding distance location in the correlation record.
The device integrates acquisitions until the signal peak in the correlation record reaches a maximum value. If the returned signal is not strong enough for this to occur, the device stops at the predetermined maximum acquisition count. Signal strength is calculated from the magnitude of the signal record peak and a valid signal threshold is calculated from the noise floor. If the peak is above the threshold the measurements is considered valid and the device will calculate the distance, otherwise it will report 1 cm. When beginning the next measurement, the device clears the signal and starts the sequence again.[15]

3.1.2 Interface

Initialization On power-up or reset, the device performs a self-test sequence and initializes all registers with default values. After roughly 22 ms distance measurements can be taken with the I2C interface or the Mode Control Pin. So before explaining how LiDAR Lite v3's I2C works it's better to give to the reader some basic and general informations related to I2C itself. Only then it will be possible to understand clearly how the LiDAR data transmission works.[15]

I2C description I2C-bus compatible ICs don't only assist designers, they also give a wide range of benefits to equipment manufacturers because:

- The simple 2-wire serial I2C-bus minimizes interconnections so ICs have fewer pins and there are not so many PCB tracks; result - smaller and less expensive PCBs;

- The completely integrated I2C-bus protocol eliminates the need for address decoders and other 'glue logic';

- The multi-master capability of the I2C-bus allows rapid testing and alignment of end-user equipment via external connections to an assembly-line;

- The availability of I2C-bus compatible ICs in SO (small outline), VSO (very small outline) as well as DIL packages reduces space requirements even more.

These are just some of the benefits. In addition, I2C-bus compatible ICs increase system design flexibility by allowing simple construction of equipment variants and easy upgrading to keep designs up-to-date. In this way, an entire family of equipment can be developed around a basic model. Upgrades for new equipment, or enhanced-feature models (i.e. extended memory, remote control, etc.) can then be produced simply by clipping the appropriate ICs onto the bus. If a larger ROM is needed, it's simply a matter of selecting a micro-controller with a larger ROM. As new ICs supersede older ones, it's easy to add new features to equipment or to increase its performance by simply unclipping the outdated IC from the bus and clipping on its successor.
For 8-bit oriented digital control applications, such as those requiring microcontrollers, certain design criteria can be established:

- A complete system usually consists of at least one microcontroller and other peripheral devices such as memories and I/O expanders;
- The cost of connecting the various devices within the system must be minimized;
- A system that performs a control function doesn't require high-speed data transfer;
- Overall efficiency depends on the devices chosen and the nature of the interconnecting bus structure.

To produce a system to satisfy these criteria, a serial bus structure is needed. Although serial buses don't have the throughput capability of parallel buses, they do require less wiring and fewer IC connecting pins. However, a bus is not merely an interconnecting wire, it embodies all the formats and procedures for communication within the system. Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss and blockage of information. Fast devices must be able to communicate with slow devices. The system must not be dependent on the devices connected to it, otherwise modifications or improvements would be impossible. A procedure has also to be devised to decide which device will be in control of the bus and when. And, if different devices with different clock speeds are connected to the bus, the bus clock source must be defined. All these criteria are involved in the specification of the I2C-bus.
The I2C-bus supports any IC fabrication process (NMOS, CMOS, bipolar). Two wires, serial data (SDA) and serial clock (SCL), carry information

between the devices connected to the bus. Each device is recognized by a unique address (whether it's a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device. Obviously an LCD driver is only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.[16]

I2C Lidar Lite Interface This device has a 2-wire, I2C-compatible serial interface. It can be connected to an I2C bus as a slave device, under the control of an I2C master device. It supports 400 kHz Fast Mode data transfer. The I2C bus operates internally at 3.3 Vdc. An internal level shifter allows the bus to run at the maximum of 5 Vdc. Internal 3k ohm pull-up resistors ensure this functionality and allow for simple connection to the I2C host. The device has a 7-bit slave address with the default value of 0x62. The effective 8 bit-bit I2C address is 0x64 write and 0xC5. The device will not respond to a general call. Support is not provided for 10-bit addressing. The sensor module has a 7-bit slave address with a default value of 0x62 in hexadecimal notation. The effective 8 bit I2C address is: 0x64 write, 0x65 read. The device will not presently respond to a general call. Please note some additional informations:

- This device does not work with repeated START conditions. It must first receive a STOP condition before a new START condition;
- The Ack and NACK item are responses from the master device to the slave device;
- The last NACK in the read is technically optional, but the formal I2C protocol states that the master shall not acknowledge the last byte.

So, the I2C serial bus protocol operates as follow:

- The master initiates data transfer by establishing a start condition, which is when a high-to-low transition on the SDA line occurs while SCL is high. The following byte is the address byte, which consists of the 7-bit slave address followed by the read/write bit with zero state indicating a write request. A write operation is used as the initial stage of both read and write transfer. If the slave address corresponds to the module's address the unit responds by pulling SDA low during the ninth clock pulse (this is termed the acknowledge bit). At this stage, all other devices on the bus remain idle while the selected device waits for data to be written to or read from its shift requester;

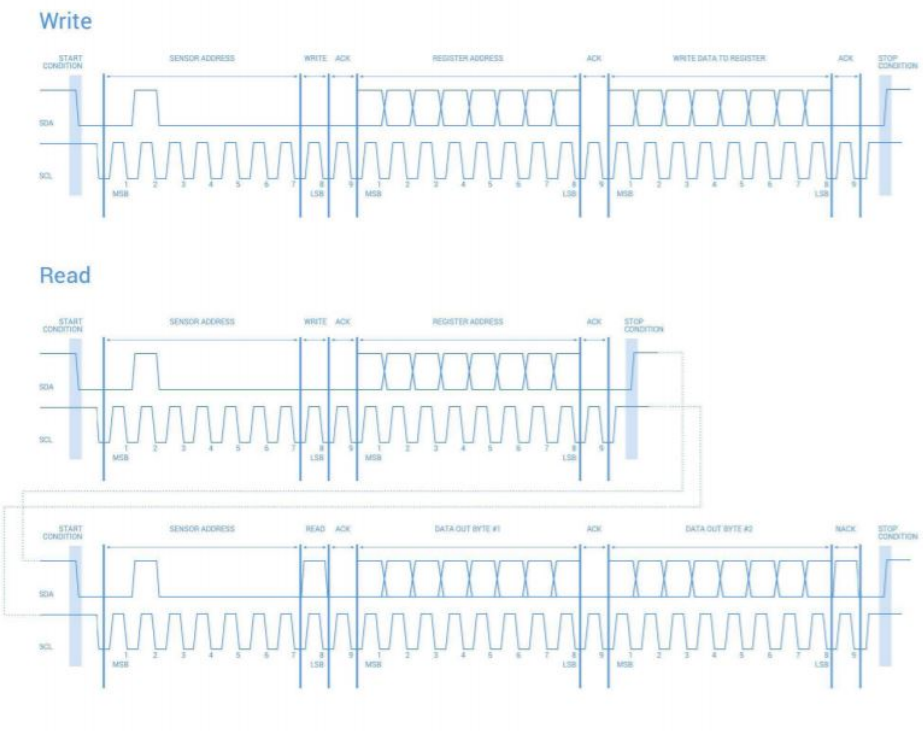


Figure 3.1: LiDAR Lite I2C read/write operations

- Data is transmitted over the serial bus in sequences of ninth clock pulses (eight data bits followed by an acknowledge bit). The transition on the SDA line must occur during the low period of SCL and remain stable during the high period of SCL;
- An 8 bit data byte following the address loads the I2C control the register with the address of the control register to be read along with flags indicating if auto increment of the addressed control register is desired with successive reads or writes; and if access to the internal micro or external correlation process register space is requested. Bit locations 5:0 contain the control register address while bit 7 enables the automatic increment of control register with successive data blocks. Bit position 6 selects correlation memory external to the micro-controller if set. (Presently an advanced feature);
- If a read operation is requested, a stop bit is issued by the master at the completion of the first data frame followed by the initiation of a new start condition, slave address with the read bit set (one state). The new address byte is followed by the reading of the one more data bytes succession. After the slave has acknowledged receipt of a valid address, data read operations proceed by the master releasing the I2C

Specifications	Measurements
Range (70% reflective Target)	40 m
Resolution	± 1 cm
Accuracy <5 m	± 2.5 cm
	± 10 cm
	Mean
Accuracy ≥ 5 m	± 1% of maximum distance
	Ripple
	± 1% of maximum distance
	270 Hz typical
Update Rate (70% reflective Target)	650 Hz fast mode
	>1000 Hz Short range only
Repetition Rate	50 Hz default
	500 Hz max

Table 3.3: Garmin Lidar Lite v3 Performances

Specifications	Measurements
Wavelength	905 nm (nominal)
Total Laser Power (peak)	1.3 W
Mode of operation	Pulsed (256 pulsed max. pulse train)
Pulse width	0.5 μs (50% duty cycle)
Pulse train repetition frequency	10-20 kHz (nominal)
Energy per pulse	< 280 nJ
Beam Diameter at laser aperture	12 x 2 mm
Divergence	8 m Radian

Table 3.4: Laser Propriety Garmin Lidar Lite v3

data line continuously clocking SCL. At the completion of the receipt of a data byte, the master must strobe the acknowledge bit before continuing the read cycle;

- For a write operation to proceed, Step 3 is followed by one or more 8 bit data blocks with acknowledges provided by the slave at the completion of each successful transfer. At the completion of the transfer cycle a stop condition is issued by the master terminating operation.[15]

3.2 Arduino

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Table 3.5: Arduino MCU specs

analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. It's possible to use Arduino UNO without worring too much about doing something wrong, worst case scenario it's needed to replace the chip for a few dollars and start over again. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; The Arduino Uno has a resettable polyfuse that protects computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

3.2.1 Programming

Arduino Uno schematics is open-source hardware. The ATmega328 comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then rese ing the 8U2;
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode;

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.
And then it's possible to update and upload the code via USB thanks to the appropriate software. It important to say that the arduino code is quite simple and intuitive. To have an idea and understand how the basic Arduino IDE environment is, please take a look at Fig. 3.2.

3.2.2 Connection Between Arduino and the Sensor

This example shows how to initialize, configure, and read distance from a LIDAR-Lite connected over the I2C interface. Connections:

- LIDAR-Lite 5 Vdc (red) to Arduino 5v
- LIDAR-Lite I2C SCL (green) to Arduino SCL
- LIDAR-Lite I2C SDA (blue) to Arduino SDA
- LIDAR-Lite Ground (black) to Arduino GND

Somebody could raise the end and ask why it's highly recommended to put a capacitor between the ground and 5V pin. The answer is quite immediate and easy: Capacitor is recommended to mitigate inrush current when device is enabled and, as said before, and as it's possible to see in Fig. 3.3
- 680uF capacitor (+) to Arduino 5v;
- 680uF capacitor (-) to Arduino GND;

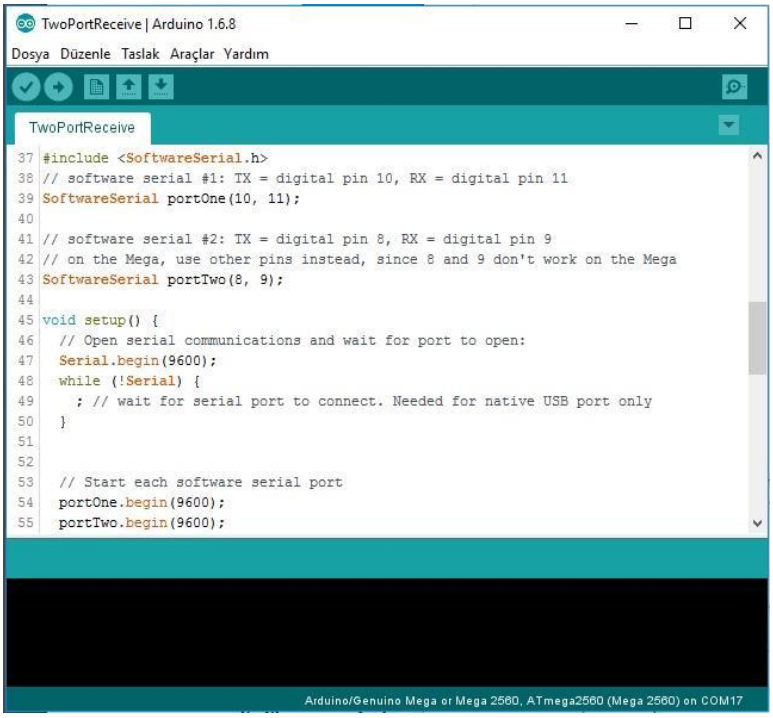


Figure 3.2: Arduino IDE

3.2.3 Power

The Arduino Uno board can be powered via USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector. The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The power pins are as follows (recap):

- Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin;

- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack

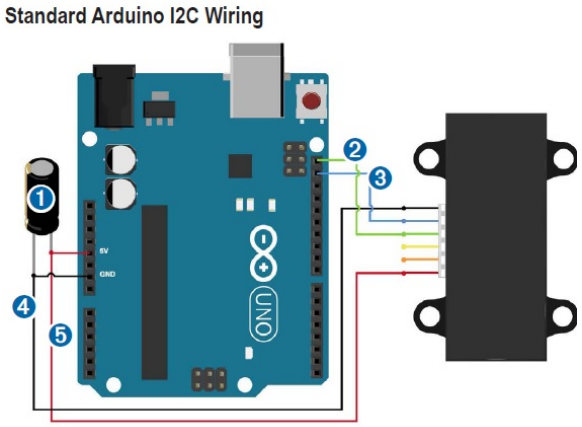


Figure 3.3: Aduino-LiDAR Connection

Item	Description	Notes
1	680 μF electrolytic capacitor	you must observe the correct polarity when installing the capacitor.
2	Power Ground (-) connection	Black Wire
3	I2C SDA connection	Blu wire
4	I2C SCL connection	Green wire
		Red wire
5	5 Vdc power (+) connection	The sensor operates at 4.75 through 5.5 Vdc, with a max. of 6 Vdc.

Table 3.6: Arduino-LiDAR Connection

(7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board;

- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA;

- GND. Ground pins;

- IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

3.2.4 Input and Outputs

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.

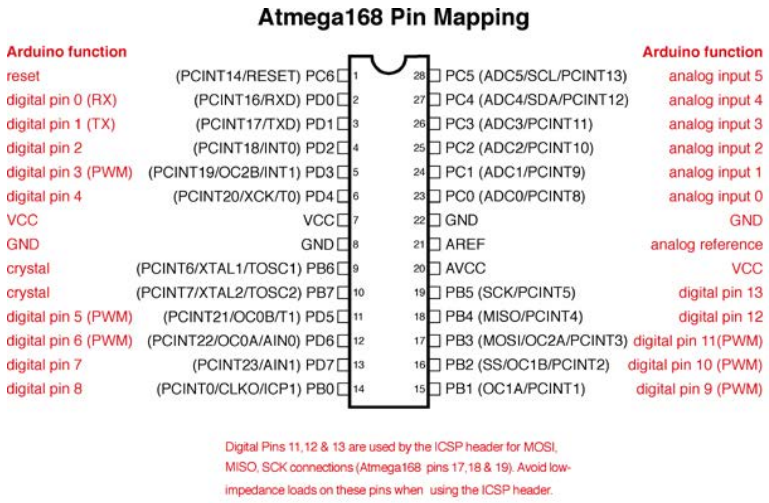


Figure 3.4: Arduino ATMega Chipset

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50kΩ. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the micro-controller. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip;
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details;
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function;
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library;
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off;

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

Communication Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other micro-controllers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library. Automatic (Software) Reset Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

3.3 Servo Motors

Servo motors are specially designed motors to be used in control applications and robotics. They are used for precise position and speed control at high torques. That's why we thought that the servo could have been the best choice for the LiDAR project. It consists of a suitable motor, position sensor and a sophisticated controller. Servo motors can be characterized according the motor controlled by servomechanism. Servo motors are available in power ratings from fraction of watt up to few 100 watts. They are having high torque capabilities. The rotor of servo motor is made smaller in diameter and longer in length, so that it has low inertia. What Is Servomechanism? Servomechanism is basically a closed-loop system Fig. 3.5, consisting of a controlled device, controller, output sensor and feedback system. The term servomechanism most probably applies to the systems where position and speed is to be controlled. Mechanical position of the shaft can

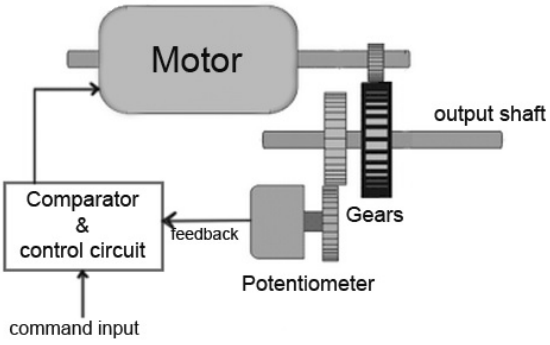
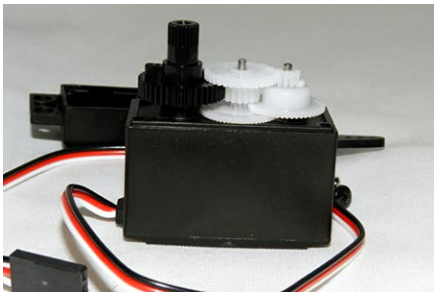


Figure 3.5: Servo Motor, scheme

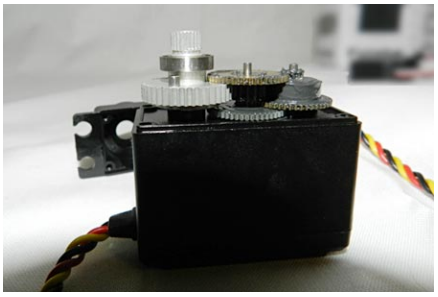
be sensed by using a potentiometer, which is coupled with the motor shaft through gears. The current position of the shaft is converted into electrical signal by the potentiometer, and then compared with the command input signal. In modern servo motors, electronic encoders or sensors are used to sense the position of the shaft. Command input is given according to the required position of the shaft. If the feedback signal differs from the given input, an error signal is generated. This error signal is then amplified and applied as the input to the motor, which causes the motor to rotate. And when the shaft reaches the required position, error signal becomes zero, and hence the motor stays standstill holding the position.

