Final Report

Design, modelling and construction of a Mobile Robot

Guillermo Gómez Peña

2017

3rd Year Individual Project

I certify that all material in this thesis that is not my own work has been identified and that no material has been included for which a degree has previously been conferred on me.

Signed.....Guillermo Gómez Peña.....

College of Engineering, Mathematics, and Physical Sciences
University of Exeter

Final Report

ECM3101/ECM3102/ECM3149

Title:
Design, modelling and construction of a Mobile Robot

Date of submission: Tuesday, 08 May 2018

Student Name: Guillermo Gómez Peña
Programme: Mechanical Engineering
Student number: 670057675
Candidate number: 055451

Supervisor:
Dibin Zhu

Abstract

Automation gives efficiency to industrial processes and makes daily lives easier. Industries invest great amounts of money in automatic devices which imply an increase in the profit and quality of the services and products. One way to achieve automation is using AIVs (Autonomous Intelligent Vehicles) in production chains and storages.

The aim of this project was to design and test an AIV or mobile robot, focusing on the logic development. This robot should move from one point to a target one avoiding fixed obstacles. To achieve this, an Arduino board was programmed and connected to three ultrasonic sensors and two stepper motors which allowed it to recognise its position and the environment. The design of the structure was drawn in Solidworks and built in an acrylic sheet, shaped using a laser cutter.

The two main stages of design were: constant development of the logic and integrated robot test. The first stage consisted on checking the performance of each electronic component separately and improving the movement code from simple objectives, such as avoiding just one obstacle, to more complex ones until reaching the most desirable code for the robot. The second stage was putting it all together, the electronics and the structure, in order to accomplish the final movement tests.

The final result was tested using different types of obstacles, distributed randomly in the space, and setting various target points. The robot completed the path correctly if the obstacles were the specific ones. This means that they had a rectangular shape and the enough dimensions to be detected by the sensors. Apart from this, the results show an actual AIV with a logic that could be used for logistic activities in a place which satisfies the conditions. Moreover, the developed code is a solid base to create a bigger and more complex logic that could lead to a wider industrial use.

Keywords: Mobile Robot, Automation, Programming, Arduino, AIV, Mechatronics.

Table of contents

1. Introduction

Industry is constantly developing its processes in order to increase its efficiency in terms of time, costs and quality. The automatization of its processes brings the chance to achieve that aim. It has been implemented in industry since the XVIII century [1] but its ways to achieve it has varied along time. The use of robots is a current one. Robots provide flexibility, security and protection to the workers and also contribute to make processes more accurate and faster [2].

The term robot comes from the Czech word “robota” which means “slave labour” and was first use for fictional creations [1]. Technically the official definition is:

“Actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks” [3].

The scope of robots goes further than its appliance in industrial activities. They are also gaining importance in our daily lives, in a direct or indirect way. To illustrate that, automatic vacuum cleaners or personal assistant robots are seen in service sector whereas assembly robotic arms or painting robots are used in industrial sector.

The aim of this project is to design, test and evaluate a mobile robot for use in engineering applications such as storage or production chains. Its design process is intended to be representative of any other mobile robots with different targets, while its logic has the goal to serve as a starting point to facilitate the development of a higher functionality robot. Throughout this report, the steps for creating a product will be shown. This process of development could be summarized by the following figure:

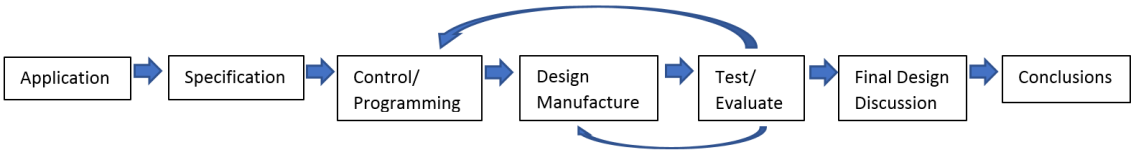


Figure 1. Process of development

The structure of the report is established according to this process. The information that shapes the project will be detailed in section 2. Once the project has been contextualized, the application of the mobile robot and its consequent specifications will be defined in section 3. Moreover, the planning and the explanation of concepts that were used along the development process are also mentioned in that section. In section 4 and 5, the designing process and the final result, respectively, will be presented in detail, describing its performance objectively and highlighting its strengths and weaknesses.

With all the features of the mobile robot clarified, a conclusion will reveal if all the required features are covered. It would be also discussed its possible introduction and application in an industrial process, commenting feasible improvements that could be made. The conclusion and the discussion will appear in section 6.

At the end of the report, the management of the project and the measures taken for avoiding risks are explained. These two aspects will be also accompanied by a discussion about the impact that the mobile robot could have in terms of economy, sustainability and society. All these aspects will be covered in section 7.

2. Literature review

2.1. Contextualization

Automation can be defined as the replacement of humans by mechanic or electronic devices which could do their tasks. It started with the Industrial Revolution, in the second half of the XVIII century in UK. Before the appearance of programmable devices, it consisted on a mechanization process, with new technologies such as the James Watt's steam engine (1769), Jacquard loom (1801) or Ford's model T production (1908). It was not until 1950s, with the numerically controlled machine tool made by MIT, that it could be talked about "robots". The first industrial robot was introduced by General Motors (New Jersey) in 1961. Since then, the use of industrial robots began to grow, spreading its production to Japan and Europe. 66000 robots were used in 1983 [1]. The most common robots were manipulator robots, which worked fixed in a place doing repetitive tasks. As they could only work in a specific area with movements restrictions, mobile robots where created as an alternative [4]. A mobile robot is:

"Robot able to travel under its own control. A mobile robot can be a mobile platform with or without manipulators" [3].

It can be distinguished between two types of mobile robots:

- AGVs (Autonomous Guided Vehicles): These robots need an external help to be able to move between two different points such as a track.
- AIVs (Autonomous Intelligent Vehicles): These robots can move independently. AVLs became popular at the end of the XX century as an alternative of those which could move only following a track. They offer more flexibility of movement and a higher level of perception [4].

Although mobile robots tend to be considered service robots, they have several applications in the industry. Service robots can be divided in two groups: professional use and public/domestic use.

The main applications of robots for professional use are: Logistic systems, which could be used in a production chain or in storages; Field robots, used in agriculture and livestock; Medical robots; Defense robots; Public relations robots; Powered human exoskeletons; construction, professional cleaning, inspection and maintenance systems, rescue and security applications and underwater systems [5].

The mobile robot that will be developed in this project is going to be a AIVs for use in a logistic system. Today, logistic robots are used in large storages as:

- Port of Rotterdam: it is a practically full automated terminal which use AGVs to move containers to the storage area. "They eliminate tasks that are dangerous, do not mind doing work that is dull or repetitive, and are more efficient, more accurate and stronger than people. These unmanned vehicles recognize even when they are running out of battery and go to the swap station" [6].
- Amazon: the famous delivery company use around 80000 mobile robots in their storages since 2015. In Europe, these automatic storages are still settling down. The robots can move 1300kg and they reach 1.7m/s. They load the shelves in the storage and create ways depending on how urgent a product is, how its availability is and their location. They also have Wi-Fi connection to be controlled. The main advantages are that the moves are reduced, the efficiency increase, the corridors are eliminated and therefore space is gained. As a result, there is more products available and less delivery time which means more income [7], [8].

The forecast of unit sales shows a positive future for professional robots. In figures 2 and 3, it can be seen the development of the sales in the second half of the 2010s.

Figure 2. Unit sales Professional A [5]

Figure 3. Unit sales Professional B [5]

Domestic service robots have other applications such as household activities or entertainment and leisure toys. Their sales development forecast is shown in figure 4.

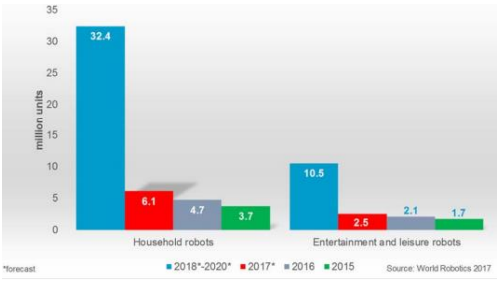


Figure 4. Unit sales Domestic [5]

Regarding the design, it can be distinguished between the logic and the body design [9]. The logic design involves the mapping by using sensors such as MEMS or Ultrasonic sensors [10]. SLAM (Simultaneous Localization and Mapping) techniques are an issue of investigation in order to minimize the time needed to cover a terrain and to avoid uncertainties in the position. There is also being research methods to navigate without relying on a map [11]. Other important issue in the logic field is the development of swarm systems, in which robots work coordinated [12].

The body design covers the chassis of the robot. Playing with materials and shapes can make a robot cheaper and more efficient in terms of dynamic and cinematic behavior [13].

2.2. Projects on the area

There is a wide range of projects covering the subject of mobile robots. They address different topics about them, from how they process the information received by the sensors to how to get a better mapping algorithm. Between all the existing information, the projects read more widely are going to be detailed below. They highlight some of the actual issues about mobile robots.

A project carried out by P. K. Mvemba, S. K. Guwa Gua Band, A. Lay-Ekuakille and N. Giannoccaro covers an analysis about the use of ultrasonic sensors in mobile robots. They based that study on the construction of a low cost mobile robot, defending that this type of sensors is used for cheap appliances, such the one that was built for this project. The two most remarkable aspects of that article were: the analysis of ultrasonic sensors and the way that the code is developed. They explained that

the distance is calculated by multiplying the half of time that the ultrasonic pulse lasts from leaving the sensor until returning and the speed of sound. To be accurate, it is necessary to consider that the speed of sound depends on the external temperature and humidity. As their report said, the humidity can be neglected but the effect of temperature should be considered. Therefore, an error in the lecture of the sensors should be expected in this project, as the speed of sound used is 343,2 m/s, independently of the external temperature [14].

In terms of the logic, they developed a fuzzy algorithm, which consists on conditional statements. They used only one sensor that could turn from 15 to 165 degrees, giving the chance to read three areas: right, left and front. Then, through a PWM output microcontroller, they were able to control two dc motors. The sensor sent for each area the signal "far" or "near" in order to move each motor "forward" or "reverse", depending on the combination of the information received [14].

Another project of interest was made by Y. Kim, D. Shin, J. Lee, Y. Lee and H.-J. Yoo. They studied the logic development of a mobile robot with artificial intelligence (AI) due to its great level of attention received by unmanned applications such as package delivery, smart surveillance and geological surveying. As they said, there are two widely used tree search algorithms for path planning of autonomous robots: A* and iterative deepening A*. These methods consist on choosing a path to reach the target point by evaluating the different positions around the robot in terms of distance to the target point. The formula to calculate that distance or, also called, cost is $f(s) = g(s) + h(s)$, where $g(s)$ is the cost of the path from the start position and $h(s)$ is the heuristic cost of the path to the target point [15].

A. Caverzasi, F. Saravia, O. Micolini, L. Mathe and L. Lichtensztein developed another mobile robot with the focus on the location and the creation of a map. They used three coordinates to ubicate the robot: x, y and θ. The robot travels by moving the Euclidian distance from (x1, y1) to (x2, y2), giving a margin of error in the target point. A Kinect sensor with a camera was used to interpret the environment by identifying particular features. The navigation process was done by three different algorithms: the one previously mentioned, Breath-First Search (BFS) and Dynamic Programming. BFS consist on choosing the optimum path by dividing the space in nodes and check all the combinations of distances that could be done between nodes in order to select the shortest one. A* is based on this method. On the other hand, Dynamic Programming focuses on finding the best path by analyzing the space at different stages due to the decisions taken in each one would affect the future development of the system [16].

2.3. Shaping this project

The mobile robot developed in the project was designed to be made with affordable components. Therefore, the environment recognition will be made by ultrasonic sensors and not detected with a camera. In terms of the logic, this project will develop a program that could be used in simple storages with rectangular shapes where the robots came across specific, punctual and fix obstacles. According to this, from all the methods presented before, this project will use a fuzzy algorithm. The other types of mapping presented, could be used as future improvements for the robot and are also mentioned to express how complex a code for a complete performance could be.

The use of a coordinated plane and the set of space limits for the performance of the robot will be also considered for the logic of the mobile robot.

