# LLUBot: The Educational Mobile Robot

Prototypes and technical designs

**Luisana Bella Zarahí Lara Briceño**

**Bachelor Thesis Advisor**
Jonay Tomás Toledo Carrillo

Universidad de La Laguna, 2020
Escuela Superior de Ingeniería y Tecnología
Department of Computer and Systems Engineering

*A todas aquellas personas que me insistieron,*
*apoyaron y animaron durante el camino*
*de este bonito proyecto.*

# Abstract

This bachelor thesis is the technical project of the educational mobile robot prototype as an innovative version of an open-source educational robot, which is destined to teach that the discipline of Robotics can include other like Electronics, Graphic Design, Mechanics and Programming. This disciplines are mainly taught when assembling the mobile robot itself and with the help of a learning guide, as this document can act as in a very technical way. This project is meant to cover the not taught subjects that most educational robots leave behind at a very low price and very accessible materials. And as a prototype, it is also meant to allow its control through wireless environments and, with deeper investigation, it can make possible the communication with other replicas of the same robot.

# Resumen

Este Trabajo de Fin de Grado es un proyecto técnico del prototipo de un robot móvil educativo como una versión innovadora de un robot educativo de open-source (fuente libre), que está destinado a enseñar que la disciplina de la Robótica puede incluir otras como Electrónica, Diseño Gráfico, Mecánica y Programación. Estas disciplinas son principalmente enseñadas a través del montaje del robot móvil en sí, con la ayuda de una guía de aprendizaje, que de forma muy técnica puede ser útil este documento. Este proyecto está pensado para cubrir las asignaturas no enseñadas que la mayoría de los robots educativos olvidan. Y como prototipo, también está pensado para que permita su control a través de entornos wireless, y con una investigación más profunda, puede hacer posible la comunicación de otras réplicas del mismo robot.

# Acknowledge

Gracias a mi madre por siempre estar atenta de si comía, si bebía agua, si dormía bien. Siempre estás cuando más te necesito, gracias.

Gracias a mi padre por apoyarme cuando inevitablemente retrasaba una vez más la fecha de entrega, a pesar de ya haber enseñado mi robotito. Gracias por creer en mí.

Gracias a mi gemela por enfadarte conmigo cuando más necesitaba que alguien me dijera que tenía que seguir, que no podía abandonar. Gracias por darme esas patadas en el culo que tanto me hacían falta.

Gracias a mi onee-chan por arrastrarme cada una de las veces que me olvidaba de lo maravilloso que es entrenar contigo. Gracias por esos momentos zen.

Gracias a D., a S. y a P. por acompañarme cuando más sola me sentí, porque sin su amistad no habría encontrado las fuerza para seguir con este proyecto.

Gracias a mi tutor, Jonay, por entender cada una de las veces que me echaba para atrás en terminar la memoria, por no indagar en mis fallos sino siempre darme la mano para seguir adelante y lograr este proyecto.

Gracias a B. por siempre preguntar qué tal me iba por Instagram, por animarme cada una de las veces que me tomaba un té en tu despacho, por no desistir en mí.

Gracias a P. por ayudarme tanto con las últimas programaciones, porque adentrarse en HTML, por muy fácil que sea el lenguaje (de marcas), siempre es difícil dar el primer paso para hacerlo mejor. Gracias por tu generosidad.

Gracias a C. por toda su ayuda en la creación del LLUBot, por imprimirme las piezas, por regalarme las regletas, por estar ahí todos los martes por la mañana, porque fuiste de gran ayuda y eso no se puede negar. Gracias por todo ese tiempo invertido en mí.

Gracias a mi gatita, Luna, por darme tanto amor en este último empujón.

Y gracias a B. y Z., porque sin ustedes no podría haber hecho un proyecto tan divertido y único.

# Contents

# List of Figures

# List of Tables

# Nomenclature

| | |
|---|---|
| ADC | analog-to-digital converter |
| cm | Centimeters |
| DC | Direct Current |
| deg | Degrees |
| DIY | Do-It-Yourself |
| FLL | FIRST® LEGO® League |
| g | Grams |
| I/O | Input/Output |
| IP | Internet Protocol |
| IR | Infrared |
| KB | Kilobytes |
| LED | Light-Emitting Diode |
| Li-Ion | Lithium-ion |
| Li-Po | Lithium-polymer |
| mA | Milliamperes |
| MHz | Megahertz |
| mm | Millimeters |
| PWM | Pulse Width Modulation |
| rpm | Revolutions per minute |
| s | Seconds |
| V | Volts |

# Chapter 1

# Introduction

There are many educational robots nowadays, such as MakerBlock® mBot [2], Wonder Workshop Robot Dash [3] and LEGO® Mindstorms® [33], which allow children and teenagers to learn basics of programming by easily assembled systems and fun interfaces. The main problem of these robots is usually the cost, making them not as accessible as they should be. Of course, we are not discussing the potential aids this enterprises can give to public education, but the actual cost of them as an obstacle to other private educations, as home education and other associations.

Due to this situation the idea of creating an open-source educational robot was founded, as well as the intention of making the learning wider and not only about programming. In this bachelor thesis, it will be exposed the design and implementation of the educational robot as well as the possible applications of it.

## 1.1 Motivation

The main thought that the current world is continuously thinking is where the technology is leading us to, and how to achieve the human purposes using those technologies. So when it comes to achieving those purposes, the design, creation and implementation of them is as much as important as the way new generations can get the necessary knowledge to keep advancing. This is why this bachelor thesis is thought to be meaningful and to allow as many people as possible to learn basics of robotics as fun as it can get.

## 1.2 Mission

And what exactly should a robotic system take to be useful? We should really consider many objectives. First, it is meant to be achievable and open source so it can be implanted or assembled in any part of the world, by a very low cost. Second, it is an adaptable system, as many materials of the robot can be different depending on the local available equipment. Third, it should be easy to understand the assembly instructions and the basics of the engineering in them. And last, but not least, it must be fun!

## 1.3 State of the Art Analysis

As it has been already mentioned, the current educational robots are many with variations between them. For example, the EV3 LEGO® Mindstorms [33] is the most common educational robot in Tenerife island, due to the high level of participation of public and private educational institutes in FLL (FIRST® LEGO® League) with this educational robot.

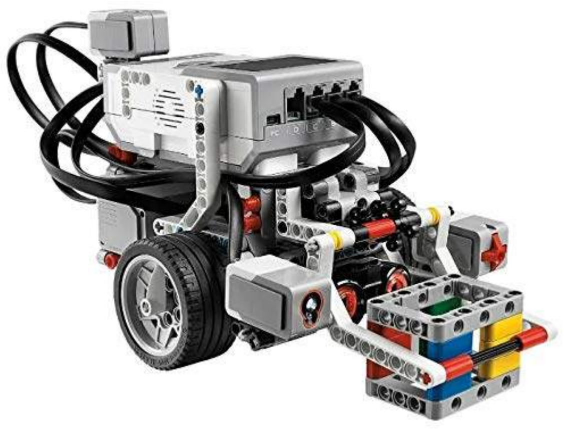


Figure 1.1: LEGO Mindstorms educational robot EV3 [1].

The characteristics of the EV3 are several, and it is a very complete robotic system. From the engineering point of view, the EV3 processor is a complex automaton provided with three communication systems to the programming software via USB, Bluetooth and WiFi, both by computer and tablet. It can enable up to 4 servo motors, between large ones "that uses tachometric feedback for speed control with one precision degree and has an integrated rotation sensor" [34] and medium ones "for applications with lower load and higher speed, and for when faster response times and a smaller size are needed in the design of a robot" [35], as well as up to 4 sensors, combining color sensor [36], ultrasonic sensor [37], touch sensor [38], gyro sensor [39] and IR sensor [40], which is a wide variety for a single educational automaton, see figure 1.1. This educational robot at the present costs around 472 € in Spain, IVA excluded.

The characteristics of the mBot are fewer than the EV3, as it had been less developed, but it also allow the attachment of several motors, as well as several Arduino based sensors, within its controller, see figure 1.2. This educational robot has the advantage of a more interactive app for mobile devices with Bluetooth than the Mindstorms one. The cost of this robot is 89.9 € [41].

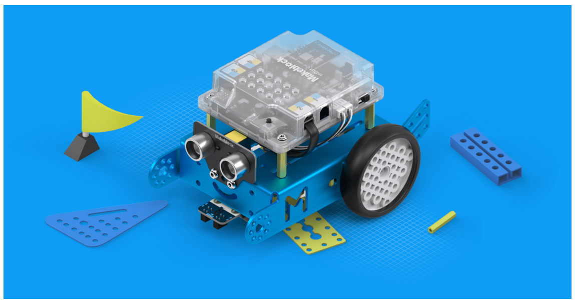


Figure 1.2: MakerBlock educational robot mBot [2].

And the characteristics of the Dash Robot are more specific onto programming features, because it is focused

Figure 1.3: Wonder Workshop educational robot Dash [3].

on fun and interactive experiences with voice control, dancing and singing learning strategies, see figure 1.3. The cost of this educational robot is around 150 €, internationally.

# Chapter 2

# Theoretical Background

In this chapter, we are going to talk about the theoretical basis of the prototype which is the result of the project, for it contains several devices and methods in its design to be described previously. There are four distinguished categories as they are disciplines used to describe a fully functional mobile robot. But first, let us describe the very concept of mobile robot and its varieties.

## 2.1 Components of a functional mobile robot and its basis

A mobile robot is a robot "that can move from one place to another autonomously, that is, without assistance of external human operators" [42]. With that purpose, the robot must have a control system for the location and interaction with the environment, which will contain a sensor system and a motion system with different characteristics and other necessary ones to be one fully functional robot, such as a power system and the communication system. This are the ones that we will be describing in the next sections, and the different possibilities depending on the uses they have in the current technologies.

### 2.1.1 Control system

A controlled system is one that must have feedback in it, for it has certain measured and controlled variables to get the requested results. In order to do so, a mobile robot must have a main controller, with at least one peripheral (sensor) which will be measuring the most important issue in a robot of this kind: the position.

For this subject, we will be talking about different open-source (meaning open-software and open-hardware) microcontroller boards, such as those seen in figure 2.1 and their features, for in the next sections will be discussed the other subsystems.

These microcontroller boards are the ones that would save, process and com-

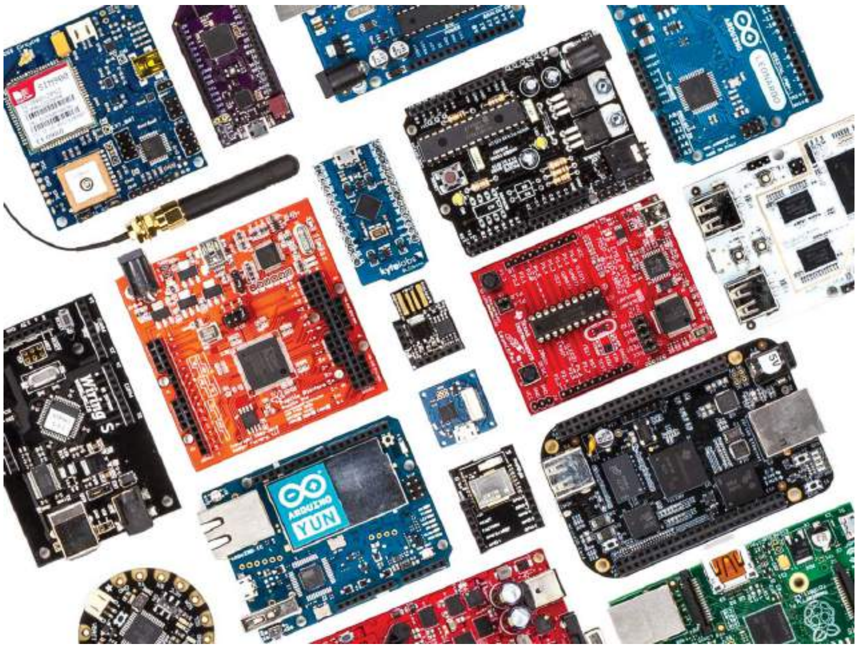


Figure 2.1: Several microcontroller boards often used [4].

municate the entire information through out the different devices in the subsystems.

### Arduino UNO

It is "the best board to get started with electronics and coding" [6] as the official Arduino web page tells, and it is true because it is designed to be easy to learn, easy to use and multifunctional, as it is in deed a programmable microcontroller board with plenty different types of possible connections and use of data.

The main characteristics of the Arduino UNO are as follows (see figure 2.2):

- Microcontroller: ATMega238P [43]. It is a High Performance, Low Power AVR® 8-Bit Microcontroller, with Advanced RISC Architecture, High Endurance Non-volatile Memory Segments, 23 Programmable I/O Lines, 6 PWM Channels.
- Operating Voltage: 5V.
- Recommended Input Voltage: 7V to 12V.
- Limits of Input Voltage: 6V and 20V.
- Digital I/O Pins: 14 (from which 6 provide PWM).
- Analog Input Pins: 6.
- DC Current per I/O Pin: 40 mA.
- DC Current for 3.3V Pin: 50 mA.

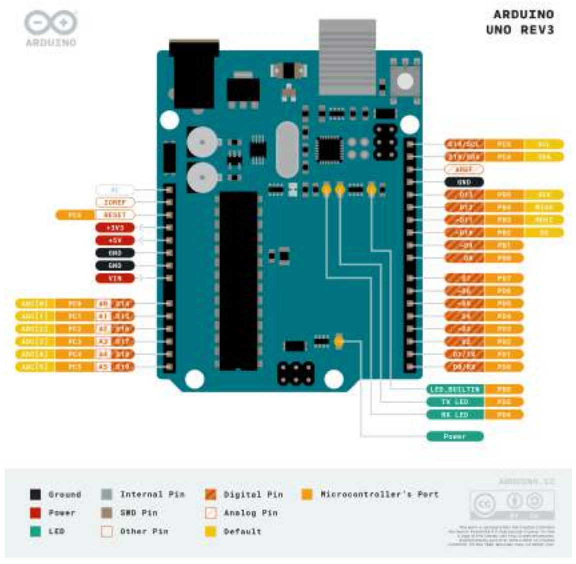


Figure 2.2: Arduino UNO Pin-out Diagram [5].

- Flash Memory: 32 KB (ATmega328) of which 0.5 KB used by bootloader.
- SRAM: 2 KB (ATmega328).
- EEPROM: 1 KB (ATmega328).
- Clock Speed: 16 MHz.
- Length: 68.6 mm.
- Width: 53.4 mm.
- Weight: 25 g.

The schematics of this microcontroller board shows, in figure 2.3, where are the 5V for power, the voltage regulator and all of the different pins, as well as the names related to the pin-out diagram of figure 2.2.

### Arduino Mega

It is a "8-bit board with 54 digital pins, 16 analog inputs, and 4 serial ports" [44], which means a much improved Arduino board, capable of communicating with 40 more digital pins, 10 more analog pins and twice the number of serial ports (UARTs) than Arduino UNO has, among several better characteristics. So, this board is made for extensive use of peripherals and communication between other devices, specially by serial ports.

The main features of the Arduino Mega 2560 are as follows (see figure 2.4):

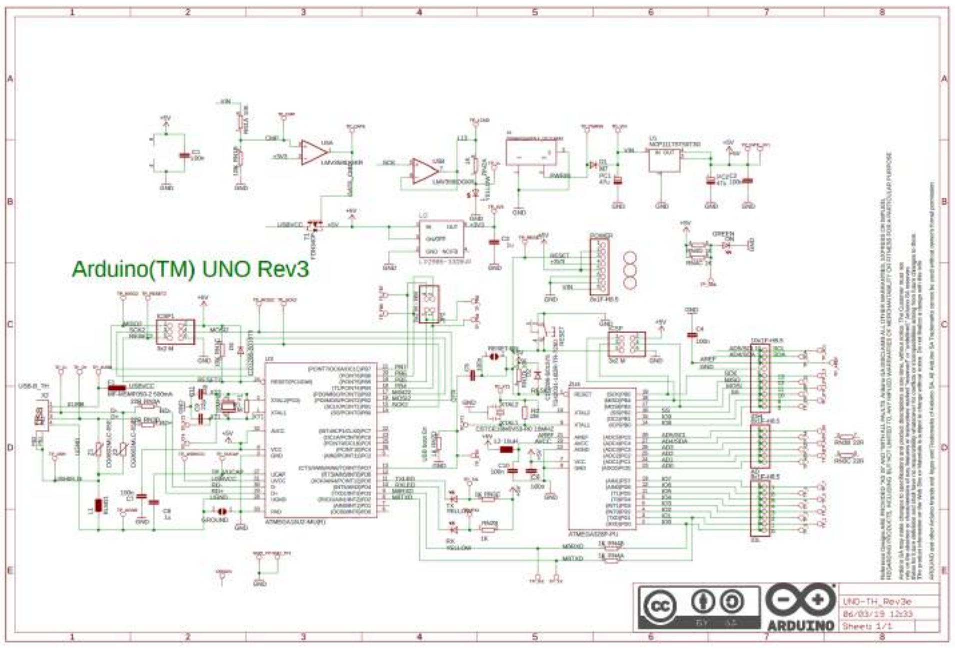


Figure 2.3: Arduino UNO Schematic [6].

- Microcontroller: ATMega2560 [45]. It is also a High Performance, Low Power AVR® 8-Bit Microcontroller, with Advanced RISC Architecture, High Endurance Non-volatile Memory Segments, two 8-bit Timer/Counters, four 16-bit Timer/Counter, an operating voltage between 1.8V and 5.5V, among dozens of other specifications.
- Operating Voltage: 5V.
- Recommended Input Voltage: 7V to 12V.
- Limits of Input Voltage: 6V and 20V.
- Digital I/O Pins: 54 (from which 15 provide PWM).
- Analog Input Pins: 16.
- DC Current per I/O Pin: 20 mA.
- DC Current for 3.3V Pin: 50 mA.
- Flash Memory: 256 KB (ATmega2560) of which 8 KB used by bootloader.
- SRAM: 8 KB (ATmega2560).
- EEPROM: 4 KB (ATmega2560).
- Clock Speed: 16 MHz.
- Length: 101.52 mm.

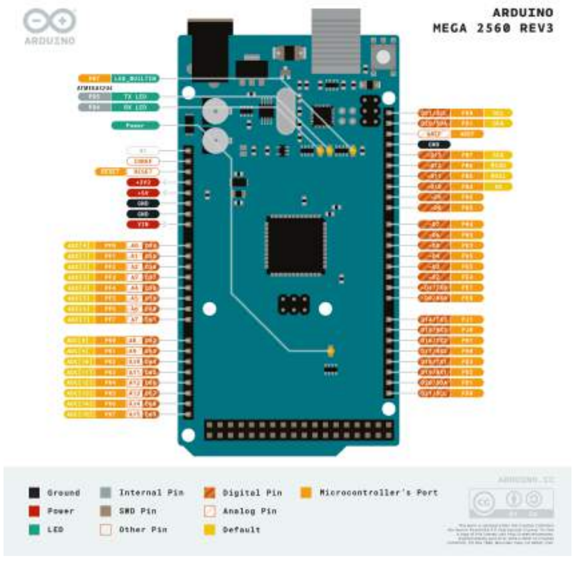


Figure 2.4: Arduino Mega 2560 Pin-out Diagram [7].

- Width: 53.3 mm.
- Weight: 37 g.

The schematics of this microcontroller board shows, in figure 2.5, how it has greater capabilities compared to the Arduino UNO seen before, as well as the relation with the pin-out diagram of 2.4.

### WeMos D1 R2

This board is a implementation of the ESP8266 microprocessor into the Arduino UNO hardware design, and it is "a comfortable board to work with ESP8266" [46] which means, an easy way of making use of WiFi connections without using external peripherals. It is a much quicker microprocessor, but with fewer I/O pins compared to the other two boards, and with a different operating voltage of them, something that must be contemplated in the electronics used.

The main characteristics of the WeMos D1 R2 as follows (see figure 2.6):

- Microcontroller: ESP8266EX. "Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development" [47].
- Operating Voltage: 3.3V.
- Recommended Input Voltage: 9V to 24V.

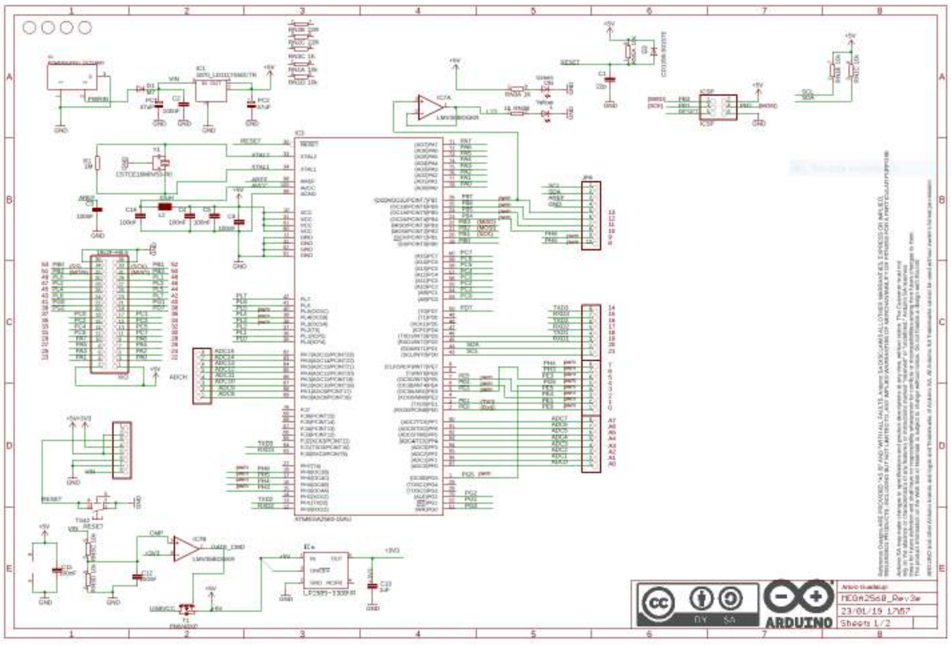


Figure 2.5: Arduino Mega 2560 Schematic [8].

- Digital I/O Pins: 11 (all pins have interrupt/PWM/I2C/one-wire supported (except for D0)).
- Analog Input Pins: 1 (Max input: 3.2V).
- DC Current I/O Pin: 80 mA.
- Flash Memory: 16 MB (ESP8266).
- SRAM: 50 KB (ESP8266).
- Clock Speed: 80 MHz / 160 MHz.
- Length: 68.6 mm.
- Width: 53.4 mm.
- Weight: 25 g.

The schematics of this microcontroller board shows, in figure 2.7, no greater pin-out capabilities compared to the Arduino UNO nor the Mega 2560 but major improvements in speed, memory and communication features, as well as the relation with the pin-out diagram of 2.6.

Each peripheral connected with a microcontroller must be electronically configured to behold the characteristics of its correct connectivity, in case, a certain motor functioning through a Arduino UNO must be specifically powered and data

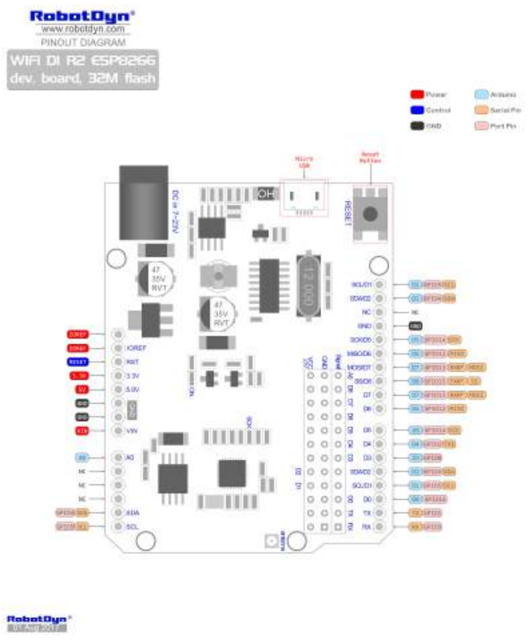


Figure 2.6: WeMos D1 R2 Pin-out Diagram [9].

must be sent by appropriate pins, as well as rightly configured in the coding, always depending on the purpose of the system. This is when the different microcontroller boards must be selected correctly, being always able to perform the specifications.

## 2.1.2 Motion system

It seems quite obvious to describe the needs of a motion system in a mobile robot, but it is not to describe the different options available in the market, because at the end the specifications will point the most appropriate selection to make a suitable motion system.

For this, we will discuss up to four different types of motors as well as the way to connect them to the previously discussed microcontroller boards.

### DC Motors

The first, and most used one because of its very low cost, is the DC motor , which is "a class of rotary electrical motor that converts direct current electrical energy into mechanical energy" [12]. There is one DC motor of great use in educational mobile robots, seen in figure 2.8, which its most relevant specification is its operating voltage: within 3V and 12V.

This kind of motor operates with an almost direct relation between the input voltage and the speed of the rotation. In the case of the previously mentioned DC motor, its functioning is based in an electromagnetic response to certain current within one or more pairs of complemented magnets, as seen in figure 2.9. This

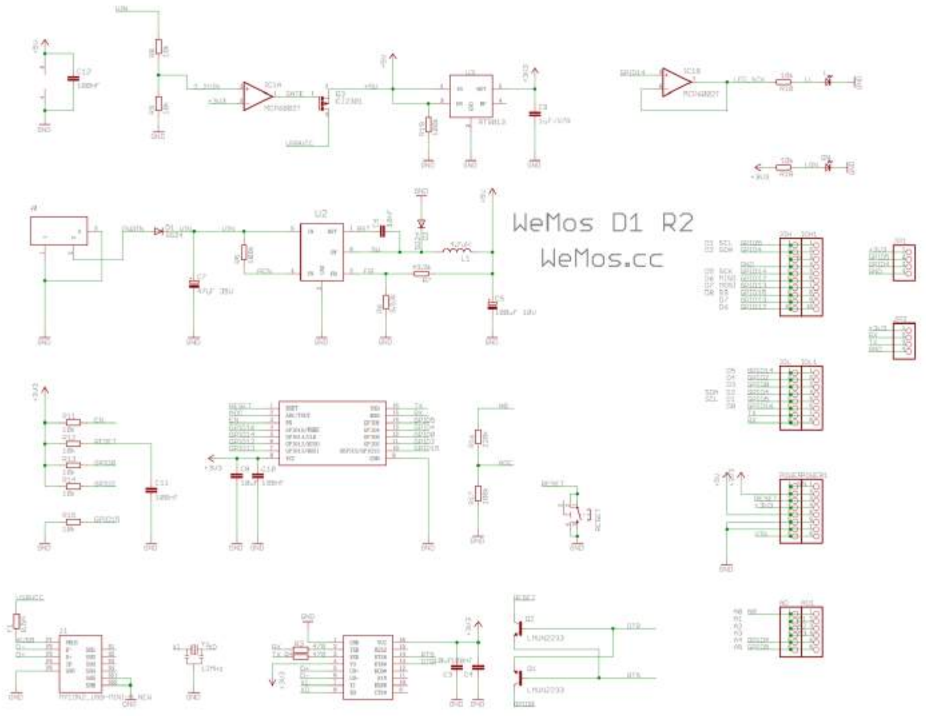


Figure 2.7: WeMos D1 R2 Schematic [10].

specific model has incorporated within itself a speed reductive device of 48:1, which means it runs 48 times slower than the same motor without this reductive device.

The main problems of the use of the DC motors in a mobile robotic system is the accurate relation of the voltage-speed as well as the correct position control system required for a mobile robot. This is because the first relies entirely on the mobile powering system, which usually will be a battery, so the changes on the power source usually cause direct changes on the applied voltages for the speed control. At the same time, this relation will have consequences on the position monitoring, but it can be controlled by a simple encoder. This last will be discussed in the next chapter.

### Stepper Motors

A stepper motor is basically an electronically commuted DC motor that "divides a full rotation into a number of equal steps" [13], and is constructed as an open-loop controlled system, this is due to its ability to move to a certain position in its cycle or hold one specifically without the need of a position sensor.

The most popular stepper motor in a maker environment is the 28BYJ-48 stepper motor, seen in figure 2.10, which has a well considered low cost and a very correct performance and critics. This one, for instance, has an operating voltage from 4.5V to 5.5V, with its nominal value in 5V. It makes it very applying for its use with one of the previously mentioned microcontroller boards as they all have this power source available within them, and so it make them the most popular stepper

Figure 2.8: DC motor used for Arduino based Rover [11].

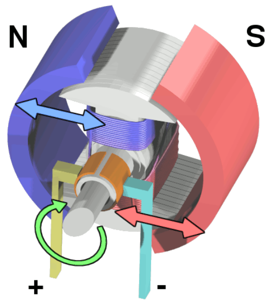


Figure 2.9: Electric motor cycle [12].

motors used for this kind of systems, at the present times of course. And because of this, it is a highly documented device with many code libraries and external control devices for it simple attaching and use in microcontroller boards such as Arduino UNO.

They also give a very easy solution of position control within their very functioning, at a very low cost, so the only important problem would be the lack of feedback in the state of a load way above the torque it can produce, which must be great enough for the motion of the robotic system. And maybe they are quite slow for a mobile robot, but still good enough.

### Servo Motors

A servo motor can be described as a High-Performance version of the stepper motor, because it bases its functioning in the same electronically commuted DC motor working, but with a position encoder sensor attached into its system, usually a potentiometer. This allows the movement to be in certain degrees, from -90º to 90º or from 0º to 90º, with a large variety of cost and power consumption, depending on the torque searched for.

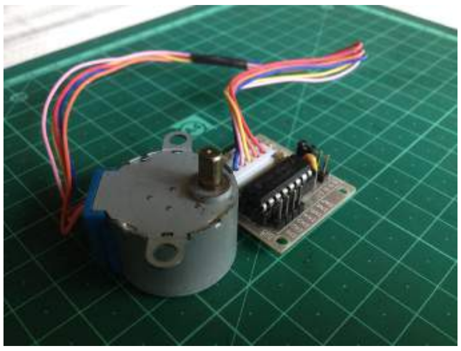
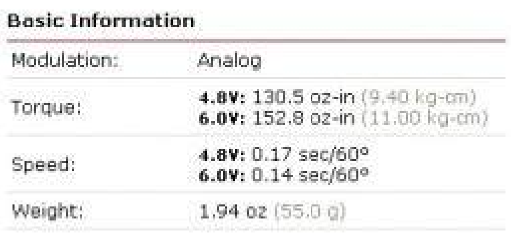


Figure 2.10: 28BYJ-48 stepper motor [13].

A commonly used type of servo motor in mobile robotics is the continuous rotation servo motor, seen in figure 2.11, which gives the ability to move the axis of the device in a way or the other without any blocking characteristic on the movement of the motor, by removing the potentiometer within it. This is the main difference with its brother the micro servo motor, see figure 2.12, which can be used instead in a different part of the motion system, for instance, moving another device to a precise position for an specific purpose (i.e. move a distance sensor for the detection of obstacles). The continuous rotation servo motor is used to other purposes, such as the motion of the wheels of a mobile robot.


**Basic Information**

| | |
|---|---|
| Modulation: | Analog |
| Torque: | **4.8V:** 130.5 oz-in (9.40 kg-cm)<br>**6.0V:** 152.8 oz-in (11.00 kg-cm) |
| Speed: | **4.8V:** 0.17 sec/60°<br>**6.0V:** 0.14 sec/60° |
| Weight: | 1.94 oz (55.0 g) |

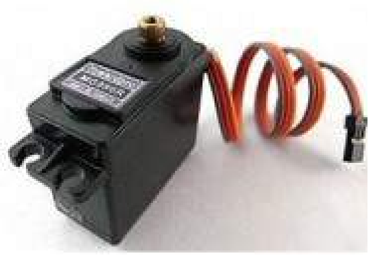


Figure 2.11: Continuous rotation servo motor [14].

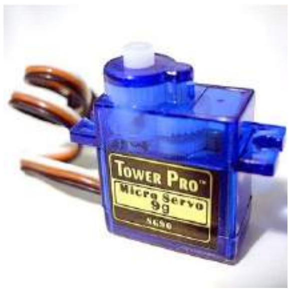


Figure 2.12: Continuous rotation servo motor [15].

### Arduino Driver Motor Shield

Although it can fit both the characteristic as control system, this device, the Arduino Driver Motor Shield (figure 2.13), is contemplated in the motion system because its purpose is to facilitate the control of the selected motors into an Arduino UNO compatible microcontroller board.

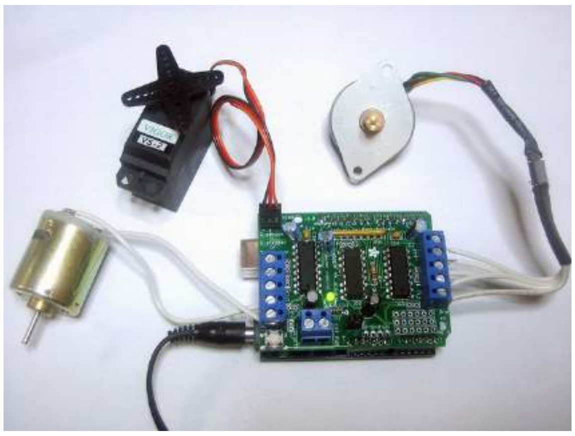


Figure 2.13: Adafruit Arduino Driver Motor Shield types of use [16].

This is a particularly special and troubling device for the electronic characteristics that must be taken into account when making a compatibility with a non-tried-before microcontroller board, such as the WeMos D1 R2, which has the adaptability for any "shield" created for Arduino UNO (which means, the same design, number of pins and its positions imitated with a specific extended and easier use of the peripherals) but with few but important differences.

It can control up to 4 DC motors and 2 servo motors at the same time, or using each 2 DC motor connections to control 1 stepper motor, seen in figure 2.14.

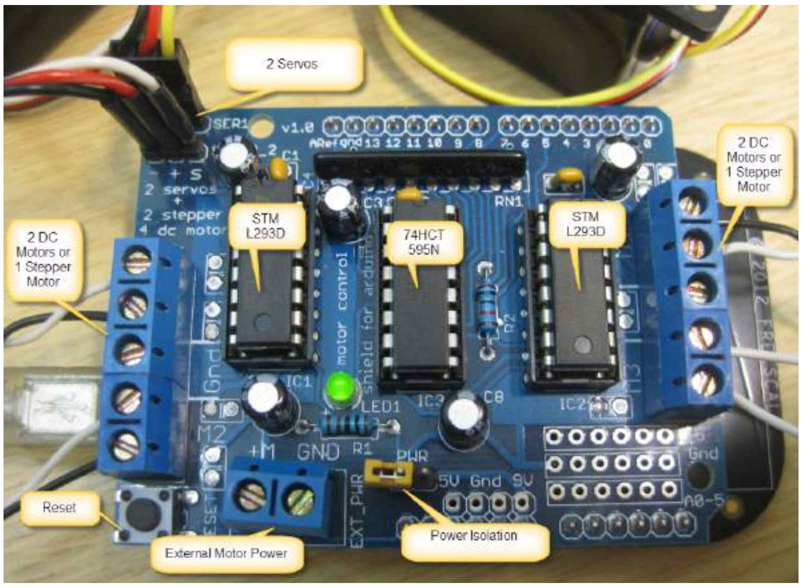


Figure 2.14: Adafruit Arduino Driver Motor Shield details [17].

The functioning of this shield is based on the use of a shift register and four H-Bridges, as well as several other components seen in figure 2.15.

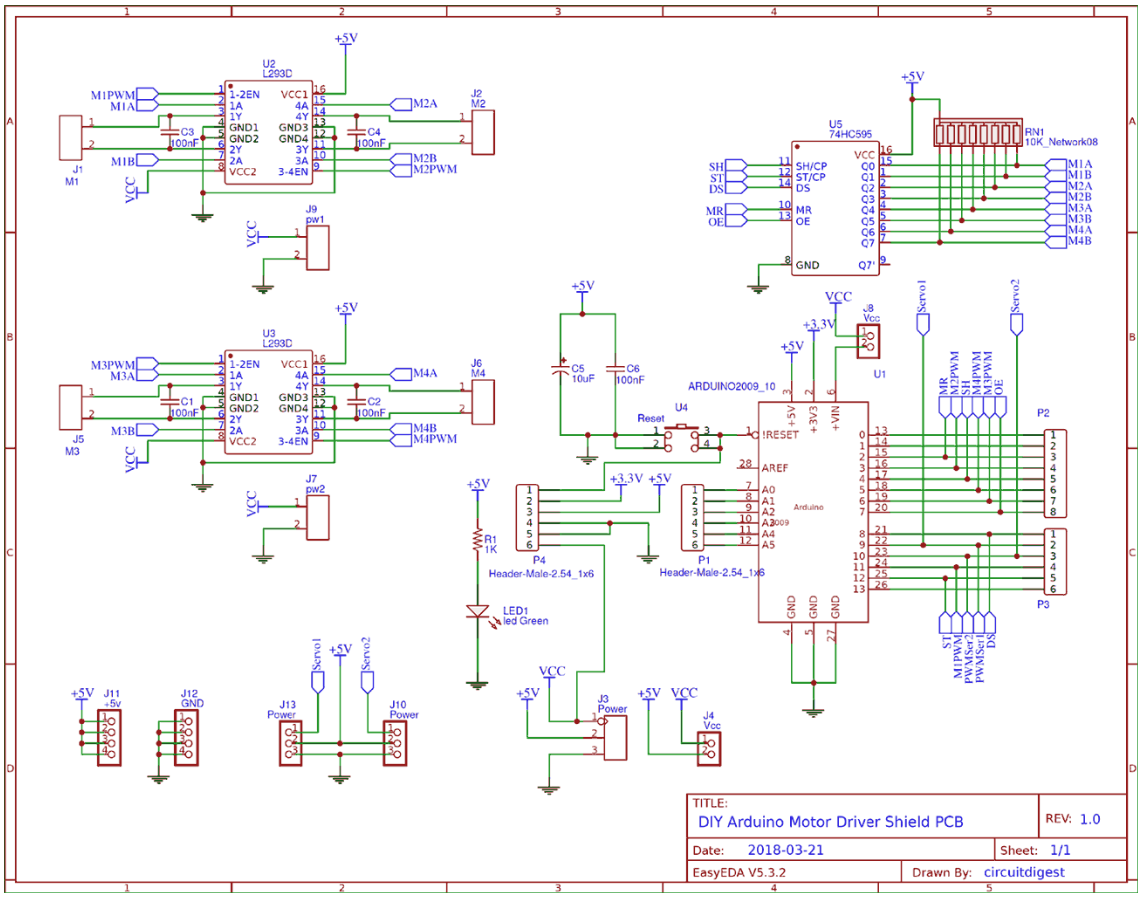


Figure 2.15: Adafruit Arduino Driver Motor Shield Schematic [18].

A shift register is a sequential logic circuit made with a series of Flip-Flops, capable of storage and transfer of data. It has the ability of reading serial or parallel data and sending serial or parallel data, when needed (see figure 2.16).

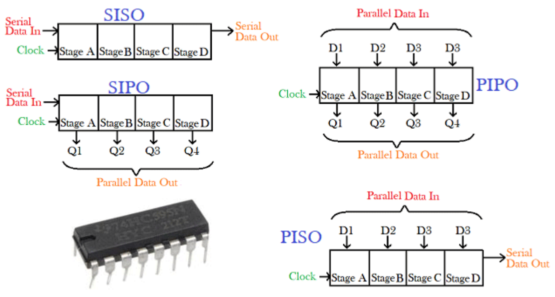


Figure 2.16: Shift Register's types of working [19]

The 74HC595 shift register is the one used in the arduino motor driver shield with its main function to translate the 5 parallel inputs (SH, ST, DS, MR and OE) into 8 parallel outputs for the ordering of movement for the different 4 DC motors connections, going through 4 H-Bridges to determine the direction of the DC motors or steppers used, see figure 2.17.

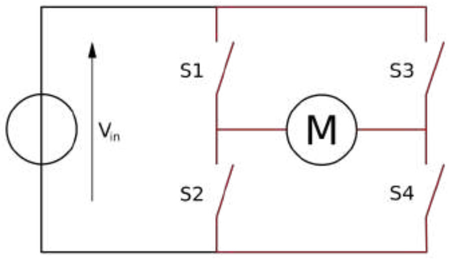


Figure 2.17: H-Bridge basic working [20]

Both pairs of H-Bridges (L293D microchips) and the 74HC595 shift register are powered by the "5V" pin on the driver shield, and this means the digital signal inputs must be between 3.5V and 5.5V when the source is 5V. If the microcontroller I/O pins are operating at a different voltage than 5V, this must be regarded.

For instance, the WeMos D1 R2 has I/O pins operating at 3.3V, so the usual arduino motor driver shield operation wouldn't read any signal coming from the WeMos, but that has an easy fix: reducing the operating voltage of the motor shield just the necessary to both adequate the minimum reading voltage ($V_{din}$) of the signals coming from the microcontroller board, as well as adequate to the most restrictive minimum operation voltage between the H-Bridges and shift register, which is 4.5V [48] (because the other minimum is 2V [49]).

The great idea to solve this problem was to reduce the input voltage to 4.5V, which is the minimum reading voltage of the H-Bridges, the most restrictive one, $V_{din}$. This would cause the logic "1" voltage to reduce to around 3.0V, and that allows the I/O pins to be properly read by the motor shield's electronics. This part of the fundamentals of electronics is very important because it is here where the motion system relies on to be able to work or not, as much a great code is achieved: the powering system is fundamental.

### 2.1.3 Sensor system

In a functional robotic system, there must always be a sensor system for the feedback analysis and control of the entire robot. For instance, a mobile robot must have always a control on its position, and it can have several other inputs to have a greater controlled system, such as an obstacle sensing, via odometry or via touch control, for example. In this case, we have chosen, besides the position control, to work on the odometry, so two sensors were taking into account due to their low cost

Figure 2.18: Different sensors used in Arduino projects [21].

and easy implementation.

### Position Monitoring: Odometry

The most easy way to control a mobile robot is counting how much does it move, and that is what odometry is about, "the use of data from motion sensors to estimate change in position over time" [50].

The real issue is how to accomplish a sufficient odometry in a mobile robot: by a encoder added to a DC motor? Or maybe with the extra accurate functioning of a stepper motor, which were described in section 2.1.2. Both position control's operations will be described and compared as following:

**POSITION CONTROL BY ENCODER:**

The encoder is a device that "converts motion to an electrical signal that can be read by some type of control device in a motion control system, such as a counter or PLC" [22], in this case, a microcontroller board.

The most commonly used type of encoders is the optic one, which works with a light sensing mechanism, see figure 2.19. This is actually the usually attached one to a common servo motor, such as the one in page 34.

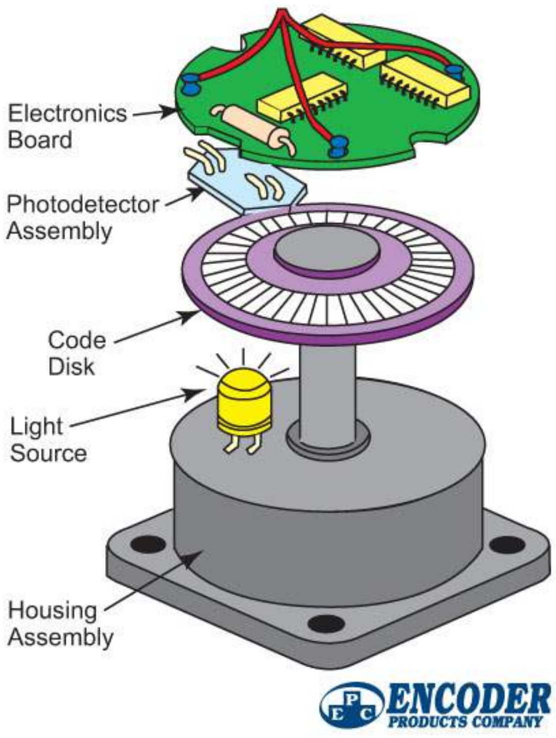


Figure 2.19: Optical encoder's design and elements [22].

