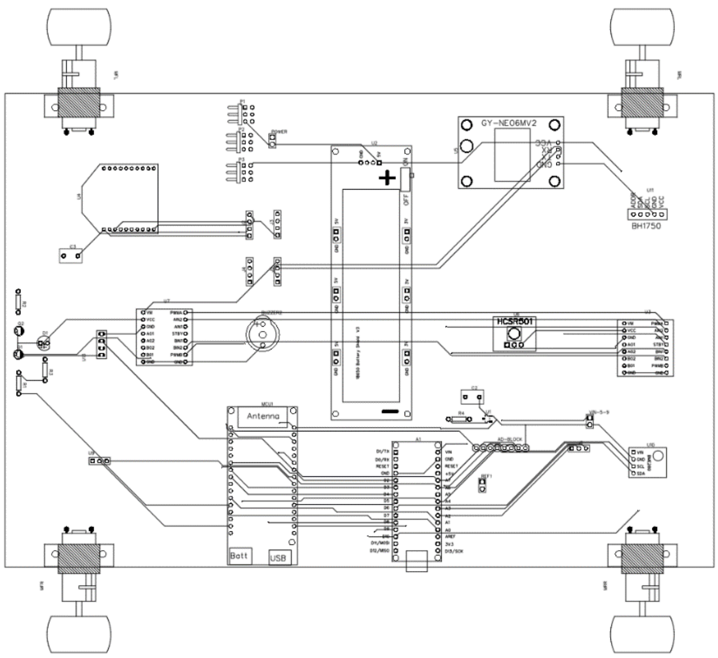
Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Escuela de Ingeniería Informática
Universidad de Oviedo

Design of a robot for teaching microcontrollers

BACHELOR'S DEGREE THESIS – SOFTWARE ENGINEERING



Author: Enrique José Rodríguez Martín

Supervisor: Juan Carlos Álvarez Antón

July 2021

School of Computer Science Engineering

University of Oviedo

Design of a robot for teaching microcontrollers

Version: 1.0

Author: Enrique José Rodríguez Martín

Supervisor: Juan Carlos Álvarez Antón

Date: February 14th, 2021

Entity: School of Computer Science Engineering of the University of Oviedo

Abstract

This project describes the process of creating a piece of hardware, a robot, with various capabilities such as remote control, line following, and collision detection. The final product will be the robot and a compilation of practical laboratory sessions aimed to teach modern microcontrollers.

Acknowledgements to *my parents, Juan Carlos Álvarez Antón, Jorge Álvarez Fidalgo and my friends and colleagues.*

Index

1. Report ... 7
1.1 Purpose ... 7
1.2 Scope ... 7
1.3 Background ... 8
1.4 Glossary & abbreviations ... 8
1.5 Design requirements ... 9
1.6 Hypothesis & restrictions ... 10
1.7 Proposed solution overview ... 11
1.8 Risk analysis ... 12
1.9 Scheduling overview ... 14
1.10 Budget overview ... 14
1.11 Priority order of project's documents ... 15
2. Annexes ... 16
2.1 Initial documentation ... 16
2.2 Analysis & design of the robot ... 18
2.2.1 Component description ... 19
2.2.2 Electrical schematic ... 29
2.2.3 PCB schematic ... 33
2.3 Prototype construction ... 36
2.3.1 Tools and materials ... 36
2.3.2 Module preparation ... 42
2.3.3 PCB assembly ... 63
2.4 Design of the robot practices ... 74
2.4.1 Practice 1: Introduction ... 74
2.4.2 Practice 2: Collision detection ... 79
2.4.3 Practice 3: Remote control ... 81
2.4.4 Practice 4: GPS ... 85

2.4.5 Practice 5: Light control 87
2.5 Effort measurement and estimation 89
3. Requirements 90
3.1 Functional requirements 90
3.2 Non-functional requirements 91
3.2.1 Documentative requirements 91
3.2.2 Technological requirements 91
4. Budget 93
4.1 Cost budget 93
4.1.1 Client budget calculation 96
4.2 Client budget 96
5. Conclusions 97
6. Bibliography 98

Figure Index

Figure 1. PCB Design of the final product. ... 11
Figure 2. Gantt diagram of the project. ... 14
Figure 3. Electric schema of previous robot. ... 16
Figure 4. PCB schema of previous robot. ... 17
Figure 5. Arduino Nano pinout. ... 19
Figure 6. Battery 18650 Li-Ion. ... 20
Figure 7. Battery shield with USB port. ... 20
Figure 8. BH1750 sensor. ... 21
Figure 9. BME280 sensor. ... 21
Figure 10. Buzzer. ... 22
Figure 11. Direct Current motor. ... 22
Figure 12. ESP32 LOLIN without pins. ... 23
Figure 13. CMD prompt during esptool.exe use. ... 24
Figure 14. HC-SR04 sensor. ... 25
Figure 15. HC-SR501 sensor. ... 25
Figure 16. Level converter module. ... 26
Figure 17. GPS sensor. ... 26
Figure 18. Photodiodes. ... 27
Figure 19. Robotic arm with SG90 motors. ... 27
Figure 20. Motor driver. ... 28
Figure 21. XBee module. ... 28
Figure 22. EasyEDA basic view. ... 29
Figure 23. EasyEDA electrical schema designer view. ... 30
Figure 24. EasyEDA component library browser. ... 31
Figure 25. EasyEDA component designer view. ... 31
Figure 26. Robot’s electrical schema. ... 32
Figure 27. EasyEDA PCB designer view. ... 33
Figure 28. Robot's final schematic render. ... 34
Figure 29. EasyEDA ordering dialog. ... 35
Figure 30. JLCPCB order status view. ... 35
Figure 31. Soldering iron tool. ... 36
Figure 32. Tin reel. ... 37
Figure 33. Soldering stand with sponge and magnifying glass. ... 38
Figure 34. Desoldering suction pump. ... 39
Figure 35. Tweezers. ... 39

Figure 36. Screwdriver. 39
Figure 37. Electrician scissors. 40
Figure 38. Needle nose pliers. 40
Figure 39. Multimeter with probes. 41
Figure 40. Wheel driver with unsoldered pins. 42
Figure 41. Wheel driver with soldered pins. 42
Figure 42. Barometric pressure sensor with unsoldered pins. 43
Figure 43. Barometric pressure sensor with soldered pins. 43
Figure 44. ESP32 LOLIN microcontroller with unsoldered pins. 44
Figure 45. ESP32 LOLIN microcontroller with soldered pins (back side). 45
Figure 46. ESP32 LOLIN microcontroller with soldered pins (front side). 45
Figure 47. Light sensor with unsoldered pins. 46
Figure 48. Light sensor with soldered pins. 46
Figure 49. Level converter with unsoldered pins. 47
Figure 50. Level converter with soldered pins. 47
Figure 51. GPS module and external antenna. 47
Figure 52. GPS module with connected antenna. 48
Figure 53. GPS module with connected and glued antenna. 48
Figure 54. Original battery shield (top view). 49
Figure 55. Original battery shield (bottom view). 49
Figure 56. USB header pinout diagram. 50
Figure 57. Battery shield without USB port (top view). 50
Figure 58. Battery shield without USB port (bottom view). 50
Figure 59. Battery shield with connection wires (soldered to +5V and GND pins). 51
Figure 60. Battery shield alternative with pins. 51
Figure 61. Robotic arm parts kit. 52
Figure 62. Base with plastic axis, original piece for reference. 53
Figure 63. Pivotal platform and SG90 motor. 54
Figure 64. Rasped motor inside the first platform piece. 55
Figure 65. Fixed pivotal platform with motor inside. 55
Figure 66. Base attached to the platform using the motor and axis. 56
Figure 67. Screw used to fix the motor in place. 56
Figure 68. Second platform with motor. 57
Figure 69. Second motor fixed by screws to platform. 57
Figure 70. Second platform before fixing to first platform. 58
Figure 71. Both platforms attached mechanically. 59
Figure 72. Second platform before adapting plastic axis. 60
Figure 73. Plastic axis adapted. 61

Figure 74. Robotic arm completed. ... 62
Figure 75. PCB top view. ... 63
Figure 76. PCB bottom view. ... 64
Figure 77. Photodiodes placed. ... 65
Figure 78. Soldered photodiodes. ... 65
Figure 79. Completed soldering for the photodiodes circuit. ... 66
Figure 80. Female pin headers soldered in place. ... 66
Figure 81. Soldered buzzer and adapted level converters. ... 67
Figure 82. Soldered headers for microcontrollers. ... 68
Figure 83. Soldered modules headers. ... 68
Figure 84. Fixed DC motor. ... 69
Figure 85. Traction system overview. ... 69
Figure 86. Wheels attached to motors. ... 70
Figure 87. DC motor connection process. ... 71
Figure 88. Fixed battery shield with battery. ... 72
Figure 89. Motors and drivers test. ... 72
Figure 90. Assembled robot. ... 73
Figure 91. H-Bridge circuit. ... 74
Figure 92. Wheel driver internal circuit. ... 75
Figure 93. Track detection electrical circuit. ... 77
Figure 94. HC-SR04, distance sensor. ... 79
Figure 95. Timing diagram of working distance sensor. ... 79
Figure 96. Remote control and robot communication diagram. ... 81
Figure 97. Remote control. ... 82
Figure 98. GPS module main parts. ... 85
Figure 99. Master/slave I2C communication. ... 87

Table Index

Table 1. Quality of components. 12
Table 2. Wrong manipulation. 12
Table 3. Shipment problems. 13
Table 4. Soldering problems. 13
Table 5. PCB adjustments. 13
Table 6. Scheduling summary. 14
Table 7. Budget summary. 14
Table 8. Comparison between both robots. 18
Table 9. Control signals of wheel driver. 75
Table 10. Microcontrollers and driver connections. 76
Table 11. Microcontrollers and track detection signals connections. 77
Table 12. Microcontrollers and buzzer connections. 78
Table 13. Microcontrollers and distance sensor connections. 80
Table 14. Remote control connections. 82
Table 15. Microcontrollers and GPS module connections. 87
Table 16. Work Breakdown Structure of the project. 89
Table 17. Worker roles and costs. 93
Table 18. Project direct costs. 94
Table 19. Amortizations. 94
Table 20. Electrical consumption. 95
Table 21. Indirect costs. 95
Table 22. Costs summary. 95
Table 23. Client budget calculation. 96
Table 24. Client budget. 96

1. Report

1.1 Purpose

The present document describes the development of an educational robot's hardware and to some degree its control software. The final product will be a modular multi-purpose teaching tool that could be put in use in various courses across the University of Oviedo related to microcontrollers.

A complete student guide will be provided with the robot. This resource would allow a person with little or no experience to comprehend the basics and fundamentals of microcontrollers, modules, and electronic circuits. The main goal of the guide is providing a teaching and learning tool with valuable activities for the better understanding of the theoretical concepts provided.

The robot will use two different kinds of microcontrollers that can be programmed in the same IDE; in this case, Arduino IDE. It will provide support for I2C modules, remote control using ZIGBEE components and WIFI connection using the ESP32 microcontroller. It will also have obstacle detection and line following capabilities.

To give a common structure to this document, we will follow the UNE 157801 norm. We should take into consideration that this norm is aimed to Information Systems, so we will adhere also to the UNE 157001 norm since it is a more flexible and generic approach. This document may lack some of the characteristics described in the norm, as they are not suitable for this kind of product.

1.2 Scope

The goal of this project is to develop a piece of hardware -a robot- that is able to use two kinds of microcontrollers: Arduino Nano and ESP32, substituting the current teaching robot and refining or upgrading its hardware and functionality.

The robot will be powered by an 18650 Battery and possess movement capabilities, using DC motors attached to wheels and operated in pairs by specific motor drivers.

The project puts emphasis on the modularity of the robot, which makes use of I2C Bus for several modules. This technology allows to integrate different components at the same time without consuming specific pins in the microcontrollers. It also will allow to change transparently to a different hardware component that uses the same I2C communication, so that only control software modifications are required.

Any main component described in the design phase of this project will not be soldered to the PCB. This will make the robot easy to adapt to the necessities required to achieve the desired capabilities, as well as providing a simple way to identify the components that may be damaged during the use of the robot or by some external use.

The lab practices will provide theoretical contents with the basics and fundamentals of each part of the robot and code examples of how to handle them in the Arduino IDE. It will also contain some small exercises that students should complete during the lab sessions, allowing them to gradually understand the principles of microcontrollers and module programming.

1.3 Background

Electronic Technology of Computers course is currently using a robot during the laboratory sessions. Such robot has a much simpler design, possessing only two driving wheels and not as many modules to work with.

This project aims to extend the capabilities of the previous version, providing a much complex robot hardware-wise while increasing its functionality.

The new robot design will provide an upgraded platform for the laboratory sessions in future courses and, if needed, an extensible base design for updates and new requirements.

1.4 Glossary & abbreviations

- **Bus:** Shared communication channel.
- **CMD:** Command Prompt. Default command-line interpreter on Windows.
- **DC:** Direct Current, one directional flow of current.
- **GND:** Ground, common return path for electric current.
- **Lux:** Derived unit of the International System of Units, measuring luminous flux per unit of area.
- **Lx:** abv. Lux.
- **I2C:** Inter-Integrated Circuit, serial communication bus.
- **ISO:** International Standard Organization. Organization in charge in standardization processes.
- **Microcontroller:** Small computer designed for embedded applications.
- **µC:** abv. Microcontroller.
- **PCB:** Printed Circuit Board, connects electrically and physically electronic components.
- **Pin:** Electrical connection.
- **UNE:** Spanish Normalization Association. Organization in charge in standardization processes.
- **V:** abv. Voltage. Electric potential difference.

1.5 Design requirements

In this section, we will briefly list the initial and fundamental requirements that this project must fulfil. The complete list will be available in the Requirements chapter, where we will describe the goals of this project with a more in-depth description of those.

Hereunder, we may observe the mentioned list of basic requirements of the project without traceability:

- The robot must be capable of:
    - Following line paths.
    - Being remotely controlled.
    - Obstacle detection.
    - Communicating via WIFI.
    - Using the sensors and modules.
- The robot will be powered by an 18650 battery.
- The chassis of the robot will be its PCB.
- The PCB must be prepared for using Arduino Nano and ESP32 microcontrollers.
- The PCB must connect the microcontrollers with the following modules:
    - 18650 Battery Shield.
    - HC-SR501, infrared movement module.
    - HC-SR04, distance sensor.
    - BME280, barometric pressure sensor.
    - BH1750, light intensity sensor.
    - NEO6MV2, GPS module.
    - Buzzer.
    - XBEE, serial communication receiver.
    - Robotic arm controlled by SG90 DC motor.
    - Power train containing:
        - 4 DC motors.
        - 4 wheels attached to the DC motors.
        - 2 wheels drivers.
    - 2 level converters to provide the correct working voltage of various signals of the microcontrollers or the modules.
    - A photodiode circuit used in the line-following functionality.
    - Open connections for other desired modules.
- The student guide must provide the fundamentals and basics of the modules listed.
- The student guide must follow the style of the one currently in use by the course.
- The student guide must provide practical exercises for each module.