3. Methodology and theory

This project was completed following an initial plan which set the procedure for the development of both the logic and the structure of the robot. This plan started with the choice of the elements required to build it according to the specifications. Due to the main aim of the project, that consist of creating an AIV with a logic that allows it to move to a certain point avoiding fixed obstacles, the main specifications for the robot were:

A. Capability to avoid rectangular obstacles.
B. Recognition of the position where the robot is placed and capability to reach the target point.
C. The logic of the robot must be applicable for logistic uses such as in industrial storages.

The achievement of these three conditions would lead the project to a successful result. A deeper view of the basis where the project started and the plan that it followed is shown in the next four sections.

3.1. Plan structure

The establishment of a clear plan to follow was essential to achieve the goal of the project. Therefore, the development of this plan was the first stage to complete. The plan set the steps that turn the first idea into a useful and feasible application. As mentioned in the introduction, these steps are:

1. Application: Determine the aim of the product and where could it be used.

   The mobile robot must be recognized as an AIV, capable to move with a considerable level of autonomy. It should reach a defined point, specified from its current position, without being stopped by randomly located fixed obstacles. This behavior could allow it to be a useful appliance in logistic activities in industrial storages.

2. Specifications: Set the main features that the product should satisfy to accomplish its target.

   According to the intended application, the main three specifications are those shown before. Knowing both the specifications and the function of the robot, the elements required could be selected. These elements were divided mainly in two: mechanical and electronic. Further information about their selection process and their own specification are shown in sections 3.2 and 3.3.

3. Programming: Develop the code from simple to complex.

   In order to create an extended and complex code, it is necessary to start doing simple codes which could satisfy one part of the whole behavior of the robot. Checking that they are working properly, they can be turned into more difficult ones until reaching the final version. Therefore, this process required constant testing and modification. Furthermore, each electronic component used was also separately tested.

4. Manufacture: Build the whole product with both mechanical and electronic parts integrated.

   Once the code was developed and checked electronically, the whole assembly was done. In parallel to the logic development, the structure was designed and the mechanical parts were obtained.

5. Test and Evaluation: Check the behavior and solve the problems.

The testing process was done throughout the whole product development. First, it was used during the programing process and then with the whole assembly. The problems were solved either by changing the code or by redesigning the structure.

6. Conclusions: Analyze the final product, referring to the specifications.

After the testing process, conclusions about the result were obtained. They refer mainly to the level of satisfaction of the whole project, the approaching of the product to the specifications and the improvements that could be done in terms of industrial application and its efficiency of operation.

Another aspect to consider is the planning for the development of the program. Before starting to write it, a flow diagram was created with the actions that the robot should do and the connections between them. Initially there were several versions of it, in order to decide which one was the most appropriate for the aim of the project and reliable for its developing process. The selected diagram was updated when errors on the performance were found. Two versions of that diagram are shown in section 4.

3.2. Electronics Basis

The electronic components are the heart of the project. Its selection was based on the specifications. Firstly, an Arduino UNO was used as the programmable controller that load the code and run the orders. It controlled the rest of elements and had the role of the brain of the robot. Its election was made due to its simplicity of use and its efficiency of work. Arduino UNO has 14 digital pins that can be used as Inputs or Outputs, sending a "high" (5V) or "low" (0V) signal. It also has six analog Inputs/Outputs, where continuous signals are treated. It is connected to the computer by an USB port and can be powered by that port or by an external supply using the pin "Vin". Moreover, it powers other components through the pin "Vout".

The language that is used to program an Arduino UNO is an adaptation of C++. It has a great range of functions, statements and data types so only the ones used for the project will be explained in this section. The data types which mainly appear in the code are "int" and "float". The first type refers to integer numbers and the second one refers to rational numbers. This data sometimes follows the word "const", which makes the program unable to change the values.

The functions used are:

- digitalWrite(pin,state): it tells a pin to be in a high or low state [17].
- pinMode(pin,mode): it tells a pin if it is an Input or an Output [17].
- delay(time): it pauses the program for milliseconds [17].
- delayMicroseconds(time): it pauses the program for microseconds [17].
- pulseIn(pin,value): it reads a pulse that comes from an specific pin [17].
- Serial.begin(): it sets the data rate in bits per second for serial data transmission [17].
- Serial.printIn(): it prints data to the serial port, followed by a carriage return character [17].

The most used statement was the conditional one which is declared by writing: if (condition) {statement one} else {statement two}. The statements are placed in the las section of the Arduino program. The code is split in three parts: declaration of variables, the code that only runs once and the

code that runs continuously. This two last sections are written in the program as “void setup(){code}” and “void loop(){code}”.

The second choice was the motors. As the robot should recognize its position respect to the target point, stepper motors were chosen instead of dc motors or servo motors. Stepper motors offer the chance to control their turn and make easier the process to determine the distance that the robot has travelled, knowing that 200 pulses or repetitions of the code correspond to one complete turn. However, it is also important to point out that their main disadvantages are that their energy consumption is higher than other motors and that their force decreases when its speed increases [18].

In the market, there are two types of stepper motors: unipolar and bipolar. Independently of their type, stepper motors have two parts: rotor and stator. The stator is made by coils, two for bipolar motors and four to unipolar ones. The principle of movement of unipolar motors is that the current magnetize different pair of coils during time, creating a movement in the magnetic field. While activating and deactivating coils, the rotor turns searching for the magnetic balance [18].

Bipolar motors behave differently. The current that goes through them changes its direction continuously. As a result, the magnetic poles change position from north/south to south/north and vice versa. This effect makes the rotor turn in a more accurate way that in unipolar motors [18].

The difference between the types is that, although unipolar motors are simpler to program, they are less efficient and weaker than bipolar stepper motors [18]. Due to this fact, bipolar stepper motors were used for this project.

Stepper motors cannot be connected directly to the Arduino ONE board because they require drivers to work. The changes in current commented before are arranged by them. Their function is to control the movement of the motors following the orders from the Arduino UNO and supply energy to them.

The last choice was the number and type of sensors. They should allow the robot to detect the obstacles and avoid them. A quantity of three sensors was selected because it was necessary to detect front obstacles and sides ones, in order to make it able of decide when and where it should turn.

The type of sensors chosen were ultrasonic sensors. They constantly emit a sound signal that is rebounded by the obstacles in front of the sensors and captured by them again. This process allows them to calculate the distance to the obstacles by recording the time passed from the signal sending to the capture and knowing the speed of the sound. These sensors are effective for long distances but they lose accuracy for smaller distances than 3 centimeters [14].

3.3. Mechanics Basis

The mechanical part of this project is mainly focused on the structure of the robot and more especially in the design of the base where all the electronic and mechanical components would be put together. This base should be light and resistant in order to be able to support the weight of the robot but also allow a fluent movement. The material chosen was acrylic, which could be bought in sheets. This allows it to be cut easily, giving to it the appropriate shape and dimensions.

Another important aspect were the wheels. To allow more flexibility in the turns of the robot, a system of three wheels was planned. Two of them would be connected to the motors and the other would have a loose movement.

3.4. Testing Procedure

The testing procedure was divided in two parts. The first one was the checking process of all the electronic and mechanical components separately and the second one was the checking of the robot behavior.

Before starting to develop a complex logic that requires several electronic components, each component should be tested one by one in order to make sure that they are working properly and there is no anomaly on its performance. These tests consisted on simple trial circuits which will be shown in section 4.

The mechanical components were judged by their dimensions. They should fit together without compromising the stability and movement of the robot. Therefore, the design of the base was carefully done in order to respect the main dimensions and it was ensured that there was no component loose in the final assembly.

Once all the components were checked, an analysis of the behavior of the robot at different stages was carried out. Each stage corresponds to a different level of development of the code. To continue to the next testing stage, it was necessary to have verified that the previous stages were completed and satisfied. The testing stages were planned as the following:

1. Position recognition. The elements required were the Arduino UNO, the drivers and the motors.

   1.1 The longitudinal distance that the robot has travelled is controlled.
   1.2 The turning movement of the robot is controlled.
   1.3 The distance travelled along the horizontal direction or the x axis and the distance travelled along the vertical direction or the y axis are controlled.

2. Obstacle detection. The sensors were added to the circuit.

   2.1 The robot detects an obstacle and turns.
   2.2 The robot detects an obstacle, turns changing the direction and turns again when the obstacle is no longer in the original direction.

3. Complete performance. Different trial paths were created.

   3.1 First path: one front obstacle in x direction.
   3.2 Second path: one front obstacle with a blocked side in x direction.
   3.3 Third path: one front obstacle in y direction.
   3.4 Forth path: one front obstacle with a blocked side in y direction.
   3.5 Fifth path: two front obstacles, one in x direction and other in y direction.
   3.6 Sixth path: two front obstacles with a blocked

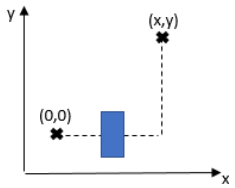


Figure 5. Front obstacle in x direction

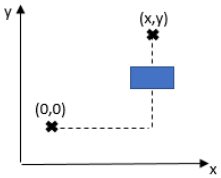


Figure 6. Front obstacle in y direction

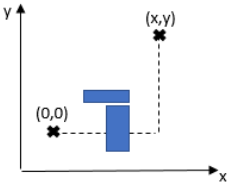


Figure 7. Front obstacle with one side blocked in x direction

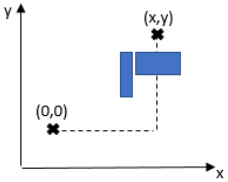


Figure 8. Front obstacle with one side blocked in y direction

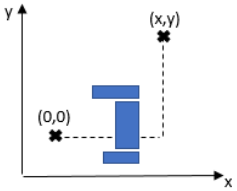


Figure 9. Front obstacle with both sides blocked in x direction

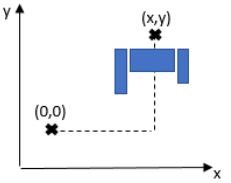


Figure 10. Front obstacle with both sides blocked in y direction

side, one in x direction and other in y direction.

3.7 Seventh path: one front obstacle with both sides blocked in x direction.

3.8 Eighth path: one front obstacle with both sides blocked in y direction.

3.9 Ninth path: two front obstacles with both sides blocked, one in x direction and other in y direction.

The types of obstacles used in those paths are shown in figures 5, 6, 7, 8, 9 and 10, where the axes "x" and "y" represent the direction of movement. The starting and target points refer to the position where the robot begins its movement.

If one stage could not be satisfied it was necessary to check the previous ones, starting by evaluating the performance of the components again. The testing procedure was applied twice, firstly without the mechanical structure and secondly with it.

3.5. Problems

The problems that this project had to face were multiple. They could be classified in two types: random and expected. The random were those that were not predicted beforehand. These problems were mainly caused by the failure of some electronic components that at the beginning of the project were working accurately and passed the initial checking but, during the development of the code, they stopped working properly and altered the whole perform of the circuit. These components were one driver and one ultrasonic sensor. Regarding this last component, during the performance of the sensors, anomalous distances were detected when the level of noise was higher than it is normally.

Random failures also affected the budget. The whole robot could have been built by only spending the budget set by the university, however these failures lead to an over-budget than was not planned from the beginning. Moreover, some of the components were delivered with a delay that affected the constant progress of the project.

Another unexpected problem was the loss of accuracy in distances when the speed of the wheels was high. Before adding the sensors, the assembled robot was tested only by telling the motors what distance they should move. Initially, they moved with a higher speed that the final version, leading to a misinterpretation of the distance that they actually travelled. This effect could be related to the friction that the motors should overcome for moving. Stepper motors lose strength when the speed increases. Therefore, a balance between accuracy and speed had to be done.

There was also a change in the working process of the sensors throughout the project. At the beginning, they were working constantly, reading the distances each time the code ran. This allowed the robot to act against unexpected obstacles. However, each cycle of code, the robot should read the distances and then move, resulting in a very slow movement. To avoid that, the sensors were only called to measure at the beginning of a forward movement and while avoiding obstacles.

The expected problems were those related to the development of the code and performance of the robot. As the purpose of this project is to create a complex code, care had to be taken from the beginning of its development. Despite this, as the code was getting bigger, some problems appeared. As an example, in one instance, the robot completed only the distance along the x axis and stopped without completing the distance along the y axis. It stopped at the target point in the previous trial, so it should have done it again. However, there was a difference between both paths, the first one had no obstacles and in the second there was one. The cause of the problem was a lack of a command that

allow the robot to turn more than once. Therefore, after turning for avoiding the obstacle, it could not turn to the y direction and complete the distance.

