ZIENTZIA
ETA TEKNOLOGIA
FAKULTATEA
FACULTAD
DE CIENCIA
Y TECNOLOGÍA

Bachelor Final Thesis

Degree in Electronic Engineering

Design and implementation of an immersion 3D scanner
Development of the scanning technique and machine manufacturing

Author:
Pello Usabiaga Eizmendi

Supervisors:
Iñigo Arredondo López de Guereñu
Jorge Feuchtwanger Morales

Leioa, 23th Jun 2021

Contents

Abstract

In this research, an underdeveloped 3D scanning technology is presented. It consists of immersing an object in a liquid, and using Archimedes' principle to get its volume. Measuring many times the volume of different slices of the object can be obtained. If the scan is performed with an object with axial symmetry, immersing it along its symmetry axis, the object can be rebuilded stacking cylindrical slices one over the other. This method can also have applications in quality control or supporting other techniques.

The research also covers the development of a 3D scanner. It works using a load cell to measure buoyant force, and moving an object up and down in order to immerse it in water. It is controlled with an Arduino, which receives commands from an external controller using serial connection or MQTT. The firmware to make it work is also explained. To control it from a PC, a Python application has been developed too, to communicate with the Arduino and scan objects. Finally, the scanner was able to scan objects with axial symmetry with adequate precision.

1 Introduction and objectives

The technologies linked to 3D are nowadays a wide range of different techniques to manipulate representations of three dimensional objects, and to establish an interface between these representations and real objects. So, on one side, we have computer aided design (CAD) programs to manipulate files representing objects (Autocad, Fusion 360, etc.), on the other side, we have machines to manufacture objects based on these files (3D printers, CNC machines), and machines to create files based on real objects, i.e. 3D scanners. This research is focused on this last subject, 3D scanners, and develops a not commonly used technology to carry out the task of obtaining the shape and size of a three-dimensional object.

Before anything else, it is necessary to do a review of what has been done in this area before, and take a look at what the most used 3D technologies to scan objects are. One of the most popular options to scan small objects as well as large surfaces is photogrammetry. It is a technique that creates a 3D surface based on multiple surface points, generated from a large number of pictures of the desired object taken from different angles [1]. It yields very good results on representing surfaces' geometry, however it requires a lot of processing to make the reconstruction, and it is very sensitive to the quality of the pictures used. Another commonly used technology is LIDAR. It also consists in generating a cloud of points, but in this case by using a laser distance sensor to measure the distance from a sensor to a point on the desired surface. In this way, it is possible to know where each point is located, and the object's surface can be reconstructed [2]. There is also computerized tomography, which consists of getting 2D slices of the object, one every certain distance along an axis, to then reconstruct the complete 3D object by stacking the layers. This is very useful when an internal reconstruction is required, because there are computerized tomography techniques that can get information from inside the object [3]-[4], nevertheless this requires the object to be exposed to a large dose of radiation. Other common solutions consist of probing the object's surface with a mechanical touch sensor [5]. Some other solutions are presented by M. Abdelmomen, F. O. Dengiz and M. Tamre [6].

Overall, those solutions are based on distance measures from a known point to objects points. In any case, they do not have any direct measure of the scanned object's volume. So, the scanned object's volume fidelity may not be as good as required, or the scanning process may not give any certainty about this fidelity [7]-[8].

The objective of this work is to present a 3D scanning technique, which measures objects' volume characteristics in order to rebuild them. So, this process should give some reliability on the fidelity of the performed scans' volume representation. Because of this, it can be used on its own to rebuild objects, or it can be used in combination with other techniques, to enhance their results. The first work that develops a more or less similar technique can be found in [9].

In order to prove that the new technique works, it is intended to build a scanner that uses it to scan simple objects. It will work using Archimides' principle, measuring the buoyant force that an object makes when immersed using a load cell. Also, all the firmware to make the machine work controlled with an Arduino is going to be developed. For that, different communication protocols are going to be considered, and the device is going to be ruled from the outside, using commands and a dedicated software environment. After this, the machine is expected to be able to scan objects with axial symmetry, rebuilding them with the data taken in the scanning process. So, if this is achieved, the scanner's characteristics and its development are going to be measured.

By doing this development, the objective is to see if the method is viable, and to determine a little of its potential, alongside to lay some groundwork for any future investigation around this 3D scanning technique. Apart from this, the present work pretends to carry out a complete product development process, from the idea and the objectives, to the completion of the prototype, through the development of all the electronics, firmware and software to make it work.

2 The method and applications

2.1 Working principle

Even though most of the 3D scanning techniques can scan objects with any shape, or at least objects of many different shapes, the method that is going to be explained has difficulties doing so. This method is focused on scanning objects that have axial symmetry. For that, the objective of the technique is to divide an object with axial symmetry into slices along the axis of symmetry, and to measure the volume of each slice, as can be seen in figure 2.1.1. The object can then be reconstructed by stacking several cylinders that represent the volume of each slice. If the height of each slice is small enough, the object's reconstruction will be accurate. Objects with no axial symmetry can not be scanned in this way, because if they are divided in slices along one axis, these slices will not be cylinders. The possibility to scan other types of objects will be discussed in subsection 2.3.

To scan an object, it will be immersed in a liquid a known depth slice by slice. Because of Archimedes' principle, the object will experience a buoyant force. Measuring the buoyant force for each depth increment and calculating the differences between measurements, it is possible to know how much the buoyant force has increased when immersing each slice. Taking these measurements and the liquid's density, the volume of each slice can be calculated using (2.1.1). The process is illustrated in figure 2.1.1.

$$V_{slice\_i} = \frac{F_{buoyant\_i} - F_{buoyant\_i-1}}{\rho_{liquid} \cdot g} \tag{2.1.1}$$

where:

ρ_{liquid} is the density of the liquid used.

g is the acceleration of gravity.

V_{slice_i} is the volume of the slice i.

$F_{buoyant\_i}$ is the measured buoyant force when all the slices until i are immersed. That is, the volume of a slice depends on the increment of the force between it and the previous slice.

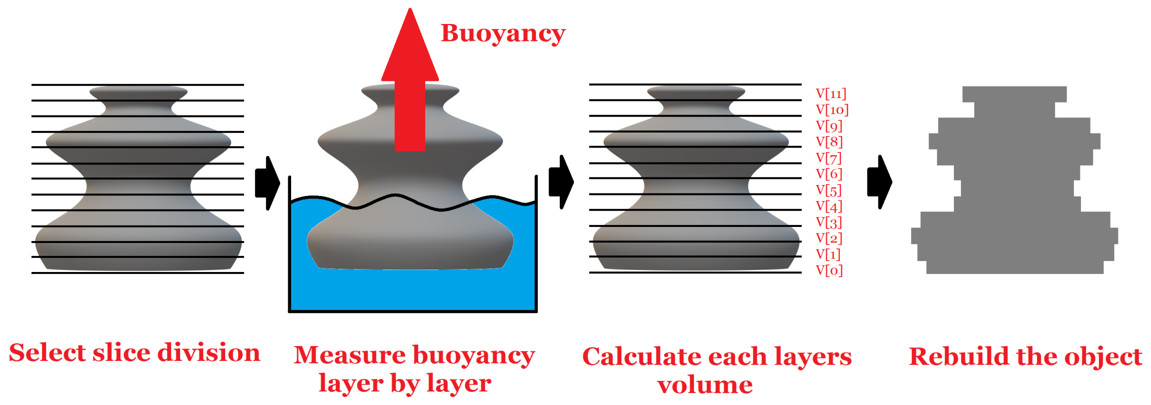


Figure 2.1.1: The process of scanning an object.

This way is possible to rebuild an object with axial symmetry. The thinner the slices the more accurate the reconstruction will be (within the resolution of the sensor used).

2.2 Measuring the liquid level rise instead of the buoyant force

Although a direct way of getting an object's volume is by measuring its buoyant force, the Archimedes' principle has one more result. It also determines that an object immersed in a fluid displaces an amount of fluid equivalent to its volume. So, the displaced liquid can be used to measure an object's volume.

When an object is lowered a distance h_{object_moved}, as can be seen in figure 2.2.1, the liquid level will rise a different amount $h_{liquid\_level\_raised}$. The relation between these two distances is determined by the sections of the object and the used container. So, the volume that can be obtained with this measure is the volume of the part of the object that is submerged. This causes a problem, because in order to make the slices height constant (or to control the slice height before immerse them), it must be ensured that the sum of h_{object_moved} and $h_{liquid\_level\_raised}$ equals to the desired slices height. This can be done by implementing a feedback system, which measures the liquid level rise, and lowers the object until the condition set by equation in figure 2.2.1 is met.

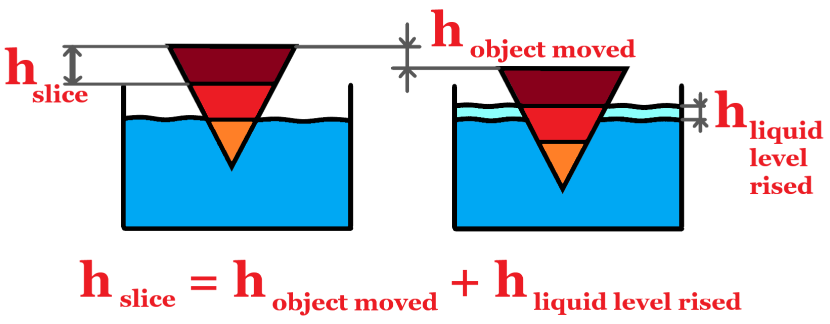


Figure 2.2.1: The object immersing process, with a feedback system implemented. Note that $h_{object\_moved} \leq h_{slice}$.

So, if an object is immersed exactly slice by slice, measuring liquid level and knowing the recipient cross section the liquid's displacement can be measured for each slice. Therefore, the volume of the slice can be calculated with the liquid level rise, following (2.2.1).

$$\begin{aligned}
V_{slice\_i} &= \Delta h_i \cdot (Sec_{container} - Sec_{slice\_i}) \\
V_{slice\_i} &= \Delta h_i \cdot (Sec_{container} - \frac{V_{slice\_i}}{h_{slice}}) \\
V_{slice\_i} &= \frac{\Delta h_i \cdot h_{slice} \cdot Sec_{container}}{\Delta h_i + h_{slice}}
\end{aligned} \tag{2.2.1}$$

where:

V_{slice_i}	is the volume of the slice i.
$Sec_{container}$	is the section of the container in which the object is immersed.
Sec_{slice_i}	is the section of the slice i.
h_{slice}	is the high of the slice i.
Δh_i	is the liquid level rise between immersing slice $i-1$ and slice i. That is, the volume of a slice depends on the increment of the liquid level between it and the previous slice.

The reconstruction process's sensibility depends on the container's section, instead of the liquid's density like when the buoyant force is used. This may be good, because liquid density varies with temperature and other factors, and it can be complicated to estimate it accurately. Nevertheless, measuring liquid level has its own drawbacks.

First of all, the sensor used to measure Δh_i can be either a liquid level sensor, or a distance sensor which measures the distance to a buoy. In both cases, the higher Δh_i, the easier it is to measure it. So, to get better resolution in the measurements, and therefore in the reconstruction, it is good to maximize Δh_i. This would happen if the section of the object is close in size to the section of the container, as can be seen in figure 2.2.2. So, to get an optimal reconstruction of the object, the containers section has to be chosen according to the section of the object to be scanned. If not, Δh_i would be small, and the reconstruction would be less precise.

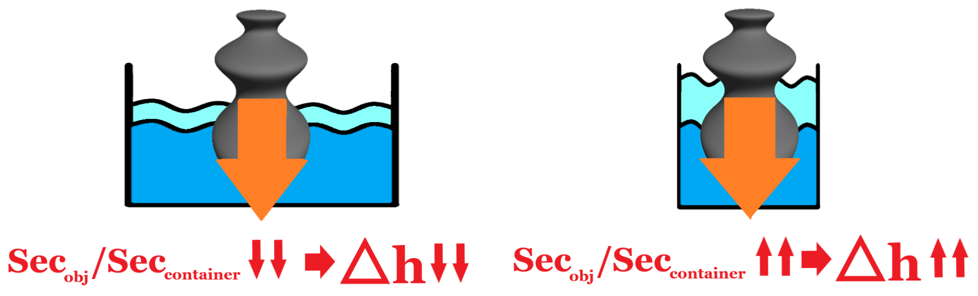

$Sec_{obj}/Sec_{container} \downarrow\downarrow \Rightarrow \Delta h \downarrow\downarrow$
$Sec_{obj}/Sec_{container} \uparrow\uparrow \Rightarrow \Delta h \uparrow\uparrow$

Figure 2.2.2: Liquid level rises when different objects are immersed in a container. Note that cyan water is the liquid that is raised when the object has been immersed in a certain step.

If the container's cross section is much bigger than the object's, the liquid level rise will be small, and using the feedback system can be avoided. Therefore, if the buoyant force is used to rebuild an object, a very wide container can be used and the liquid level rise can be neglected. However, if the liquid level rise is used to calculate slices' volume this can not be done, and the feedback system becomes necessary.

Also, if the measured magnitude is the liquid level or a distance to a buoy, those magnitudes are going to be measured in one point of the liquid's surface. So, the measurements are going to reflect the disturbances of the liquid level at this point, which will harm them. Instead, when what is being measured is the buoyant force, the measured magnitude depends on the total displaced volume, which will not reflect that many disturbances.

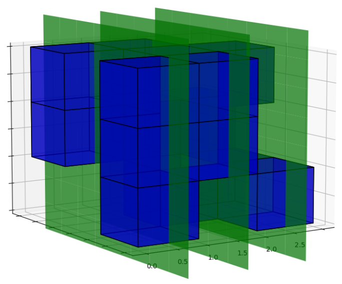
When measuring the buoyant force to rebuild an object, the buoyant force grows with the section of the object, and the resolution keeps the same in the force sensor. So, better reconstructions can be obtained for wider objects, using smaller slice heights, because the magnitude of the buoyant force allows it. In contrast, with the other method the measured Δh_i only depends on the slice height, and on the relation between the containers and the slices' section. So, a liquid level sensor will not measure the liquid level rise better for wider objects. Therefore, when the measured magnitude is liquid level, the slice height can not be improved for wider objects. In fact, the scanning resolution is worse using wider containers.

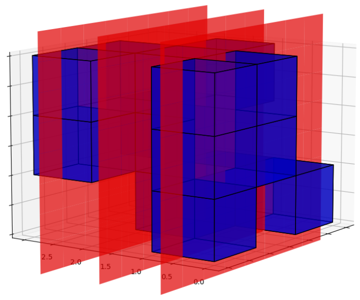
With all those drawbacks, measuring buoyant force seems like a better alternative to scan objects. Anyway, both techniques are not mutually exclusive, and they can be used together to compensate for the weaknesses of each other.

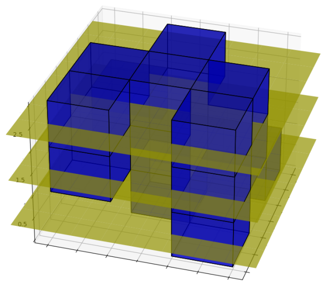
2.3 Possibility to scan general objects

As it was previously said, the object was divided in slices along the object's axis of symmetry. However an object has infinite axes along which the slices can be taken. Using them, a better reconstruction of the object can be achieved. For example, it is possible to place the object over the liquid, and scan it along one first axis. Then, the object can be rotated 90° in Θ, using spherical coordinates being the normal to the liquid the 0 inclination vector, get scanned along the second axis, and finally rotate it again, to scan it along a third axis, perpendicular to the previous two. In this way, the volume profiles along the axes of an orthogonal reference system would be obtained. Therefore, much more information about the object can be obtained. You can get the volume profile of the object along any axis. Perhaps, objects that do not have axial symmetry could be scanned.