The variations of position are calculated through the detection of light received in two separated photodiodes as sensors with the use of a LED and several holes, so that each time the light is detected by both photodiodes, a turn has been taken and so the movement is easy to calculate (in relation with the perimeter of the wheel, as seen in equation 2.1). For the direction of the turning, the order of the pulses in each photodiode has to be taken into account, so that the position is properly measured in ways of the direction of the movement.

$$P = 2\pi \times R \tag{2.1}$$

Where, $P$ is the value of the perimeter of the wheel and $R$ is the radius of the wheel.

The travelled distance can be determined in relation of the number of slits counted, as the total number of slits would give us one complete turn, and so the travelled distance can be measured if we know the radius of the wheels, in a relation such as equation 2.2.

$$D = N_s \times \frac{P}{N_t} \tag{2.2}$$

Where $D$ is the travelled distance, $N_s$ is the counted number of slits detected thought the optical encoder, $P$ is the perimeter of the wheel and $N_t$ is the total number of slits in the optical encoder that make one turn.

The cost and reliability on them are the worst concern when using this devices, specially when designing a prototype accessible and trustworthy. This would be discussed in next chapter.

**POSITION CONTROL BY STEPPER MOTOR:**

The steppers, as we already know from the short explanation on page 34, have an open-loop position control system within them, so they are both cheap, easily found in any electronics store and serve as "position sensor" it self (as long as its torque isn't exceeded).

The relation between the number of steps this kind of motor can take is exactly 1 revolution, so the position-steps relation is as equation 2.3, where $K$ is the relation between, $S_t$, the number of steps that a stepper can take in a revolution and the perimeter of the wheels, and $R_w$ is the radius of the wheels in cm.

$$K = \frac{2\pi R_w}{S_t} \tag{2.3}$$

Once gotten this, and knowing that $S_L$ and $S_R$ are the number of steps taken in the left wheel and right wheel, respectively, and the sampling interval of this movement measurement is $I$, so that the $\Delta D_{L,i}$ and $\Delta D_{R,i}$ are the incremental travel distance for the left and right wheel, respectively is as shown in equation 2.4.

$$\Delta D_{L/R,i} = K \times S_{L/R,i} \tag{2.4}$$

This allows the calculation of the incremental linear displacement of the robot's centerpoint C, denoted $\Delta D_i$, as seen in equation 2.5, where b is the distance between the two contact points between the wheels and the floor.

$$\Delta D_i = (\Delta D_{L,i} + \Delta D_{R,i})/2 \tag{2.5}$$

The next calculation is the incremental change of orientation of the robot $\Delta\Theta_i$, as seen in equation 2.6

$$\Delta\Theta_i = (\Delta D_{L,i} + \Delta D_{R,i})/b \tag{2.6}$$

So the new relative orientation of the robot $\Theta_i$ can be calculated as in equation 2.7

$$\Theta_i = \Theta_{i-1} + \Delta\Theta_i \tag{2.7}$$

Therefore, the relative position of the centerpoint is as written in equations 2.8 and 2.9, where $x_i$ and $y_i$ are the two coordinates of the robot's relative position in its centerpoint $c$ at instant $i$.

$$x_i = x_{i-1} + \Delta U_i \times cos\Theta_i \ [cm] \tag{2.8}$$

$$y_i = y_{i-1} + \Delta U_i \times sin\Theta_i \ [cm] \tag{2.9}$$

All this equations are based on the well-known ones for odometry seen in "Where am I? Sensors and Methods for Mobile Robot Positioning" [51].

Something very important to take care when steppers are in use is the need to create movement by portions of steps at a time, as they tend to fail steps when commanded to move complete and multiple steps with the commuted electronics. Instead, the tendency is to use microsteps of 1/4 or 1/8 of a step for the signals to move the motor. This means that when we program the microcontroller into moving the stepper, each microstep would be a quarter or an eight part, and so we must multiply this relation for the previous equations to work properly.

The entire position control could be figured by a series of sums of the steps taken by each motor, creating a trajectory mathematically modelled.

### Obstacle Sensing

When creating a mobile robot, there should always be a way to detect if there is an obstacle in the way, just to be certain that the robot itself won't collide and suffer damages. For this, there are plenty solutions, but the most usual one in educational robotics are the ones using distance sensors.

**ULTRASONIC SENSOR:**

The ultrasonic sensor module HC-SR04, see figure 2.20, is the most commonly used distance sensor in the Arduino learning world, as it is widely documented and easily understandable.

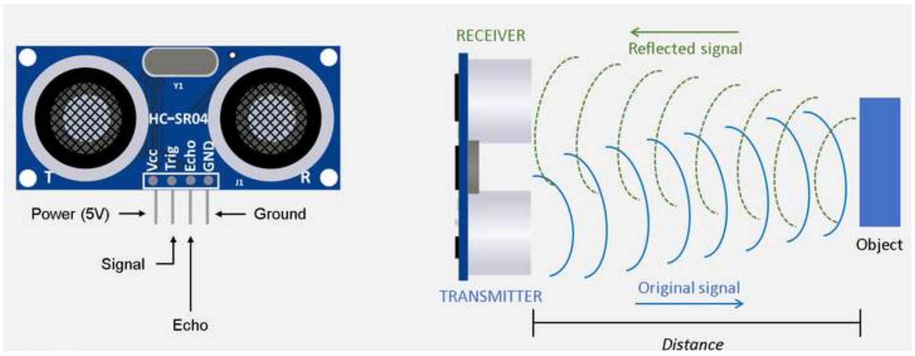


Figure 2.20: Arduino Ultrasonic Sensor Module HC-SR04 [23].

The operation of this sensor is through the physical behaviour of the ultrasonic waves: a "trigger" produces a signal so that when that same signal has a rebound and gets to "echo", a sound sensor, the distance would be calculated through equation 2.10, where $D$ is the distance between the sensor and the detected object in centimeters, $c$ is speed of sound [52] in the air with an approximate value of 343.2 $m/s$ or $34,320$ $cm/s$ and $\Delta t$ is the time between the departure and the reception of the ultrasonic signal in seconds.

$$D = \frac{c \times \Delta t}{2} \ [cm] \tag{2.10}$$

This module can reach distances from 2 cm to 4 m [53], which has both advantages and disadvantages: it allows to have a great range but at low accuracy and reliability, as the sound waves are very easily interfered by other mechanical waves.

**SHARP SENSOR:**

The sharp sensor GP2Y0A21YK0F seen in figure 2.21 is a type of IR sensor, with a different way of measuring the distance between a transmitter and receiver, which works with a position-sensible photo detector (PSD) for which the horizontal position in which the light beam strikes makes the distance measure possible, as seen in figure 2.22.

Figure 2.21: Sharp GP2Y0A21YK0F IR sensor [24].

The relation between distance and the light beam strike [25] is in equation 2.11, where $D$ is the distance in cm, $ADC$ is the digitalized value of the input voltage into the microcontroller board, $a$ and $b$ are constants determined from the datasheet, as the source page proportioned, with approximate values of 27.726 and $-1.2045$, respectively.

$$D = a \times ADC^{b} \ [cm] \tag{2.11}$$

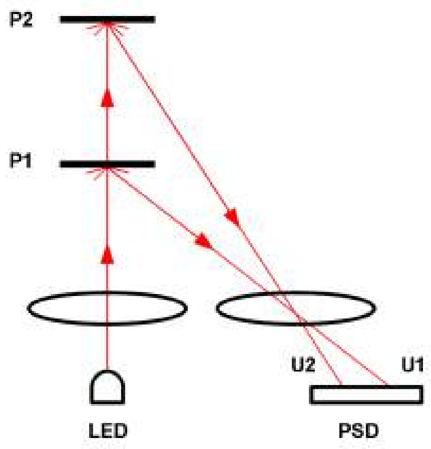


Figure 2.22: Sharp sensor working principle [25].

This is a most precise way of measuring distance, as a PSD system has a much better behaviour than other commonly accessible sensors.

### 2.1.4 Communication System

The purpose of the communication system is to provide an easy way for the students to learn a bit of computational thinking, applied mathematics and other possible techniques useful in many different pure and applied sciences, providing them with practical knowledge in their learning process with this educational robot.

There are many ways in which one robot can be programmed or instructed to perform one or several tasks, which are the ones that will be discussed in this section, as follows.

#### USB

A system programmed via an USB is limited usually by the availability of a computer from which a previously installed software must transmit the information of a code, for instance, Arduino IDE to any Arduino board or similar, such as WeMos D1, as well as Scratch, and so on with the different brands of educational robots.

This is one of the most used system for creating a proper educational environment, specially when focusing on the programming learning, but it may also be inaccessible at certain economical levels.

#### WiFi

The use of a WiFi designed communication system may provide an easy way for several students to connect to a web server through any device connected to the same WiFi network, via different IP addresses for each connected educational robot.

This allows the education to be mobile, easy to connect and very accessible as any mobile phone capable of connecting to the internet can be the programming device, when talking about the learning program for the students.

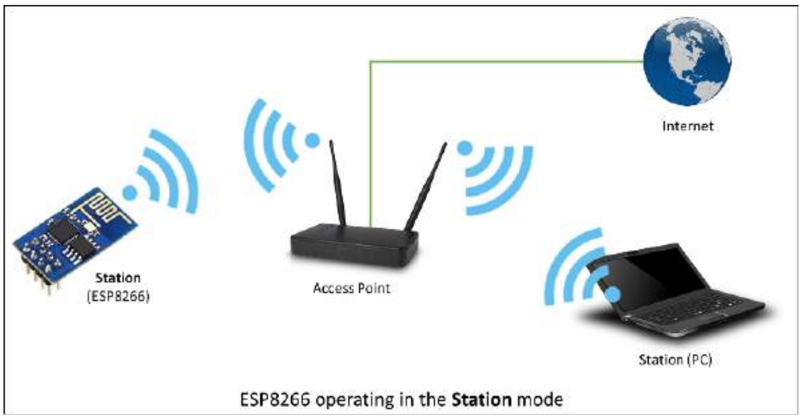


Figure 2.23: WiFi station mode description [26].

The basic functioning of this system is that a station (mobile phone, computer, tablet, etc) connects to another station (the robot with a ESP8266 module or similar) through an access point (router or another mobile phone providing a WiFi network), as seen in figure 2.23.

#### Swarm System

Originally, the design was thought to be for a swarm system of educational robots, creating a greater leap between the usual robotic systems in current education. Although the results are not this kind of system, its requirements for creating an appropriate communication system are that all of the robots have an adequate path, such as WiFi, with certain communication levels for the creation of an educational use of various robots.

For instance, the idea of creating a swarm system of robots is to make the students learn about the team work and planning of events in order to resolve certain problems.

### 2.1.5 Power System

It is very important that the mobile robot is properly powered, as the whole system depends on the correct voltage supply, as the specifications require sufficient autonomy and intensity of electric current for the powering of both wheels and another motor for the movement of the sensor implemented, at the same time that the microcontroller is powered. For this, there are a few types of batteries that can be taken into account, and the battery power control must be considered for the correct care of the selected battery.

### Types of battery

Although there are several useful kinds of battery, the most used ones in mobile robotics are the lithium batteries. There are basically two fundamental types of lithium batteries: lithium-ion (Li-Ion) and lithium-polymer (Li-Po), as seen in figure 2.24.

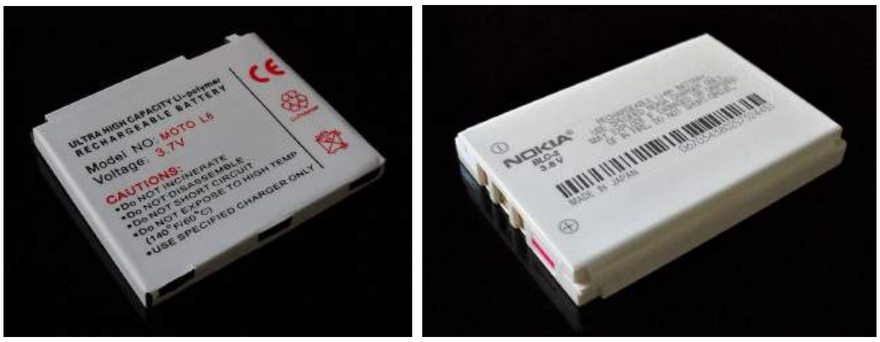


Figure 2.24: Li-Po battery (left) [27] and Li-Ion Battery (right) [28].

The main characteristic of this type of batteries is that they are very light, so when a small mobile system require the least weight possible, lithium batteries are the choice. And the main difference between both lithium battery kinds, are that Li-Ion has better life span, is very strong (against crashes) and a bit more expensive than the standard Li-Po battery. The comparison between the difference in the dimensions in the most accessible of this batteries is what makes most of the users of drones, for example, to choose the Li-Po batteries instead.

### Battery power control

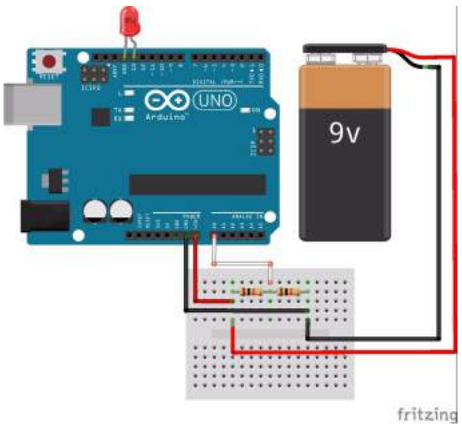


Figure 2.25: Example of a voltage sensor through an analog pin with an Arduino UNO [29].

It is necessary to care thoughtfully on the lifespan of a battery, and so there must be a control system for the detection of the battery drop to the selected battery. And so several considerations must be taken: the minimum and maximum values of the battery in its normal behaviour, the type of input pin from where the microcontroller will measure the drop.

There are two separate ways to measure this voltage drop from a microcontroller: from analog pin and from the digital pin. This means that once selecting the other systems (control, sensing, motion and communication), the remaining pins will determine the level of precision of the control system for the battery drop.

On one hand, if the input pin is analog, the voltage can be measured in a percentage, knowing the equation 2.12 as the mathematical method, where $V_b$ is the battery voltage in percentage, $V_{max}$ is the maximum value of the battery voltage, $V_{min}$ is the minimum value of the battery voltage, $ADC_b$ is the digital value of the analog pin measured through the microcontroller, $ADC_{max}$ is the maximum digital value of the analog-to-digital converter (ADC) in the analog pin, $A$ is the accuracy in decimal that allows the ADC converter, usually 10 bits which equals to 1024 in decimal, noticing that there are two identical resistors that prevent the battery from overcharging the analog pin as seen in figure 2.25, which will reduce the incoming voltage to half and that is why it must be multiplied by 2 in the already mentioned equation.

$$V_b = ADC_b \times \frac{ADC_{max}}{A} \times \frac{R_1 + R_2}{R_2} \times 100\ [\%] \tag{2.12}$$

On the other hand, if the input pin is digital, which will be only in case there are no analog pins available, the measurement will be through the minimum input voltage in the digital pins of the microcontroller, meaning that once a certain drop voltage is reached in the battery, the digital pin will respond directly as an indicator of the arrival to the established limit.

For this, it will be needed a voltage divider for the proper measuring with a microcontroller digital pin, as seen in figure 2.26 with the equation 2.13, where $V_{dp}$ is the voltage measured in the digital pin, $V_b$ is the battery voltage, $R_1$ and $R_2$ are the resistors which creates the voltage divisor.

$$V_{dp} = \frac{R_2}{R_1 + R_2} \times V_b\ [V] \tag{2.13}$$

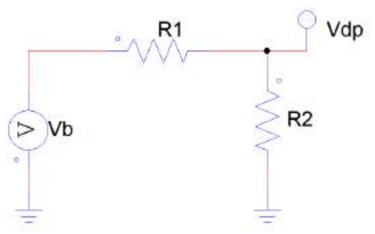


Figure 2.26: Voltage divisor for the battery drop control.

The use of this last equation is for the detection of the minimum value permitted in the drop of the battery, $V_{b_{min}}$, which will coincide with the minimum value of a HIGH value in the digital input of a microprocessor, which will rely on the operating voltage of it. That information can be found in the datasheet of the microprocessors, and in pages 26, 27 and 29.

Therefore, the real question when using a digital pin to detect the voltage drop is the values of both resistors, so that in theory one can be selected at a high value, to prevent from using too much power in this task, and then calculate the value of the other one as in equation 2.14 is reflected.

$$R_2 = \frac{V_{dp_{min}}}{V_{dp_{min}} + V_{b_{min}}} \times R_1 \tag{2.14}$$

This equation will be useful in case there are no analog pins available, selecting the appropriate values of the $R_1$, $V_{dp_{min}}$ and $V_{b_{min}}$.

### 2.1.6 Structure of the mobile robot

There are several ways to design a suitable structure where all the components will be placed and will able the adequate assembly and motion of the entire system.

One way is through the usually known as rapid prototyping method: 3D printing. This is the most useful one when talking about industry, and can be of great use when there is a 3D printing in a school, academy or college were this educational robot can be most used.

Another way is through the design of an assembly system in other more common materials, such as plastic, cardboard or thin wood sheets. This is a way that requires competences in more complex design systems, which can be more an extension to the project than a requirement.

The real and taken option is through rapid prototyping, and this is because if you don't have access to buy a 3D printer, you can always find a place where an open source project like this can be of use and would allow to 3D print this educational robot.

#### Computer-Aided Design and 3D printing

It is important to know how the 3D design works, when talking about dimensions and properties of the materials that will be implemented. When using a computer-aided design (CAD) software, such as AutoCAD, Fusion 360, TinkerCAD, Solid-Works among others, the objects are created in a drawing into a file .draw, which can be selected individually to create a stereolithography file .stl which is described "only the surface geometry of a three-dimensional object without any representation of color, texture or other common CAD model attributes" [30], see figure 2.27 and this is the type of file that will allow a 3D printing software to create the proper file to run the printing.

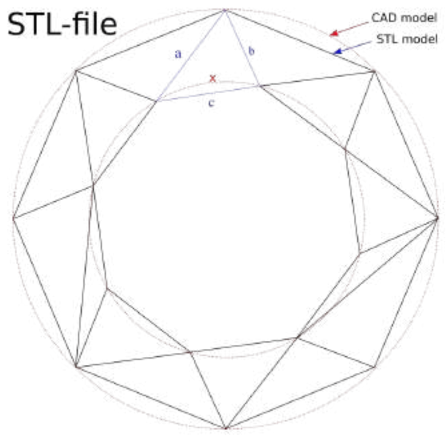
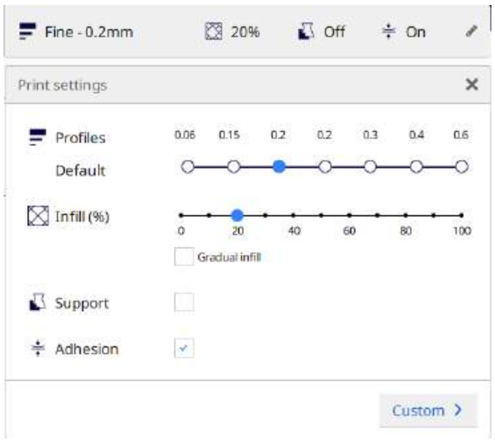


Figure 2.27: Stereolithography vs CAD object [30].

A 3D printer is quite a new tool in the engineering world, and a very useful one, as the name *rapid prototyping* correctly says. This technology allows to precisely create a three dimensional object from strips of melted plastic into the complete structure from a CAD project.

There are several properties to be taken into account when designing a 3D model to get it printed, and those are the ones that will determine the accuracy and expense of a 3D printing.



Fine - 0.2mm | 20% | Off | On

Print settings

Profiles: Default (0.06, 0.15, **0.2**, 0.2, 0.3, 0.4, 0.6)

Infill (%): 20 (0, 20, 40, 60, 80, 100)

Gradual infill: ☐

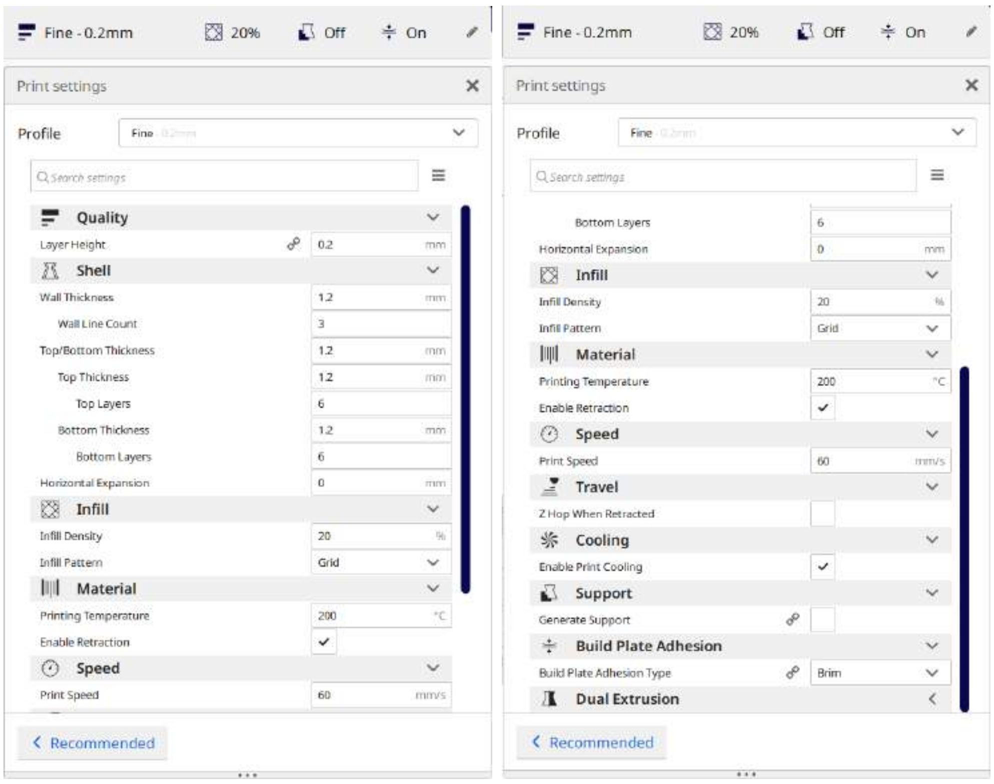
Support: ☐

Adhesion: ✓

Figure 2.28: Basic settings for the BQ Hephestos 2 3D printer in the Ultramaker Cura.

For start into the configuration of a 3D print, the basic properties are detailed in figure 2.28, a capture of one of the most commonly used software that creates the file .gcodes where the specifications for the 3D printer are selected and translated for the printer.

The functioning of this kind of printer is to heat into melting a plastic wire called filament and push it through an extruder in a number of 2 dimension layers that will perform as the final object. As always there are differences between the theoretical or ideal and the practical result, and this will apply to the dimensions of a 3D printed model. For this, we should take care of some of the features seen in figure 2.29, which shows the complete list of configurations in a 3D printer through the Ultramaker Cura software. The most important ones are the layer height and the infill density.



Fine - 0.2mm | 20% | Off | On

Print settings

Profile: Fine - 0.2mm

Search settings

**Quality**
Layer Height: 0.2 mm
**Shell**
Wall Thickness: 1.2 mm
Wall Line Count: 3
Top/Bottom Thickness: 1.2 mm
Top Thickness: 1.2 mm
Top Layers: 6
Bottom Thickness: 1.2 mm
Bottom Layers: 6
Horizontal Expansion: 0 mm
**Infill**
Infill Density: 20 %
Infill Pattern: Grid
**Material**
Printing Temperature: 200 °C
Enable Retraction: ✓
**Speed**
Print Speed: 60 mm/s
**Travel**
Z Hop When Retracted: ☐
**Cooling**
Enable Print Cooling: ✓
**Support**
Generate Support: ☐
**Build Plate Adhesion**
Build Plate Adhesion Type: Brim
**Dual Extrusion**

Figure 2.29: Two captures of the extended settings for the BQ Hephestos 2 3D printer in the Ultramaker Cura.

The layer height is the thickness of the thread in each layer, which will determine the quality of the printing. Whenever it is a lower number, measured in millimeters, it has a better finish or smoother looks, which means: better quality. This also means the printing will take more time to finish, so a higher layer height will make the object more of a draft than a finished product. The more or less allowed layer height will be determined by the type of 3D printer, as an quality specification of them.

The infill density will determine the strength of the object and also will affect the total printing time, as for more infill, will take longer and with a stronger finish, and for less infill, means less time and weaker object. It can also be changed of shape, as seen in figure 2.30, which will provide more or less equality of the distribution of strength and more or less material cost. For instance, one of the preferred patterns for most balanced prints is the octet one.

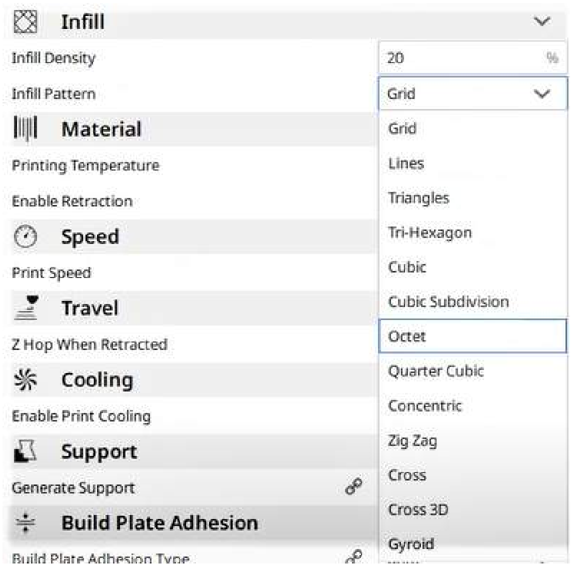

Infill Density 20 %
Infill Pattern Grid

Figure 2.30: Different infill patterns in the Ultramaker Cura.

## 2.2 Analysis for selection of devices and materials

For the selection of the different devices previously mentioned, there must be considered several approaches: quality-cost relation, local and global accessibility and its learning ease within the DIY world. In this section will be discussed the three characteristics for each subcategory within the functional mobile robot requirements.

### 2.2.1 Quality-Cost Relation

The main problem when creating a technological educational environment is the high cost of it. Every commercial educational robot has a design cost, but when doing an open-source robot the design cost is not there, so it becomes most accessible than other commercial ones.

And so, the different choices have to be taken into account for the design, as you can see in tables 2.1, 2.2 and 2.3, where "Local store" is referred to several shops in Tenerife and Spain, such as K-Electrónica and RS Componentes, and "Global" to several online distributors, such as AliExpress and eBay. Each cost is taken from the different cites next to the values, and they are very variable in time as technology is nowadays. These prices were taken on April 13th 2020.

Table 2.1: Table of different costs of microprocessors.

| Microprocessor | Local store | Worldwide store |
|---|---|---|
| Arduino UNO | 6.42€ [54] | 3.35€ [55] |
| Arduino Mega 2560 | 16.05€ [56] | 8.19€ [57] |
| WeMos D1 R2 | 12.84€ [58] | 4.57€ [59] |

The main difference between the three microprocessors is the availability of a larger or shorter number of pins, where the ascendant order is WeMos D1 R2 with the lesser, Arduino Mega 2560 with the higher, and Arduino UNO in between. Also, the WeMos D1 R2 has a great advantage, which is the inclusion of a WiFi module within the processor, unlike the other two.

Table 2.2: Table of different costs of motors.

| Motors | Local store | Worldwide store |
|---|---|---|
| DC motor | 3.74€ [60] | 1.60€ [61] |
| Stepper motor | 3.75€ [62] | 2.39€ [63] |
| Continuous rotation servo motor | 8.57€ [64] | 4.56€ [65] |

The main difference between the three types of motors is the speed, where the fastest one is the DC motor, the slowest one is the stepper motor and in between the remaining one. And the other important difference would be the power consumption, where the lesser power consumption could be with the DC motor, the higher in the stepper, and the continuous rotation servo motor in the middle.

Table 2.3: Table of different costs of sensors.

| Sensors | Local store | Worldwide store |
|---|---|---|
| Ultrasonic sensor (HC-SR04) | 2.68€ [66] | 1.73€ [67] |
| IR Sharp sensor (GP2Y0A21YK0F) | 18.17€ [68] | 4.93€ [69] |

The difference between the sensors are the obvious variation of precision, which is remarkably noticeable that the IR sharp sensor has a better distance measuring and precision than the ultrasonic one. And the only disadvantage of this sensor is its accessibility locally, as the only way to get one is through very specific online shops, at least in Tenerife.

### 2.2.2 Local & Global Accessibility

It is fundamental that all the materials are available wherever the project is used to teach, so all of the devices, components and utilities must be able to be found both in local places, for example, Tenerife, and worldwide. This feature must be considered when selecting time.

### 2.2.3 DIY Learning Ease

When talking about educational robots, it is usually found that the electronics are barely described or taught, not like programming or mechanical designs. This is why the DIY system may be useful to understand a little bit of electronics while building the robot.

It must also be considered the easiness of the robot use when learning, and that is why a more accessible design should be when the device controlling the robot is most common, for example, an smartphone.

### 2.2.4 Decision

When creating a prototype like this, the first step is to set an approach on which devices to use and just start creating. So when creating an accessible and easy-to-use educational robot, the devices decided to use are the ones that allow the use of a mobile phone with access to a WiFi zone to perform as the controller as well as the possible expansion to a swarm system: WeMos D1 R2 as the microcontroller board, two stepper motors for the movement of the robot controlled through the arduino motor driver shield and the IR sharp sensor connected to a micro servo motor, powered by a Li-Po battery. This will allow the easiest way to connect to the robot through a web server for the students to program the basic functions in robotics.

It is necessary to say that although the first decision of the sensor was using the ultrasonic sensor, at certain point it was changed due to the higher quality of the IR sensor, although it can also be implemented as the motion system is prepared for the adaptation in case there cannot be bought the IR sharp sensor.

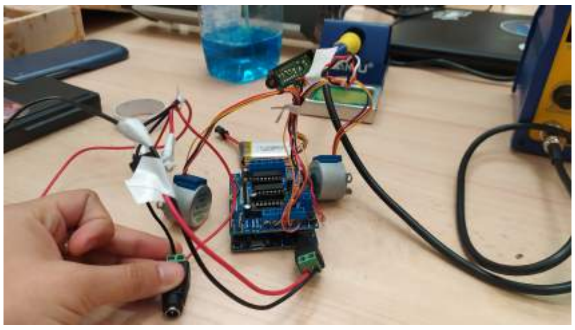


Figure 2.31: Basic components of the mobile educational robot.

# Chapter 3

# Materials & Methods

In this chapter the used methodology will be described, as much as the materials needed, for the design, prototyping process and implementation of the educational robot.

## 3.1 Electronics system design

It was the first part of the process, to create an appropriate electronic system within the connection of the different devices, so that they would perform the basic functions of a mobile robot. At start, the thought of a driver module connecting the stepper motors with the microprocessor was implemented, but then there were a few problems about the availability of digital pins so it had to be changed into a more sophisticated solution: the use of the arduino motor driver shield.

### 3.1.1 Arduino Motor Driver Shield modifications

It is the easiest way to control multiple motors in a DIY project as it is very easy to access to and there is no need for much changes within the shield itself for the proper use in the WeMos D1. This is because this microcontroller board has several pins interconnected with each other, as seen in figure 2.6, which leaves only 9 available digital pins, from which 6 digital pins must be used for the motor control system.

Originally, the arduino motor driver shield has to be attached to 8 different digital pins in order to move the 3 different types of motors, but as there are few pins from were to choose and only two types of motors needed, prioritizing the steppers, only 6 digital pins are necessary for the programming from the WeMos D1 R2: pins D7, D3, D8, D2, D5 and D6 for the two stepper enables, the motor latch, the motor clock, the motor enable and the motor data respectively.

For this, three modifications had to be made into the shield, cutting the pin 12 and welding it to the pin 10, cutting the pin 11 and welding it to the pin 3 and cutting the pin 6 and welding it to the pin 5.

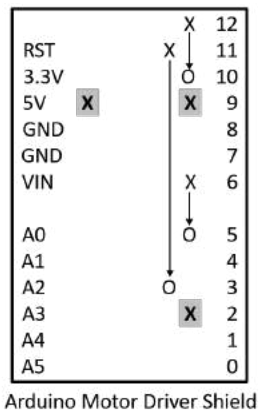


Figure 3.1: Cuttings and weldings made for the Arduino Motor Driver Shield.

Other cuttings had to be performed from the shield in order to attach the other electronic systems into the microcontroller, as the sensing system, the motion system of the sensor and the battery drop control system. The pins 2 and 9, as well as cutting the 5V pin off to create the voltage drop for the proper powering of the H-bridges and shift register of the shield (go back to 2.1.2 to more clarification), the three had to be removed, as seen in figure 3.1. For this, a 1N4002 diode for the proper voltage drop (2.1.2) has to be connected from the 5V bridge between the WeMos and the "5V" pin of the powering system of the shield, as seen in figure 3.3, where X means which pin to cut and O where to weld a bridge from the cut pin and the bolt and highlighted X means which pin to remove.

Once the cuttings are done, the connections between the WeMos D1 R2 and the Arduino Motor Driver Shield V2 are perfectly matched. And this is when the steppers are connected to the shield, as seen in figure 3.2. In the next figures the steppers will be omitted in order to facilitate the better comprehension of the figures.

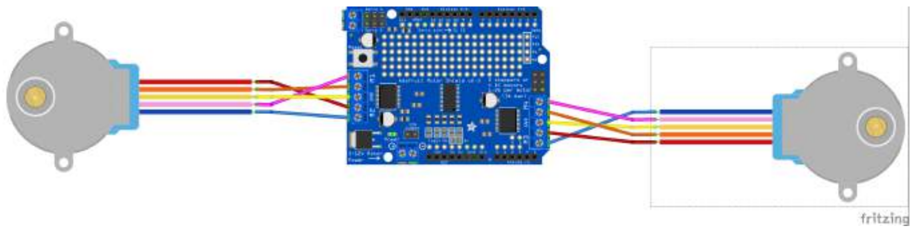


Figure 3.2: Connections between Arduino Motor Driver Shield and the steppers.

See that the steppers wires are intertwined to introduce the appropriate signals, so the connections are like in table 3.1, being W1, W2, W3, W4 and W5 the five wires from the left to the right of the 28BYJ-48 stepper and M1, M2, M3, M4 and M5 being the motor connectors of the shield from the bottom to the top in both sides (with a horizontal view of the shield, as in figure 2.14).

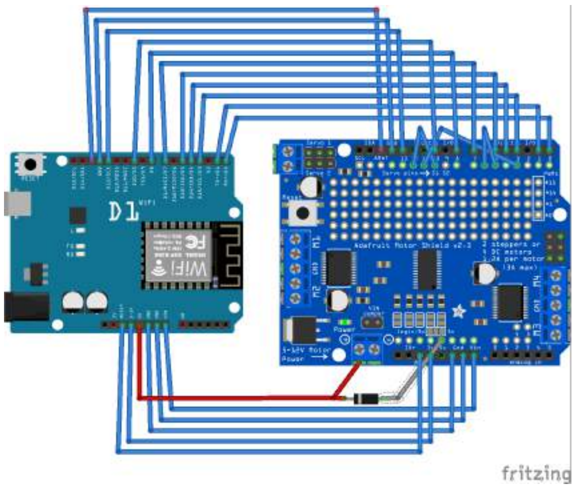


Figure 3.3: Connections between WeMos D1 R2 and Arduino Motor Driver Shield

Table 3.1: Connections of the stepper motors to the arduino motor driver shield.

| Stepper wire | W1 | W2 | W3 | W4 | W5 |
|---|---|---|---|---|---|
| Motor Shield Connector | M1 | M5 | M3 | M4 | M2 |

### 3.1.2 Sensor attachment

The next part was to check that the sensor was properly connected to the microcontroller, which has no more science than connecting its three pins, SIGNAL, GND and Vcc into the extra A0 and GND pins of the arduino motor driver shield, and a 5V pin of the WeMos D1 R2 (through a breadboard), respectively, as seen in figure 3.4.

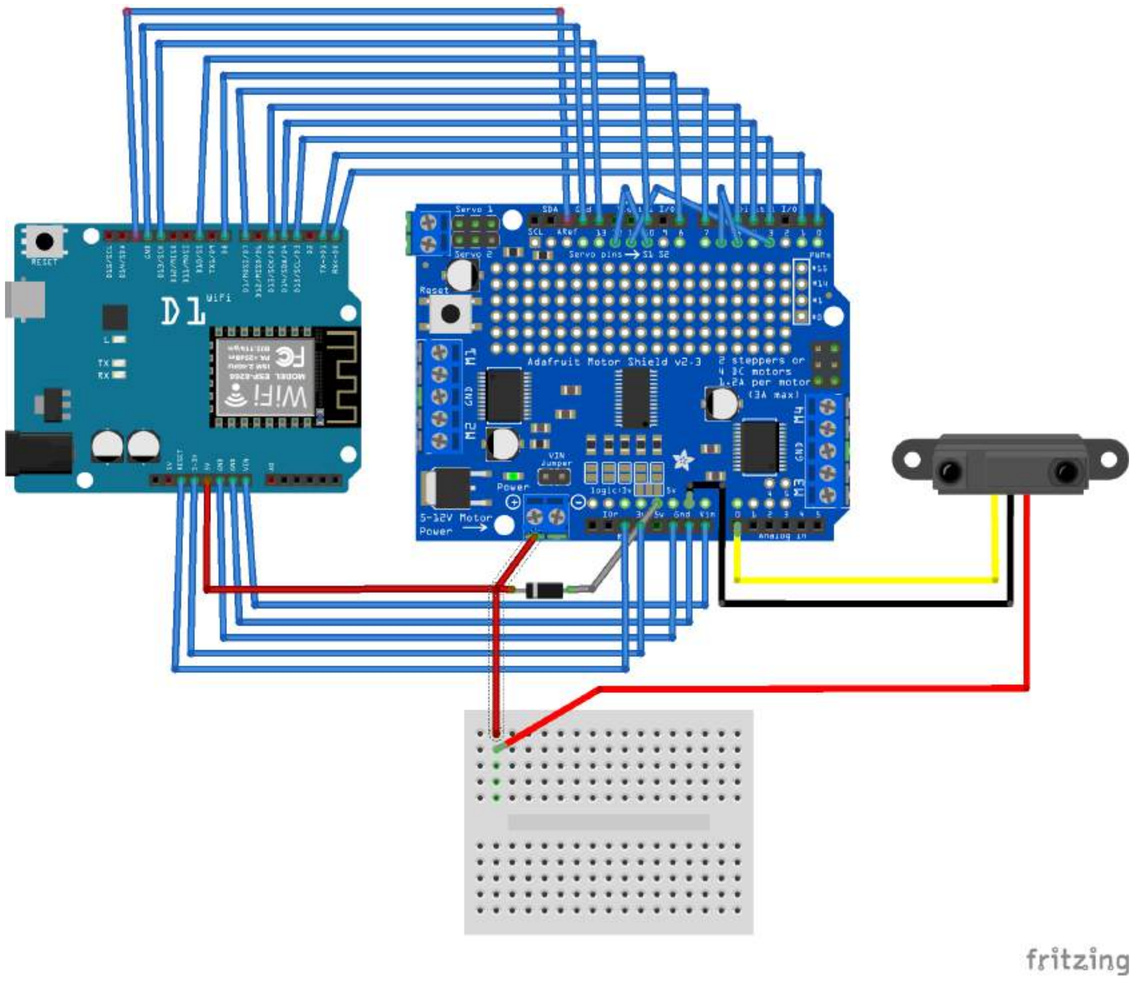


Figure 3.4: Sensor's connections to the system.

For this, the wires had to be welded into the sensor's pins and then male-female connectors as sockets were implemented for an easier disconnection of the sensor, as seen in figure 3.5.

After that, the micro servo motor had to be incorporated into the connections for the movement of the sensor, that is connecting its three pins, Vcc, GND and SIGNAL into a 5V pin and a GND pin both of the WeMos D1 R2 and connected to a breadboard as well as a spare digital pin of the microprocessor, respectively, as seen in figure 3.6.

For this, as well as it was done for the Sharp sensor, a socket had to be incorporated for an easier connection and disconnection of the micro servo motor, as seen in figure 3.7.

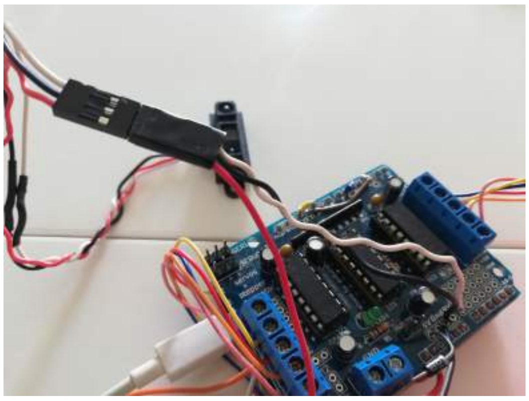


Figure 3.5: Socket for the sensor's connections.

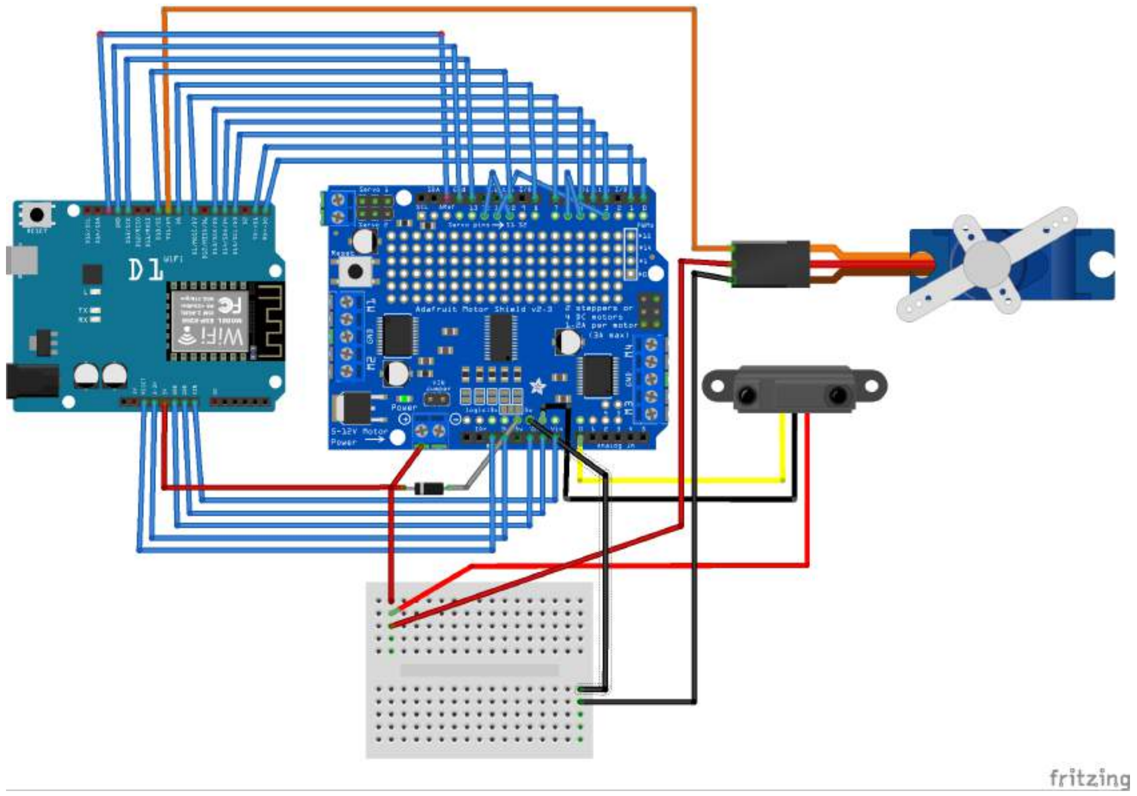


Figure 3.6: Sensor's movement connections to the system.

This will facilitate the acknowledge of the different aspects of electronics once the mechanical design is incorporated an the assembly has to take place. Also, it will allow the disconnection of the sensor and the micro servo motor to be faster if one or both need to be changed. This may also be useful in case a different sensor wants to be implemented instead of the Sharp IR sensor. For instance, the ultrasonic sensor could be implemented easier if connections and disconnections DuPont sockets are incorporated.