1.6 Hypothesis & restrictions

There exist some hypothesis and restrictions which, despite being events or circumstances outside of the development of this project, may affect its success. Such circumstances will be considered as real events or facts required for the execution of this project. If these events do not materialize or become a fact, the budget and the scheduling will be affected. The following events are considered:

- **Quality of the components**. Some of the required modules are manufactured in China. This may not be a major concern, but it could be troublesome if some of the modules are in a bad condition after shipment or have some technological disadvantage from the original product that may cause integration or operative problems. These Chinese clones aim to cut costs in the budget compared to original parts, but could cause more problems in the long run as they tend to be poorly made.
- **Wrong manipulation of the components**. The components of the robot are sensible pieces of hardware; a wrong manipulation or human error to interpret the blueprints may derive to a short-circuit and cause damage to the modules, the microcontrollers, or the PCB. Careful storage of the components and verification of all connections before providing power to the modules must be a priority. Damaging the component may cause delay in the project scheduling and an increment in costs.
- **Shipment problems.** The components are not bought in a physical store; shipments are required to fulfil the hardware need for this project. Most Chinese shipments are slow and do not provide enough tracking information. Nevertheless, using alternative, more reliable couriers may increase the budget exorbitantly.
- **Minimum purchase.** Many online platforms charge extra costs if a minimum purchase is not satisfied; we should take into consideration this concern and act properly. We ought to minimize the order of single parts or take into consideration the ordering of extra equipment if that decision may reduce the cost of the order.

1.7 Proposed solution overview

As described in the previous sections, the final product will be a modular multipurpose teaching tool. The delivered robot is an evolution of the current in use at Computer Electronics.

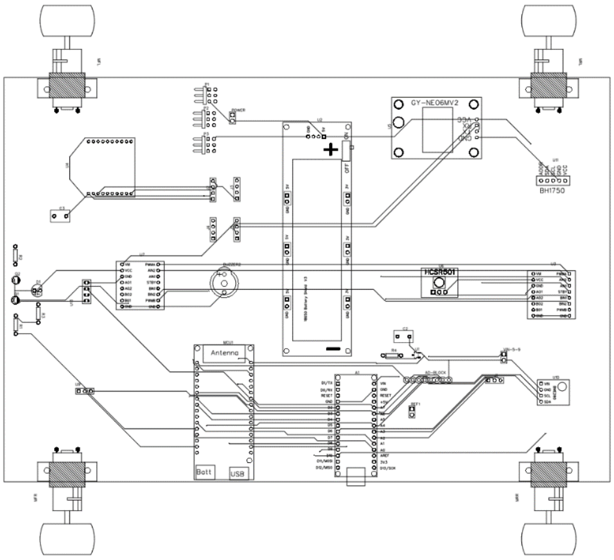


Figure 1. PCB Design of the final product.

The power unit, defined by the 18650 battery and its shield, will be placed in the middle of the robot to make its centre of gravity as balanced as possible. The microcontrollers must be placed on the sides of the robot for better access to its USB ports and loading the control programs.

The robotic arm is positioned in a forward position; in the prototype presented, it only has its basic movement available (lateral rotation), but it can be upgraded connecting a second SG90 motor. The second motor provides the arm vertical movement alongside the current rotational one proposed in the basic configuration. Also, a module can be placed in the platform provided in the upper part of the arm so that it may have extra functionality; for example, a camera that can be controlled by the Arduino or ESP32.

The movement of the robot is ensured by the robot drive trains; two drivers simplify the control of the four wheels by the microcontroller.

The PCB has space for various sensors such as: barometric pressure sensor, GPS module, light sensor, or distance sensor. Some components use I2C communication, allowing the components to be changed or removed transparently without any modification in the PCB.

The student guide follows the same structure and style as its predecessor, including the new features of the upgraded system with enhanced descriptions, exercises, and theoretical concepts.

1.8 Risk analysis

In this section, we are going to list the possible risks that may affect the success of this project. We should identify feasible situations and how to respond in case they materialize during the execution of this project. This analysis will be taken as a compilation of strategies, although the responses to the risks may change depending on the situation.

Hereunder, a list of the risks analysed:

Probability	**High**
Title	Quality of the components
Description	Many components are Chinese clones. These clones should replicate the functionality of the official components; they usually use some workarounds that are prone to fail, though.
Strategy	Internalise the risk
Response	Search for solutions on the internet. Many people may have encountered the same problem, so it is possible that some fixes are available on personal blogs or similar resources. Another possibility is searching and buying high quality clones from reliable sources, but this strategy may increase the total cost of the desired resource.

Table 1. Quality of components.

Probability	**High**
Title	Wrong manipulation
Description	A wrong manipulation or connection of the components may derive to a short-circuit and cause an irreversible damage to key parts of the robot or the controller. This occurrence will generate delays on the project' schedule and an increase of costs.
Strategy	Mitigate the risk
Response	Take special care of the components used in the robot. Make sure all connections are in accordance with the blueprints before any electrical input is provided. Revise the complete circuit several times. Connect the components one by one to determine if the connections are as expected.

Table 2. Wrong manipulation.

Probability	**High**
Title	Shipment problems
Description	Many necessary modules of this project come from exotic places; this leads to some sorts of shipment problem which may occur. In this scenario, patience is key, as many of these factors are out of our control.
Strategy	Internalize the risk
Response	If the tracking system of the vendor does not provide any hope of a successful delivery within a reasonable amount of time, look for more reliable sources. Another option is to pay an extra charge for a far better shipping option, but this will increase the budget reserved for the desired part.

Table 3. Shipment problems.

Probability	**Medium**
Title	Soldering problems
Description	Soldering requires a great level of concentration and experience; pins of most modules are spaced 2.54mm between them.
Strategy	Mitigate the risk
Response	Try to reserve a calm place to proceed to solder; patience is a key part in the process. Many components require more than 10 pins to be soldered and in any of them you could have some sort of problem. You must be extremely careful with small variations in angle of the pin strip from the component, as it could cause an adjustment problem when the component is put in place.

Table 4. Soldering problems.

Probability	**Medium**
Title	PCB adjustments
Description	Some of the components placed in the PCB schema required for the manufacture process are from a user library. This could cause that the measurements are estimated or come from unreliable sources; they even could have some reversed features such as ground (GND) and input voltage (VCC).
Strategy	Mitigate the risk
Response	Check that the PCB footprint of the selected component is the same as the desired component counterpart. This minimizes the fitting problem of the module, which could be solved with relative ease.

Table 5. PCB adjustments.

1.9 Scheduling overview

In this section, we are going to show the proposed schedule for this project. To have a more in-depth analysis, read *Effort measurement and estimation*. The proposed scheduling was considered for the *Budget* calculations.

Table 6. Scheduling summary. shows the main parts of this project as well as their estimated starting and ending dates:

Index	Name of task	Time spent	Start date	End date
1	Design of the robot	19 hours	15/01/21	19/01/21
2	Hardware acquisition	1528 hours	19/01/21	29/03/21
3	Robot Assembly	14 hours	30/03/21	31/03/21
4	Lab session elaboration	15 hours	03/04/21	05/04/21

Table 6. Scheduling summary.

Figure 2. Gantt diagram of the project. illustrates the time scheduling:

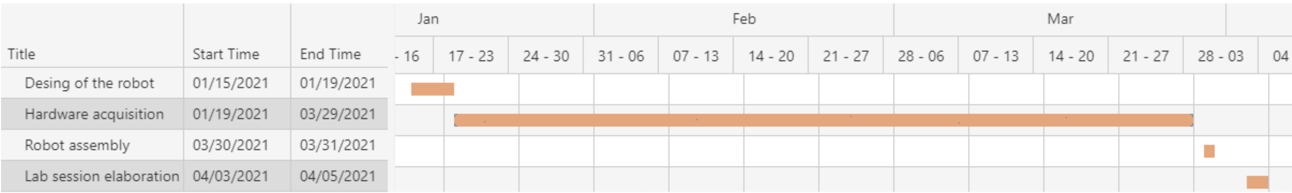


Figure 2. Gantt diagram of the project.

1.10 Budget overview

In this section, we will provide a summary of the calculated budget of this project. For a more detailed overview, see *Budget* section.

Code	Item	Price	Total
1	Design of a robot for teaching microcontrollers		1949.20€
1.1	Design of the robot	473.22€	
1.2	Hardware acquisition	754.95€	
1.3	Robot assembly	348.08€	
1.4	Lab session elaboration	372.95€	
TOTAL COST			**1949.20€**

Table 7. Budget summary.

1.11 Priority order of project's documents

In this section, we are going to stablish a priority order of the project's documents. In this case we will use the UNE 157801 norm sequence. Such norm establishes the following order: *Report*, *Annexes*, *Budget*, *Requirements* and *Conclusions*.

2. Annexes

2.1 Initial documentation

This project is the evolution of the current robot used at Computer Electronics course. In this section, we are going to describe the features of the previous iteration, and thus obtaining a whole picture of the improvements that will be described in *Analysis & design of the robot* annexe.

The document provided for the laboratory sessions only focuses in two functionalities of the robot: line following and remote control. Although the information provided in the document is enough for understanding the hardware, it may seem that the robot acts as a complementary task more than a key part of the learning experience. One of the goals of our new design is to emphasise the power of modern microcontrollers; as a result, the practices will be more aimed to the new hardware.

First, we are going to illustrate the electrical schematic of the robot and describe its main features.

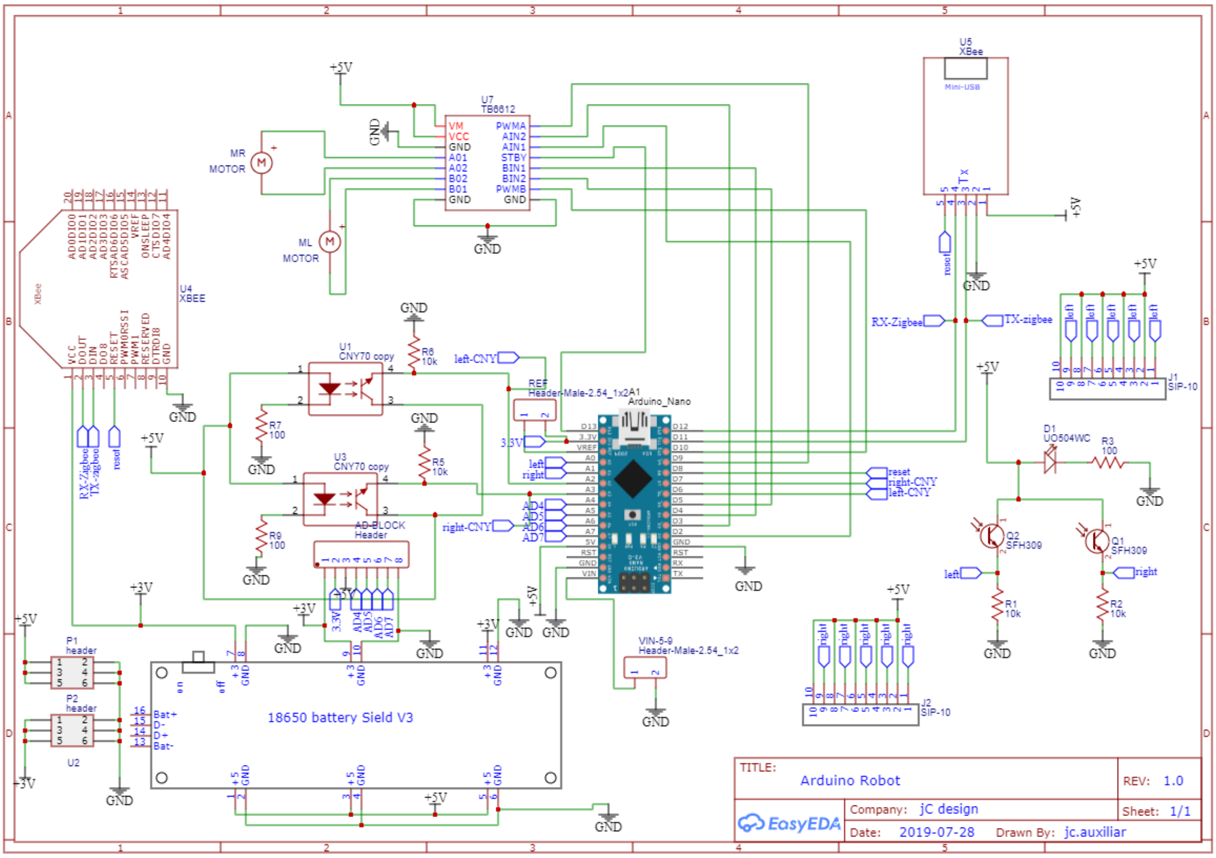


Figure 3. Electric schema of previous robot.

The robot has an 18650-battery shield. The PCB is powered directly by the +5V and +3V of the shield with no control mechanism; in this configuration, the components will drain the battery and the only way to stop it is by removing the battery.

The only microcontroller available is the Arduino Nano, which coordinates: a motor driver connected to two DC motors (traction system), XBEE serial communication module (allows remote control) and the photodiodes needed for the line following capabilities. The Arduino Nano is also connected to several headers which allow the connection of sensors that are not listed in the schematic.

The PCB also has components to adjust the digital signals of the XBEE that run at +3V-0V range to those of the Arduino Nano which run at +5V-0V range.

As we have described before, the traction system of the robot is composed by two wheels, which can be moved independently using a driver and the microcontroller.

The electrical schematic only provides the connections and how the modules are powered. Now, we are going to analyse the PCB diagram of the original robot. Such diagram, was sent to production and depicts how the tracks of the PCB are positioned and where the components are placed.

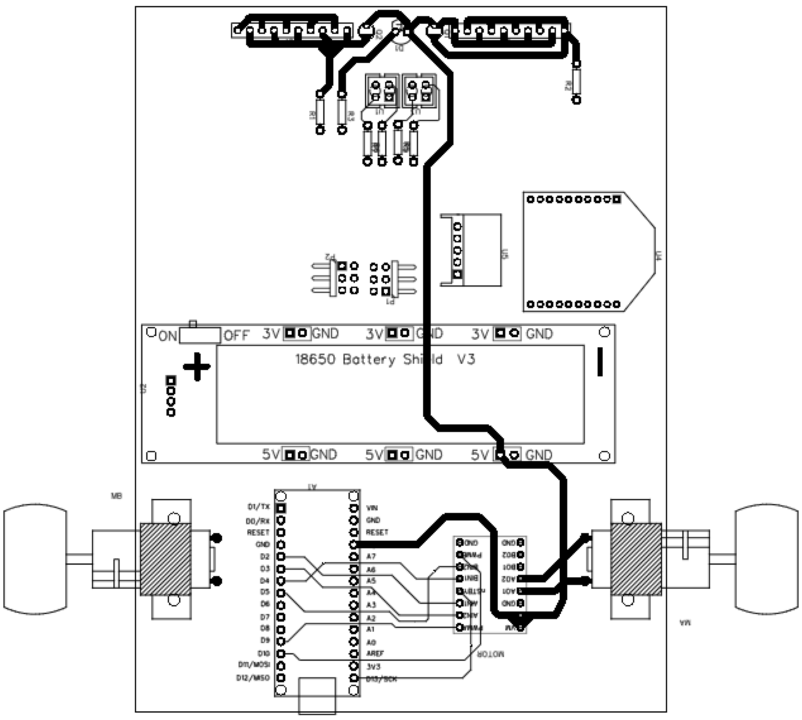


Figure 4. PCB schema of previous robot.