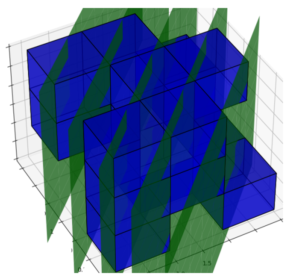
So, first of all, I checked if any kind of object can be scanned with this technique in a relatively practical way. For that, I simulated obtaining certain information from the object, and I tried to rebuild the object from this information. I treated the objects as an array of cubes, which helps a lot when working with the data. So, the reconstructed objects are also arrays of cubes. The object's complexity can be adequately captured by making the array large enough. With a large number of cubes, a smoothing algorithm can be used to get a more manageable mesh.

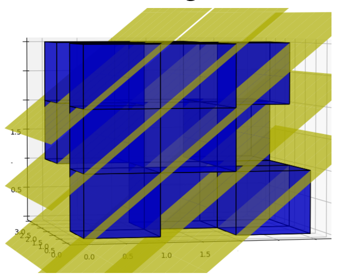

(a1) Along x axis.

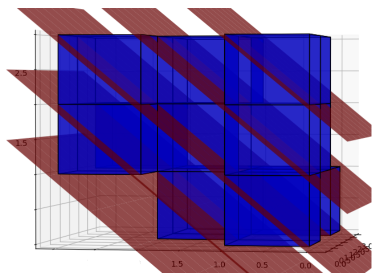

(a2) Along y axis.


(a3) Along z axis.

(a) Cuts along one axis direction.


(b1) Along axes x and y.


(b2) Along axes x and z.


(b3) Along axes y and z.

(b) Cuts along the combination of two axes.

Figure 2.3.1: Cuts in which the cubes have been counted.

The information I initially used is the number of cubes in each layer, along the three

Cartesian axes, as shown in figure 2.3.1a. So, for each object I scan, as they are figures with the same layers in each axis, I get $3n$ readings, where n is the layer number in each axis. Nevertheless, after doing some first tests, this information was not enough to reconstruct the object, so I added more readings. I get these readings along the combination of two axes, as shown in figure 2.3.1b. In this way, I add $3(2n-1)$ more readings for the reconstruction.

With this information, I trained a neural network to solve the reconstruction problem. Neural networks are a type of computing system, which uses large structures of nodes connected between them to transform input data into output data. Each node has some parameters, called weights, which rules what calculations that node performs into the data. So, changing these weights, the behavior of the neural network can change completely. Also, the performed operations are not linear, so this type of systems are good to represent nonlinear systems or algorithms. However neural networks alone can not resolve many problems, because it is very hard to design them in order to perform a desired task. What usually is done is to train them. By using the back propagation algorithm [11], the weights of a neural network can be changed to make them predict specific outputs for some given inputs. So, after training it with the desired results of the problem to be solved, a neural network model can get correct outputs for new input cases [10].

Using neural networks, I can solve my problem in a simple way, because it is very easy to generate large numbers of 3D objects, generated from the random stacking of cubes. Then, I simulate the data I would get from scanning the generated 3D objects. This data is fed into the neural network model, in order to train it. Finally, I can test the model with a new set of testing data, to see if it is able to solve the proposed problem effectively for unknown data.

I used the Keras [12] library for Python to work with the neural network. I created $n \times n \times n$ arrays, randomly filled with 1 or 0, which would represent the objects. A one in the array represents that there is a cube in that position, and a zero that there is a void. Then, I get the hypothetical readings of the scanner for each object, counting the cubes in the directions seen in figure 2.3.1. Finally, I used each $3n + 3(2n-1)$ readings to train a Keras model, with 5 dense layers and 75000 parameters. Really, each model only has one output, so I created a $n \times n \times n$ array of models, each of them to calculate if a cube is or not present in the object.

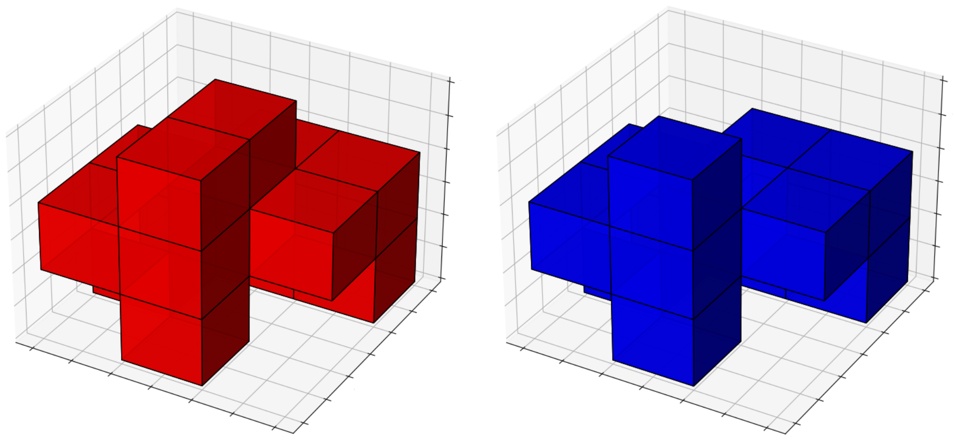


Figure 2.3.2: Real (red) and predicted (blue) $3 \times 3 \times 3$ objects.

Once I trained the models with 30000 figures of size $3 \times 3 \times 3$, I tested them with a test set of

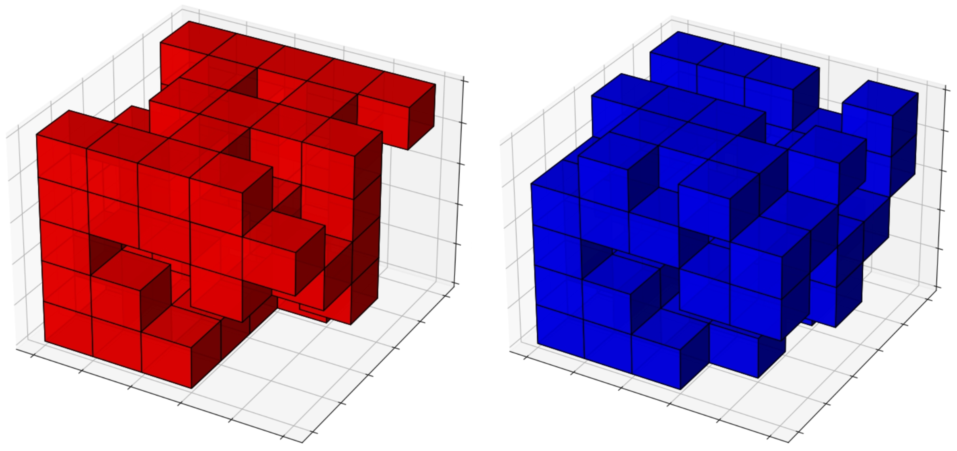


Figure 2.3.3: Real (red) and predicted (blue) $5 \times 5 \times 5$ objects.

1000 objects. The model array managed to rebuild the objects with moderate performance. It correctly places 89.3% of the cubes where they should be. An example of how one of these objects was reconstructed can be seen in figure 2.3.2. However, if I do the same with figures of shape $5 \times 5 \times 5$, the performance guessing where the cubes are dropped to 71.4% of correct cubes. The reconstructed objects are more or less similar to the real ones, but they have a lot of errors, as in figure 2.3.3. The used code can be found in the repository [34].

Therefore, objects with any shape are hard to scan with this method. If I increase the complexity of the figures, by adding more detail, the reconstruction quality drops down. If more data is taken from the object, the prediction will get better, but the number of data taken increases linearly with n, and the number of cubes to be predicted increases as n to the 3rd power. So, it seems logical that the more I increase the object's detail, the harder the reconstruction becomes. Another thing to improve, could be making the neural network model to have $n \times n \times n$ outputs, and avoid using $n \times n \times n$ models, one for each output. This would improve the predictions, because they would not be independent from each other.

Another improvement to this technique can be to limit the shape of the objects to scan. A scanner can be focused on scanning, for example, humanoid figures. As the shape of humans is more limited, a specific algorithm and/or model can be created to rebuild the scanned humanoids. This way specific patrons of specific types of objects can be used to enhance the reconstruction process, and the results should get better.

In [9], they obtain an accurate reconstruction of objects with complex shapes by immersing them in hundreds of directions. This means that if very large amounts of data are used it is possible to scan any type of object.

2.4 Support other techniques

The way of obtaining object information that is being discussed can be useful not only on its own, but also as a support to other techniques. In section 1, the overall limitation of the techniques used in 3D scanning to correctly represent the volume characteristics of the scanned objects were presented. Commonly used techniques do not measure volume directly, so they cannot guarantee the accuracy of their reconstruction of this parameter. By correcting this lack, and introducing the information of the volume characteristics of the object to the scanning process, any scan performed by those techniques could be improved.

The volumetric immersion reconstruction technique can be one way to achieve this objective. We have seen in subsection 2.3 that this method is not able to easily guess the scanned objects' shape very accurately. Even so, the amount of information that can be obtained from the object about its volumetric properties is huge. So, incorporating this information to the scanning process would be valuable to improve its representation of the object's volume.

One way of doing this could be by combining two techniques sequentially, for example, photogrammetry and volumetric immersion. First, the pictures for photogrammetry are taken and the object is immersed in the liquid. Then, the classic photogrammetry scan is performed, getting a point cloud representing the object's surface, which is converted directly to a 3D object file. After that, using a simple algorithm, these virtual objects' volume characteristics are measured, and these measurements are compared to the real object's ones. If they do not coincide, the point cloud is slightly distorted until they do so. For example, the points may move along the normal of the surface around them, to increase or to decrease the object's volume in some area. The process can be repeated as many times as required. The corrected mesh can be processed again to newly fit the photogrammetry information while trying to keep correct volume characteristics. The process can be repeated until a satisfying result is obtained.

Similar processes with other scanning techniques can be proposed apart from photogrammetry. Another option to combine techniques can be to generate a primitive shape of the object with a classic technique, turn the shape into small cubes, and randomly change their position until they fit the volumetric readings.

2.5 Scanning inner surfaces

Another advantage of the technique I am explaining is the ease to scan inner surfaces of an object. The other techniques to 3D scan objects hardly can achieve this. In the case of photogrammetry, pictures from inside the object must be taken, and their light conditions must be good. In the case of techniques based on distance measure, the measuring element must be able to reach the inner surface. Both options could be hard when dealing with small objects. And even if it is true that computerized tomography is able to scan these surfaces, its cost may be prohibitive. In contrast, the volumetric immersion scanning technique is able to get information from inside objects without the need of expensive or complicated systems.

To illustrate this, I will show a simple case. Let us assume an object with axial symmetry, and an internal cavity opened only from the top. For example, a glass. In figure 2.5.1 we can see the process. First, the object is scanned by immersing it in one orientation and then in the other. In the first immersion, the liquid would not enter the internal cavity, and in the second one it would. Note that for the second immersion a little hole is needed in order to let the air escape from the cavity. Nevertheless, in the first immersion the hole must be covered. So, in the first reading the data would represent the full object's volume, without the cavity, and in the second one the data would only represent the volume of the walls. Subtracting the second measurements from the first ones, the cavity volume profile could be obtained. With this information, the reconstruction can be achieved, as seen in figure 2.5.1.

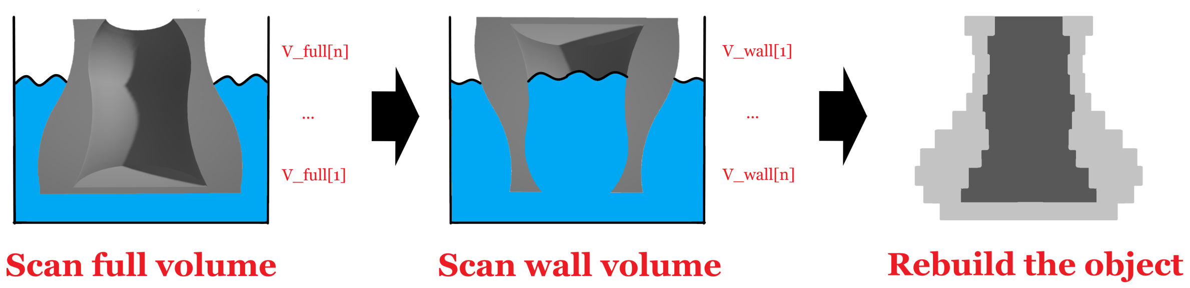


Figure 2.5.1: Inner surfaces reconstruction scheme.

This case is one of the simplest. But the potential of the concept is interesting. The possibility to obtain information from internal cavities in a non destructive way is always welcomed. With the appropriate development of techniques and algorithms, this approach may be used to address many different problems.

2.6 Quality control

Another interesting application for this technology uses it for quality control. The ability to obtain direct and precise information about the volume of the object makes this technology capable of checking if a manufactured object has deviations or imperfections. In a manufacturing chain, the properties expected for the objects could be loaded in a volumetric immersion 3D scanner, and this scanner could automatically check if any of the mass-produced items do not match the parameters on record with the standard requirements.

For example, let us assume that the object we are manufacturing is an hexagonal cup with a handle, and the volume profile along the cup's axis is the one shown in figure 2.6.1. Note that the cup has been scanned with the closed part down, without any liquid going into it. So, this profile can be loaded in the scanner which is located at the end of the manufacturing process. For each cup that goes to the scanner, a robotic hand would grab it, and it would immerse the cup in the scanner's liquid along the correct axis. If the cup is correct and the read profile matches the loaded one, the cup will pass to the next step. But if suddenly a cup gets broken in the manufacturing process, the profile that the scanner would get would not match like is shown in the discrepancy between the blue and the orange lines in figure 2.6.2. Because it does not match with the loaded one, the broken cup would be discarded.

This is a pretty simple solution to check if a manufactured piece is correct or not. The scanner is simple, so it can be cheap. And the scanner's resolution could be good too, so it would be able to detect small imperfections. This point is going to be developed further in section 6.

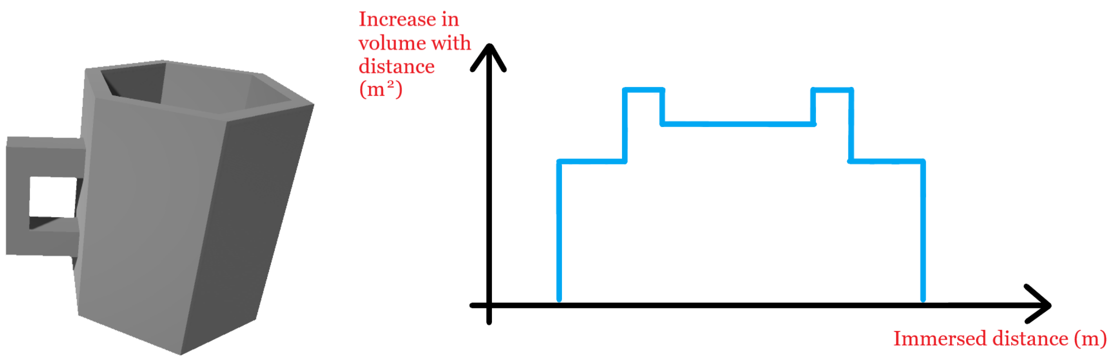


Figure 2.6.1: The manufactured cup (left), and its ideal profile (right).