3.3.1 How It Works

The simplicity of a servo is among the features that make them so reliable. The heart of a servo is a small direct current (DC) motor, similar to what you might find in an inexpensive toy. These motors run on electricity from



(a) Plastic Gears



(b) Metal Gears

Figure 3.6: Stepper Gear examples

a battery or proper DC source and spin at high RPM (rotations per minute) but put out very low torque. An arrangement of gears takes the high speed of the motor and slows it down while at the same time increasing the torque. It's common sense the basic law of physics: $work = force \times distance$. A tiny electric motor does not have much torque, but it can spin really fast (small force, big distance). The gear design inside the servo case converts the output to a much slower rotation speed but with more torque (big force, little distance). Even though, the amount of actual work is the same. Gears in an inexpensive servo motor are generally made of plastic to keep it lighter and less costly (see Fig. 3.6a). On a servo designed to provide more torque for heavier work, the gears are made of metal (see Fig. 3.6b) and are harder to damage. With a small DC motor, you apply power from a battery, and the motor spins. Unlike a simple DC motor, however, a servo's spinning motor shaft is slowed way down with gears. A positional sensor on the final gear is connected to a small circuit board (see Fig. 3.7). The sensor tells this circuit board how far the servo output shaft has rotated. The electronic input signal from the computer feeds into that circuit board. The electronics on the circuit board decode the signals to determine how far the user wants the servo to rotate. It then compares the desired position to the actual position and decides which direction to rotate the shaft so it gets to the desired position. That is what makes servo motors so useful: once you tell them what you want done, they do the job without your help. This automatic seeking behaviour of servo motors makes them perfect for many robotic applications.
A lot of different kinds of servo motors are these days available:

- Positional rotation servo, this is the most common type of servo motor. The output shaft rotates in about 90 degrees, 180 degrees or 270 degrees. It has physical stops placed in the gear mechanism to prevent turning beyond these limits to protect the rotational sensor;

- Continuous rotation servo, this is quite similar to the common po-

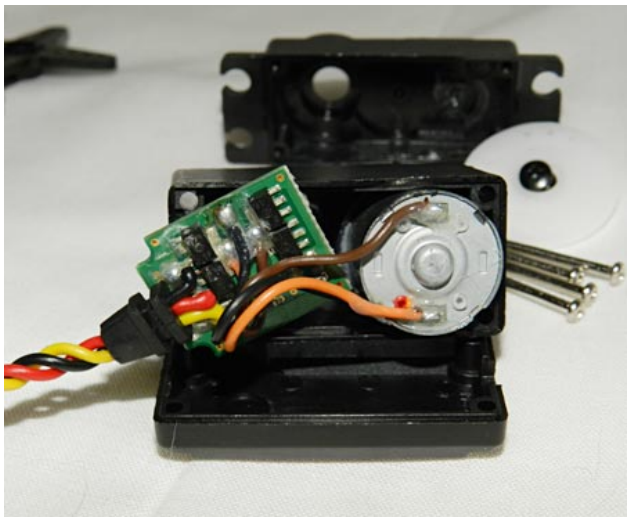


Figure 3.7: Servo sensor

sitional rotation servo motor, except it can turn in either direction indefinitely. The control signal, rather than setting the static position of the servo, is interpreted as the direction and speed of rotation. The range of possible commands causes the servo to rotate clockwise or counter-clockwise as desired, at varying speed, depending on the command signal;

- Linear servo, this is also like the positional rotation servo motor described above, but with additional gears (usually a rack and pinion mechanism) to change the output from circular to back-and-forth. These servos are not easy to find though.

3.3.2 Control Strategy

Usually, the servo is controlled thanks to a micro-controller. In fact, the motor presents 3 wires: GND, 5V and signal or better pulse (most of the time, respectively, black or brown, red and orange or yellow). Intuitively, the two wires responsible to power the motor must be connected to the respective board pin of the controller, while the signal wire must be connected to a PWM output pin.
Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° or 135° in either direction for a total of 180° or 270° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the motor determines the position of

the shaft, and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) or in other words 50Hz and the length of the pulse will determine how far the motor turns. For example for a common 180° servo, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter-clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position. The motor's speed, then, is proportional to the difference between its actual position and desired position. So if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called proportional control. This means the motor will only run as hard as necessary to accomplish the task at hand, a very efficient way to achieve the rotation, see Fig. 3.8.

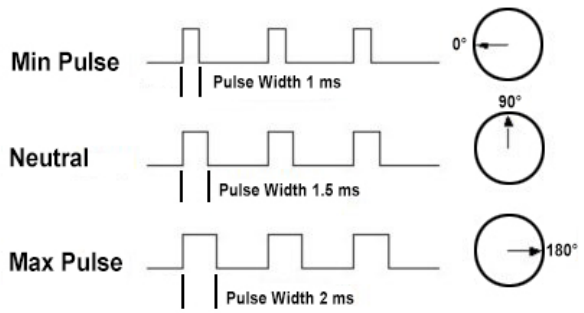


Figure 3.8: Servo Control Strategy

In the next chapter will be presented the result of the servo test that has been carried on in lab thanks to an oscilloscope (See §Chap. 4, §Sec. 4.2.1).

3.4 Processing v3

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping. Its features are:[18]

- Free to download and open source;
- Interactive programs with 2D, 3D, PDF, or SVG output;
- OpenGL integration for accelerated 2D and 3D;

- For GNU/Linux, Mac OS X, Windows, Android, and ARM;
- Over 100 libraries extend the core software;
- Well documented, with many books available;

Processing is an open-source graphical library and integrated development environment (IDE) / playground built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context. Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. As well as this, it also has a graphical user interface for simplyfing the compilation and execution stage. The Processing language and IDE were the precursor to numerous other projects, notably Arduino, Wiring and P5.js. Processing includes a sketchbook, a minimal alternative to an integrated development environment (IDE)(See Fig. 3.9) for organizing projects.[19]

Every Processing sketch is actually a subclass of the PApplet Java class (formerly a subclass of Java's built-in Applet) which implements most of the Processing language's features.[20] When programming in Processing, all additional classes defined will be treated as inner classes when the code is translated into pure Java before compiling.[21] This means that the use of static variables and methods in classes is prohibited unless Processing is explicitly told to code in pure Java mode. Processing also allows for users to create their own classes within the PApplet sketch. This allows for complex data types that can include any number of arguments and avoids the limitations of solely using standard data types such as: int (integer), char (character), float (real number), and color (RGB, RGBA, hex). Processing relates software concepts to principles of visual form, motion, and interaction. It integrates a programming language, development environment, and teaching methodology into a unified system. The Processing language is a text programming language specifically designed to generate and modify images. Processing strives to achieve a balance between clarity and advanced features. Beginners can write their own programs after only a few minutes of instruction, but more advanced users can employ and write libraries with additional functions. The system facilitates teaching many computer graphics and interaction techniques including vector/raster drawing, image processing, color models, mouse and keyboard events, network communication, and object-oriented programming. Libraries easily extend Processing's ability to generate sound, send/receive data in diverse formats, and to import/export 2D and 3D file formats.[22]
So, thanks to the concepts presented in this chapter it will be possible to explain how the first LiDAR platform has been built.

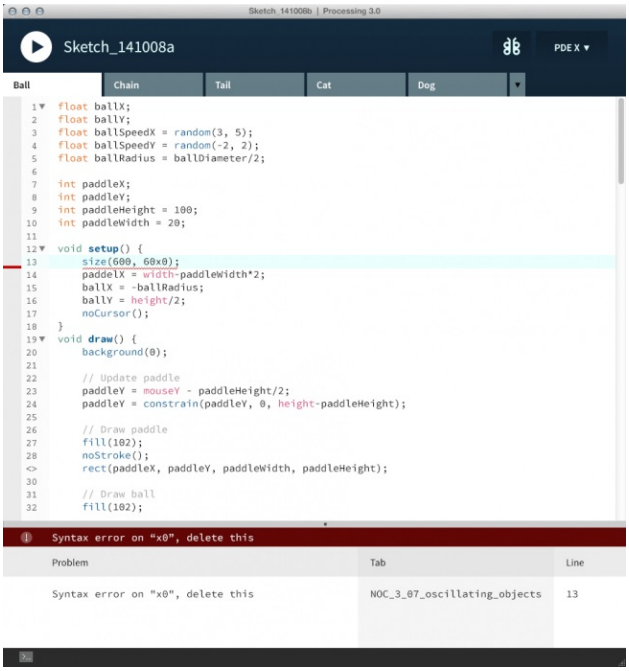


Figure 3.9: Processing v3 IDE

3.4.1 Export

The export feature packages a sketch to run within a Web browser. When code is exported from Processing it is converted into Java code and then compiled as a Java applet. When a project is exported, a series of files are written to a folder named applet that is created within the sketch folder. All files from the sketch folder are exported into a single Java Archive (JAR) file with the same name as the sketch. For example, if the sketch is named Sketch_123, the exported file will be called Sketch_123.jar. Every time a sketch is exported, the contents of the applet folder are deleted and the files are written from scratch. Any changes previously made to the index.html file are lost. Media files not needed for the applet should be deleted from the data folder before it is exported to keep the file size small. For example, if there are unused images in the data folder, they will be added to the JAR file, thus needlessly increasing its size. In addition to exporting Java applets for the Web, Processing can also export Java applications for the Linux, Macintosh, and Windows platforms. When "Export Application" is selected from the File menu, folders will be created for each of the operating systems specified in the Preferences. Each folder contains the application, the source code for the sketch, and all required libraries for a specific platform. Additional and updated information about the Processing environment is available at www.processing.org/reference/environment or by selecting the

"Environment" item from the Help menu of the Processing application.[22]

Chapter 4

LiDAR Platform v1

Firstly, it's necessary to reaffirm that this first platform will be part of a bigger learning phase. So, we will investigate how it's possible to build such a similar structure, discovering step by step problems and figuring out how to overcome these issues. Moreover, this first version will be essential to understand and define the boundaries of version one; at the same time foresee a possible solution to improve the machine's performance, towards version 2.
Now, I will list the key components of the version that will be discussed in detail in further sections of this chapter:

- Sensor;
- Servo motors (x 2);
- Arduino;
- Arduino Expansions;
- Simple Structure;
- Bluetooth, master and slave;
- Power Bank;
- Jumpers & cables.

As it's possible to image, before putting all the parts together it's necessary to focus on the single parts, if required test in lab some specific components and then assemble the whole platform solving integration problems or issues and then test again. This last test is essential, since it's like a benchmark between the requirements on which the platform has been built and the performances that the final product is actually able to deliver.
At the beginning of the process a general timeline has been deployed. The

timeline itself presents different steps and phases, which correspond to different implementations and functionalities. Going on with the process, taking into account time constraint and possible improvements of the platform, we will decide what to do as first or what to exclude. See Fig. 4.1.

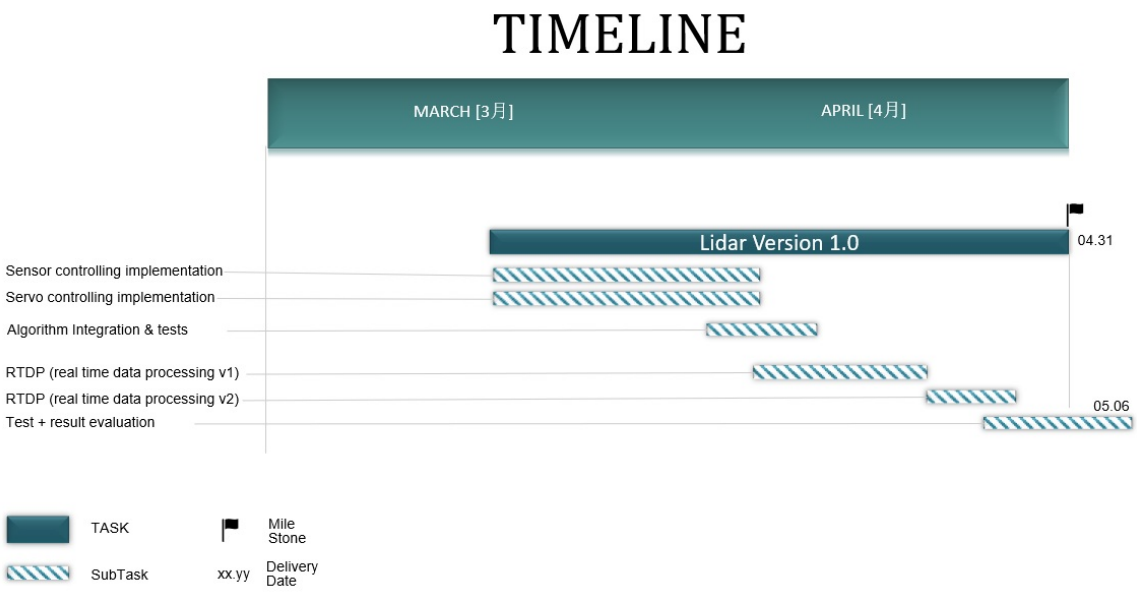


Figure 4.1: Timeline detail v1

Anyway, right now it's time to show what is the architecture of this first prototype, see Fig. 4.2. It's possible to see how the parts and tools have been assembled together to form the LiDAR system. Since the structure has to be portable, we have decided to provide the power necessary to feed the equipment through a 5V output power bank. The full capacity of the power supply will be 10000mAh, necessary to feed sensor, controller, motors and the bluetooth. So, in simple words, the two motors will move the sensor (pan and tilt movement), as soon as the movement is completed the sensor acquisition is triggered by the Arduino, which contemporary knows and control the servos, the measurement will be then elaborated by the MCU and sent to the PC via Serial. Thanks to the adoption of the master and slave wireless transmission of the informations, it's possible to keep the circuit simple, reduce the number of wires that is used without compromising the transmission speed. So, the master device will be connected to the Arduino Board expansion (directly mounted on the Arduino PCB itself) with four pins: two for power (GND and 5V) and other two for data communication (Tx and Sx). Obviously, then, the master will be coupled with his slave and that one will be directly plugged into the USB port of the PC. In this way, the slave could be powered and at the same time transmit data via Serial Port to the Processing v3 Sketch that runs on the machine.

The only thing that is missing, though it can be seen on the schematic, is the presence of the Arduino servo expansion. By using that, we achieve: an independence of the motors power supply for the Arduino and at the same time the possibility of an easy control and feeding of the two servos.