As we can see, most of the weight of the robot is placed in the rear. This design decision aims to compensate the fact that the robot has only rear traction. In front, alongside the photodiodes, a caster wheel provides the front contact point. A much better configuration for this robot would be a rear mounted battery; however, the PCB is not wide enough to allocate the battery shield. Such design results in a forward tilted "chassis" that potentially could allow for sharper turns.

2.2 Analysis & design of the robot

In this section, we are going to explain the design process of the robot and what decisions were taken. First of all, we will take the characteristics of the previous model and compare it with the new selected features. For a more detailed comparison, read *Initial documentation* and *Requirements* sections.

Feature	Component	Old version	New Version
Line following	Photodiodes	Yes	Yes
Remote control	Xbee	Yes	Yes
Drive wheels	DC motor	Yes	Yes
Arduino microcontroller	Arduino Nano	Yes	Yes
ESP32 microcontroller	ESP32	No	Yes
Robotic Arm	SG90	No	Yes
GPS	NEO6MV2	No	Yes
Light sensor	BH1750	No	Yes
Barometric sensor	BME280	No	Yes
WIFI	ESP32	No	Yes
Buzzer	Buzzer	No	Yes
Distance sensor	HC-SR04	No	Yes
Movement sensor	HC-SR501	No	Yes

Table 8. Comparison between both robots.

2.2.1 Component description

In this section, we are going to list the components selected in the *Requirements* section as well as describing what their intended purpose is, how do they work, and how do they connect with other elements.

2.2.1.1 *Arduino Nano*

Microcontroller. It is a smaller version of the Arduino Uno; breadboard friendly but as capable as the Uno. In our robot, the Nano controls the sensors and functionalities. Moreover, it is also the component prepared to accept the control program, which is loaded from a computer by a USB cable. The pinout is shown in Figure 5. Arduino Nano pinout. As a summary, the Arduino has various pins for specific functions. For example, the analogical pin 4 (A4) and the A5 are used for handling the I2C components (SDA and SCL).

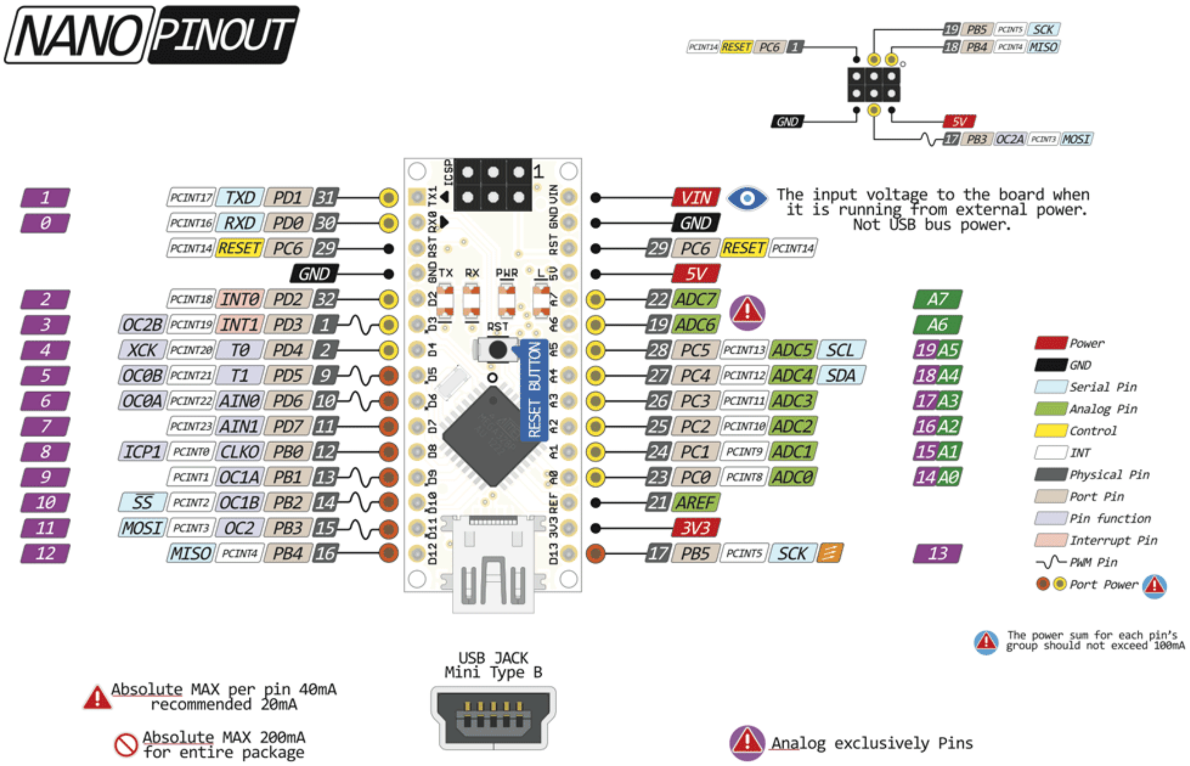


Figure 5. Arduino Nano pinout.

Keep in mind that both Arduino original devices and good quality clones use a proprietary chip to connect with your computer via USB. This is possible thanks to the FT32 chip, which has no problem connecting with a Windows system. Chinese clones have a workaround chip called CH340; their free drivers may or may not work on your computer. This clone chip is aimed to reduce the costs of the hardware as the manufacturers do not need to pay any license to the proprietary chip company.

2.2.1.2 Battery 18650

It is a lithium-ion battery of 2200mA. This type of battery provides a good portability for the amount of energy they are able to store. They are prone to overheat if mishandled and could start a fire or cause a small explosion.

Figure 6. Battery 18650 Li-Ion.

2.2.1.3 Battery shield

Component used for holding an 18650 battery. The USB port will be removed to control the output of the battery. It has a micro-USB charging port and various pin holes to obtain +5V and +3V.

Figure 7. Battery shield with USB port.

2.2.1.4 BH1750

Light sensor. It has 5 pins: VCC power supply, GND ground, SCL serial clock line used for I2C communication, SDA serial data address used for transferring communication using I2C, ADDR device address pin. In our case, we are not going to use the ADDR pin, although it may be so in a possible future design improvement.

Figure 8. BH1750 sensor.

2.2.1.5 BME280

Barometric pressure sensor. It is also a I2C module and can measure humidity, pressure, and temperature. It has 4 pins: VIN power supply, GND ground, SCL serial clock line used for I2C communication and SDA serial data address used for transferring communication using I2C. Its IIR filter can be used to reduce the noise in the data collected (e.g., a window or door is opened causing a fluctuation on the Ambiental conditions).

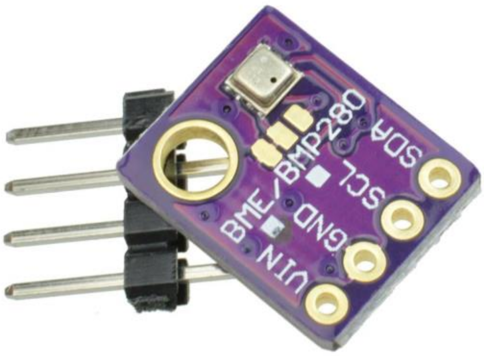


Figure 9. BME280 sensor.

2.2.1.6 Buzzer

Used for producing sound. It has a positive pin that can be powered by +6V DC, as well as a negative pin connected to the ground; the negative pin is identified thanks to a shorter length compared to the positive one.

Figure 10. Buzzer.

2.2.1.7 DC Motor

Element used in the movement of the robot. It has a positive pin which powers the motor and a negative pin connected to the ground. Depending on the voltage provided, the number of rpm (revolutions per minute) varies. The wheels are attached to the driving shaft of the motor.

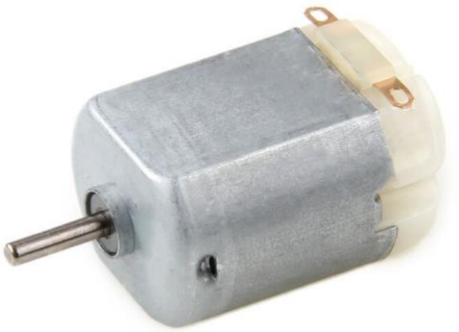


Figure 11. Direct Current motor.

2.2.1.8 ESP32

Microcontroller. It possesses the features and capabilities of the Arduino Nano alongside the WIFI functionality. The mentioned microcontrollers cannot work at the same time; if we want to load a program to the robot, we must choose one of two kinds of microcontrollers. The ESP32 can accept Arduino-C programs from our Arduino IDE (specific drivers must be installed beforehand in the IDE), but also it can run micropython using other tools. In our case, we have decided to use the ESP32 LOLIN; there are other feasible possibilities such as the ESP-DEVKIT, but the size of the microcontroller make interchangeability unfeasible between ESP microcontrollers.

Figure 12. ESP32 LOLIN without pins.

Sometimes the micropython installed in the ESP32 makes the microcontroller incapable of receiving an Arduino-C program. To fix this issue, there are some external tools that enable us to erase such unwanted functionality. In our case, we used ESPTOOL and the command: *"esptool.exe --port [communication port of the ESP32 connected to our computer] erease_flash"*. Depending on the hardware, you may have to push or not the reset button of the ESP while running the command. As we are trying to explain this issue in a general way, if the results are not as shown below, we highly encourage you to search for a particular solution on the internet. There are many user communities out there that might help you out fixing your specific hardware and making it run as intended.

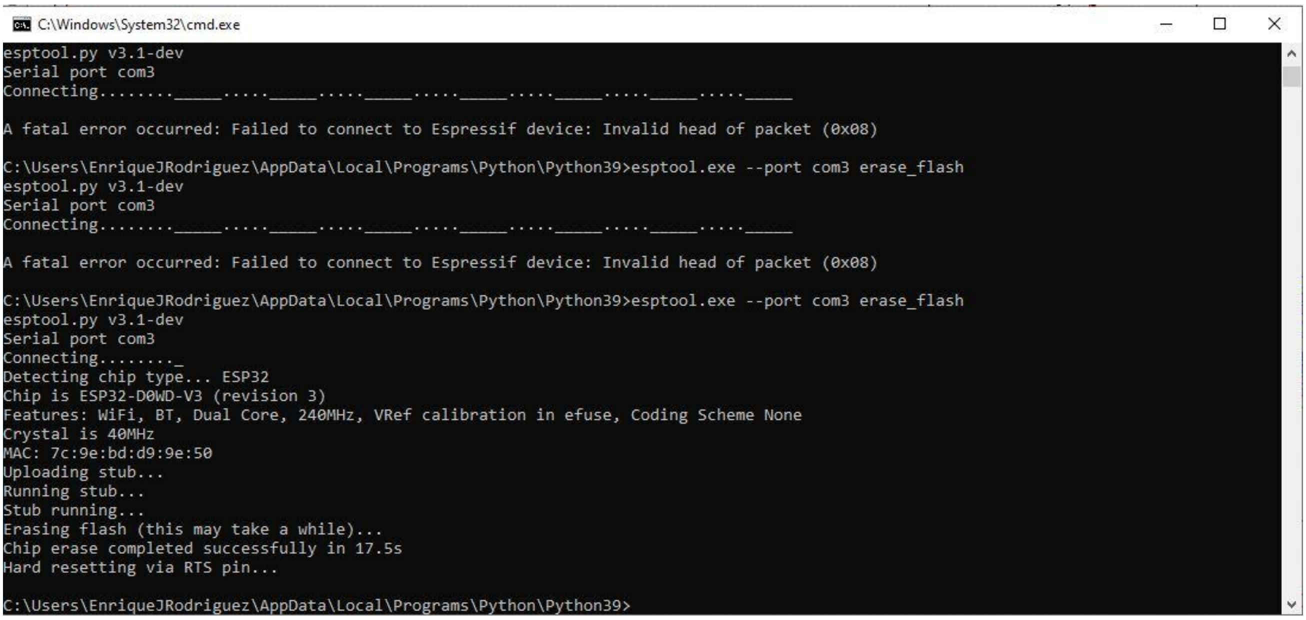


Figure 13. CMD prompt during esptool.exe use.

You may also encounter another problem related to uploading your programs to the ESP32 microcontroller. The device can run in two bootloader modes: normal execution and bootload. In order to use ESPTOOL or to upload new programs, the device must switch to bootload mode. A good ESP32 does this change using its own internal software and hardware, although this may not be your case. Sometimes, it may be as simple as pressing the reset button while uploading your program from the Arduino IDE; however, sometimes you may have to ground the GPIO0 pin (check your specific hardware pinout for reference) in order to properly upload the program. Once the fix is done and the program is loaded, the device is ready to use.

2.2.1.9 HC-SR04

Ultrasonic ranging module. It has 4 pins: +5V power supply, trigger pin to send the signal used to operate the module, the echo pin that sends back to the microcontroller the read data, and a ground pin. It makes use of ultrasound to determine the distance to an object in a range between 2 cm and 4 m, with an accuracy of ± 3mm.

Figure 14. HC-SR04 sensor.

2.2.1.10 HC-SR501

Infrared sensor. Using the PIR technology it can detect movement in a 120º angle within 7 metres. It has 3 pins: VCC power supply, DOUT for obtaining the read of the module, and a GND ground. This model (501) has built in controls which handle the following: how the infrared pulse is sent, sensitivity adjustment, and time delay.

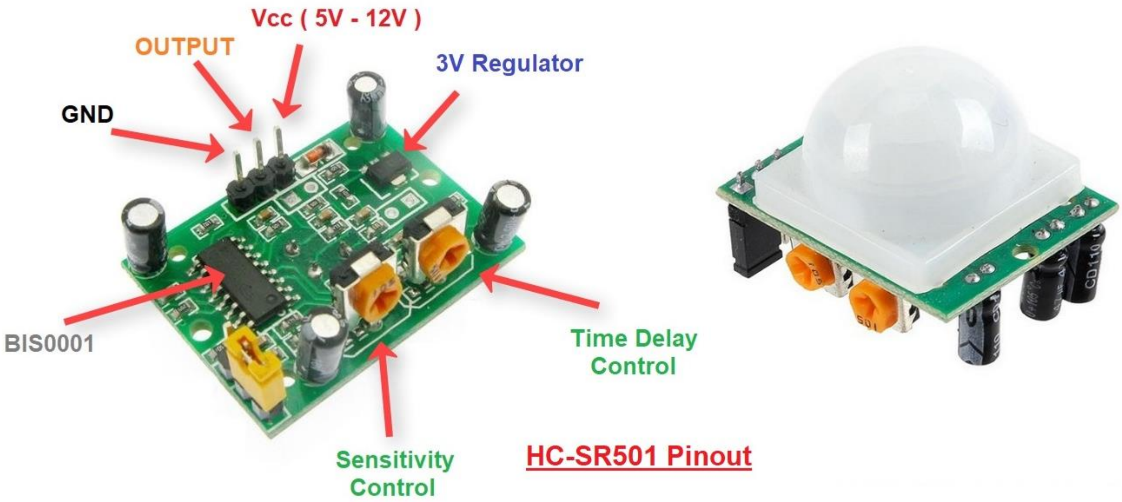


Figure 15. HC-SR501 sensor.

2.2.1.11 Level converter

Pre-built component that enables us to transform a signal ranging from +5V to 0V to +3V to 0V and vice versa. It could be done manually, but as the small components needed for that option must be soldered in a reduced space, the option was discarded. Our level converter is able to handle the transformation of two signals simultaneously. The B stands for +3V and the A for the +5V signals.