4. Designing Process

This section is the main part of the project. As the intention is to design and build an actual mobile robot, its development process has to be shown in a detailed way, combining the use of pictures with the description of it.

4.1. Motors Testing

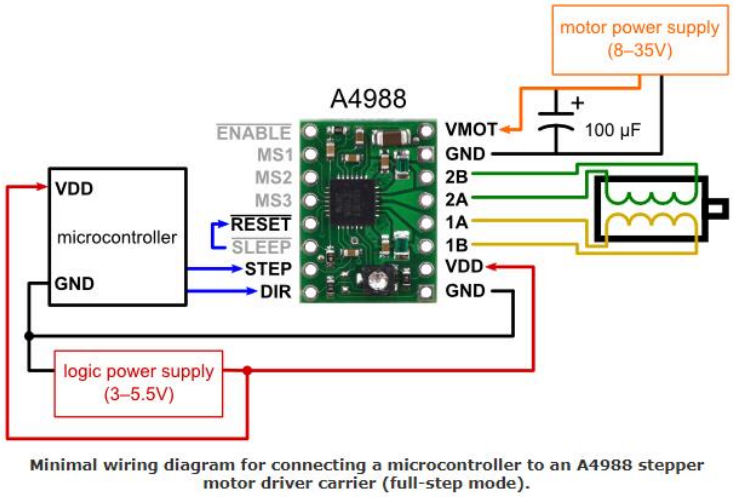
Two bipolar stepper motors and two drivers were bought. Their technical information and specifications are summarized in tables 1 and 2 and in figure 11.

Table 1: Nema 17 Stepper Motor Technical Information [19]

Nema 17 Stepper Motor	
Weight	230 g
Step angle	1.8 degrees
Body length	34 mm
D-cut length	15 mm
Number of leads	4
Leads length	300 mm
Shaft diameter	5 mm
Frame size	42 x42 mm
Front shaft length	20 mm
Phase resistance	30 ohms
Rated current/phase	0.4 A
Motor type	Bipolar Stepper
Inductance	37mH +/- 20%(1kHz)
Holding torque	26 Ncm(36.8.oz.in)
Connection	Black(A+), Green(A-), Red(B+), Blue(B-)

Table 2: A4988 Stepper Motor Driver Carrier Technical information [20]

A4988 Stepper Motor Driver Carrier	
Load supply Voltage Range (V_{BB})	8-35 V
Logic supply Voltage Range (V_{DD})	3-5.5 V
Overcurrent Protection Threshold (I_{OCPST})	2.1 A
Logic supply current	8 mA (f_{PWM}< 50kHz) or 5 mA (Outputs off)
Motor supply current	4 mA (f_{PWM}< 50kHz) or 2 mA (Outputs off)
Body diode forward voltage (V_F)	1.2V
Output On Resistance (R_{DSON})	430mΩ



Minimal wiring diagram for connecting a microcontroller to an A4988 stepper motor driver carrier (full-step mode).

Figure 11. Driver connection Specifications [20]

The driver must have two power inputs, one for running the logic (VDD) and another one for running the motor (VMOT). As shown above, the voltage admitted goes from 3 to 5.5V for the first input and from 8 to 35V for the second one. A capacitor of 100 µF has to be connected between VMOT and GND in order to protect the driver from voltage spikes higher than 35V [20].

The motor wires, Black(A+), Green(A-), Red(B+), Blue(B-), are connected to pins 2B, 2A, 1A and 1B respectively. The pins STEP and DIR control the movement of the motor. STEP makes the motor turn and DIR indicates the direction of the turn.

The connection of the motor, the driver and the Arduino UNO was done as shown in the figure 12, following the specifications of the motor and the driver.

At the beginning of the project, the external power came from a power supply which could vary the voltage and the current. It was connected to the electrical grid. Therefore, as the robot was developing the power source was changed to a 9V battery that could give it autonomy.

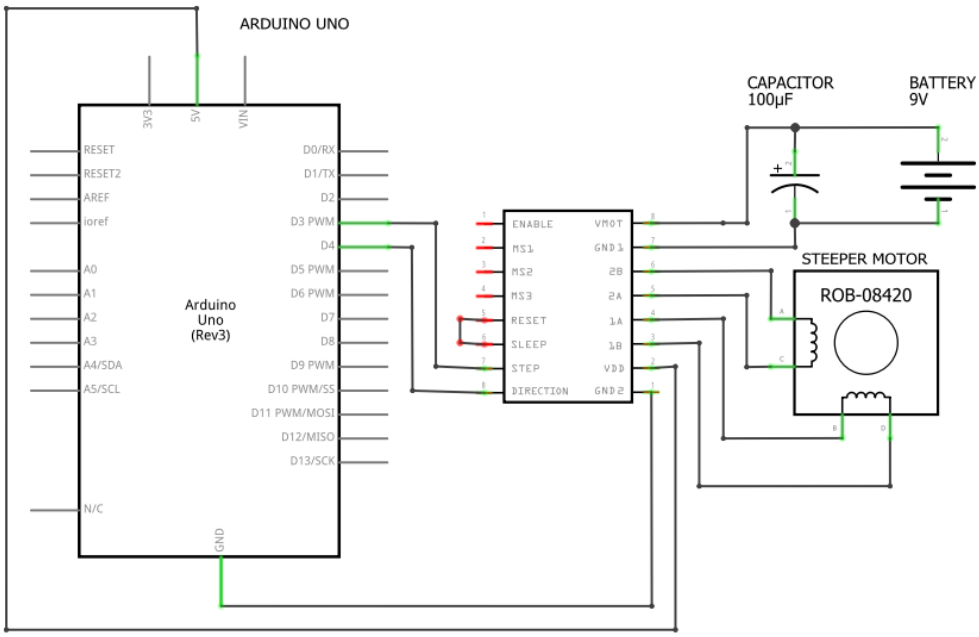


Figure 12. Stepper motor connection

The testing of the motors consisted of creating a program which could control how much they were moved. Each time that the program run the motors turned 1.8 degrees. Therefore, they needed 200 cycles to turn 360 degrees. Initially, the control of the movement was done by ordering it to turn the specific number of turns needed to cover a certain distance because the focus was on the turning of the axis. At the first moment, there were no wheels attached to the motor axes. To make the movement visible, two pieces of blue insulating tape were placed around the axes.

The first code created (figure 13) controlled one stepper motor by setting the number of turns. This number should be multiplied by 200 in order to obtain the number of times that the program should run. The understanding of this code is crucial to be able to follow the final one.

All the pins and variables were declared at the beginning of the code. In this case, the movement was controlled by pin 3 of the Arduino UNO and the direction was established by pin 4. The variable "d" indicates the number of turns that the motor axis should do, "r" is the number of times that the code has run and "cont" is used to count it.

In the "setup" function, both the stepPin and the dirPin were declared as output pins because they had to tell the motor what to do. In the "loop" function, the direction of turn is declared and the counting routine is done. Each time that the code ran, "cont" increased one unit. Using the "if" statement, the code could recognize whether the number of turns was satisfied or not. Inside it, the sequence to move the motor is written. It consists on a pulse that last 500 microseconds in high mode and another 500 microseconds in low mode. By changing the delay, the speed of the motors could be regulated.

```
// Define pins and variables
const int stepPin = 3;
const int dirPin = 4;
int cont = 0 ;
int d=10;
int r=0;

void setup() {
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
}

void loop() {
 // Indicate direction
 digitalWrite(dirPin,HIGH);
 r=cont;
 //Count
  if (r < d*200){
    // Start pulse
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
    // End pulse
    cont=1+cont;
  }
}
```

Figure 13. Stepper motor movement code [21].

It was done in both motors individually and they responded accurately. This code allowed to check that the two motors and the two drivers were in good condition and also to create the first idea to control the distance and the position of the robot.

4.2. Sensors testing

Before showing the testing process of the sensors, their technical features are shown below:

Table 3: Ultrasonic Module HC-SR04 Technical information [22]

Ultrasonic Module HC-SR04	
Voltage	5 V DC
Static Current	Less than 2mA
Induction Angle	Not more than 15 degrees
Detection distance	2cm – 510cm
High precision	3 mm
Wiring	VCC, trig, echo, GND

In section 3, the behavior of the sensors was explained. Now, the internal process to read the distance by sending and receiving a sound signal is going to be described. As shown in table 3, an ultrasonic sensor has four pins. Two of them are for the power supply (VCC and GND), which should be 5V, and the other two control the performance of the sensor. The pin "trig" activates the sensor and tells it when it has to send the sound signal. Therefore, this pin has to be defined as an output from the Arduino. A pulse is sent to "trig". This pulse is initially in a low state to change later to a high state that lasts 10 microseconds until it returns to a low state. After this pulse occurs, the sensor sends the sound signal. Since the signal is sent and until it is received by the sensor, the "echo" pin is in a high state. The code reads how much time "echo" has been in high state. This time corresponds to the time that the sound signal takes to get to the obstacle, bounce and return. Therefore, it is necessary to divide it by two for calculating the distance between the sensor and the obstacle. The distance is obtaining by using the mentioned time and the speed of sound. The calculation that has to be carried out is:

$$\text{Distance (cm)} = [\text{Duration } (\mu s)\ /2] \times \text{Sound speed}$$

$$\text{Distance (cm)} = [\text{Duration } (\mu s)\ /2] \times [343\ m/s] \times [10^{2}\ cm/m] \times [10^{-6}\ s/\mu s] = \text{Duration } (\mu s)\ /58.31$$

According to the specifications, the sensor has to be connected with the Arduino UNO as shown in the figure 14. VCC is connected to the 5V, GND of the sensor with GND of the Arduino, "trig" to one of the digital pins and "echo" to other of the digital pins. In order to test the performance of the sensors, they were integrated one by one to the circuit of figure 15, with both stepper motors.

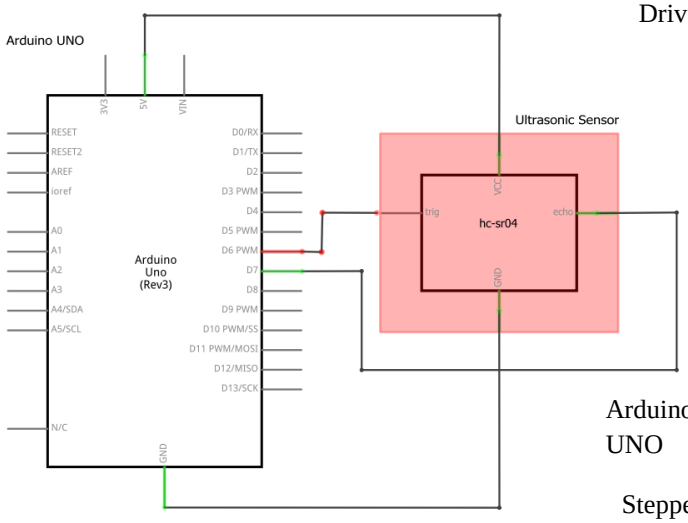


Figure 14. Ultrasonic Module HC-SR04 Connection

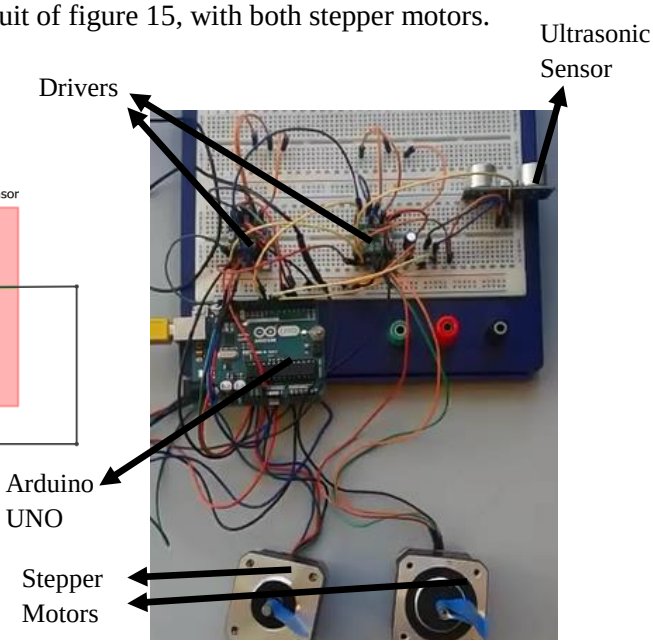


Figure 15. Steeper motors with one sensor

The program created (figure 16) to check that the sensors were working appropriately consisted of setting a distance that the wheels should do and stopping the movement when the sensor detects an obstacle at a distance of 6 cm or less. As usual, the pins and variables were declared at the beginning. Two pins for each motor and the pins 6 and 7 for the sensor. The variables used were the same ones that were from the last subsection. Moreover, the lines of code referring to the motors are the same ones as before, with the only difference that, in this code, two motors are being used. Therefore, each line is repeated twice, for motor 1 and motor 2.