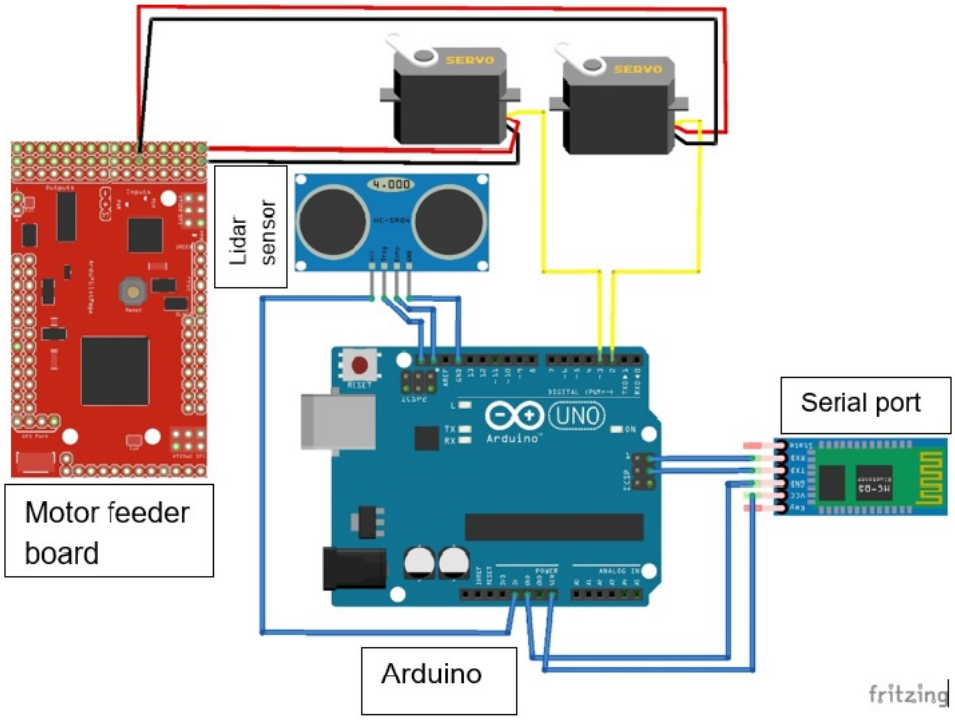


Figure 4.2: LiDAR v1 Architecture

4.1 Sensor control Strategy

The Sensor control strategy will influence the whole project. The idea is to control the sensor as a single shot gun: in fact, knowing the exact position of the sensor in the space will, then, allow the triggering of the measurement. When the process is completed the sensor will be then moved to the further position and triggered again; and so on.
In this way the sensor will be used and exploited as a point-to-point acquisition system. This control mode, seems to be the best taking into account the features of the sensor itself and the hardware that will be coupled.
But one thing for sure it's very important: how to use the sensor combined to the Arduino Board. We have previously seen how to connect physically the two entities. But, what about the software interface? Luckily, Garmin (the sensor manufacturer) has released an Arduino library that makes easier and intuitive how to control the LiDAR. In particular, it's for sure firstly

necessary to install the library in the Arduino library folder. Once that this step has been done, the sensor can be easily controlled and trigger, as said before.
Firstly, it's mandatory to define the LiDAR OBJECT with a proper syntax. Then, it's necessary to initiate the communication via I2C (the transmission mode we have chosen). Then, with a single command is possible to trigger the sensor and capture the measurement. Anyway, the sensor can operate in different modes and with two levels of accuracy. Obviously, the deal it's a compromise between speed and reliability of the data gathered by the sensor. The following listed code explains basically how to operate the sensor:

```
/*------------------------------------------------------
  This example shows how to initialize, configure,
  and read distance from a LIDAR-Lite connected over I2C
    interface.

  Connections:
  LIDAR-Lite 5 Vdc (red) to Arduino 5v
  LIDAR-Lite I2C SCL (green) to Arduino SCL
  LIDAR-Lite I2C SDA (blue) to Arduino SDA
  LIDAR-Lite Ground (black) to Arduino GND

  (Capacitor recommended to mitigate inrush current when device
    is enabled)
  680uF capacitor (+) to Arduino 5v
  680uF capacitor (-) to Arduino GND
------------------------------------------------------*/
#include <Wire.h>   // Standard Library
#include <LIDARLite.h> // LiDAR's Library

LIDARLite myLidarLite; // LiDAR OBJECT DEFINITION

void setup()
{
  Serial.begin(115200); // Initialize serial connection to
    display distance readings
/*
begin(int configuration, bool fasti2c, char lidarliteAddress)
-> Starts the sensor and I2C.

Parameters
-------------------------------------------------
  configuration: Default 0.
  Selects one of several preset configurations.
  fasti2c: Default 100 kHz. I2C base frequency.(If true I2C
    frequency is set to 400kHz) */
  myLidarLite.begin(0, true); // Set configuration to default
    and I2C to 400 kHz
/* configure(int configuration, char lidarliteAddress)
   Selects one of several preset configurations.
Parameters
```

```
36 ----------------------------------------------------
37   configuration:  Default 0.
38     0: Default mode, balanced performance.
39     1: Short range, high speed. Uses 0x1d maximum acquisition
       count.
40     2: Default range, higher speed short range. Turns on quick
       termination detection for faster measurements at short range(
       with decreased accuracy).
41     3: Maximum range. Uses 0xff maximum acquisition count.
42     4: High sensitivity detection. Overrides default valid
       measurement detection algorithm, and uses a threshold value
       for high sensitivity and noise.
43     5: Low sensitivity detection. Overrides default valid
       measurement detection algorithm, and uses a threshold value
       for low sensitivity and noise. */
44   myLidarLite.configure(0); // Change this number to try out
       alternate configurations
45 }
46 void loop()
47 {
48   /* distance(bool biasCorrection, char lidarliteAddress)
49      Take a distance measurement and read the result.
50 Parameters
51 ----------------------------------------------------
52  biasCorrection: Default true. Take acquisition with receiver
       bias correction. If set to false measurements will be faster.
        Receiver bias correction must be performed periodically. (e.
       g. 1 out of every 100 \readings). */
53
54   // Take a measurement with receiver bias correction and print
       to serial terminal
55   Serial.println(myLidarLite.distance());
56
57   // Take 99 measurements without receiver bias correction and
       print to serial terminal
58   for(int i = 0; i < 99; i++)
59   {
60     Serial.println(myLidarLite.distance(false));
61   }
62 }
```

Listing 4.1: Code Listing-sensor acquisition

Taking a look at the code listing 4.1; it's possible to see that on:

- Line 18, LiDAR Object has been defined;
- Line 32, I2C communication starts, with default settings (value = 0) and speed set to 400kHz (true);
- Line 44, set LiDAR operational mode (value = 0, balanced performances);
- Line 55, acquisition trigger (no bias correction). It means that the measure is faster but not super accurate; as we have said the measure

is a compromise between acquisition speed and accuracy. So it's recommended to re-calibrate the sensor every 100 points, to get a more precise distance value.

4.2 Servo control Strategy

The servo control strategy it's pretty straight forward. Again, Arduino helps us since the servo library it's available. As we have said, standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. Once, the servo is powered and the pulse wire is connected to a PWM Arduino output pin, it's necessary to declare two servo entities (pan and tilt motor, motor_x and motor_y). Then, the game is quite simple: in fact it's necessary to remember that the servo's shaft can assume each position between 0° and 180°. To change the shaft's position it's just needed to use the library's instruction *write*. While, if we want to know the exact position of the servo we can use the instruction *read*. Both, *read* and *write*, are used as attributes or methods applied to the servo entities.

Servo Trajectory At this point it will be briefly discussed an hot topic, that will be treated for sure with a deeper focus in following sections. The final density of the point cloud[1] is linked for sure to the reliability of the sensor and its capability to trigger and gather distances in the smallest possible period of time. But, assuming that the sensor is fast enough, as it actually is, the main driver that determines the density of the point cloud is the minimum step that the servo can execute. Taking a look on the internet and LiDAR commercial products, it's noticeable that extremely detailed point clouds reach, at least, densities around 1 or 2 million points (over a 3D sapace which is roughly 180° × 90°, in other words half of a semi-sphere). For this first prototype, though, the density is absolutely lower, around 200000 points. This comes from a very simple mathematical calculation, considering the smallest servo's movement equal to 1 degree and a scanned 3D space which is actually close to half of a semi-sphere. At this point, we don't know yet if this density is enough to distinguish, at least roughly, obstacles and scanned objects. In the final phase of the development step this aspect will be checked and discussed: if the result it's not considered acceptable, the version2 will be completely focused on a

[1] A point cloud is a set of data points in space. Point clouds are generally produced by 3D scanners, which measure a large number of points on the external surfaces of objects around them. As the output of 3D scanning processes, point clouds are used for many purposes, including to create 3D CAD models for manufactured parts, for metrology and quality inspection, and for a multitude of visualization, animation, rendering and mass customization applications.

solution able to increment dramatically the density and performances in general. Unfortunately, at least for version1, the servo library allows a minimum step of 1°, any smaller rotation is prohibited and impossible to be executed.
Last but not the least, it's necessary to talk a bout the trajectory that the pan and tilt motor will describe: the pan motor will sweep the horizontal plane (rotation around a vertical axis), while the tilt motor will sweep the vertical plane (rotation around a horizontal axis). When the Arduino board is turned on, the reset operation brings the servo at homing position. In particular, the homing position is around the middle of the entire interval. So, then, the servo will swipe more or less 85 degrees clockwise and 85 degrees anti-clockwise. Once, a complete 180° angle is described, the tilt servo will move 1 degree upward.

4.2.1 Servo characteristic determination

At this point, though, we posed ourselves a question: "How do we know that the servo characteristic is linear?". In other words, who is assuring us that when we send the instruction *servo.write(37)* the signal sent through the pulse wire corresponds to a precise PWM 50 Hz signal interpreted by the servo as a rotation to position 37, for example. For this reason, we designed a simple test in lab, able to define and give a precise idea of the servo characteristic. By using a common digital oscilloscope synchronized to the servo pulse wire and sending firstly the pulses from PC, thanks to a testing tool provided with servos, it's possible to check this out. The first thing that we discovered was that the PWM train is actually 50Hz, as expected; but then we noticed that for our servos, the width of the pulse varies between 500μs (which corresponds to 0°) and 2500μs (which corresponds to 180°). Before taking a look at the data, it's reasonable to explain better how the measurement has been carried on. On the Pc has been installed a servo test tool, specifically design to control the Arduino Servo expansion. This tool, is able to produce a pulsed signal that drives the servos, through the control board. The notable advantage is that it's easily possible to drive the motor thanks to the Pc interface. The general purpose of the experience is to collect roughly 20 points and identify the characteristic of the servo motor. By varying the position of the shaft on the Pc interface and then measuring the PWM signal thanks to the oscilloscope, it's possible to check the amplitude of the signal, as well as the frequency. See Fig. 4.4. In the picture, it's possible to notice, on the right bottom part of the oscilloscope's screen, that the signal's frequency it's around 50Hz (precisely 49,951Hz) and the width of the pulse corresponding to 180° is 2500μs, as expected. Anyway, in the preliminary part of the lab experience it has been noticed that the PWM signal near the boundaries (0 and 180) in quite unstable, so it's been decided not to drive the servo with angles smaller than 10° and not grater

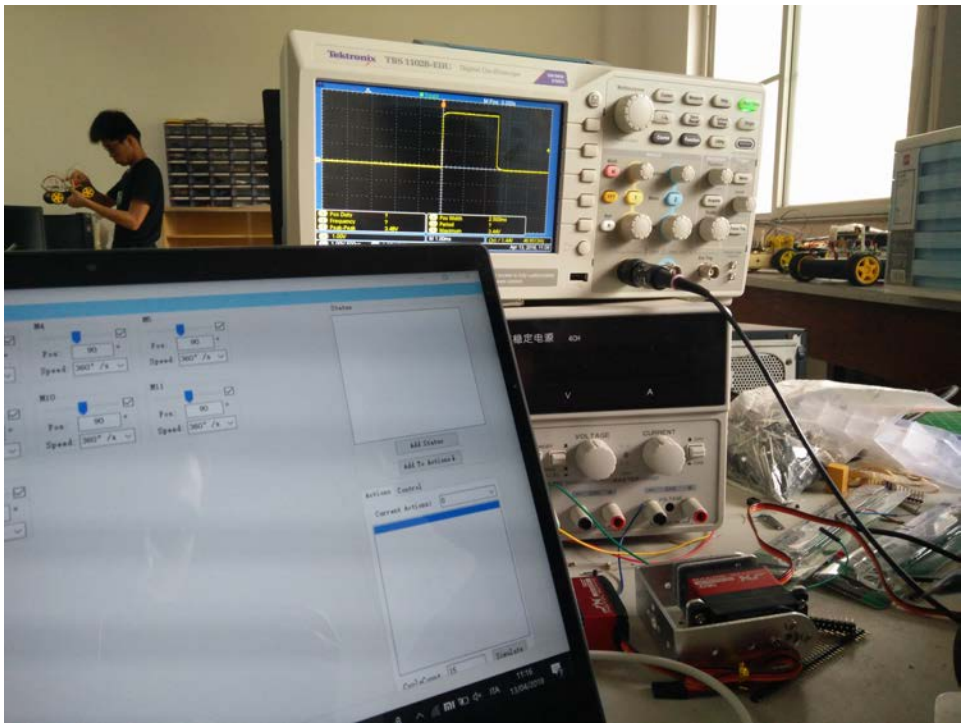


Figure 4.3: Servo lab test 1

than 170°, since there's no linearity in this portion of the characteristic. So, then, the characteristic has been drawn testing 20 positions, especially close to the upper and lower limit of the servo previously fixed. Taking a look at Tab. 4.1 and to Fig 4.5 it's possible to see clearly that the servo characteristic is pretty linear, or at least it's quite possible to consider it linear. This was the result we were looking for, considering the fact that the resolution of the point cloud and the accuracy of the point positioning is strictly dependent on the servos.

4.3 Data Elaboration and Transmission

Right now, this chapter has explained the main mechanism around the sensor acquisition and servo controlling. The output of the LiDAR sensor measurement is a distance, associated to two angular positions, respectively related to the two motors. It's intuitive that the sensor describes a trajectory that moves on a semi-sphere. At the same time, though, the final representation that we want is a three dimensional picture based on a Cartesian coordinate system.

$$\mathcal{T}_{(\rho,\theta,\phi)} \Rightarrow \mathcal{T}_{(x,y,z)} \tag{4.1}$$

Degrees	Pulse Amplitude (μs)
10	600
12	640,8
14	660,4
16	680,5
18	700,3
20	720,5
22	740,4
24	760,2
26	780,5
28	800,5
30	820,3
32	840
90	1502
150	2161
151	2180
153	2200
155	2220
157	2241
160	2280
163	2030
165	2320
169	2360
170	2400

Table 4.1: Servo Characteristic

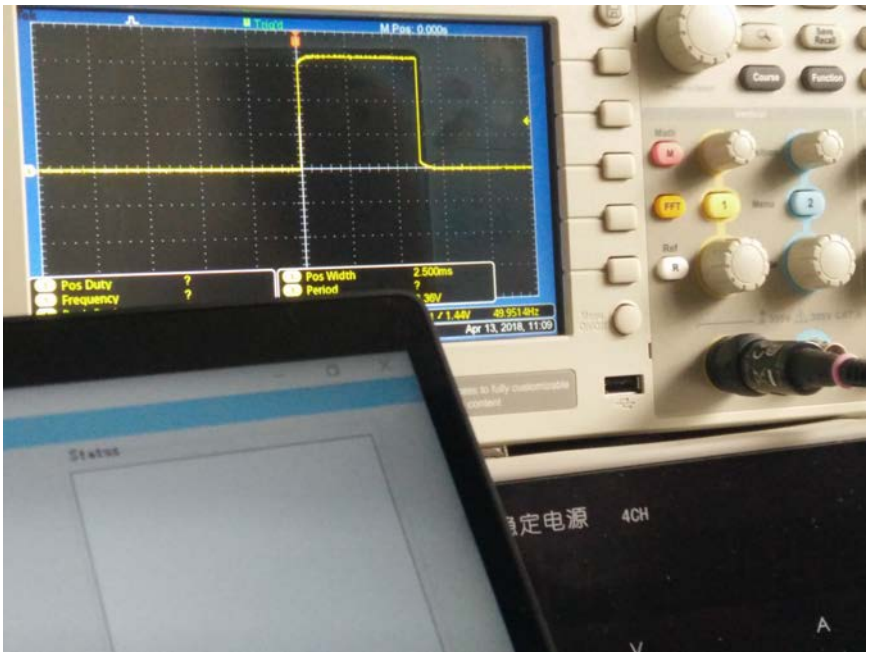


Figure 4.4: Servo lab test 2

In mathematics, a spherical coordinate system is a coordinate system for three-dimensional space where the position of a point is specified by three numbers: the radial distance of that point from a fixed origin, its polar angle measured from a fixed zenith direction, and the azimuth angle of its orthogonal projection on a reference plane that passes through the origin and is orthogonal to the zenith, measured from a fixed reference direction on that plane. It can be seen as the three-dimensional version of the polar coordinate system. The radial distance is also called the radius or radial coordinate. The polar angle may be called colatitude, zenith angle, normal angle, or inclination angle. The use of symbols and the order of the coordinates differs between sources. In one system frequently encountered in physics (ρ, θ, ϕ) gives the radial distance, polar angle, and azimuthal angle, whereas in another system used in many mathematics books (ρ, θ, ϕ) gives the radial distance, azimuthal angle, and polar angle. Other conventions are also used, so great care needs to be taken to check which one is being used. A number of different spherical coordinate systems following other conventions are used outside mathematics. In a geographical coordinate system positions are measured in latitude, longitude and height or altitude. There are a number of different celestial coordinate systems based on different fundamental planes and with different terms for the various coordinates. The spherical coordinate systems used in mathematics normally use radians rather than degrees and measure the azimuthal angle counter-clockwise from