Figure 16. Level converter module.

2.2.1.12 NEO6MV2

GPS module. It provides an external antenna attached to the main component and needs up to 10 mins to properly receive a complete GPS signal. This device cannot be tested in a breadboard powered by the Arduino; some sort of external power supply is required. It has 4 pins: VCC power supply, TX transmission pin, RX receiving pin, and a GND ground.

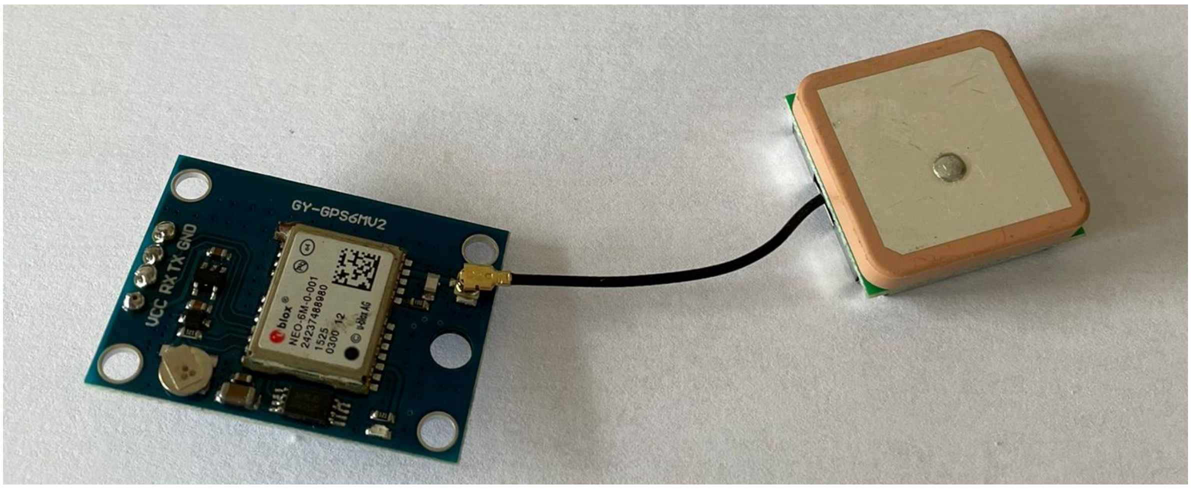


Figure 17. GPS sensor.

2.2.1.13 Photodiodes

Used in the line following functionality. The circuit is composed of a normal white LED and two photoreceptors (tinted in black) that output a signal depending on the reflected light they detect from the normal LED. The "bulbs" have two pins: a long one (anode, positive) and a short one (cathode, negative).

Figure 18. Photodiodes.

2.2.1.14 Servo SG90

Robotic arm. Its movement is determined by a SG90 DC motor. The motor has 3 pins: +5V VCC power supply, PMW analogic control signal, and GND ground.

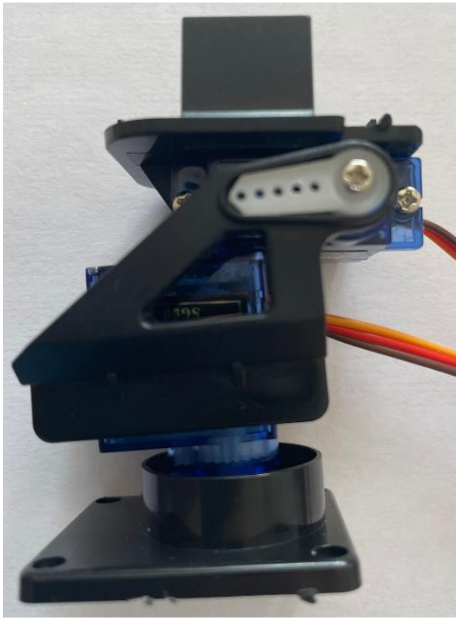


Figure 19. Robotic arm with SG90 motors.

2.2.1.15 TB6612FNG (Motor driver)

Motor control driver. It enables us to handle two wheels with the microcontroller. It consists of an H-Bridge that allows us to change the polarity of voltage, permitting the motors to run forwards and backwards. It has pins for: each wheel, ground, control, and power supply.

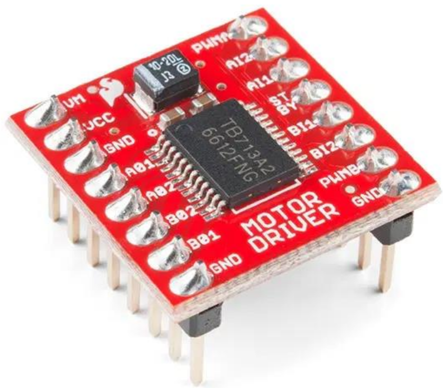


Figure 20. Motor driver.

2.2.1.16 XBEE

Serial communication receiver. It allows us to send/receive messages to the robot; it uses its own protocol and must be configured using the corresponding tool, XCTU. Its purpose is enabling the robot with an alternative way of communicating with other devices. In our case, we are only using 3 pins: GND, DIN for receiving the data from the microcontroller, and DOUT to send the data wherever.

Figure 21. XBee module.

2.2.2 Electrical schematic

In the previous section, we have reviewed the sensors and components that are projected for our robot. We are going to use EasyEDA to create the schematics. EasyEDA is an electronic computer-aided design tool, which is available on web version and desktop application. The tool integrates various functionalities in an interface similar to that of Draw.io, so that almost everyone can create complex schematics in a simple, yet powerful environment.

In our case, we will be using the web version of the platform. Once we have registered a new account, we may use the designer tool. Its interface should look something like Figure 22. EasyEDA basic view.:

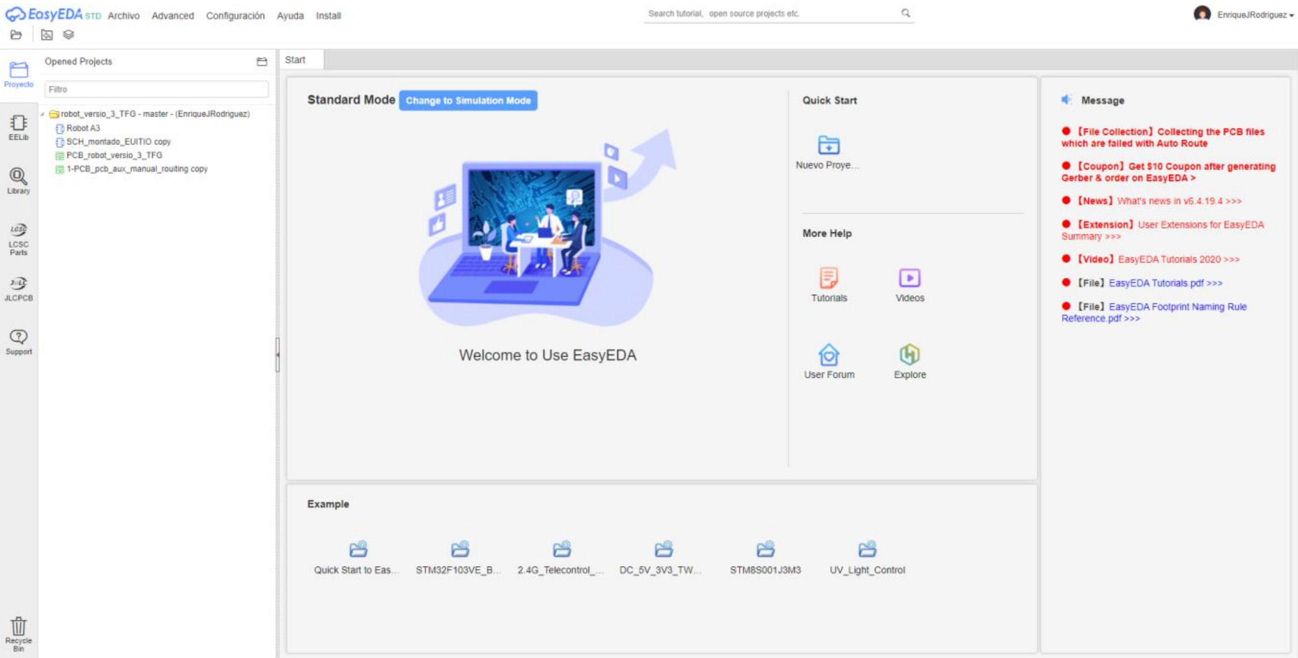


Figure 22. EasyEDA basic view.

To start a new project, we simply access to Archive < New < Project… You have to provide the basic information of the project: owner, title, route (if you are using the desktop version) and a brief description. EasyEDA has an integrated version control, so you can share the project with other users that may contribute to the project. It also provides permission control and ownership transferals.

Once we have our project created, we may start our electrical schematic; to do so, we select Archive<New<Schematic. We will have something like Figure 23. EasyEDA electrical schema designer view.:

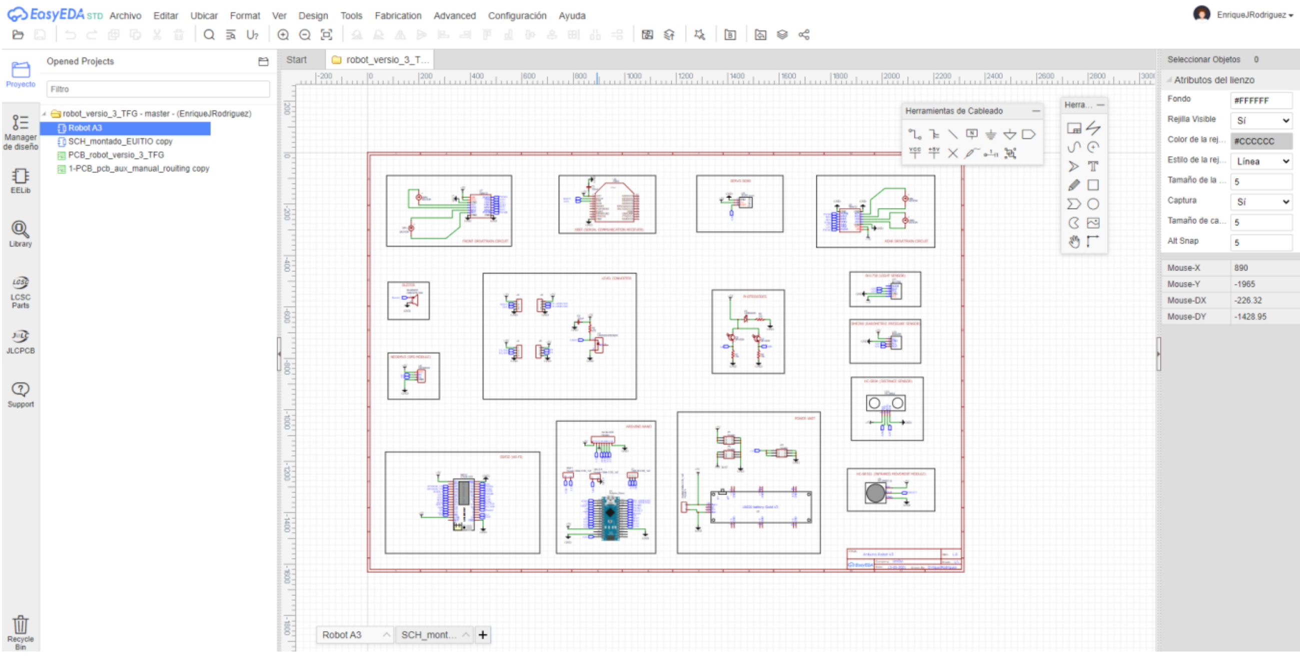


Figure 23. EasyEDA electrical schema designer view.

Even though a large set of various tools is at hand, the most important ones are wiring tools, drawing tools, EELib and Library. EELib allows us to place basic components such as capacitors, resistors, or pin headers. The library tool allows us to search for complex components: you can use official parts which can be bought from JLCPCB shop and ordered with your PCB, or employ user created components. You must take into consideration that user created components may have design issues, so a more careful selection must be done.

Once you have the desired component selected, the tool lets you place the component representation in the sheet. Then, you can start putting labels to the component's pins and "connect" each individual pin using this feature. You must take into account that the more detailed the labelling you perform is, the easier and less user-dependant the process will be once you decide to transform the electrical schematic into a PCB diagram.

There are some components that may not appear even in the user contribution part of the library; such occurrence is not a major problem. EasyEDA provides as well the necessary tools to create new components that you can use and share with the community.

The electrical schematic is just a representation of your design. It is a good practise to organize the components by functionality and thus reducing the complexity of the diagram.

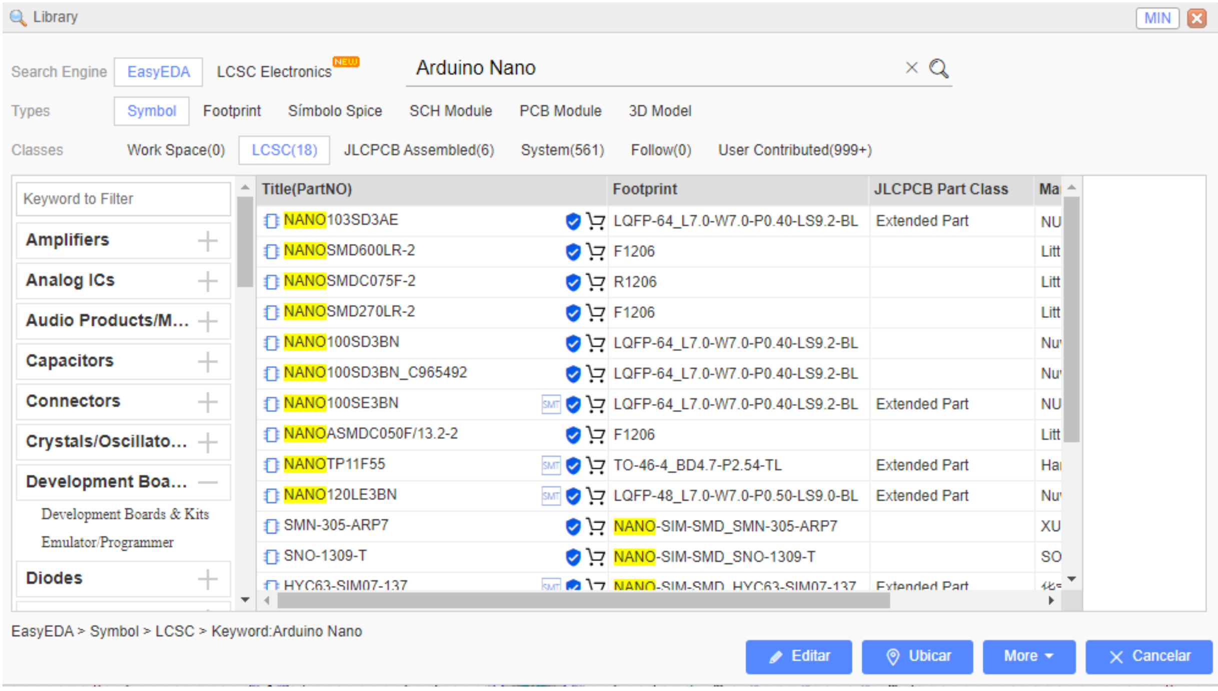


Figure 24. EasyEDA component library browser.