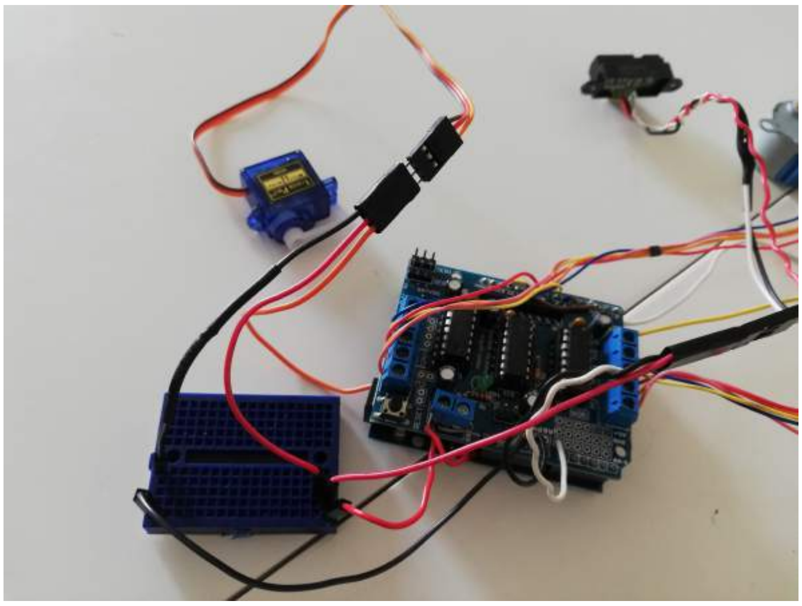


Figure 3.7: Socket for the micro servo motor's connections.

### 3.1.3 Power Connections and Battery Drop Control System

Once this is done, the connections of the powering system must be included, as the switch, the power male and female jack connectors and the battery are placed in the system. For this, the female jack connector for the charging process is attached directly to the battery and a bridge is created to connect in parallel the male jack connector and a switch in series for control of the powering of the entire system, as seen in figure 3.8.

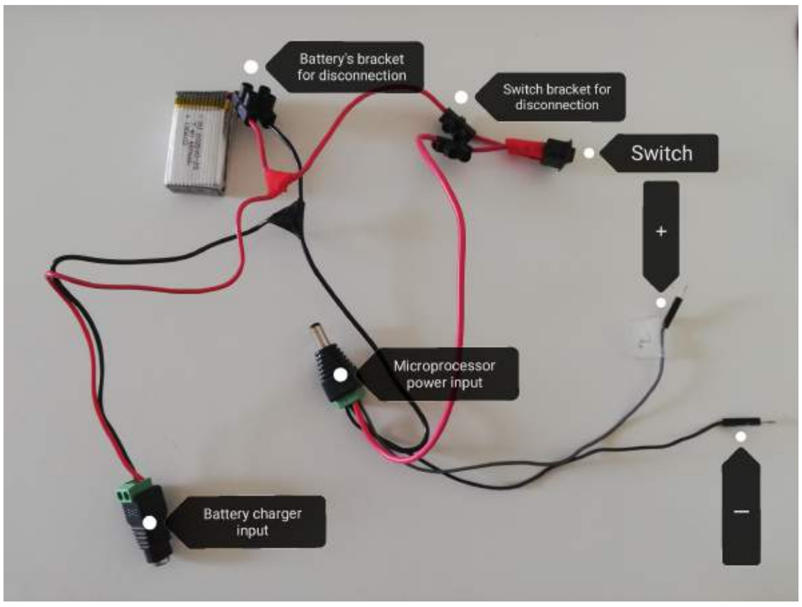

Battery's bracket for disconnection
Switch bracket for disconnection
Switch
+
Microprocessor power input
Battery charger input
−

Figure 3.8: Battery's connections.

To this male connector are attached two wires that will introduce the battery voltage to the voltage drop sensor, with the easy system first seen in figure 2.26 and now shown in figures 3.8 and 3.9, where the first resistor on the left is of 220 $k\Omega$, the second resistor is a combination of resistors giving an approximate value of 148.5 $k\Omega$ which can be as a sum of resistors from the available pack or a potentiometer as in the figure already mentioned or as in the prototype (see figure 3.8).

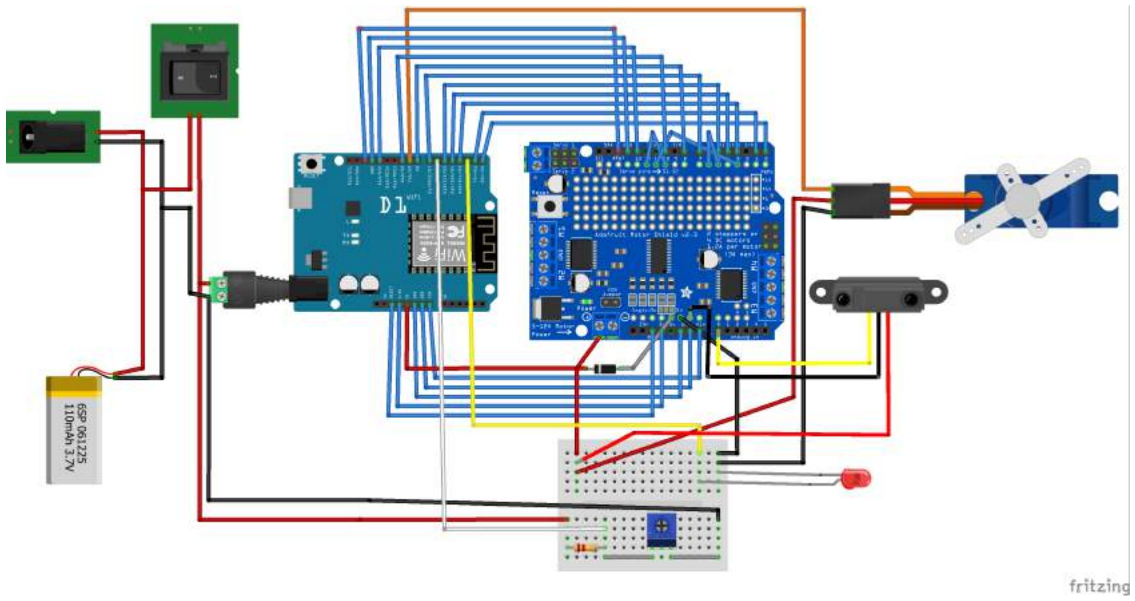


Figure 3.9: Battery drop control system's connections.

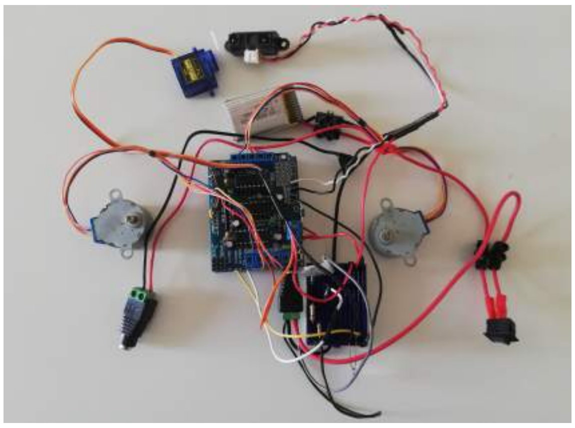


Figure 3.10: All of the connections and components.

In between both resistors, a wire is connected to pin D6 of the WeMos D1 R2 for the "measuring" of the voltage. It is also included a LED (the colour can vary depending on the availability) with the purpose of warning when the battery is too low and must be charged (which will be explained in the programming chapter), and it is connected to the pin D9 of the WeMos D1 R2.

Once all of this is connected, the electronic design is finished and the other elements have to be prepared for the proper assembly of the mobile robot, which will be similar to the distribution seen in figure 3.10.

## 3.2 Structure design

A fundamental aspect of the utility of the mobile robot is achieving an appropriate design for the assembly of all the components, and for this the 3D design has been used into the creation of the different mobility and structure parts of the robot.

### 3.2.1 Chassis and Mobile System

The first part was to create the chassis of the mobile robot, as the elements are jointed together in a single structure, which must be precise, tough and light enough for the mobile purpose. There will be three support points of the movement, two wheels specialized for the used steppers, and another one that will be explained.

For this, the wheels were searched in an open-source platform called Thingiverse as the features were adequate to this mobile robot, and they were 3D printed and adapted for the use, with several attempts, as seen in figure 3.11, and using two elastic bands in each wheel as tyres.

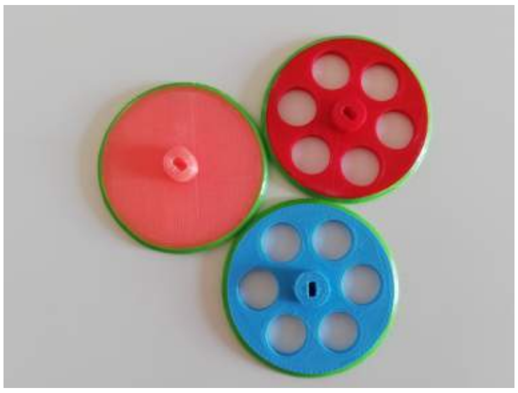
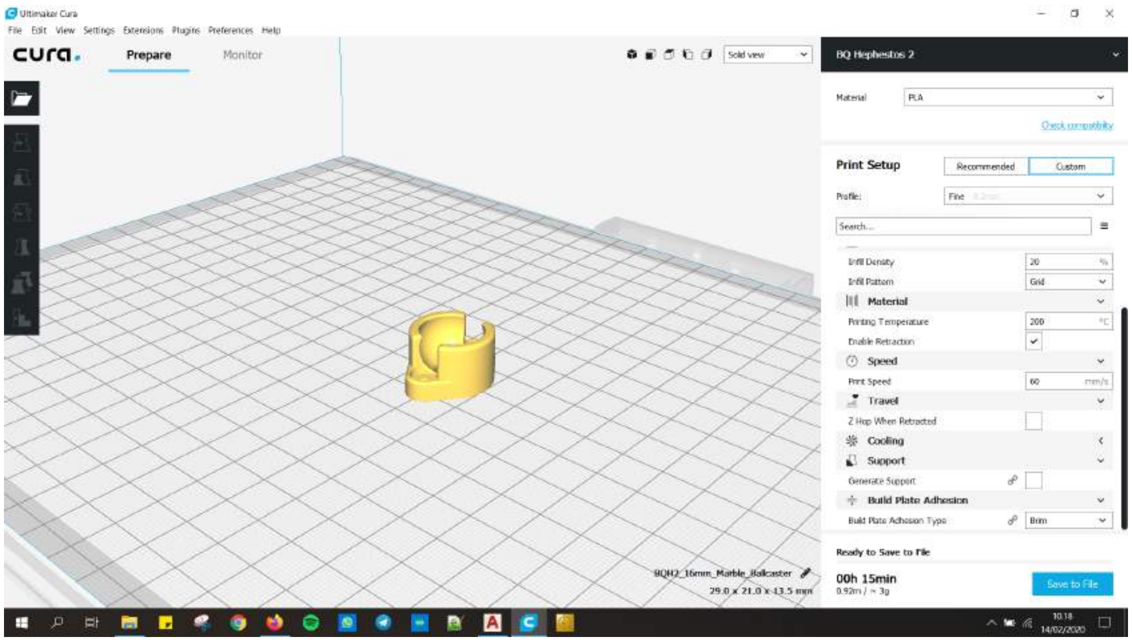


Figure 3.11: 3D printed wheels from Thingiverse [31].

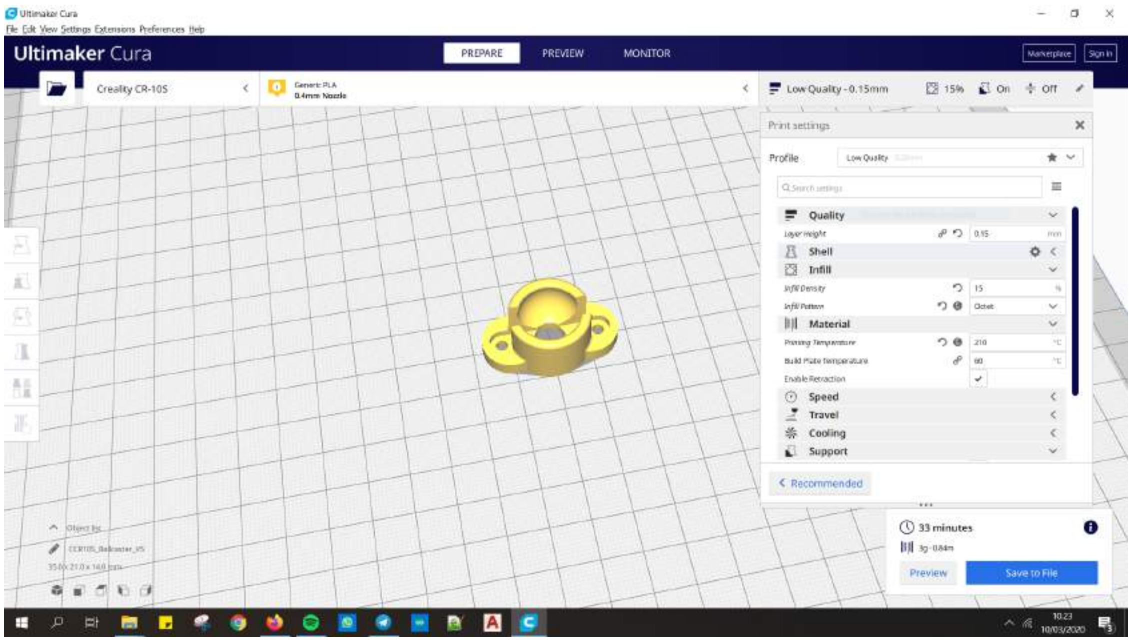


BQ Hephestos 2
Material: PLA
Print Setup: Custom
Profile: Fine
Infill Density: 20 %
Infill Pattern: Grid
Material
Printing Temperature: 200 °C
Enable Retraction: ✓
Speed
Print Speed: 60 mm/s
Travel
Z Hop When Retracted: ☐
Cooling
Support
Generate Support: ☐
Build Plate Adhesion
Build Plate Adhesion Type: Brim
Ready to Save to File
00h 15min
0.92m / ~ 3g
BQH2_16mm_Marble_Ballcaster
29.0 x 21.0 x 13.5 mm
10:18 14/02/2020

Figure 3.12: 3D model ball caster from Thingiverse ready to be printed [32].

The third support point is an element attached to the chassis for the enabling of the movement in any direction is inspired in the use of a marble as an omnidirectional wheel, therefore an open-source object was looked in Thingiverse and 3D printed (see figure 3.12) and a similar one was created through AutoCAD 3D.

There are up to 6 versions of this prototype of a "ball caster", with different modifications in its wideness, its thickness on the ball's walls, and other small features to enable the assembly to the chassis (see appendix B). The most useful one is the version 5 (see figure 3.13).



Low Quality - 0.15mm 15% On Off

Print settings

Profile: Low Quality

Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 210 °C
Build Plate Temperature: 60 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support

33 minutes
3g · 0.84m

Figure 3.13: 3D model of the ball caster version 5 ready to be printed.

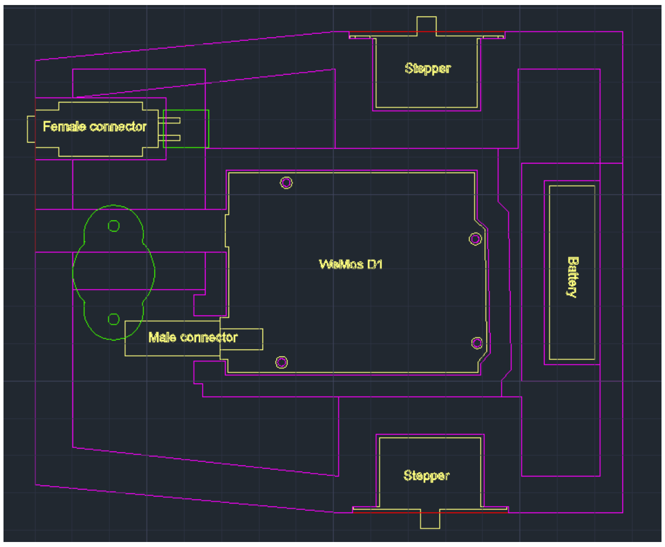


Figure 3.14: 3D modelling of the chassis with the electronic components included.

The chassis had to be designed taking into account the different electronic elements, so that the space between them had sufficiently strong walls and also space for the wires. This is why the distribution of the elements are as seen in figure 3.14.

There was a first prototype of the chassis of the mobile robot, which was 3D printed and tested to resolve several problems that it may contain. The methodology followed was to create several individual objects for the walls and floor of the chassis, as well as other objects acting as holes for the screws (for the assembly), everything detailed in different layers in AutoCAD 3D, to finally create a copy of all the pieces and join together all of them into one object and round all the borders to take care of the wires (see figure 3.15).

The time and specifications of the printing had to be adjusted to the 3D

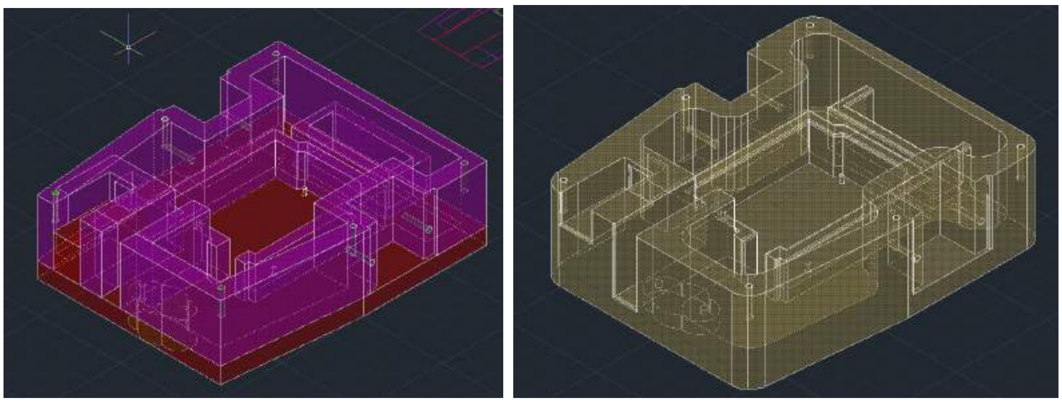
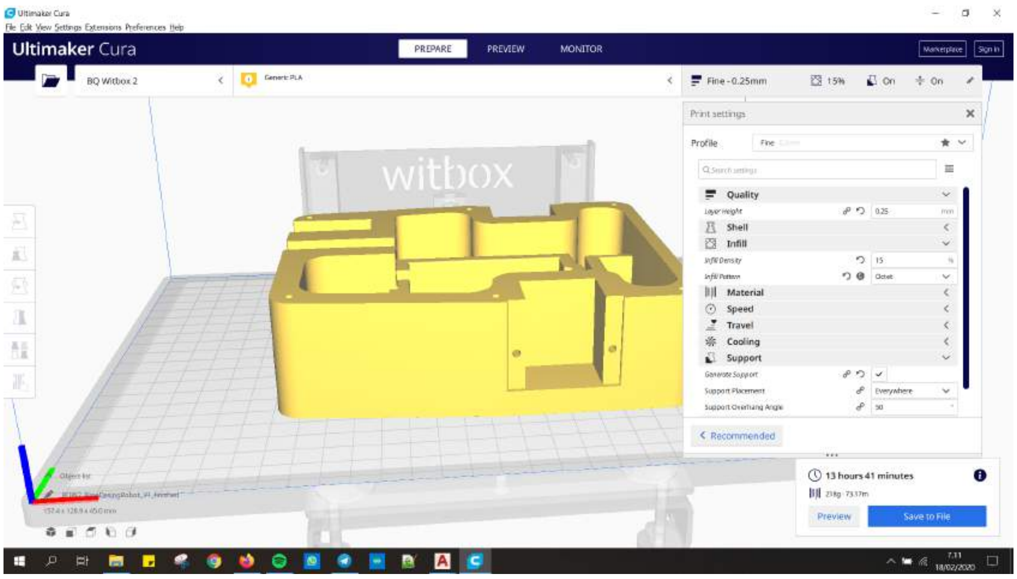


Figure 3.15: 3D model of the chassis version 1, before the joint on the left and after on the right.

printer that was available, a BQ WitBox 2, as seen in figure 3.16.



Fine - 0.25mm 15% On On

Print settings

Profile Fine

Quality
Layer Height 0.25 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Speed
Travel
Cooling
Support
Generate Support ✓
Support Placement Everywhere
Support Overhang Angle 50

13 hours 41 minutes
218g · 73.17m

Figure 3.16: 3D model of the chassis version 1 ready to be printed.

The second model was after noticing several aspects to improve: firstly, a hole had to be included for the LED of the battery control system; secondly, the space for the connection of the USB to the WeMos D1 R2 had to be expanded; thirdly, the walls were reduced for a better use of resources and to make the structure lighter; fourthly the space for the female power jack connector (with the purpose of charging the battery) had to be thinner in the exterior wall and wider on the back for a better assembly; fifthly, a hole had to be added for the assembly of a cover on the space for the USB connection; sixthly, the holes for the screws on the assembly of the ball caster had to be longer; and finally, the space for the placement of the microprocessor had to be lifted so that the assembly system would be holes in the 3D printing and screws instead with small cylinders emerging from the surface of the 3D printing (which would break very easily).

The printing of the version 2 was done and then noticed that the space for the female power jack connector and the hole for the cover on the USB space both had to be slightly modified to match the assembly adequately into version 2.1. See figure 3.17 for a better view of the final version of the chassis.

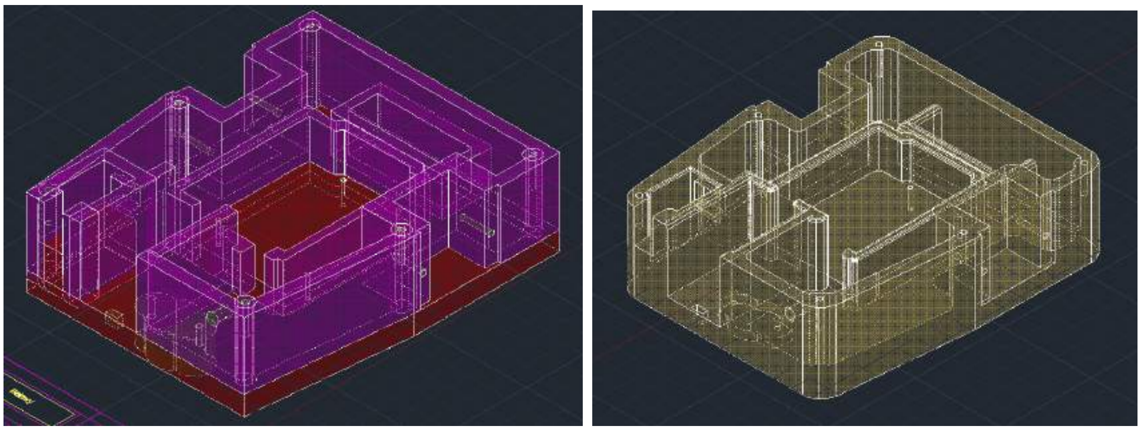
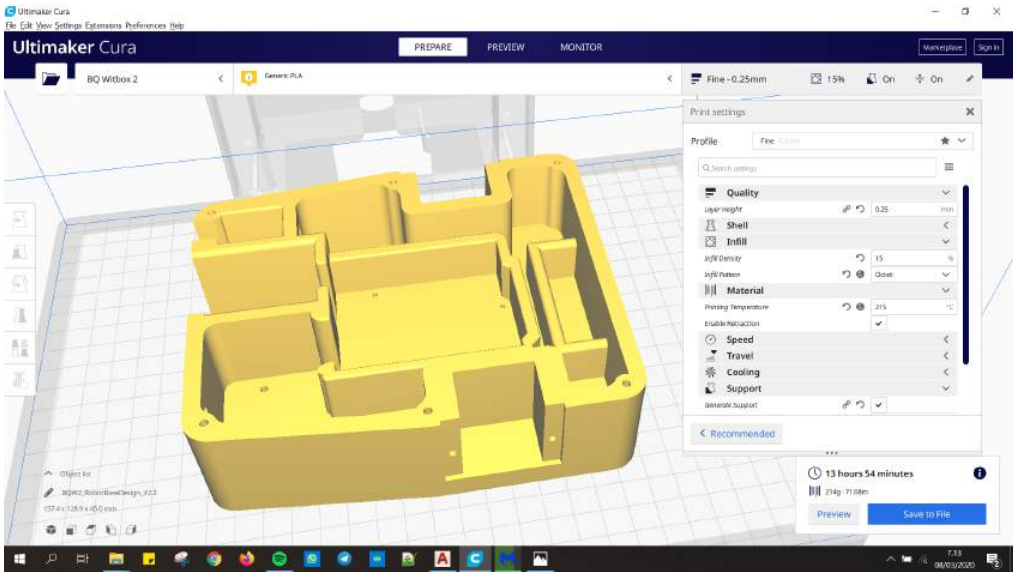


Figure 3.17: 3D model of the chassis version 2.1, before the joint on the left and after on the right.

Just as in version 1, the time and specifications of the printing had to be adjusted to the BQ WitBox 2, as seen in figure 3.18.



BQ Witbox 2

Generic PLA

Fine - 0.25mm 15% On On

Profile: Fine

Quality
Layer Height 0.25 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

13 hours 54 minutes
214g · 71.68m

BQW2_RobotBaseDesign_V2.2
157.4 x 128.9 x 45.0 mm

7:13
08/03/2020

Figure 3.18: 3D model of the chassis version 2.1 ready to be printed.

### 3.2.2 Hubcap and Sensor's Motion System

The hubcap is the term usually taken for the cover on a car, which can be used in this case. The sensor's motion system is a unique design created for the adaptation of the sensor to the electronic connections into the hubcap, which allows the adequate sensor's motion for the uses of this mobile robot.

The first part to be design in this subsection was the cover and attachment of the sensor into the micro servo motor, as this model brings several but standardized small pieces for its use in attachment between the motor and any mechanism, as seen in figure 3.19.

So a faster between the micro servo motor and the sensor was designed, with several versions into the most useful one, which provides the sufficient adhesion to the sensor and a perfect attachment to the micro servo motor, as seen in figure 3.20. There are up to 5 versions of this fastener. For more information, see appendix B.

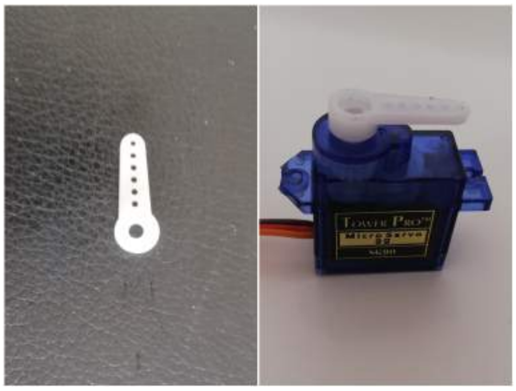


Figure 3.19: Standardized attachment piece used in the micro servo motor.

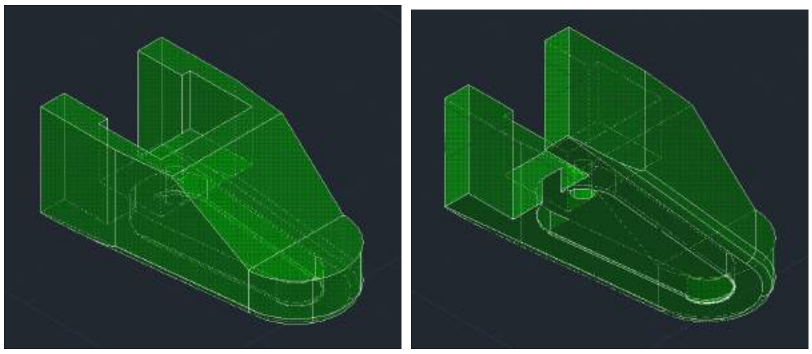


Figure 3.20: Fastener for sharp sensor to micro servo motor version 5, from the front and up to the left and from the back and down to the right.

There was needed another fastener to attach the sensor to the fastener seen before, and also to protect the wires of the sensor (it may me called cover-fastener), as seen in figure 3.21, which is the fifth version of this object. To see all the other version, check appendix B.

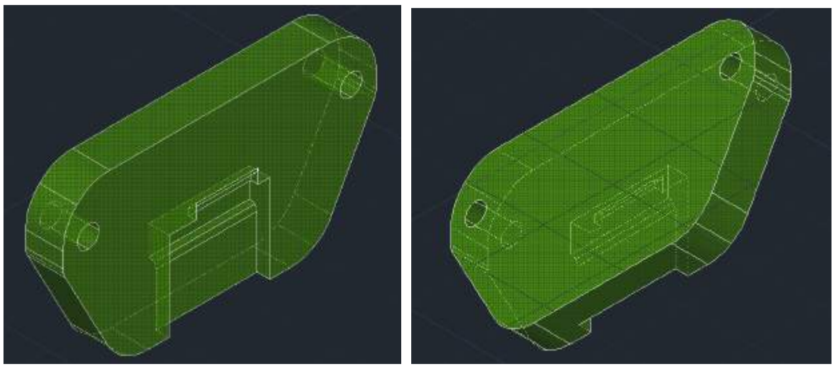


Figure 3.21: Cover-fastener for sharp sensor to micro servo motor version 4, from the front and up to the left and from the back and down to the right.

Both fastener are connected to another necessary object to allow the movement of the sensor, which is the micro servo fastener to the hubcap of the robot structure. This hubcap fastener has up to 7 versions, both 6 and 7 useful. See figure 3.22 to observe the design, which contains the hole of the screw that attaches the first fastener to the micro servo motor, a rounded gap for the movement of the sensor's wires when turning and 4 screw holes for the assembly with the hubcap.

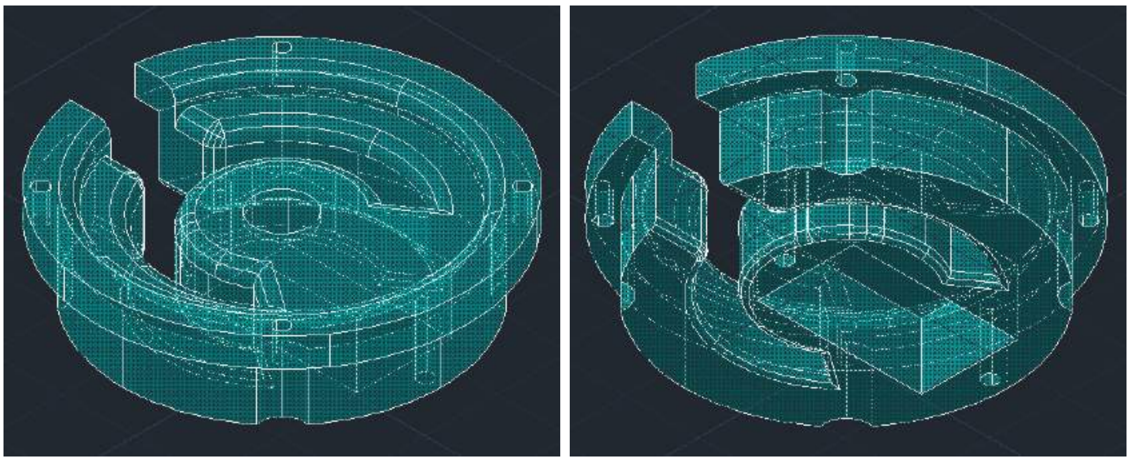


Figure 3.22: Fastener for micro servo motor to the hubcap version 7, from the front and up to the left and from the back and down to the right.

All these three fasteners are assembled to each other as seen in figure 3.23, where the separate pieces are seen on the left, then the first fastener is assembled with a screw in the center of the joint, and finally the cover-fastener adjusted to the motion system.

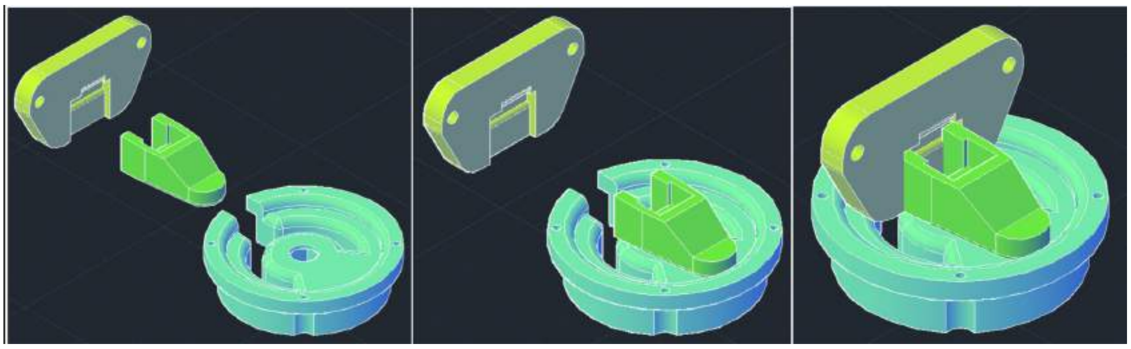


Figure 3.23: Assembly of the three fastener of the sensor's motion system.

Once done this, the hubcap had to be designed, caring for the position of this motion system as well as the assembly with the chassis. It had to allow the insertion of the switch with its specific measures, which was a rectangular gap on the back of the hubcap. And finally, the name was 3D inserted on the front surface of this final object, the hubcap, with a first distribution as seen in figure 3.24.

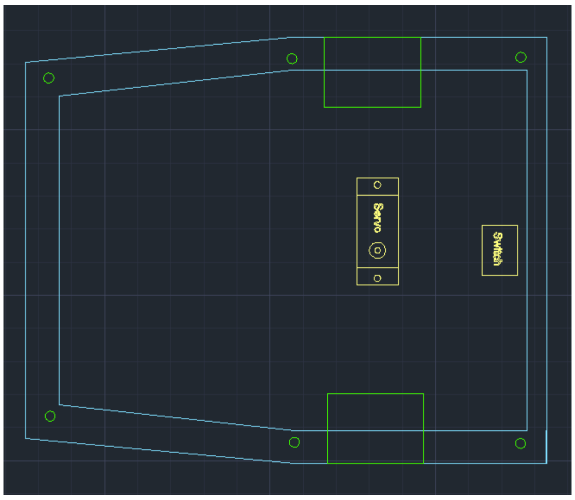


Figure 3.24: First distribution of elements in the hubcap.

Two versions were made, as well as what happened with the chassis because several measures had to be edited: the gap for the switch had to be slightly shorter in two of its dimensions, the thickness of the walls had to be reduced (just as in the chassis) and the name went from a slight incision or slope in the surface to a gap in that wall.

The first version was using the first approximation of the distribution seen above, and in the second version the methodology was just to implement the changes explained in the previous paragraph. See figures 3.25 and 3.26 to see both versions, and appendix B for more detail.

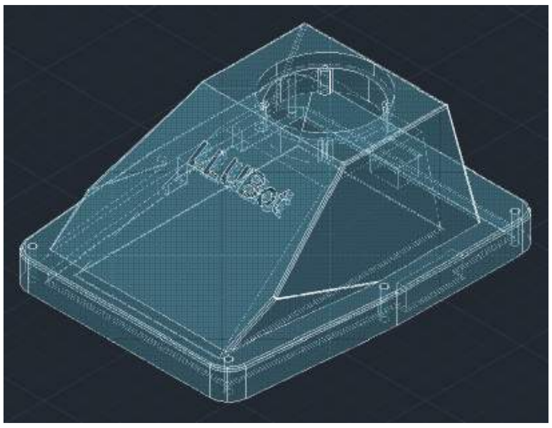


Figure 3.25: Hubcap version 1 in AutoCAD 3D.

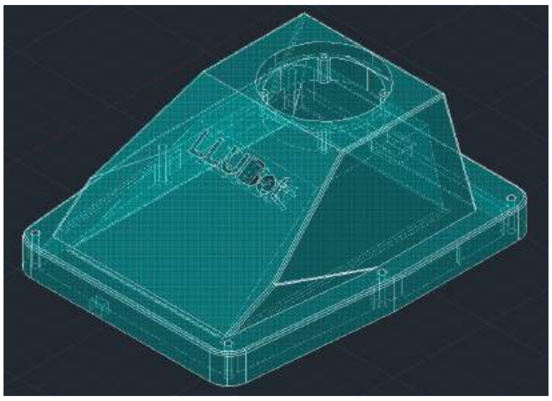
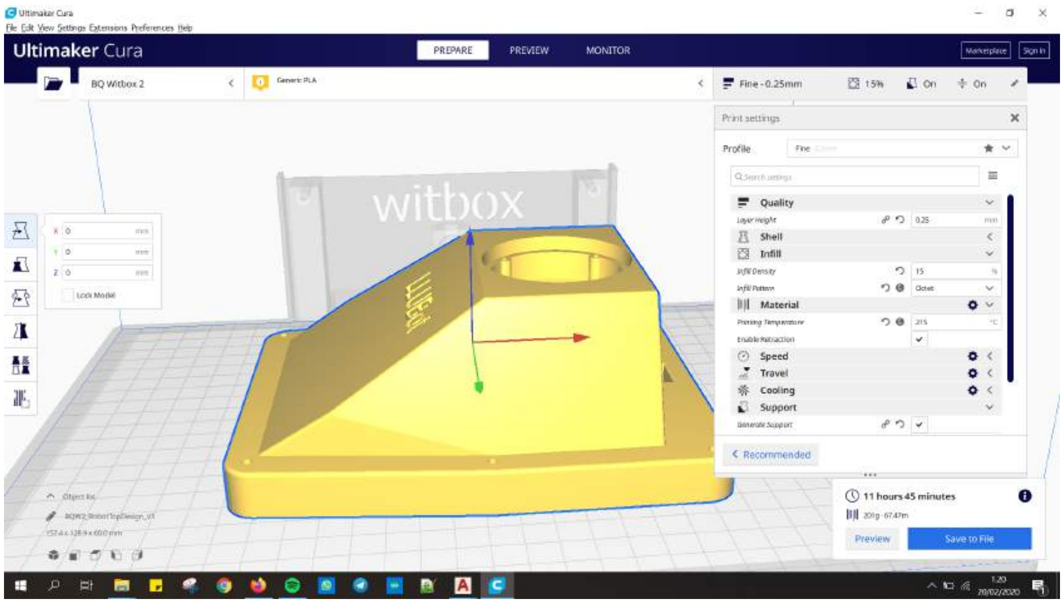


Figure 3.26: Hubcap version 2 in AutoCAD 3D.

The time and specifications of the printing had to be adjusted to the 3D printer that was available, a BQ WitBox 2, as seen in figure 3.27.

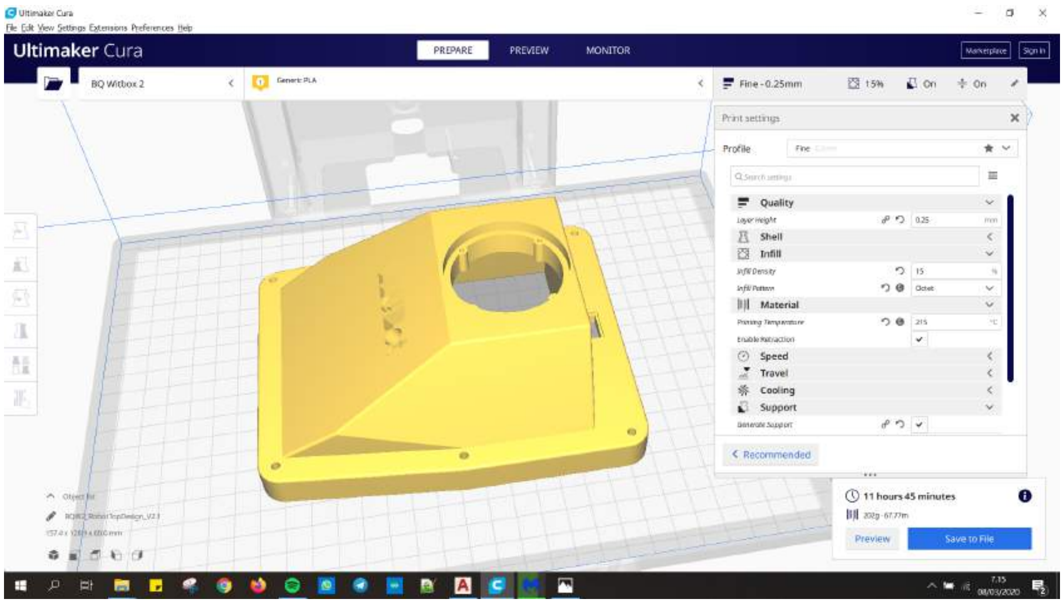

Ultimaker Cura — BQ Witbox 2 — Generic PLA — Fine - 0.25mm — 15% — On — On

Print settings
Profile: Fine
Quality
Layer Height: 0.25 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

11 hours 45 minutes
201g · 67.47m

Figure 3.27: 3D model of the hubcap version 1 ready to be printed.

Just as in version 1, the time and specifications of the printing had to be adjusted to the BQ WitBox 2, as seen in figure 3.28.


Ultimaker Cura — BQ Witbox 2 — Generic PLA — Fine - 0.25mm — 15% — On — On

Print settings
Profile: Fine
Quality
Layer Height: 0.25 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

11 hours 45 minutes
202g · 67.77m

Figure 3.28: 3D model of the hubcap version 2 ready to be printed.

### 3.2.3 Final Touch: WiFi Cover

A final piece to introduce in between both the chassis and the hubcap (version 2) is the WiFi cover, which is a small plain piece which allows the opening and closing of the gap for the connection of a USB for the detection of the IP direction of the web server created with the WeMos D1 R2. See the design in figure 3.29

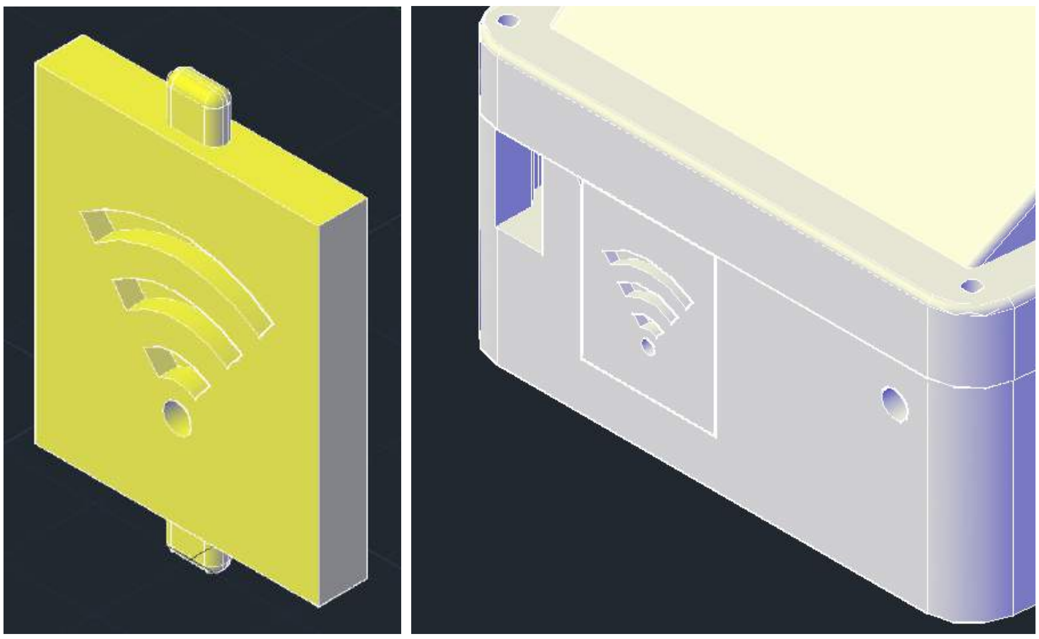


Figure 3.29: 3D model of the WiFi cover and its placement.