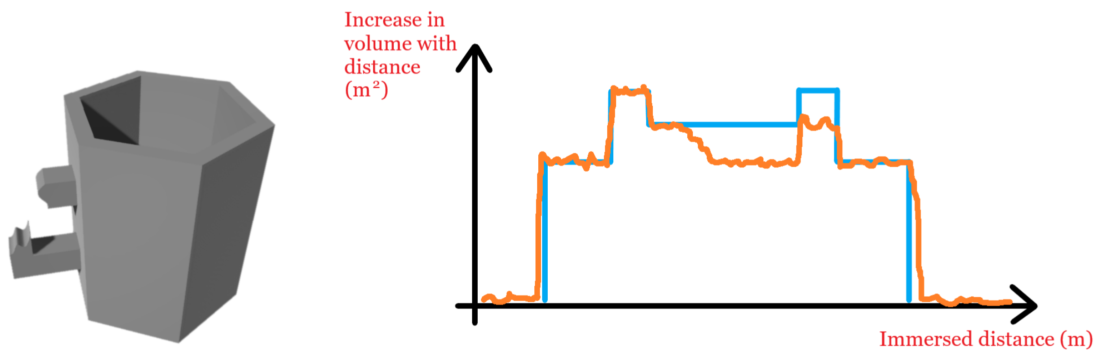


Figure 2.6.2: The broken cup (left), and its profile in orange over the ideal one in blue (right).

The only big counterpart to this technique is that any object checked by this method is going to be wet, because of the immersion on a liquid. This may or may not be a problem, depending on the object's and industrial process's characteristics.

Therefore, this method may have some advantages. In terms of economy and/or speed, this technique may beat distance measurement based solutions. And compared to photogrammetry, the fact that the volumetric immersion technique directly measures objects properties gives it potentially better resolution and reliability. However there are a lot of cases in which more simple solutions exist. To check a piece's integrity, an easier solution is to weigh it, even if this way a deformation can not be detected. For detecting deformations many times a simple photo can be used, using machine vision. But small or inconspicuous errors can be hard to detect in a photo. In conclusion, the volumetric immersion reconstruction technique can be considered as an alternative in these cases, as a relatively low cost and high reliability technique.

3 Machine's hardware

3.1 Approach to the machine

In following sections, I will describe the development of the experimental work I did for my research. It consists of a 3D scanning machine, both the hardware & firmware, and the software setup for its use. The objective is to have a machine which is able to scan objects with axial symmetry, or which is able to perform all the different applications explained in section 2. So, with a given object, previously prepared to be fixed to the machine, it would immerse the object in a liquid along the axis of symmetry. It would measure the buoyant force for each desired distance along the axis, so it can get the volume profile of the object.

For the scanner structure, I used a repurposed and old 3D printer, model Geeetech Prusa I3 Pro W [13]. It has a carriage intended for the extruder, which can move along 3 axes, x, y and z. This is shown in figure 3.1.1. So, I took advantage of its ability to move along the z-axis, to immerse the test pieces slice by slice. The Z-axis motion is achieved by a pair of stepper motors. These motors drive a pair of lead screws, which push the carriage along a pair of linear guides.

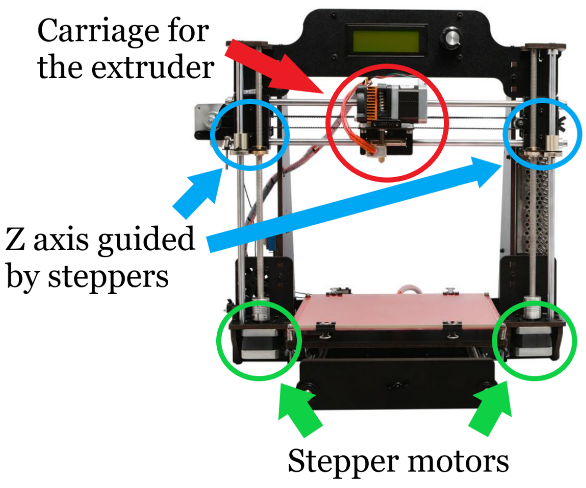


Figure 3.1.1: Reused printer's parts. Original picture is extracted from [13].

The sensor I used to measure the buoyant force is a load cell, purchased from BricoGeek [14]. The test piece is attached to the load cell with a screw, and the load cell is attached to the carriage of the extruder, which was previously removed. So, when the gantry goes down, the piece is immersed in a container of water. The container's section is considerably wider than the object's, so the problem discussed in subsection 2.2 is avoided.

The stepper motors and the sensor are controlled by an Arduino UNO WiFi Rev2 board [15]. The board takes care of managing the scanner's hardware and its communication with external devices. This communication goes through an USB connection and over WiFi, using the MQTT protocol. In this way, an external program can use any of these communication channels to send text commands to the board and to receive data from it. The commands' format is inspired by SCPI commands. Finally, I developed a cross-platform application programmed in Python to control the scanner through a graphical interface.

In the following sections, I will develop all these aspects in depth. First, I will explain the scanner's hardware. This includes the load cell, the stepper motors and all the extra hardware I needed to get them working properly and integrated with the Arduino. Then, I will explain the circuitry required and its assembly as well as the PCB manufacturing. In section 4, I will talk about the firmware that runs in the Arduino. There, I will talk about how the used program is structured, and how I managed to get it run in a low resource microcontroller. Finally, in the section 5, I will briefly explain the software environment beyond the Arduino, that is, the MQTT broker and the Windows application to control the scanner.

3.2 Sensor selection

To turn a real object into a digital representation of it, some kind of sensor is needed. In my scanner's case, I decided to use a load cell, for converting a buoyant force into an electrical signal. However many other sensors had been considered. Even if in the end I used this sensor to measure the buoyant force, I also considered sensors based on the liquid's level rise, as discussed in subsections 2.1 and 2.2.

About the options to measure buoyant force, I did not consider a lot of options. I could have used strain gauges, attached to some support and calibrated their response. However essentially this is what a load cell does, but ensuring a linear response, due to the Wheatstone bridge configuration, and maximizing the sensitivity, due to its structural design. So the load cell was better for what I need.

Another option would have been to use a piezoelectric sensor, which turns strain in a potential difference. So, it can be used to measure a force, as can be seen for example in [16]. However, these tend to be more appropriate to measure changes in forces, even at relatively high frequencies. For a small and constant force it is better to use a load cell.

For liquid level sensors there is a wide variety of different technologies that can be used. There are capacitive sensors [17] or resistive sensors [18], among others. And, if a buoy is used, any contactless distance sensor to measure the distance to that buoy, or any other distance sensor attached to the buoy can be used. For example, a time-of-flight (ToF) laser distance sensor could be used to measure distance to a buoy, or a digital encoder may be used to obtain the buoy's position.

Nevertheless, overall the liquid level sensors are prepared for large liquid volumes, with large height changes. The distance sensors are not usually intended for sub-millimeter measurements, which are the specifications I require. Of course, there are liquid level sensors and distance sensors with better resolution, but they are expensive, or they are hard to obtain. So, I finally decided to use the load cell.

I was able to obtain a linear variable differential transformer (LVDT). It is a type of inductive sensor, its working principle is explained in [19]. Typically, it is able to achieve an accuracy of about 1 μm. And because of the LVDT being frictionless, it is ideal to attach its magnetic core to a buoy and let it move through the sensor head attached to the container. This sensor's disadvantage is its price, being much more expensive than the load cell. So I first made the machine using the load cell, and then I did some tests with the LVDT. However I was not able to attach it adequately to the machine using a buoy and it had some friction to move up or down. So, and because of the high price of the LVDT, I did not consider it as important as properly testing the load cell.

3.3 Load cell

For what was said in the previous subsection, I used the load cell acquired from BrecoGeek [14]. The supplier does not provide a datasheet for the load cell and the information given about it in their web page is limited. For this reason, I had to perform a calibration in order to know its performance. To test and calibrate the load cell, I used two different analog digital converters (ADC-s), the HX711 [20] and the ADS1115 [21]. More about the ADC-s is in subsection 3.6. They are connected to an Arduino UNO WiFi Rev2 board [15]. There, I run a simple program, which constantly reads from the ADC and plots the value to the serial plotter.

After testing that the setup works, the next thing to do was to attach the load cell to the machine. For this, I designed an adapter piece to attach the load cell to the carriage for the extruder. The carriage is solid enough to be considered as a fixed point. So, placing the load cell as in figure 3.3.1 works properly and measures the buoyant force when the piece is immersed.

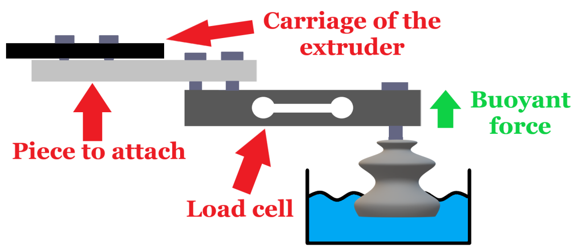


Figure 3.3.1: The load cell attached to the carriage of the extruder.

For initial testing, I used a cylindrical piece, made of polyamide, 18.63 mm in diameter by 88.3 mm long. It can be seen in figure 3.3.2. It is slender, so it is hard to scan with a small slice high. I used it to test if the scans were correct, because all the slices should have the same thickness.

To check the correct operation, I immersed the piece using the controls that are used to move the old 3D printer's stepper motors. That is, the motors for this measure were controlled with an independent hardware. The readings are shown in figure 3.3.3. The scanner was not calibrated yet and the steps did not have the same length, but the buoyant force increased when the piece was immersed.

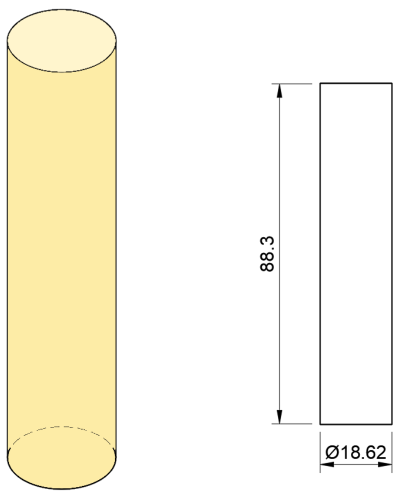


Figure 3.3.2: Used piece and its projection view. The distances are in mm.

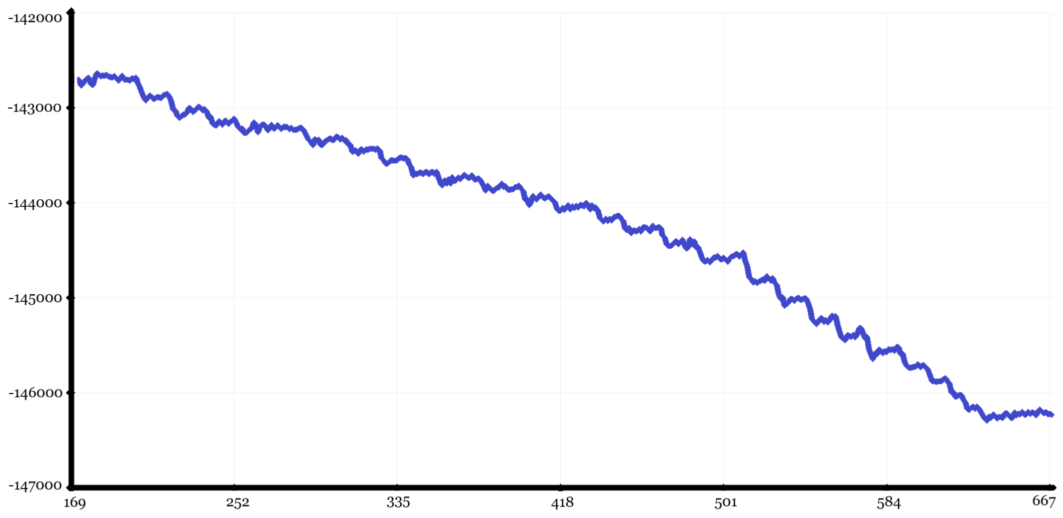


Figure 3.3.3: Readings used to check that the load cell works properly. The y axis are signed integers given by the ADC, and the x axis are the number of readings of the ADC. Higher buoyant forces correspond to more negative readings.

3.4 Stepper motors and their interference

To be able to scan objects, I had to be able to move them up and down, in order to immerse them. So, I changed the stepper motors' control to my system, controlled by the Arduino. I installed motor drivers in my circuit and I connected the motors to them. The drivers are the same that controlled the steppers in the printer, the Allegro MicroSystems A4988 [22].

There are 2 motors, which have to be moved simultaneously. The drivers have the option to control the motors with microstepping, which is a technique to control stepper motors with more resolution, however, since I do not need such a high resolution on the z axis, I did not use it. Hence, I did not connect the microstepping pins to the Arduino.

To supply the drivers, I used a 12 V external power supply. With the 5 V that the Arduino board supplies it is not possible to move the steppers. The steppers were tested, and they worked well. The steppers move 1.8° for each step, which equals 200 steps per turn. Each turn, The lead screws connected to the motors have a $1.25\,mm$ pitch, resulting in a resolution in the z axis of $0.00625\ mm$ per step.

However, the inclusion of the steppers has resulted in an undesired effect. The motors work with coils so they take a lot of current. Hence, they introduce a lot of interference in the circuit. The supply voltage changes due to common impedance at the source, and inductive effects cause voltage spikes. So, when reading the load cell output, the measurements are noisy and disturbed. This can be seen in figure 3.4.1, where the readings of the ADC can be seen before and after connecting the steppers to the supply. The normal deviation of the readings increases from being of the order of 10 to being of the order of 1000, and they also introduce a DC offset.

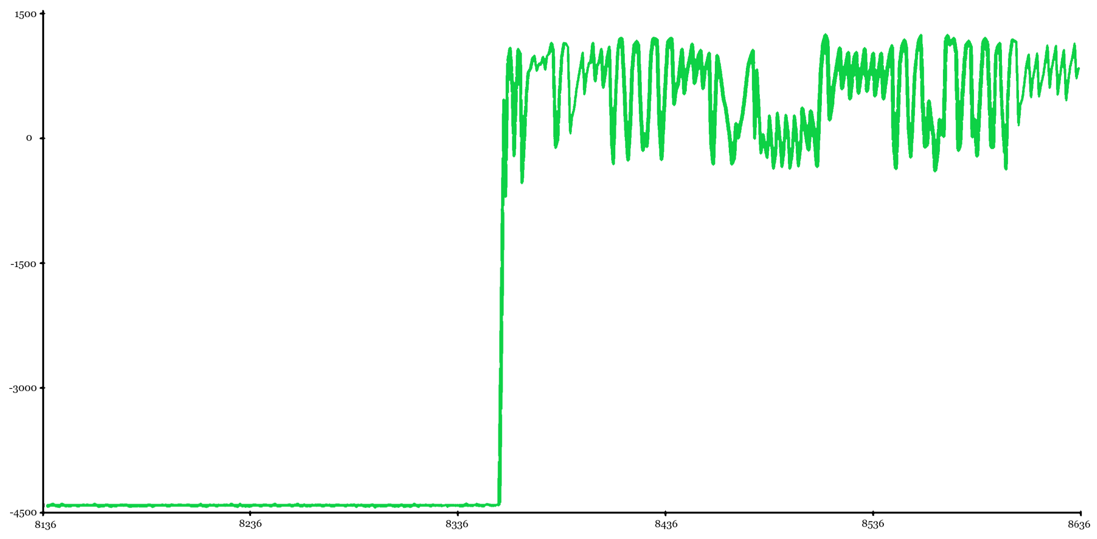


Figure 3.4.1: The readings from the load cell, before and after connecting stepper motors to the supply. The y axis are signed integers given by the ADC, and the x axis are the number of readings of the ADC.