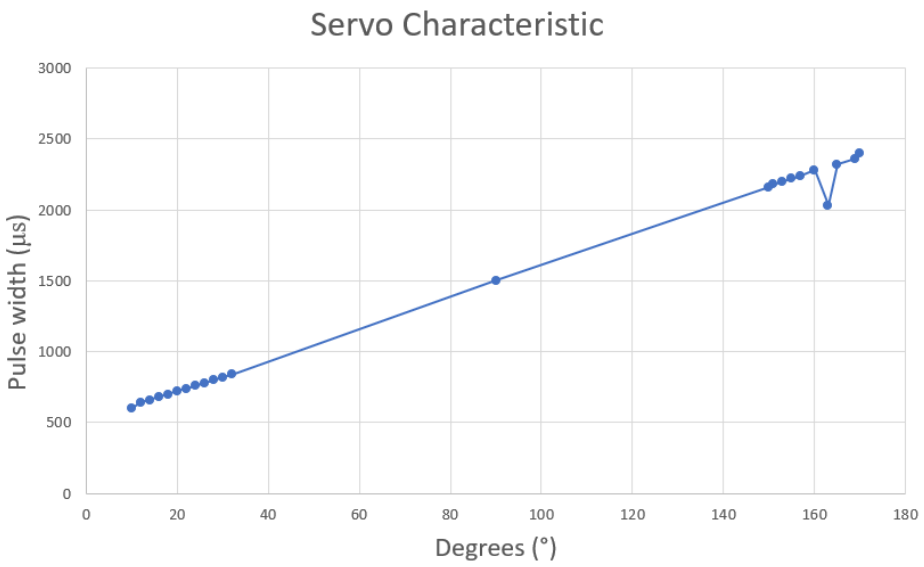


Figure 4.5: Servo Characteristic, graph

the x-axis to the y-axis rather than clockwise from north (0°) to east (90°) like the horizontal coordinate system. The polar angle is often replaced by the elevation angle measured from the reference plane. Elevation angle of zero is at the horizon. The spherical coordinate system generalizes the two-dimensional polar coordinate system. It can also be extended to higher-dimensional spaces and is then referred to as a hyper-spherical coordinate system.

Spherical coordinates determine the position of a point in three-dimensional space based on the distance ρ from the origin and two angles θ and ϕ. The following graphics (See Fig. 4.6) may help to understand spherical coordinates better. On this page, we derive the relationship between spherical and Cartesian coordinates.

Relationship between spherical and Cartesian coordinates Spherical coordinates are defined as indicated in the following figure, which illustrates the spherical coordinates of the point $\mathcal{P}$. The coordinate ρ is the distance from P to the origin. If the point $\mathcal{Q}$ is the projection of $\mathcal{P}$ to the xy-plane, then θ is the angle between the positive x-axis and the line segment from the origin to $\mathcal{Q}$. Lastly, ϕ is the angle between the positive z-axis and the line segment from the origin to $\mathcal{P}$. We can calculate the relationship between the Cartesian coordinates (x, y, z) of the point $\mathcal{P}$ and its spherical coordinates (ρ, θ, ϕ) using trigonometry. The pink triangle (See Fig. 4.7) is the right triangle whose vertices are the origin, the point $\mathcal{P}$, and its projection onto the z-axis. As the length of the hypotenuse is ρ and ϕ

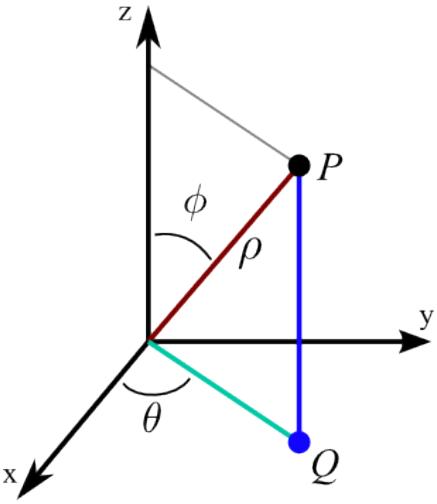


Figure 4.6: Spherical Coordinates

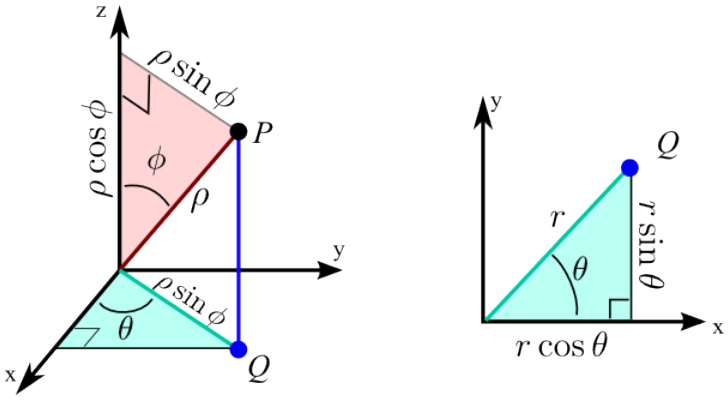


Figure 4.7: Spherical & Cartesian Coordinates

is the angle the hypotenuse makes with the z-axis leg of the right triangle, the z-coordinate of $\mathcal{P}$ (i.e., the height of the triangle) is $z = \rho \cos \phi$. The length of the other leg of the right triangle is the distance from $\mathcal{P}$ to the z-axis, which is $r = \rho \sin \phi$. The distance of the point $\mathcal{Q}$ from the origin is the same quantity. The cyan triangle, shown in both the original 3D coordinate system on the left and in the xy-plane on the right, is the right triangle whose vertices are the origin, the point $\mathcal{Q}$, and its projection onto the x-axis. In the right plot, the distance from $\mathcal{Q}$ to the origin, which is the length of hypotenuse of the right triangle, is labelled just as r. As θ is the angle this hypotenuse makes with the x-axis, the x- and y-components of the point $\mathcal{Q}$ (which are the same as the x- and y-components of the point $\mathcal{P}$) are given by $x = r \cos \theta$ and $y = r \sin \theta$. Since $r = \rho sin\theta$, these components can be rewritten as $x = \rho \sin \phi \cos \theta$ and $y = \rho \sin \phi \sin \theta$. In summary, the formulas

for Cartesian coordinates in terms of spherical coordinates are:

$$x = \rho \sin\phi \cos\theta$$
$$y = \rho \sin\phi \sin\theta$$
$$z = \rho \cos\theta$$

So, once the sensor has revealed the distance, there are some lines in the Arduino code that operates the transformation and then print a string with (x, y, z) on the serial port. This operation it's quite easy thanks to the instruction *println*. It is as well possible to print the data to file, but actually we preferred to transmit the data via Serial: in this way it's immediate a real time elaboration.

4.4 Data Processing, Visualization and Results

Since the scanning process has been totally explained, now it's the visualization turn. Once, the data has been collected, then there's the need to show them and create a virtual environment which let us appreciate the shapes of scanned objects. Moreover, as we have chosen, we want it to be real time, which means that the scanning process and visualization process are contemporary.
In other words, as previously hinted in §Chap. 3, §Sec. 3.4, a processing sketch running on the Pc will show the point cloud. The sketch should:

- Read real time the Serial Port;
- Store temporarily the data in a matrix;
- Plot the data without delays;
- Assign a proper color to each point;
- Offer a proper navigation system;
- Allow to save data.

As a matter of facts, the Arduino output is (x, y, z) coordinates, so the processing v3 sketch accepts (x, y, z) data as input. The big deal is to set the correct Serial Port speed, synchronized to the Arduino transmission baud rate.

```
1 // This sketch accepts XYZ coordinates from Arduino LIDAR
    scanner and displays them graphically as a 3D point cloud
    that you can pan, zoom, and rotate using keyboard.
2
3 import processing.serial.*; // Serial Port Java library
4 import java.io.FileWriter;   // libraries necessary to export
    file
```

```
5  import java.util.Calendar;
6  import java.text.SimpleDateFormat;
7  \\global variable initialization
8  Serial serial;
9  int serialPortNumber = 0;
10 float angle = 6.5f;
11 float angleIncrement = 0;
12 float xOffset = 3.0;
13 float xOffsetIncrement = 0;
14 float yOffset = 152.0f;
15 float yOffsetIncrement = 0;
16 float scaleIncrement = 0;
17 float h;
18 ArrayList<PVector> vectors; \\ vector initialization to store
       data
19 int lastPointIndex = 0;
20 int lastPointCount = 0;
21
22 void setup() {
23   size(800, 600, P3D);  // environment initialization
24   colorMode(HSB, 360, 100, 100);
25   noSmooth();
26   vectors = new ArrayList<PVector>();
27   String[] serialPorts = Serial.list(); // serial port read ->
28   String serialPort = serialPorts[serialPortNumber];
29   println("Using serial port \"" + serialPort + "\"");
30   println("To use a different serial port, change
       serialPortNumber:");
31   printArray(serialPorts);
32   serial = new Serial(this, serialPort, 115200);} // <-
33 void draw() {
34   String input = serial.readStringUntil(10);
35   if (input != null) {
36     String[] components = split(input, ' ');
37     if (components.length == 3) {
38       vectors.add(new PVector(float(components[0]), float(
       components[1]), float(components[2])));}}
39   background(0);
40   translate(width/2, height/2, -50);
41   rotateY(angle);
42   int size = vectors.size();
43   for (int index = 0; index < size; index++) {
44     PVector v = vectors.get(index);
45     if (index == size - 1) {
46       // draw red line to show recently added LIDAR scan point
47       if (index == lastPointIndex) {
48         lastPointCount++;
49       } else {
50         lastPointIndex = index;
51         lastPointCount = 0;
52       }
53       if (lastPointCount < 10) {
54         stroke(0, 100, 100);
55         line(xOffset, yOffset, 0, v.x * scale + xOffset, -v.z *
```

```
    scale + yOffset, -v.y * scale);}}
56    h= sqrt(v.x*v.x+v.y*v.y+v.z*v.z);
57    stroke(360-(h*0.4), 100, 100);
58    point(v.x * scale + xOffset, -v.z * scale + yOffset, -v.y *
      scale);}
59  angle += angleIncrement;
60  xOffset += xOffsetIncrement;
61  yOffset += yOffsetIncrement;
62  scale += scaleIncrement;}
63 void keyPressed() {
64  if (key == 'q') {
65    // zoom in
66    scaleIncrement = 0.02f;
67  } else if (key == 'z') {
68    // zoom out
69    scaleIncrement = -0.02f;
70  } else if (key =='p') {
71    // erase all points
72    vectors.clear();
73  } else if (key == 'a') {
74    // move left
75    xOffsetIncrement = -1f;
76  } else if (key == 'd') {
77    // move right
78    xOffsetIncrement = 1f;
79  } else if (key == 'w') {
80    // move up
81    yOffsetIncrement = -1f;
82  } else if (key == 'x') {
83    // move down
84    yOffsetIncrement = 1f;
85  }else if (key == CODED) {
86    if (keyCode == LEFT) {
87      // rotate left
88      angleIncrement = -0.015f;
89    } else if (keyCode == RIGHT) {
90      // rotate right
91      angleIncrement = 0.015f;}}}
92 void keyReleased() {
93  if (key == 'q') {
94    scaleIncrement = 0f;
95  } else if (key == 'z') {
96    scaleIncrement = 0f;
97  } else if (key == 'a') {
98    xOffsetIncrement = 0f;
99  } else if (key == 'd') {
100   xOffsetIncrement = 0f;
101 } else if (key == 'w') {
102   yOffsetIncrement = 0f;
103 } else if (key == 'x') {
104   yOffsetIncrement = 0f;
105 } else if (key == 's') {
106     saveToFile();
107 }else if (key == CODED) {
```

```
108     if (keyCode == LEFT) {
109       angleIncrement = 0f;
110     } else if (keyCode == RIGHT) {
111       angleIncrement = 0f;}}}
112 // Function to save the point cloud in the Processing install
    directory
113 void saveToFile(){
114   String fileName = "./points_"+
115         new SimpleDateFormat("yyMMdd_HHmmss").format(Calendar.
    getInstance().getTime())+".xyz";
116   PrintWriter pw = null;
117  try{
118     pw = new PrintWriter(new FileWriter(fileName,true));
119     for(int i=0;i<vectors.size();i++)
120       pw.println((int)vectors.get(i).x + " " +
121                  (int)vectors.get(i).y + " " +
122                  (int)vectors.get(i).z);}
123   catch(Exception e){
124   }finally {
125     if(pw != null) pw.close();}}
```

Listing 4.2: Processing sketch

In the following part the algorithm will be commented to explain the different parts and functionalities. As for each scratch or program at the beginning are reported the libraries that are needed to run, compile and interpret the algorithm. Then the global variables are exposed.

- Line 23, this instruction declares and initialize the size and the colour of the environment trough which the point cloud will be showed. It will be a rectangle 800 × 600 pixels. Before drawing 3D form in Processing, it's necessary to tell the software to draw with a 3D renderer. The default renderer in Processing draws only two-dimensional shapes, but there are additional options (P3D and OPENGL) to render 3D form. P3D is the simplest and most compatible renderer, and it requires no additional libraries;

- Line 24, definition of the color mode and the background colour of the environment. Processing uses the RGB color model as its default for working with color, but the HSB specification can be used instead to define colors in terms of their hue, saturation, and brightness. The hue of a color is what most people normally think of as the color name: yellow, red, blue, orange, green, violet. A pure hue is an undiluted color at its most intense. The saturation is the degree of purity in a color. It is the continuum from the undiluted, pure hue to its most diluted and dull. The brightness of a color is its relation to light and dark.[22]
  For example the color black is represented as (HSB, 360, 100, 100). Somebody could raise the hand and ask why the system HSB has been

chosen rather the more common and intuitive RGB. The answer lies on something that well be explained in detail later. In fact, a color will be assigned to each point taking into account the distance from the sensor. If the system RGB is adopted, the result is a saturation of the color and points which are close to each other are displayed with the same color. This is something that we don't want since the color to the points contributes to create a more realistic three-dimensional effect;

- Line 26-32, initialization of the vector in which data are then stored. Establishment of the Serial Port communication;

- Line 33, this part is responsible to draw the points, in the proper position, according to the coordinates. In order to make it easier to understand and intuitive, the last point that is displayed on the screen is connected by means of a red line to the origin of the axis. In this way, it's possible to follow the progression in the representation of the point cloud;

- Line 66-111, this part of the code is responsible of the navigation system. By pressing the keyboard arrows is possible to rotate the point cloud around a vertical or horizontal axis. While, pressing W,S,D,A buttons is possible to translate the point cloud horizontally or vertically. Finally, pressing the Q is possible to zoom in, while Z corresponds to zoom out. Last but not least important, P erases all points and S saves the point cloud to file;

- Line 113-125, is the part related to the function save to file. By pressing the S button, the function save is invoked. The file in output will contain the three-dimensional coordinates, the file will be saved in Processing v3 installation folder with a name "yyMMdd_HHmmss.xyz".