### 3.2.4 Assembly

Once designed all of these object, the next part is to illustrate the assembly of all of the structure, as seen in figures 3.30 and 3.31, where the first one shows the first functional version and the second one the second and final prototype.

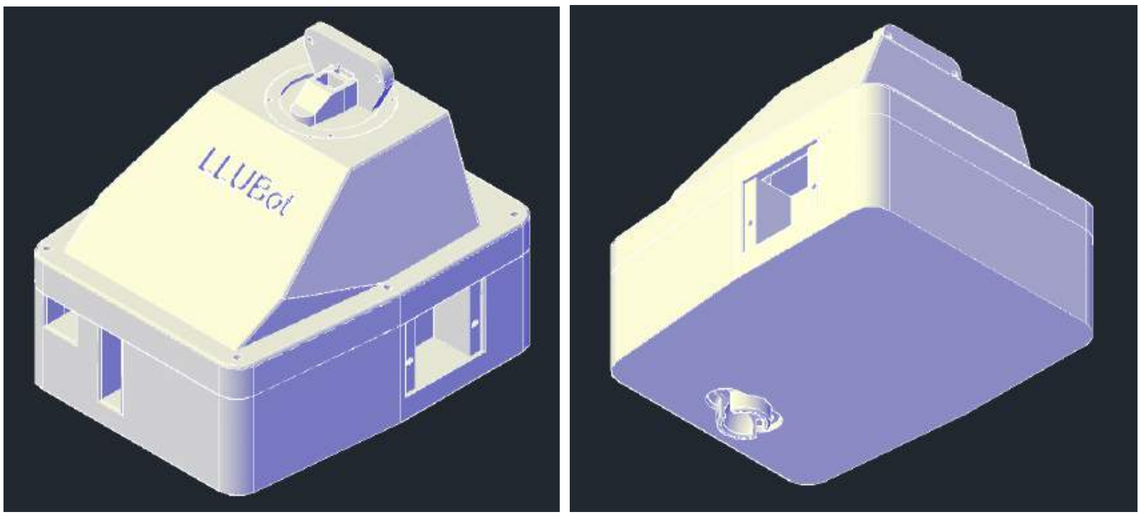


Figure 3.30: 3D model of the assembly version 1, from the front and up to the left and from the back and down to the right.

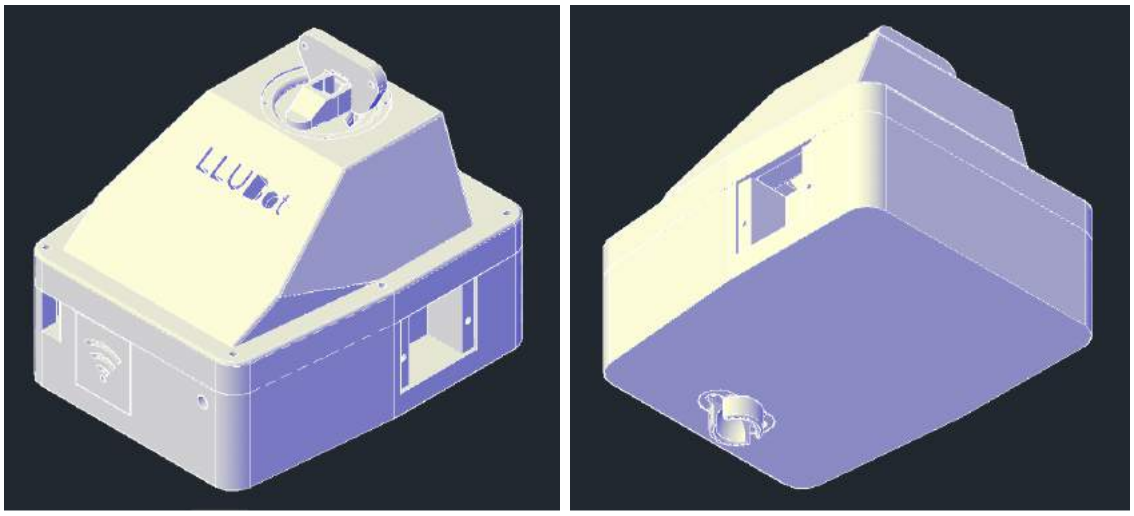


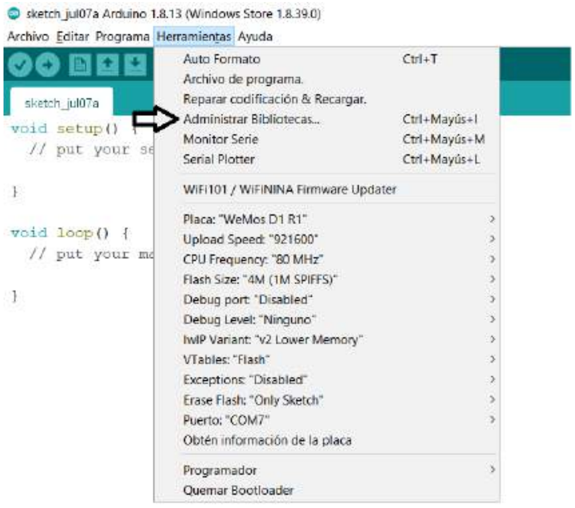
Figure 3.31: 3D model of the assembly version 2, from the front and up to the left and from the back and down to the right.

## 3.3 Design of the Program

In this section, the different aspects of the programming area are discussed, so that the desired behaviour is achieved.

### 3.3.1 Configuration

Firstly, the Arduino IDE had to be configured for the use of the WeMos D1 R2 into the programming system. For this, the board had to be introduced into the system, selecting the WeMos D1 R1 as the microcontroller board to use as seen in figure 3.32. It's the easiest way to configure the microcontroller board, and it's the way it was done. All the other configurations needed are explained later on.

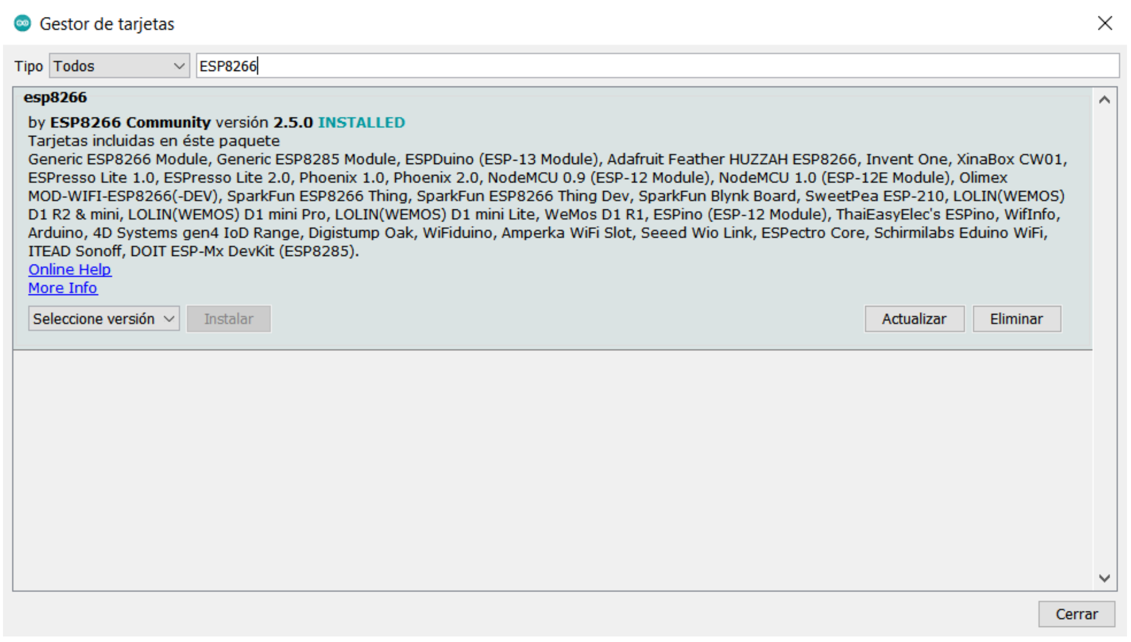

sketch_jul07a Arduino 1.8.13 (Windows Store 1.8.39.0)
Archivo Editar Programa Herramientas Ayuda
sketch_jul07a
Auto Formato Ctrl+T
Archivo de programa.
Reparar codificación & Recargar.
Administrar Bibliotecas... Ctrl+Mayús+I
Monitor Serie Ctrl+Mayús+M
Serial Plotter Ctrl+Mayús+L
WiFi101 / WiFiNINA Firmware Updater
Obtén información de la placa
Programador
Quemar Bootloader

```
void setup() {
  // put your se

}

void loop() {
  // put your ma

}
```

Placa: "WeMos D1 R1"
Upload Speed: "921600"
CPU Frequency: "80 MHz"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "Ninguno"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Disabled"
Erase Flash: "Only Sketch"
Puerto: "COM7"

Figure 3.32: Configuration of the WeMos D1 R2 into Arduino IDE.

If it doesn't come within the system, select the board manager and add the package to the software as seen in figure 3.33 (see figure 3.32 to locate the board manager).


Gestor de tarjetas

Tipo Todos ESP8266

**esp8266**
by **ESP8266 Community** versión **2.5.0** **INSTALLED**
Tarjetas incluidas en éste paquete
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
Online Help
More Info

Figure 3.33: Board manager in Arduino IDE.

Remember to always select the port in which your controller board is connected so the software can upload properly.

We also had to know the proper nomenclature to use of the pins in the micro-controller board, which can be seen in table 3.2. It's also included the nomenclature in case the WeMos D1 R2 is not available and the one that is available is the WeMos D1 R1.

Table 3.2: Corresponding pins of the different WeMos D1 into the Arduino IDE.

| Pin in Arduino IDE | Pin in WeMos D1 R1 | Pin in WeMos D1 R2 |
|---|---|---|
| "D0" | D0 | RX |
| "D1" | D1 | TX |
| "D2" | D2 | *D0 |
| "D3" | D3 | D1 |
| "D4" | D4 | D2 |
| "D5" | D5 | D5 |
| "D6" | D6 | D6 |
| "D7" | D7 | D7 |
| "D8" | D8 | D3 |
| "D9" | D9 | D4 |
| "D10" | D10 | D8 |

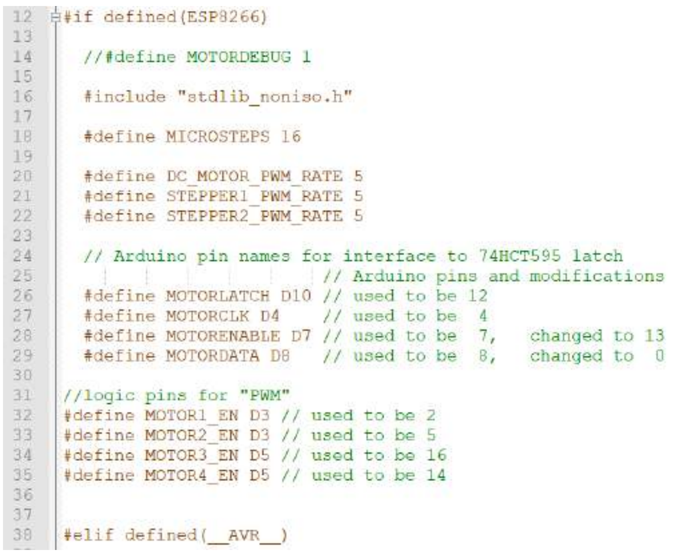
Once this is done, the rest of the coding is as shown as follows, where all the codes will be contained within the appendix C.

### 3.3.2 Position movement of the robot

This section was chronologically started before the structure design, once the microprocessor and the motors were chosen. For start, a quick and draft code was created for the movement of the steppers, as seen in code C.2.1, using an open-source library special for the use of the Arduino motor driver shield in the WeMos D1 R2 [70] and then modified into the one seen in appendix B in library C.1.1. This three files (code C.2.1 and libraries C.1.1 and C.1.2) would allow the movement of the robot forwards.

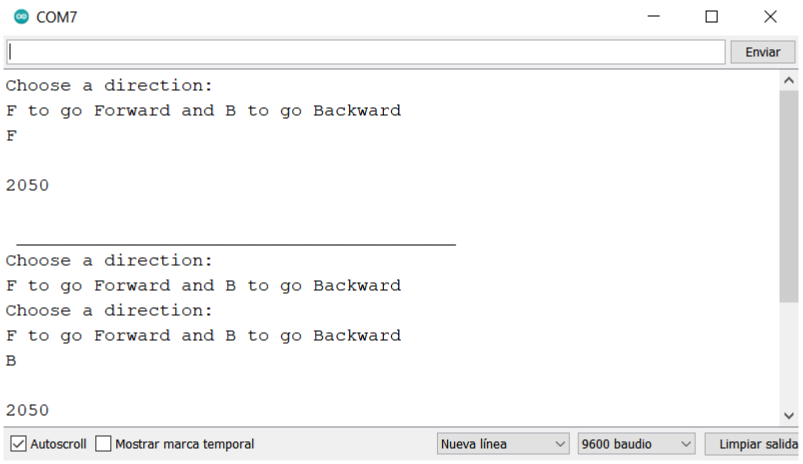
The modifications of the library were just to exchange the output pins to the corresponding ones in the WeMos D1 terminology (for the R1 model for a reason that will be explained in page 73 , as seen in figure 3.34. Some useless code lines were deleted from the original.

After this, two functions were implemented into another code C.2.2 for the detection via the Serial port of a command to either go Forward (F) or Backwards (B), introducing the corresponding letter into the keyboard of the serial monitor in the Arduino IDE, as seen in figure 3.35.



```
#if defined(ESP8266)

  //#define MOTORDEBUG 1

  #include "stdlib_noniso.h"

  #define MICROSTEPS 16

  #define DC_MOTOR_PWM_RATE 5
  #define STEPPER1_PWM_RATE 5
  #define STEPPER2_PWM_RATE 5

  // Arduino pin names for interface to 74HCT595 latch
                               // Arduino pins and modifications
  #define MOTORLATCH D10 // used to be 12
  #define MOTORCLK D4    // used to be  4
  #define MOTORENABLE D7 // used to be  7,   changed to 13
  #define MOTORDATA D8   // used to be  8,   changed to  0

//logic pins for "PWM"
#define MOTOR1_EN D3 // used to be 2
#define MOTOR2_EN D3 // used to be 5
#define MOTOR3_EN D5 // used to be 16
#define MOTOR4_EN D5 // used to be 14


#elif defined(__AVR__)
```

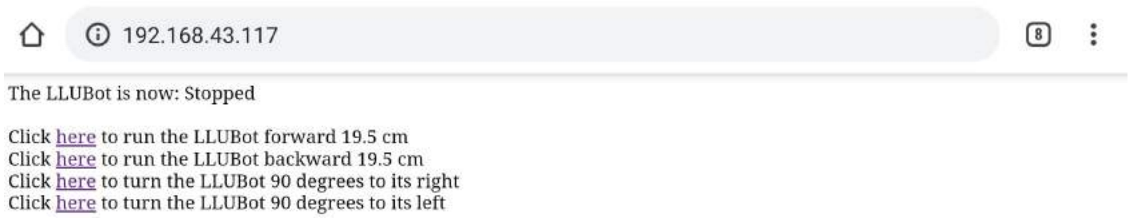
Figure 3.34: Changes in the DMotor_mod.h library (for WeMos D1 R1 use).



```
Choose a direction:
F to go Forward and B to go Backward
F

2050


 ________________________________________
Choose a direction:
F to go Forward and B to go Backward
Choose a direction:
F to go Forward and B to go Backward
B

2050
```

Figure 3.35: Serial monitor view for the FunctionGoStraightSerial.ino code.

To follow, the system used for the GoStraight function in last code was implemented to create a "turn" function inside an implementation of the system to create a very simple web server for the selection of the required movement, as the code C.2.3 responds to, and where the web server is as shown in figure 3.36, where 19.5 cm are approximately the perimeter of the wheels of the robot.

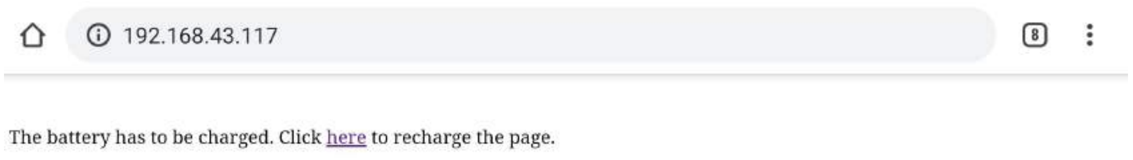

192.168.43.117

The LLUBot is now: Stopped

Click here to run the LLUBot forward 19.5 cm
Click here to run the LLUBot backward 19.5 cm
Click here to turn the LLUBot 90 degrees to its right
Click here to turn the LLUBot 90 degrees to its left

Figure 3.36: Horizontal view of the web server of FunctionsMovementWebServer.ino in a mobile phone.

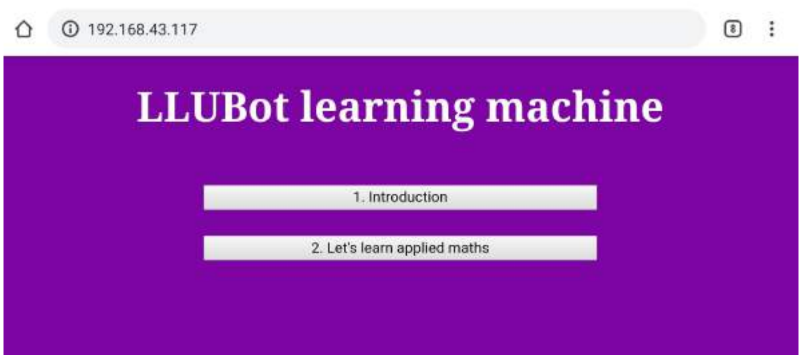
The next step was to detect the battery drop, which was made separately in code C.2.4 and then implemented into the web server as a different page when the battery drop was detected into code C.2.5, which has the same main page as in the previous version, and once the battery has dropped, the web server turns into the one seen in figure 3.37.


192.168.43.117

The battery has to be charged. Click here to recharge the page.

Figure 3.37: Horizontal view of the web server of FunctionsMovementWebServer.ino in a mobile phone once the battery drop has been detected.

The last code implemented was a trial of the introduction of graphic web environment for a client using of HTML, CSS and Java-Script within the Arduino code to make a suitable appearance for two different activities for the educational purpose, as seen in figures 3.38, 3.39 and 3.40 (for more information see code C.2.6).


192.168.43.117

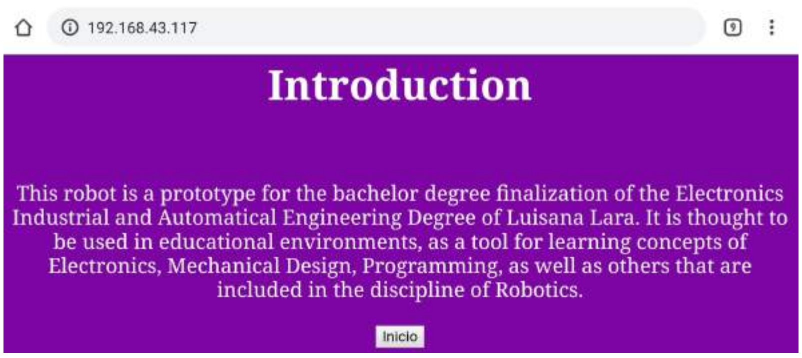
LLUBot learning machine

1. Introduction

2. Let's learn applied maths

Figure 3.38: Horizontal main view of the web server of EducationalWebServer.ino in a mobile phone.

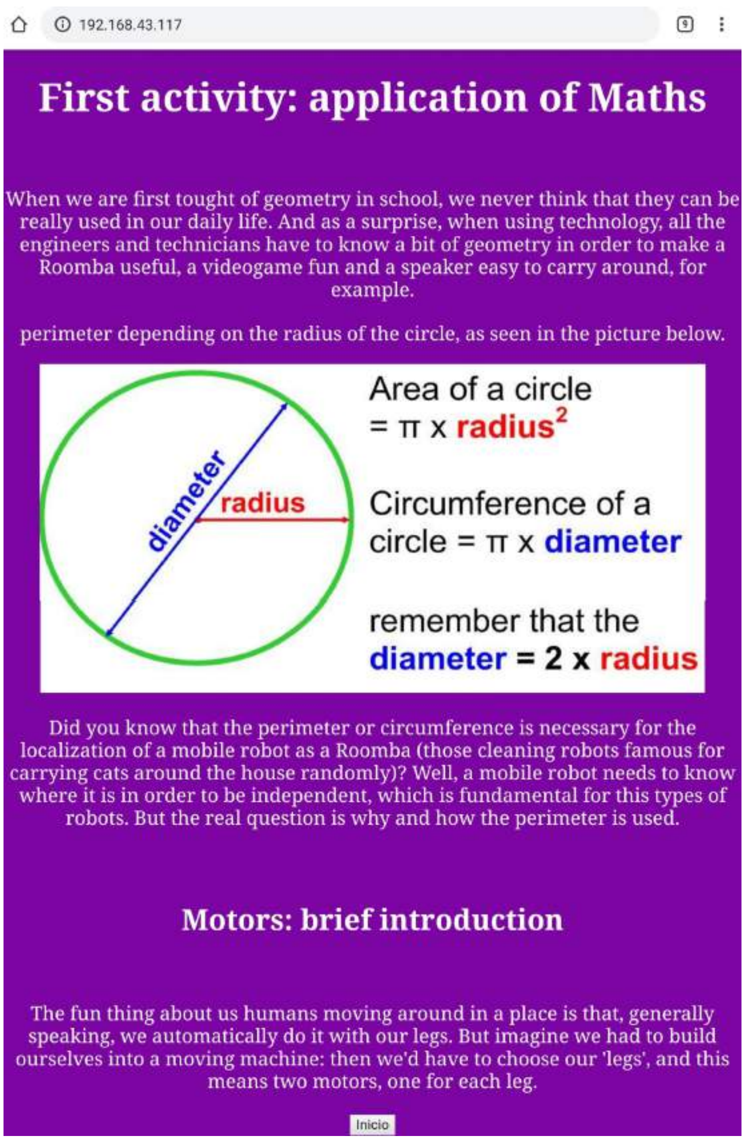
This last code allows the editing of the same page without having to reload the web itself, but it hasn't been implemented for the use of the motors of the robot.


192.168.43.117

Introduction

This robot is a prototype for the bachelor degree finalization of the Electronics Industrial and Automatical Engineering Degree of Luisana Lara. It is thought to be used in educational environments, as a tool for learning concepts of Electronics, Mechanical Design, Programming, as well as others that are included in the discipline of Robotics.

Figure 3.39: Horizontal view of the first button (introduction) of the web server of EducationalWebServer.ino in a mobile phone.


192.168.43.117

First activity: application of Maths

When we are first tought of geometry in school, we never think that they can be really used in our daily life. And as a surprise, when using technology, all the engineers and technicians have to know a bit of geometry in order to make a Roomba useful, a videogame fun and a speaker easy to carry around, for example.

perimeter depending on the radius of the circle, as seen in the picture below.

Area of a circle = π x radius$^2$

Circumference of a circle = π x diameter

remember that the diameter = 2 x radius

Did you know that the perimeter or circumference is necessary for the localization of a mobile robot as a Roomba (those cleaning robots famous for carrying cats around the house randomly)? Well, a mobile robot needs to know where it is in order to be independent, which is fundamental for this types of robots. But the real question is why and how the perimeter is used.

Motors: brief introduction

The fun thing about us humans moving around in a place is that, generally speaking, we automatically do it with our legs. But imagine we had to build ourselves into a moving machine: then we'd have to choose our 'legs', and this means two motors, one for each leg.

Figure 3.40: Several horizontal views of the second button (introduction) of the web server of EducationalWebServer.ino in a mobile phone jointed together.

The implementation of the motor control with a web server as this last code shows, as well as the use of the sensor is a challenge for a posterior continuation of this prototype.

## 3.4 Extra: Exchanging to the Ultrasonic Sensor

There was an idea of accessibility to the only material that couldn't be bought in Tenerife (physically), and it was the Sharp IR sensor. The solution to this problem to be solved is to exchange the sensing system into an ultrasonic one, which is less reliable but good enough in case the Sharp sensor couldn't be found.

This give us the challenge to determine several aspects of the design once again, as two digital pin are needed instead of an analog one for this sensor. And those two pins are the ones used in the battery drop control system, so they are going to be used for this purpose, and latter on we'll see what happens to the battery control system. See figure 3.41

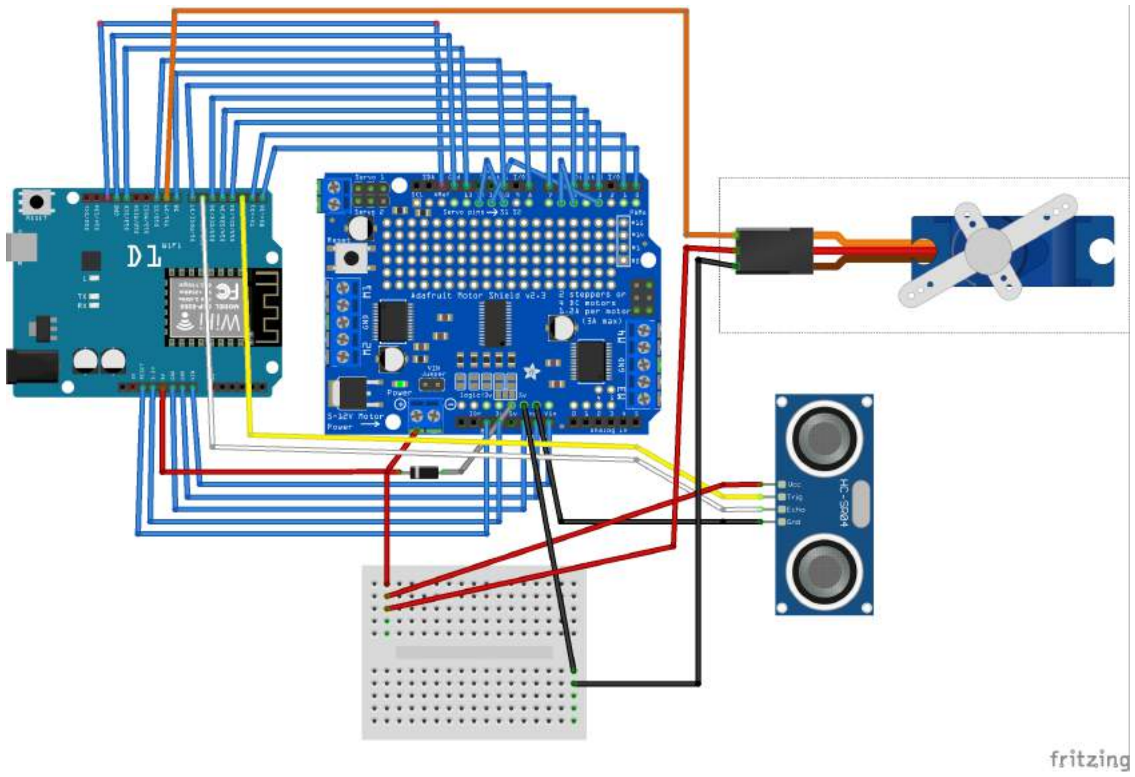


Figure 3.41: Ultrasonic sensor's connections to the system.

Then the battery control system would be from the analog pin, and that means that the maximum input value of the A0 pin is 3.3V, so the calculation of the resistors is other, giving 220 $k\Omega$ as R1 and 141 $k\Omega$ as R2 (the potentiometer), for figure 3.42.

The second part of the entire system to adapt is the two fasteners for the attachment of the sensor into its motion system, so the objects were created and

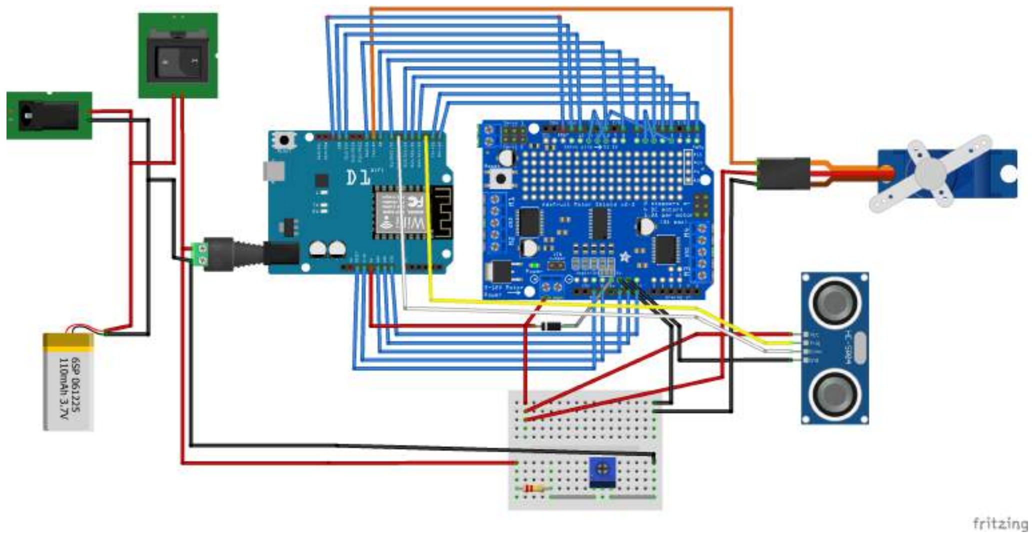


Figure 3.42: Battery control system's connections with Ultrasonic sensor.

verified for its proper use into the prototype (see figures 3.43 and 3.44).

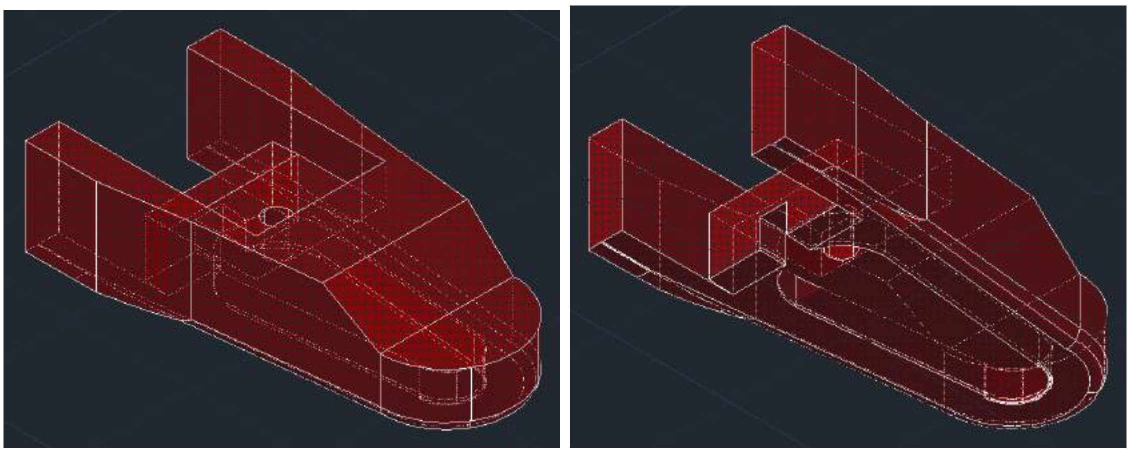


Figure 3.43: Fastener for ultrasonic sensor to micro servo motor, from the front and up to the left and from the back and down to the right.

Then the code has to be modified for its appropriate use. For instance, the digital pins that are used are D2 and D6, for the Trig and Echo output and input of the ultrasonic sensor, so those would have to be introduced following the usual methodology in the maker environment [71]. But the most important change is in the way the battery level is detected, as before we could not count with the analog pin to a better measurement, and now we do. This allow the detection of the percentage of battery, which can be introduced in the web server as a constantly measured value to take into account when using the mobile robot. The only disadvantage is that there can't be a warning LED because there is no digital pin left for it. In this case, it is a problem for a continuation of the project post graduation, as it is a prototype that can be always improved.

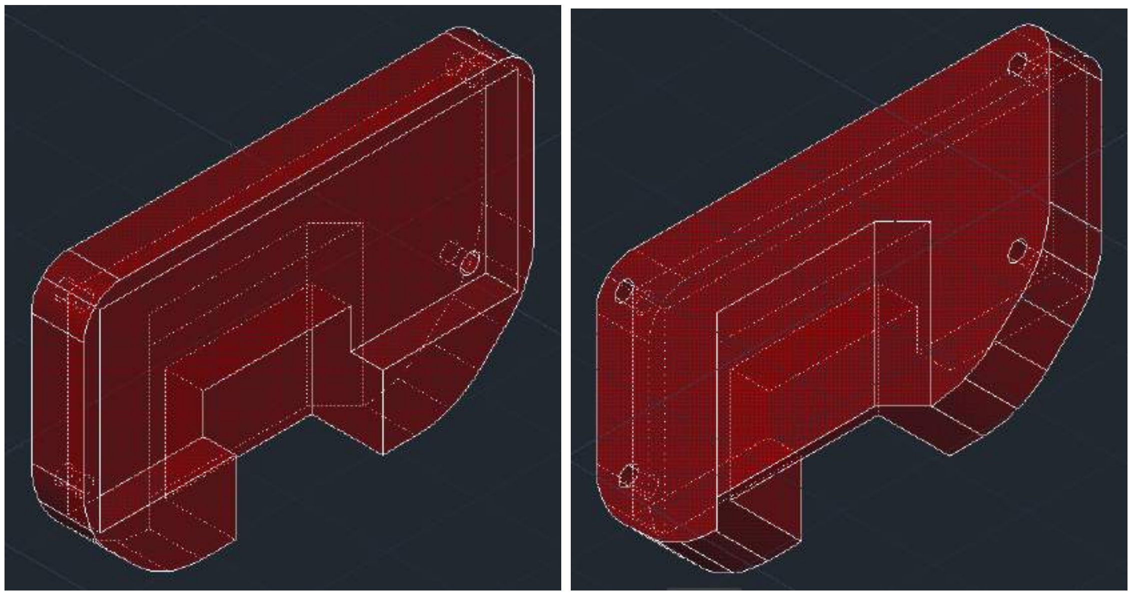


Figure 3.44: Cover-fastener for ultrasonic sensor to micro servo motor, from the front and up to the left and from the back and down to the right.

## 3.5 WeMos D1 R2 non-availability: WeMos D1 R1

It happened that the microcontroller board suffered a several damage during the confinement we all overcame in this 2020 and it had to be replaced for a new one, but in Tenerife the only WeMos D1 available was the R1 (which is a previous version of the same microcontroller board).

This lead to the performance of several changes within the system, as external pinout was called different, they had not extra connections to the same pin in the board, as it did with the previous version (see figure 3.45, and the nomenclature was slightly different, but very important. This last has the easiest solution already seen in table 3.2.

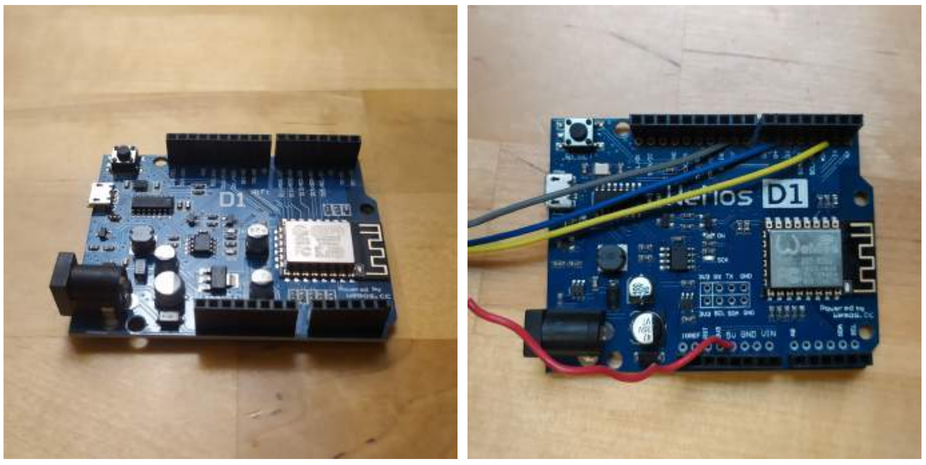


Figure 3.45: WeMos D1 R1 new on the left and R2 broken on the right.

## 3.6 List of materials

For each of the three sections of the creation of this prototype, materials were needed, most of them provided by the department within the robotics laboratory. In the following list are shown the materials used for the electronic design:

- Oscilloscopes with connection wires, usually BNC - crocodile or BNC - testing point.
- Power Supply with connection wires, usually banana - crocodile.
- Voltmeter with its connection wires.
- Welding gun with a loupe and tin.
- Precision screwdrivers.
- Tweezers and sniper sheers for modifying the connection wires of the prototype.
- Google Spread Sheets (software) for the selection of the resistors of the battery drop control.
- PSIM (software) for illustrating the battery drop control system.

For the structure design, there were two softwares used: AutoCAD 3D for the 3D modelling and Ultimaker Cura for the definition of the features when printing (as well as the selection of the appropriate 3D printer). And the other materials used were as follows:

- 3D printer: Creality CR10-S
- 3D printer: BQ Hephestos 2
- 3D printer: Creality Ender 3
- 3D printer: BQ Witbox 2
- Blades for removing the extra material from the 3D printings.
- Mini tool Dremel for slights modifications of the 3D printings.
- Drill for opening the holes of the screws into its proper dimension.

For the design of the program within the mobile robot there were two softwares needed: Arduino IDE for the writing, compiling and uploading of the programs into the microprocessor board, and the Notepad++, which allowed the edition of the different libraries used for the different codes.

And finally, for the creation of this document, several softwares were used, as Google Documents for the writing of a diary book, Screenshots of Windows for the introduction of most of the figures, the app Polish for the edition of several figures, and Overleaf for the redaction, structuring and formatting of this bachelor thesis.

# Chapter 4

# Results

In this chapter, it will be shown the final prototype of the project. It has to be taken into account that the final microcontroller board is the older version of the WeMos D1, the R1, , which will coincide with all of the coding in the appendix C.

## 4.1 Physical Assembly of the LLUBot

Firstly, it has to be introduced the way each of the components were assembled into the 3D printings, as this is the practical way of learning the different disciplines of Robotics into the prototype: the LLUBot.

Once the connections are set up as in the previous chapter is specified, the 3D printings are done, then the real assembly is done as seen in figure 4.1.

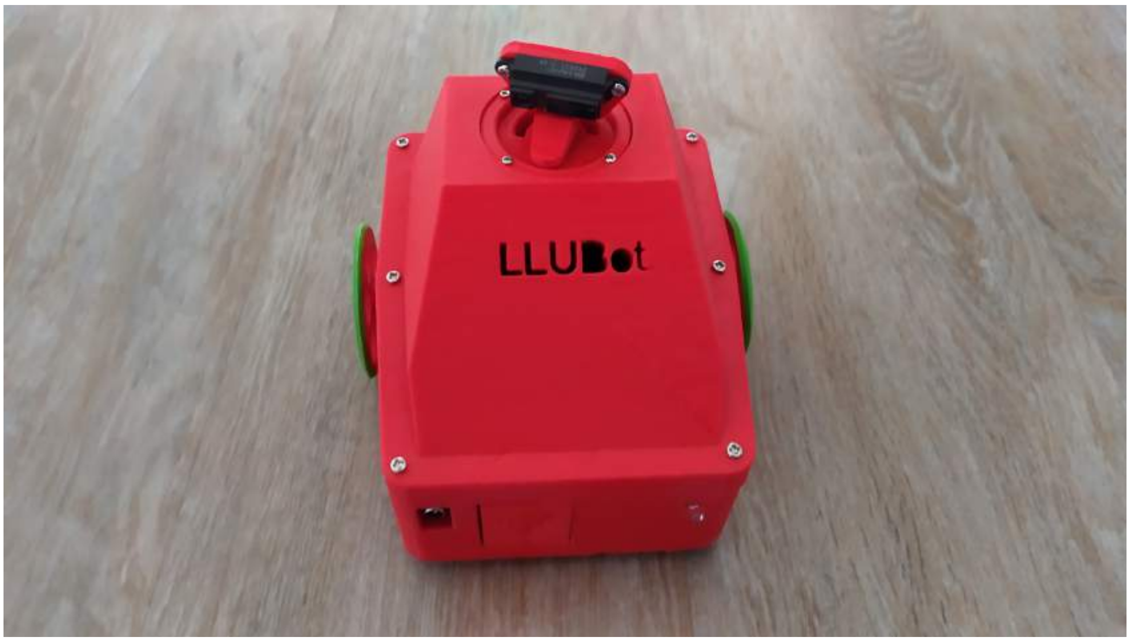


Figure 4.1: Final assembly.

Notice that there are 5 components that must be disconnected to the 3D printing assembly are the battery (very important for safety!), the female power jack connector, the switch, the micro servo motor and the sensor, as seen in figure 4.2.

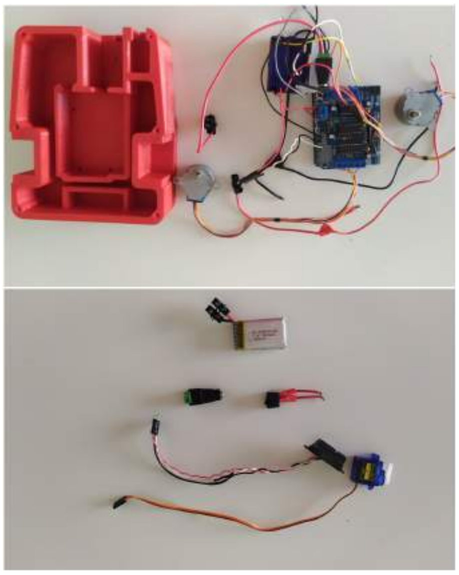


Figure 4.2: Safety of the assembly and other necessary disconnections.

The microcontroller board can be fastened through 4 screws directly connected to the chassis, for then the motor driver shield be placed very carefully into the corresponding pins of the WeMos D1, as seen in figure 4.3.

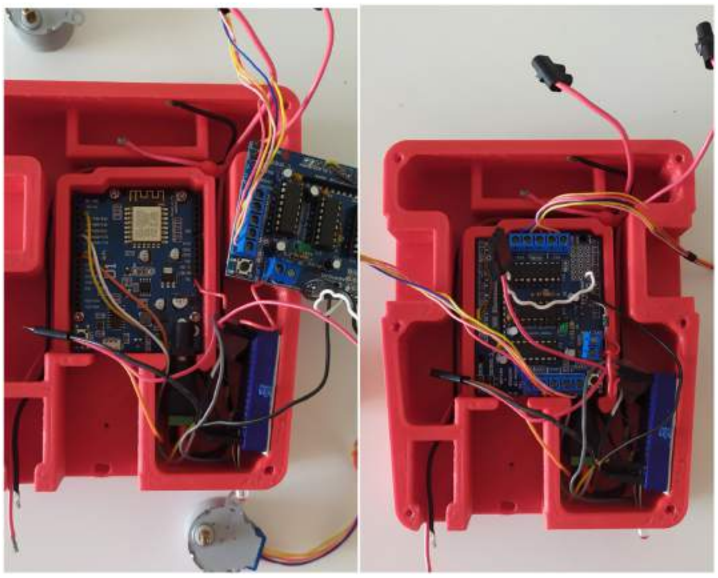


Figure 4.3: Correct assembly of the WeMos D1 and the Motor Driver Shield to the chassis.

Remember to place the 5V wire and the diode into the connector of the shield at the same time as making sure a dupont wire is connected to the breadboard, as seen in figure 4.4.