This is absolutely inadmissible for the readings. No reliable values can be taken under these conditions and a solution must be found. The problem comes when I connect the drivers supply, so the solution I found was to turn them off when reading the load cell, using a relay. My circuit has two supply voltages, $12\ V$ and $5\ V$. The $5\ V$ one is used to supply the ADC and the drivers' digital electronics, and the $12\ V$ one is used for the motors' power supply. The relay was placed between the $12\ V$ power supply and the driver's power input, to be able to turn them on or off.

The relay is an electromechanical device, which uses an electromagnet to switch from an output to another. The relay's electromagnet also introduces noise and perturbations into the sensor, but they are considerably smaller than the ones from the motors. This only happens when the relay closes the circuit from its normally opened position, i.e. when the relay is polarized. In order to avoid this, the motors are only activated when the relay is polarized, therefore, when measuring from the sensor there are no perturbations from them in the circuit.

With these modifications, I get back the previous readings, which were good. Even so, some logic must be implemented to switch the motors on and off, in order to move the piece up and down and to get buoyancy readings.

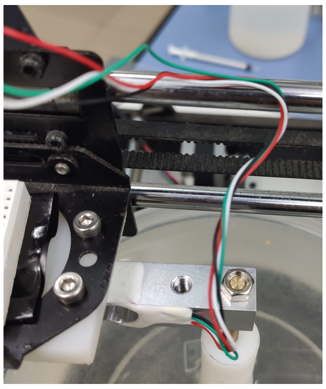
3.5 The remaining analog signal conditioning

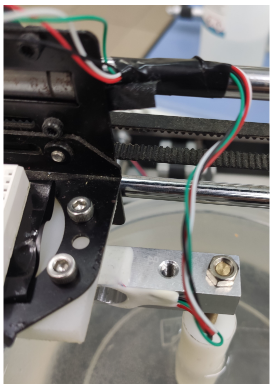
With all the above done, I still don't get the desired behavior from the circuit. The main problem comes from the wires that connect the load cell and the rest of the circuit, which are pretty long. So, when they are perturbed, they produce interference in the circuit. Also, touching some circuit elements, not necessarily related with the load cell signal, causes the signal to change or to become noisy.

This is because the load cell signal is weak. In other words, according to Friis' equation for noise factor 3.5.1, the noise depends mainly on the elements of the circuit before the first amplifier, including it. So, in my case, these elements are the load cell, the long wires connecting it to the rest of the circuits and the ADC's amplifier. In order to correct this, the best option is to amplify the signal before the noisiest element, the wires. For this, I used an instrumentation amplifier, directly after the load cell. With it, I amplified the differential signal between both outputs of the Wheatstone bridge before it goes through the wires to the ADC.

$$F_{tot} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 \cdot G_2} + ... + \frac{F_n - 1}{G_1 \cdot G_2 \cdot ... \cdot G_n} \tag{3.5.1}$$

After this, the circuit's behavior has considerably improved. Now, touching the wires hardly introduces noise in my readings, and neither does touching other elements of the circuit. Even so, touching the wires introduces a small offset in the readings, which return to their undisturbed position when I stop touching them. The cause of this behavior is very simple, and it is because the cables are directly connected to the load cell, as can be seen in figure 3.5.1a. When I touch them, I apply a force to the load cell, which is read by the sensor. A simple solution is to attach the wires to a fixed point on the carriage and then from there to the ADC, as can be seen in figure 3.5.1b.


(a) The load cell with the wires without attaching them.


(b) The load cell with the wires attached to a fixed point.

Finally, I considered that the Arduino's 5 V supply is not stable enough to supply the load cell's Wheatstone bridge. To fix this, I used a precision voltage reference (Texas Instrument LM329 temperature compensated Zener reference [23]). I polarized it with the 12 V supply, used also for the steppers. Apart from the load cell, for convenience I polarized the instrumentation amplifier with this precision reference, in order to have just 4 wires between the carriage and the rest of the circuit.

3.6 Analog-to-digital converters

As I mentioned in subsection 3.3, while testing the load cell and the remaining the analog signal conditioning I was using two different ADCs, the HX711 [20] and the ADS1115 [21]. This way I could switch between them and I could rule out that any error came from the ADC. But when all the analog signal conditioning was done, it was time to choose one of them.

The HX711 is intended to be used to read load cells, with no analog signal conditioning at all. So, it includes an on chip low noise programmable gain amplifier (PGA), with selectable gains of 32, 64 or 128 [20]. Also, It has a high resolution, 24 bits [20], in order to be able to read very small voltages from a Wheatstone bridge. However it has a very low output rate, of about 10 samples per second. Due to this, if random noise appears, it is much more difficult to eliminate it by averaging.

The ADS1115 is a general purpose delta-sigma ADC. It also has an on chip PGA, with selectable gains of 2/3, 1, 2, 4, 8 or 16 [21], that is, much smaller gains than the HX711. Anyway, this is not a problem, because the signal is already amplified by the instrumentation amplifier. The resolution is slightly lower, of 16 bits [21]. In return, the output rate is higher, about 860 samples per second [21]. Using this ADC will make the readings a lot faster, so it will provide a more convenient way to remove random noise.

Finally, it turned out that the random noise that comes from the sensor is much higher than the resolution that can be obtained with both ADCs. So, the main advantage of the HX711 over the ADS1115 is irrelevant. For all these reasons, and taking in consideration the higher speed, I chose the ADS1115 for my system.

Anyway, I still have a problem with the readings. Most of the time, the data is good and has random noise, which I can remove by averaging an appropriate number of samples. But sometimes, for no apparent reason, the readings become much noisier and some inconsistent readings appear, like can be seen in figure 3.6.1. I was not able to determine why this is happening, and reading the signal with an oscilloscope I could not reproduce this behavior, so I suppose that is a problem due to the ADC. I was not able to avoid it, but I designed a way to correct it.

In order to get a correct average value from these readings, I can filter them. I decide to get all the data in the scanner, send it to the PC, and there make a program which filters all this data. The main working strategy is to remove the readings that are too far from the average and keep the remaining points. Also, if at some point the ADC readings were too

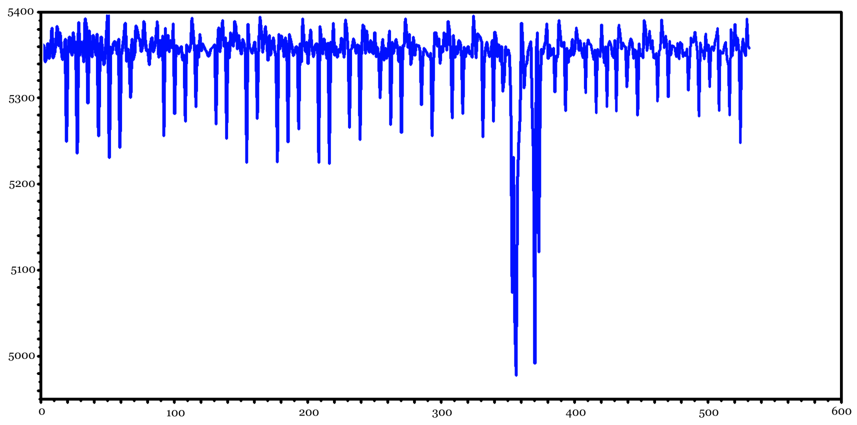
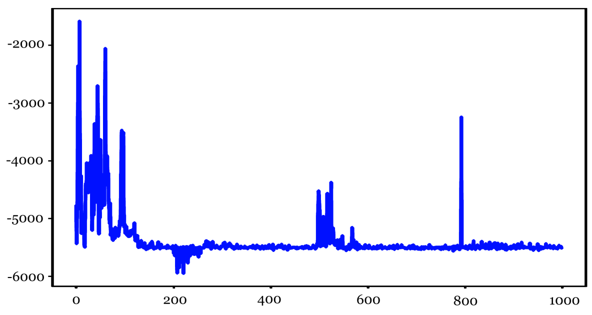


Figure 3.6.1: ADC readings with inconsistent readings, in form of peaks. The readings were taken with a constant buoyant force. The y axis are signed integers given by the ADC, and the x axis are the number of readings of the ADC.

noisy, I discarded all the readings around these times. This is done by dividing the readings in sections and calculating each section's standard deviation. If the standard deviation of a section is greater than the normal values, that section is also removed from the data set. With the resulting presumably good data, I calculate a new average, not affected by the anomalous effects of the ADC. Before the filter, the mean value oscillated or had disturbances for a non-changing buoyant force. Nevertheless, with the filter I get a constant reading. The data filter can be seen in figure 3.6.2, and the filtering script can be found in repository [36], function `measure_buoyancy_and_filter(averages)` on class `immersion_scanner`.


(a) Data before being filtered.

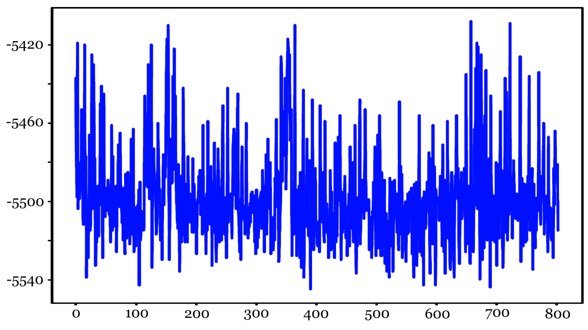

(b) Data after filter, note that a lot of points were removed, so the length of the data is shorter.

Figure 3.6.2: The filter of the readings, to remove noisy and inconsistent readings. The y axis are signed integers given by the ADC, and the x axis are the number of readings of the ADC.

3.7 Load cell calibration

The ADC converts electrical voltage into an integer number. But if I want to get volume readings, I must calibrate the circuit. In order to do so, I used some patron pieces, which I weigh with the precision weight which can be seen in figure 3.7.1. With them, I calibrated the response of a force applied in the load cell at the output of the ADC. Also, knowing the water density, the relation between an immersed piece's volume and the ADC's output can be found.

Figure 3.7.1: The precision weight, brand Sartorius, from the electricity and electronic department, UPV/EHU, Spain.

I used an aluminum piece and a bronze piece. I measured their mass with the precision weight, and I also took measurements of the pieces with the load cell next to measurements of the load cell without any force in it. The results can be seen in table 3.7.1

Table 3.7.1: Measurements for the calibration.

	Aluminum	Bronze	No force
ADC measure (int)	-4738.6620	-3739.0932	-4858.3424
Mass (grams)	9.53505	88.0264	0.00000

So, if I compare the readings between them, and calculate $\frac{\Delta\, ADC\, measure}{\Delta\, mass}$ with the three possible combinations of the data in pairs, I get the calibrations in table 3.7.2. I decided to use the mean value between the two first calibrations, because they are very close to each other. So, the final calibration value is -12.72 units/gram.

Table 3.7.2: Different calibrations for each data pair.

Data pair	Calibration in units/gram
Bronze - Aluminum	-12.734764
Bronze - No force	-12.714926
Aluminium - No force	-12.551626

I assume that water density is 10^6 grams/m^3, which is the pure water density at 3.8°C. So, the relation between an immersed volume and the ADC reading is $-12.72 \cdot 10^6$ units/m^3.

3.8 Final schematic and PCB design and manufacturing

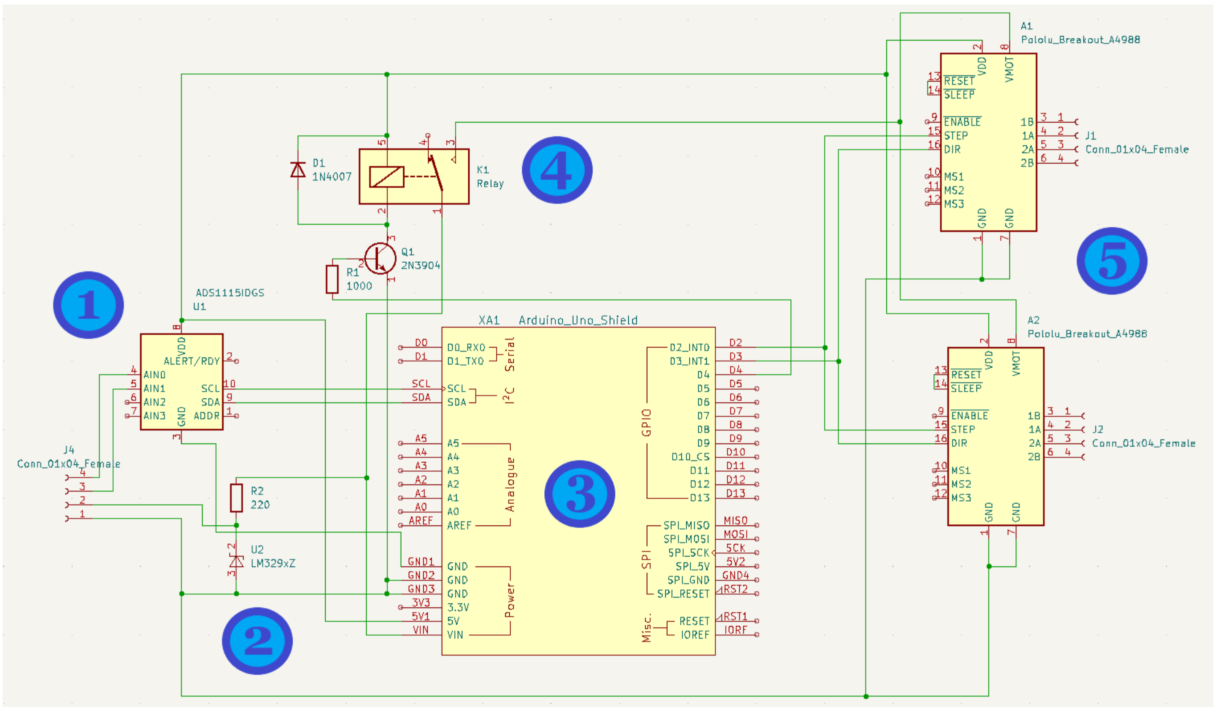


Figure 3.8.1: The schematic of the complete circuit in Kicad [24]. 1: the ADC and the connections to the load cell. 2: the precision voltage reference. 3: The Arduino. 4: The relay to control motors' supply. 5: The motor drivers and their connections to the motors.

After developing all the machine, all the circuits can be put together. In figure 3.8.1 all the hardware attached to the Arduino can be found. It includes the ADC, the motor drivers, the relay and the stable voltage supply for the load cell. The steppers are connected with 4 wires each, using a connector for each one. The module for the load cell, with the instrumentation amplifier and the sensor, can be seen in figure 3.8.2. They are connected to the rest of the circuit with 4 wires and a connector.

With the schematic of the figure 3.8.1, and in order to manufacture a final version of the circuit, I did a printed circuit board (PCB) design. I managed to make the circuit fit on a single layer. In figure 3.8.3 the final design can be found. Its overall dimensions were configured in order to fit in an Arduino shield format. This format consists of a PCB, stacked over the Arduino board, that connects to all the Arduino I/O pins. This way, the shield circuit is powered and works with the Arduino, assembled with it, and retains the Arduino's expandability, stacking other circuits connected together. The final circuit stacked over an Arduino board can be seen in picture 3.8.4.