Regarding the sensor, “trig” is set as an output and “echo” as an input in the “setup()” function. It is also stated “Serial.begin(9600)” which opens the serial port and sets the data rate to 9600 bits per second [23]. In the “loop()” function, the variable duration is declared. Then the pulse sent through “trig” is defined. It starts with a low state for 2 microseconds, changes to a high state for 10 microseconds and goes again to a low state for 2 microseconds. Using “pulseIn()” function, the program detects how many time has the “echo” pin been in a high state and transfers the data to the variable duration. After that, the variable distance is declared, associating to it a value of duration divided by 58, as explained before. To check what the sensor has read, the program writes the distance in the serial port with the command “Serial.printIn()”. The following code lines refer to the control of the movement. The new aspect that appears now is that another condition has been added. This condition is the minimum distance that could exist between the sensor and the obstacle. If any of the two conditions is not true, then the wheels stop.

As there was no actual movement, the obstacles were placed in front of the sensor at certain point of the execution of the code.

After checking that the sensors were working properly and that they could change the performance of the motors, another code was written. It was similar to this one but, instead of stopping, the motors changed direction when reaching the obstacle and discounted the distance travelled while moving backwards. For doing that, an “else” statement was written after the last “if” as shown in figure 17.

```
else{
  digitalWrite(dirPin1,LOW);
  digitalWrite(dirPin2,LOW);
  digitalWrite(stepPin1,HIGH);
  digitalWrite(stepPin2,HIGH);
  delayMicroseconds(200);
  digitalWrite(stepPin1,LOW);
  digitalWrite(stepPin2,LOW);
  delayMicroseconds(200);
  cont=cont-1;
}
```

Figure 17. Alternative ending code

```
const int stepPin1 = 3;
const int dirPin1 = 4;
const int stepPin2 = 9;
const int dirPin2= 10;
const int trig=6;
const int echo=7;
int cont = 0 ;
int d=20;
int r=0;

void setup() {
  pinMode(stepPin1,OUTPUT);
  pinMode(dirPin1,OUTPUT);
  pinMode(stepPin2,OUTPUT);
  pinMode(dirPin2,OUTPUT);
  Serial.begin(9600);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
}

void loop() {
  //Sensor measurement
  unsigned long duration;
  digitalWrite(trig,LOW);
  delayMicroseconds(2);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  delayMicroseconds(2);
  duration = pulseIn(echo,HIGH);
  float distance=duration/58;
  Serial.println(distance);
  delay(100);

  r=cont;
  digitalWrite(dirPin1,HIGH);
  digitalWrite(dirPin2,HIGH);
  //Move and count while the
  //condition is true
  if (r!=d*200 && distance > 6){
   digitalWrite(stepPin1,HIGH);
   digitalWrite(stepPin2,HIGH);
   delayMicroseconds(5500);
   digitalWrite(stepPin1,LOW);
   digitalWrite(stepPin2,LOW);
   delayMicroseconds(500);
   cont=1+cont;
  }
  }
```

Figure 16. One sensor and two motors code

4.3. Code development

Having all the components tested and the first steps to the final code developed, the final circuit was created. It combines the main connections shown in the previous two subsections. Figure 18 shows this final circuit.

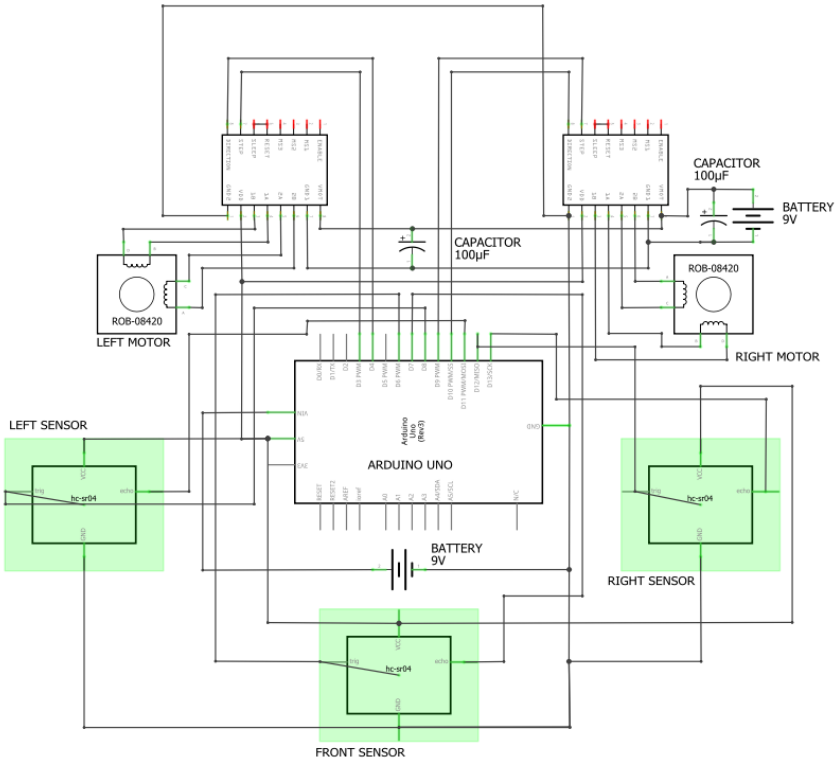


Figure 18. Final circuit.

This circuit was tested firstly only with the electronic components shown in the figure above and secondly with the whole assembly done. The first testing process was carried out with a power supply and the energy supplied by the Arduino USB connector from the computer and the second was done using two lithium-ion rechargeable batteries of 9V and 800mAh [24].

The starting point of the final code development process was to establish what the robot should do. The robot should reach a point given by setting its coordinates in millimeters on the code. To get to that point, the robot had to complete the distance in the x axis direction and then the distance in the y axis direction. It would complete the path following that order because it is easier to program the avoidance of obstacles that could appear than if the movement were diagonal. Moreover, the code for obstacle recognition would be very similar in both directions, with slight differences mainly in the counters of distance. Having into account that criteria, the next step was to plan the performance of the robot to complete a longitudinal distance. With that aim, the flow diagram of figure 19 was created.

The robot could avoid three types of obstacles: a front obstacle with both sides unblocked, a front obstacle with one of the side blocked and a front obstacle with both sides blocked. The avoidance process of these obstacles is what is shown in the small flow diagram. Firstly, the robot asks the front sensor whether there is a front obstacle or not. If the answer is no, the robot moves forward until it finds an obstacle or it reaches its final position. If the answer is yes, it asks the left sensor whether there is an obstacle at the left side or not. With a negative answer, it turns left and move forward until reaching the end of the obstacle, when it turns right and starts the process again. With a positive

answer, the robot asks the right sensor if there is an obstacle on the right. If not, it acts as in the previous case but in the opposite direction. If both sides are blocked, the robot move backwards until any of the sides is unblocked. Then it acts as with a normal obstacle.

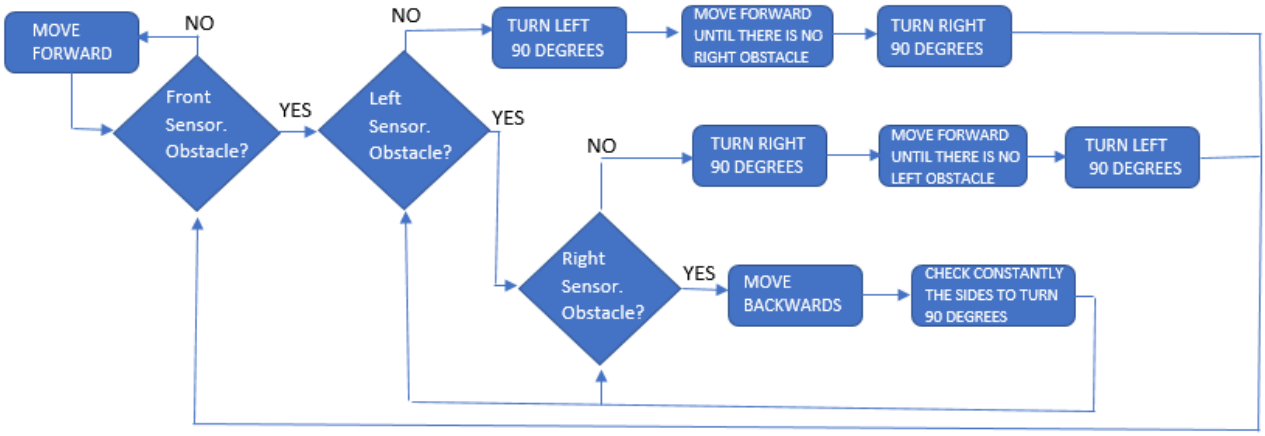


Figure 19. Small Flow Diagram.

The diagram gives an idea of how the program acts. However, its code is more complex than three conditional statements, so it will be explained step by step. It is also important to mention that some more code trials were done before starting the main code. Those trials tested the coordination between the three sensors and the turn movement of the robot. Their advances on the code were used for the final code so they will be explained there.

Before starting the explanation, it is necessary to say that the whole code is written in Appendix1. In this section, only the main variables and functions will be shown. The code will be explained as a way to achieve the desirable performance of the robot, however there could exist other ways to accomplish the same functionality.

Like the previous codes, the final one begins with the declaration of pins and variables (figure 20). The motors are associated with two pins each, the "stepPin" and the "dirPin". The sensors are also assigned two pins each, "trig" and "echo". Then, the target point is defined in millimeters by giving its coordinates.

```
// Declaration of variables
// Motor 1 (left)
    const int stepPin1 = 3;
    const int dirPin1 = 4;
// Motor 2 (right)
    const int stepPin2 = 9;
    const int dirPin2= 10;
// Sensor 1 (front)
    const int trig1=6;
    const int echo1=7;
// Sensor 2 (right)
    const int trig2=12;
    const int echo2=13;
// Sensor 3 (left)
    const int trig3=8;
    const int echo3=11;
//Target Point
    float x=1000;
    float y=700;
//Robot Dimensions(mm)
    float diameter=63.49;
    float wheels=3.1416*diameter;
    float turn90=(125+15)*2*3.1416/4;
    float turn180=turn90*2;
    float varxd=150;
//Wheel turns
    float dx=x/wheels;
    float dy=y/wheels;
    float varx=varxd/wheels;
    float d90=turn90/wheels;
    float d180=turn180/wheels;
    int turn=0;
    float var=0;
//Direction
    int dir=0;
```

Figure 20. Final Code Part1

The next part is the robot dimensions. They are mentioned in the subsection 4.4., called "Mechanical Structure". The dimensions are required to measure the movement of the robot. In the codes before, only the turns were specified. Now, the turns required to complete a distance are calculated with the length of the wheels. This fact explains why there is a section in the code called "wheel turns".

In “robot dimensions”, the diameter of the wheels is specified in “diameter”, the length of the wheels is calculated in “wheels”, the distance required to turn 90 degrees is calculated in “turn90”, the distance required to turn 180 degrees is calculated in “turn180” and the distance that the robot should move when the lateral sensors do not detect a side obstacle, in order to avoid collisions, in “varxd”. These dimensions are expressed in millimeters.

The distance for “turn90” is determined using the radius of the base of the robot plus half of the width of the wheels, which excel the perimeter of the base. Having the length of the corresponding circumference, it has to be divided by four.