Results In this paragraph are reported samples from the first test carried on with LiDAR platform v1. As it's easy to understand, at the very beginning the processing sketch was not able to display colors (See Fig. 4.8 and Fig. 4.9). While Fig. 4.10 represent the evolution to colors. As, said before, colors help to identify better objects and give a more accurate three-dimensional aspect. Anyway, it's necessary to be honest and say that this first platform has given a result which is not good as expected. In fact, the density of the point cloud is not sufficient to give us a precise idea of the object or obstacle. The only thing that we can infer is the approximate shape of things or rooms. This prototype just underlines and shows which is possible to have a 3D representation of a surrounding environment employing a simple hardware as we did. But, for sure, the platform requires improvements, as well as upgrades. The first noticeable thing is the point

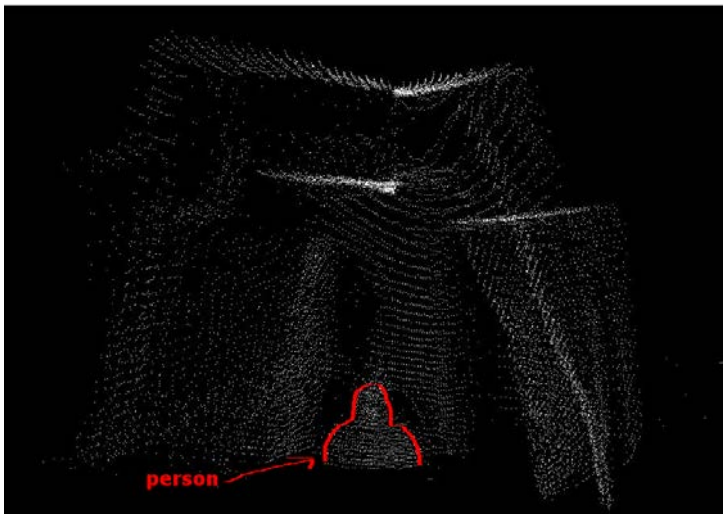


Figure 4.8: Scan 1

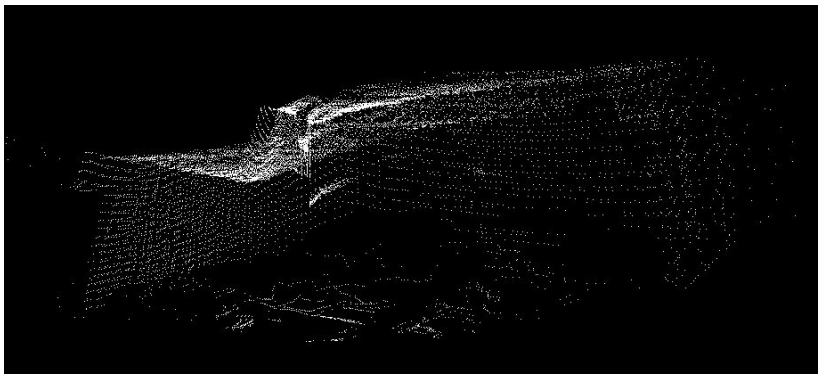


Figure 4.9: Scan 2

cloud density: it's not enough to show with precision the scanned objects. For this reason, the next version should improve dramatically this parameter. In the following section new proposal to solve this problem will be presented.

4.5 Towards LiDAR v2

The first step, has been to find out the critical parts of our project, starting from a free flow of thoughts or brainstorming. But, as well known, is not possible to randomly presents the list that has emerged, so the second step has been to sort that list. From the most critical to the less critical. In this way is possible to establish a priority and fix these problems following a well-defined path and process. It's not possible to think that the

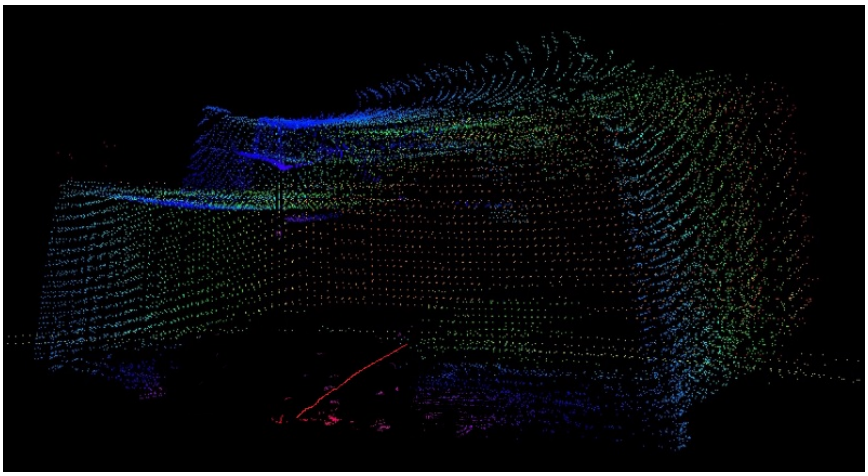


Figure 4.10: Scan 3

Rank	Priorities
1	Code – algorithm (Servo control, Serial Data Transmission, Data Processing)
2	Motors
3	Structure

Table 4.2: Table of enhancement

improvement of the project only relies on a better hardware. Rather than that, the hardware upgrade must be followed by an improvement in the code and the algorithm. So many times, the experience has showed us that the hardware limits (constraints related to a limited amount of resources, such as money and time) can be overcome by software manipulation and smart solutions. Following this approach, we identified three fields of interventions:

1)

a. Servo control technique: our technique seems to be not so reliable and it presents some unsolved problems and bugs (for example the execution of an angle greater than 120°). Or another issue is related to the possibility to reduce the angle between two steps. Right now, it's approximately around 1°, but in following version we would like to bring this value to 0.5° or smaller. This, could lead to a more efficient and accurate scanning performance;

   - + More efficient, accurate scanning technique;
   - + It's possible to get a 3D representation of small objects or details;
   - - Non linearities. When the step is smaller than 1° we're not sure about the real entity of the servo movement;
   - - Lack of accuracy and reliability.

b. Serial COM: a reliable and affordable serial data transmission permits a real data acquisition and the possibility to show a real time point cloud. Right now, the serial communication presents some problems and it's not so stable. In fact, there are some errors in the data format, so this implies a more difficult data elaboration and as consequence the process will be more expensive in terms of time and less stable.

   - + No dead time between data acquisition and processing;
   - + Show Real Time point cloud;
   - + More effective and efficient way to operate the scan.

c. Data processing: implement a better data elaboration means to have a precise idea of what you're scanning. So, a smart data processing involves an advanced algorithm able to discern which points are useful and which are not (errors) and showing them in an easy, immediate and understandable way. Another issue which is close to that is to assign to each point the right color based on the distance from the detecting device.

   - + Real Time Solution;

- + Easy to display.

All of those three options, software related, are not cost effective, since there's no need to buy anything but just programming and code. Moreover, it's true that on the internet it's possible to find some open source software that can guide you in the development of a new application or a new software part. But at the same time, we should say that we should spend a lot of time to make this process smooth and reliable. Anyway, regarding the motors, it's necessary to evaluate if a different technology could bring to better performances.

2)

a. To reach a high level of accuracy it's possible to buy new servos with better performances and specs. The main problem that we must face is the instability of the PWM signal to the servos. In fact, we have tested the servo with an oscilloscope and we have found out that the PWM signal is not stable and oscillates. This can lead to servo rotations that are not totally controllable and at the same time uncertain. Considering these problems, it's a wise decision to take into account different motor technologies, for example stepper motors. Stepper are used in critical applications where the minimum required step is really small or in task where the precise positioning is an issue. With an improvement of the motor performances it's possible to scan more accurately the surrounding environment.

   - + Accuracy;
   - + Reliability;
   - + Increase the performance of the whole system;
   - - Time and Cost, expensive (especially switch to other technologies);
   - - Test in lab;
   - - Slow down the scanning speed (↑ accuracy ↓ speed).

3)

a. The mechanical structure is very important because it has to prevent the diffusion of vibrations and place the sensor in a precise position, well fixed and still. The idea is to design a more robust and reliable, stable and safe structure. One thing that we want to avoid is glued parts (we only accept glued parts if the structure is a demo or trial version). The final aim will be to design a three-dimensional model of our structure and manufacture it. The future structure will be made of metal and we will put a specific attention to make it compact and

to distribute the mass as close as possible to the axis of rotation of the servos. Moreover, re-designing the structure we will try other solutions and a different allocation of the motors.

- + Faster scanning process;
- + Robust;
- + Customized (suits perfectly);
- - Requires time and money to re-adjust;
- - high level of complexity related to the manufacturing process.

Chapter 5

LiDAR platform v2

This is, probably, the most important chapter of this work. In fact, it's fundamental to show that we understood problems and issues emerged in the first version. And furthermore, we have been able to propose a possible solution that suits our requirements and allows to reach the expected quality standard. The decision that has been carried out consists in the radical change from servo motors to steppers. This solution brings as consequence the need to re-write the code that control pan and tilt movements, adapting it to the new technology that has been implemented. Beside that, the other parts will substantially remain the same.
Just to summarize, the big change is related to hardware. This, involves big changes in the software part related to motion control. Other small software implementations will be carried on in this phase, but the impact on the final products is really limited or not substantially relevant. Anyway, considering big changes, it's mandatory to conceive a new set of experiments, able to reveal the reliability and the performances of the new motion system in particular.
In the second version of our LiDAR, we adopted stepper motors. On paper, this kind of motors can execute very small rotations. If we take for example, a stepper motor NEMA 17[1] it can deliver a minimum rotation of 1.8° degrees. If a stepper motor is coupled with a micro-stepper driver is possible to obtain fractions of this angle (typically 1/2, 1/4 , 1/8, 1/32). Since the micro-stepper represents an open loop control there's the need to check and verify the mechanical accuracy of the motors.

Experimental Structure As consequence of what said before, we are really interested in the validation of the new pan & tilt movement system. To achieve this, we need to set up a mechanical experiment able to testify the accuracy of the stepper's movements. Which are the main characteristics

[1]NEMA 17 stepper motor is a stepper motor type specified by NEMA. NEMA 17 steppers have a 1.7 x 1.7 inch faceplate.

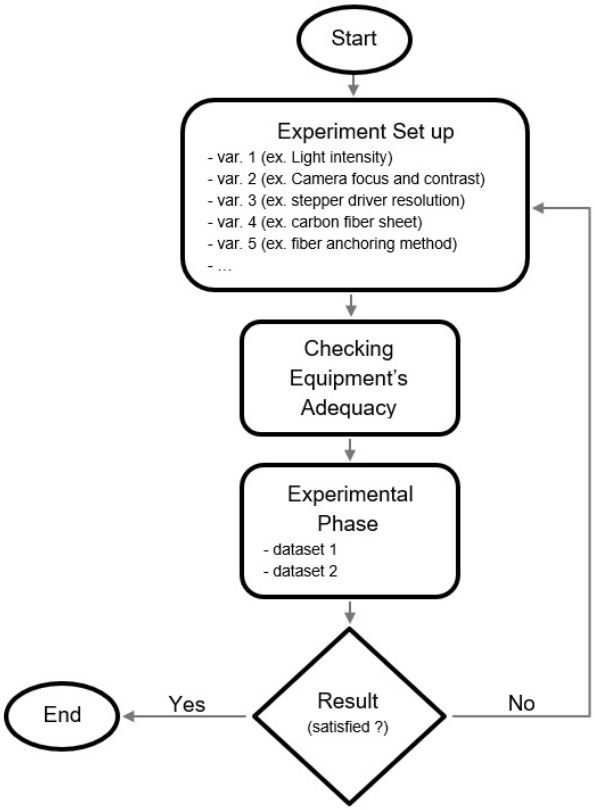


Figure 5.1: Experiment step by step

and features for this experiment?

- able to measure precisely;
- be accurate;
- reliable;
- give a repeatable result.

For the experiment we want to follow a precise and well-organized mind set (See Fig. 5.1). Thanks to that, it will be possible to reduce the failure rate of the experiment to values close to zero. And at the same time, by controlling the variables that can affect the result we can get more accurate and reliable results.
Starting from the problem and setting the goals of our experiment we can proceed defining the set up. With the word set up we mean the total equipment that we need and the whole procedure that we necessarily need to follow to get a successful experiment. At this stage, it's at the same time important to take into account external factors which can affect the results,

such as temperature, presence of dust, humidity, etc. If one of the things we have just mentioned (and others) will influence in some ways our activity, we must consider how and when mitigate their effect. Next step, it's essential to check the functionalities of the equipment available, calibrate the instruments and correct some biases due to non-zero errors or non-linearities.
When this stage has handed, it's necessary to write a list containing the phases that, step by step, constitute our experiment. Strictly following the process guarantees that the measure will be reliable and repeatable. If then the result satisfies what was set as the goal of the experiment it's possible to say that the process has ended, while if the data elaboration gives unexpected results it's necessary to reiterate and try a better set up for the experiment.

5.1 Stepper Motors

Stepper Motors are used in a wide variety of devices ranging from 3D printers and CNC machines to DVD drives, heating ducts and even analogue clocks. Stepper motors are DC motors that rotate in precise increments or "steps". They are very useful when you need to position something very accurately. They are used in 3D printers to position the printhead correctly and in CNC machines where their precision is used to position the cutting head.
Unlike DC motors, stepper motors are controlled by applying pulses of DC electricity to their internal coils. Each pulse advances the motor by one step or by a fraction of a step, the latter is known as "micro-stepping" and will be explained shortly. Some people confuse stepper motors with servo motors, but they are actually two different things. A servo motor is unique in that its motor shaft can be moved to a precise angle, most servos only rotate 180 or 270 degrees. A servo motor is "aware" of its position (as seen in the previous chapter) and can be moved to a specific angle even if an external force moves the motor shaft.
Steppers, on the other hand, are "unaware" of their position. They can be moved to an exact position in reference to where they start stepping but unlike servos they can be misaligned if their shaft is moved by an external force. In many applications a servo is first moved to a "homing" or reference position before being controlled. Because the move in discrete steps, a stepper motor is not often used where a smooth continuous rotation is required, however with the use of gearing and micro-stepping they can approach a smooth rotation and their ability to be very accurately positioned often outweighs the roughness of their movement.
Another advantage stepper motors have over DC motors is the ability to move at very slow speeds without stalling. They also have a high-torque density, so it means that they can provide a high torque if compared to

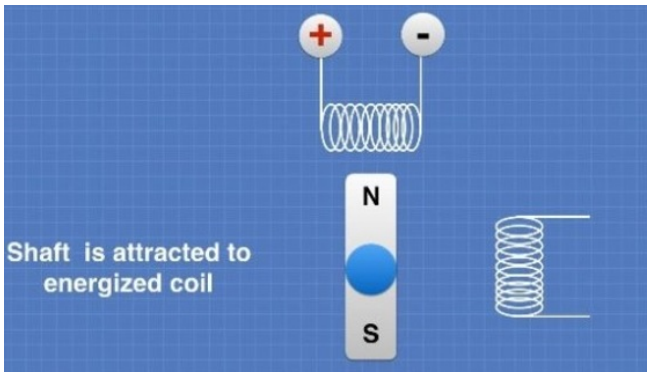


Figure 5.2: Working principle Stepper (a)

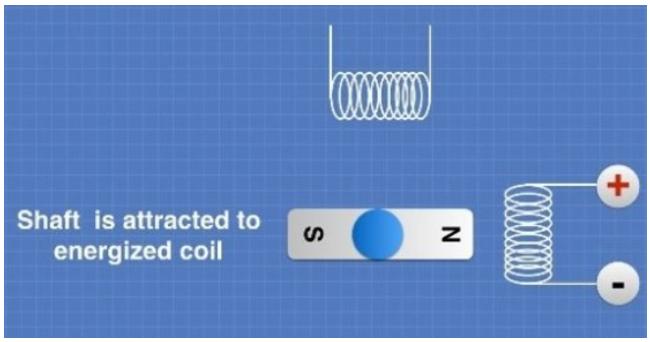


Figure 5.3: Working principle Stepper (b)

their size.
Stepper motors have a magnetized geared core that is surrounded by a number of coils which act as electromagnets. Despite the real number of coils, electrically, we can draw a simplified scheme that comprehends only two coils in a stepper motor, divided into a number of small coils. By precisely controlling the current in the coils, the motor shaft can be made to move in discrete steps, as illustrated in the following diagrams (Fig. 5.2): In the first diagram the coil at the top is energized by applying electricity in the polarity shown. The magnetized shaft is attracted to this coil and then locks into place. Now look what happens when the electricity is removed from the top coil and applied to the other coil (Fig. 5.3). The shaft is attracted to the second coil and locks into place there. The jump between the two positions is one step (in this illustration a step is 90 degrees, in actual fact a stepper motor usually steps just a fraction of this. The diagrams are simplified for clarity). We have seen how the motor shaft moves to lock itself into place in front of an attracting electromagnet, each magnet represents one step. It is, however, possible to move the motor shaft into positions between steps. This is known as "micro-stepping". In order to understand