The PCB was manufactured using a prototyping PCB board (C.I.F. REF. AA16). It consists of a support made of FR4, a thin copper layer and a positive photosensitive resin over it. When the photosensitive resin is exposed to ultraviolet (UV) light, it gets soft and can be removed using *NaOH* as solvent. So, the desired copper tracks have to be masked before exposing the resin to UV, avoiding these parts to get soft and then removed. To do so, I used a transparency, with the outline of the copper tracks printed on it. The transparency was

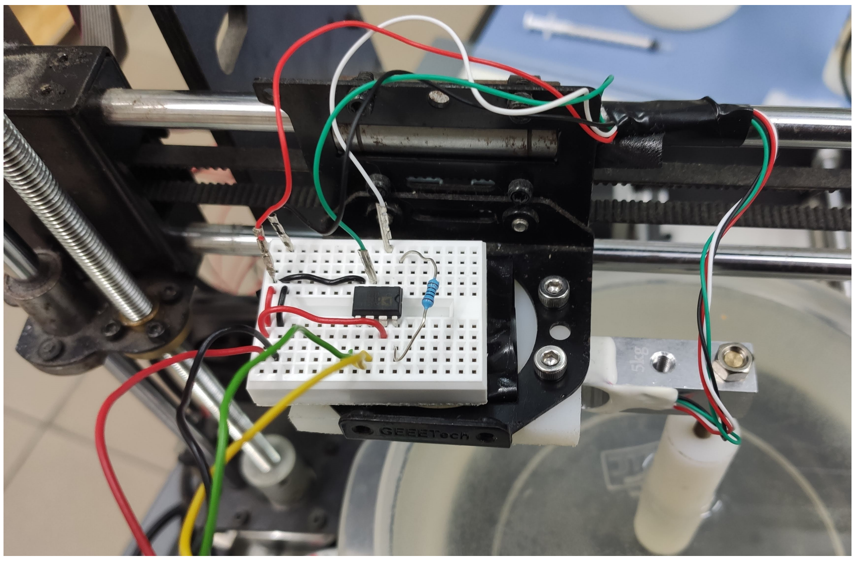


Figure 3.8.2: The load cell's module, with the instrumentation amplifier, the resistance R_G to set its gain, the load cell and the connection to the rest of the circuit.

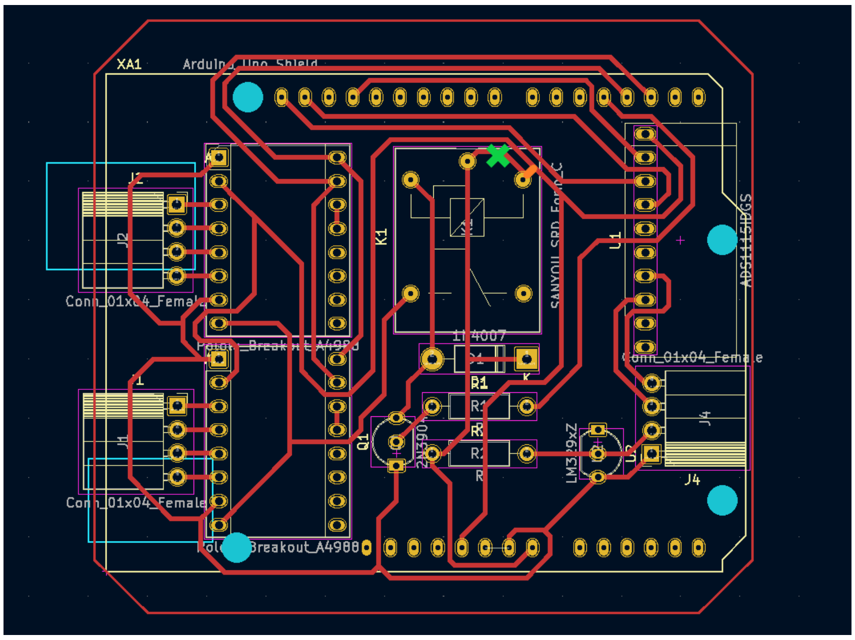


Figure 3.8.3: The PCB of the complete circuit in Kicad. Note that an error was made in the design, marked in green. The correct connection is marked in orange.

placed over the PCB board before the UV exposure, and when the resin was removed with $NaOH$ the desired pattern appeared over the copper layer, formed on the non exposed resin. This non exposed resin was used as a mask over the copper, to protect the circuit's tracks from the next step.

To obtain the circuit's traces on the PCB, all the superfluous copper must be removed. A wet etching process is used in which the previously prepared PCB board is immersed in a solution of $HCl + H_2O_2$. With this method, the copper was removed from the entire surface not protected by the mask. After the process ended, the circuit was cleaned first with water and then with acetone to stop the reaction. Finally, to protect copper from oxidation, the

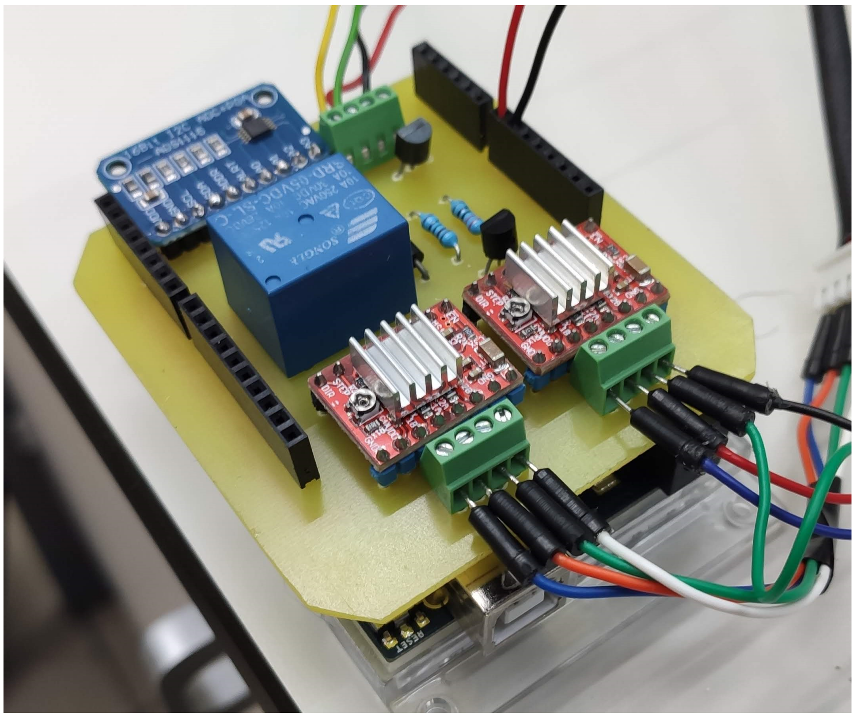


Figure 3.8.4: The final circuit, in shield format, stacked over an Arduino UNO WiFi

PCB was protected with a thin layer of clear lacquer. The result after the PCB fabrication process can be seen in figure 3.8.5.

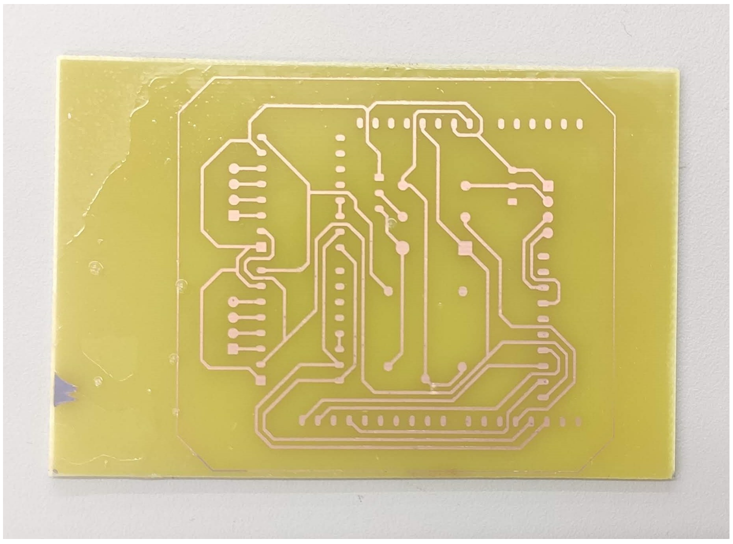


Figure 3.8.5: The PCB after its fabrication process, before drilling.

Finally, the points where a component had to be inserted were drilled with a milling machine. The error shown in figure 3.8.3 was manually corrected with tin. The components were put in their place and the circuit was soldered with tin. All the connections were checked and the circuit was tested.

It worked, but its performance was bad. The readings were noisy, and any physical disturbance in the circuit made them change their value or introduced voltage spikes. This turned out to be because I was supplying the Arduino with the 12 V supply through V_{in} connection.

This voltage is used by the voltage regulator in the Arduino to supply all the board and also the external elements like the ADC and the drivers' digital logic. In consequence, common impedance problems make the voltage at the precision voltage reference to slightly change, and this causes the readings to get noisy. The unique solution I found to this problem is to get back to the moment where the readings were reliable, which was when the circuit and the board where supplied with the PC using USB, and the 12 V supply is only used for the stepper motors and the load cell module.

4 Machine's firmware

4.1 Introduction to the Arduino environment

The firmware is the set of instructions that controls the operation of a digital device. It is written permanently in its memory, and it is usually interpreted by a microprocessor or a microcontroller. So, it consists of a compiled program, a binary file in machine code, which directly represents the instructions that the processor must execute.

In the case of Arduino, it is important to speak a little about its environment, in order to understand better how it works. Arduino is a company, which provides open source hardware and software intended to provide an easy way to work with electronic hardware. Their products are the Arduino boards, single-board microcontrollers, equipped with an interface of I/O pins, voltage supply regulator, serial communication interfaces (for example USB), and some facilities to program their integrated microcontrollers. They are very useful for prototyping, for students, or for hobbyists who only want to start approaching programming and electronics.

One of the big advantages Arduino boards have over using any other microcontroller is how easy are to program. Other microcontrollers commonly must be programmed with a specific hardware for it. In some cases, they have to be attached to a programmer board, and once programmed be moved to their target board. Or instead, they can have some specific pins through which they can be programmed, using some dedicated hardware to connect a computer to the microcontroller and care about the programmation. However, the Arduino boards can be programmed directly with a standard serial communication port, USB, for example. This makes programming them comparatively simple for anyone. They achieve this in a simple but brilliant way.

The only thing that distinguishes the Arduino UNO board's ATMega328p microcontroller from any other ATMega328p is the bootloader program. It is a simple and light program, which is stored in the main memory of the microcontroller and which runs at startup. It simply takes control of the available serial communication interface, and reads it waiting for a compiled program to get into it. If this program does not arrive in a short period of time, the execution sequence passes to any previously loaded program. However if a new program comes in, the bootloader will write it in the main memory, and then it will pass the execution sequence to the new program. This way, the user can reprogram the Arduino boards microcontroller at any time, when restarting it. This is automatically done by the integrated development environment (IDE) provided by Arduino, which is also used to develop programs for their microcontrollers in C or in C++.

Regarding the programming of the Arduino programs, the simplest possible program consists of two functions, setup() and loop(). The first of them is executed at the start of the program, right after the bootloader gives the control to it. Once setup() ends its execution, the loop() program is executed continuously. So, any program that is intended to be executed, must be somehow called from one of these two functions. Apart from this, any other code can be written. From the code any of the board resources can be referenced to make use of I/O pines, the serial port, or any other feature included. Also, to ease some of the things that can be done with the board, a lot of libraries are available. For example, to control any external device, the microcontroller must be able to handle all the communication with the device, for example using I2C, and must be able to know what commands and protocols to use to manage the external devices. All these things can be more easily done using already available libraries.

4.2 Program requirements

I have already explained what hardware I am working with. My microcontroller must be able to make it work in an appropriate way, and at the same time, the board must be able to be controlled and to communicate with, for example, a PC. The board must be constantly listening to the different ways a command can come in. Then, it must recognize this command, and process it to fulfill what it orders. And finally, it must be able to get data back to the external controller, so it must be able to write messages in the communication way it is using at each moment.

For the communication, I used the Arduino's serial communication, which is a universal asynchronous receiver-transmitter (UART), which sends and receives data over the USB port. I also used MQTT, which is a network protocol based on a publish-subscribe structure. I will explain this in subsection 5.1, but for now I will treat it like a simple communication protocol over which the board can write and receive messages connected to WiFi. When a command is received from one of the forms of communication, the Arduino must return a response through the same channel.

The commands are formatted like the standard commands for programmable instruments (SCPI). SCPI is a standard format for commands for programmable instrumentation devices and it is described as an extension of the IEEE 488.2 standard, which defines the "Standard Codes, Formats, Protocols, and Common Commands" to work over a GPIB bus [26], even if it can be used with other types of hardware. It provides the format requirements to the commands structure, which are ASCII strings composed of one or more keywords separated by colons [27]. So, following a hierarchical structure, the different commands for the instrument are defined. If any argument is given to the instrument at the same time as the command, it comes separated by a white space from the command [27]. For example, one command may be MEAS:BUOY 100, which may indicate that a measure must be taken, that the measurement must be of a buoyant force, and that the measurement must be an average of 100 readings. In order to make this possible, the Arduino must implement some command parsing logic, which can be memory intensive.

The program also must be able to move the motors, connect and disconnect them from the power supply, using the relay, and it must be able to read from the ADC, the remaining analog devices need no intervention from the board. In brief, the board must be waiting for commands, and when one arrives it must parse it and classify it. Then, the command must be executed, and finally the results must be returned to the communication channel.

4.3 Program implementation

Since I already set the specifications for my program, I created a program to make them possible. I decided to work with object oriented programming, in order to make my code more readable and more modular. Therefore, I decided to create two classes, one for the reading and parsing of the SCPI commands, named `SCPI_Handler`, and another one for the control of the hardware, named `core`.

Both classes do not depend on each other, and they communicate between them over the main program, which can be seen in figure 4.3.1. The communication input is handled by the main program and the output is directly handled by the `core` class. So, when a message is received in the main program, it is sent to `SCPI_Handler`, which parses it, and returns a value back with the command to be executed and the argument attached to it. Finally, the main program tells to `core` what command to execute, which argument to use, and where the result must be sent, either MQTT or Serial.

This way, the parsing class does not have to care about any hardware, it only transforms strings in commands and arguments. Its class structure can be seen in figure 4.3.2b, and in repository [35] the complete code can be found. There it can be seen that the process of adding or removing a command in the tree structure is simple, so it can be reused in other devices working with SCPI commands.

Also, the `core` class and the main program does not consider in which format the commands are structured. All the commands' structure can be changed, and this would not affect anything but the `SCPI_Handler` class. And of course any changes in the hardware management may be easily done in the `core` class without affecting the rest of the program. Speaking about the communication methods, it can be easy to implement a new one. The only things to do would be to make the main program listen to more types of input channels, and to make `core` able to write from these new channels. `core` programs' class structure can be seen in figure 4.3.2a, and in repository [35] the complete code can be found.