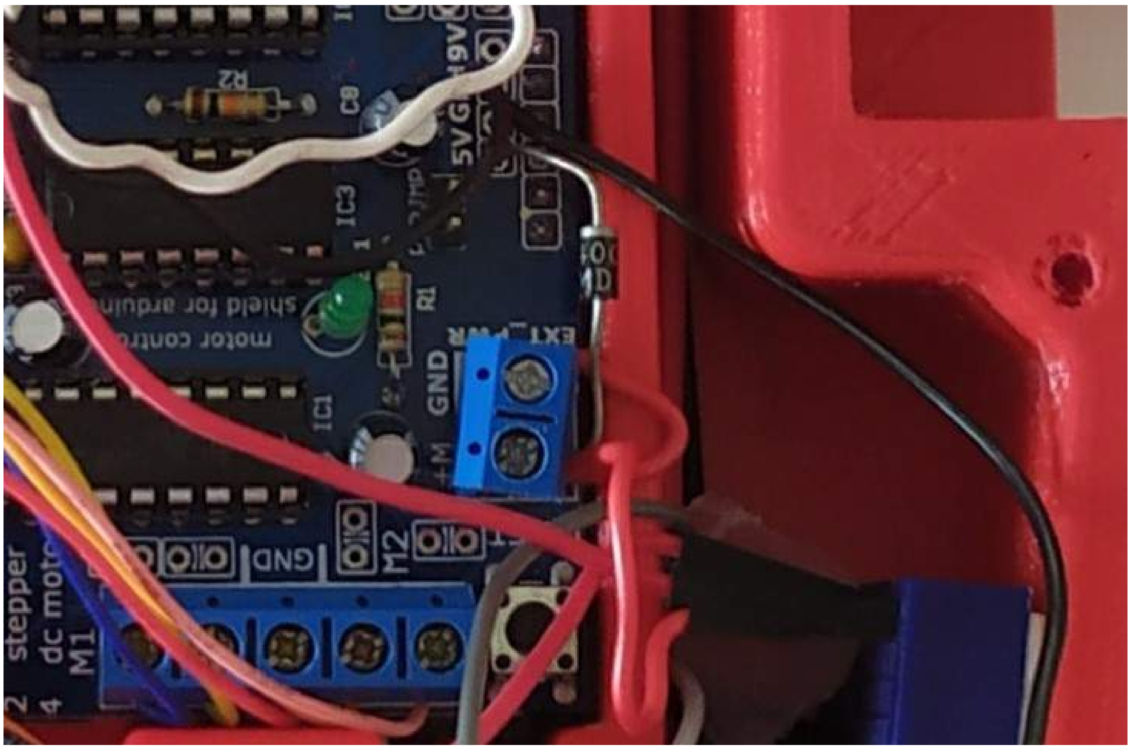


Figure 4.4: 5V power connection between the WeMos D1 and the Motor Driver Shield.

The steppers are assembled just with two screws adequately fastened, to continue connecting the female power jack connector, as seen in figure 4.5.

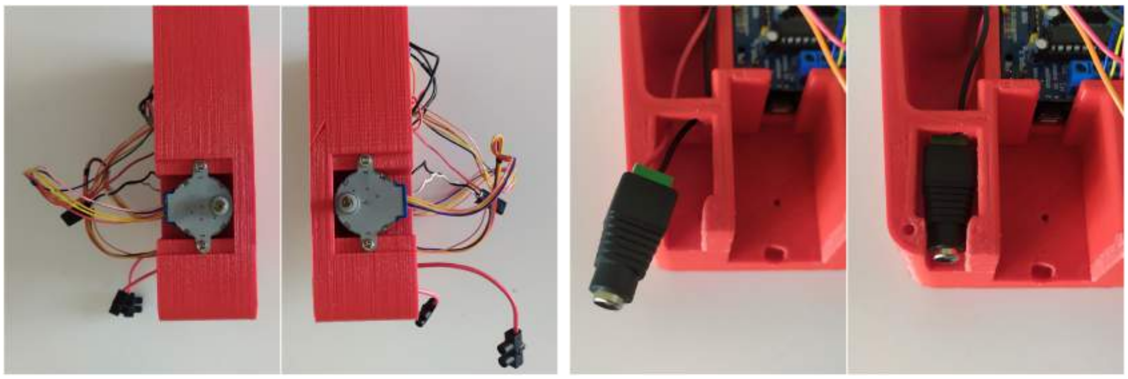


Figure 4.5: Steppers and female power jack connector assemblies to the chassis.

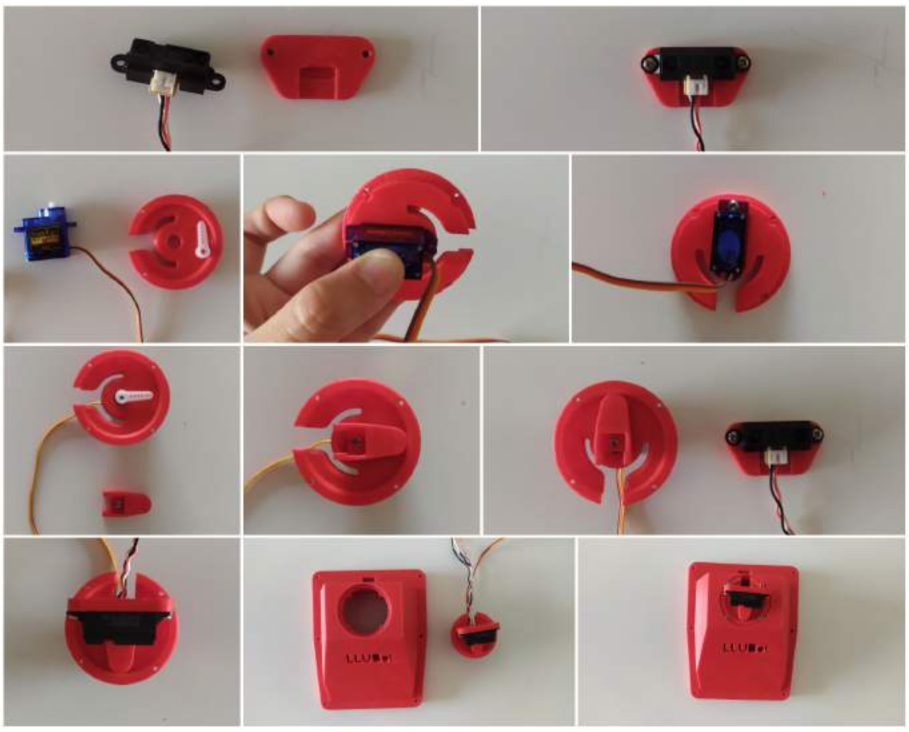


Figure 4.6: Assembly of the fasteners of the sensor's motion system and into the hubcap.

The fasteners of the sensor's motion system are assembled as seen in figure 4.6: firstly the sensor fastener into its back, secondly, the micro servo motor into the biggest of the fasteners (before attaching it to the hubcap), then the third fastener into the micro servo above the second fastener, after that the sensor into its place and finally the whole fastener system into the hubcap.

The switch must be introduced from the exterior of the hubcap into its place for its proper connection, as seen in figure 4.7.

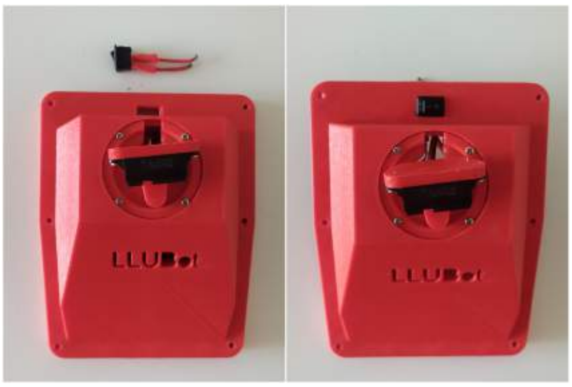


Figure 4.7: Assembly of switch into the hubcap.

Once all of this is done, the connections of the switch, the micro servo motor and the sensor are performed, as seen in figure 4.8.

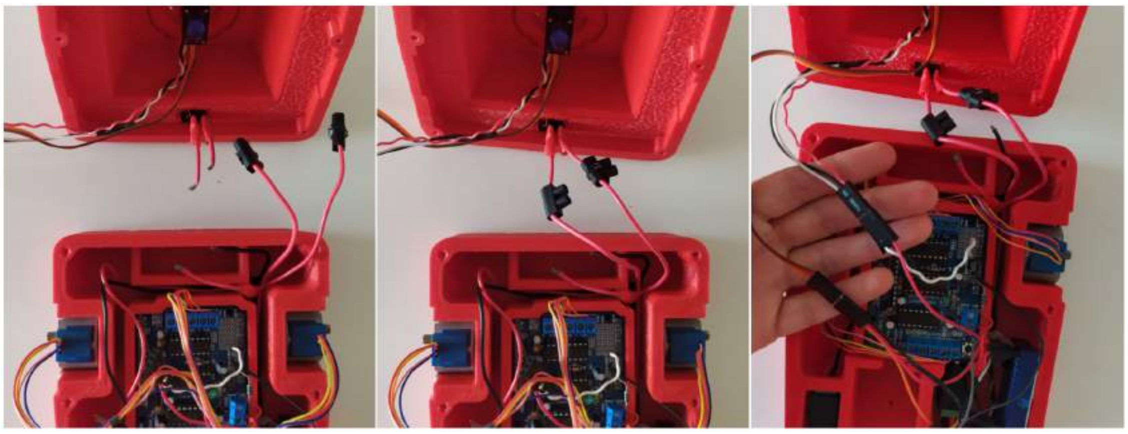


Figure 4.8: Connection of the wires that connect the elements of the hubcap.

And now, making sure that the switch is turned OFF, the battery can be connected, as seen in figure 4.9.

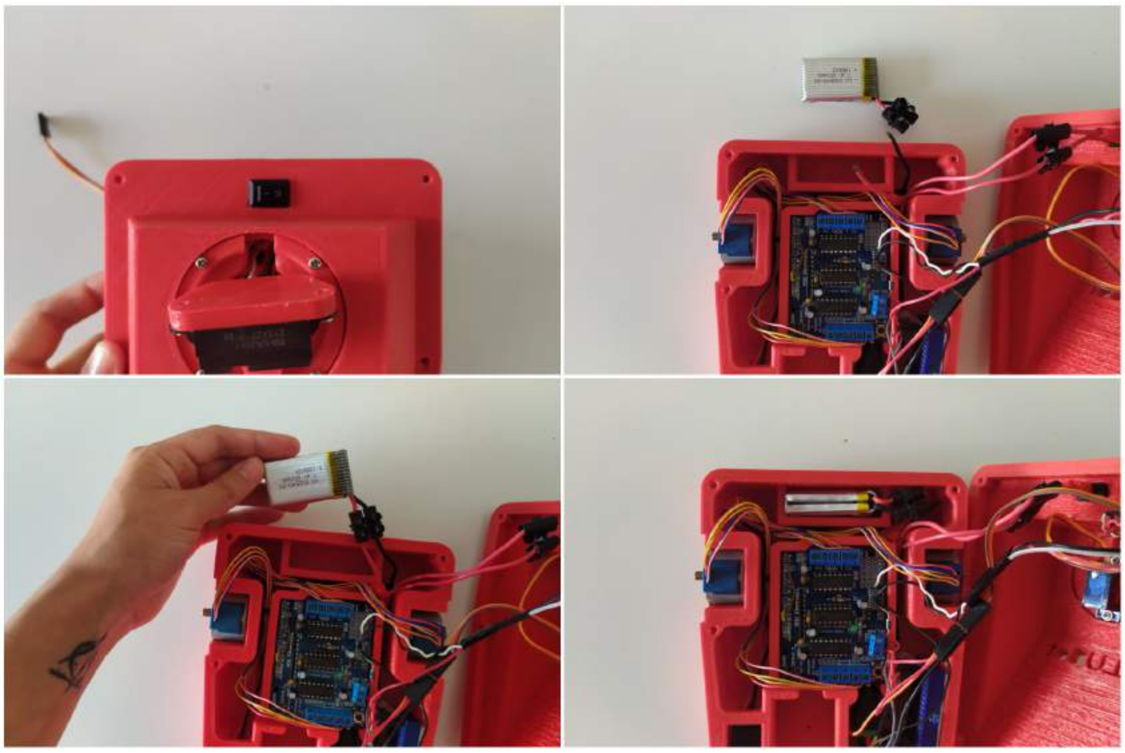


Figure 4.9: Connection of the battery (caution with the switch).

Then the chassis can be connected to the hubcap with 6 screws (don't forget to place the WiFi cover into its hole), to fasten just after the ball caster with two screws as seen in figure 4.10.

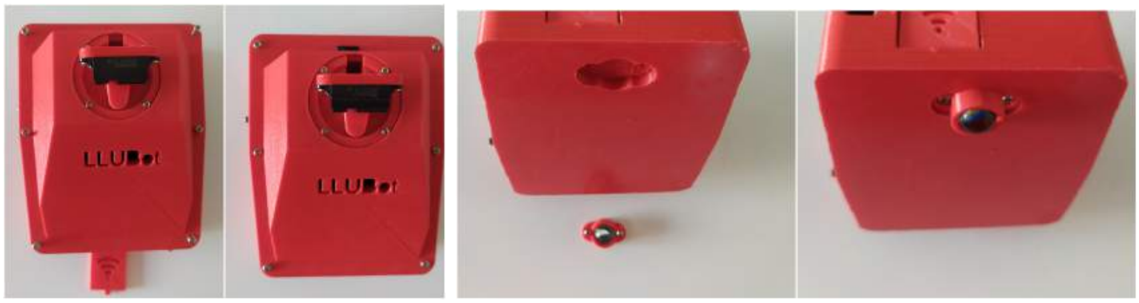


Figure 4.10: Assembly of the chassis, the hubcap and the WiFi cover, as well as the ball caster into the chassis.

And the final touch is to place the wheels into each stepper, which will fit nicely into the motor, as seen in figure 4.11.

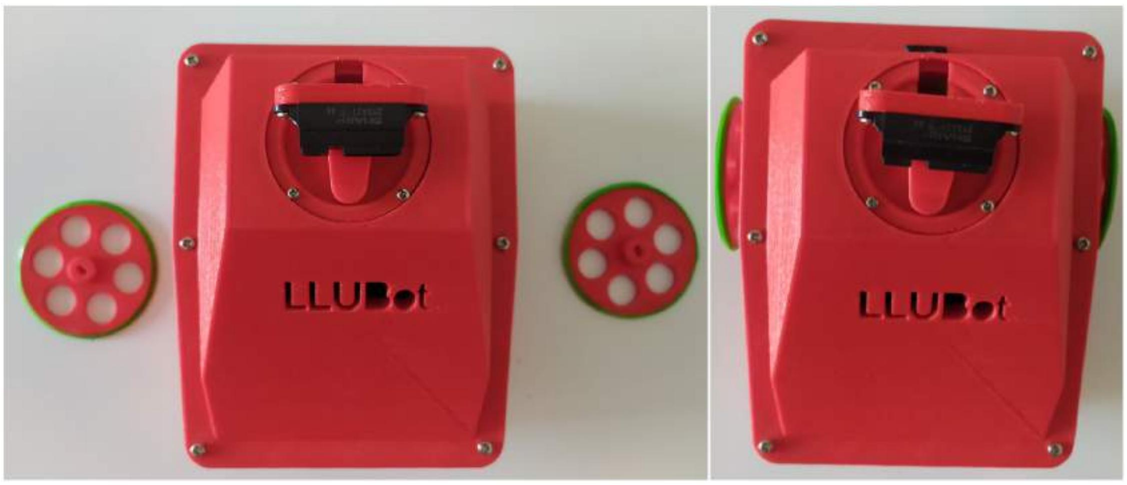


Figure 4.11: Assembly of the wheels into the LLUBot.

## 4.2 Cost of the LLUBot

There's always a very important thing to look at when selecting a mobile robot, and that is the cost of it. In this case, the assembly is part of the learning process about the different disciplines that are included into the big one that is Robotics, and that is a big advantage, just as it is the price of the materials.

For this, a list of components and materials was necessary for the implementation of this mobile robot, and to illustrate the different costs when buying online or in a local shop, as shown in table 4.1.

Table 4.1: Table of costs of the used materials.

| Material | N | Local store price/unit | Worldwide store price/unit |
|---|---|---|---|
| WeMos D1 R1 or R2 | 1 | 12.84 € [58] | 4.57 € [59] |
| Stepper motor | 2 | 3.75 € [62] | 2.39 € [63] |
| Sharp IR sensor (GP2Y0A21YK0F) | 1 | 18.17 € [68] | 4.93 € [69] |
| Arduino Motor Driver Shield | 1 | 5.35 € [72] | 2.22 € [73] |
| 7.4V Lipo Battery | 1 | 6.96 € [74] | 7.00 € [75] |
| Bipolar Jack Connector Female | 1 | 0.43 € [76] | 1.63 € [77] * |
| Bipolar Jack Connector Male (Screwed) | 1 | 0.43 € [78] | — |
| Bipolar Jack Connector Male (Welded) | 1 | 0.43 € [78] | 1.63 € [77] |
| Switch | 1 | 0.32 € [79] | 1.01 € [80] |
| Breadboard | 1 | 1.28 € [81] | 1.42 € [82] |
| Battery charger | 1 | 10.70 € [83] | 7.68 € [84] |
| 1kg of 3D Printer Filament | 1 | 17.99 € [85] | 14.99 € [86] |
| Dupont Wires Female - Male | 1 | 2.68 € [87] | 2.67 € [88] * |
| Dupont Wires Male - Male | 1 | 3.90 € [89] | — |
| Micro USB to USB Wire | 1 | 2.20 € [90] | 1.20 € [91] |
| Micro Servo Motor SG-90 | 1 | 2.95 € [92] | 1.86 € [93] |
| Resistor | 1 | 0.05 € [94] | 2.20 € [95] * |
| Variable Resistor | 1 | 0.24 € [96] | — |
| At least 1 meter of wire | 1 | 10.70 € [97] | 2.10 € [98] |
| Wire connectors | 1 | 1.39 € [99] | 2.60 € [100] |
| LED | 1 | 0.05 € [101] | 1.64 € [102] |
| Diode | 1 | 0.21 € [103] | 1.60 € [104] |
| Total per complete robot | | 106.77 € | 69.57 € |

The cost using a ultrasonic sensor instead would vary locally [66] to 91.28 € and globally [67] 66.37 €.

This doesn't include all of the other materials necessary for the assembly, which has been discussed at the end of the previous chapter.

## 4.3 Useful codes

The useful programming results are the final 2 codes seen in page 69, codes C.2.5 and C.2.6. This two files can be mixed up to make a better learning platform, which is the next step for the continuing of this project. And if you check pages for the sharp sensor object detection [105] or for the ultrasonic sensor object detection [71], an even greater educational method can be created with this prototype, including the methods into a final coding.

# Chapter 5

# Conclusion

This gives us the LLUBot, an open-source mobile robot made with very accessible and low cost hardware, reaching a global cost of 106.77 € in local stores or 69.57€ € in online shops. Stepper motors have been used to allow a economical and precise position control, and a graphic web interface made with HTML has been programmed to control the robot through the most accessible devices: mobile phones, connected through WiFi, and making it possible to learn that robotics is much more than programming, as most of the educational robots teach, but to understand also about electronics and design when getting to put together your own LLUBot.

It has been implemented the use of a distance sensor with the capability of turning its direction up to 180°, with a motion system designed with 3D environments, as well as the rest of the robot structure. All the created parts have been tested through the use of simple testing codes, as C.2.1, and modified when necessary, turning into the final prototype seen in figure 5.1.

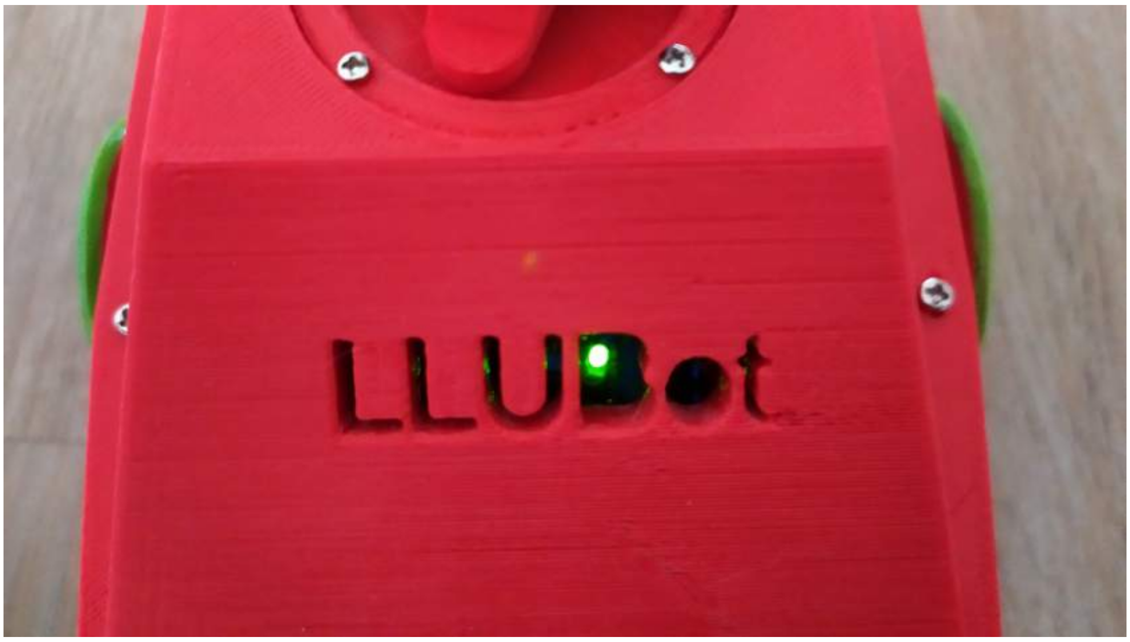


Figure 5.1: LLUBot ON.

# Appendices

# Appendix A

# Electronic Diagrams

In this appendix will be shown with more detail the complex circuit schematics of the different components previously specified in the document.

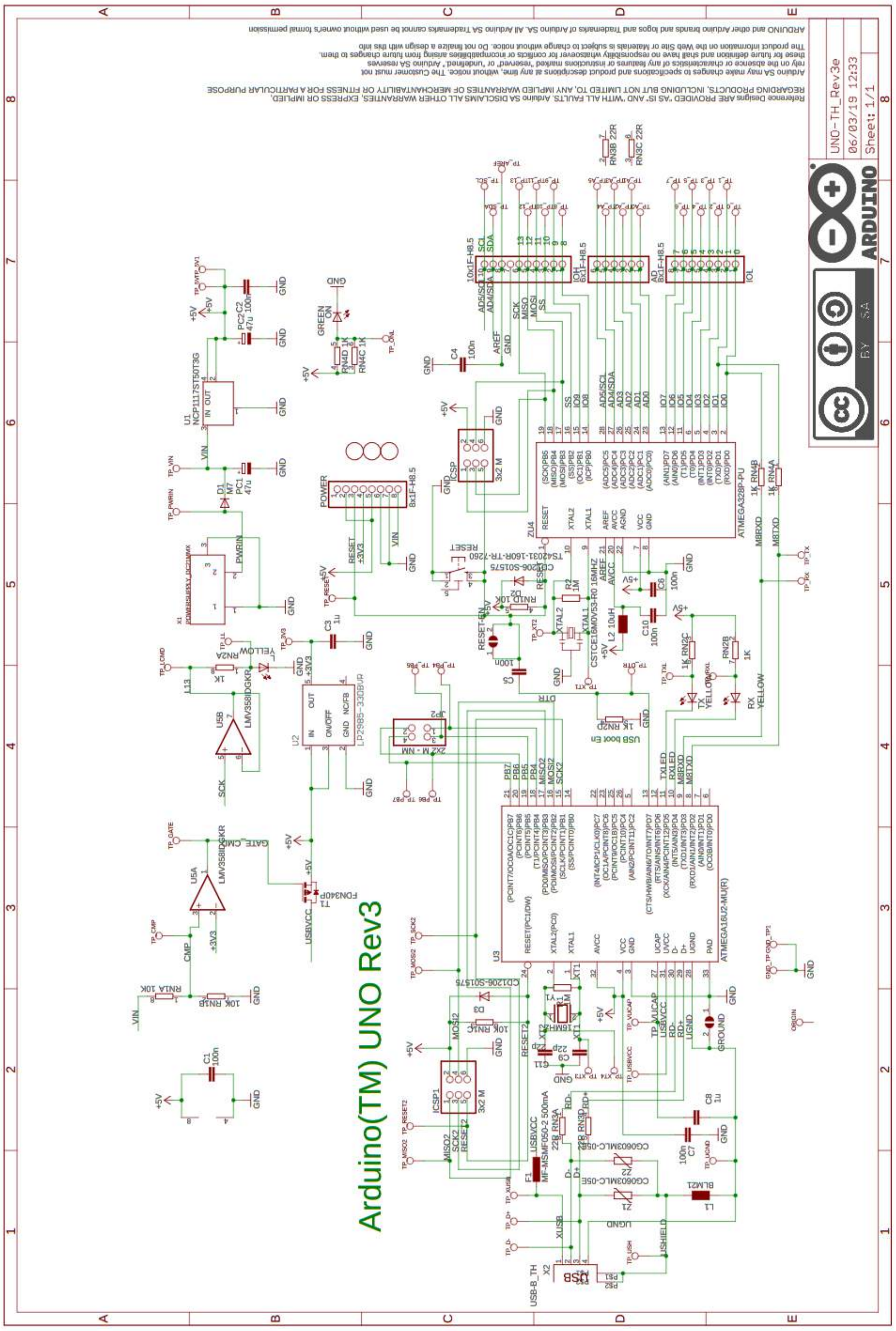


Figure A.1: Arduino UNO Schematics.

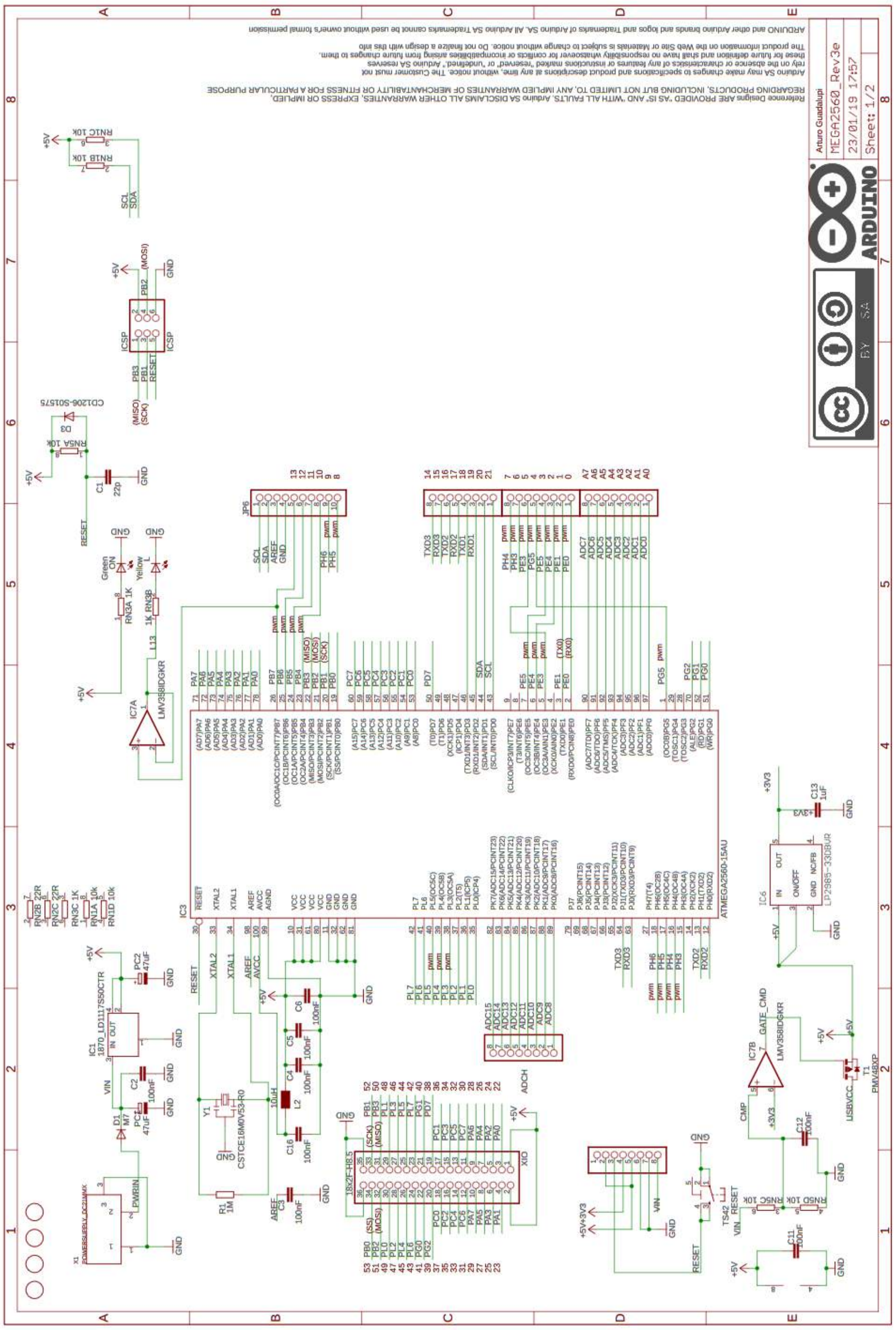


Figure A.2: Arduino Mega Schematics, sheet 1.

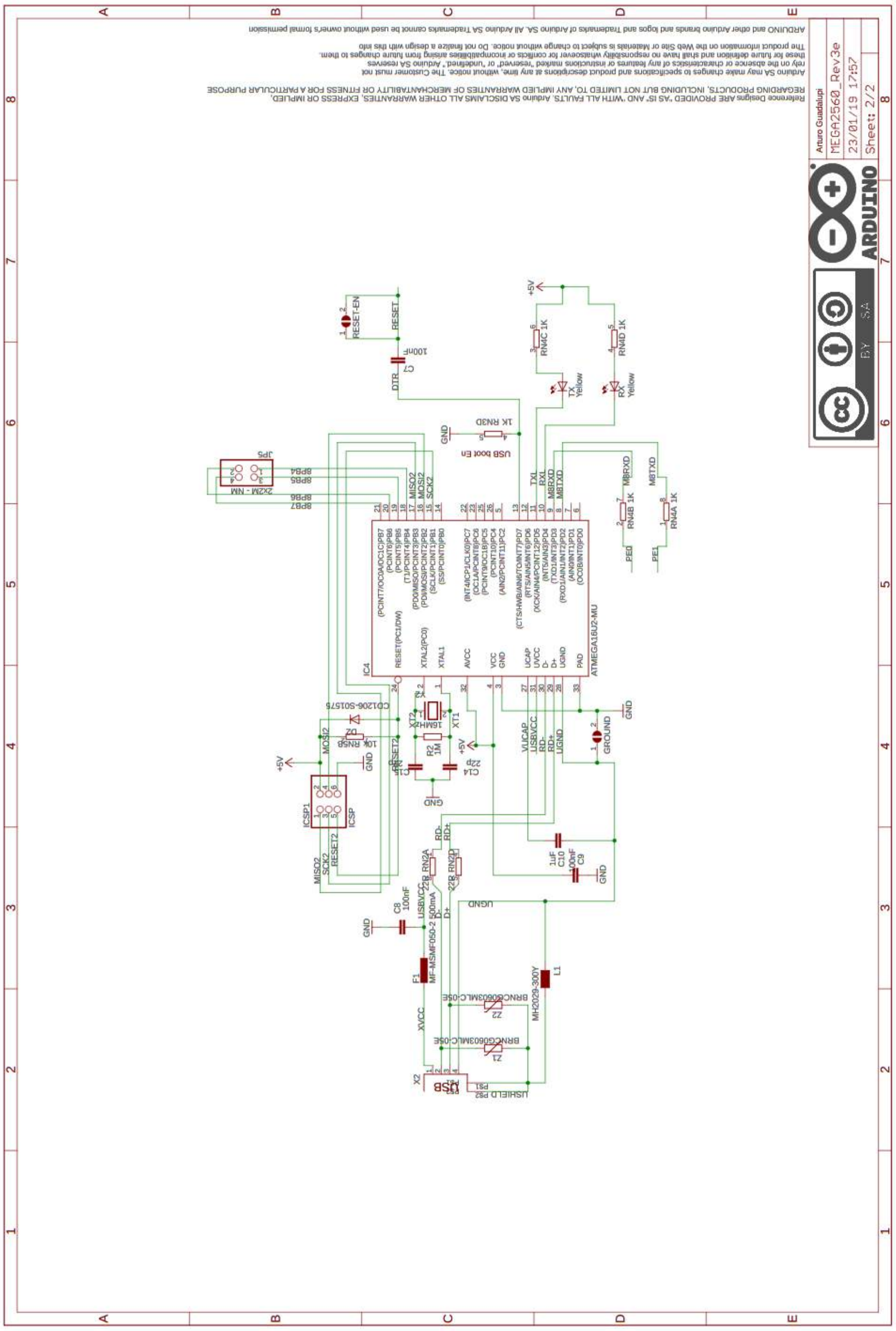


Figure A.3: Arduino Mega Schematics, sheet 2.

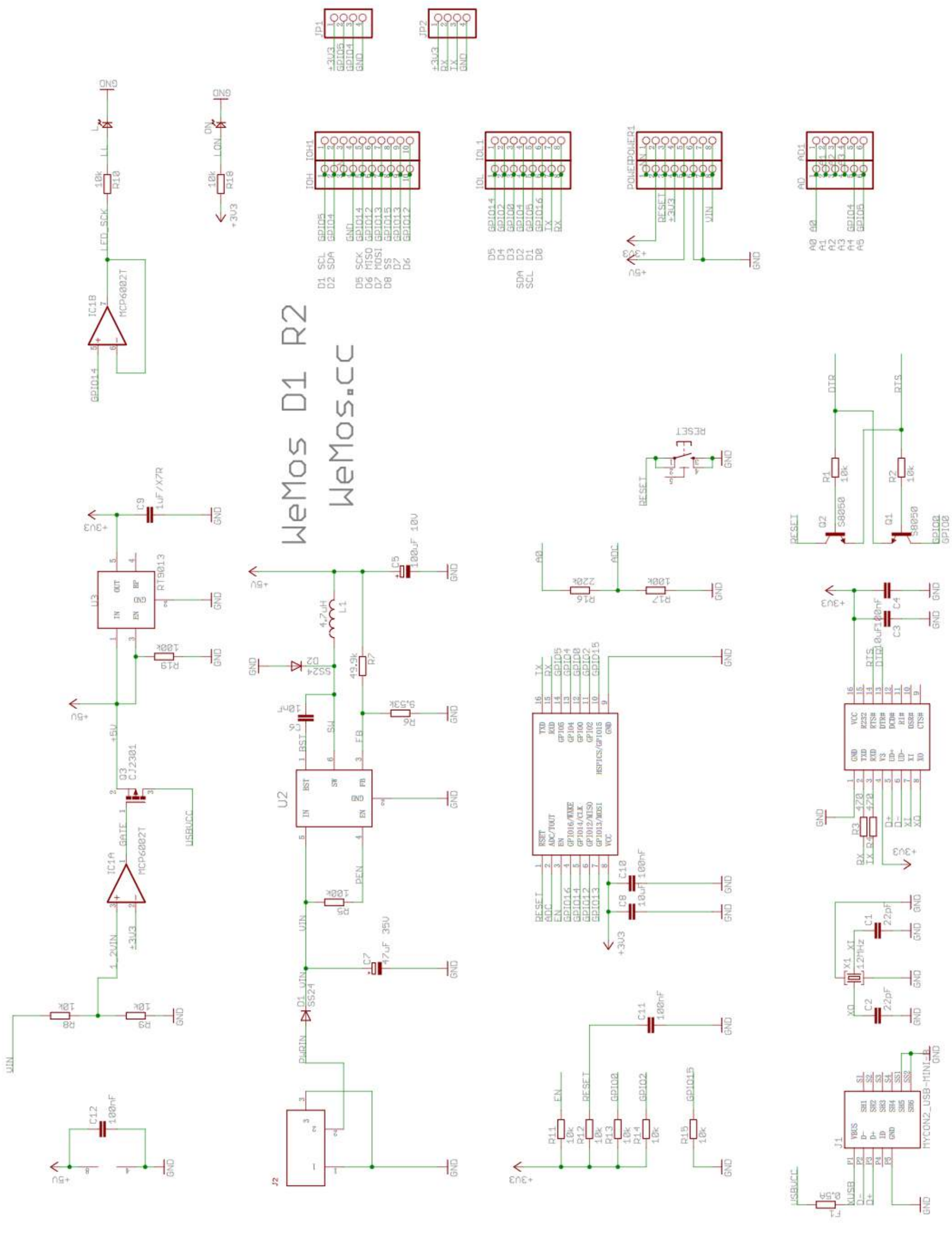


Figure A.4: WeMos D1 R2 Schematics.

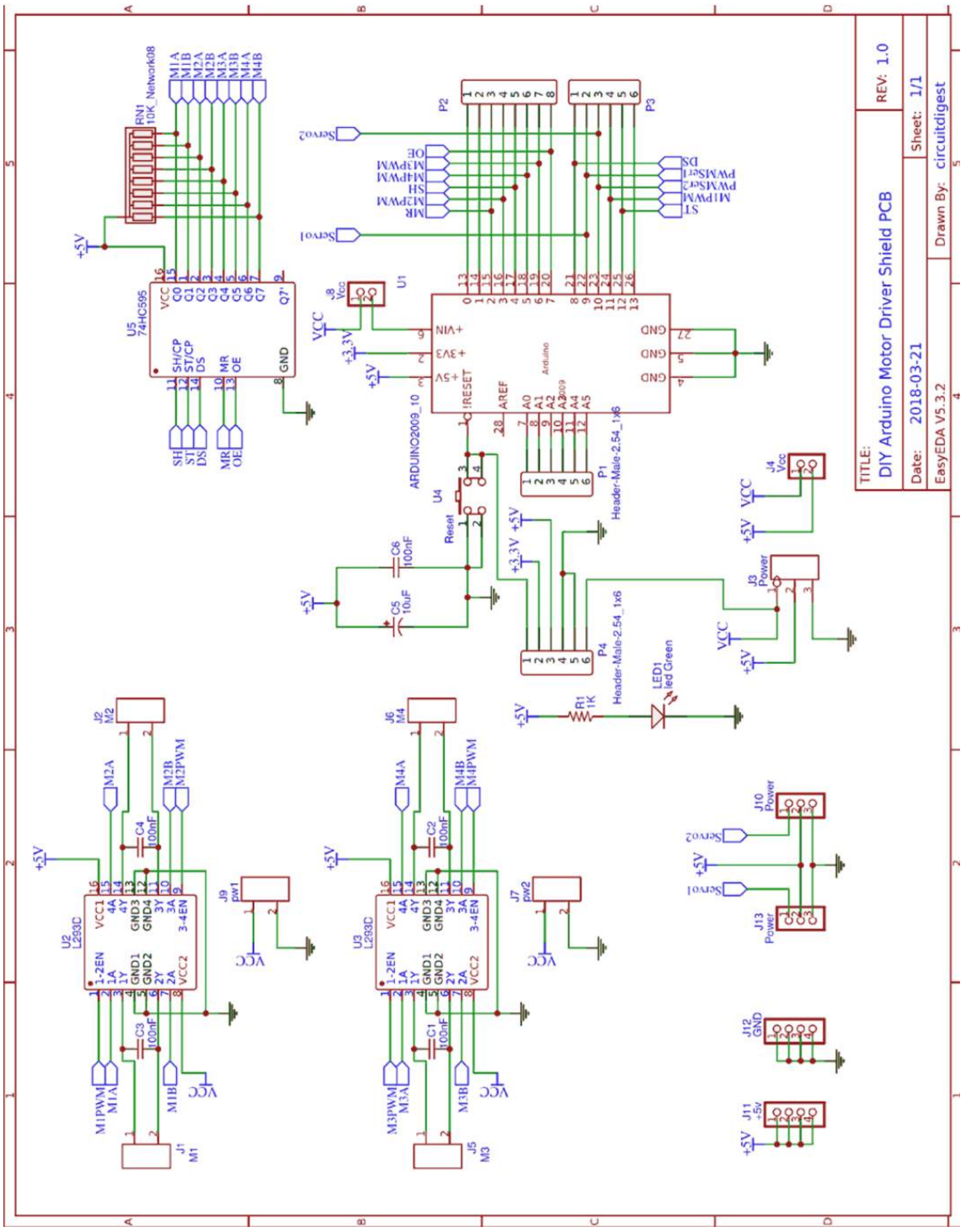


Figure A.5: Arduino Motor Driver Shield Schematics.

# Appendix B

# 3D Designs

All the 3D designs and their modifications are shown as follows, as well as uploaded to a project called LLUBot in GitHub [106].

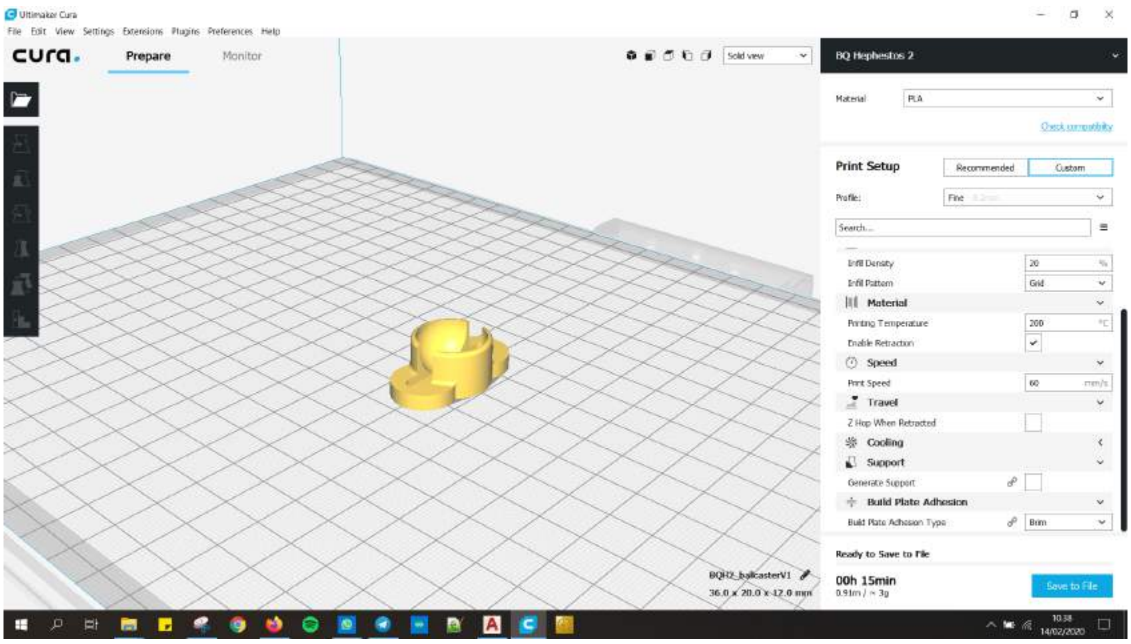
In tables B.1 and B.2 it is made an analysis of the time and material spent in the different versions of the LLUBot prototype.