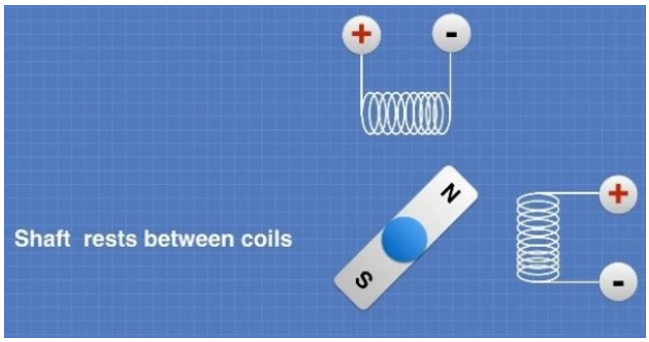


Figure 5.4: Working principle Stepper (micro stepping)

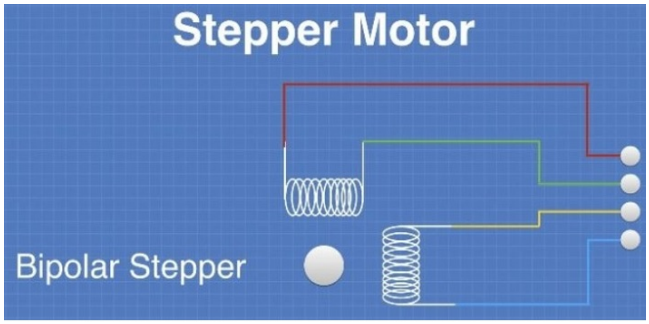


Figure 5.5: Bipolar Stepper

how micro-stepping works, it's possible to See Fig. 5.4: In this illustration the current has been applied to both coils in an equal amount. This causes the motor shaft to lock into place halfway between the two coils. This would be known as a "half step". The principle can be extended to include quarter steps, eight steps and even sixteenth steps. This is done by controlling the ratio of the current applied to both coils to attract the motor shaft to a position between the coils but closer to one coil than the other. Commonly this operation is executed by a micro-stepper driver: choose a good quality driver is quite important to reduce vibrations and for smooth operations. By using micro-stepping, it is possible to move the shaft of a stepper motor to a fraction of a degree, allowing for extremely precise positioning.
Actually, it's possible to divide stepper motors in two categories: bipolar and unipolar. Since we don't want to explain everything in detail I'll just talk about bipolar stepper (the one we are interested in and the one on which the experiment is based and conceived). Bipolar stepper motors, See Fig. 5.5, consist of two coils of wire (electrically, actually split into several physical coils) and generally have four connections, two per coil. The simplified diagrams of stepper operation presented in the previous section are all bipolar stepper motors. An advantage of bipolar stepper motors is that they make use of the entire coil winding, so they are more efficient.

However, they require a more complex controller or driver to operate, as to reverse direction the polarity of the voltage applied to the coils needs to be reversed.

5.1.1 Micro Stepper driver and wiring scheme

These chips keep the power that drives the motors separate from the power that is on the Arduino (in our case, but others micro-controller can be used). The Arduino can't provide enough juice to power the stepper motors directly. This is why it's necessary to use separate chips to sort of act as valves that control how the motor spins. Another benefit that stepper driver chips provide, is that they provide fractional steps, as mentioned before. This helps smooth out the motion of the stepper motor. Without fractional steps, stepper motors have a tendency to vibrate or resonate at certain RPMs. To run a stepper motor, two things are normally required: A controller to create step and direction signals (at ± 5 V normally) and a driver circuit which can generate the necessary current to drive the motor. In some cases, a very small stepper may be driven directly from the controller, or the controller and driver circuits may be combined on to one board. The stepper controller drives 3 wires – traditionally labelled PUL (or pulse), DIR (direction), GND (voltage reference, sometimes, like in our case, the GND pin is substituted by the VCC pin and requires a stable 5V voltage reference from the microcontroller) – which carry motion information to the stepper driver. Typically, these 3 lines are opto-isolated at the front end of a stepper driver. The stepper controller is typically a pure digital logic device and requires relatively little power. The stepper driver (See Fig. 5.6) connects to the 4 thick wires of the stepper motor. It contains the power transistors and requires a thick power cable to a DC power supply, because all the power to drive the motors runs through it. The following scheme represents the wiring diagram of the stepper controller, the stepper driver and the power source. This sub part of the system is the most important one, since it's the one on which we want to focus our attention. Once, we have explained how the stepper module works we can represent the general set up of the whole experiment and finally present the results we have gathered. As it's possible to see on the front cover of the driver (Fig. 5.7), two tables are reported: the first one describes the configuration of three switch (SW5, SW6, SW7) that determines the micro-stepping function. The other table reports the maximum current that the driver can use to feed the stepper. This value depends on the specifications and features of the stepper itself. So, for this choice it should be taken into account the stepper datasheet, and then set the position of other three switch (SW1, SW2, SW3). Last but not the least is SW4 responsible to prevent the motor to be damaged by current spikes. By writing a specific code compatible with the microcontroller, and by setting carefully the micro-stepping configuration on the driver, is possible to

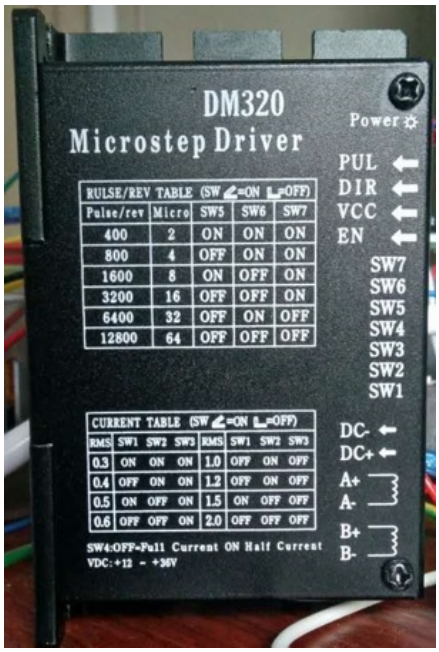


Figure 5.6: Micro Stepper Driver (DM320)

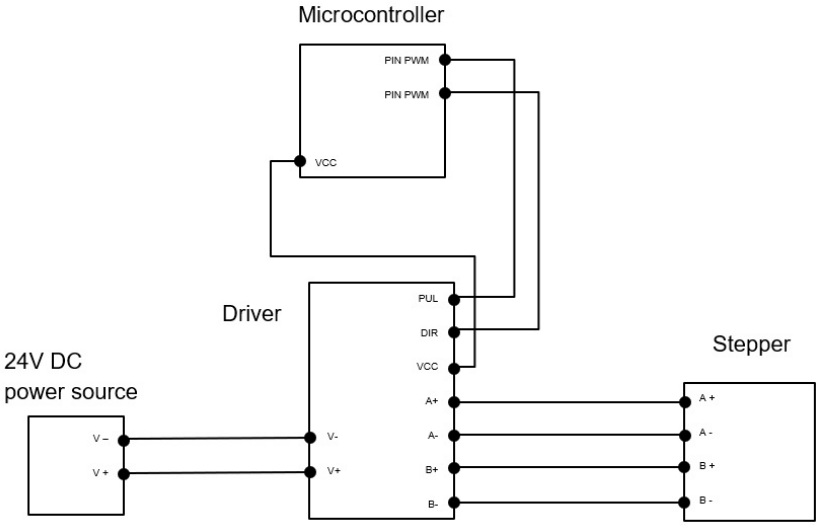


Figure 5.7: Stepper Module

. Captures full resolution of megapixel cameras . Low distortion (Less than 1.0%) . Locking set screws for focus and iris	. High contrast & sharp picture in all areas of the screen . Compact design. Diameter 33.5mm (H0514-MP: 44.5mm)

Table 5.1: Camera Features

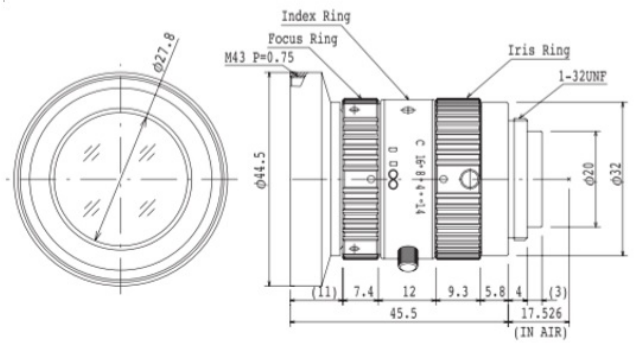
control and command the stepper motor. In our case we use Arduino as microcontroller and we flash the board with a specific IDE. Other microcontrollers can be used, getting a pretty similar result. Right now, it has been explained how a stepper and a stepper control work, so it's possible now to go further presenting the experiment.

5.2 Experiment Set up

Actually, we can say that the experiment's aim is to evaluate a mechanical entity, which is the minimum rotation executed by a stepper motor. But how to do so?
The question can have multiple answers. The approach that has been followed in this paper is practically opto-mechanical. In other words, the experiment can reveal, thanks to a high-resolution camera, the orientation of a carbon fiber pre-preg. By varying the orientation of the fiber sheet is possible to detect small rotations through the CCD sensor of the camera. Before starting the measure acquisition process is fair to pose ourselves some questions: is the system reliable? Is the system able to detect the minimum angle that we are looking for? So, in other words, which are the limits of the experimental set up. Since it's the first time that I personally use this equipment, I cannot answer these questions in advance. I will dedicate a specific section related to the topic after the data will be presented.

5.2.1 Camera specifications and features

The CCD sensor that is used in the experiment is a MegaPixel Lens Camera. It is manufactured by Computar, a Japanese company and presents the characteristics reported in Tab. 5.1. On the camera's body are present two sliding gears: the lower one is responsible of the accommodation process and the upper one is responsible to set the quantity of light that flows through the CCD sensor. In other words, the first lens is relative to the accommodation process that modifies the focus of the camera by varying the distance of the lenses and the second one is a diaphragm. These two processes, actually, replicate what the human eye does (See Fig. 5.8).



Focal Length		5mm			
Max. Aperture Ratio		1:1.4			
Max. Image Format		6.4mm x 4.8mm (Φ8mm)			
Operation Range	Iris	F1.4 - F16C			
	Focus	0.1m - 0.9m			
Control	Iris	Manual			
	Focus	Manual			
Object Dimension at M.O.D.		15.0(H)cm x 11.1(V)cm 1/2"			
Angle of View	D	1/2"	76.7°	1/3"	62.3°
	H		65.5°		51.4°
	V		51.4°		39.5°
Operating Temperature		- 10° C ~ +50° C			

Distortion	1/2"	-0.48% (y=4.0)	1/3"	-2.26% (y=3.0)
Back Focal Length	10.8mm (WD=900mm)			
Flange Back Length	17.526mm			
Mount	C-Mount			
Filter Size	M43 P=0.75mm			
Dimensions	Φ 44.5mm x 45.5mm			
Weight	102g			

Figure 5.8: Camera drawing & features table

5.2.2 Carbon fiber single layer pre-preg

As we have mentioned before, a special importance for the experiment is related to the carbon fiber pre-preg. Pre-preg is "pre-impregnated" composite fibers where a thermoset polymer matrix material, such as epoxy, is already present (See Fig. 5.9). The fibers often take the form of a weave and the matrix is used to bond them together and to other components during manufacture. The thermoset matrix is only partially cured to allow easy handling; this B-Stage material requires cold storage to prevent complete curing. B-Stage pre-preg is always stored in cooled areas since heat accelerates complete polymerization. Hence, composite structures built of pre-preg will mostly require an oven or autoclave to cure. The main characteristic of the pre-preg lies in the fiber orientation: as said before fibers are stretched along the sheet to form a straight line that is than drowned into resin. Thanks to that the structure is well organized and defined.

5.2.3 Equipment List

For the complete equipment list See Tab. 5.2.
While the Process (See Fig. 5.10) follows these steps:

1. The camera captures the absolute orientation angle of the fiber pre-preg;

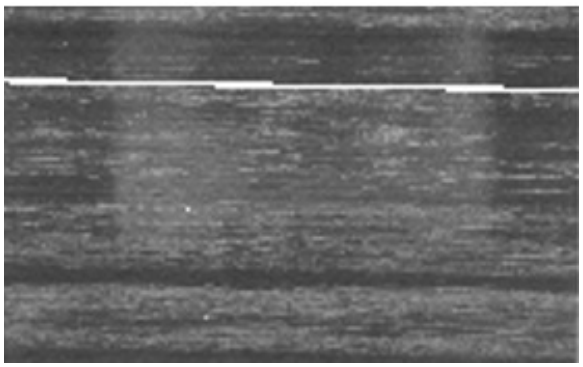


Figure 5.9: Carbon fiber pre-preg sample

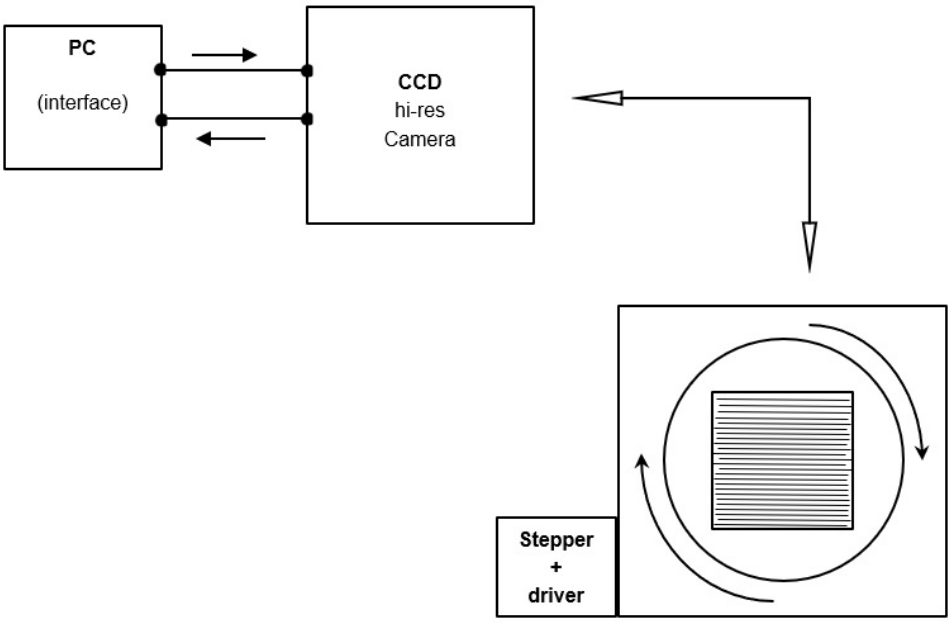


Figure 5.10: Experimental Set up

Item	Function
Structure	The aim of the mechanical structure is to hold the CCD sensor (high-res) camera in place, as well as to position the metallic plate fixed to the stage (following).
Rotary Stage	A rotary stage is a tool that couples a stepper motor with a gearing system and transforms the mechanical rotation of the stepper's shaft to a high precision rotation of a mechanical stage. The stage gear ratio is 1:90; Size of the stage plate: 60mm;
Stepper motor & driver	Nema 17 1.8° minimum rotation Maximum current: 1.3 A Torque: 0.22 Nm Code: 42HAP34BL4 The motor is then coupled and driven by the DM320, showed and presented above. If that motor is coupled with a driver that provides 1/20 micro-stepping, considering the gear ratio of the stage, it's possible to get 0.001° minimum rotation of the stage.
CCD Camera	In a CCD image sensor, pixels are represented by p-doped metal-oxide-semiconductors (MOS) capacitors. These capacitors are biased above the threshold for inversion when image acquisition begins, allowing the conversion of incoming photons into electron charges at the semiconductor-oxide interface; the CCD is then used to read out these charges. More details in the dedicated section.
Software interface	The software interface triggers the acquisition of the image and saves the result in the pc. Then the picture is processed and the inclination of the fibers is detected and showed on the screen.
Single layer Carbon Fiber prepreg	This is the tool that reveals the rotation. Thanks to its stretched profile it's possible to detect the rotation of the fiber positioned firmly on the stage.

Table 5.2: Equipment List

2. The stepper motor steps by a certain number of steps (this number is set up precisely by coding the micro-controller) as previously described. At the same time, the stage will rotate, and the orientation of the fiber will change. Now, it's necessary to take into account the fact that one step of the stepper motor corresponds to 1:90 degrees rotation of the platform if the micro-stepping is avoided. Conversely, the angle should be adjusted dividing by the micro-stepping factor.

   Examples:

   Stepper minimum step 1.8°;
   no micro-stepping /1;
   Gear ratio: 1/90;
   ⇒ rotary stage rotation per step: 0.02°;

   Stepper minimum step 1.8°;
   micro-stepping 1/2 ;
   Gear ratio: 1/90;
   ⇒ rotary stage rotation per step: 0.01°;

   Stepper minimum step 1.8°;
   micro-stepping 1/4;
   Gear ratio: 1/90;
   ⇒ rotary stage rotation per step: 0.005°.