4.4 Memory constraints

As I said, there are a lot of different Arduino boards. The one I used was the Arduino UNO WiFi Rev2, but before this, I used the Arduino UNO R3. Their specifications can be found in [15] and [25], respectively. So, almost all the code is able to run in the Arduino

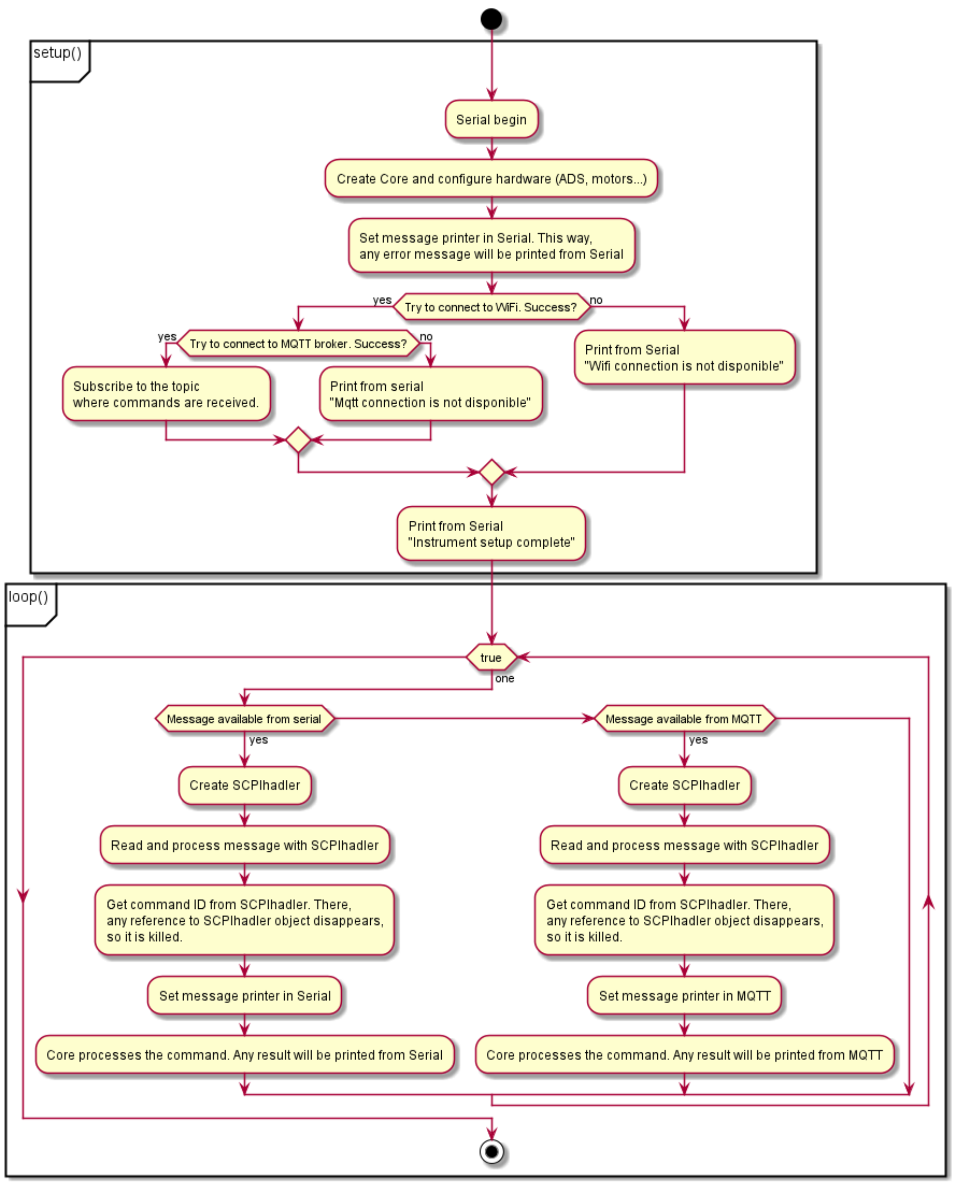
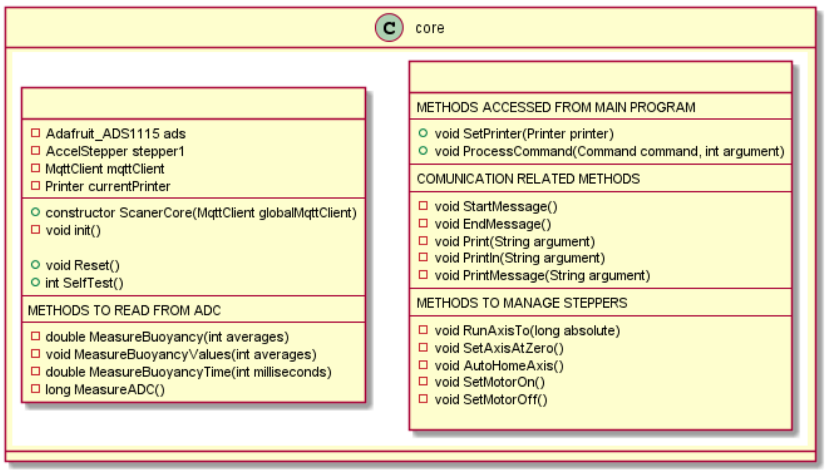

setup()
Serial begin
Create Core and configure hardware (ADS, motors...)
Set message printer in Serial. This way, any error message will be printed from Serial
Try to connect to WiFi. Success?
yes
no
Try to connect to MQTT broker. Success?
yes
no
Print from Serial "Wifi connection is not disponible"
Subscribe to the topic where commands are received.
Print from serial "Mqtt connection is not disponible"
Print from Serial "Instrument setup complete"
loop()
true
one
Message available from serial
yes
Message available from MQTT
yes
Create SCPIhadler
Create SCPIhadler
Read and process message with SCPIhadler
Read and process message with SCPIhadler
Get command ID from SCPIhadler. There, any reference to SCPIhadler object disappears, so it is killed.
Get command ID from SCPIhadler. There, any reference to SCPIhadler object disappears, so it is killed.
Set message printer in Serial
Set message printer in MQTT
Core processes the command. Any result will be printed from Serial
Core processes the command. Any result will be printed from MQTT

Figure 4.3.1: Main programs unified modeling language (UML) activity diagram.

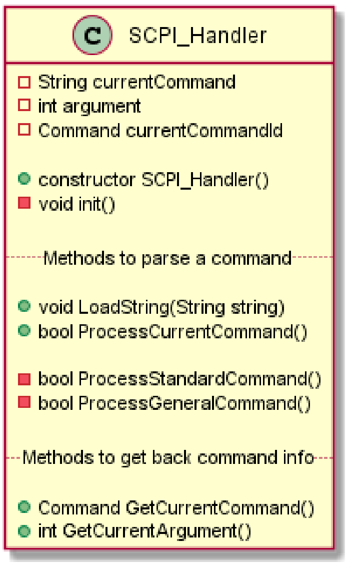
UNO, which has much less dynamic SRAM memory, 2 $KBytes$ in the case of Arduino UNO and 6 $KBytes$ in the UNO WiFi. Also, the WiFi version has more flash memory, 48 $KBytes$ instead of 32 $KBytes$ in the Arduino UNO. Of course, the code related to wireless connection can not be executed in the Arduino UNO, so the WiFi version is just a little more demanding.

In order to make the program runnable in an Arduino UNO, the most restrictive condition is the use of dynamic memory. With only 2 $KBytes$ of SRAM, if a lot of variables are deployed, the microcontroller will run out of memory, and it will crash. If this is considered in a context in which the microcontroller must manage I2C communication, digital signal processing, steppers control with their drivers, incoming commands parsing and process, and sending operation results back to a PC, it becomes a non-trivial task. Now I will enumerate the most memory demanding parts of the program.

First of all, the libraries I used. I used a library for MQTT connection, another one for


core
Adafruit_ADS1115 ads
AccelStepper stepper1
MqttClient mqttClient
Printer currentPrinter
constructor ScanerCore(MqttClient globalMqttClient)
void init()
void Reset()
int SelfTest()
METHODS TO READ FROM ADC
double MeasureBuoyancy(int averages)
void MeasureBuoyancyValues(int averages)
double MeasureBuoyancyTime(int milliseconds)
long MeasureADC()
METHODS ACCESSED FROM MAIN PROGRAM
void SetPrinter(Printer printer)
void ProcessCommand(Command command, int argument)
COMUNICATION RELATED METHODS
void StartMessage()
void EndMessage()
void Print(String argument)
void Println(String argument)
void PrintMessage(String argument)
METHODS TO MANAGE STEPPERS
void RunAxisTo(long absolute)
void SetAxisAtZero()
void AutoHomeAxis()
void SetMotorOn()
void SetMotorOff()

(a) The UML class diagram of `core`.


SCPI_Handler
String currentCommand
int argument
Command currentCommandId
constructor SCPI_Handler()
void init()
Methods to parse a command
void LoadString(String string)
bool ProcessCurrentCommand()
bool ProcessStandardCommand()
bool ProcessGeneralCommand()
Methods to get back command info
Command GetCurrentCommand()
int GetCurrentArgument()

(b) The UML class diagram of `SCPI_Handler`.

Figure 4.3.2: UML class diagrams of program classes.

WiFi connection, another to control the steppers and a last one to manage the ADC using I2C. Only the libraries and the objects to use them use 47% of the flash memory and 13% of the SRAM memory in the Arduino UNO WiFi Rev2. And in the Arduino UNO R3, without using MQTT and WiFi libraries, 19% of the flash memory and 6% of the SRAM memory are used. However there is not a lot to do about this, I can not avoid using these libraries without getting lost with a lot more code.

After this, another demanding but not avoidable part of the program is the `core` class. I must have a `core` object deployed through all the execution of the program, and this class has to manage the steppers and the ADC using an object for each of them, because their state has to be saved over time. The steppers' absolute position and the ADC's selected gain, for example, have to be stored in the memory. Also, the `core` class has to manage MQTT connections to send back the output to the external controller.

Nevertheless, there is a part of the program which can and must be lightened. The `SCPI_Handler` class does not store any relevant information, it simply takes a string like input and it returns a command and its arguments. So, having an object of the class `SCPI_Handler` deployed at runtime consumes memory that is not being used, most of the time. Having this, in fact, makes the program crash the Arduino UNO R3 when a relatively complex command is executed, which needs some local variable to be created. The board runs out of dynamic memory, and it reboots. This is because the string operations in general are too heavy in the Arduino, and the `SCPI_Handler` class uses a class variable to store the current command, so this also consumes some memory.

An easy solution to this problem is to create and destroy a `SCPI_Handler` class object each time a command must be parsed. This way, I can avoid any local variable of the class `core` from being deployed at the same time as any `SCPI_Handler` object. And now yes, the Arduino has enough memory to have a `core` object deployed but without doing anything and a `SCPI_Handler` object deployed and working, or just a `core` object working. The implementation of this beside all the main program's structure can be seen in figure 4.3.1, and the complete code can be consulted at repository [35].

With all this firmware configuration, the program runs on the Arduino UNO WiFi rev2 correctly, and can also run on the Arduino UNO R3 if all the MQTT functionalities are removed.

4.5 Commands set

With the Arduino running its program, the external controllers can control it using commands sent over USB or over MQTT. The command set I programmed for controlling the device do not match all the IEEE 488.2 standard requirements [26], and neither does with all the SCPI commands characteristics, but they are inspired by them. In the table 4.5.1 all the usable commands are explained. If a command is sent to the Arduino while it is processing a previous one, it will ignore it.

Table 4.5.1: Supported commands for controlling the scanner. Note that the names between () indicate arguments, for example, MEAS:BUOY 100.

Command	Description
*IDN?	Returns the machine identification string.
*RST	Resets the machine.
*TST?	The machine makes a self test.
MEAS:BUOY (n)	Returns the average of n readings of the ADC.
MEAS:BUOY:TIME (ms)	Reads the ADC for ms milliseconds, and returns the average of all the readings.
MEAS:BUOY:VALUES (n)	Returns n readings from the ADC, in order to be filtered in an external device.
OUTP:MOVE (pos)	Moves the steppers to the absolute position pos, in steps relative to the origin. The origin is set to 0 at startup.
CONT:CONF:MOTR:ON	Activates the relay to turn on the stepper motors.
CONT:CONF:MOTR:OFF	Deactivates the relay to turn off the stepper motors.
CONT:CONF:AXIS:HOME	Sets the current motors position as 0.

5 Software beyond the machine

5.1 MQTT

I have already talked about MQTT and about how it has been implemented in Arduino, but I have not explained it completely yet. As I said, it is a network communication protocol over which a machine can send and receive messages. In the case of the scanner, it receives commands from the external controller and sends the readings back to it.

As a network communication protocol, it requires a network transport protocol to carry messages from one point to another. The most common case is, of course, to use TCP/IP, because it is the protocol over which the internet works. So, MQTT uses the Internet protocol (IP) addresses and ports to address messages. Thus, a device that wants to use MQTT must be connected to a network, for example being connected to a local area network (LAN) which is connected to the Internet via a router. So, a MQTT device must have an IP address.

Since MQTT is a scalable communication protocol which can handle devices with limited resources or limited bandwidth, it can not depend on individual network actors to work. So, it centralizes all the communication flow in a single network element, a central server called the broker. The broker is where any device sends messages, and it also takes care of forwarding these messages to their destination. So, if some device connected to MQTT fails, the remaining would not be affected. And this also makes it possible for devices to communicate with each other without having to know the IP address of the other machine. They only need to know the broker's IP address, and how to tell the broker who the message's recipient is.

The MQTT message addressing system is based on a publish subscribe architecture. Any device can publish or subscribe to topics. Topics are message channels, identified with a string and managed by the broker, where any published message is immediately sent back to all the devices that are subscribed to the topic. So, if a device wants to send a message to another, they must know which topic to use. In figure 5.1.1 the process can be seen. First, the device which is going to receive the message subscribes to the selected topic, and the broker starts tracking it. Then, the other device publishes the message to the same topic. So, the broker forwards the message to anyone that has previously subscribed to this topic, in this case, the first device. This structure gives a lot of flexibility, allowing many devices to be subscribed or to publish to the same topic.

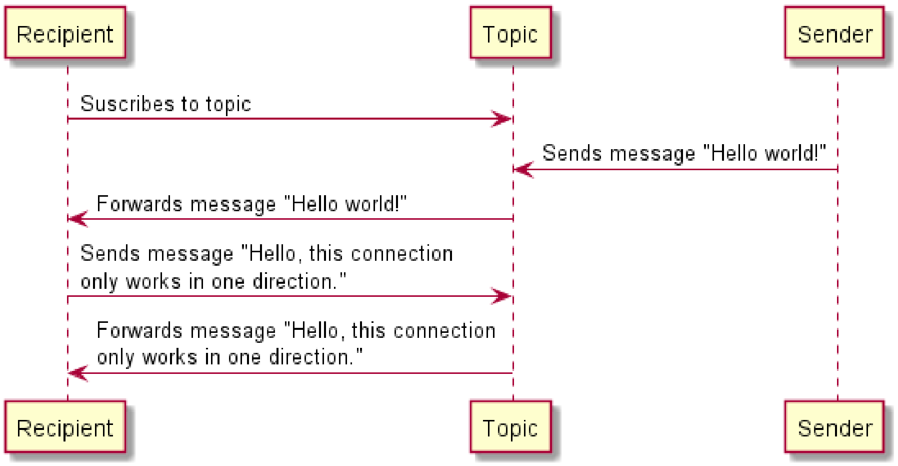

Recipient
Topic
Sender
Suscribes to topic
Sends message "Hello world!"
Forwards message "Hello world!"
Sends message "Hello, this connection only works in one direction."
Forwards message "Hello, this connection only works in one direction."
Recipient
Topic
Sender

Figure 5.1.1: A message delivering process in one topic.

In my case, I simply need the scanner to communicate with an external controller. So, my MQTT system is very simple. I have one channel in which commands are written by the controller, and another where results are written by the scanner. If a controller tries to control the scanner when it is not connected to MQTT, all commands would be lost, and the same would happen in the other direction. However, thanks to using MQTT I am able to control a scanner from many devices, or to control it from one device but get the readings in another one.

In order to make it possible to use MQTT, I first had to deploy a broker on a server. It could be deployed in a PC, in a dedicated server, or in any other device whose IP address can be accessed from the Arduino. I decided to use a Raspberry Pi 3 Model B [28]. It is a single board computer, in which I installed a Linux distribution. The board can be connected to LAN, so I connected it to the same LAN as the Arduino and the PC I used to control it.