Table B.1: Table of costs of the used materials.

| Model | Time (h:m) | Weight (g) |
|---|---|---|
| Wheels spooked (V2) | 1:50 | 15 |
| Wheels solid (V1) | 2:29 | 21 |
| Every small piece except wheels V1 | 2:29 | 22 |
| Every small piece except wheels V2.0 | 3:02 | 27 |
| Every small piece except wheels V2.1 | 3:02 | 26 |
| Every small piece except wheels V3 | 5:14 | 24 |
| Every small piece V1 | 4:59 | 43 |
| Every small piece V2.0 | 5:31 | 48 |
| Every small piece V2.1 | 4:49 | 41 |
| Every small piece V3 | 7:32 | 36 |
| Chassis V1 | 13:41 | 218 |
| Chassis V2.0 | 13:45 | 214 |
| Chassis V2.1 | 13:54 | 214 |
| Hubcap V1 | 11:45 | 201 |
| Hubcap V2.0 (used for V2.1) | 11:46 | 202 |

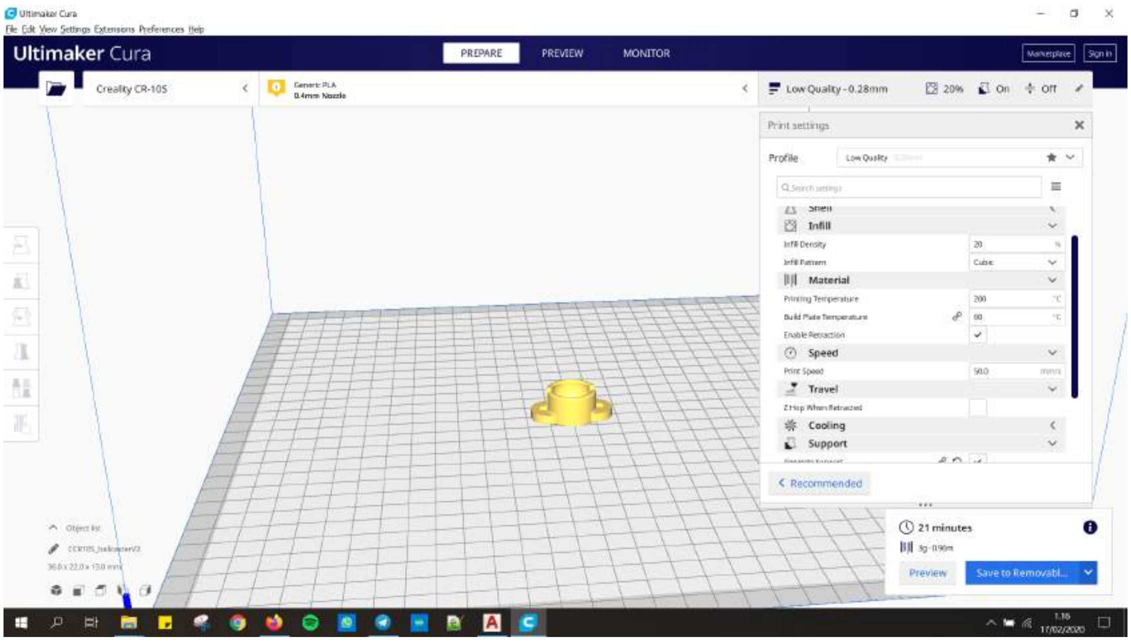
Table B.2: Table of costs of the used materials.

| Model | Time (h:m) | Weight (g) |
|---|---|---|
| LLUBot version V1 | 30:24 | 43 |
| LLUBot version V2.0 | 30:23 | 48 |
| LLUBot version V2.1 | 4:49 | 41 |
| LLUBot version V3 | 7:32 | 36 |



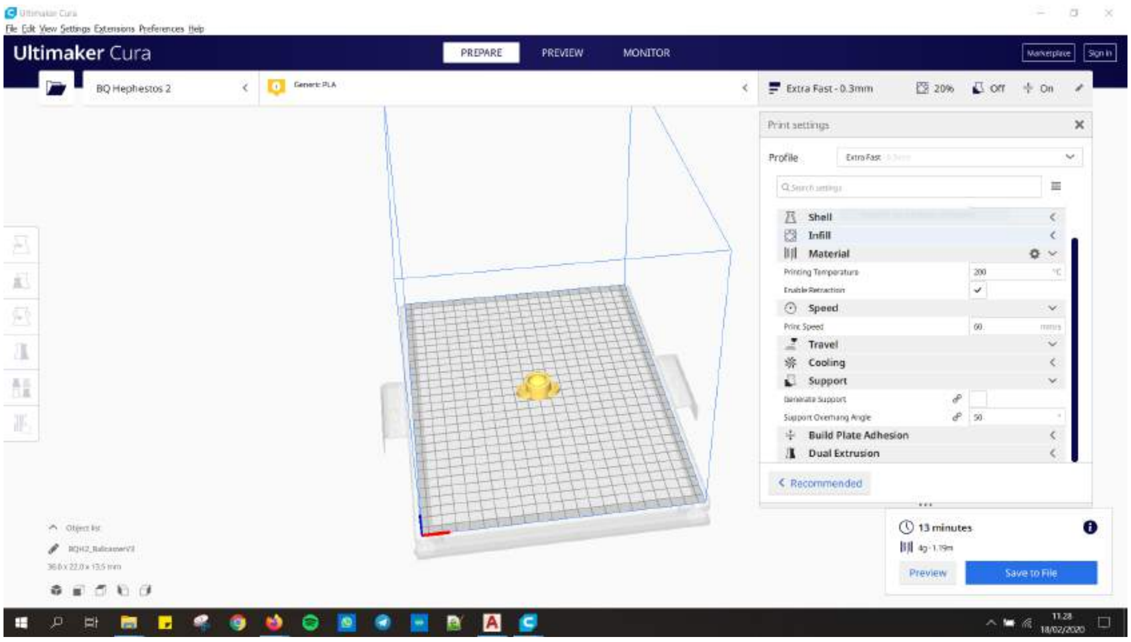
Material: PLA
Print Setup: Custom
Profile: Fine
Infill Density: 20 %
Infill Pattern: Grid
Material
Printing Temperature: 200 °C
Enable Retraction: ✓
Speed
Print Speed: 60 mm/s
Travel
Z Hop When Retracted: ☐
Cooling
Support
Generate Support: ☐
Build Plate Adhesion
Build Plate Adhesion Type: Brim
Ready to Save to File
00h 15min
0.91m / ~ 3g
BQH2_ballcasterV1
36.0 x 20.0 x 12.0 mm
10:38
14/02/2020

Figure B.1: Printing setups: Ball caster V1.



Creality CR-10S
Generic PLA
0.4mm Nozzle
Low Quality - 0.28mm 20% On Off
Profile: Low Quality
Infill
Infill Density: 20 %
Infill Pattern: Cubic
Material
Printing Temperature: 200 °C
Build Plate Temperature: 60 °C
Enable Retraction: ✓
Speed
Print Speed: 50.0 mm/s
Travel
Z Hop When Retracted: ☐
Cooling
Support
21 minutes
3g · 0.90m
CCR10S_ballcasterV2
36.0 x 22.0 x 13.0 mm
1:16
17/02/2020

Figure B.2: Printing setups: Ball caster V2, slightly taller and thicker than previous.



Ultimaker Cura

BQ Hephestos 2 | Generic PLA

Extra Fast - 0.3mm | 20% | Off | On

Print settings

Profile: Extra Fast

Shell

Infill

Material

Printing Temperature: 200 °C

Enable Retraction: ✓

Speed

Print Speed: 60 mm/s

Travel

Cooling

Support

Generate Support: ☐

Support Overhang Angle: 50 °

Build Plate Adhesion

Dual Extrusion

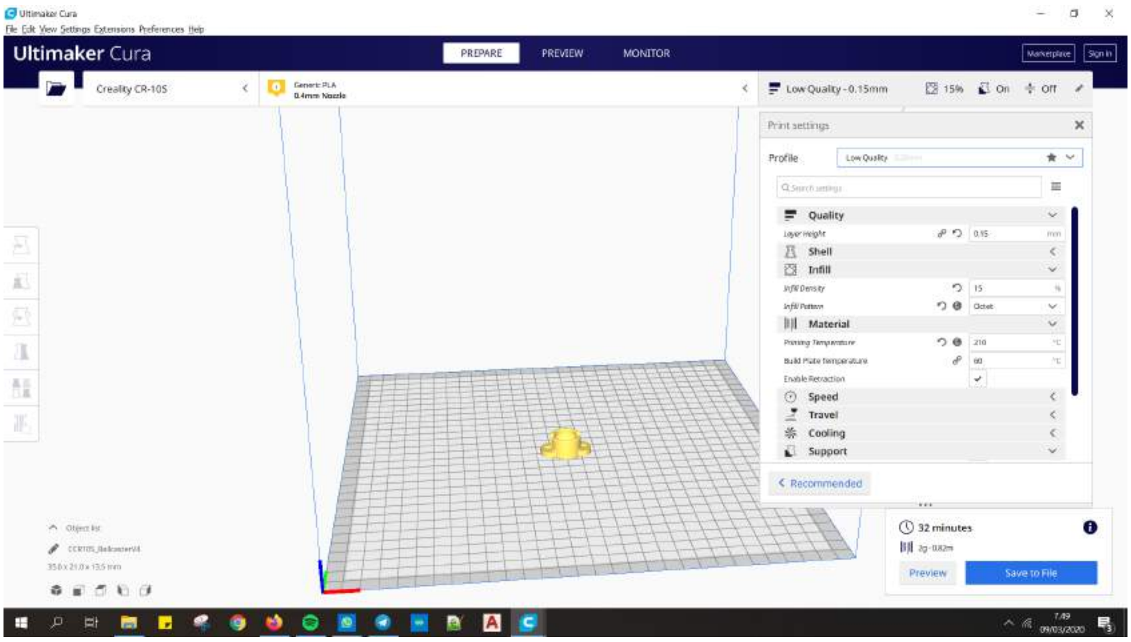
13 minutes

4g · 1.19m

BQH2_BallcasterV3

36.0 x 22.0 x 13.5 mm

Figure B.3: Printing setups: Ball caster V3: slightly thicker than previous.



Ultimaker Cura

Creality CR-10S | Generic PLA 0.4mm Nozzle

Low Quality - 0.15mm | 15% | On | Off

Print settings

Profile: Low Quality

Quality

Layer Height: 0.15 mm

Shell

Infill

Infill Density: 15 %

Infill Pattern: Octet

Material

Printing Temperature: 210 °C

Build Plate Temperature: 60 °C

Enable Retraction: ✓

Speed

Travel

Cooling

Support

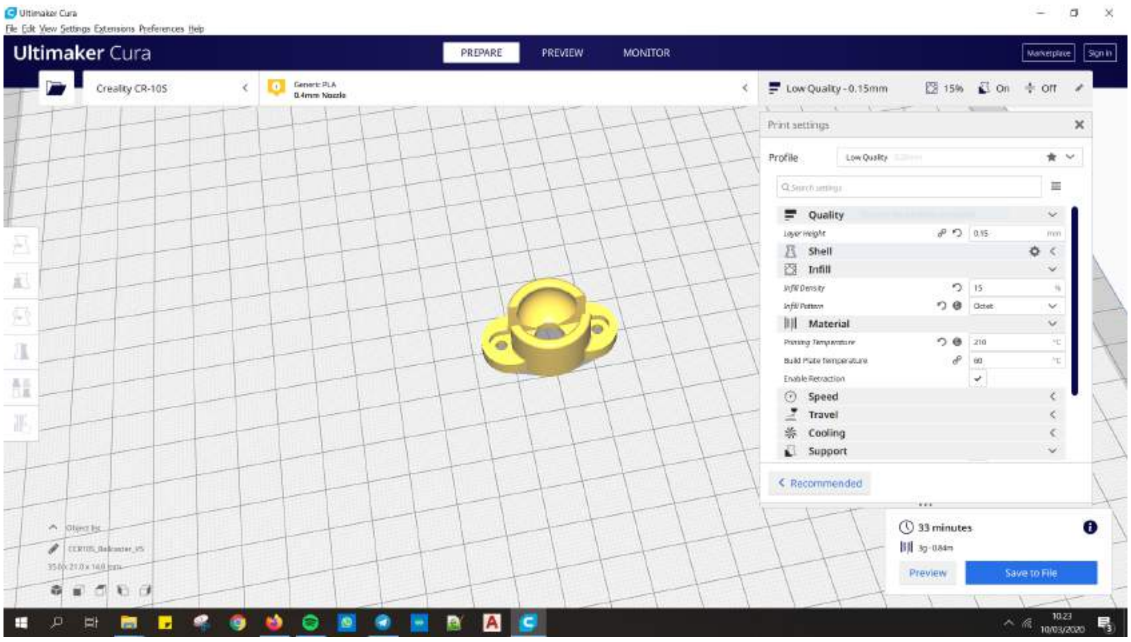
32 minutes

2g · 0.82m

CCR10S_BallcasterV4

35.0 x 21.0 x 13.5 mm

Figure B.4: Printing setups: Ball caster V4: slightly wider the fastening holes than previous.


Ultimaker Cura

Creality CR-10S | Generic PLA 0.4mm Nozzle | Low Quality - 0.15mm | 15% | On | Off

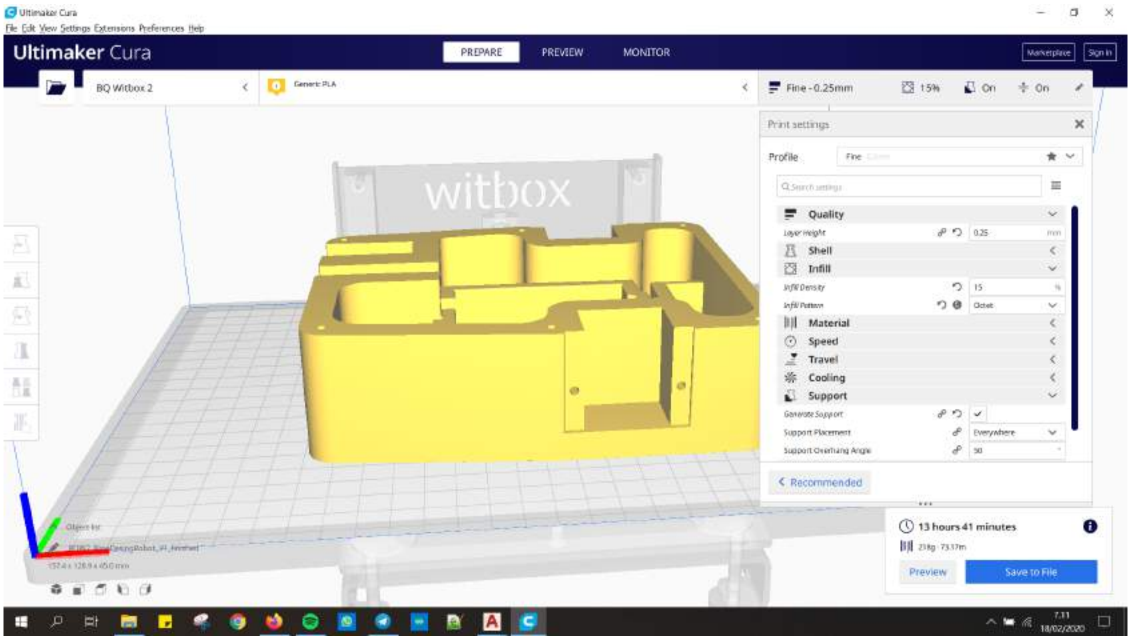
Print settings

Profile: Low Quality

Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 210 °C
Build Plate Temperature 60 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support

33 minutes
3g · 0.84m

Figure B.5: Printing setups: Ball caster V5: slightly thicker than previous.


Ultimaker Cura

BQ Witbox 2 | Generic PLA | Fine - 0.25mm | 15% | On | On

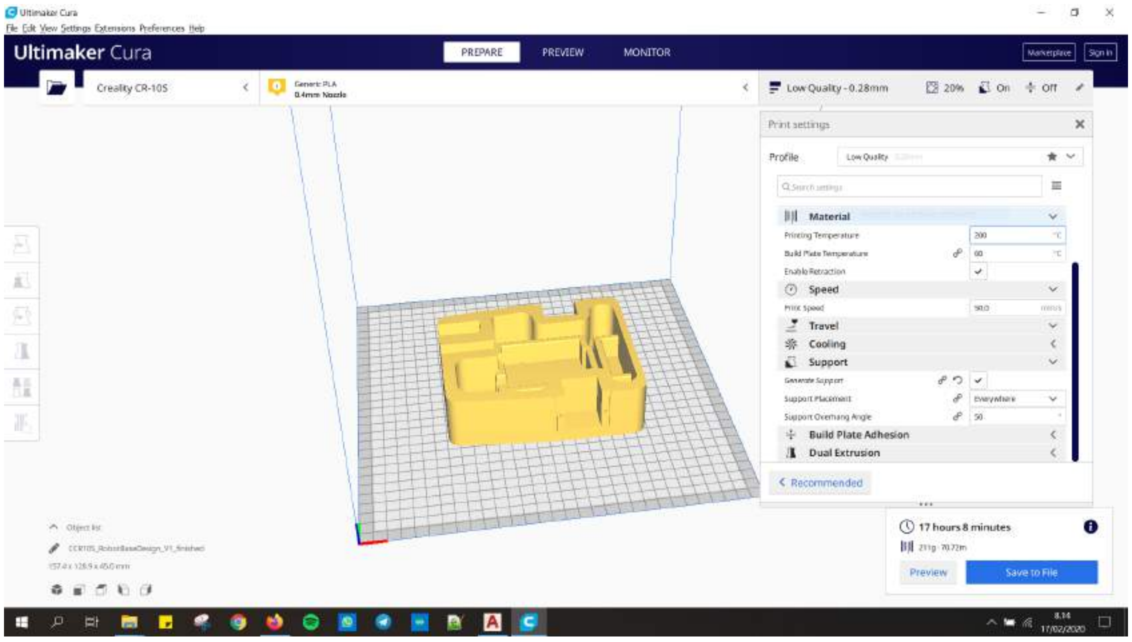
Print settings

Profile: Fine

Quality
Layer Height 0.25 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Speed
Travel
Cooling
Support
Generate Support ✓
Support Placement Everywhere
Support Overhang Angle 50 °

13 hours 41 minutes
218g · 73.17m

Figure B.6: Printing setups: Chassis V1 on WitBox 2 (used).

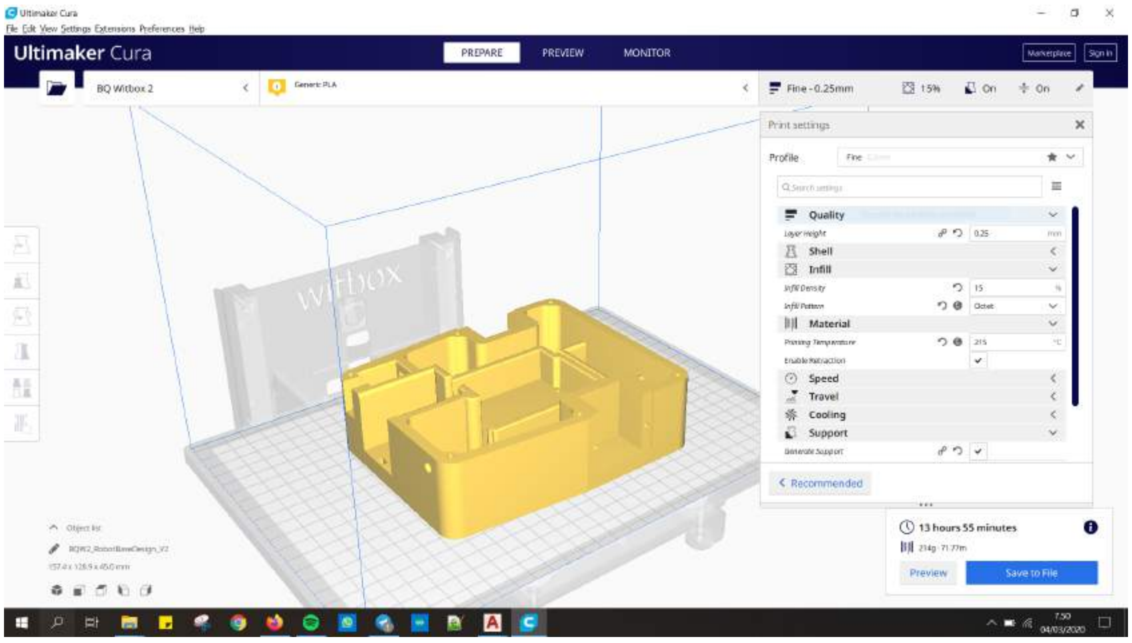

Creality CR-10S
Generic PLA
0.4mm Nozzle
Low Quality - 0.28mm 20% On Off

Print settings
Profile Low Quality

Material
Printing Temperature 200 °C
Build Plate Temperature 60 °C
Enable Retraction ✓
Speed
Print Speed 50.0 mm/s
Travel
Cooling
Support
Generate Support ✓
Support Placement Everywhere
Support Overhang Angle 50 °
Build Plate Adhesion
Dual Extrusion

17 hours 8 minutes
211g · 70.72m

Figure B.7: Printing setups: Chassis V1 on Creality CR10S (not used).

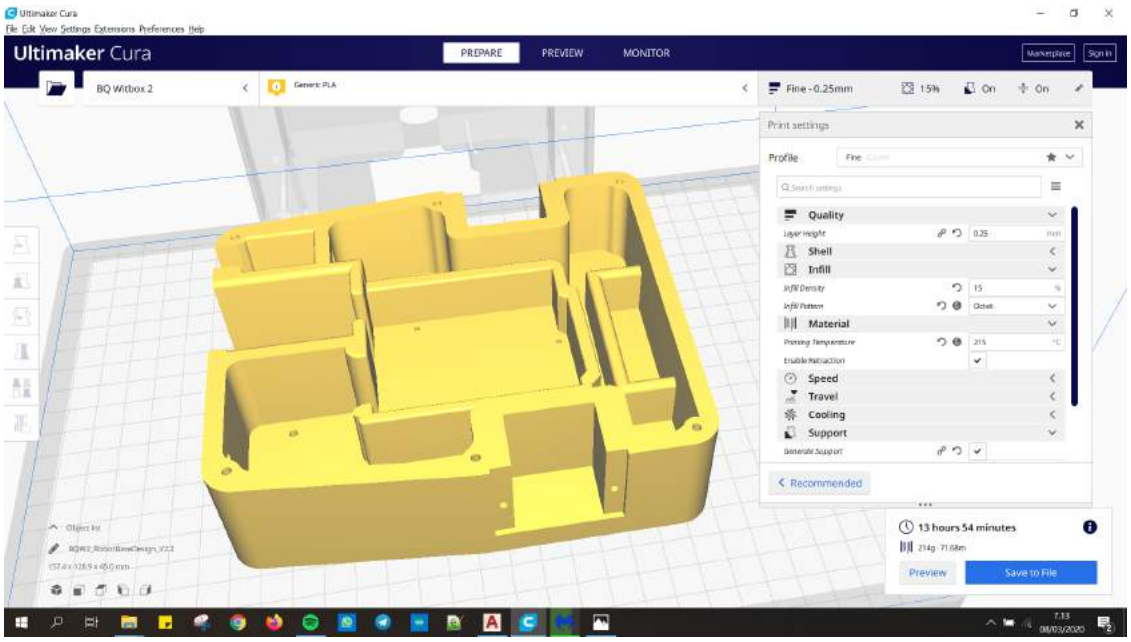

BQ Witbox 2
Generic PLA
Fine - 0.25mm 15% On On

Print settings
Profile Fine

Quality
Layer Height 0.25 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

13 hours 55 minutes
214g · 71.77m

Figure B.8: Printing setups: Chassis V2.0: included hole for the LED of the battery control system, expanded space for the connection of the USB to the WeMos D1 R2, reduction of the walls for a better use of resources and to make the structure lighter, more precise space for the shape of the female power jack connector, added hole for the assembly of the WiFi cover on the space for the USB connection, longer holes for the screws on the ball caster assembly, lifted space for the microprocessor's base and holes for attachment through screws.



Fine - 0.25mm | 15% | On | On

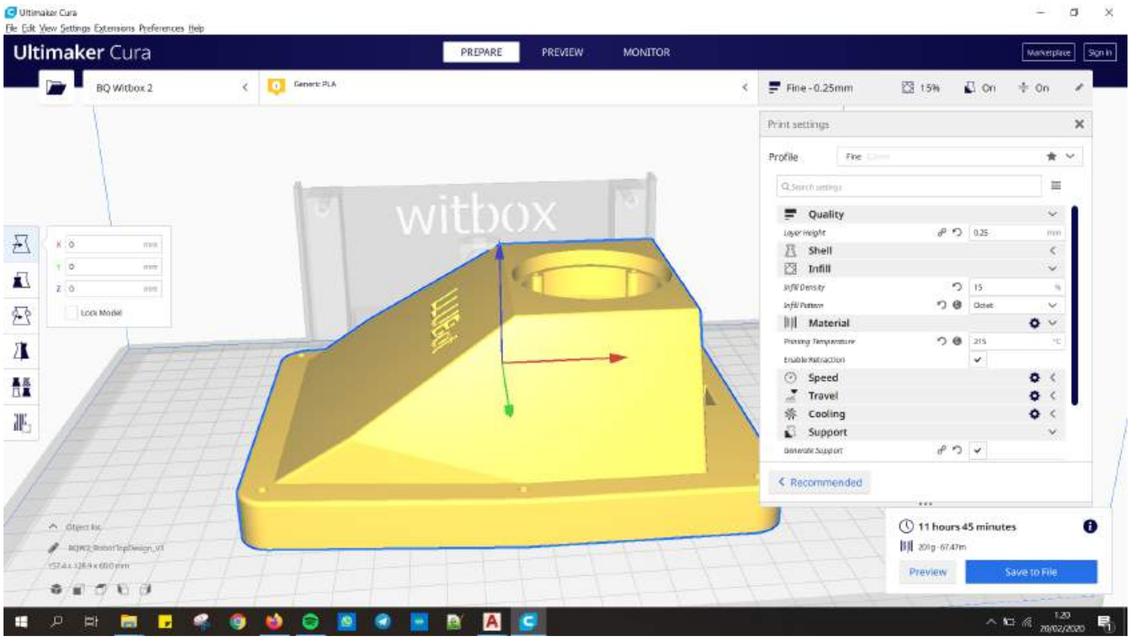
Print settings

Profile: Fine

Quality
Layer Height: 0.25 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

13 hours 54 minutes
214g · 71.68m

Figure B.9: Printing setups: Chassis V2.1: even more precise space for the shape of the female power jack connector than previous.



X 0 mm
Y 0 mm
Z 0 mm

Fine - 0.25mm | 15% | On | On

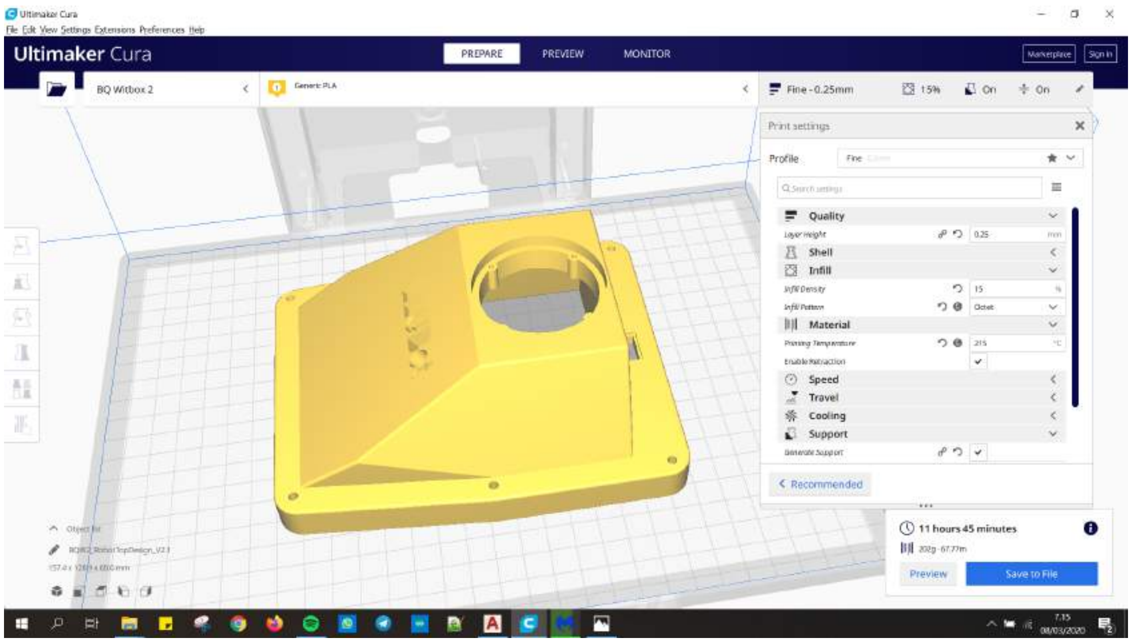
Print settings

Profile: Fine

Quality
Layer Height: 0.25 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

11 hours 45 minutes
201g · 67.47m

Figure B.10: Printing setups: Hubcap V1.



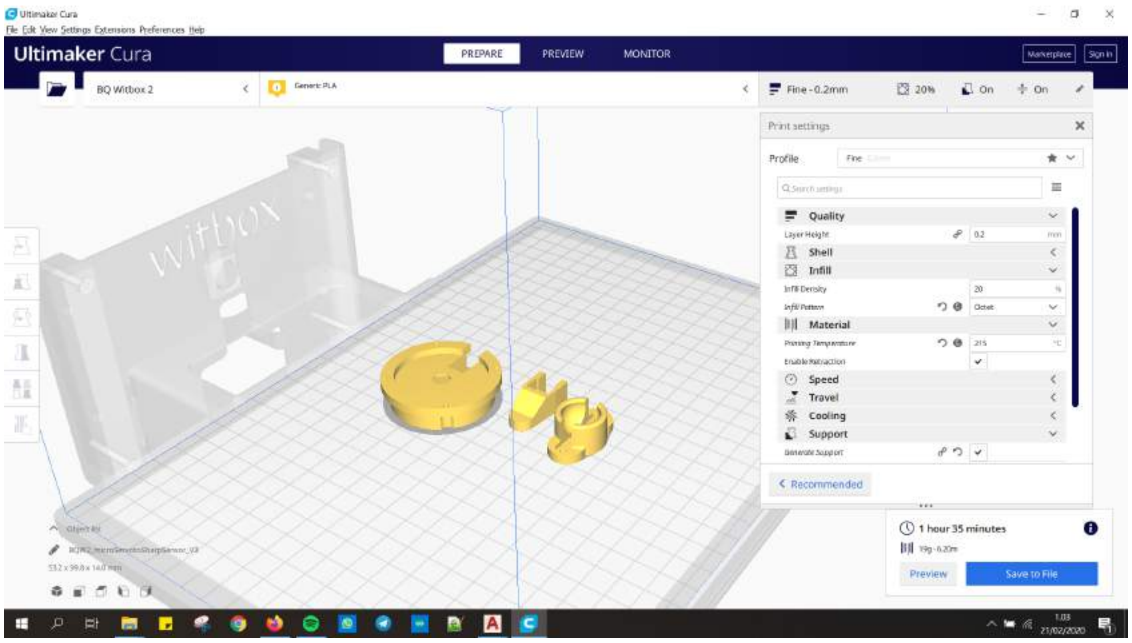
Fine - 0.25mm | 15% | On | On

Print settings

| | |
|---|---|
| Profile | Fine |
| **Quality** | |
| Layer Height | 0.25 mm |
| **Shell** | |
| **Infill** | |
| Infill Density | 15 % |
| Infill Pattern | Octet |
| **Material** | |
| Printing Temperature | 215 °C |
| Enable Retraction | ✓ |
| **Speed** | |
| **Travel** | |
| **Cooling** | |
| **Support** | |
| Generate Support | ✓ |

11 hours 45 minutes
202g · 67.77m

Figure B.11: Printing setups: Hubcap V2: thinner walls and more precise space for the switch.



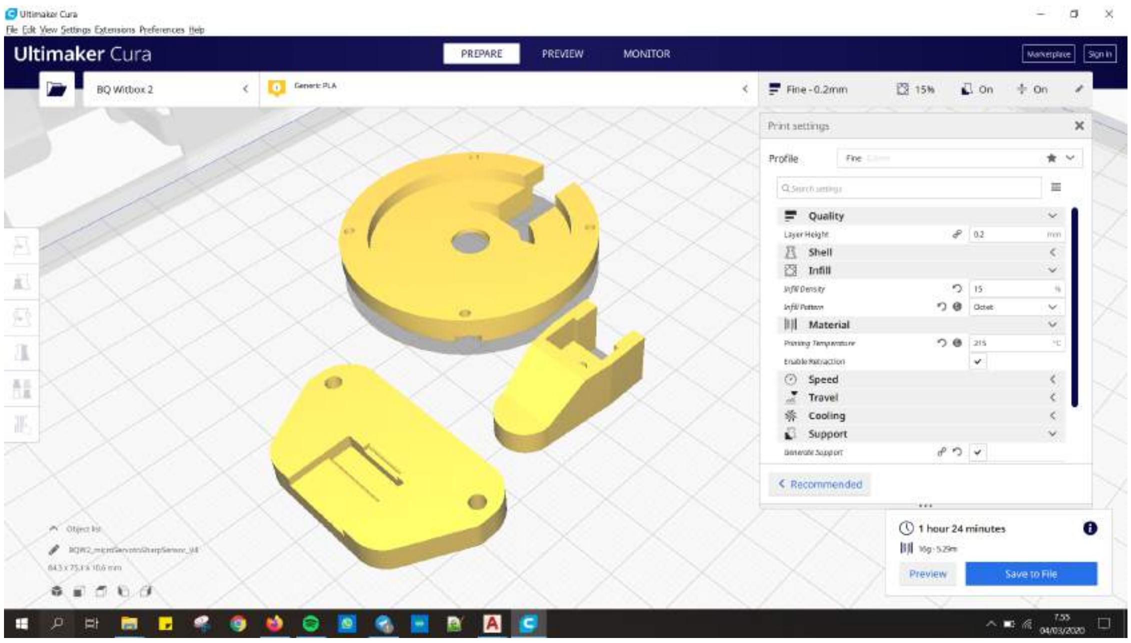
Fine - 0.2mm | 20% | On | On

Print settings

| | |
|---|---|
| Profile | Fine |
| **Quality** | |
| Layer Height | 0.2 mm |
| **Shell** | |
| **Infill** | |
| Infill Density | 20 % |
| Infill Pattern | Octet |
| **Material** | |
| Printing Temperature | 215 °C |
| Enable Retraction | ✓ |
| **Speed** | |
| **Travel** | |
| **Cooling** | |
| **Support** | |
| Generate Support | ✓ |

1 hour 35 minutes
19g · 6.20m

Figure B.12: Printing setups: Sensor motion system V1.

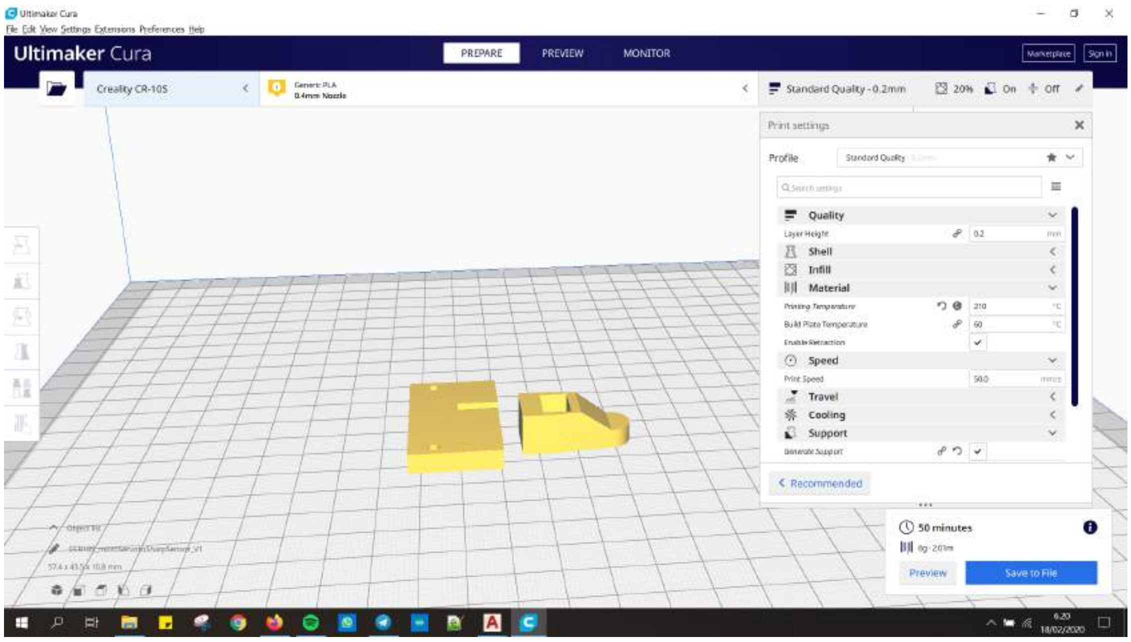


BQ Witbox 2
Generic PLA
Fine - 0.2mm 15% On On

Print settings
Profile Fine
Quality
Layer Height 0.2 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

1 hour 24 minutes
16g · 5.29m

Figure B.13: Printing setups: Sensor motion system V4: the three modified fastener (not final prototype).

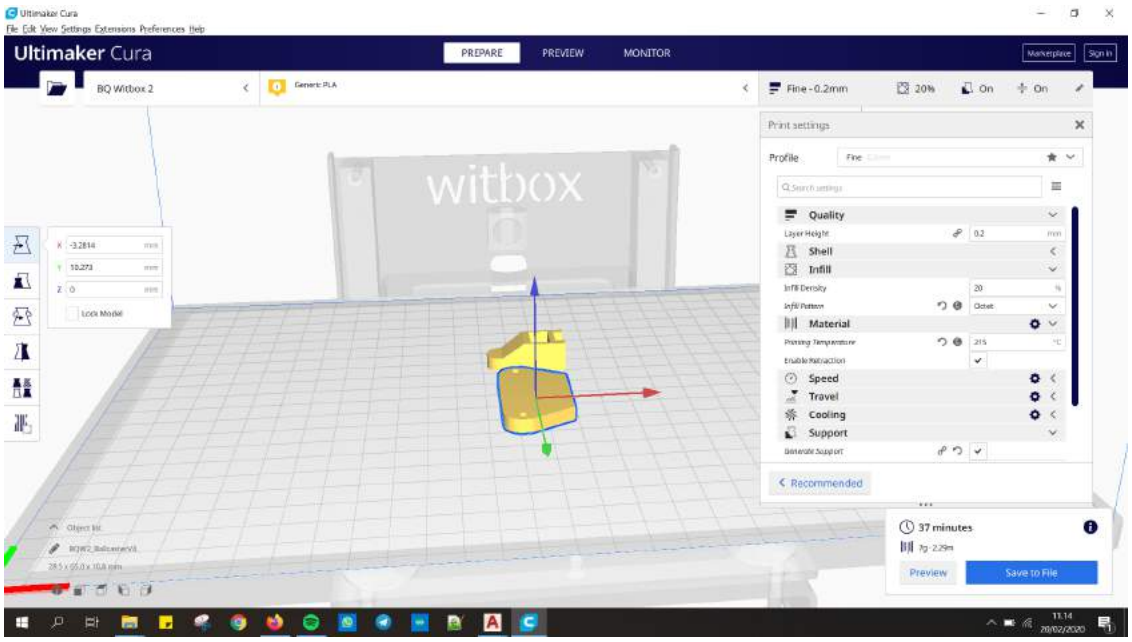


Creality CR-10S
Generic PLA
0.4mm Nozzle
Standard Quality - 0.2mm 20% On Off

Print settings
Profile Standard Quality
Quality
Layer Height 0.2 mm
Shell
Infill
Material
Printing Temperature 210 °C
Build Plate Temperature 60 °C
Enable Retraction ✓
Speed
Print Speed 50.0 mm/s
Travel
Cooling
Support
Generate Support ✓

50 minutes
6g · 2.01m

Figure B.14: Printing setups: Micro servo motor to sharp sensor fastener V1 and sharp sensor cover-fastener V1.



Fine - 0.2mm | 20% | On | On

Print settings

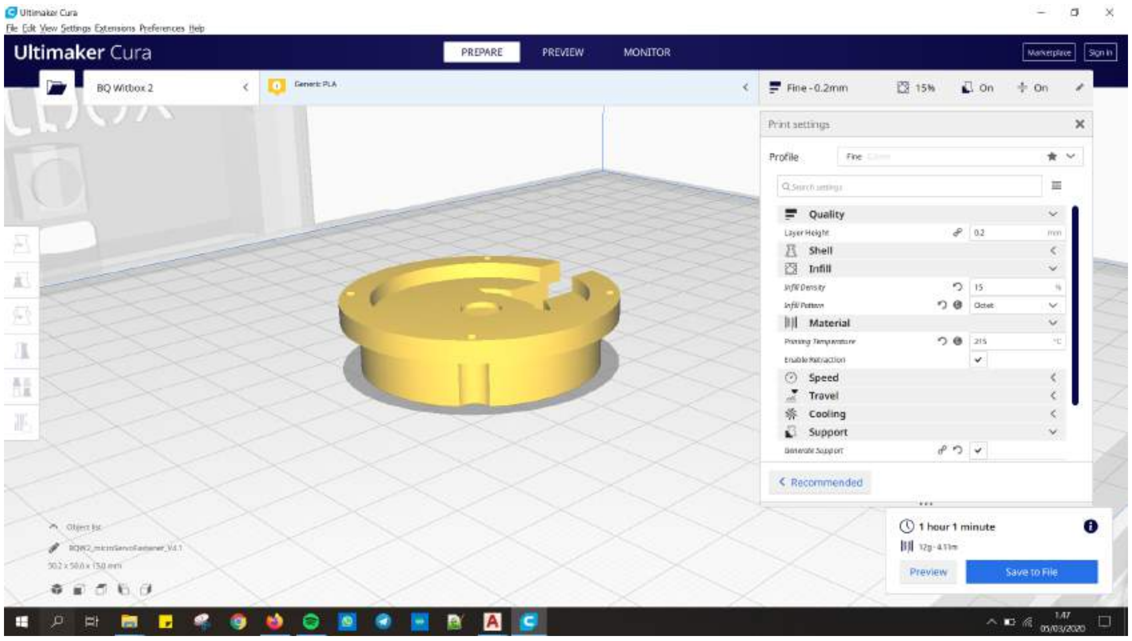
Profile: Fine

| | |
|---|---|
| Quality | |
| Layer Height | 0.2 mm |
| Shell | |
| Infill | |
| Infill Density | 20 % |
| Infill Pattern | Octet |
| Material | |
| Printing Temperature | 215 °C |
| Enable Retraction | ✓ |
| Speed | |
| Travel | |
| Cooling | |
| Support | |
| Generate Support | ✓ |

37 minutes
7g · 2.29m

11:14 28/02/2020

Figure B.15: Printing setups: Micro servo motor to sharp sensor fastener V2 and sharp sensorcover-fastener V2: .



Fine - 0.2mm | 15% | On | On

Print settings

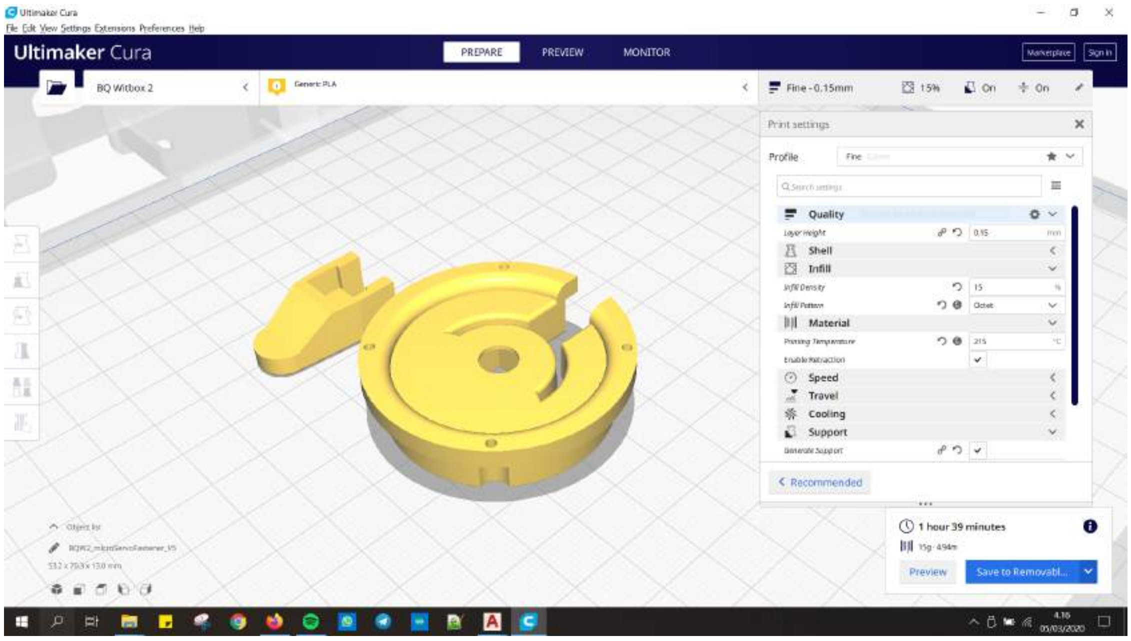
Profile: Fine

| | |
|---|---|
| Quality | |
| Layer Height | 0.2 mm |
| Shell | |
| Infill | |
| Infill Density | 15 % |
| Infill Pattern | Octet |
| Material | |
| Printing Temperature | 215 °C |
| Enable Retraction | ✓ |
| Speed | |
| Travel | |
| Cooling | |
| Support | |
| Generate Support | ✓ |

1 hour 1 minute
12g · 4.11m

1:47 05/03/2020

Figure B.16: Printing setups: Micro servo motor to hubcap fastener V4.