Another consideration while utilizing the user contributed components is pin layout. You can select a component you need; however, the pins might be mirrored, thus forcing you to solder the component according to that specific layout or manually modifying your own copy of the component to fulfil your needs.

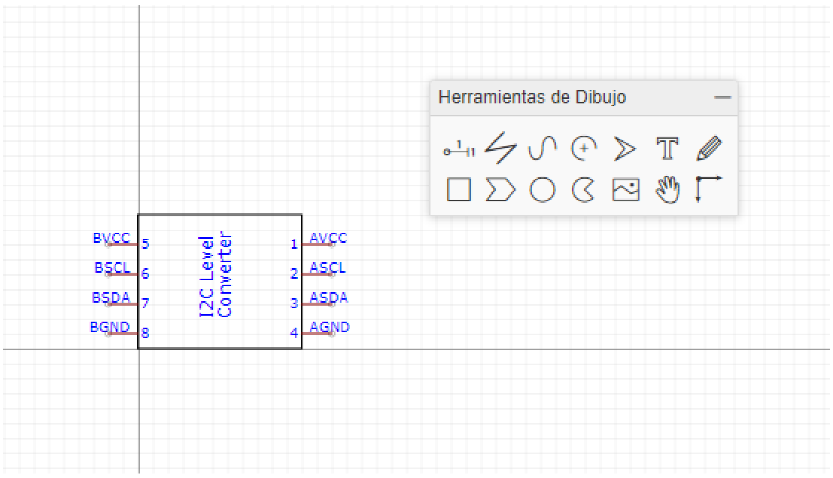


Figure 25. EasyEDA component designer view.

While you are filling the diagram with all the components required there is a moderate possibility of having to take some compromises. In this project, for example, the microcontrollers did not have enough pins to individually control the two drive trains. Depending in the modularity and flexibility targeted, you may want to reduce the number of "fixed pins"; in other words, a more flexible design is one whose components' pins are not labelled for a specific functionality. The components listed in *Component description* are placed and labelled thus obtaining the final electrical schematic (Figure 26. Robot's electrical schema.):

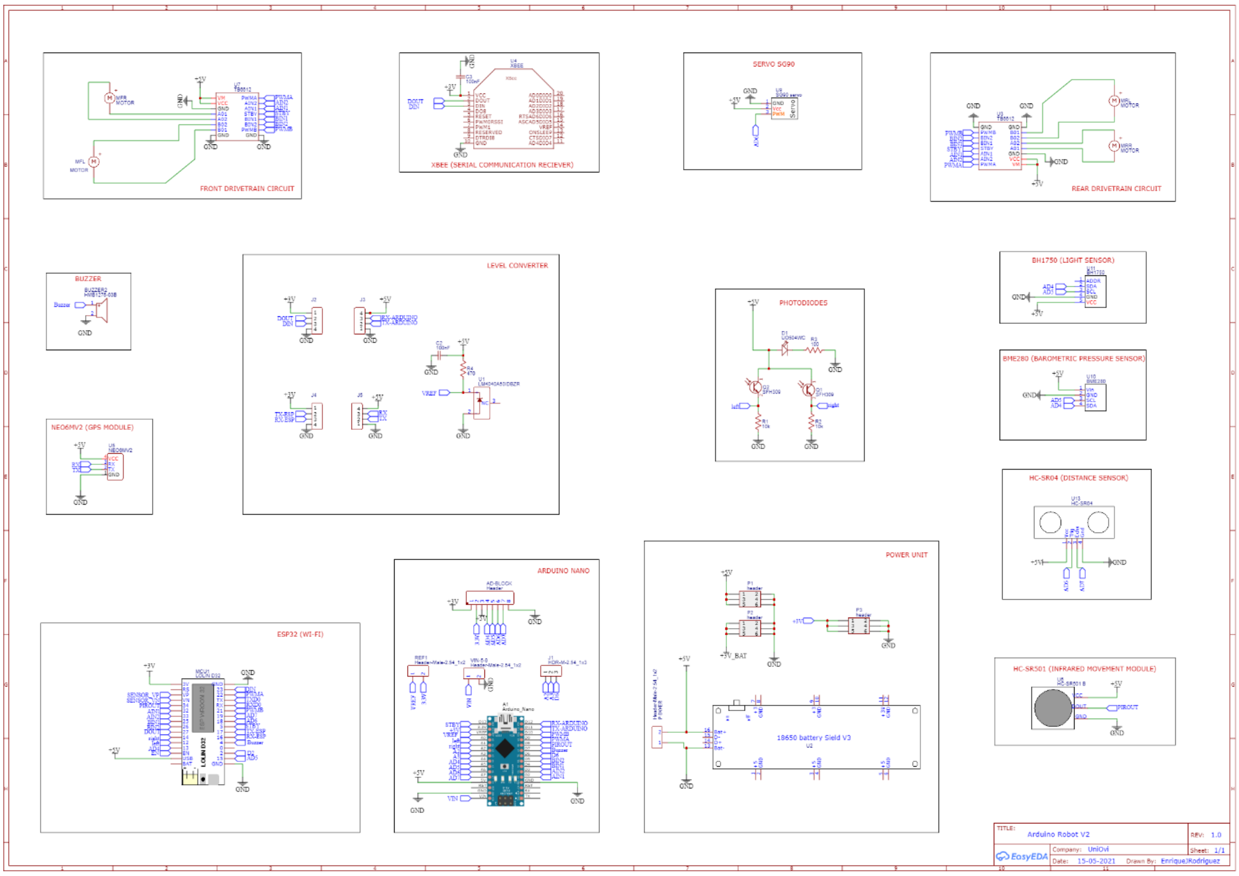


Figure 26. Robot's electrical schema.

The labelling process is the result obtained by the combination of the analysis for each pinout belonging to the components listed as well as the tests done to the components in protoboards. As we already saw, some components "reutilize" microcontroller pins, which is the result of functionalities such as I2C components.

2.2.3 PCB schematic

Once we have the electrical schematic ready, we may start the PCB design process. EasyEDA allows their users to create a PCB from scratch (Archive<New<PCB); such option is not recommended, as the effort required may duplicate or even worse. Some features, such as using the same pins to connect multiple sensors (I2C), would not be available from the start. The best option is transforming our electrical schematic into a PCB diagram; to do so, you should open the electrical schematic and press Alt-P or Design<Convert Schematic to PCB. If every component is labelled, after some dialogs, you will be sent to the PCB design view. In this view, you can adjust the size of the PCB and you can manually arrange all the components described in the electrical diagram.

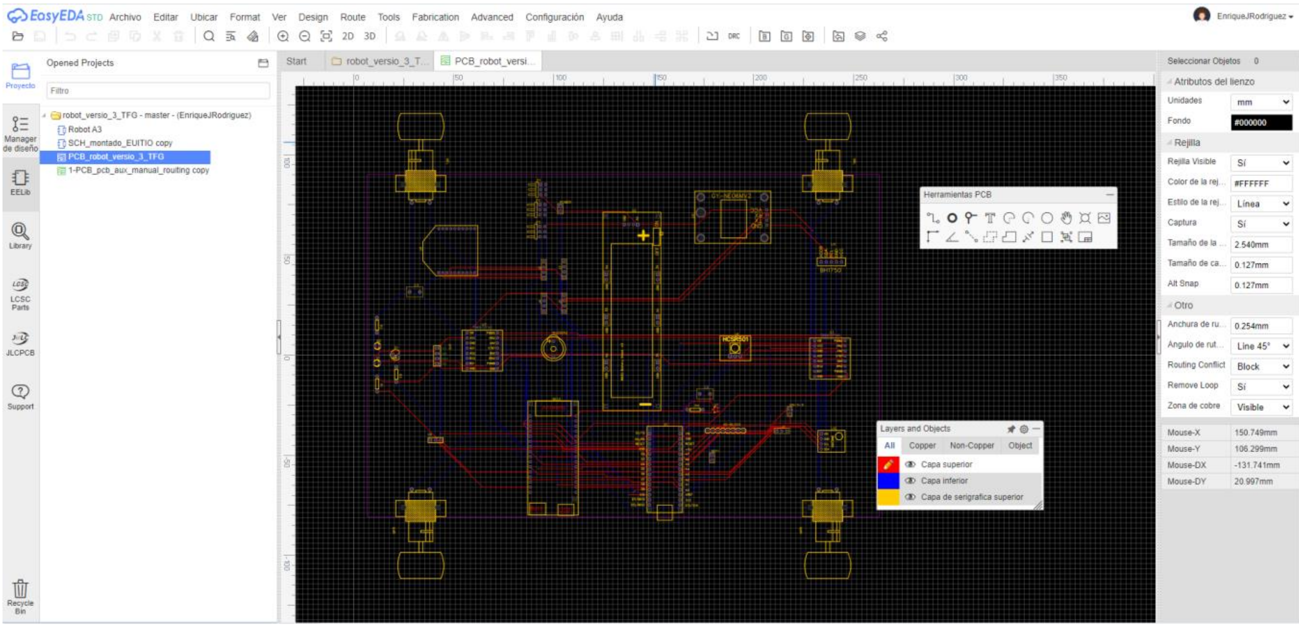


Figure 27. EasyEDA PCB designer view.

This view also provides some interesting tools such as a ruler or a hole design tool (to create holes for wires or screws). The ruler enables you to manually measure some components in case you need some precise placement, or if you want to adapt some component to your necessities.

Once you have all the components placed where desired, the pins and holes printed in the PCB, you may start wiring the components. There is a manual tool to handle the tracks in the PCB; nevertheless, that is an advanced user feature. For this project as well as non-complex diagrams, EasyEDA provides an auto-wiring tool that should work properly; this feature also has the advantage that the track layout is short-circuit proof. To use the auto-wiring tool use Route < Auto Router… In the web application, there is a server-side routing tool (usually, you are automatically disconnected for performance reasons) or you have the possibility to download a script which you must run locally and send the data to the web application. After the routing tool has completed the processing, you obtain something like Figure 28. Robot's final schematic render.:

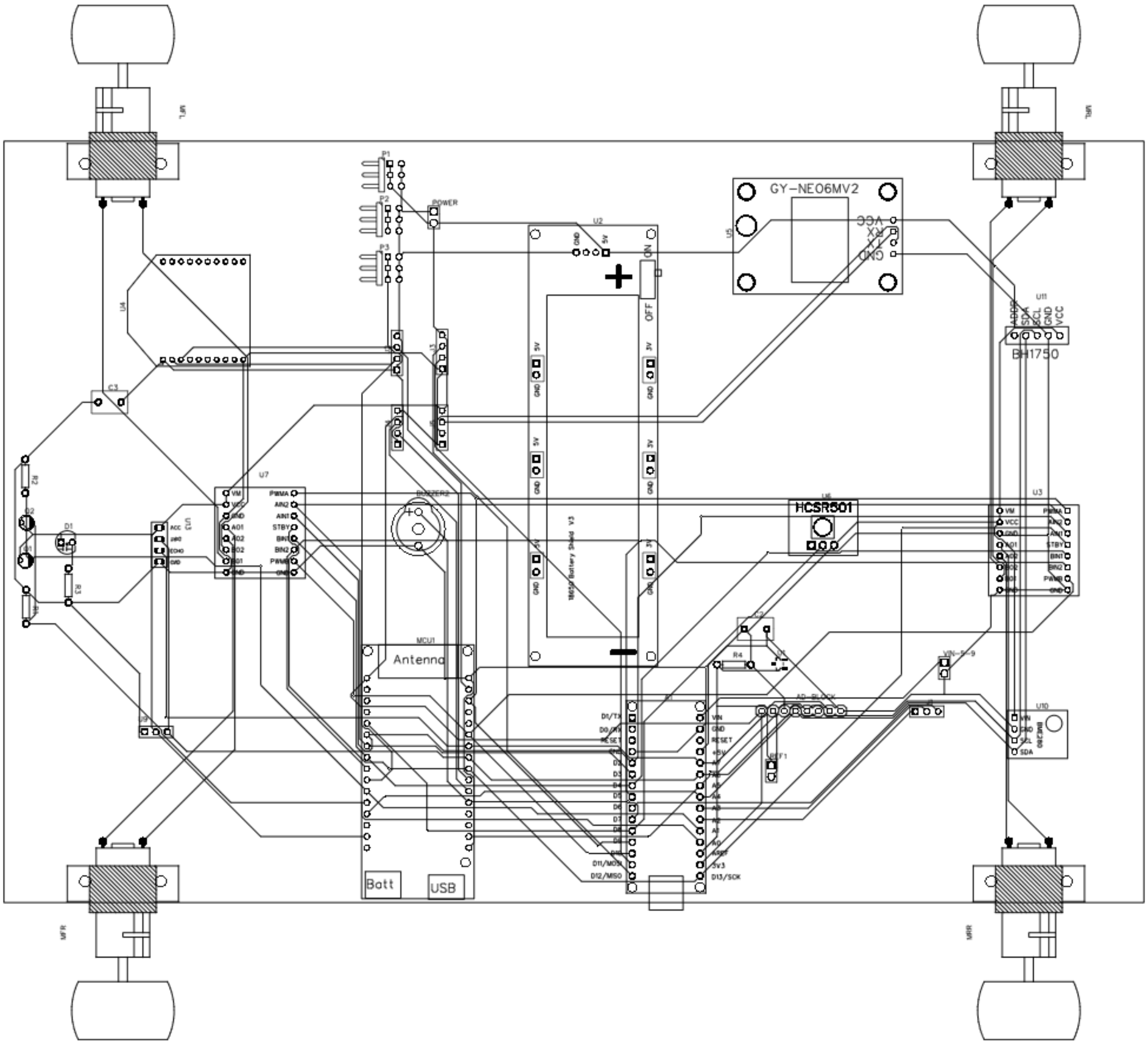


Figure 28. Robot's final schematic render.

After this phase is completed, you can order from JLCPCB (integrated in EasyEDA) the PCB. To do so, Fabrication < Order PCB. You will need to provide some mailing data and credit card information in a series of dialogs before completing your order.

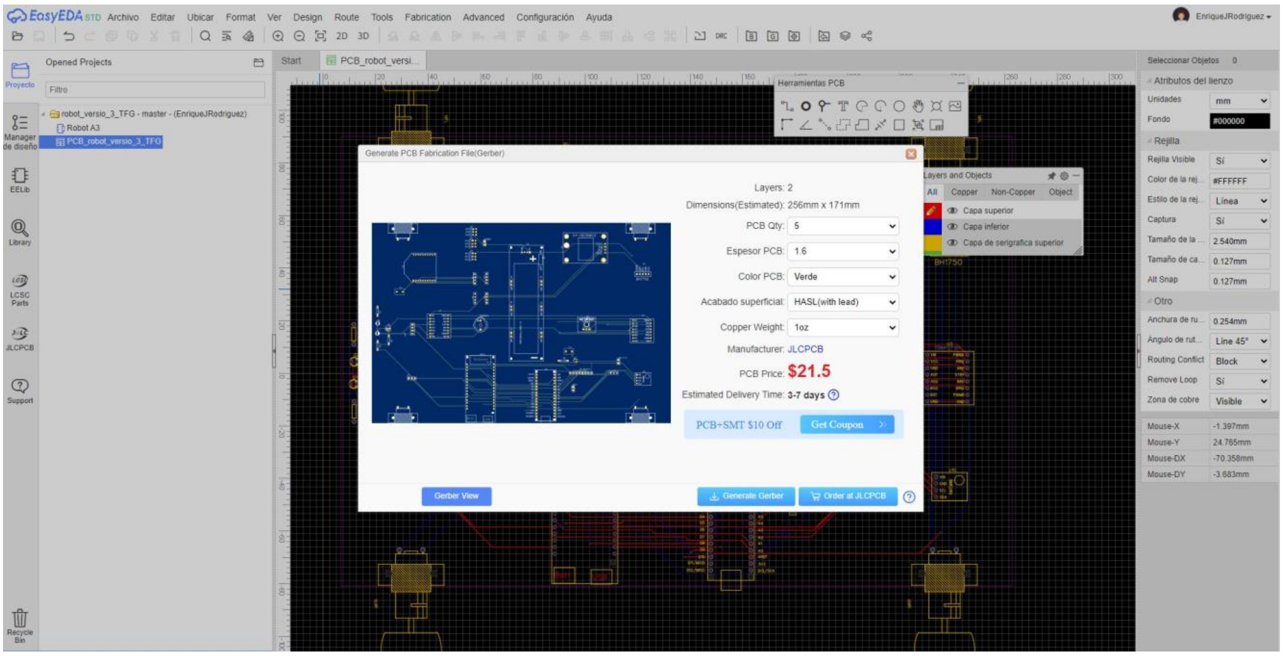


Figure 29. EasyEDA ordering dialog.