The distance “vardx” is the distance between the end of the sensors, taking as a reference the forward movement, and the back of the robot plus a margin of error.

```
//Extras
    int cont = 0 ;
    int cont1=0;
    int r=0;
    int N=0;
    int conty=0;
    int contyn=0;
    int contx=0;
    int contxn=0;
    int negative=1;
    int backwards=0;
    int obs=0;
    int P=0;
    int help=0;
    float distance1=0.0;
    float distance2=0.0;
    float distance3=0.0;
    int free_way=0;
    float path=7*10*200/wheels;
    int ayu=0;
    float dist=0;
 // Sign "y coordinate"
 //(0 for positive and 1 for negative)
    int coordinate=0;
```

Figure 21. Final Code Part2

In “wheel turns”, all the distances are divided by the length of the wheels or “wheels” in order to obtain the number of turns equivalents to those distances. Thereby, “dx” is obtained for “x”, “dy” for”y”, “varx” for “varxd”, “d90” for “turn90” and “d180” for “turn180”. Moreover, two more variables appear in this part of the code: “turn”, which indicates if the robot is avoiding an obstacle (1) or not (0), and “var”, that is used to count the distance for complete “varx” when avoiding a lateral obstacle.

In the section “extras” and “sign y coordinate” (figure 21), more elements created for helping the robot are declared. They are:

- **“cont”**: it is used to count the distance travelled along the x axis or in the x direction.
- **“cont1”**: it is used to count the distance travelled along the y axis or in the y direction.
- **“r”**; it is used to update the distance travelled each time the code runs.
- **“N”**: it is used to count the distance travelled during the turning movement.
- **“conty”**: it is used to count the distance travelled along the positive y direction during a turning sequence.
- **“contyn”**: it is used to count the distance travelled along the negative y direction during a turning sequence.
- **“contx”**: it is used to count the distance travelled along the positive x direction during a turning sequence.
- **“contxn”**: it is used to count the distance travelled along the negative x direction during a turning sequence.
- **“negative”**: it is used to know whether the turning sequence is being doing to the left (0) or to the right (1)
- **“backwards”**: it is used to know whether the robot is doing a backwards sequence (1) or not (0).
- **“obs”**: it is used to know whether a front obstacle has been detected (1) or not (0).
- **“P”**: it is used to count the distance travelled in the x direction until reaching the final point, when both distances, in x axis and in y axis, are completed.
- **“help”**: it is used to ensure that the robot is turning left (0) or right (1).

- **"free_way"**: it is used to tell the robot whether it can move forward without using the front sensor (1) or not (0)
- **"ayu"**: it tells the robot whether it has reached the lateral obstacle (0) or not (1). It is used after the backwards sequence, when the robot has turn 90 degrees.
- **"dist"**: it tells the robot the distance that has to add or discount to reach the distance in the y direction.
- **"coordinate"**: it tells the robot whether it has to turn to the positive y axis (0) or to the negative (1).
- **"path"**: the minimum distance between the obstacle and the front sensor is defined here and converted into number of cycles of the program.

It is necessary to understand these variables in order to follow the code fluently. Once they are declared, the code executes the "setup()" function. It is shown in figure 22. There is nothing new that have not been explained before.

```
void setup() {
  //Motor 1
  pinMode(stepPin1,OUTPUT);
  pinMode(dirPin1,OUTPUT);
  //Motor 2
  pinMode(stepPin2,OUTPUT);
  pinMode(dirPin2,OUTPUT);
  //Sensor 1
  Serial.begin(9600);
  pinMode(trig1,OUTPUT);
  pinMode(echo1,INPUT);
  //Sensor 2
  pinMode(trig2,OUTPUT);
  pinMode(echo2,INPUT);
  //Sensor 3
  pinMode(trig3,OUTPUT);
  pinMode(echo3,INPUT);
}
```

Figure 22. Final Code Part3

Following "setup()", the functions created for the code are written. These functions gather lines of code that are used constantly in the program. Therefore, these functions were created, allowing space saving and better organization of the code. They are the following:

- **"forward()"**: it is used to move the robot forwards. Due to the disposition of the motors in the robot, the motor 1 needs a low state to move forward and the motor 2 needs a high state. Another important aspect of this code, is that the speed is controlled. The second delay is 10000 microseconds in order to reduce the speed. The code is shown in figure 23.

```
void forward(){
    digitalWrite(dirPin1,LOW);
    digitalWrite(dirPin2,HIGH);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
}
```

Figure 23. Final Code Part4 [forward()].

- **"toleft()", "toright()"**: they are used to turn the robot to the left or to the right by changing the direction of one of the wheels. For moving to the left, the motor1 direction is changed and, for going to the right, the motor 2 direction is changed. The codes are shown in figures 24 and 25 respectively.

```
void toleft(){
    digitalWrite(dirPin1,HIGH);
    digitalWrite(dirPin2,HIGH);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
}
```

Figure 24. Final Code Part5 [toleft()].

- **"left_turning()", "right_turning()"**: the difference with the previous functions is that these ones make the whole turning sequence. These codes count the full turn and then activate the turning mode (turn=1) for avoiding the obstacle. Moreover, if the turning mode is towards the left, "negative" is equal to 0 and, if the turning mode is towards the right, "negative" is equal to 1. The obstacle mode is deactivated (obs=0) and variables that has been used before for counting (N) or

```
void toright(){
    digitalWrite(dirPin1,LOW);
    digitalWrite(dirPin2,LOW);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
}
```

Figure 25. Final Code Part6 [toright()].

as a reference (distance, help) are initialized to 0. The codes are shown in figures 26 and 27 respectively.

- **"move_backwards()"**: it is used to move the robot backwards. This code changes the direction of both wheels. The rest of the function is equal to the "forward()" function. It is shown in figure 28.

- **"Sensor1()", "Sensor2()", "Sensor3()"**: these three functions control the performance of the three sensors. They have the same code but referring to a different sensor. Sensor 1 is on the front, Sensor2 is on the right and Sensor 3 is on the left. The code used is the one explained before in subsection 4.2 for the measuring process of the sensors. The functions can be seen in figures 29, 30 and 31 respectively.

```
void left_turning(){
    if (N<200*d90){
    digitalWrite(dirPin1,HIGH);
    digitalWrite(dirPin2,HIGH);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
    N=1+N;
   }
  else{
    turn=1;
    negative=0;
    N=0;
    obs=0;
    distance2=0;
  }
}
```

Figure 26. Final Code Part7 [left_turning()].

```
void right_turning(){
    if (N<200*d90){
    digitalWrite(dirPin1,LOW);
    digitalWrite(dirPin2,LOW);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
    N=1+N;
   }
  else{
    turn=1;
    negative=1;
    N=0;
    obs=0;
    help=0;
    distance3=0;
  }
}
```

Figure 27. Final Code Part8 [right_turning()].

```
void move_backwards(){
    digitalWrite(dirPin1,HIGH);
    digitalWrite(dirPin2,LOW);
    digitalWrite(stepPin1,HIGH);
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin1,LOW);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(10000);
}
```

Figure 28. Final Code Part9 [move_backwards()].

```
void Sensor1(){
  unsigned long duration1;
  digitalWrite(trig1,LOW);
  delayMicroseconds(2);
  digitalWrite(trig1,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig1,LOW);
  delayMicroseconds(2);
  duration1 = pulseIn(echo1,HIGH);
  distance1=duration1/58;
  Serial.println(distance1);
  delay(25);
}
```

Figure 29. Final Code Part10 [Sensor1()].

```
void Sensor2(){
  unsigned long duration2;
  digitalWrite(trig2,LOW);
  delayMicroseconds(2);
  digitalWrite(trig2,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig2,LOW);
  delayMicroseconds(2);
  duration2 = pulseIn(echo2,HIGH);
  distance2=duration2/58;
  Serial.println(distance2);
  delay(25);
}
```

Figure 30. Final Code Part11 [Sensor2()].

```
void Sensor3(){
 unsigned long duration3;
 digitalWrite(trig3,LOW);
 delayMicroseconds(2);
 digitalWrite(trig3,HIGH);
 delayMicroseconds(10);
 digitalWrite(trig3,LOW);
 delayMicroseconds(2);
 duration3 = pulseIn(echo3,HIGH);
 distance3=duration3/58;
 Serial.println(distance3);
 delay(25);
}
```

Figure 31. Final Code Part12 [Sensor1()].

With all the variables, pins and functions explained, the big flow diagram that represents the whole logic of the robot can be shown. It appears in figure 32 and it shows how the desirable performance of the robot is achieved, using the variables mentioned before. Their default values are those shown in figures 20 and 21.

This diagram was used to develop the "loop()" function of the code, where the code lines that are repeated constantly are written. To explain how it works, the structure of the code will be followed.

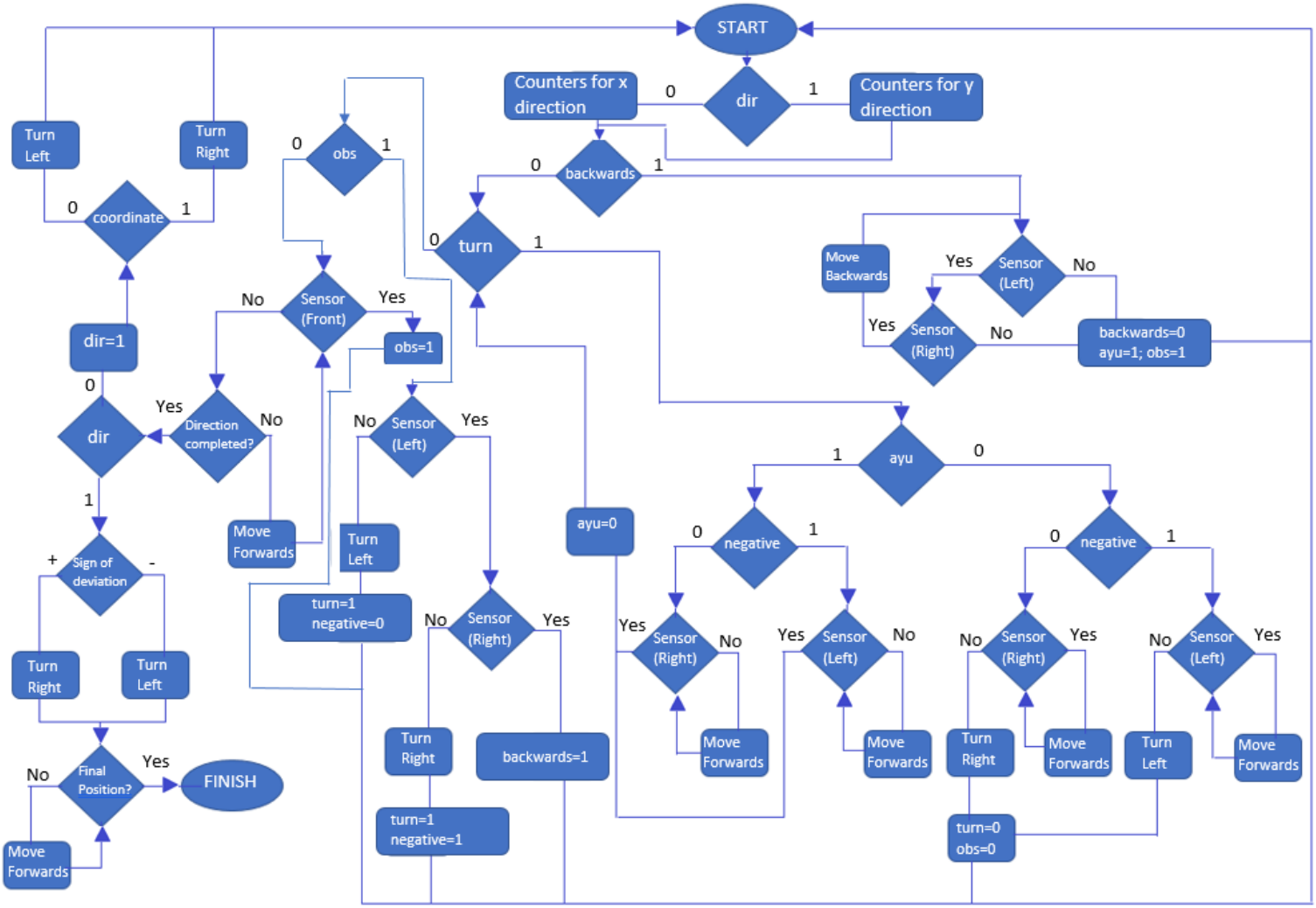


Figure 32. Big Flow Diagram.