To deploy a broker in the Raspberry, I had several options. Brokers are computer programs that can handle MQTT connections, but there are many different broker programs available, both open source or proprietary. The one I chose was EMQX [29], because it is simple, open source and has a by default dashboard which allows one to monitorize the MQTT network's status in real time. Also, it is easy to deploy in Docker [30], which is a program that automates application deployment in containers, pieces of software that are able to be executed without the need of an independent operating system, avoiding the creation of virtual machines. So, I installed Docker in the Raspberry Pi 3, I deployed EMQX broker in a Docker container, and I forward it in the port 1883 of the Raspberry, allowing any external device to connect to it, create topics, and communicate over MQTT.

5.2 Python environment to work with the machine

As I have been saying, the scanner must be controlled from the output in order to make it work. In fact, the scanner itself is unable to perform a scan on its own. As seen in subsection 4.5, it only returns data from the reading it is performing in each moment. So, an external software must take care of handling this data and of making the reconstruction. For that I created a Python library, `immersion_scanner_lib.py`, which can connect to a scanner via serial connection or MQTT, can send commands to it and receive data back, and can store and process this data to perform scans.

For the communication via serial connection using USB, I used the virtual instrument software architecture (VISA), which is an application programming interface (API) usually used to communicate with test and measurement instruments [31]. It allows the development of bus independent programs, delegating the communication with the specific device driver to the VISA Communications Driver, a program which supports many of the different buses that are used for instrumentation, for example, GPIB and USB. So, using a Python library to use VISA API, pyvisa [32], I manage the serial connection with the scanner. In the case of MQTT connection, the library simply connects to a broker to subscribe and publish in topics.

The `immersion_scanner_lib` consists of a single class, `immersion_scanner`, to be simple to be used as a library. As can be seen in figure 5.2.1, all the functionality to manage the connections, send commands, receive responses back and perform scans is stored in there. Also the calibration is in charge of this class. So, a user who wants to use the scanner would create a `immersion_scanner` object specifying which connection to use and where the connection can be found, and all the functionality that has been mentioned would be directly available.

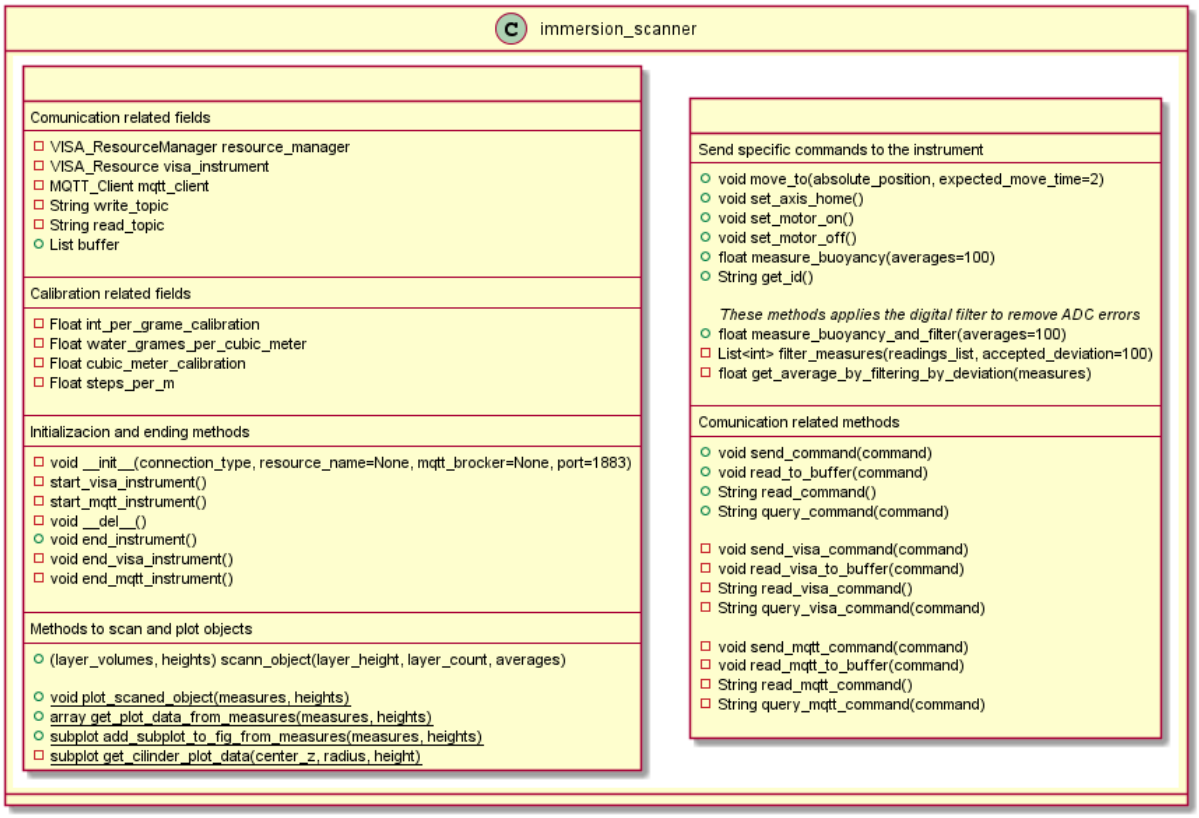

immersion_scanner
Comunication related fields
□ VISA_ResourceManager resource_manager
□ VISA_Resource visa_instrument
□ MQTT_Client mqtt_client
□ String write_topic
□ String read_topic
○ List buffer
Calibration related fields
□ Float int_per_grame_calibration
□ Float water_grames_per_cubic_meter
□ Float cubic_meter_calibration
□ Float steps_per_m
Initializacion and ending methods
□ void __init__(connection_type, resource_name=None, mqtt_brocker=None, port=1883)
□ start_visa_instrument()
□ start_mqtt_instrument()
□ void __del__()
○ void end_instrument()
□ void end_visa_instrument()
□ void end_mqtt_instrument()
Methods to scan and plot objects
○ (layer_volumes, heights) scann_object(layer_height, layer_count, averages)
○ void plot_scaned_object(measures, heights)
○ array get_plot_data_from_measures(measures, heights)
○ subplot add_subplot_to_fig_from_measures(measures, heights)
□ subplot get_cilinder_plot_data(center_z, radius, height)
Send specific commands to the instrument
○ void move_to(absolute_position, expected_move_time=2)
○ void set_axis_home()
○ void set_motor_on()
○ void set_motor_off()
○ float measure_buoyancy(averages=100)
○ String get_id()
These methods applies the digital filter to remove ADC errors
○ float measure_buoyancy_and_filter(averages=100)
□ List<int> filter_measures(readings_list, accepted_deviation=100)
□ float get_average_by_filtering_by_deviation(measures)
Comunication related methods
○ void send_command(command)
○ void read_to_buffer(command)
○ String read_command()
○ String query_command(command)
□ void send_visa_command(command)
□ void read_visa_to_buffer(command)
□ String read_visa_command()
□ String query_visa_command(command)
□ void send_mqtt_command(command)
□ void read_mqtt_to_buffer(command)
□ String read_mqtt_command()
□ String query_mqtt_command(command)

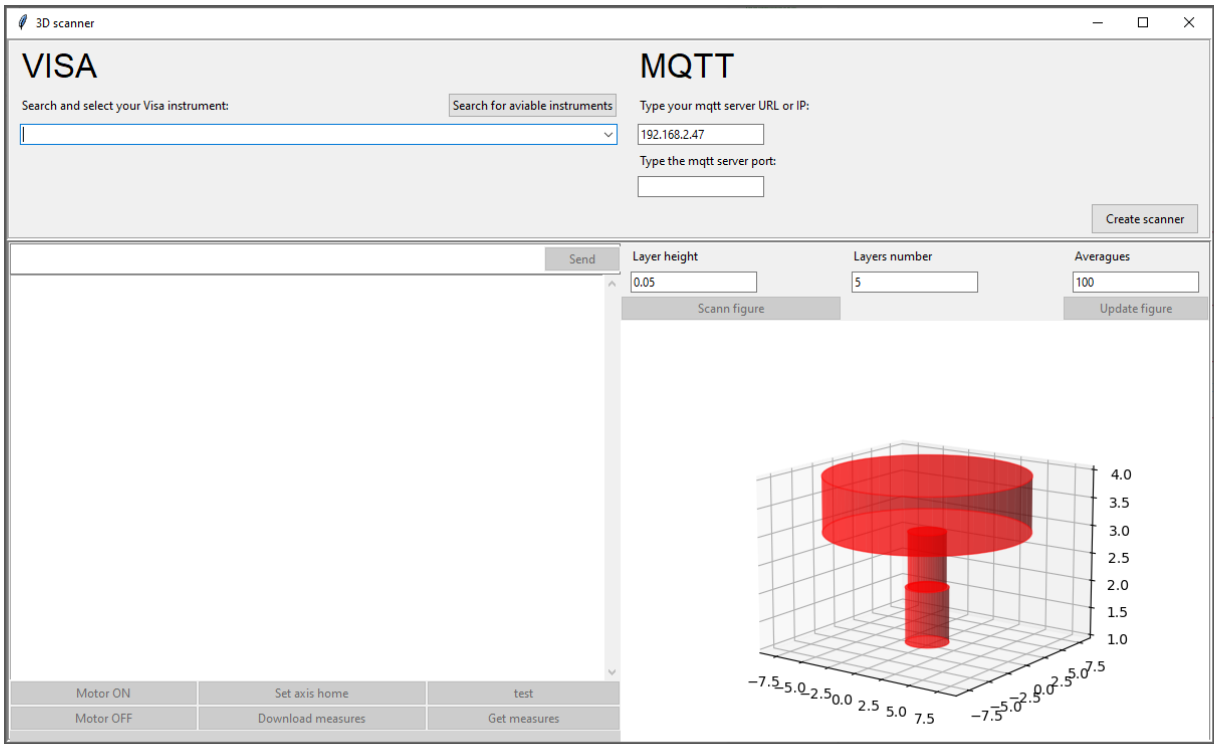
Figure 5.2.1: The UML class diagram of `immersion_scanner`.

To be able to easily access this class and use the scanner through the PC, I created a desktop application with Python, using tkinter [33], which is a library to make graphical user interfaces (GUI). Its layout can be seen in figure 5.2.2. It allows the user to create the connection with the device over VISA or MQTT, to send commands and receive back data, and to scan objects using `immersion_scanner`. It also has some shortcuts to the most commonly used commands. Its complete code can be found, alongside the code of `immersion_scanner_lib`, in repository 36.

6 Measurements

With the scanner complete and working, I performed some measurements to see how it works. I tried to scan 3 different objects, which are the cylinder I used for testing, shown in 3.3.2, a cone which can be seen in figure 6.0.1a, and a piece with curved shapes described in figure 6.0.1b. They were scanned and the shape that the scanner predicts for them is overall correct, these results can be seen in 6.0.2.

When I do a scan, I can configure some parameters. I can adjust the slice height, the number of slices, and how much readings of the ADC will be used to calculate each measurement, with the technique that was explained in subsection 3.6. So, performing readings with different configurations, I can compare between different data, to see how the scan configuration affects the reconstruction quality. For comparison, I will use two different figures



Search and select your Visa instrument:

Type your mqtt server URL or IP: 192.168.2.47

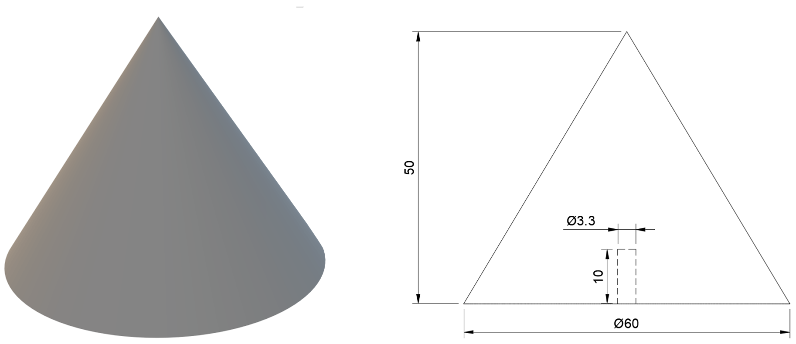
Type the mqtt server port:

Layer height: 0.05

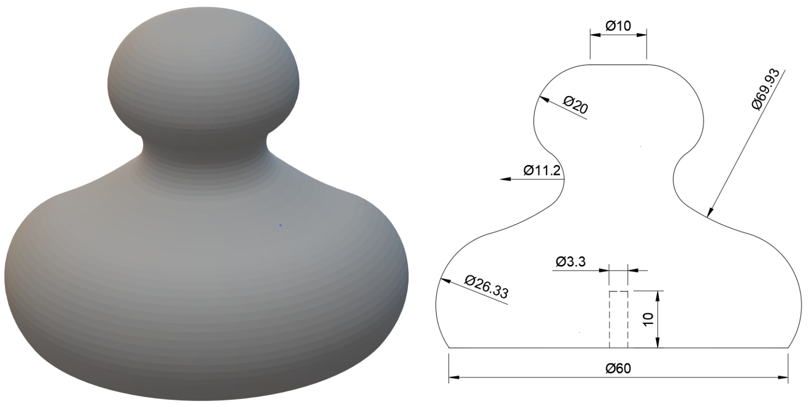
Layers number: 5

Averagues: 100

Figure 5.2.2: The layout of the desktop application for controlling the scanner.

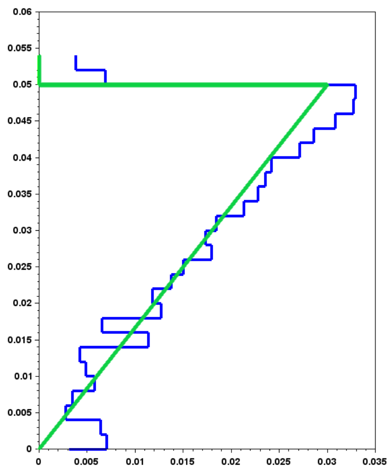


(a) The cone.

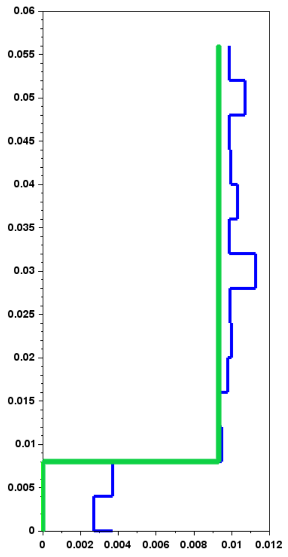


(b) The piece with a curved shape.

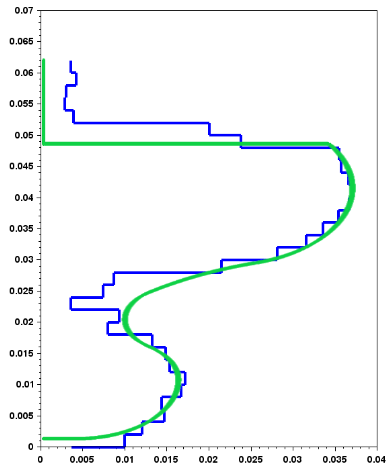
Figure 6.0.1: The pieces to be scanned. All distances are in *mm*.



(a) The cone scan.



(b) The cylinder scan.



(c) The piece with a curved shape scan.

Figure 6.0.2: The results of the scanning of the pieces. In blue, the predicted radius for each slice, and in green the real one. All axes are in meters.

of merit.