Depending on the options selected, the PCB will take a number of days of manufacturing and shipment. Some features, like a specific PCB colour or the quality attributes of the PCB, may increase several days the process.

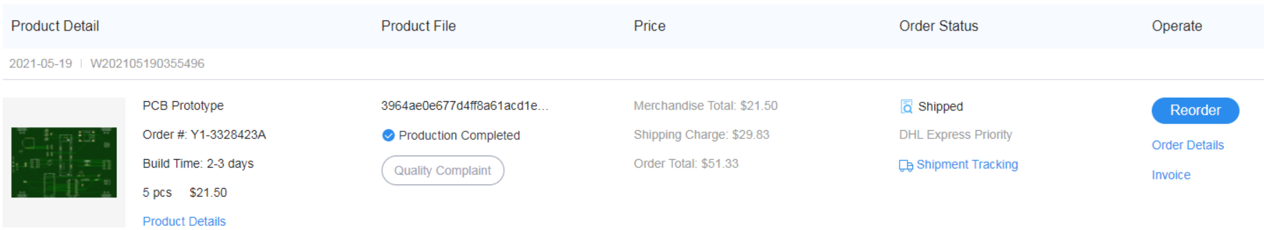


Figure 30. JLCPCB order status view.

2.3 Prototype construction

In this section, we are going to see how the prototype of the robot is finally assembled. We can divide this process in two important parts: the first one is to assemble those modules which are not already prepared to be used from the factory; the second part is to adapt our PCB for the components that will be mounted using female pin strips.

2.3.1 Tools and materials

For achieving our build goals, we need a set of tools that will make our progress much easier. In this section we are going to list the tools and other resources that will allow us to achieve our objectives.

2.3.1.1 Soldering Iron

It is used to provide heat to the tin required for soldering our components' pins. The extremely hot soldering tip must be handled carefully, as it may cause serious burns to the user if it is not properly manipulated. It is powered by a wire connected to a plug. The tip can be changed depending on the operation performed while the device is disconnected from the grid and the tip is cold.

The temperature is controlled by a knob and activated by a switch next to it. It also has a small LED to indicate that the device is hot and ready.

Figure 31. Soldering iron tool.

2.3.1.2 Tin

It is the material used in our solders. Such choice is due to its remarkable properties: relatively low melting point (less than 450ºC), anti-corrosion, rusting-proof, and great electrical conduction. It is usually referred as tin (Sn); actually, it is an alloy called *eutectic* composed by tin, lead (Pb), silver (Ag), copper (Cu) and nickel (Ni). The composition of the alloy determines the melting point.

To use it, cut a small piece using the tin reel. It can be bent with your bare hands with little to no effort in order to provide a better handling. You must melt the tin with the soldering iron, while trying to create a drop shape around the fixing point. It tends to stick to the soldering tip, so regular tip cleansing is highly recommended.

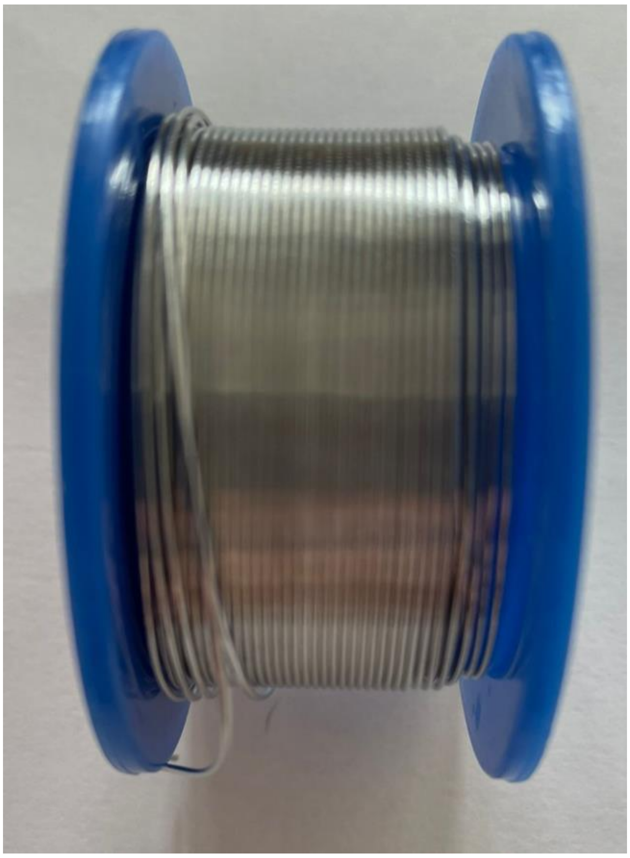


Figure 32. Tin reel.

2.3.1.3 Soldering stand

A device to grip temporarily the components and the pin strips in place while soldering. It includes a magnifying glass in order to see more clearly where you are placing the tin and the soldering iron. An important feature is the sponge; after soldering, the wet sponge allows us to safely remove the excess of tin in the soldering tip. The final accessory is the soldering holster, where you can place the soldering tool while it is not being used.

Figure 33. Soldering stand with sponge and magnifying glass.

2.3.1.4 Desoldering suction pump

A tool which is used after something goes wrong while soldering. Its purpose is quite simple: it just extracts the tin of a hot solder in order to remove two joint parts. Its internals resemble those of a syringe with the addition of a button which releases the tension of the internal piston.

Figure 34. Desoldering suction pump.

2.3.1.5 Tweezers

Multipurpose tool. Used to poke holes in obstructed pin holes thanks to its metallic pointy ends; handy when the desoldering suction pump is not enough. Also, it could be used to place some tiny parts such as nuts or pins.

Figure 35. Tweezers.

2.3.1.6 Screwdriver

Used to fix screws in place. Our model has the capability to change the tip for various types of screw heads.

Figure 36. Screwdriver.

2.3.1.7 Scissors

Tool used for cutting and strip wires.

Figure 37. Electrician scissors.

2.3.1.8 Pliers

Multipurpose tool: utilized for bending pins, wire pins, gripping materials and fixing nuts in place.

Figure 38. Needle nose pliers.

2.3.1.9 Multimeter

Our electrical "debugging" tool. It enables us to measure various electrical properties such as voltage, resistance and current. It is a precision instrument and could be easily damaged if mishandled. It has an LCD display to show the results of the measurements. A dial or rotary switch is used for selecting the property we are measuring with the red and black probes connected to input jacks. The black probe is connected to a common ground (GND) and the red one to a multipurpose jack, except for the ammeter capabilities.

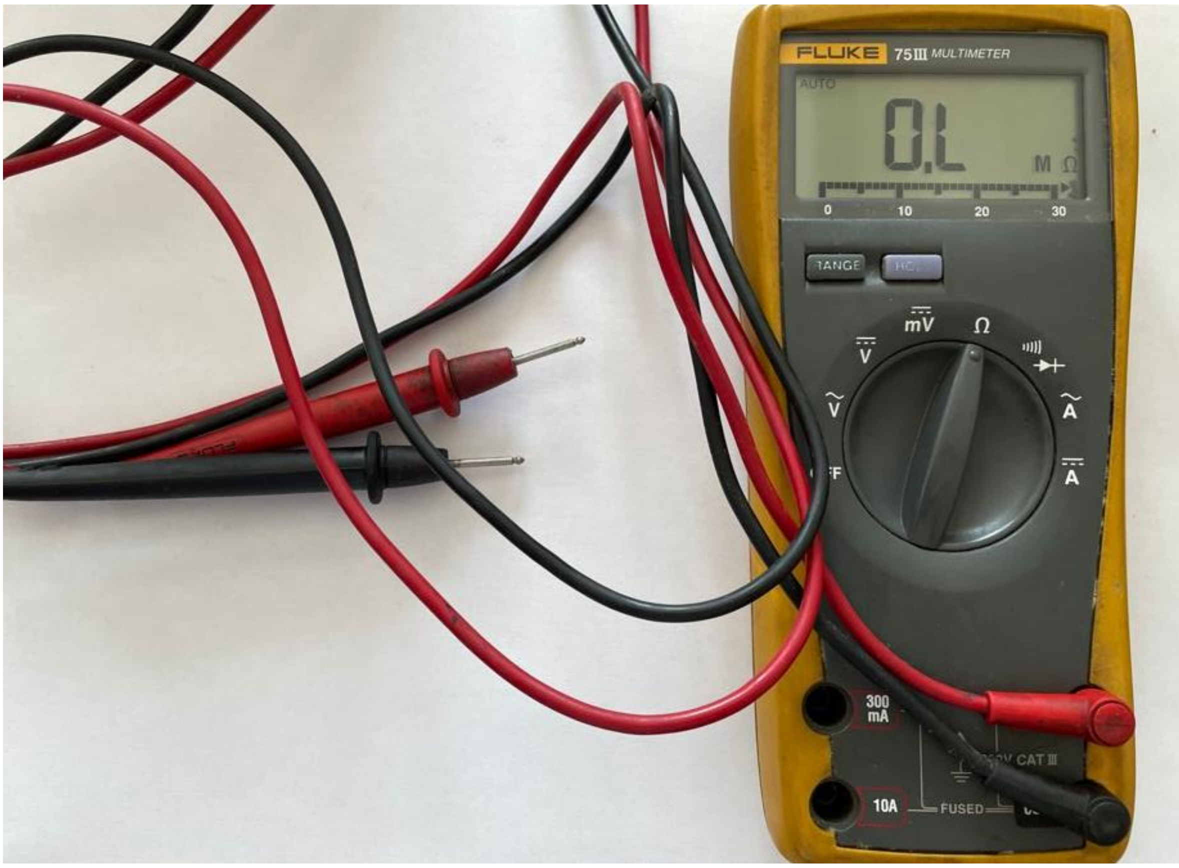


Figure 39. Multimeter with probes.

2.3.2 Module preparation

Some of the components required in this project are not ready to use from the package; you may have to manually solder the pin strips to the main component in order to be employed as intended. In this section, we are going to show which components required assembly while utilizing the tools listed in *Tools and materials section*.

The process is very simple: grip tight the main piece and the pin strip to the clamps of the support tool; then, using the solder, apply a small quantity of tin to only fix one pin in each movement. Use the magnifying glass of the support to see clearly where you are applying the tin. If a mistake occurs, make use of the desoldering suction tool to remove the tin and repeat the process again.

The following images illustrates the described process in the previous paragraph:

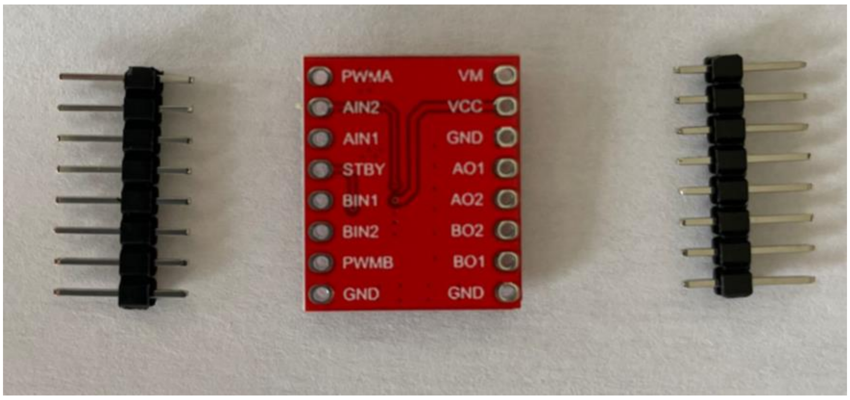


Figure 40. Wheel driver with unsoldered pins.

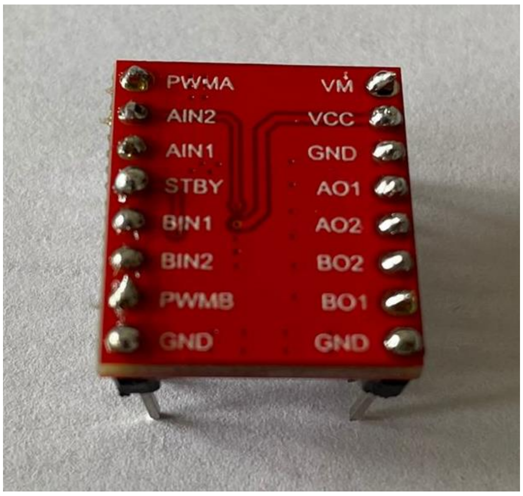


Figure 41. Wheel driver with soldered pins.

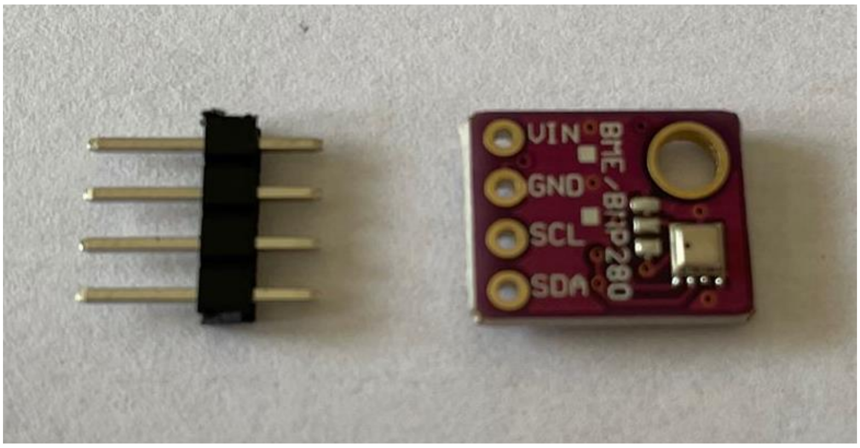


Figure 42. Barometric pressure sensor with unsoldered pins.