The logic is divided in two parts: x axis direction and y axis direction. The code is practically the same in both parts so it will be explained in detail for one of them and then the differences from the other part will be clarified.

Firstly, the robot detects the direction reading the value of "dir". Having a 0 value, the robot will complete the x axis distance. After reading the direction, the code says whether the robot is doing a backward or a turning mode. With both variables, "backwards" and "turn", deactivated (0 value), the robot will move forward. Before starting the movement, it activates the front sensor in order to read the distance that can be travelled without crashing with an obstacle. The "path" initial value is discounted to the distance in order to have a minimum distance between the robot and the obstacle. In this case, the minimum distance is 7 centimeters. Once the robot has travelled forward until reaching

the minimum distance from the obstacle, the front sensor is activated again in order to check if there is actually an obstacle in front of the robot. If the answer is yes, "obs" value is changed to 1 and, with a negative answer, the process is repeated again from the beginning. When "obs" is 1, the robot will ask where the obstacle is by activating the lateral sensors. If one of the sides is unlocked or the distance read by the sensor is bigger than 30 centimeters, the robot will turn to that side activating the turning mode (turn=1). If both sides are blocked or the distances are less than 30 centimeters, the robot will move backwards activating the backwards mode (backwards=1). It is necessary to mention that the value of 30 centimeters can be modified depending on the precision that the robot should have. This first part of the code is identified in the appendices as Part13.

The next part of the code (left part of the flow diagram) tells the robot what to do when the distance along the x axis has been completed, which means that r=dx*200. In that case, it will turn 90 degrees to the left, if "coordinate" is equal to 0, or to the right, if "coordinate" is equal to 1. When the turning movement has finished, "dir" changes to 1 and "dist" is calculated with the difference between the distances along the y axis that it has done while avoiding obstacles, as it will be seen in the following parts of the code. This part is identified in the appendices as Part14.

The last bracket of Part14 closes the "if" statement for turn=0, activating the turning mode (after turn=1 in the flow diagram). The turning mode recognizes whether the turn has been to the left (negative=0) or to the right (negative=1). Previously, the robot asks if "ayu" is 0 or 1. This is used after the backwards mode, when the robot has to know when the obstacle on the lateral side appears, so it can avoid it. When "ayu" is 1, the robot activates the corresponding lateral sensor and moves forwards until it detects the obstacle. Then "ayu" is deactivated (0) and the normal turning mode continues. In this mode, the robot moves forwards, parallel to the obstacle until the corresponding lateral sensor does not detect the obstacle. After that, the robot continues moving towards until it completely passes the obstacle, travelling the distance specified in "varxd". Following this action, the robot can turn 90 degrees again to return to the x axis direction. While it was moving along the y axis, the distance was counted in "conty" after a left turning and in "contyn" after a right turning. These two variables are used to calculate "dist", which has been mentioned before. The code for this part is identified in the appendices as Part15 (Note that the recognition of the value of "negative" is done before the recognition of "ayu" on the code. The result is the same).

The last bracket of Part15 ends the conditional statement "backwards==0" and activates the backwards mode (after backwards=1 in the flow diagram). In this mode, the robot moves backwards until any of the side sensors detects that there is no longer an obstacle. When this happens, the robot continues moving backwards until it completely passes the obstacle, travelling the distance specified in "varxd" plus a margin of error of seven millimeters. After that, the backwards mode is deactivated (backwards=0) and the cycle starts again from the beginning, having "ayu" activated (1). During the backwards movement, the distance travelled along the x axis is discounted. The backwards mode code is shown in Part16 of appendix 1.

When the backwards mode finishes, the code for the movement along the x axis direction is closed, which means that the condition "dir==0" has ended. Using an "else {}", the program introduces the movement along the y axis direction (dir=1). As said before, the code for that movement is the same as the one described but with some differences in the variables used. It uses "cont1" instead of "cont"; "contx" instead of "conty" and "contxn" instead of "contyn". Another difference is the reaction of the robot when the distance along the y axis direction has been completed. It compares "contx" and "contxn" in order to know in which direction it should turn to reach the final point (In the

flow diagram it is asked: “signal of deviation?”). If “contx” is bigger (+), the robot turns right and, if “contxn” is bigger (-), it turns left. After turning to the appropriate direction, the robot moves towards in the x axis direction travelling a distance equivalent to the difference between “contx” and “contxn”, in absolute value. After that, the final point is reached.

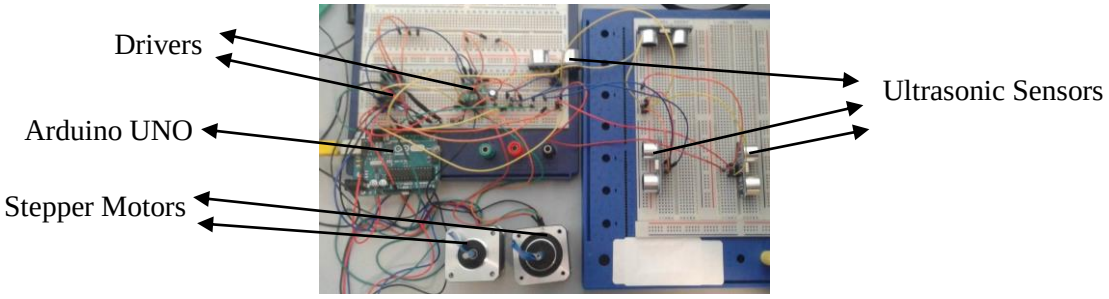


Figure 33. Circuit created in the electronics lab.

The developed logic allowed the robot to have the level of autonomy specified at the beginning of the project. It controls its position and it is able to avoid obstacles, with the restrictions that will be commented on section 5. In figure 33, the actual circuit used to develop the code is shown. It is connected as indicated in figure 18.

4.4. Mechanical structure

The mechanical structure consists of three parts: the circular base, the mounting brackets for the stepper motors and the wheels. The circular base is the connection of all the mechanical parts and it supports the electronic components. An acrylic sheet of 500mm x 400mm x 3mm [25] was bought to create the base. Before cutting it, its shape was drawn using SolidWorks. Their dimensions were adjusted to allow all the parts of the robot to fit in it. They are shown on figure 2.1 of Appendix 2. SolidWorks was also used to draw the internal structure of the wheels, which makes possible the connection between the stepper motors and the wheels. Their dimensions are shown on figure 2.2 of Appendix 2. Both parts, the base and the internal connection of the wheels, were cut using a laser cutter. The machine loaded a “dxf” file of each component and cut the acrylic sheet with the dimensions ordered.

The cases of the motors were bought with them. Therefore, they fitted perfectly together. The dimensions obtained from the manufacturer appears on figure 2.3 of Appendix 2.

The two types of wheels selected were: the ones connected to the motors and the one that follows the movement. This last one can rotate freely around all the axis (x, y and z) and is suitable for making the robot stable while allowing it to move. A picture of this wheel can be seen in figure 34.

The other two wheels were taken from an iRobot Roomba Discovery 4210 [26]. That robot was disassembled at the beginning of the project to see if any of its components were useful. The internal connections of the wheels were removed, leaving the tires. As they were originally made for a mobile robot, they offered a good level of friction and resistance. Therefore, they were appropriate for the project. The only problem that they presented was the connection to the motors axes, which was solved by creating the piece mentioned before. That piece is shown in figure 35.

In figure 36, the whole assembly can be seen. It gathers all the elements mentioned and has already the motors installed.

Figure 34. Free-movement wheel.

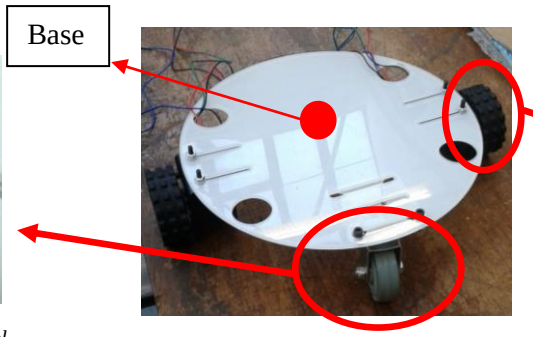


Figure 36. Robot assembly.

Figure 35. Wheels Connection

4.5. Path Testing

The path testing was carried out by two different ways. The first way consisted of testing the behaviour of the robot by the performance of its electronic components, shown in figure 33. This was a static testing process, which focused on the turning movement of the axes of the motors. Its direction of movement, its speed and the number of turns was observed in order to check that the performance was correct and that the circuit followed the instructions given by the code. The obstacles were simulated by approaching the hands to the sensors.

Once the mobile robot was built (figure 37), its performance was checked on a flat surface where different obstacles were placed. These obstacles were mainly rectangular boxes, with appropriate dimensions and flat faces that allowed the robot to detect them. More detail about the restrictions concerning the obstacles will be mentioned in section 5.

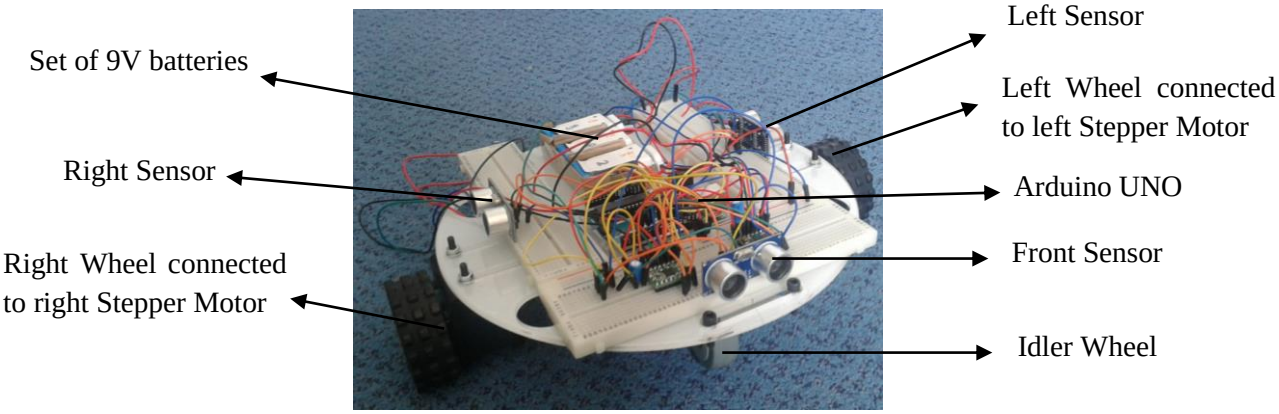


Figure 37. Mobile Robot.

The testing procedure followed in both cases was the one described in section 3.4. The two first stages, position recognition and obstacle detection, were evaluated by using a free obstacle path and a path with one front obstacle. Then, the different paths mentioned in the third stage were simulated, when testing the electronic components, and created, when testing the actual performance. Some of the paths created were a combination of those mentioned.

When testing the performance, the coordinates of the target point were declared in the program and loaded on the Arduino UNO. The robot should end its movement always in that point, given in millimetres and referred to the start position of the robot, which corresponded to the position (0,0). In order to ensure that the distances travelled were correct, three marks were placed on the floor: one in the starting position, another one in the position were the movement along the x axis should end and

another one in the target point. Some pictures taken during the testing process are shown in figures 38, 39, 40, 41, 42, 43 and 44.

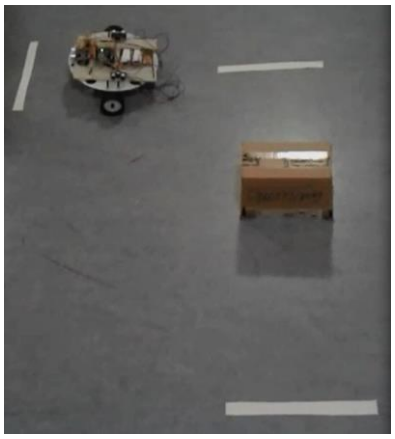


Figure 38. Front obstacle in x axis direction.

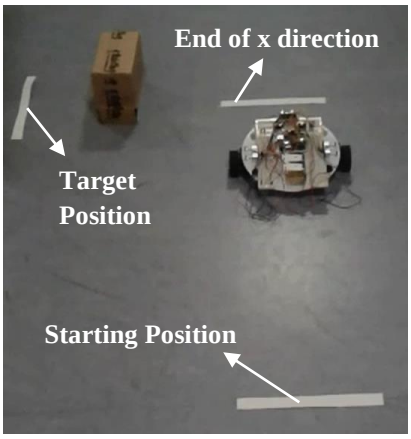


Figure 39. Front obstacle in y axis direction.