3. The camera is triggered a second time to capture the new picture of the fiber orientation;

4. Thanks to simple calculation is possible to obtain the Δ rotation (between status 1 and status 2). See Fig. 5.11 to have an intuitive and visual idea of the whole process.

By knowing the number of steps that the stepper has executed and taking into account what has been said in point 2), it's possible to make a comparison between the expected value and the value that has been measured by the camera with the second picture. One thing is valuable to be underlined: if we analyze the process we understand that the measure does not depend on the absolute values gathered by the camera. In fact, we are interested in the difference between the two values rather than the absolute numbers. This means that we can tolerate some errors related to non-zero alignment, since the error will, presumably, affect the two measures in the same way. This is a very good result, in fact we have as consequence a stable and reliable method to measure what we are looking for. The only thing that is unknown at this point is if the camera will be able to detect a single step of the whole system. In that case, it won't be a problem let the motor step

by a precise number of steps (i.e. 10, 100, 1000, 10000, etc.) and then check the amplitude of the average step, by simply dividing the angle that is measured with the camera by the number of steps. Last but not the least we need to make some annotations. Talking about the experimental set up, it's necessary to say that:

- The fiber should be well positioned and fixed on the rotary platform, that lies on the rotary stage. In detail, four pieces of thin tape has been attached on the plate and the Pre-Preg sheet has been firmly anchored;
- The Pre-preg sheet has been cleaned with some alcohol in order to remove some pollution and dirt from the surface. This step requires patience since it's very easy to damage the carbon fiber layer. So, if the sheet is damaged the software struggles to recognize the fiber pattern and at the same time the output measure is not accurate;
- A special light has been used to illuminate the fiber sample, in this way it's possible to increase the contrast factor and this helps a lot in the patter recognition of the fiber;
- As we have mentioned before the camera presents two sliding gears. It's important to calibrate the focus and the diaphragm aperture to increase the resolution and contrast of the picture.

5.3 Experimental Data

In addition of what reported before, we can spend some more words on the methodology that will be used during the data gathering process. First of all, a micro-stepping factor will be set, and for each f_{ms}, 4 or 5 measurements will be taken, especially the ones related to 10-100-1000-10000 steps. In that way, we want to test if the stepper is accurate, but moreover we want to underline which are the boundaries and the limit of our set up. Only then, it will be possible to make a diagram that expresses $\frac{\Delta}{n_s}$ versus n_s itself. Just to be clear and complete, the process already presented in the previous chapter will be explained with pictures (only one measurement) and then the other data will be reported in charts.

5.3.1 Data Report

The first set of measurements is obtained by using a micro-stepping factor equal to $1/2$. As shown in the following pictures, the camera has been triggered twice: the first time, as reference, while the second time, after the rotation, to measure the new angular position of the fiber. In this case the

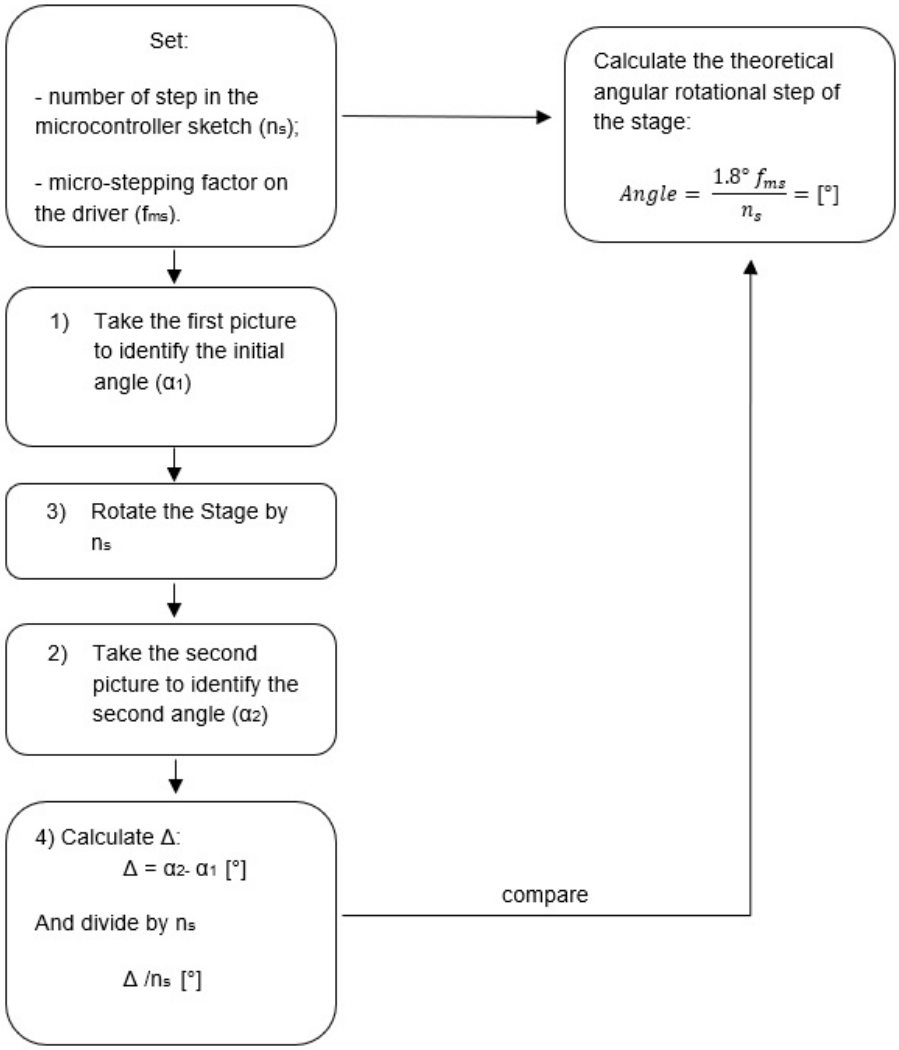


Figure 5.11: Computational Process

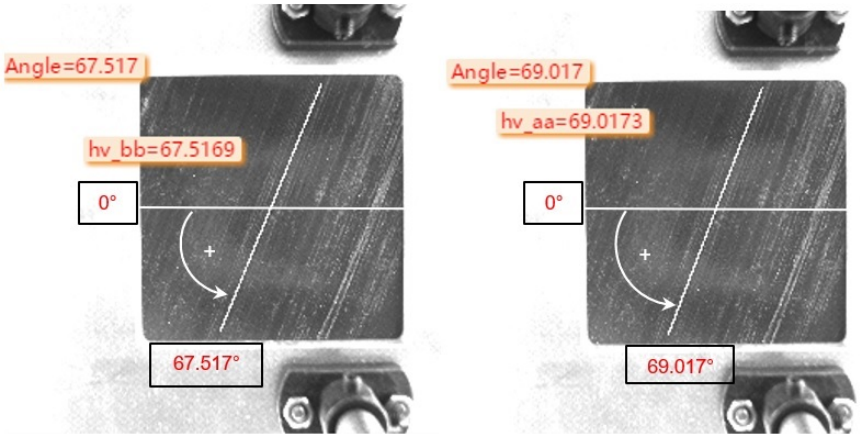


Figure 5.12: Measurement Sample

number of step that has been chosen is 5485. So:

$$\begin{aligned} Measure_1 &= 35.1457\ [^o]; \\ Measure_2 &= 88.1025\ [^o]; \\ Result &= 52.9568\ [^o] \quad \Delta(Measure_2 - Measure_1); \end{aligned}$$

Right now, the next step is to calculate the average amplitude of one step. The operation in pretty easy, it's just necessary to divide Δ by the number of steps n_s.

$$\Rightarrow \frac{52.9568\ [^o]}{5485\ [step]} = 0.00966\ [^o/step]$$

With simple math is possible to calculate the theoretical stepping angle and the relative error:

$$\frac{1.8\ \frac{1}{2}}{90} = 0.01$$
$$\frac{(0.01 - 0.00966)}{0.01} \times 100 = 3.431\%$$

As it's possible to see, the data are quite reliable (See Tab. 5.3). That means that the experimental set up is able to detect what we wanted to measure. As already mentioned, in the following data analysis we will underline the boundaries of the system. By the way, the data prove that the stepper is a good solution to implement and integrate to the LiDAR project, since it's possible to obtain small and accurate rotations. Before showing the second set of data measurements, it's possible to show and present some diagrams. In this way it will be possible to have an immediate idea of the data set. In the following diagrams (Fig. 5.13 and Fig. 5.14) will be showed the

N_{step}	M_1	M_2	$\Delta(M_2 - M_1)$	Avg_{step}	Error (%)
1	14.1264	13.9957	- 0.1307	0.1307	1207 %
10	13.4796	14.1264	0.6468	0.06468	546.8 %
100	12.8213	13.7613	0.94	0.0094	6 %
125	67.5169	69.0173	1.5004	0.012	20 %
250	65.0771	67.5169	2.4398	0.0097592	2.408 %
500	59.9773	65.0771	5.0998	0.0101996	1.996 %
1000	10.8079	21.7317	10.9238	0.0109238	9.238 %
5485	35.1457	88.1025	52.9568	0.00966	3.431 %
10000	2.09782	101.1254	99.02758	0.009903	0.9724 %
18000	2.81901	182.29837	179.47936	0.009971	0.29 %

Table 5.3: Data Set 1, $f_{ms} = 1/2$

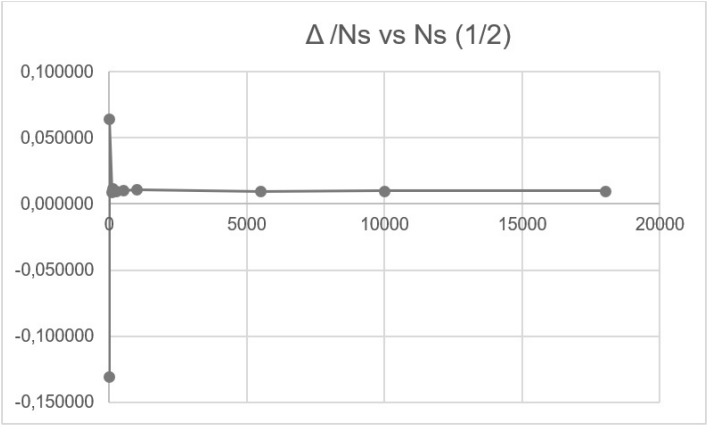


Figure 5.13: $\Delta/_{N_s}$ *vs* N_s $(1/2)$

correlation between the average micro-step and the total number of steps. In this way it's easy to see that the asymptotic value is quite close to the expected value (theoretical step, as computed before). The second diagram (Fig. 5.14) shows a zoomed area close to the origin of the axis. The system turns out to be not accurate considering small rotations and small angles. Especially for a number of step that is smaller than 100. That means that when the rotation is smaller that 1° the experimental set up struggles to detect the orientation of the fiber. The next step is to present the second data set, to see if the data are confirmed or not and then propose a motivation of the inaccuracy of the measurements. As said, the second set will be acquired setting the micro-stepping factor to 1/4 (Tab. 5.4). And the relative diagram is the one reported in Fig. 5.15. In this case too, the experiment shows an inadequacy to detect very small rotations, but asymptotically it reaches the expected value of 0.005°.

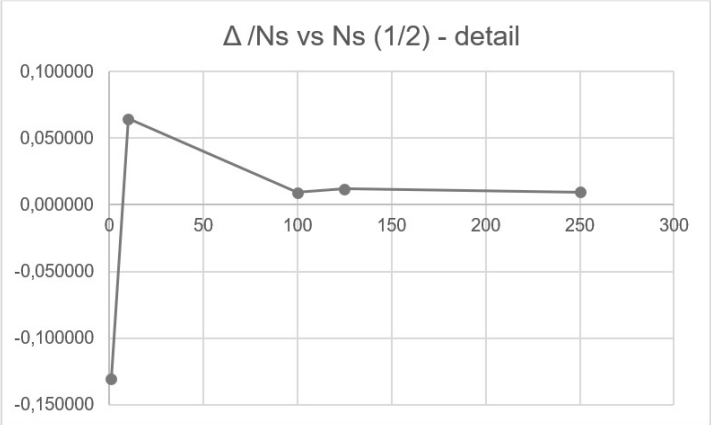


Figure 5.14: $\Delta/_{N_s}$ *vs* $N_s\,(1/2)$ - Detail

N_{step}	M_1	M_2	$\Delta(M_2 - M_1)$	Avg_{step}	Error (%)
10	23,0919	23,0354	-0,0565	-0,005650	43,5000%
100	22,9508	23,0919	0,1411	0,001411	85,8900%
200	31,5734	32,5250	0,9516	0,004758	4,8400%
1000	2,2984	6,1492	3,8508	0,003851	22,9832%
9000	59,9773	104,8369	44,8596	0,004984	0,3120%
10000	6,1492	56,9976	50,8484	0,005085	1,6968%

Table 5.4: Data Set 2, $f_{ms} = 1/4$

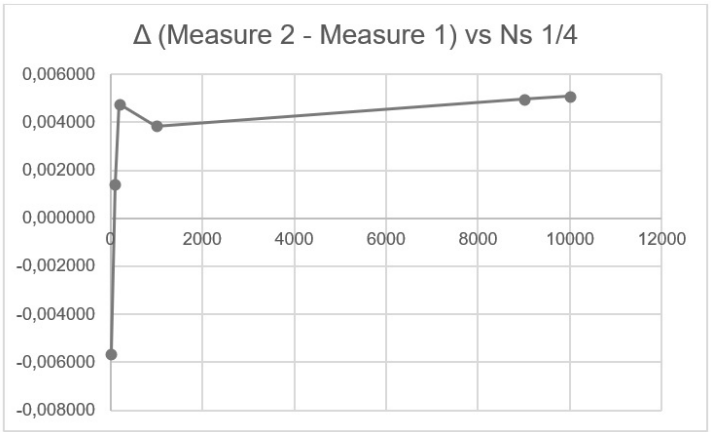


Figure 5.15: $\Delta/_{N_s}$ *vs* $N_s\,(1/4)$

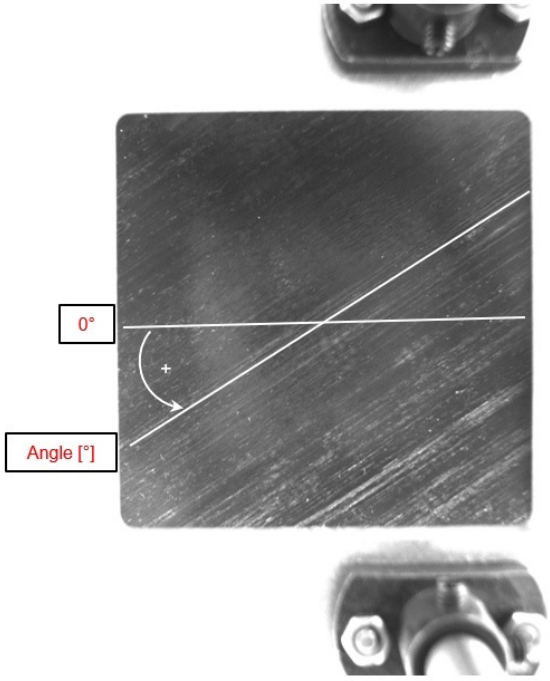


Figure 5.16: Carbon Fiber Sample Detail

5.3.2 Conclusions

- The data show that the system is reliable and helps us to check what we wanted to check. In this way is possible to test the accuracy of a stepper motor. This was the initial goal;

- It's possible to see that the values are quite similar to the ones we expected to find, at least for a sufficient big number of steps. It's possible to inflict that, when the rotation of the stage is smaller than 1°, the relative error is huge. A possible answer could be related to the average operation. In fact, with small angles the error is split on a small number of steps, while with a bigger angle the error, related to the detection of the carbon fiber orientation, is split on a big number of steps. This bring us to think that the algorithm that is used to identify the carbon fiber pattern could be revised and it can become more robust and reliable. The algorithm has proved to be sufficiently efficient with quite wide rotations, not the same with small or super small ones;

- It's very important to set the camera with the right diaphragm aperture and focus, and try to increase as much as possible the luminance

contrast:

$$C_w = \frac{(L_s - L_b)}{L_b}$$

Where C_w indicates the luminance contrast, L_s the luminance of the target object and Lb the luminance of the background;

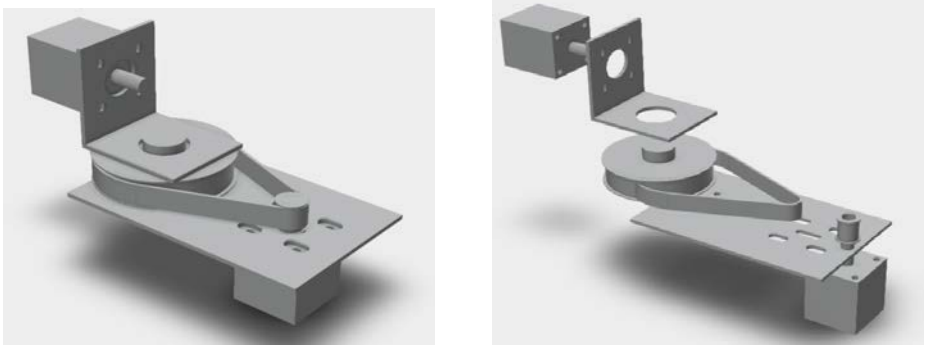
- It's important too to pay attention to the positioning of the carbon fiber on the metallic plate. Some vibrations, or accidental collision (with other structural parts) can modify its position and give wrong results;
- At the same time is relevant the quality of the carbon fiber that is used. Generally, when the carbon fiber is flexible and malleable it's more difficult to recognize the pattern. That's why is recommended to partially activate the resin through heat, UV light or other equivalent methods;
- The adoption of the stepper technology has proved to be the right solution to improve the density of the LiDAR point cloud. By coupling ad-hoc electrical driving systems and mechanical system is possible to obtain densities close to 1Million points, and at the same time guarantee a high level of accuracy.