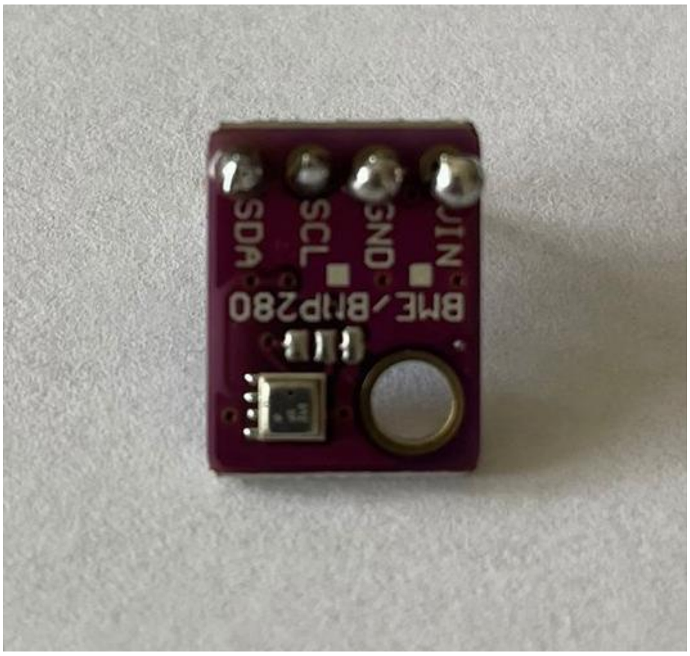


Figure 43. Barometric pressure sensor with soldered pins.

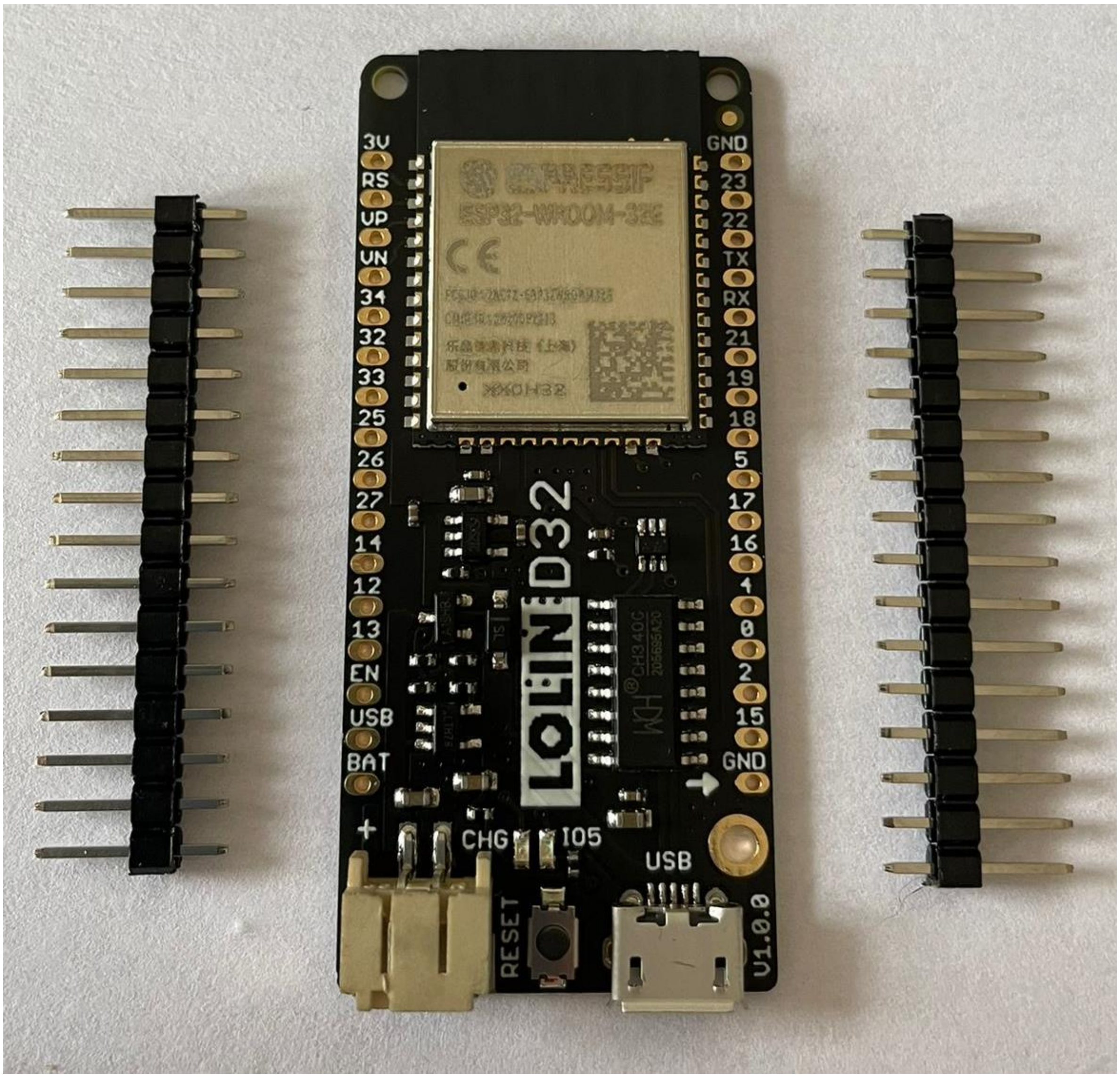


Figure 44. ESP32 LOLIN microcontroller with unsoldered pins.

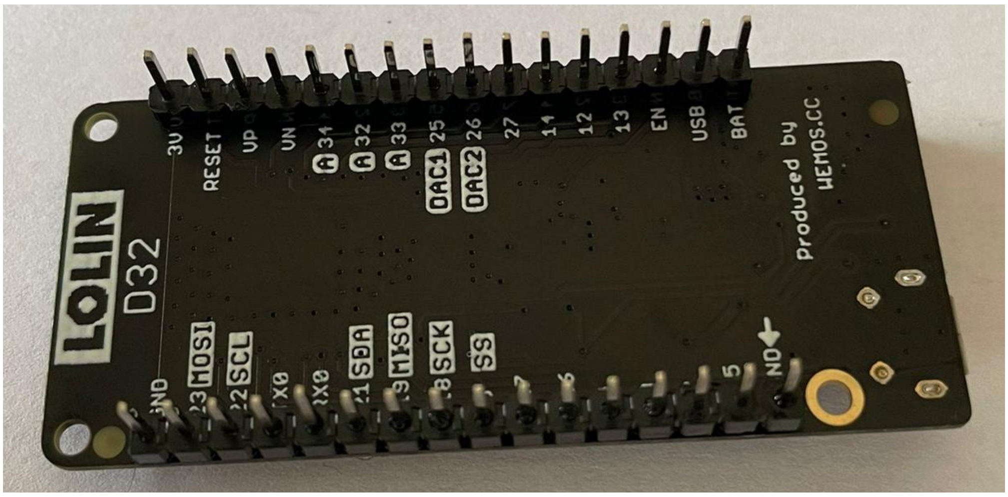


Figure 45. ESP32 LOLIN microcontroller with soldered pins (back side).

Figure 46. ESP32 LOLIN microcontroller with soldered pins (front side).

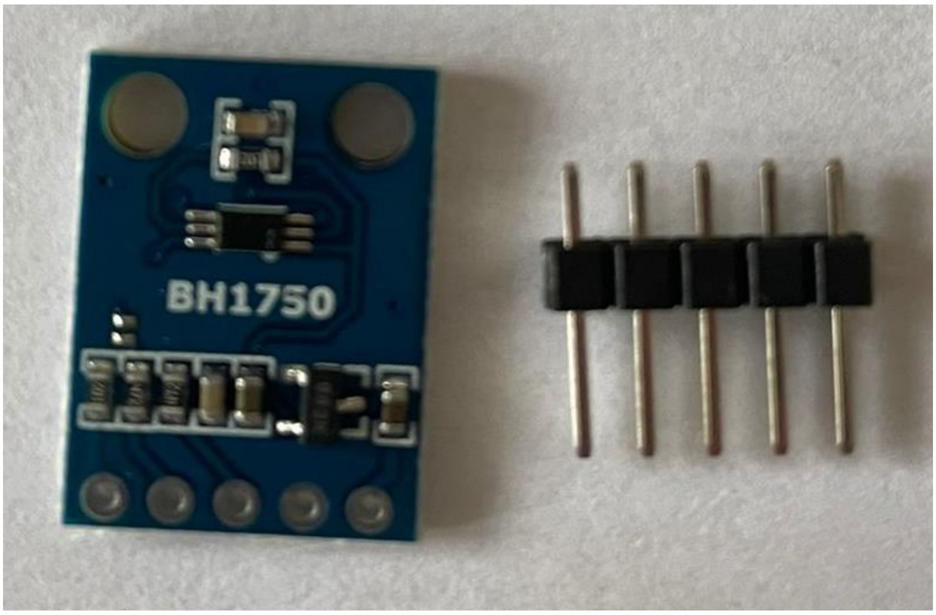


Figure 47. Light sensor with unsoldered pins.

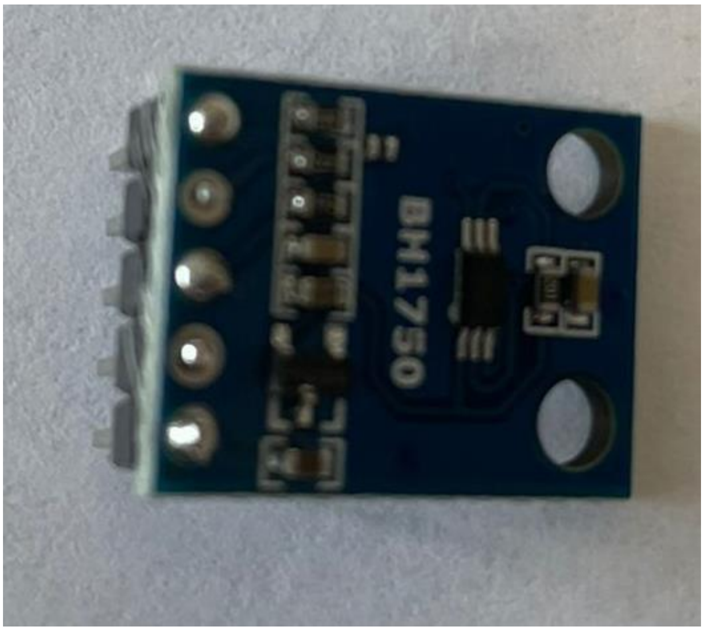


Figure 48. Light sensor with soldered pins.

Figure 49. Level converter with unsoldered pins.

Figure 50. Level converter with soldered pins.

The next module is the NEO6MV2. This GPS module demands an external antenna that must be connected to a special port in the main component; after soldering the pin strip, we connect the antenna to the mentioned port. The final position could be under the component, where glue may be used to fix the antenna in place.

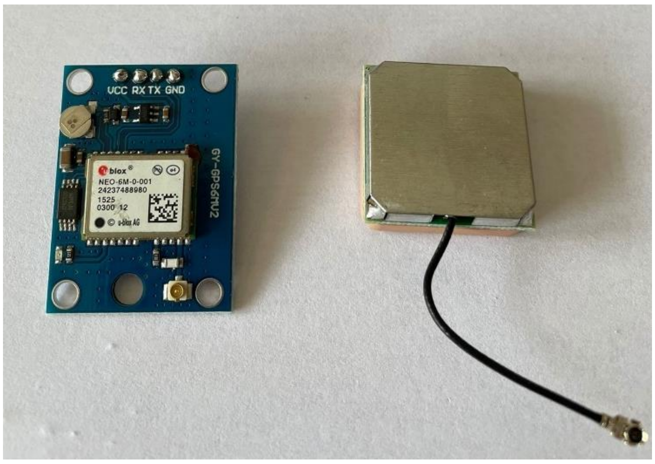


Figure 51. GPS module and external antenna.

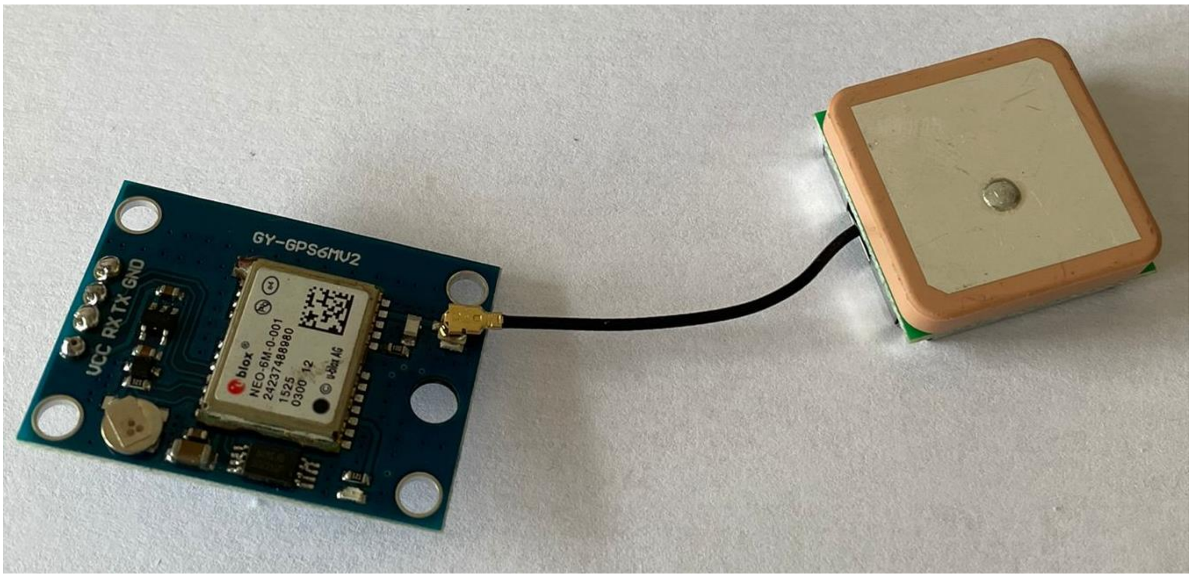


Figure 52. GPS module with connected antenna.

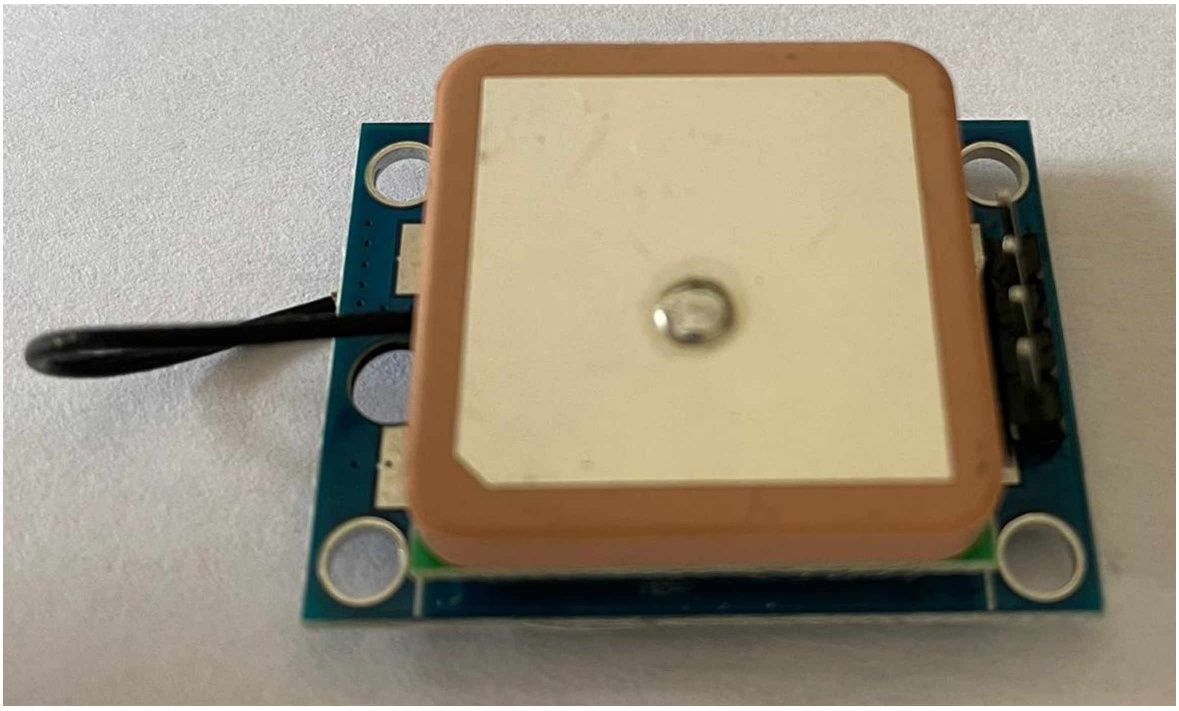


Figure 53. GPS module with connected and glued antenna.