First of all, I will compare the standard deviation in volume readings. The scanner itself only can measure volumes, so I must evaluate its precision. To do it, I will get each of the individual measurements of the volume immersed, and I will compare them with the real ones, which I can know because I know the measured object's properties. Then, I will calculate the normal deviation of the difference between them, and this will be my first figure of merit.

However, what is intended with the scanner is to rebuild objects, so I must evaluate this aspect of the process too. In order to do it, I can measure how good the scanned objects are, compared to the real objects. Therefore, to evaluate the quality of the reconstruction, I will take the predicted volume of each slice, and compare it to the real volume these slices have. With the difference between them I will calculate the normal deviation. But with higher slices it is easier to get a precise volume prediction, because the deviation in each of the two readings that are used to get it will be relatively smaller. So, to represent that scans with higher slices are more accurate in the representation of each slice's volume, I will use the previous normal deviation value divided by the slice height. It represents how much the cross section typically deviates from the real one, in m^2.

To measure the figures of merit I used the cylinder, because its shape is ideal to be scanned with different slice heights, and the volume of each real slice is easy to calculate. First of all I scanned it with different amounts of readings for each measurement, and I measured the normal deviation of each individual volume measurement. The results can be found in table 6.0.1. A slice height of 4 mm was used.

Table 6.0.1: Normal deviation of volume measurements with different number of readings of the ADC for each volume measurement.

Number of readings of the ADC	Normal deviation of the measurements in m^3
125	$3.2 \cdot 10^{-6}$
250	$2.7 \cdot 10^{-6}$
500	$1.8 \cdot 10^{-6}$
1000	$1.6 \cdot 10^{-6}$
2000	$1.4 \cdot 10^{-6}$

As expected, the normal deviation of each individual measurement is higher the less readings are used for each measurement. Anyway, the more the number of readings increases, the standard deviation decreases at a lower rate. So, if a very high precision is not required, 500 seems like a good number of readings for each measure.

To continue with the analysis, I measured how the reconstruction changes with the change of the slice height. For that, I measured the second figure of merit, using the cylinder and 2000 readings for each measurement. The results can be seen in table 6.0.2. Two of the reconstructions can be also seen in figure 6.0.3.

Table 6.0.2: Normal deviation of slice volume predictions divided with slice height, with different slice heights and 2000 readings for each measurement.

Slice height in m	Normal deviation of the predictions in m^2
0.002	$0.2516 \cdot 10^{-3}$
0.004	$0.2002 \cdot 10^{-3}$
0.006	$0.1520 \cdot 10^{-3}$
0.008	$0.1052 \cdot 10^{-3}$

Again as expected, the section of the object is better predicted the higher the slices are. However, the higher the slices, the lower the resolution of the scan, and curved objects render worse. So, a compromise must be made between resolution and accuracy. It is possible to scan an object with very low slice height and try to represent its curbs very well, but the reconstruction can become noisy as in figure 6.0.3. This shows that with this technique taking into account the scanned objects characteristics is very important. In fact, if objects with some very curbed parts and other more flat parts are scanned, the process should be made with a non constant slice height.

7 Conclusions and future work

A new and underdeveloped way to scan objects was proposed. It has some unique characteristics compared to other scanning methods, which gives it some potential to be better in specific tasks or to complement other methods. However, its main points in favor are its

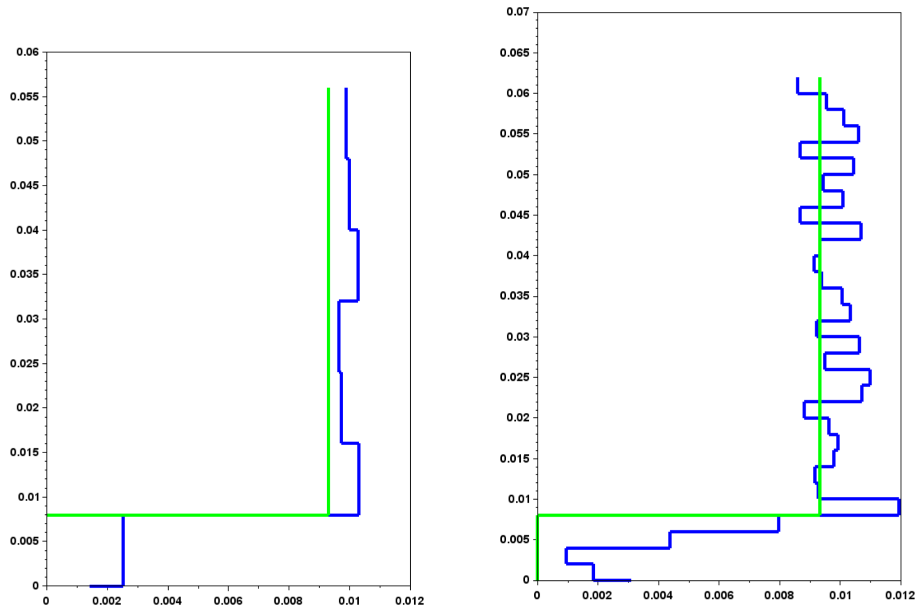


Figure 6.0.3: The reconstructions of the cylinder (blue) with a slice height of 0.002 m (left) and with a slice height of 0.008 m (right), over the real cylinder (green), using 2000 readings for each measurement. Note that when decreasing slice height, the reconstruction becomes noisy.

reliability, since it gets direct measures of objects' properties and because of its potentially good resolution. So, to get the maximum benefit from the technique, the resolution needs to be improved regarding the one I have in my measures in section 6. This work is just an approach to the technique, a lot of work can be done in order to explore its possibilities. For example, real experiments about the applications explained in subsections 2.3, 2.4, 2.5 and 2.6 can help to determine the real usefulness of the technique in these tasks.

About the machine itself, a lot of work can be done to improve it. I only used one sensor, but so many others can be tested to see which is the better one, or even they can be used in combination. The tests done with the LVDT were promising, and if a good way to attach it to a buoy is found they can be easily continued. About the electronics, it has been very hard to make readings in such a little signal. Especially considering the presence of the motors and the rest of the devices, like the ADC or the Arduino itself, which causes interference to the signal. In the end, the PCB I manufactured did not work because of the interference of the common independence at the source. Perhaps it would have been better to try another approach, and place the ADC directly in the load cell module. In fact, the HX711 [20] includes an integrated precision reference, prepared to supply a Wheatstone bridge. So, the ADC could have been used next to the load cell, supplied directly with the 5 V from the Arduino, and this would not have been a problem, due to the HX711's precision reference. Also better ways to get a precision reference can be explored, apart from the LM329 [23]. Doing so can help to avoid a lot of interference from the rest of the circuit.

About the firmware and the software, they are functional, even not definitive. Some details about the better way to perform communications over MQTT and the serial bus must be decided yet, related to the maximum length of messages in MQTT. Also, the definitive command set must be decided and implemented, because there are some commands that should be defined yet, above all commands to check machine status, and there are commands that probably are not really necessary, or which functionality should be revised. Another important question about this subject is to consider if the machine should be able to perform scans on its own, or if it can depend on an external controller to perform the reconstruction, like it does now.

The external programs to control the device are also usable only in limited cases of use. A scanner can now be easily controlled using Python, but it is not controllable from any other framework without performing some low level programming. So, its integration with other frameworks should be enhanced. Also, these kinds of devices are usually used over Ethernet, so the possibility to connect the scanner with it should be considered too. Finally, if devices of this type are intended to be used in an industrial environment, they must get a lot of industry cybersecurity features, in order to be secure to use and connect to a network.

Finally, speaking about the measurements, the machine is able to distinguish volumes with a precision of cubic centimeters, so it is not bad at all, considering the instruments used. Using a better load cell and a better ADC, higher resolutions should be reached. Also, the calibration I performed in subsection 3.7 may be not as accurate as it must be. Some figures, like figure 6.0.3, suggest that the device can obtain better measurements if it is better calibrated. And as I have said in section 6, to get the best reconstruction possible of an object, the scan's configuration must be adjusted to the object's characteristics.

References

[1] S. Rode, H. Singh, N. Thube, M. Mhaske, H. Ansari and M. Vadhel, "Design and Fabrication of a 3-D scanning system using Optical Computed Tomography and Photogrammetry," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1-5, doi: 10.1109/ICNTE51185.2021.9487696.

[2] J. Liu, Q. Sun, Z. Fan and Y. Jia, "TOF Lidar Development in Autonomous Vehicle," 2018 IEEE 3rd Optoelectronics Global Conference (OGC), 2018, pp. 185-190, doi: 10.1109/OGC.2018.8529992.

[3] Z. Chen and L. Li, "Material Decomposition of Energy Spectral CT by AUTOMAP," 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC), 2018, pp. 1-3, doi: 10.1109/NSSMIC.2018.8824439.

[4] Carl Zeiss AG, “Computed Tomography, Made by ZEISS”, document EN_60_025_0003I.

[5] Carl Zeiss AG, “More than a robot. The Hambot.”, document EN_60_020_0029II.

[6] M. Abdelmomen, F. O. Dengiz and M. Tamre, "Survey on 3D Technologies: Case Study on 3D Scanning, Processing and Printing with a Model," 2020 21st International Conference on Research and Education in Mechatronics (REM), 2020, pp. 1-6, doi: 10.1109/REM49740.2020.9313881.

[7] J. Ren, M. You, H. Wang, B. Tang and Y. Liu, "A comparative evaluation of cone beam computed tomography and multi-slice computed tomography on the volume of tooth in-vitro," 2021 IEEE International Conference on Medical Imaging Physics and Engineering (ICMIPE), 2021, pp. 1-5, doi: 10.1109/ICMIPE53131.2021.9698963.

[8] Z. Cai, C. Jin, J. Xu and T. Yang, "Measurement of Potato Volume With Laser Triangulation and Three-Dimensional Reconstruction," in IEEE Access, vol. 8, pp. 176565-176574, 2020, doi: 10.1109/ACCESS.2020.3027154.

[9] K Aberman, "Dip transform for 3D shape reconstruction," in ACM Transactions on Graphics 36(4):1. Association for Computing Machinery, 2017, doi: 10.1145/3072959.3073693.

[10] Zhiyuan Liu and Jie Zhou, *Introduction to Graph Neural Networks.* San Rafael, CA: Morgan & Claypool, 2020, pp. 16-17.

[11] Zhiyuan Liu and Jie Zhou, *Introduction to Graph Neural Networks.* San Rafael, CA: Morgan & Claypool, 2020, pp. 14-16.

[12] Keras-team. [Online]. Available: `https://github.com/keras-team/keras`. [Accessed: 2022 Jun. 22].

[13] Shenzhen Getech Technology Co.,Ltd. Shenzhen, PRC. "Geeetech Prusa I3 Pro W 3D Printer DIY kit". [Online]. Available: `https://www.geeetech.com/Documents/I3_pro%20W%20packing%20list.pdf`. [Accessed: 2022 Jun. 7].

[14] E-pulse Servicios de Internet SL. Carballo, La Coruña, Spain. c2005-2022. [Online]. Available: `https://tienda.bricogeek.com/sensores-presion/1291-celula-de-carga-5kg-con-amplificador-hx711.html?search_query=celula+de+carga&results=5`. [Accessed: 2022 Jun. 7].

[15] Arduino. Somerville (MA). c2022. [Online]. Available: `https://docs.arduino.cc/hardware/uno-wifi-rev2`. [Accessed: 2022 Jun. 7].

[16] Y. Hu, Y. Li, C. Zhou and S. Zhang, "Development of a large range piezoelectric hot stamping forming force measurement sensor," 2021 4th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM), 2021, pp. 409-412, doi: 10.1109/WCMEIM54377.2021.00089.

[17] A. Pietrikova, S. Zuk and I. Vehec, "Coplanar Capacitive Liquid Level Sensor," 2019 42nd International Spring Seminar on Electronics Technology (ISSE), 2019, pp. 1-6, doi: 10.1109/ISSE.2019.8810232.

[18] V. Kedambaimoole et al., "Reduced graphene oxide based resistive liquid level sensor," 2018 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), 2018, pp. 1-3, doi: 10.1109/EDAPS.2018.8680904.

[19] Nathan Ida, "Position and displacement sensing: variable inductance sensors," in *Sensors, Actuators, and Their Interfaces, a multidisciplinary introduction*, 2nd ed. London, United Kingdom: The Institution of Engineering and Technology, 2020, ch. 5, sec. 4, pp. 239–241.

[20] Avia Semiconductor Co., Ltd. Xiamen, PRC. *24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales,*. [Dataset]. Available: `https://www.alldatasheet.com/datasheet-pdf/pdf/1132222/AVIA/HX711.html`. [Accessed: 2022 Jun. 7].

[21] Texas Instrument, *Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference,* SBAS444B datasheet, May. 2009. [Revised Oct. 2009]. [Dataset]. Available: `https://cdn-shop.adafruit.com/datasheets/ads1115.pdf` [Accessed: 2022 Jun. 7].

[22] Allegro MicroSystems, Inc. *DMOS Microstepping Driver with Translator and Overcurrent Protection,* 4988-DS datasheet, 2010. [Dataset]. Available: `https://www.allegromicro.com/~/media/Files/Datasheets/A4988-Datasheet.ashx` [Accessed: 2022 Jun. 7].

[23] Texas Instrument, *LM329 Precision Reference,* SNVS748F datasheet, May. 2000 [Revised Apr. 2013] [Dataset]. Available: `https://pdf1.alldatasheet.com/datasheet-pdf/view/761002/TI/LM329.html` [Accessed: 2022 Jun. 7].

[24] KiCad community. [Online]. Available: `https://www.kicad.org/`. [Accessed: 2022

Jun. 22].

[25] Arduino. Somerville (MA). c2022. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed: 2022 Jun. 10].

[26] IEEE Standard Codes, Formats, Protocols, and Common Commands for Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation, Institute of Electrical and Electronics Engineers, 1992, ISBN 1-55937-238-9, IEEE Std 488.2-1992

[27] Electronics Group, "Gpib Programming Tutorial," Free University Amsterdam, The Netherlands, 2000. [Online]. Available: http://g2pc1.bu.edu/~qzpeng/gpib/manual/GpibProgTut.pdf. [Accessed: 2022 Mar. 13].

[28] Raspberry Pi Foundation. Cambridge, England, UK. c2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/. [Accessed: 2022 Jun. 18].

[29] EMQ Technologies Co., Ltd. Hangzhou, Zhejiang, PRC. c2013-2022. [Online]. Available: https://www.emqx.com/en. [Accessed: 2022 Jun. 18].

[30] Docker, Inc. Palo Alto, CAL. c2022. [Online]. Available: https://www.docker.com/. [Accessed: 2022 Jun. 18].

[31] Tektronix, Inc. Beaverton, OR. c2022. [Online]. Available: https://www.tek.com/en/support/faqs/what-visa. [Accessed: 2022 Jun. 19].

[32] Gregor Thalhammer, Matthieu C. Dartiailh. c2005-2022. [Online]. Available: https://pyvisa.readthedocs.io/en/latest/. [Accessed: 2022 Jun. 19].

[33] Python Software Foundation. Wilmington, DE. c2001-2022. [Online]. Available: https://docs.python.org/es/3/library/tkinter.html. [Accessed: 2022 Jun. 19].

[34] Pello Usabiaga "Neural networks to try to rebuild general objects with a liquid immersion 3D scanner". [Online]. Available: https://github.com/PelloUsabiaga/TFG_general_objects.

[35] Pello Usabiaga, "Arduino firmware for a liquid immersion 3D scanner". [Online]. Available: https://github.com/PelloUsabiaga/TFG_firmware.

[36] Pello Usabiaga, "Python environment to control a liquid immersion 3D scanner". [Online]. Available: https://github.com/PelloUsabiaga/TFG_software.