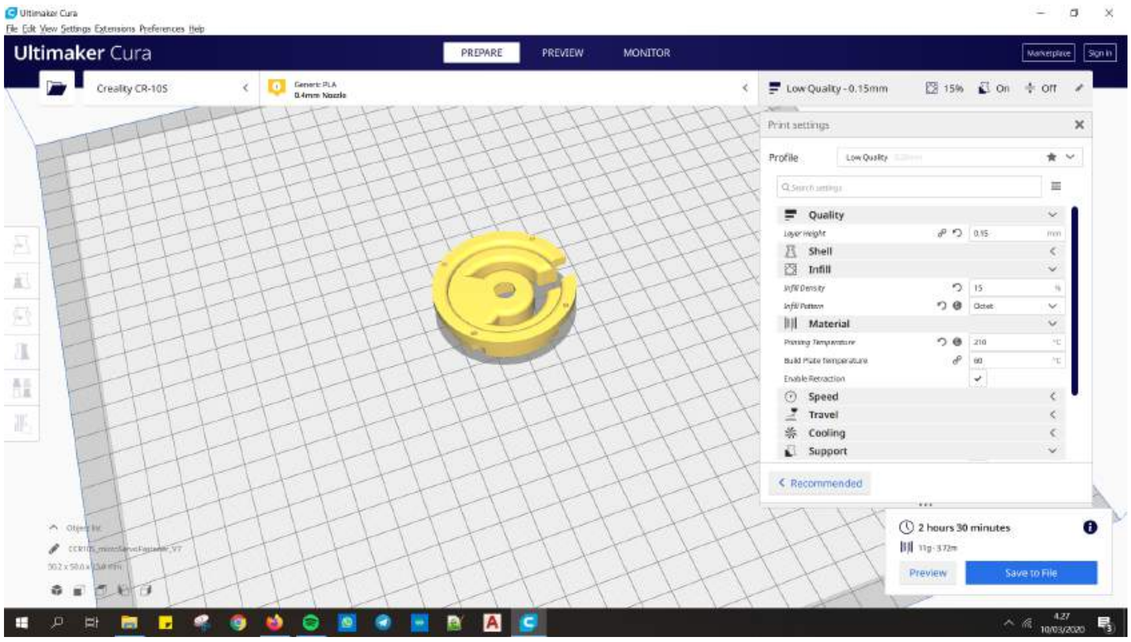


BQ Witbox 2 | Generic PLA | Fine - 0.15mm | 15% | On | On

Print settings
Profile: Fine
Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

1 hour 39 minutes
15g · 4.94m

Figure B.17: Printing setups: Micro servo motor to hubcap fastener V5 and Micro servo motor to sharp sensor fastener V5: more precise attachment than previous versions.

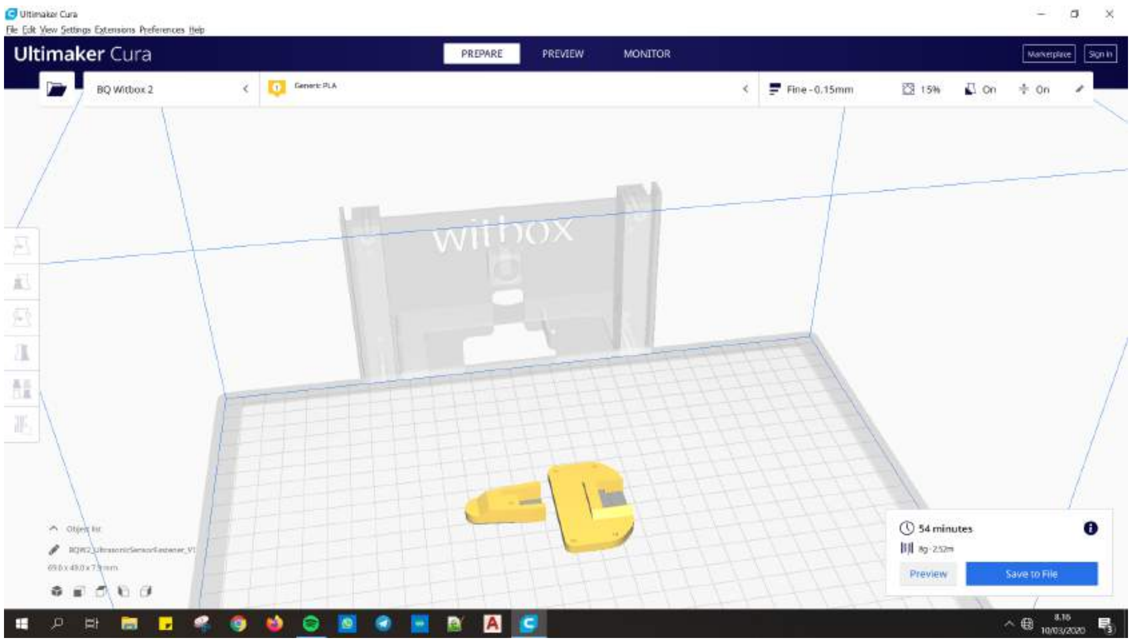


Creality CR-10S | Generic PLA 0.4mm Nozzle | Low Quality - 0.15mm | 15% | On | Off

Print settings
Profile: Low Quality
Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 210 °C
Build Plate Temperature: 60 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support

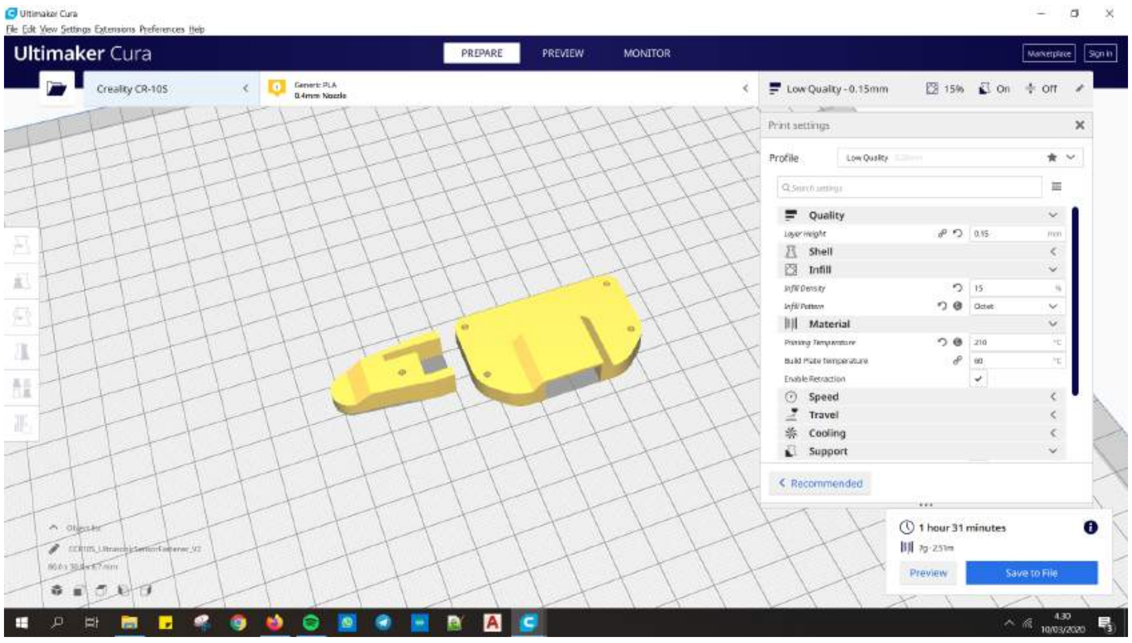
2 hours 30 minutes
11g · 3.72m

Figure B.18: Printing setups: Micro servo motor to hubcap fastener V7: wider and more adequate space for the movement of the sensors wire than previous versions. Useful for both sharp sensor and ultrasonic sensors and their fasteners.



BQ Witbox 2
Generic PLA
Fine - 0.15mm
15%
On
On
54 minutes
8g · 2.52m

Figure B.19: Printing setups: Micro servo motor to ultrasonic sensor fastener V1 and ultrasonic sensor cover-fastener V1.

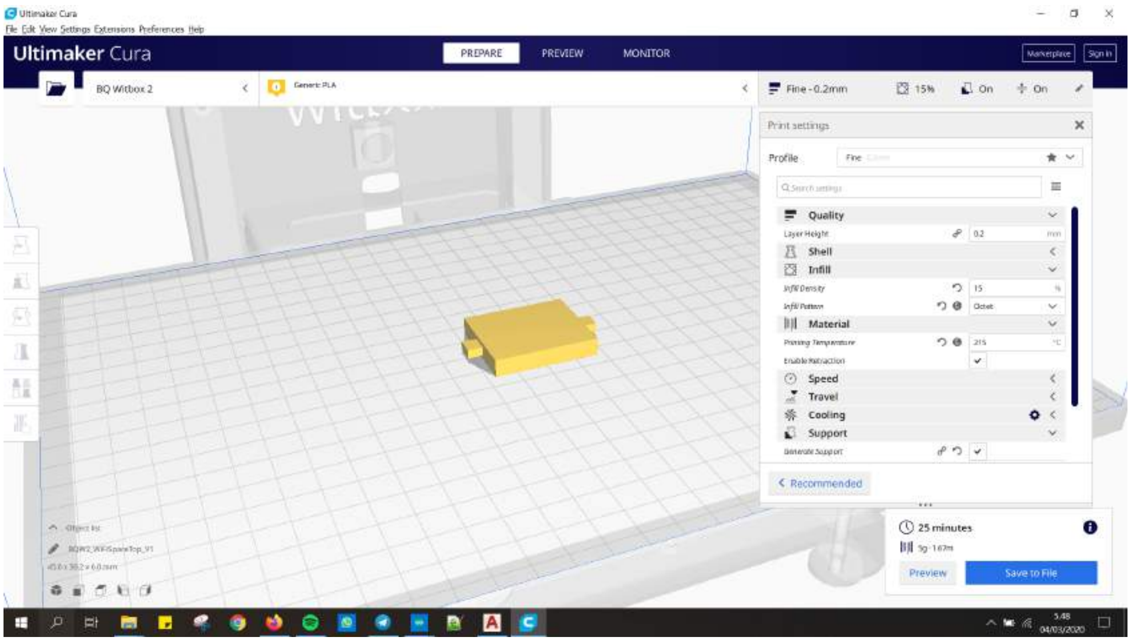


Creality CR-10S
Generic PLA
0.4mm Nozzle
Low Quality - 0.15mm
15%
On
Off

Print settings
Profile: Low Quality
Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 210 °C
Build Plate Temperature: 60 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support

1 hour 31 minutes
7g · 2.51m

Figure B.20: Printing setups: Micro servo motor to ultrasonic sensor fastener V2 and ultrasonic sensor cover-fastener V2: better attachment than previous.



BQ Witbox 2
Generic PLA
Fine - 0.2mm 15% On On

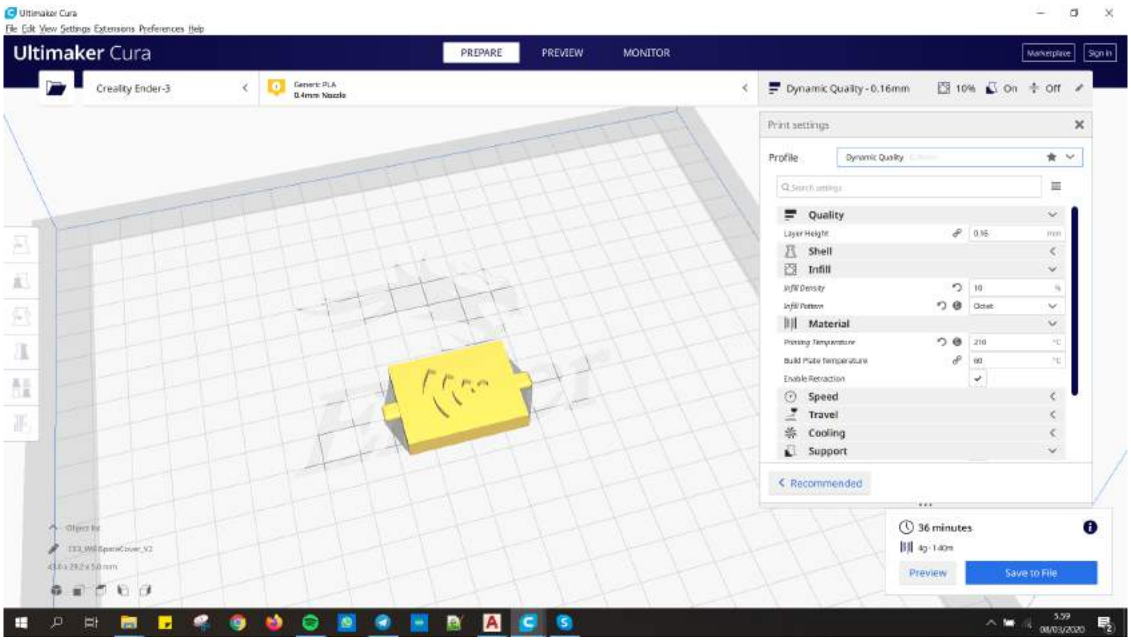
Print settings
Profile Fine
Quality
Layer Height 0.2 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

25 minutes
5g · 1.67m

Object list
BQW2_WiFiSpareTop_V1
45.0 x 38.2 x 6.0 mm

5:48
04/03/2020

Figure B.21: Printing setups: WiFi cover V1.



Creality Ender-3
Generic PLA
0.4mm Nozzle
Dynamic Quality - 0.16mm 10% On Off

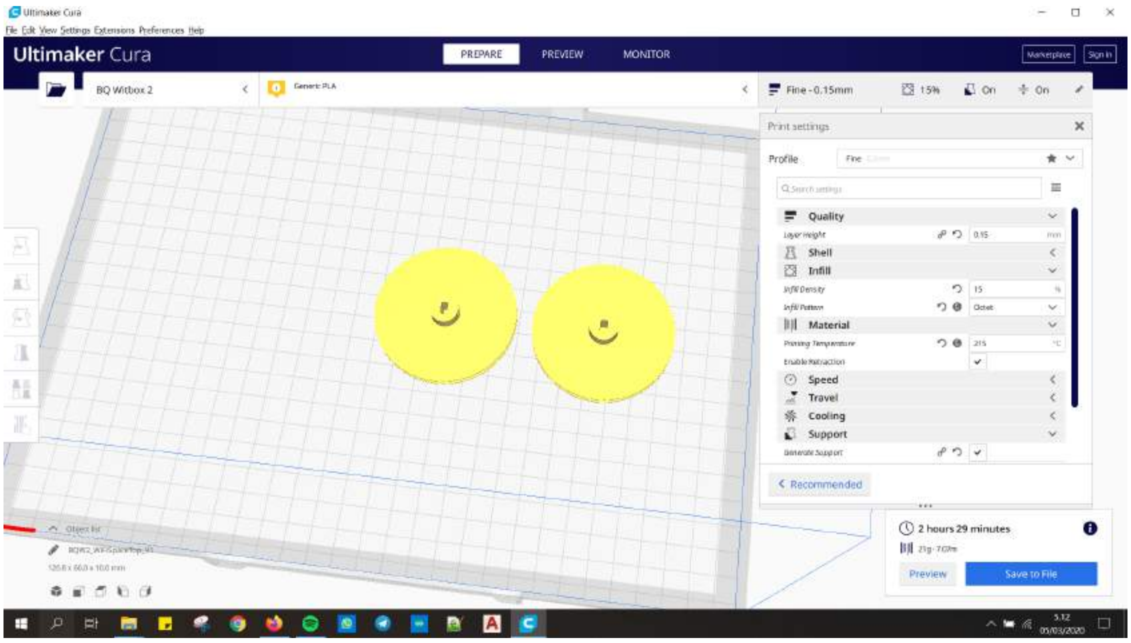
Print settings
Profile Dynamic Quality
Quality
Layer Height 0.16 mm
Shell
Infill
Infill Density 10 %
Infill Pattern Octet
Material
Printing Temperature 210 °C
Build Plate Temperature 60 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support

36 minutes
4g · 1.40m

Object list
CE3_WiFiSpareCover_V2
43.6 x 29.2 x 5.0 mm

5:59
08/03/2020

Figure B.22: Printing setups: WiFi cover V2: slightly rounded the attachment pegs.



Fine - 0.15mm 15% On On

Print settings

Profile: Fine

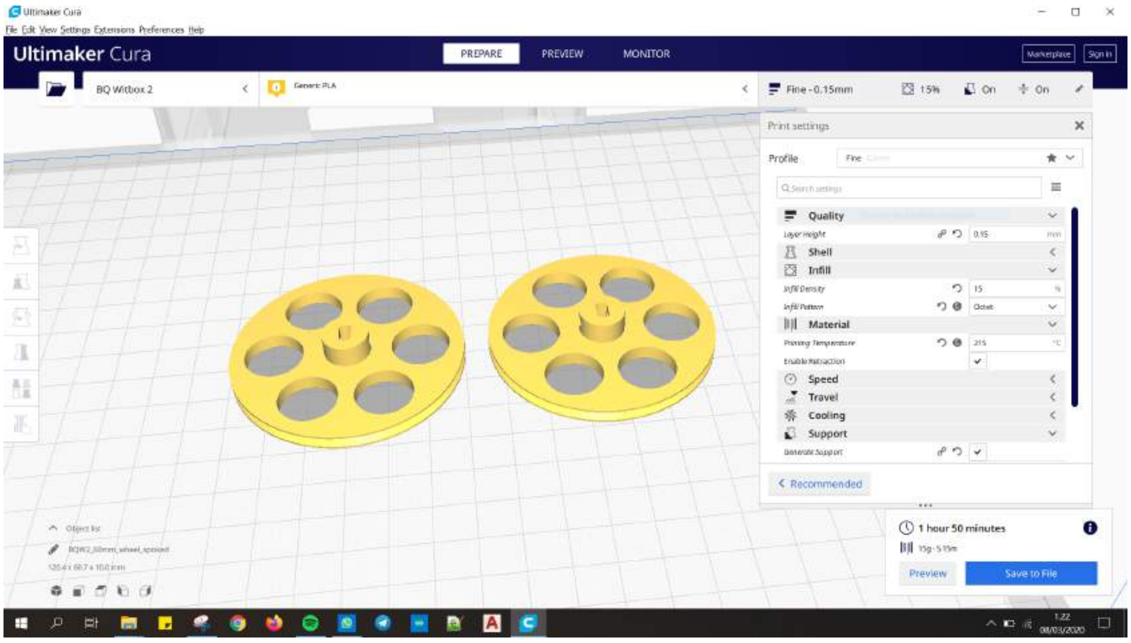
Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

Recommended

2 hours 29 minutes
21g · 7.07m

Preview
Save to File

Figure B.23: Printing setups: Wheels V1 [31].



Fine - 0.15mm 15% On On

Print settings

Profile: Fine

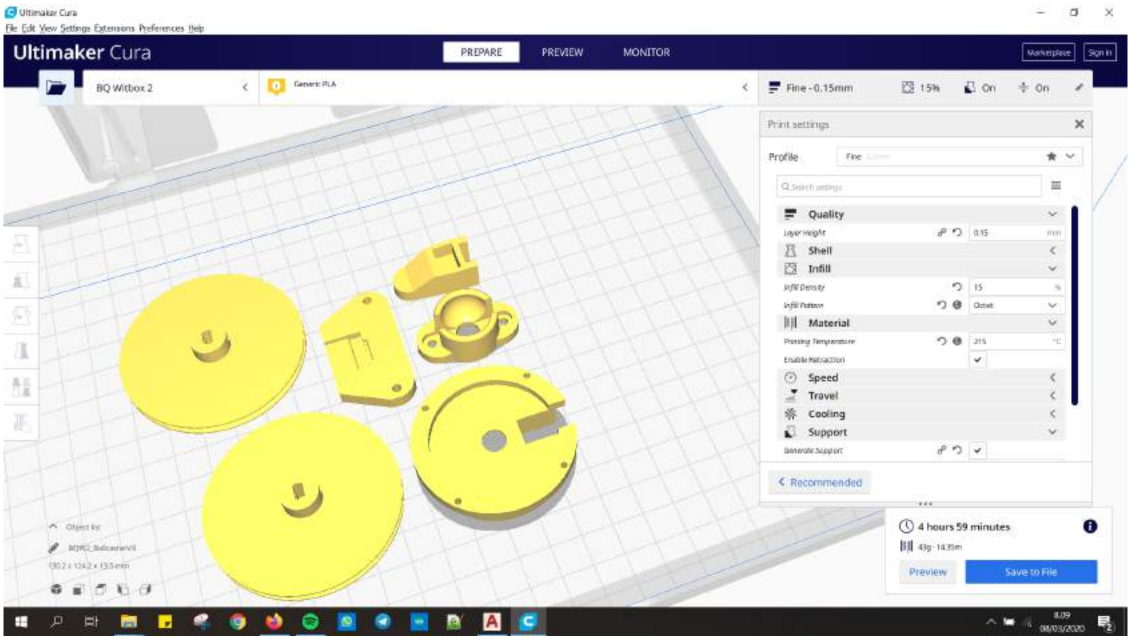
Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

Recommended

1 hour 50 minutes
15g · 5.15m

Preview
Save to File

Figure B.24: Printing setups: Wheels V2 [31].



Fine - 0.15mm 15% On On

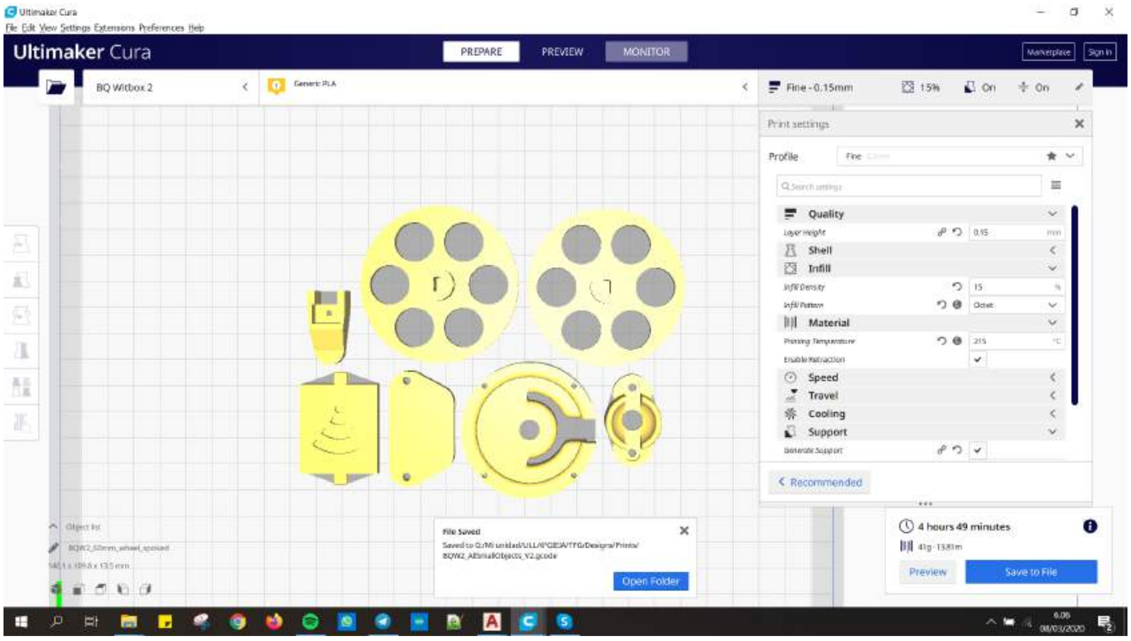
Print settings

Profile Fine

Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

4 hours 59 minutes
43g · 14.35m

Figure B.25: Printing setups: Every small object V1: for the chassis and hubcap version 1 both, and using the sharp sensor.



Fine - 0.15mm 15% On On

Print settings

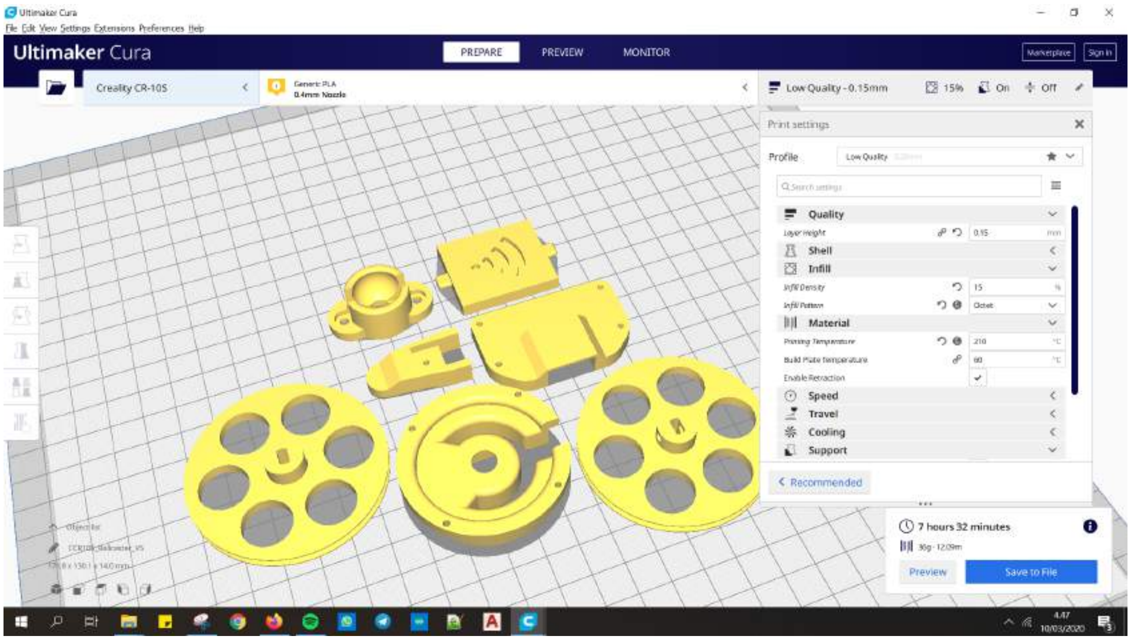
Profile Fine

Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

File Saved
Saved to Q:/Mi unidad/ULL/4ºGEIA/TFG/Designs/Prints/
BQW2_AllSmallObjects_V2.gcode

4 hours 49 minutes
41g · 13.81m

Figure B.26: Printing setups: Every small object V2: for the chassis and hubcap version 2.1 and 2 respectively, and using the sharp sensor.



Creality CR-10S

Generic PLA
0.4mm Nozzle

Low Quality - 0.15mm 15% On Off

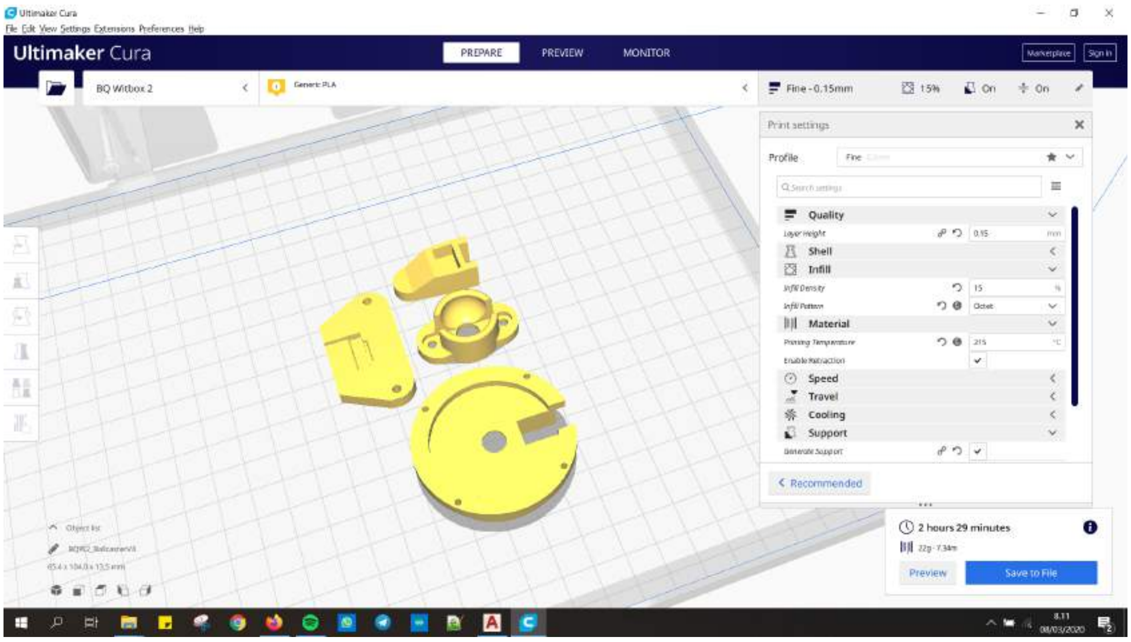
Print settings

Profile: Low Quality

Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 210 °C
Build Plate Temperature 60 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support

7 hours 32 minutes
36g · 12.09m

Figure B.27: Printing setups: Every small object V3: for the chassis and hubcap version 2.1 and 2 respectively, and using the ultrasonic sensor.



BQ Witbox 2

Generic PLA

Fine - 0.15mm 15% On On

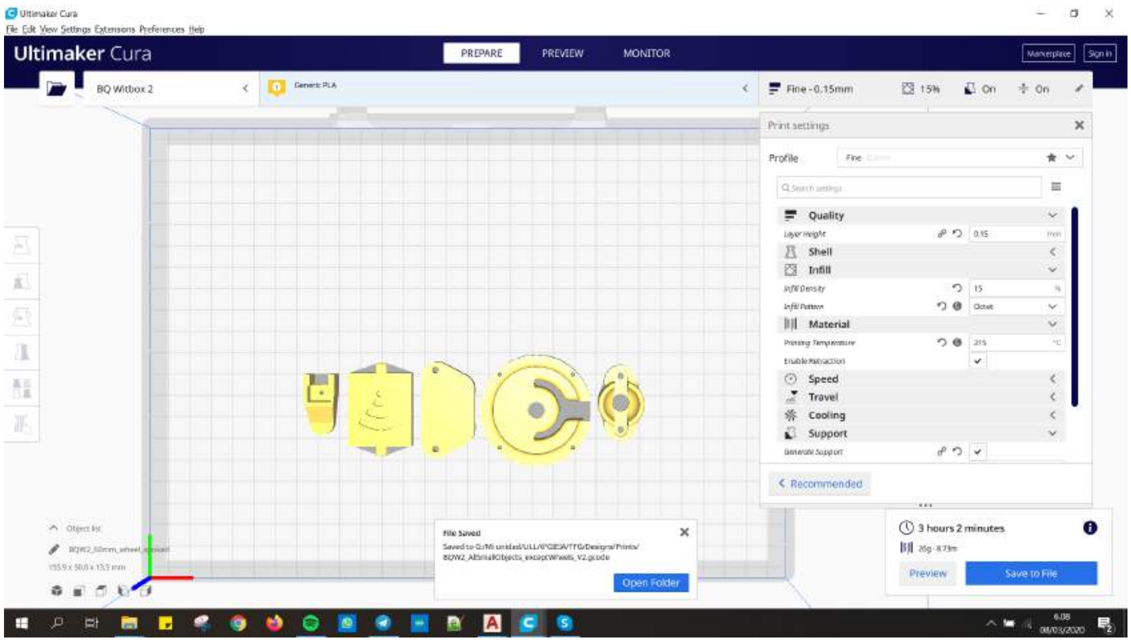
Print settings

Profile: Fine

Quality
Layer Height 0.15 mm
Shell
Infill
Infill Density 15 %
Infill Pattern Octet
Material
Printing Temperature 215 °C
Enable Retraction ✓
Speed
Travel
Cooling
Support
Generate Support ✓

2 hours 29 minutes
22g · 7.34m

Figure B.28: Printing setups: Every small object except the wheels V1: for the chassis and hubcap version 1 both, and using the sharp sensor.



Fine - 0.15mm 15% On On

Print settings

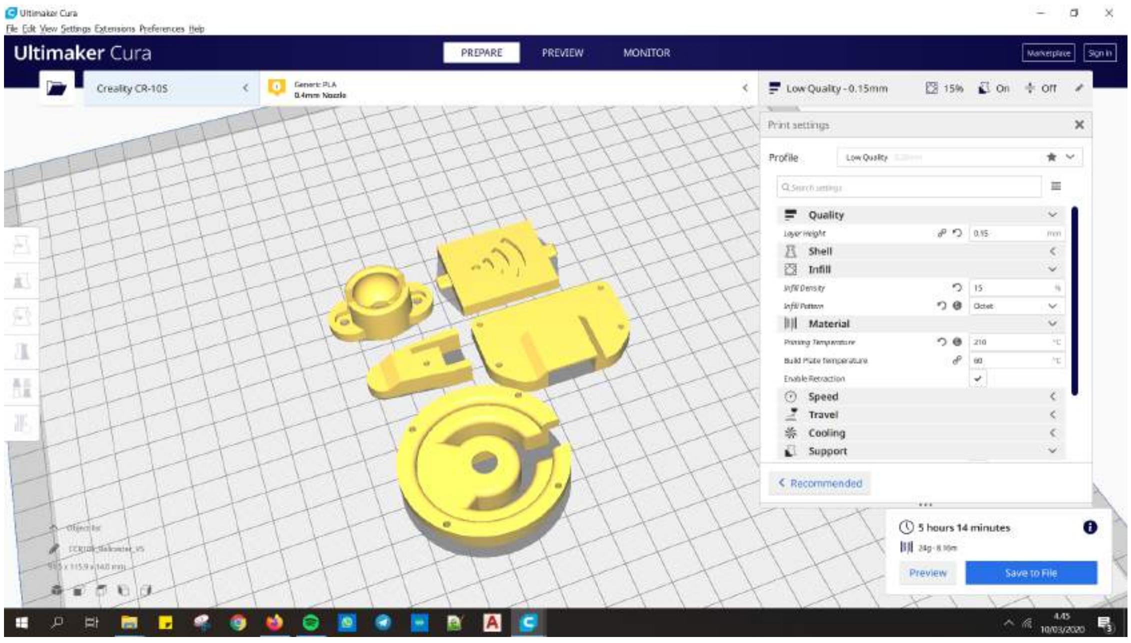
Profile: Fine

Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 215 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support
Generate Support: ✓

3 hours 2 minutes
26g · 8.73m

File Saved
Saved to G:/Mi unidad/ULL/4ºGEIA/TFG/Designs/Prints/BQW2_AllSmallObjects_exceptWheels_V2.gcode

Figure B.29: Printing setups: Every small object except the wheels V2: for the chassis and hubcap version 2.1 and 2 respectively, and using the sharp sensor.



Low Quality - 0.15mm 15% On Off

Print settings

Profile: Low Quality

Quality
Layer Height: 0.15 mm
Shell
Infill
Infill Density: 15 %
Infill Pattern: Octet
Material
Printing Temperature: 210 °C
Build Plate Temperature: 60 °C
Enable Retraction: ✓
Speed
Travel
Cooling
Support

5 hours 14 minutes
24g · 8.16m

Figure B.30: Printing setups: Every small object except the wheels V3: for the chassis and hubcap version 2.1 and 2 respectively, and using the ultrasonic sensor.

# Appendix C

# Codes

All the codes, including libraries, are uploaded to the LLUBot project in GitHub [106].

## C.1 Libraries

First, the two files of the library modified for the control of the steppers through the Arduino motor driver shield in the WeMos D1 R2. Notice that the DMotor_mod.cpp has no modification from the reference of GitHub. The modifications are only made in the file DMotor_mod.h, as the assigned pins had to be exchanged for the fitting of the two stepper motors.

### C.1.1 DMotor_mod.h

```
// Adafruit Motor shield library
// for ESP8266 Wemos D1 R2
#ifndef _AFMotor_h_
#define _AFMotor_h_
#include <inttypes.h>
#if defined(ESP8266)
//#define motorREBUG 1
#include "stdlib_noniso.h"
#define MICROSTEPS 16
#define DC_MOTOR_PWM_RATE 5
#define STEPPER1_PWM_RATE 5
#define STEPPER2_PWM_RATE 5
// Arduino pin names for interface to 74HCT595 latch
// Arduino pins and modifications
#define MOTORLATCH D10 // used to be 12
#define MOTORCLK D4 // used to be 4
#define MOTORENABLE D7 // used to be 7, changed to 13
#define motorRATA D8 // used to be 8, changed to 0
//logic pins for "PWM"
#define MOTOR1_EN D3 // used to be 2
#define MOTOR2_EN D3 // used to be 5
#define MOTOR3_EN D5 // used to be 16
#define MOTOR4_EN D5 // used to be 14
#elif defined(__AVR__)
#include <avr/io.h>
//#define motorREBUG 1
#define MICROSTEPS 16 // 8 or 16
#define MOTOR12_64KHZ _BV(CS20) // no prescale
#define MOTOR12_8KHZ _BV(CS21) // divide by 8
#define MOTOR12_2KHZ _BV(CS21) | _BV(CS20) // divide by 32
#define MOTOR12_1KHZ _BV(CS22) // divide by 64
#define MOTOR34_64KHZ _BV(CS00) // no prescale
#define MOTOR34_8KHZ _BV(CS01) // divide by 8
#define MOTOR34_1KHZ _BV(CS01) | _BV(CS00) // divide by 64
#define DC_MOTOR_PWM_RATE MOTOR34_8KHZ // PWM rate for DC motors
#define STEPPER1_PWM_RATE MOTOR12_64KHZ // PWM rate for stepper 1
#define STEPPER2_PWM_RATE MOTOR34_64KHZ // PWM rate for stepper 2
// Arduino pin names for interface to 74HCT595 latch
#define MOTORLATCH 10
#define MOTORCLK 4
#define MOTORENABLE 7
#define motorRATA 8
//logic pins for "PWM"
#define MOTOR1_EN 3
```

```
#define MOTOR2_EN 3
#define MOTOR3_EN 5
#define MOTOR4_EN 5
#endif
// Bit positions in the 74HCT595 shift register output
#define MOTOR1_A 2
#define MOTOR1_B 3
#define MOTOR2_A 1
#define MOTOR2_B 4
#define MOTOR3_A 5
#define MOTOR3_B 7
#define MOTOR4_A 0
#define MOTOR4_B 6
// Constants that the user passes in to the motor calls
#define FORWARD 1
#define BACKWARD 2
#define BRAKE 3
#define RELEASE 4
// Constants that the user passes in to the stepper calls
#define SINGLE 1
#define DOUBLE 2
#define INTERLEAVE 3
#define MICROSTEP 4
/*
// Arduino pin names for interface to 74HCT595 latch
#define MOTORLATCH 12
#define MOTORCLK 4
#define MOTORENABLE 13 //used to be 7
#define motorRATA 0 //used to be 8
*/
class AFMotorController
{
public:
```

```
AFMotorController(void);
void enable(void);
friend class AF_DCMotor;
void latch_tx(void);
uint8_t TimerInitalized;
};
class AF_DCMotor
{
public:
AF_DCMotor(uint8_t motornum, uint8_t freq =
DC_MOTOR_PWM_RATE);
void run(uint8_t);
void setSpeed(uint8_t);
private:
uint8_t motornum, pwmfreq;
};
class AF_Stepper {
public:
AF_Stepper(uint16_t, uint8_t);
void step(uint16_t steps, uint8_t dir, uint8_t style = SINGLE);
void setSpeed(uint16_t);
uint8_t onestep(uint8_t dir, uint8_t style);
void release(void);
uint16_t revsteps; // # steps per revolution
uint8_t steppernum;
uint32_t usperstep, steppingcounter;
private:
uint8_t currentstep;
};
uint8_t getlatchstate(void);
#endif
```