5.4 New Structure

Another terrific improvement on LiDAR platform v2 is related to the structure. Actually, there's no much to say: the new structure is solid and more firm. It allows a better positioning of the sensor as well a better general structure stability. The structure has been firstly drawn in SolidWors^{TM}and then manufactured in Steel. In order to improve the overall system performances and reduce the minimum stepper angle we used a mechanical solution. On the pan stepper's shaft is mounted a gear system. The small gear is linked to the pan motor, while the big one is linked to a secondary shaft on which the a small platform is mounted. This platform has been conceived to be the housing of the tilt motor as well as the sensor's housing. Reducing the speed allows to reduce contextually the minimum angle that the pan stepper delivers. The two gears have a ratio 1:5 and they're mutually connected through a belt, specifically designed. Here we have some pictures: See Fig. 5.17a and Fig. 5.17b.

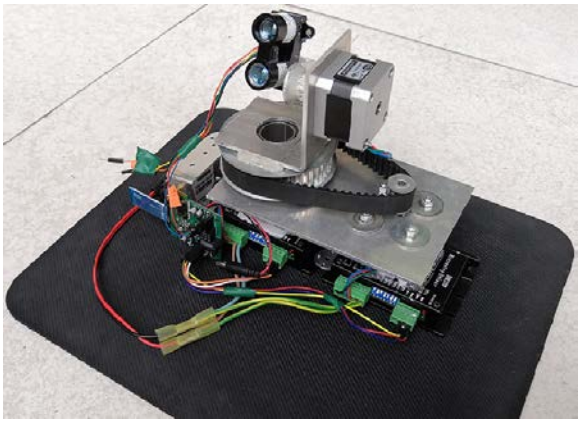
5.5 Results and Scan

In this section the final version of the LiDAR will be presented, showing the relative pictures (See Fig. 5.18a and Fig. 5.18b). Moreover, the results of further scan will be showed and then commented.

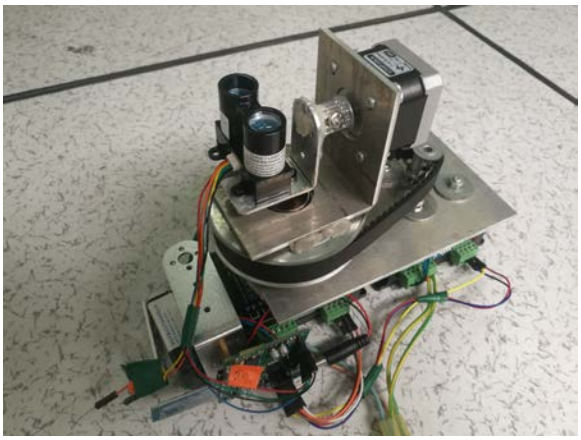


(a) Mechanical Assembly (b) Mechanical Assembly Exploded

Figure 5.17: Mechanical Assembly, Detail



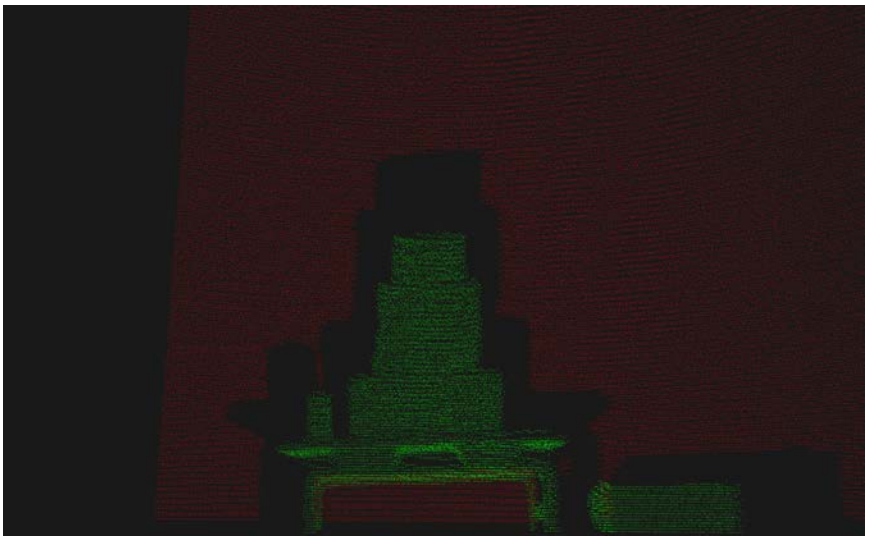
(a) LiDAR v2, Detail



(b) LiDAR v2, Detail

Figure 5.18: LiDAR v2

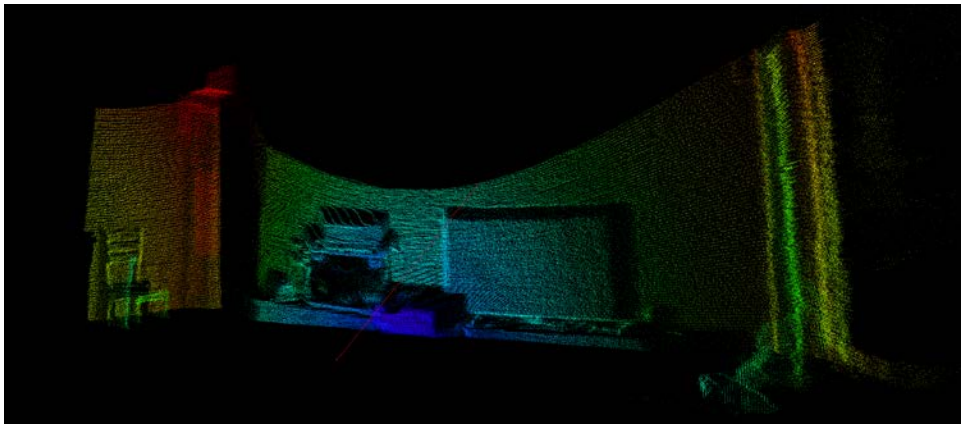
(a) Real Picture



(b) Point Cloud LiDAR v2

Figure 5.19: Table and Boxes Scan, v2

(a) Real Picture



(b) Point Cloud LiDAR v2

Figure 5.20: Living Room Scan, v2

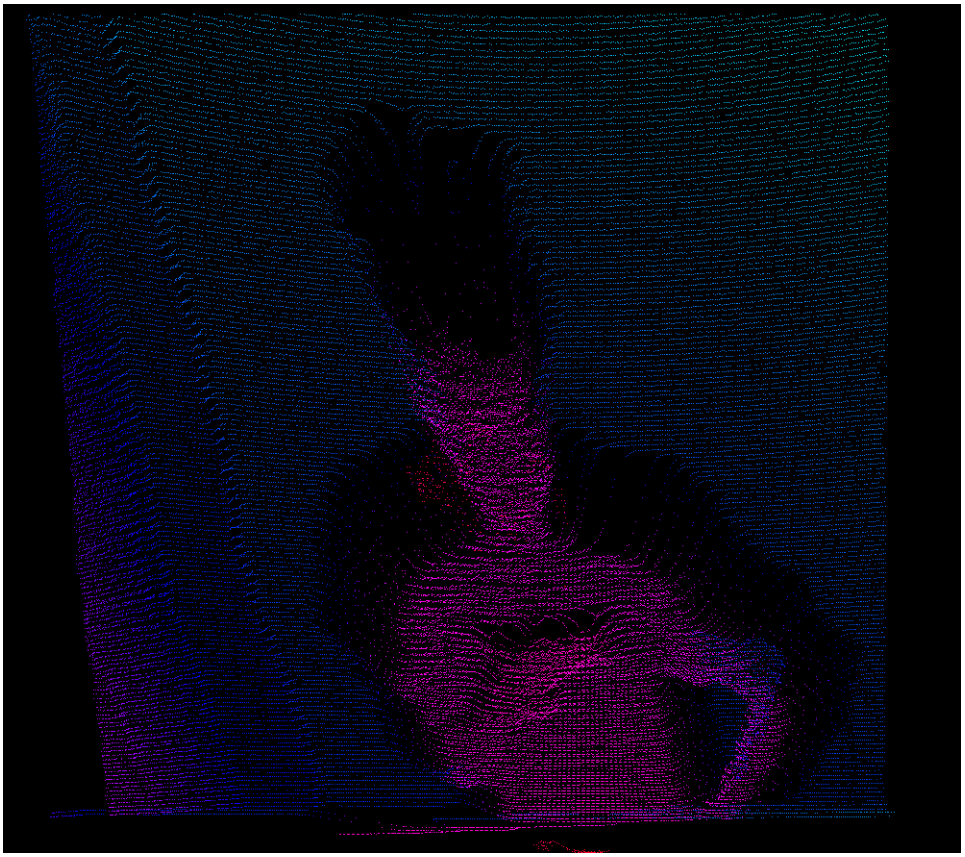


Figure 5.21: Human Body Scan

Chapter 6

Conclusions

This last chapter is fundamentally a recap of what has been treated and presented in this work. The main goal of the project was to develop a versatile LiDAR platform able to scan the surrounding environment with a decent accuracy and quality of the point cloud. We repeat another time that quality of the point cloud, actually means density: the more dense a point cloud is, the more accurate the final representation is. Anyway, the main goal was to meet these requirements, dealing with a constraint on costs. The final platform, indeed, should have cost roughly 250$. With the following list the whole process will be recapped and some conclusions or notes will be added:

1. We started from the basic knowledge, from scratch, studying the single components and parts of the project, learning the theory and trying to figuring out how to solve a complex problem as the one we had in input;

2. Finally, the LiDAR v1 has been designed first and then manufactured: the development of this platform has been challenging and required us to put together a vast knowledge and deploy multidisciplinary skills; basically related to servo motor control, Arduino servo management and integration, the LiDAR sensor and other various equipment. But it's not finished, indeed, the project required us to develop from scratch a brand new visualization environment which allows us to show the real time point cloud that represents the virtualization of the surrounding reality. This environment has been written on a Java platform called Processing v3 (virtual image processing system);

3. Unluckily, LiDAR v1's performances were not as good as expected. So, after the testing phase, we focused our attention on which were the limits of the first version and which were the updates or adjustments thanks to which LiDAR's performances could have improved. Only

then, the most efficient and profitable solutions has been adopted in version 2;

4. Then, LiDAR v2 has been released. This second version required an impressive effort, in particular due to the new pan & tilt motor module (brand new technology) and the design of the new platform' structure. As said, moving from servos to steppers has been a huge leap. So we had to face lots of unknowns and doubts. In order to understand if the stepper actually suited our application, we conceived a test carried on in the lab, aimed at testing these components, in terms of reliability, accuracy and performance. The test gave us the answers we were looking for. Beside that, as hinted, a new structure has been released. A final test phase then proved huge improvements in real word scenario representation.

5. Talking about the LiDAR prototype and its possible applications, it's quite easy to say 3D virtual scan and point cloud representation of interiors (what we did in the test phase). But then, with further improvements and updates that machine could be used as a low cost platform applicable to:

   – Urban planning and modelling (new infrastructures planning in critical existing contexts);
   – 3D Building modelling;
   – Infrastructure monitoring and control (status check of power lines, railways, ...). This last application is strictly related to my major and my academic background, so it takes on a truly interesting meaning.

Bibliography

[1] Robert K. Wysocki, Effective Project Management. Traditional, Agile, Extreme. Fifth edition(2009). ISBN 978-0-470-42367-7.

[2] LIDAR—Light Detection and Ranging—is a remote sensing method used to examine the surface of the Earth". NOAA. Archived from the original on June 4, 2013. Retrieved June 4, 2013.

[3] J. Young, Lidar for Dummies, (2011), Wiley Publishing Inc., ISBN: 978-0-470-94225-3.

[4] Cracknell, Arthur P.; Hayes, Ladson (2007) [1991]. Introduction to Remote Sensing (2 ed.). London: Taylor and Francis. ISBN 0-8493-9255-1. OCLC 70765252.

[5] M. Doneus, I. Miholjek, G. Mandlburger, N. Doneus, G. Verhoeven, Ch. Briese, M. Pregesbauer, "Airborne laser bathymetry for documentation of submerged archaeological sites in shallow water". ISPRS – International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Bibcode:2015ISPArXL55...99D.

[6] Chiu, Cheng-Lung; Fei, Li-Yuan; Liu, Jin-King; Wu, Ming-Chee. "National Airborne Lidar Mapping and Examples for applications in deep seated landslides in Taiwan". Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International. ISSN 2153-7003.

[7] G. Vosselman,H. Maas, (2012). Airborne and terrestrial laser scanning. Whittles Publishing. ISBN 978-1-904445-87-6.

[8] S. Lee, J. Joon Im, B. Lee, Leonessa, K. Alexander. "A real time grid-map generation and object classification for ground based 3D lidar data using image analysis techniques". Image Processing (ICIP), 2010 17th IEEE International Conference on image processing. ISSN 1522-4880.

[9] A. Jadhav, D. Gambhir. "Use of LiDAR in Extraction of 3D City Models in Urban Planning". September 1, 2009. Geospatial World Journal.

[10] J. A. Jakubiec, C. F. Reinhart. "A method for predicting city-wide electricity gains from photovoltaic panels based on LiDAR and GIS data combined with hourly Daysim simulations". 12 May 2013, ELSEVIER Solar Energy 93 (2013) 127–143.

[11] D. Lv, X. Ying, Y. Cui, J. Song, K. Qian, M. Li. "Research on the Technology of LIDAR Data Processing". The 28th Research Institute of China Electronics Technology Group Corporation Nanjing, China.

[12] NTSB 2016 Report:"Highway Deaths Lead National Increase in Transportation Fatalities". 11.27.2017. https://www.ntsb.gov/news/press-releases/Pages/PR20171121.aspx

[13] The picture is available here. https://www.continental-automotive.com/Landing-Pages/CAD/Automated-Driving/GlobalHighlights/3D-Flash-Lidar (Continental Website).

[14] Extract from here. https://www.cosworth.com/news/electronic-news/demonstrating-the-car-of-the-future-with-lidar-sensor-technology

[15] Lidar Lite v3 Operation Manual and Technical Specifications. GARMIN

[16] The I2C-Bus Specification Version 2.1. Philips Semiconductors. January 2000.

[17] ttps://www.sciencebuddies.org/science-fair-projects/references/introduction-to-servo-motors

[18] https://processing.org/

[19] Ira Greenberg (31 December 2007). Processing: Creative Coding and Computational Art. Apress. pp. 151–. ISBN 978-1-4302-0310-0.

[20] Jeanine Meyer (15 June 2018). Programming 101: The How and Why of Programming Revealed Using the Processing Programming Language. Apress. pp. 121–. ISBN 978-1-4842-3697-0.

[21] Ira Greenberg (25 March 2010). The Essential Guide to Processing for Flash Developers. Apress. pp. 412–. ISBN 978-1-4302-1980-4.

[22] Casey Reas and Ben Fry. Processing: A Programming Handbook for Visual Designers, Second Edition. December 2014, The MIT Press. 720 pages.