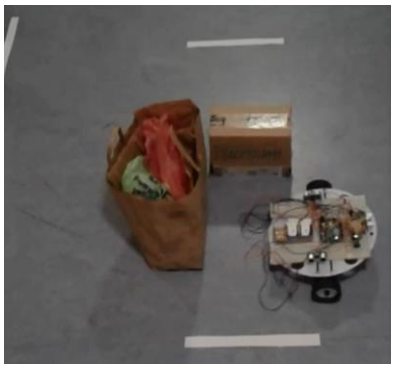


Figure 40. Obstacle with one side blocked in x axis direction.

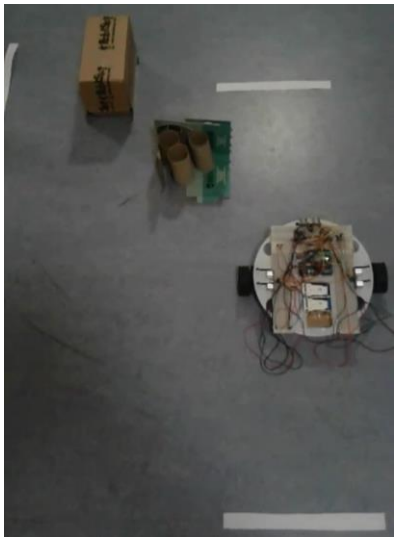


Figure 41. Obstacle with one side blocked in y axis direction.

Figure 42. Obstacle with both sides blocked in x axis direction.

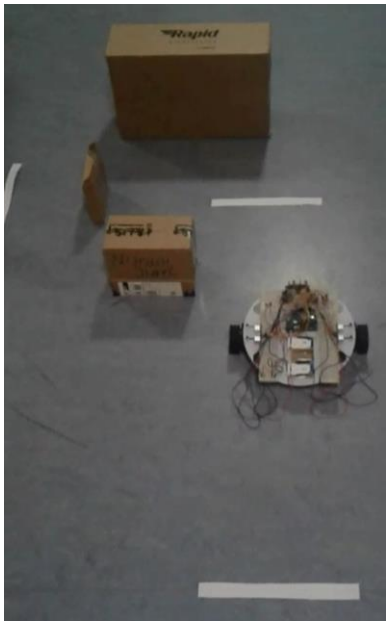


Figure 43. Obstacle with both sides blocked in y axis direction.

Figure 44. One obstacle with one side blocked in x axis direction and one front obstacle in y axis direction.

All these pictures are taken from different videos that were recorded to check the full performance of the mobile robot. Figures 38, 39, 40, 41, 42 and 43 show all the types of obstacles that the robot can recognise and avoid. It is necessary to mention that some of the obstacles did not have flat faces in order to observe how the robot reacted to them.

In figure 44, a combination of the obstacles is shown. All the obstacles were combined differently to test the performance of the robot.

5. Descriptions of final constructed product

The results will be commented on this section. Firstly, a description of the mobile robot will be carried out, giving details of its performance. Then, its working conditions and the restrictions for achieving proper behaviour will be included. Both these sections will lead the reader to the final conclusion and discussion of the project, shown in section 6.

5.1. Features and performance

The mobile robot that has been developed throughout this project is shown in figure 37. In this section, its features will be commented one by one.

The robot is composed of two motorized wheels, set on the middle axis of the base, and one idler wheel, attached to the front of the robot. This structure allows the robot to do static turns, which are essential to control the position of the robot.

Three sensors let the robot perceive its environment: one at the front and the other two at both sides, specifically at the middle axis of the base. The sensors have a long-distance precision and read the distance accurately. This allows to reduce the minimum distance required between the front sensor and an obstacle because the robot will stop exactly at that distance and it is not necessary to leave a great margin of error.

The stepper motors give also precision to the movement. The fact that each cycle of the code they move only 1.8 degrees allows counting the distance travelled with a very small error that could be negligible. A wheel turn of 1.8 degrees implies a distance of approximately 1 mm, which is the maximum error that the distance could have. The error could vary between 0 and 1 millimetres because the counting is done with integer numbers and the distance specified is a rational number. Therefore, when the distances are compared, the program can interpret one more cycle to reach the distance when actually less than a 1.8 degrees turn is required.

Its case allows it to carry three batteries of 9V: two for suppling energy and a spare one. Nine volts is enough power to move the motors, which can be supplied on a range between 8 and 35 V, and to activate the Arduino UNO, which needs 5V and has a controller. When the batteries are running out of power, the robot cannot perform properly and acts in an unexpected way.

Regarding the connections, three protoboards are used to facilitate the wires distribution and set all the elements in a correct position.

After knowing all the features of the robot, its performance can be commented on. It was observed that the robot completed all the distances previously defined on the code. Starting from the white line shown previously (figure 39), it reached the axis indicated by the next white line, independently of the type of obstacle that was set on the x axis, and turned statically, changing from the x axis direction to the y axis direction. Then, it always finished its movement on the last white line, which coincided with the target point. The robot ended on the target point avoiding any type of obstacle from those shown previously.

In the case of a front obstacle, the robot turned left and continued moving parallel to the face of obstacle, with a lower speed than when it was moving towards the obstacle. This effect is due to the fact that the sensors are deactivated before reaching the obstacle in order to increase the speed of the robot. If they are constantly reading the distance, the robot has to wait to receive their information before moving the wheels one step, reducing considerably the speed of movement. Therefore, when

the robot was trying to avoid the obstacle, the sensors were activated to know if it had been passed or not, sending information constantly. When the obstacle had been passed, the sensors were deactivated again and an increase on the speed was appreciated. After that, the robot passed completely the obstacle and then turned right.

In the case of an obstacle with a blocked side, if the blocked side was the left one, the robot turned statically right, or, if the blocked side was the right one, it turned statically left. After turning, the robot acted as done with the previous type of obstacle.

In the case of an obstacle with both sides blocked, the robot moved backwards with a lower speed because the sensors were activated. When one of the sides was recognised as an unblocked side, the robot turned to that side and acted as before, with the only difference that it had to also to detect the initial part of the lateral obstacle in order to not turn before having avoided it. This mistake could have happened because the robot does not detect any close obstacle after going backwards and turning. When it turns, the corresponding lateral sensor detects the distance until the front obstacle, which is not close after having gone backwards.

In all the cases, when the robot started moving towards a direction or returned to the direction of movement after avoiding an obstacle, it activated the front sensor only once in order to detect how long it could move with the sensors deactivated and without crashing. Therefore, it works when the obstacles are fixed. If a new obstacle is placed in front of the robot while it is moving towards, it would crash because the sensors are not working.

Another observation of its performance is that the robot successfully recognised the variance of the original distance to the final point, which is a consequence of avoiding obstacles. If the obstacle was in the x axis direction, the robot added or discounted the distance along the y axis. If the obstacle was in the y axis direction, the robot turned to the corresponding side to compensate the distance along the x axis direction that was modified, just after completing the distance along the y axis direction.

5.2. Behaviour restrictions.

The main restriction of the mobile robot is the typology of obstacles. The obstacles have to be fixed in a position since the beginning of the movement as said in the previous subsection. Moreover, their shape must be rectangular and they must have flat faces. The sensors might have problems when reading distances if the surface where the sound signal bounces is not flat. The reflection of the signal could be affected and sent in a random direction. If the shape of the obstacles were rounded or irregular, the reading of the sensors would also vary and result in an unexpected performance of the robot, as it is programmed to detect straight corners. It is also important the fact that the dimensions of the robot should fit with the obstacles and its distribution through the space. The robot cannot avoid an obstacle which is under its range of detection. For example, an obstacle with both sides blocked should be enough wide to allow the robot to move into it or a front obstacle should be high enough to reflect the signal. These restrictions in dimensions are closely related with the number of sensors and its range of view. As a consequence of all these conditions, the suitable obstacles are those three types used during the testing process, with flat faces and dimensions bigger than the ones of the robot.

In terms of speed, the robot seemed to be more accurate with low and medium speeds rather than high ones. When the speed was too high, the robot did not complete the distances ordered by the program, it stopped the movement before finishing the distances with a great error up to twenty fifth percent when the distance was one meter. Therefore, the speed was reduced until the position could be controlled accurately, with the precision mentioned before.

Another restriction was in the amount of time that the sensors are used. Initially, the robot used all the sensors continuously in order to constantly check the distance and detect obstacles that were not placed at the beginning of the movement. However, this resulted in a very slow movement as explained in subsection 3.5. To fix it, the use of the sensors was reduced to certain moments, obtaining a more reasonable speed.

The last restriction is on the displacement. The robot could act in an unexpected way when a certain arrangement of obstacles meant that the robot exceeds the limits of an imaginary rectangle with the vertices (0,0) and the target point. It could also act bizarrely when the obstacles are placed in the sides of that rectangle.

6. Discussion and conclusions

6.1. Evaluation of the result and future improvements

The project has ended with a satisfactory result. The robot designed and built has satisfied the specifications established at the beginning of the project. Its position was controlled and it was able to avoid rectangular obstacles by recognizing its environment. As the robot was able to reach a specified point with a high grade of precision, its logic could be used for logistic activities. Its independence from a guidance track and its autonomy under a certain type of conditions allow recognizing it as an autonomous intelligent vehicle (AIV).

The mobile robot created could be a first step to create something bigger. Its range of detection could be improved by adding more sensors, which could be a mix of different types in order to avoid the limitations of the ultrasonic ones. Consequently, the typology of obstacles could be extended, which would mean a development of the code in order to allow the robot to distinguish them. The code presented in this project could be used as a base where new functions could be introduced, improving the performance of the robot. Moreover, a mechanical structure could be added or connected to the robot in order to be guided by it and make possible the carrying process of weights. Therefore, the technology developed could be applied in actual industrial storages. Its use for automating processes could be offered as a way for gaining efficiency and improving production, affecting the economy of the business positively. Logistic activities could be done faster and more precisely.

6.2. Project discussion

Apart from reaching a satisfactory result, this project has helped to understand in depth the different stages that should be done in a designing process. It has highlighted how important it is having a good plan and organization since the beginning of the project in order to complete it on time and properly. Creating a product means thinking about its scope and set the specifications that it should satisfied before starting the designing activities. These activities are the main part of the project and the longest one. A continuous cycle of designing or, in this case, programming and then testing is vital to reach the final result. It allows finding the problems and solving them before presenting the product.

The project also means a self-challenge. It has given the chance to apply knowledge acquired through years in something useful and creative. The project is a way to deepen two areas of interest, electronics and programing. It could also be seen as a motivation to learn and to work further in these areas in the future.

7. Project management, consideration of sustainability and health and safety

7.1. Gantt Chart and management

The project management was divided in two parts: management of activities and management of components orders. The activities were planned at the beginning of the project and the track to follow was developed in a Gantt-Chart. This Gantt-Chart has been updated along the project, in terms of dates and activities. It was affected mainly by the delays on the deliveries of the components. The last Gantt-Chart and the one that better reflects the process followed is shown in figure 45. It shows the stages of the project and the activities done in each step, with less intensity colours.

The components orders were planned to be made a week and a half before using them. However, some arrived almost a month after the order. Moreover, one of the criteria for choosing the products was its price. Nevertheless, the amount of money that had to be spent was 40% more of the budget established by the University. In table 4, the prices and dates of order of each product ordered are shown. The rest of the components used were facilitated by the University or bought before the project for other uses.

Table 4: Information about the order of products.