### C.1.2 DMotor_mod.cpp

```
// Adafruit Motor shield library
// for ESP8266 Wemos D1 R2
#if (ARDUINO >= 100)
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include "DMotor_mod.h"
static uint8_t latch_state;
#if (MICROSTEPS == 8)
uint8_t microstepcurve[] = {0, 50, 98, 142, 180, 212, 236, 250, 255};
#elif (MICROSTEPS == 16)
uint8_t microstepcurve[] = {0, 25, 50, 74, 98, 120, 141, 162, 180, 197, 212, 225, 236, 244, 250, 253, 255};
#endif
AFMotorController::AFMotorController(void) {
TimerInitalized = false;
}
void AFMotorController::enable(void) {
// setup the latch
pinMode(MOTORLATCH, OUTPUT);
pinMode(MOTORENABLE, OUTPUT);
pinMode(motorRATA, OUTPUT);
pinMode(MOTORCLK, OUTPUT);
latch_state = 0;
latch_tx(); // "reset"
//ENABLE_PORT &=  _BV(ENABLE); // enable the chip outputs!
digitalWrite(MOTORENABLE, LOW);
}
void AFMotorController::latch_tx(void) {
uint8_t i;
//LATCH_PORT &=  _BV(LATCH);
```

```
digitalWrite(MOTORLATCH, LOW);
//SER_PORT &= _BV(SER);
digitalWrite(motorRATA, LOW);
for (i=0; i<8; i++) {
//CLK_PORT &= _BV(CLK);
digitalWrite(MOTORCLK, LOW);
if (latch_state & _BV(7-i)) {
//SER_PORT |= _BV(SER);
digitalWrite(motorRATA, HIGH);
} else {
//SER_PORT &= _BV(SER);
digitalWrite(motorRATA, LOW);
}
//CLK_PORT |= _BV(CLK);
digitalWrite(MOTORCLK, HIGH);
}
//LATCH_PORT |= _BV(LATCH);
digitalWrite(MOTORLATCH, HIGH);
}
static AFMotorController MC;
/******************************************
MOTORS
******************************************/
inline void initPWM1(uint8_t freq) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer2A on PB3 (Arduino pin #11)
pinMode(MOTOR1_EN,OUTPUT);
#elif defined(__PIC32MX__)
#if defined(PIC32_USE_PIN9_FOR_M1_PWM)
```

```
// Make sure that pin 11 is an input, since we have tied together 9 and 11
pinMode(9, OUTPUT);
pinMode(11, INPUT);
if (!MC.TimerInitalized)
{ // Set up Timer2 for 80MHz counting fro 0 to 256
T2CON = 0x8000 | ((freq & 0x07) « 4); //
ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=<freq>, T32=0, TCS=0; // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=0, T32=0, TCS=0
TMR2 = 0x0000;
PR2 = 0x0100;
MC.TimerInitalized = true;
}
// Setup OC4 (pin 9) in PWM mode, with Timer2 as timebase
OC4CON = 0x8006; // OC32 = 0, OCTSEL=0, OCM=6
OC4RS = 0x0000;
OC4R = 0x0000;
#else
// If we are not using PWM for pin 11, then just do digital pinMode(11, OUTPUT);
digitalWrite(11, LOW);
#endif
#elif defined(ESP8266)
// adjust this
pinMode(MOTOR1_EN,OUTPUT);
//pinMode(2,OUTPUT);
//digitalWrite(15, LOW);
#else
#error "This chip is not supported!"
#endif
}
inline void setPWM1(uint8_t s) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer2A on PB3 (Arduino pin #11)
OCR2A = s;
#elif defined(ESP8266)
// adjust this
analogWrite(MOTOR1_EN,s);
#else
#error "This chip is not supported!"
#endif
}
inline void initPWM2(uint8_t freq) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer2B (pin 3)
pinMode(MOTOR2_EN, OUTPUT);
#elif defined(__PIC32MX__)
if (!MC.TimerInitalized)
{ // Set up Timer2 for 80MHz counting fro 0 to 256
T2CON = 0x8000 | ((freq & 0x07) « 4); //
ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=<freq>, T32=0, TCS=0; // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=0, T32=0, TCS=0
TMR2 = 0x0000;
PR2 = 0x0100;
MC.TimerInitalized = true;
}
// Setup OC1 (pin3) in PWM mode, with Timer2 as timebase
OC1CON = 0x8006; // OC32 = 0, OCTSEL=0, OCM=6
OC1RS = 0x0000;
OC1R = 0x0000;
#elif defined(ESP8266)
```

```
// adjust this
pinMode(MOTOR2_EN, OUTPUT);
#else
#error "This chip is not supported!"
#endif
}
inline void setPWM2(uint8_t s) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer2A on PB3 (Arduino pin #11)
OCR2B = s;
#elif defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
// on arduino mega, pin 11 is now PB5 (OC1A)
OCR3C = s;
#elif defined(__PIC32MX__)
// Set the OC1 (pin3) PMW duty cycle from 0 to 255
OC1RS = s;
#elif defined(ESP8266)
// adjust this
analogWrite(MOTOR2_EN,s);
#else
#error "This chip is not supported!"
#endif
}
inline void initPWM3(uint8_t freq) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
```

```
// use PWM from timer0A / PD6 (pin 6)
TCCR0A |= _BV(COM0A1) | _BV(WGM00) | _BV(WGM01); // fast PWM, turn on OC0A
//TCCR0B = freq & 0x7;
OCR0A = 0;
#elif defined(__PIC32MX__)
if (!MC.TimerInitalized)
{ // Set up Timer2 for 80MHz counting fro 0 to 256
T2CON = 0x8000 | ((freq & 0x07) « 4); // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=<freq>, T32=0, TCS=0; // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=0, T32=0, TCS=0
TMR2 = 0x0000;
PR2 = 0x0100;
MC.TimerInitalized = true;
}
// Setup OC3 (pin 6) in PWM mode, with Timer2 as timebase
OC3CON = 0x8006; // OC32 = 0, OCTSEL=0, OCM=6
OC3RS = 0x0000;
OC3R = 0x0000;
#elif defined(ESP8266)
// adjust this
pinMode(MOTOR3_EN, OUTPUT);
#else
#error "This chip is not supported!"
#endif
}
inline void setPWM3(uint8_t s) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer0A on PB3 (Arduino pin #6)
OCR0A = s;
```

```
#elif defined(ESP8266)
// adjust this
analogWrite(MOTOR3_EN,s);
#else
#error "This chip is not supported!"
#endif
}
inline void initPWM4(uint8_t freq) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer0B / PD5 (pin 5)
TCCR0A |= _BV(COM0B1) | _BV(WGM00) | _BV(WGM01); // fast PWM, turn on oc0a
//TCCR0B = freq & 0x7;
OCR0B = 0;
#elif defined(__PIC32MX__)
if (!MC.TimerInitalized)
{ // Set up Timer2 for 80MHz counting fro 0 to 256
T2CON = 0x8000 | ((freq & 0x07) « 4); // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=<freq>, T32=0, TCS=0; // ON=1, FRZ=0, SIDL=0, TGATE=0, TCKPS=0, T32=0, TCS=0
TMR2 = 0x0000;
PR2 = 0x0100;
MC.TimerInitalized = true;
}
// Setup OC2 (pin 5) in PWM mode, with Timer2 as timebase
OC2CON = 0x8006; // OC32 = 0, OCTSEL=0, OCM=6
OC2RS = 0x0000;
OC2R = 0x0000;
#elif defined(ESP8266)
// adjust this
```

```
pinMode(MOTOR4_EN, OUTPUT);
#else
#error "This chip is not supported!"
#endif
}
inline void setPWM4(uint8_t s) {
#if defined(__AVR_ATmega8__) ||
defined(__AVR_ATmega48__) ||
defined(__AVR_ATmega88__) ||
defined(__AVR_ATmega168__) ||
defined(__AVR_ATmega328P__)
// use PWM from timer0A on PB3 (Arduino pin #6)
OCR0B = s;
#elif defined(ESP8266)
// adjust this
analogWrite(MOTOR4_EN,s);
#else
#error "This chip is not supported!"
#endif
}
AF_DCMotor::AF_DCMotor(uint8_t num, uint8_t freq) {
motornum = num;
pwmfreq = freq;
MC.enable();
switch (num) {
case 1:
latch_state &=  _BV(MOTOR1_A) &  _BV(MOTOR1_B); // set both motor pins to 0
MC.latch_tx();
initPWM1(freq);
break;
case 2:
latch_state &=  _BV(MOTOR2_A) &  _BV(MOTOR2_B); // set both motor pins to 0 MC.latch_tx();
```

```
initPWM2(freq);
break;
case 3:
latch_state &= _BV(MOTOR3_A) & _BV(MOTOR3_B); // set both motor pins to 0 MC.latch_tx();
initPWM3(freq);
break;
case 4:
latch_state &= _BV(MOTOR4_A) & _BV(MOTOR4_B); // set both motor pins to 0 MC.latch_tx();
initPWM4(freq);
break;
}
}
void AF_DCMotor::run(uint8_t cmd) {
uint8_t a, b; //bit positions in shift register
switch (motornum) {
case 1:
a = MOTOR1_A; b = MOTOR1_B; break;
case 2:
a = MOTOR2_A; b = MOTOR2_B; break;
case 3: a = MOTOR3_A; b = MOTOR3_B; break;
case 4:
a = MOTOR4_A; b = MOTOR4_B; break;
default:
return;
}
switch (cmd) {
case FORWARD:
latch_state |= _BV(a);
latch_state &= _BV(b);
MC.latch_tx();
break;
case BACKWARD:
```

```
latch_state &= _BV(a);
latch_state |= _BV(b);
MC.latch_tx();
break;
case RELEASE:
latch_state &= _BV(a); // A and B both low
latch_state &= _BV(b);
MC.latch_tx();
break;
}
}
void AF_DCMotor::setSpeed(uint8_t speed) {
switch (motornum) {
case 1:
setPWM1(speed); break;
case 2:
setPWM2(speed); break;
case 3:
setPWM3(speed); break;
case 4:
setPWM4(speed); break;
}
}
/******************************************
STEPPERS
******************************************/
AF_Stepper::AF_Stepper(uint16_t steps, uint8_t num) {
MC.enable();
revsteps = steps;
steppernum = num;
currentstep = 0;
if (steppernum == 1) {
latch_state &= _BV(MOTOR1_A) & _BV(MOTOR1_B) &
```

```
_BV(MOTOR2_A) & _BV(MOTOR2_B); // all motor pins to 0
MC.latch_tx();
// enable both H bridges
pinMode(MOTOR1_EN, OUTPUT);
//pinMode(MOTOR2_EN, OUTPUT);
digitalWrite(MOTOR1_EN, HIGH);
//digitalWrite(MOTOR2_EN, HIGH);
// use PWM for microstepping support
//initPWM1(STEPPER1_PWM_RATE);
//initPWM2(STEPPER1_PWM_RATE);
digitalWrite(MOTOR1_EN, HIGH);
digitalWrite(MOTOR2_EN, HIGH);
//setPWM1(255);
//setPWM2(255);
} else if (steppernum == 2) {
latch_state &= _BV(MOTOR3_A) & _BV(MOTOR3_B) &
_BV(MOTOR4_A) & _BV(MOTOR4_B); // all motor pins to 0
MC.latch_tx();
// enable both H bridges
pinMode(MOTOR3_EN, OUTPUT);
//pinMode(MOTOR4_EN, OUTPUT);
digitalWrite(MOTOR3_EN, HIGH);
//digitalWrite(MOTOR4_EN, HIGH);
// use PWM for microstepping support
// use PWM for microstepping support
// initPWM3(STEPPER2_PWM_RATE);
//initPWM4(STEPPER2_PWM_RATE);
digitalWrite(MOTOR1_EN, HIGH);
digitalWrite(MOTOR2_EN, HIGH);
//setPWM3(255);
//setPWM4(255); }
}
void AF_Stepper::setSpeed(uint16_t rpm) {
```

```
usperstep = 60000000 / ((uint32_t)revsteps * (uint32_t)rpm);
steppingcounter = 0;
}
void AF_Stepper::release(void) {
if (steppernum == 1) {
latch_state &= _BV(MOTOR1_A) & _BV(MOTOR1_B) &
 _BV(MOTOR2_A) & _BV(MOTOR2_B); // all motor pins to 0
MC.latch_tx();
} else if (steppernum == 2) {
latch_state &= _BV(MOTOR3_A) & _BV(MOTOR3_B) &
 _BV(MOTOR4_A) & _BV(MOTOR4_B); // all motor pins to 0
MC.latch_tx();
}
}
void AF_Stepper::step(uint16_t steps, uint8_t dir, uint8_t style) {
uint32_t uspers = usperstep;
uint8_t ret = 0;
if (style == INTERLEAVE) {
uspers /= 2;
} else if (style == MICROSTEP) {
uspers /= MICROSTEPS;
steps *= MICROSTEPS;
#ifdef motorREBUG
Serial.print("steps = "); Serial.println(steps, DEC);
#endif
}
while (steps–) {
ret = onestep(dir, style);
delay(uspers/1000); // in ms
steppingcounter += (uspers % 1000);
if (steppingcounter >= 1000) {
delay(1);
steppingcounter -= 1000;
}
}
if (style == MICROSTEP) {
while ((ret != 0) && (ret != MICROSTEPS)) {
ret = onestep(dir, style);
delay(uspers/1000); // in ms
steppingcounter += (uspers % 1000);
if (steppingcounter >= 1000) {
delay(1);
steppingcounter -= 1000;
}
}
}
}
uint8_t AF_Stepper::onestep(uint8_t dir, uint8_t style) {
uint8_t a, b, c, d;
uint8_t ocrb, ocra;
ocra = ocrb = 255;
if (steppernum == 1) {
a = _BV(MOTOR1_A);
b = _BV(MOTOR2_A);
c = _BV(MOTOR1_B);
d = _BV(MOTOR2_B);
} else if (steppernum == 2) {
a = _BV(MOTOR3_A);
b = _BV(MOTOR4_A);
c = _BV(MOTOR3_B);
d = _BV(MOTOR4_B);
} else {
return 0;
}
// next determine what sort of stepping procedure we're up to
if (style == SINGLE) {
```

```
if ((currentstep/(MICROSTEPS/2)) % 2) { // we're at an odd step, weird
if (dir == FORWARD) {
currentstep += MICROSTEPS/2;
}
else {
currentstep -= MICROSTEPS/2;
}
} else { // go to the next even step
if (dir == FORWARD) {
currentstep += MICROSTEPS;
}
else {
currentstep -= MICROSTEPS;
}
}
} else if (style == DOUBLE) {
if (! (currentstep/(MICROSTEPS/2) % 2)) { // we're at an even step, weird
if (dir == FORWARD) {
currentstep += MICROSTEPS/2;
} else {
currentstep -= MICROSTEPS/2;
}
} else { // go to the next odd step
if (dir == FORWARD) {
currentstep += MICROSTEPS;
} else {
currentstep -= MICROSTEPS;
}
}
} else if (style == INTERLEAVE) {
if (dir == FORWARD) {
currentstep += MICROSTEPS/2;
} else {
```

```
currentstep -= MICROSTEPS/2;
}
}
if (style == MICROSTEP) {
if (dir == FORWARD) {
currentstep++;
} else {
// BACKWARDS
currentstep–;
}
currentstep += MICROSTEPS*4;
currentstep %= MICROSTEPS*4;
ocra = ocrb = 0;
if ( (currentstep >= 0) && (currentstep < MICROSTEPS)) {
ocra = microstepcurve[MICROSTEPS - currentstep];
ocrb = microstepcurve[currentstep];
} else if ( (currentstep >= MICROSTEPS) && (currentstep < MICROSTEPS*2))
{
ocra = microstepcurve[currentstep - MICROSTEPS];
ocrb = microstepcurve[MICROSTEPS*2 - currentstep];
} else if ( (currentstep >= MICROSTEPS*2) && (currentstep < MICROSTEPS*3))
{
ocra = microstepcurve[MICROSTEPS*3 - currentstep];
ocrb = microstepcurve[currentstep - MICROSTEPS*2];
} else if ( (currentstep >= MICROSTEPS*3) && (currentstep < MICROSTEPS*4))
{
ocra = microstepcurve[currentstep - MICROSTEPS*3];
ocrb = microstepcurve[MICROSTEPS*4 - currentstep];
}
}
currentstep += MICROSTEPS*4;
currentstep %= MICROSTEPS*4;
#ifdef motorREBUG
Serial.print("current step: "); Serial.println(currentstep, DEC);
Serial.print(" pwmA = "); Serial.print(ocra, DEC);
Serial.print(" pwmB = "); Serial.println(ocrb, DEC);
#endif
if (steppernum == 1) {
/*setPWM1(ocra);
setPWM2(ocrb);*/
if (ocra > 128)
digitalWrite(MOTOR1_EN, HIGH);
else
digitalWrite(MOTOR1_EN, LOW);
if (ocrb > 128)
digitalWrite(MOTOR2_EN, HIGH);
else
digitalWrite(MOTOR2_EN, LOW);
} else if (steppernum == 2) {
//setPWM3(ocra);
//setPWM4(ocrb);
if (ocra > 128)
digitalWrite(MOTOR3_EN, HIGH);
else
digitalWrite(MOTOR3_EN, LOW);
if (ocrb > 128)
digitalWrite(MOTOR4_EN, HIGH);
else
digitalWrite(MOTOR4_EN, LOW);
}
// release all
latch_state &=  a &  b &  c &  d; // all motor pins to 0
//Serial.println(step, DEC);
if (style == MICROSTEP) {
if ((currentstep >= 0) && (currentstep < MICROSTEPS))
latch_state |= a | b;
if ((currentstep >= MICROSTEPS) && (currentstep < MICROSTEPS*2))
latch_state |= b | c;
if ((currentstep >= MICROSTEPS*2) && (currentstep < MICROSTEPS*3))
latch_state |= c | d;
if ((currentstep >= MICROSTEPS*3) && (currentstep < MICROSTEPS*4))
latch_state |= d | a;
} else {
switch (currentstep/(MICROSTEPS/2)) {
case 0:
latch_state |= a; // energize coil 1 only
break;
case 1:
latch_state |= a | b; // energize coil 1+2
break;
case 2: latch_state |= b; // energize coil 2 only
break;
case 3: latch_state |= b | c; // energize coil 2+3
break;
case 4:
latch_state |= c; // energize coil 3 only
break;
case 5:
latch_state |= c | d; // energize coil 3+4
break;
case 6:
latch_state |= d; // energize coil 4 only
break;
case 7:
latch_state |= d | a; // energize coil 1+4
break;
}
}
MC.latch_tx(); return currentstep; }
```

## C.2 Codes

### C.2.1 Check_Movement_Steppers.ino

```
#include <DMotor_mod.h>
//Initializate the two motors Right and Left
AF_Stepper motorR(256, 1);
AF_Stepper motorL(256, 2);
void setup() {
motorR.setSpeed(100);
motorL.setSpeed(100);
void loop() {
// Using single coil steps
motorR.step(5, FORWARD, SINGLE);
motorL.step(5, FORWARD, SINGLE);
}
```

### C.2.2 FunctionGoStraightSerial.ino

```
#include <DMotor_mod.h>
AF_Stepper motorR(256, 1);
AF_Stepper motorL(256, 2);
float wheelRadius = 31.0; //in mm
int chosenSpeed = 100;
char chosenDirection;
float chosenQuantity = 0.0;
void setup() {
Serial.begin(9600); // set up Serial library at 9600 bps
Serial.println("Choose a direction:");
Serial.println("F to go Forward and B to go Backward");
motorR.setSpeed(chosenSpeed);
motorL.setSpeed(chosenSpeed); }
void loop() {
if(Serial.available() > 0){
chosenDirection = Serial.read();
```

```
chosenQuantity = 19.5;
if(chosenDirection == 'F' || chosenDirection == 'B'){
goStraight(chosenDirection, chosenQuantity);
}
Serial.println("Choose a direction:");
Serial.println("F to go Forward and B to go Backward");
}
}
void goStraight (char way, float cm){
char way_ = way;
float cm_ = cm;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = cm_ * 256 * 8 / (2 * 3.1416 * wheelRadius / 10);
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == 'F') {
//Running the robot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("_________________________________________");
}
if(way_ == 'B') {
//Running the robot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("_________________________________________");
}
else {
} }
```

### C.2.3 FunctionsMovementWebServer.ino

```
#include <ESP8266WiFi.h>
#include <DMotor_mod.h>
AF_Stepper motorR(256, 1);
AF_Stepper motorL(256, 2);
float wheelRadius = 31.0; //in mm
int chosenSpeed = 100;
char chosenDirection;
float chosenLenght = 19.5;
float chosenAngle = 90;
const char* ssid = "SSID";
const char* password = "password";
int ledPin = D5;
WiFiServer server(80);
void goStraight (char way, float cm){
char way_ = way;
float cm_ = cm;
//String motorsDirection;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = cm_ * 256 * 8 / (2 * 3.1416 * wheelRadius / 10);
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
```

```
if(way_ == ’F’) {
//motorsDirection = "FORWARD";
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("_____________________________________________");
}
if(way_ == ’B’) {
//motorsDirection = "BACKWARD";
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("___________________________");
}
else {
}
}
void turn (char way, float angle){
char way_ = way;
float angle_ = angle;
//String motorsDirection;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = angle_ * (256 * 8 / (2 * 3.1416 * wheelRadius / 10)) * 2 * 3.1416 *
(16 /2) / 360;
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == 'R') {
//motorsDirection = "FORWARD";
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("________________________");
}
if(way_ == 'L') {
//motorsDirection = "BACKWARD";
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("__________________________");
}
else {
}
}
void setup() {
Serial.begin(115200);
delay(10);
pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, LOW);
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
// Start the server
server.begin();
Serial.println("Server started");
// Print the IP address
Serial.print("Use this URL : ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");
//Steppers speed
motorR.setSpeed(chosenSpeed);
motorL.setSpeed(chosenSpeed);
}
void loop() {
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
return;
}
// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
delay(1);
}
// Read the first line of the request
```

```
String request = client.readStringUntil(’˚’);
Serial.println(request);
client.flush();
// Match the request
int value = 0;
if (request.indexOf("/MOTOR=FORWARD") != -1) {
//goStraight(’F’, chosenLenght);
value = 1;
}
if (request.indexOf("/MOTOR=BACKWARD") != -1) {
//goStraight(’B’, chosenLenght);
value = 2;
}
if (request.indexOf("/MOTOR=RIGHT") != -1) {
//goStraight(’R’, chosenAngle);
value = 3;
}
if (request.indexOf("/MOTOR=LEFT") != -1) {
//goStraight(’L’, chosenAngle);
value = 4;
}
// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
//client.print("The LLUBot waits for your instruction:");
client.print("The LLUBot is now: ");
if(value == 1) {
client.print("Running Forward");
} if (value == 2) {
client.print("Running Backward");
} if (value == 3) {
client.print("Turning Right");
} if (value == 4) {
client.print("Turning Left");
} else {
client.print("Stopped");
}
client.println("<br><br>");
client.println("Click <a href=/̈MOTOR=FORWARD>̈here</a> to run the LLUBot forward 19.5 cm<br>");
client.println("Click <a href=/̈MOTOR=BACKWARD>̈here</a> to run the LLUBot backward 19.5 cm<br>");
client.println("Click <a href=/̈MOTOR=RIGHT>̈here</a> to turn the LLUBot 90 degrees to its right<br>");
client.println("Click <a href=/̈MOTOR=LEFT>̈here</a> to turn the LLUBot 90 degrees to its left<br>");
client.println("</html>");
delay(1);
Serial.println("Client disconnected");
Serial.println("");
if(value == 1) {
goStraight('F', chosenLenght);
} if(value == 2) {
goStraight('B', chosenLenght);
} if(value == 3) {
turn('R', chosenAngle);
} if(value == 4) {
turn('L', chosenAngle);
} else {
}
}
```

### C.2.4 BatteryDropControl_basic.ino

```
#define ledPin D2
```

```
#define voltagePin D9
void setup() {
//start serial connection
Serial.begin(115200);
pinMode(voltagePin, INPUT);
pinMode(ledPin, OUTPUT);
}
void loop() {
//read the pushbutton value into a variable
int sensorVal = digitalRead(voltagePin);
//print out the value of the pushbutton
Serial.println(!sensorVal);
// Keep in mind the pull-up means the input pin's logic is inverted. It goes
// HIGH when it's open, and LOW when it's pressed. Turn on pin the LED
in D2 when the
// voltage has dropped, and off when it has not:
if (sensorVal == 0) {
digitalWrite(ledPin, LOW);
} else {
digitalWrite(ledPin, HIGH);
}
}
```

### C.2.5 FunctionsMovementAndBatteryWebServer.ino

```
#include <ESP8266WiFi.h>
#include <DMotor_mod.h>
#define ledPin D2
#define voltagePin D6
AF_Stepper motorR(256, 1);
AF_Stepper motorL(256, 2);
float wheelRadius = 31.0; //in mm
int chosenSpeed = 100;
char chosenDirection;
```

```
float chosenLenght = 19.5;
float chosenAngle = 90;
const char* ssid = "SSID";
const char* password = "password";
WiFiServer server(80);
void goStraight (char way, float cm){
char way_ = way;
float cm_ = cm;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = cm_ * 256 * 8 / (2 * 3.1416 * wheelRadius / 10);
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == 'F') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("__________________");
}
if(way_ == 'B') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("_________________");
}
else {
}
}
void turn (char way, float angle){
char way_ = way;
float angle_ = angle;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = angle_ * (256 * 8 / (2 * 3.1416 * wheelRadius / 10)) * 2 * 3.1416 * (16 /2) / 360;
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == 'R') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("____________________");
}
if(way_ == 'L') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("______________________");
}
```

```
else {
}
}
void setup() {
Serial.begin(115200);
delay(10);
pinMode(voltagePin, INPUT);
pinMode(ledPin, OUTPUT);
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
// Start the server server.begin(); Serial.println("Server started");
// Print the IP address
Serial.print("Use this URL : ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");
//Steppers speed
motorR.setSpeed(chosenSpeed);
motorL.setSpeed(chosenSpeed);
}
void loop() {
// First read the voltage value for the correct functioning of the LLUBot.
int sensorVal = digitalRead(voltagePin);
```

```
if (sensorVal == 1) {
digitalWrite(ledPin, LOW);
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
return;
}
// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
delay(1);
}
// Read the first line of the request
String request = client.readStringUntil('˚');
Serial.println(request);
client.flush();
// Match the request
int value = 0;
if (request.indexOf("/MOTOR=FORWARD") != -1) {
value = 1;
}
if (request.indexOf("/MOTOR=BACKWARD") != -1) {
value = 2;
}
if (request.indexOf("/MOTOR=RIGHT") != -1) {
value = 3;
}
if (request.indexOf("/MOTOR=LEFT") != -1) {
value = 4;
}
// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
//client.print("The LLUBot waits for your instruction:");
client.print("The LLUBot is now: ");
if(value == 1) {
client.print("Running Forward");
} if (value == 2) {
client.print("Running Backward");
} if (value == 3) {
client.print("Turning Right");
} if (value == 4) {
client.print("Turning Left");
} else {
client.print("Stopped");
}
client.println("<br><br>");
client.println("Click <a href=¨/MOTOR=FORWARD¨>here</a> to run the
LLUBot forward 19.5 cm.<br>");
client.println("Click <a href=¨/MOTOR=BACKWARD¨>here</a> to run
the LLUBot backward 19.5 cm.<br>");
client.println("Click <a href=¨/MOTOR=RIGHT¨>here</a> to turn the LLUBot
90 degrees to its right.<br>");
client.println("Click <a href=¨/MOTOR=LEFT¨>here</a> to turn the LLUBot
90 degrees to its left.<br>");
client.println("<br><br>");
if(sensorVal==0){
client.println("The battery has to be charged.");
client.println("</html>");
}
else {
client.println("</html>");
}
delay(1);
```

```
Serial.println("Client disconnected");
Serial.println("");
if(value == 1) {
goStraight(’F’, chosenLenght);
} if(value == 2) {
goStraight(’B’, chosenLenght);
} if(value == 3) {
turn(’R’, chosenAngle);
} if(value == 4) {
turn(’L’, chosenAngle);
} else {
}
} else {
digitalWrite(ledPin, HIGH);
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
return;
}
// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
delay(1);
}
// Read the first line of the request
String request = client.readStringUntil(’˚’);
Serial.println(request);
client.flush();
// Match the request
int value = 0;
if (request.indexOf("/MOTOR=BATTERY") != -1) {
// void because there is no action needed
}
```

```
// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<br><br>");
client.println("The battery has to be charged.");
client.println("</html>");
delay(1);
Serial.println("Client disconnected");
client.println("Click <a href=¨/MOTOR=BATTERY¨>here</a> to recharge
the page.<br>");
Serial.println("");
}
}
```

### C.2.6 EducationalWebServer.ino

```
#include <ESP8266WiFi.h>
#include <DMotor_mod.h>
#define ledPin D2
#define voltagePin D6
AF_Stepper motorR(256, 1);
AF_Stepper motorL(256, 2);
float wheelRadius = 31.0; //in mm
int chosenSpeed = 100;
char chosenDirection;
float chosenLenght = 19.5;
float chosenAngle = 90;
const char* ssid = "SSID";
const char* password = "password";
WiFiServer server(80);
void goStraight (char way, float cm){
```

```
char way_ = way;
float cm_ = cm;
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = cm_ * 256 * 8 / (2 * 3.1416 * wheelRadius / 10);
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == 'F') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("_____________________");
}
if(way_ == 'B') {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" "); Serial.println("_____________________");
}
else {
}
}
void turn (char way, float angle){
char way_ = way;
float angle_ = angle;
```

```
int steps = 0;
Serial.println(chosenDirection);
Serial.println();
//Choosing the cm to introduce as steps in the motor motion:
steps = angle_ * (256 * 8 / (2 * 3.1416 * wheelRadius / 10)) * 2 * 3.1416 *
(16 /2) / 360;
Serial.println(steps);
Serial.println();
//Choosing a direction for the motors:
if(way_ == ’R’) {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, BACKWARD, SINGLE);
motorL.step(1, FORWARD, SINGLE);
}
Serial.print(" ");
Serial.println("______________________");
}
if(way_ == ’L’) {
//Running the LLUBot:
for(int i = 0; i < steps; i++){
motorR.step(1, FORWARD, SINGLE);
motorL.step(1, BACKWARD, SINGLE);
}
Serial.print(" ");
Serial.println("______________________");
}
else {
}
}
void setup() {
Serial.begin(115200);
delay(10);
pinMode(voltagePin, INPUT);
pinMode(ledPin, OUTPUT);
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
// Start the server server.begin(); Serial.println("Server started");
// Print the IP address
Serial.print("Use this URL : ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");
//Steppers speed
motorR.setSpeed(chosenSpeed);
motorL.setSpeed(chosenSpeed);
}
void loop() {
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
return;
}
// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
delay(1);
}
// Read the first line of the request
String request = client.readStringUntil(’’̊);
Serial.println(request);
client.flush();
// Match the request
int value = 0;
if (request.indexOf("/MOTOR=FORWARD") != -1) {
value = 1;
}
if (request.indexOf("/MOTOR=BACKWARD") != -1) {
value = 2;
}
if (request.indexOf("/MOTOR=RIGHT") != -1) {
value = 3;
}
if (request.indexOf("/MOTOR=LEFT") != -1) {
value = 4;
}
// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE html>");
client.println("<html lang=’es’>");
client.println(" <head>");
client.println(" <title> LLUBot </title>");
client.println(" <style>");
client.println(" body { background-color: #7c05a3; text-align: center; font:
Verdana; font-size: 25px; color: white;}");
client.println(" .main button { height: 50px; width: 50%; font-size: 18px; }");
client.println(" .main { display: block; }");
client.println(" .app1 { display: none; }");
client.println(" .app2 { display: none; }");
```

```
        client.println(" button { font-size: 18px; }");
        client.println(" </style>");
        client.println(" </head>");
        client.println(" <body>");
        client.println(" <div class='main'>");
        client.println(" <h1> LLUBot learning machine </h1><br>");
        client.println(" <button id='btn1'> 1. Introduction </button><br><br>");
        client.println(" <button id='btn2'> 2. Let's learn applied maths </but-
ton><br><br>");
        client.println(" </div>");
        client.println(" <div class='app1'>");
        client.println(" <h1> Introduction </h1><br>");
        client.println(" <p> This robot is a prototype for the bachelor degree final-
ization of the Electronics Industrial and Automatical Engineering Degree of Luisana
Lara. It is thought to be used in educational environments, as a tool for learning
concepts of Electronics, Mechanical Design, Programming, as well as others that are
included in the discipline of Robotics.</p>");
        client.println(" <button class='back'> Inicio </button>");
        client.println(" </div>");
        client.println(" <div class='app2'>");
        client.println(" <h1> First activity: application of Maths </h1><br>");
        client.println(" <p> When we are first tought of geometry in school, we
never think that they can be really used in our daily life. And as a surprise, when
using technology, all the engineers and technicians have to know a bit of geometry
in order to make a Roomba useful, a videogame fun and a speaker easy to carry
around, for example. </p><br>");
        client.println(" <p> It is soon when we first knew the geometry of a circle,
as the area and the perimeter depending on the radius of the circle, as seen in the
picture below. </p>");
        client.println(" <img src='https://i1.wp.com/www.aplustopper.com/wp-content/uploads/20
of-a-circle-2.jpeg?resize=869%2C428&ssl=1'><br>");
        client.println(" <p> Did you know that the perimeter or circumference is
necessary for the localization of a mobile robot as a Roomba (those cleaning robots
famous for carrying cats around the house randomly)? Well, a mobile robot needs to
know where it is in order to be independent, which is fundamental for this types of
robots. But the real question is why and how the perimeter is used. </p><br>");
        client.println(" <h2> Motors: brief introduction </h2><br>");
        client.println(" <p> The fun thing about us humans moving around in a
```

place is that, generally speaking, we automatically do it with our legs. But imagine we had to build ourselves into a moving machine: then we'd have to choose our 'legs', and this means two motors, one for each leg. </p>");

```
client.println(" <button class='back'> Inicio </button>");
client.println(" </div>");
client.println(" <script>");
client.println(" let main = document.querySelector('.main');");
client.println(" let app1 = document.querySelector('.app1');");
client.println(" let app2 = document.querySelector('.app2');");
client.println(" let btn1 = document.getElementById('btn1');");
client.println(" let btn2 = document.getElementById('btn2');");
client.println(" let btnBack = document.querySelectorAll('.back');");
client.println(" btn1.addEventListener('click', () => {");
client.println(" main.style.display = 'none';");
client.println(" app1.style.display = 'block';");
client.println(" });");
client.println(" btn2.addEventListener('click', () => {");
client.println(" main.style.display = 'none';");
client.println(" app2.style.display = 'block';");
client.println(" });");
client.println(" btnBack.forEach( btn => {");
client.println(" btn.addEventListener('click', () => {");
client.println(" main.style.display = 'block';");
client.println(" app1.style.display = 'none';");
client.println(" app2.style.display = 'none';");
client.println(" });");
client.println(" });");
client.println(" </script>");
client.println(" </body>");
client.println("</html>");
delay(1); Serial.println("Client disconnected");
Serial.println("");
if(value == 1) {
goStraight('F', chosenLenght);
} if(value == 2) {
goStraight('B', chosenLenght);
} if(value == 3) {
turn('R', chosenAngle);
} if(value == 4) {
turn('L', chosenAngle);
} else {
}
}
```

# Bibliography

[1] Amazon. 518hrRnwjyL.jpg (Imagen JPEG, 500 × 376 píxeles).

[2] Makeblock. Robot Kits for Kids : mBot | Makeblock – Global STEAM Education Solution Provider.

[3] Wonder Workshop. Robots - Wonder Workshop - US.

[4] Make:. Which Board is Right for Me? | Make:.

[5] Arduino.cc. Pinout-UNO Rev3.

[6] Arduino Official Store. Arduino UNO Rev3 || Arduino Official Store.

[7] Scribd. Arduino Mega Pinout Diagram.

[8] Arduino.cc. Arduino Mega Schematic.

[9] RobotDyn. WIFI D1 R2 ESP8266 dev. board, 32M flash - RobotDyn.

[10] WeMos.cc. WeMos D1 R2 Schematic.

[11] K-Electrónica. Motor más rueda para rover arduino.

[12] Wikipedia. DC motor.

[13] Wikipedia. Stepper motor.

[14] K-Electrónica. Servo de rotación continua BQ Arduino Robot robotica educativa.

[15] K-Electrónica. MiniServo robot BQ Robotica educativa Arduino.

[16] Adafruit. Overview | Adafruit Motor Shield | Adafruit Learning System.

[17] MCU on Eclipse. Tutorial: Arduino Motor/Stepper/Servo Shield – Part 1: Servos.

[18] Circuit Digest. Arduino Motor Driver Shield.

[19] Circuit Digest. What is Shift Register? Working, Applications & Types of Shift Registers.

[20] Wikipedia. H Bridge.

[21] Pinterest. 16 Piece Raspberry Pi /Arduino Sensor Module Set.

[22] Encoder Products Company. What IS an encoder? > Encoder Products.

[23] Java T Point. Iot Project using Ultrasonic Sensor Arduino Distance calculation - Javatpoint.

[24] Electrónica Embajadores. Sharp GP2Y0A21YK0F - Sensor de Proximidad por Infrarrojos ("GP2Y0A21YK","SEN-00242","SEN-0016 "SPRK-GP2Y0A2").

[25] Network of Excellence Robotics & Mechatronics HomeLab Community. Infrared distance sensor [Robotic & Microcontroller Educational Knowledgepage - Network of Excellence].

[26] NodeMCU. WiFi - NodeMCU Documentation.

[27] Wikipedia. Lipolybattery - Lithium polymer battery - Wikipedia.

[28] Wikipedia. Lithium-ion battery - Wikipedia.

[29] Instructables. Arduino Battery Voltage Indicator : 5 Steps (with Pictures) - Instructables.

[30] Wikipedia. STL (file format) - Wikipedia.

[31] Thingiverse. Customizable Wheel for 28BYJ-48 stepper motor by MakersBox - Thingiverse.

[32] Thingiverse. 16mm Marble Robot Ballcaster by hardmoduino - Thingiverse.

[33] LEGO®. EV3 LEGO® MindStorms® Spanish Web Page.

[34] Robotix. EV3 Servomotor Grande.

[35] Robotix. EV3 Servomotor Mediano.

[36] Robotix. EV3 Sensor de Color.

[37] Robotix. EV3 Sensor de Ultrasonidos.

[38] Robotix. EV3 Sensor de Tacto.

[39] Robotix. EV3 Sensor Giroscopio.

[40] Robotix. EV3 Sensor de Infrarrojos.

[41] Makeblock. mBot - Bluetooth - Robot Educativo 90054- Makeblock.

[42] Spyros G Tzafestas. *Introduction to Mobile Robot Control*. Elsevier Inc, 2014.

[43] Microchip. ATmega48A/PA/88A/PA/168A/PA/328/P.

[44] Arduino Official Store. Arduino Mega 2560 Rev3 || Arduino Official Store.

[45] Microchip. ATmega640/V-1280/V-1281/V-2560/V-2561/V.

[46] WeMos.cc. D1 [WEMOS Electronics].

[47] Espressif. ESP8266EX Datasheet.

[48] Arduino.cc. Datasheet for H-Bridge Motor Driver L293D.

[49] Onsemi. Datasheet for Shift Register 74HC595.

[50] Wikipedia. Odometry.

[51] H. R. Everett J. Borenstein and L. Feng. Where am I? Sensors and Methods for Mobile Robot Positioning.

[52] Wikipedia. Speed of sound.

[53] ElecFreaks. HC-SR04 - HCSR04.pdf.

[54] K-Electrónica. Arduino UNO R3 + CABLE USB.

[55] AliExpress. WAVGAT For Arduino UNO R3 Development board High quality UNO R3 CH340G For ATMEGA328P 16Mhz ATMEGA328P AU|Integrated Circuits| | - AliExpress.

[56] K-Electrónica. Arduino MEGA 2560.

[57] AliExpress. MEGA2560 MEGA 2560 R3 ATmega2560 16AU CH340G AVR USB board Development board MEGA2560 for arduino|Integrated Circuits| | - AliExpress.

[58] K-Electrónica. Arduino WeMos D1 wifi.

[59] AliExpress. WeMos D1 R2 V2.1.0 WiFi uno en ESP8266 para arduino nodemcu Compatible|wemos d1|arduino uno wifi|uno arduino - AliExpress.

[60] K-Electrónica. Motor mas rueda para rover arduino.

[61] AliExpress. TT Motor Smart Car Robot Gear Motor for Arduino Free Shipping Wholesale for Arduino Motor Smart Robot Car|motor smart robot car|motor tt|motor car smart - AliExpress.

[62] K-Electrónica. Motor paso a paso 28BYJ-48 5V más driver compatible arduino.

[63] AliExpress. WAVGAT 5V Motor paso a paso 28BYJ 48 + ULN2003 módulo de prueba de controlador para Arduino, Micro Mini Motor eléctrico paso a paso para PIC 51 AVR|Circuitos integrados| | - AliExpress.

[64] K-Electrónica. Servo de rotación continua BQ Arduino Robot robotica educativa.

[65] AliExpress. SG5010 3KG 5KG High Torque Digital Servo Motor RC Helicopter Airplane Boat for Arduino UNO R3 SG90 MG90S|Integrated Circuits| | - AliExpress.

[66] K-Electrónica. Sensor de distancia Ultrasonido compatible Arduino robótica educativa.

[67] AliExpress. WAVGAT HC SR04 HCSR04 a Detector de onda ultrasónico mundial Módulo de alcance HC SR04 Sensor de distancia HC SR04 HCSR04|Circuitos integrados| | - AliExpress.

[68] RS Componentes. GP2Y0A21YK0F | Sensor reflector Sharp GP2Y0A21YK0F | RS Components.

[69] AliExpress. GP2Y0A21YK0F 10 80cm GP2Y0A02YK0F 20 150CM GP2Y0A41SK0F 0A41SK 4 30cm Sensor de proximidad por infrarrojos IR analógico Sensor de distancia|distance measuring sensor|ir distance|infrared ir - AliExpress.

[70] denmorru in GitHub. GitHub - denmorru/DMotor: Library for arduino ESP8266 extension. Use motor shield on Wemos D1 R2 board.

[71] How To Mechatronics. Ultrasonic Sensor HC-SR04 and Arduino Tutorial.

[72] K-Electrónica. Driver para motor L293D compatible Arduino robótica educativa.

[73] AliExpress. Freeshipping L293D motor control shield motor drive expansion board FOR Arduino motor shield|Integrated Circuits| | - AliExpress.

[74] K-Electrónica. Batería Lipo 7.4 V 650 mAh 25c.

[75] AliExpress. 7.4V 850mAH 703048 Lipo Battery For Udi U829A U829X MJXRC X600 HQ 907remote control Li po battery 7.4 V 850 mAH 20C JST SM plug|Parts & Accessories| | - AliExpress.

[76] K-Electrónica. Conector alimentación regleta hembra.

[77] AliExpress. 5pcs Female +5 pcs Male DC connector 2.1*5.5mm Power Jack Adapter Plug Cable Connector for 3528/5050/5730 led strip light|Connectors| | - AliExpress.

[78] K-Electrónica. Conector alimentación macho 2.1 x 5.5 mm para arduino.

[79] K-Electrónica. Interruptor AC 250 V 3A 2 Pin ON / OFF.

[80] AliExpress. 5pcs/lot KCD1 15*10mm 2PIN Boat Rocker Switch SPST Snap in on Off Micro Switch Position 3A/250V Mini Momentary Push Button|Switches| | - AliExpress.

[81] K-Electrónica. Mini protoboar breadboard sin soldaduras.

[82] AliExpress. Free Shipping wholesale 1pcs SYB 170 Mini Solderless Prototype Experiment Test Breadboard 170 Tie points 35*47*8.5mm|Integrated Circuits| | - AliExpress.

[83] K-Electrónica. Cargador de la batería para JJRC H8C.

[84] AliExpress. 21V 1A 8.4V 2A 12.6V 1.5A 16.8V 2A Lithium Battery Charger DC 5.5mm EU/US Plug 110 220V Li ion Battery Wall Charger|Chargers| | - AliExpress.

[85] K-Electrónica. Filamento PLA Berenjena 175 BQ impresora 3D impresora3D k-electronica.

[86] AliExpress. FILAMENTO DE PLA 3D (1 Kg) TINTA PARA IMPRESORA CALIDAD PREMIUM CREACION FIGURAS | eBay.

[87] K-Electrónica. 40 cables 20cm 2,54 mm macho a hembra para Arduino.

[88] AliExpress. WAVGAT Dupont line 120pcs 20cm male to male + male to female and female to female jumper wire Dupont cable for Arduino|jumper wire|line dupont|jumper arduino - AliExpress.

[89] TV Nalber. TV Nalber - Tienda on-line. Juego 40 cables 20cm macho macho Arduino.

[90] TV Nalber. TV Nalber - Tienda on-line. Cable USB 2.0 A-USB micro B 1mts.

[91] AliExpress. 0.2m/1m/2m/3m Micro USB Cable Fast Charging Sync Data Mobile Phone Android USB Charger Cables for Samsung Xiaomi redmi Micro 2.0|Mobile Phone Cables| | - AliExpress.

[92] TV Nalber. TV Nalber - Tienda on-line. Servo motor SG-90 compatible Arduino.

[93] AliExpress. WAVGAT official Smart Electronics Rc Mini Micro Classic servos 9g SG90 MG90S For RC Planes Fixed wing Aircraft model telecontro|Integrated Circuits| | - AliExpress.

[94] K-Electrónica. Resistencia de película metálica 2.2 K Ohm.

[95] AliExpress. 1 Pack 300Pcs 10 1M Ohm 1/4w Resistance 1% Metal Film Resistor Resistance Assortment Kit Set 30 Kinds Each 10pcs Free Shipping|metal film resistor|film resistors|resistance assortment kit - AliExpress.

[96] K-Electrónica. Potenciometro 3362p-103, 10 K Ohm alta precisión Resistencia variable.

[97] K-Electrónica. Bobina Cable 280m AWG30.

[98] AliExpress. 2M 12/16/18/20/26/28/30AWG 1M black+1M red Silicone Wire SR Wire Flexible Stranded Copper Two Wires Electrical Cables|Wires & Cables| | - AliExpress.

[99] K-Electrónica. Regleta o clema de conexión de 12 polos hasta 6 mm$^2$ color negro · LEROY MERLIN.

[100] AliExpress. 2 Pcs 10A Dual Row Connector 12 Position Wire Barrier Terminal Strip Block Wiring Accessories Connectors Terminals|Terminal Blocks| | - AliExpress.

[101] K-Electrónica. LED super brillante núcleo grande Rojo.

[102] AliExpress. 100PCS WAVGAT 5mm Round Red Super bright emitting diode LED Light 5000MCD|diode led light|5mm round|emitting diode - AliExpress.

[103] K-Electrónica. Diodo Rectificador 3A 400 V.

[104] AliExpress. 100 Uds 1A 1000V diodo de 1N4007 IN4007 hacer 41 IN4001 50V IN4002 100V IN4003 200V IN4004 400V de plástico rectificador de silicio IN4148|Diodos| | - AliExpress.

[105] Makerguides. How to use an IR Distance Sensor with Arduino (SHARP GP2Y0A21YK0F).

[106] Luisana Lara. LLUBot.