Product	Price	Date of order
Two Acrylic Sheets	2 x 11.205 pounds	15/03/2018
Two Stepper Motors	2 x 9.99 pounds	31/01/2018
Two Drivers	2 x 4.80 pounds	31/01/2018
Three Ultrasonic Sensors	3 x 0.99 pounds	31/01/2018
Arduino UNO	15.16 pounds	16/11/2017

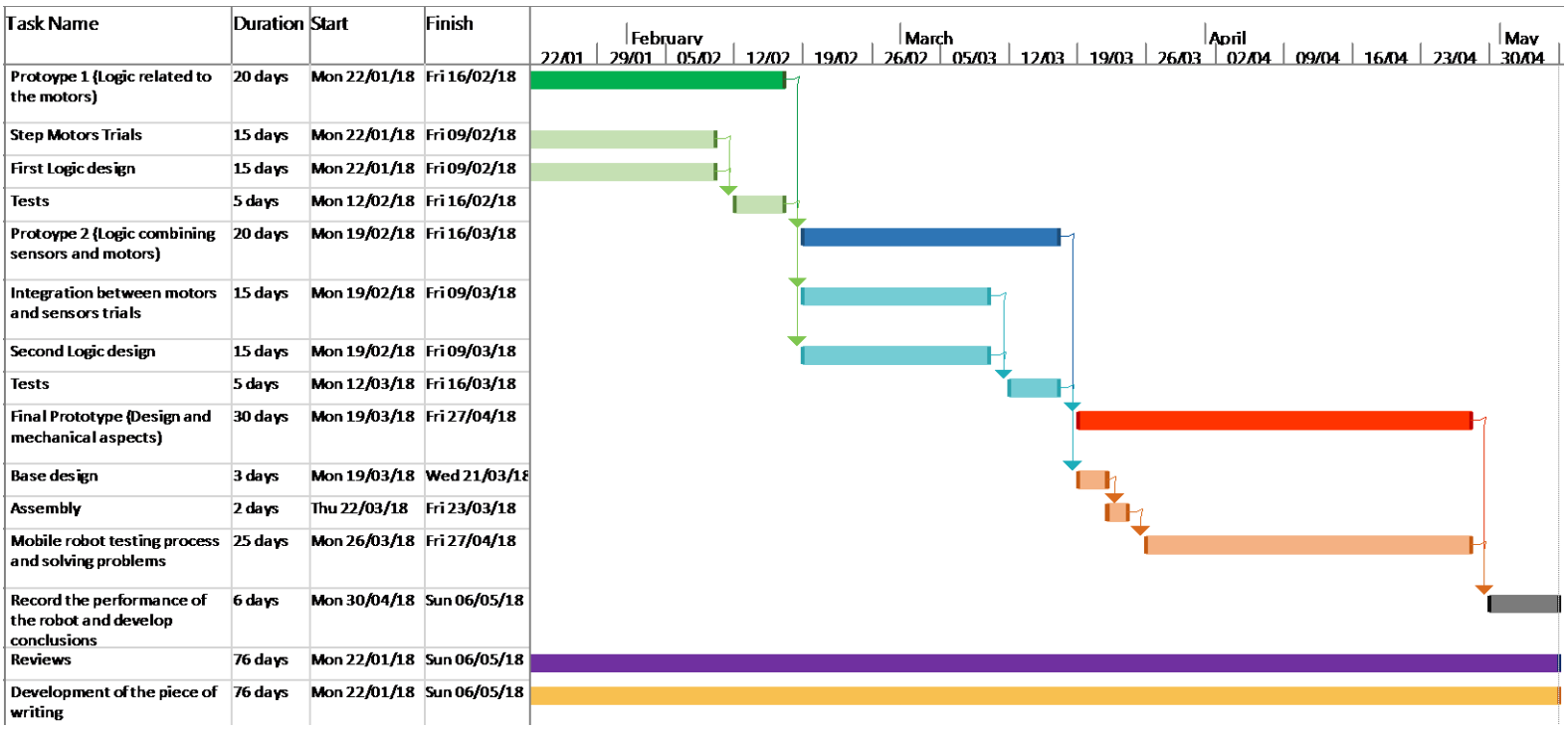


Figure 45. Gantt-Chart

7.2. Sustainability and socio-economic impact

The aim of the mobile robot developed is to contribute in the automatization of industrial processes, mainly in logistic activities. The great investment that automatization requires is repaid with the savings of money that it means in a long term. The reduction of costs for logistic activities could lead industries to spend that money into more sustainable processes for its production activities, such as changing their power supplier with renewable energies or buying modern devices with less environmental impact. Moreover, mobile robots, such the one made, depends on electricity to work, which could be also produced by renewable resources. These resources could charge a set of batteries which could charge the robots by shifts. While some robots are working, others are charging.

Automatization would mean an improvement for the economy of a business but, at the same time, could affect society. While some people enjoy the high quality of the products and the speed of their production and distribution, others could be losing their jobs due to their replacement by machines. Automatization would lead to a reduction in the workforce. Therefore, each company should analyze the balance between business economy, social impact and sustainability when considering the automatization of their processes.

7.3. Health and safety risk management

The project was developed in two different workshops: the electronics laboratory and the mechanical laboratory. The risks related to each one depended on the type of devices used, as shown in table 5.

In the electronic laboratory, a soldering device, a power supply and all the components bought for the project were used. A computer was also available for developing and loading the code to the Arduino UNO. The main safety measures taken there were related to protection while using the soldering device and the prevention from overvoltage. It was necessary to avoid direct contact to the material that was used to solder because it could have burned the skin. Moreover, the end of the soldier pen had to not be touch due to its high temperature. To avoid overvoltage, the voltage and current were controlled from the power supply, which had two wheels to vary each one. Apart from that, the specification of each electronic component had to be read carefully in order to respect its voltage and current limits. All the connections were double checked before turning the power supply on.

In the mechanical laboratory, the device used was a laser cutter to shape the acrylic sheet. This machine had a protective cover to prevent external contact during the cutting process. Therefore, the cover was opened to introduce the sheet and closed before the cutting process started. The dimensions of the sheet had to be loaded to the machine in an exact way because it could have burned the material if its width had been wrong. The machine also had a key to turn it on and off in order to prevent that it is not active after the cutting process had been done.

Table 5: Risk Assessment

ID	Risk Item	Effect	Cause	Likelihood	Severity	Importance	Action to minimise risk
1	Electronic Components	Being damaged and performing wrongly.	Overvoltage or overcurrent	M	M	H	Read carefully their specifications and respect them.
2	Soldering Device	Burning the skin.	Direct contact while using it	L	M	H	Avoid direct contact by having a clear space or using globes.
3	Laser Cutter	Cutting yourself or Fire damage	Not respecting the rules	L	H	H	Use the protective cover and the key. Respect the dimensions.

References

[1] M. Wilson, «Introduction,» de *Implementation of robot systems [electronic resource] : an introduction to robotics, automation and successful systems integration in manufacturing*, Amsterdam, Butterworth-Heinemann, 2015, pp. 1-18.

[2] C. M. D. I. E. MATERIALES, «CONACYT (Consejo Nacional de Ciencia y Tecnología),» 2018. [On line]. Available: https://centrosconacyt.mx/objeto/robotica/. [Last access: 3 May 2018].

[3] ISO 8373:2012; Robots and robotic devices — Vocabulary.

[4] I. Bambino, «Una introducción a los robots móviles,» AADECA, Buenos Aires, 2008.

[5] International Federation of Robotics, «Executive Summary WR Service Robots 2017,» *World Robotics Service Robots*, pp. 12-19, 11 October 2017.

[6] «Port of Rotterdam,» [On line]. Available: https://www.portofrotterdam.com/en/cargo-industry/50-years-of-containers/the-robot-is-coming. [Last access: 26 11 2017].

[7] J. G. Fernández, «Los robots de Amazon conquistan Europa,» *Expansión*, 16 July 2017.

[8] A. GALISTEO, «Así funcionan los robots que trabajan para Amazon,» *Expansión*, 16 10 2016.

[9] J. Lentin, Learning robotics using Python : design, simulate, program, and prototype an interactive autonomous mobile robot from scratch with the help of Python, ROS, and Open-CV!, Birmingham,UK: Packt Publishing, 2015.

[10] W. F. Deal III and S. C. Hsiung, «Sensors Expand the Capabilities of robot Devices,» *Technology & Engineering Teacher, Vol 76 Issue 7*, pp. 23-29, April 2017.

[11] D. Perea Ström, I. Bogoslavskyi and C. Stachniss, «Robust exploration and homing for autonomous robots,» *Robotics and Autonomous Systems; Elsevier B.V.*, vol. 90, pp. 125-135, April2017.

[12] A. A. S. Gunawan, William, B. Hartanto, A. Mili, W. Budiharto, A. G. Salman and N. Chandra, «Development of Affordable and Powerful Swarm Mobile Robot Based on Smartphone Android and IOIO board,» *Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017); Procedia Computer Science*, vol. 116, pp. 342-350, 2017.

[13] T. Okada, N. Mimura y T. Shimizu, «CHASSIS DESIGN OF A MOBILE ROBOT FOR REDUCING WEIGHT BY EXCLUDING SUSPENSION ELEMENTS.,» *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 9, nº 2, pp. 42-51, 2015.

[14] P. Mvemba, S. Guwa Gua Band, A. Lay-Ekuakille and N. Giannoccaro, «Advanced acoustic sensing system on a mobile robot: design, construction and measurements,» *EEE Instrumentation & Measurement Magazine IEEE Instrum. Meas. Mag. Instrumentation & Measurement Magazine, IEEE. 21(2):4-9* Apr, 2018.

[15] Y. Kim, D. Shin, J. Lee, Y. Lee and H.-J. Yoo, «A 0.55 V 1.1 mW Artificial Intelligence Processor With On-Chip PVT Compensation for Autonomous Mobile Robots,» *EEE Transactions on Circuits and Systems I: Regular Papers IEEE Trans. Circuits Syst. I Circuits and Systems I: Regular Papers, IEEE Transactions on. 65(2):567-580* Feb, 2018.

[16] A. Caverzasi, F. Saravia, O. Micolini, L. Mathe and L. Lichtensztein, «Robot móvil autónomo para crear mapas 3D en un ambiente acotado,» *2014 IEEE Biennial Congress of Argentina (ARGENCON) Biennial Congress of Argentina (ARGENCON), 2014 IEEE. :786-791* Jun, 2014.

[17] P. Armitage, "Mechatronics-Practical," University of Exeter (ECMM147), Exeter, 2017.

[18] J. Loureiro, «staticboards,» 13 June 2016. [On line]. Available: https://www.staticboards.es/blog/motores-paso-paso/. [Last access: 25 April 2018].

[19] «Amazon,» 2018. [On line]. Available: https://www.amazon.co.uk/MYSWEETY-Stepper-Stepping-36-8oz-Mounting/dp/B0761PWB1H. [Last access: 31 January 2018].

[20] «Pololu (Robotics & Electronics),» 2018. [On line]. Available: https://www.pololu.com/product/1182. [Last access: 31 January 2018].

[21] D. Nedelkovski, «How to Mechatronics,» 24 November 2015. [On line]. Available: https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/. [Last access: 1 February 2018].

[22] «Ebay,» 2018. [On line]. Available: https://www.ebay.co.uk/itm/Ultrasonic-Module-HC-SR04-Distance-Range-Sensor-Measuring-Transducer-for-Arduino/192142220006?epid=621492008&hash=item2cbc919ae6:g:aRUAAOSwSlBY2jYm. [Last access: 31 January 2018].

[23] «Arduino,» 2018. [On line]. Available: https://www.arduino.cc/en/Serial/Begin. [Last access: 28 April 2018].

[24] «Keenstone 3-Slot 9V PP3 Battery Li-ion Charger with 3 Pack 9V PP3 Li-ion battery Rechargeable 800mAh (USB Charging Cable Included),» Amazon, 2018. [On line]. Available: https://www.amazon.co.uk/dp/B078JMHHF8/ref=asc_df_B078JMHHF852274061/?tag=googshopuk-21&creative=22146&creativeASIN=B078JMHHF8&linkCode=df0&hvadid=205217260000&hvpos=1o2&hvnetw=g&hvrand=10345280015318841346&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocin. [Last access: 20 April 2018].

[25] «RS,» 2018. [On line]. Available: https://uk.rs-online.com/web/p/solid-plastic-sheets/0824654/. [Last access: 10 March 2018].

[26] «Viewpoints,» [On line]. Available: http://www.viewpoints.com/iRobot-Roomba-Discovery-Vacuum--563173-reviews. [Last access: 06 12 2017].

Appendices

They can be found on the following link:

https://universityofexeteruk-my.sharepoint.com/:w:/r/personal/gg326_exeter_ac_uk/Documents/Appendices-Guillermo%20G%C3%B3mez%20Pe%C3%B1a.docx?d=wea8743e1cc5b4258ad7c71ae86cd5e3a&csf=1&e=liT3cX