The main component of the power unit is the 18650-battery shield. This module has many outputs for +5V and +3V. The main concern about these outlets is that there is no mechanism to cut these outlets from the battery, causing its drain and possible premature degradation. To avoid this problem, a modification of the shield must be done.

Figure 54. Original battery shield (top view).

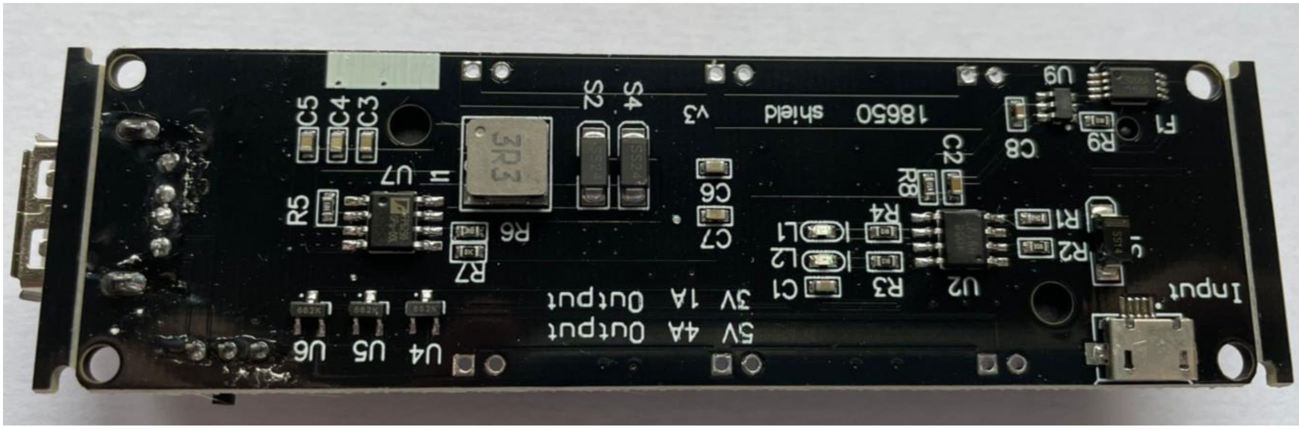


Figure 55. Original battery shield (bottom view).

The component has an USB female outlet, which is intended to use the shield as a portable charging point so that other electronic devices could be powered by this USB outlet. The USB port has a built-in switch, so it is a reasonable way to achieve our previously mentioned purposes.

As we can see in Figure 56. USB header pinout diagram., the USB port uses 4 tracks: the exterior ones are +5V and a ground while the inner ones are for data transmission. Therefore, we could obtain the desired functionality by just carefully removing the USB female head and soldering auxiliary cables that we may connect to the PCB.

If the length of the auxiliary wires is critical for the sake of aesthetics, you may cut in half a small wire as shown in Figure 59. Battery shield with connection wires (soldered to +5V and GND pins)..

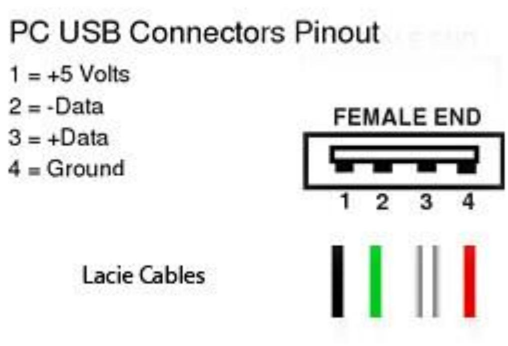


Figure 56. USB header pinout diagram.

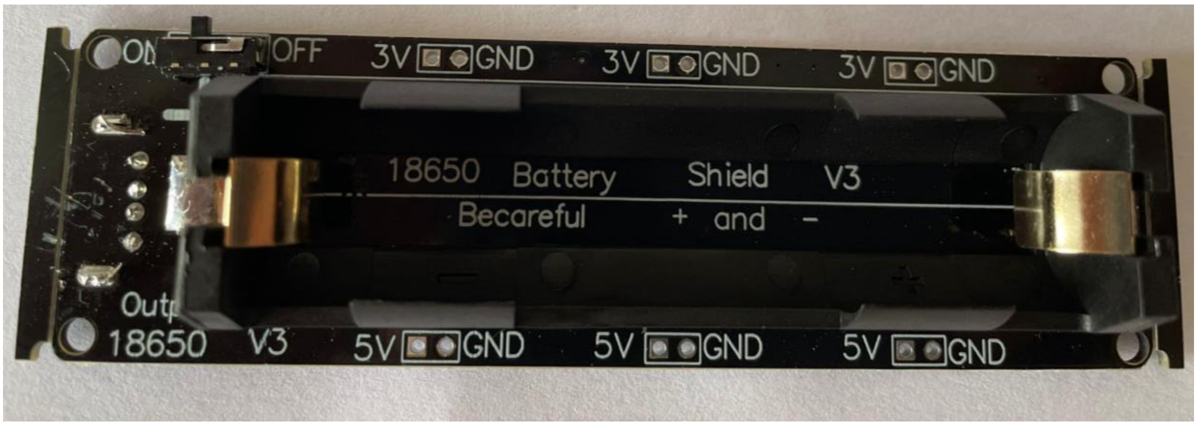


Figure 57. Battery shield without USB port (top view).

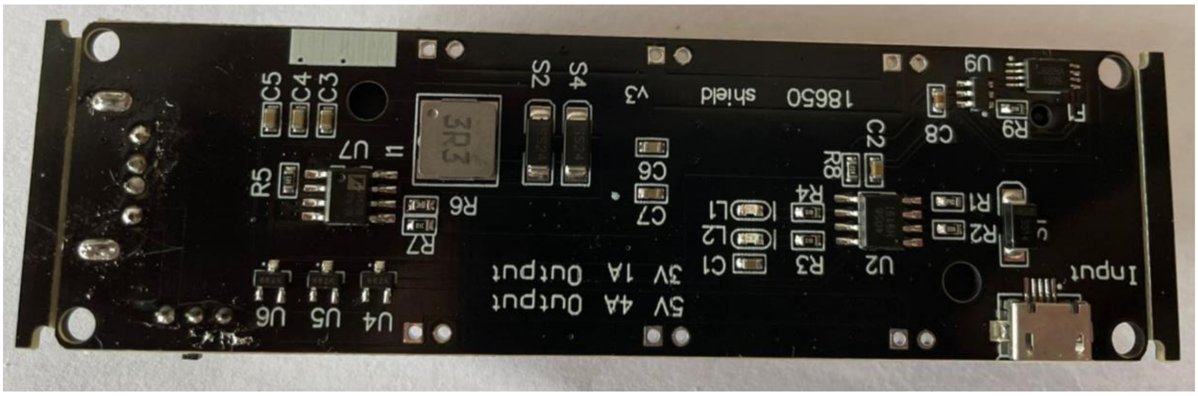


Figure 58. Battery shield without USB port (bottom view).

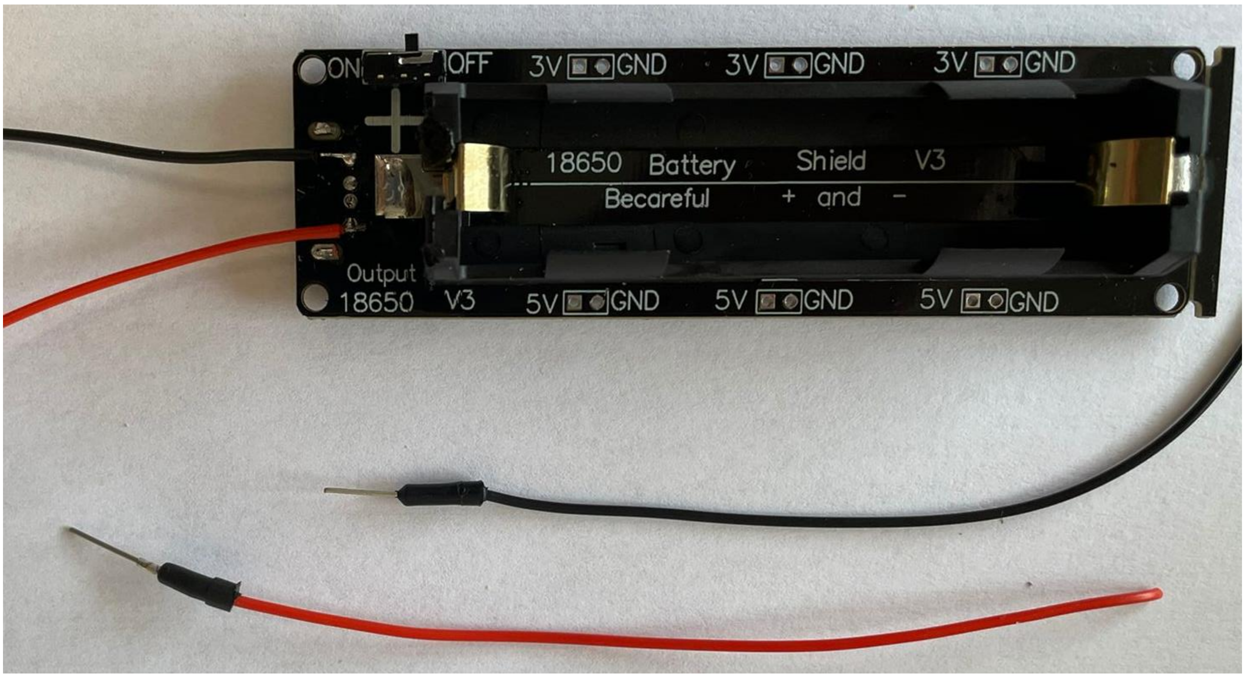


Figure 59. Battery shield with connection wires (soldered to +5V and GND pins).

Another possibility is to just solder a pin strip to the pin holes of the shield. In our specific case, this option was discarded, but it is a feasible option.

Figure 60. Battery shield alternative with pins.

The final piece that needed assembly was the robotic arm. In theory, the Chinese vendor stated that it had a total compatibility with SG90 DC motors and an easy assembly. In the following images and explanations, we will see clearly that major modifications were required to fulfil the intended purpose of the product.

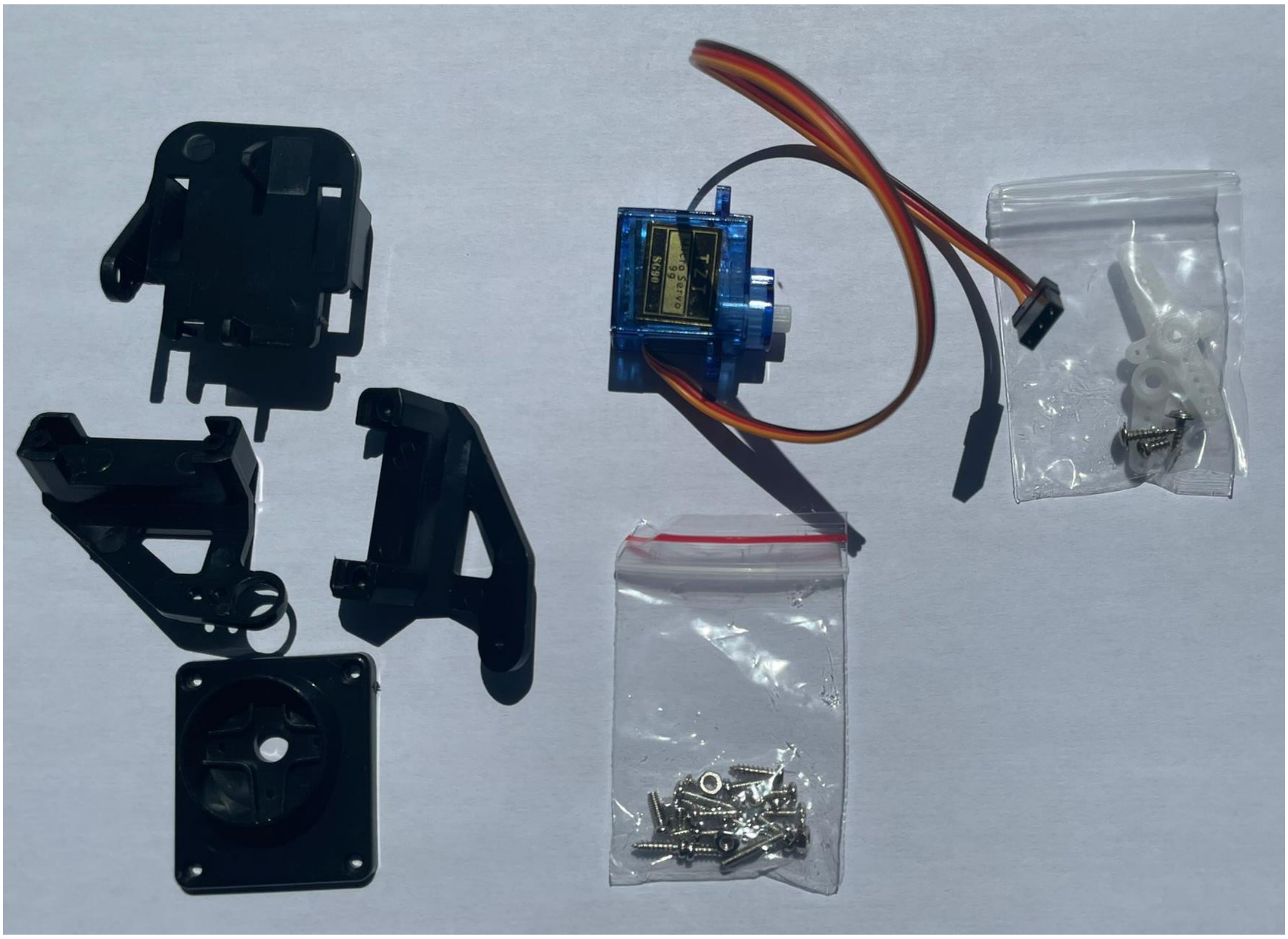


Figure 61. Robotic arm parts kit.

Firstly, we must prepare the base of the robotic arm. In this process, you will need the cited base (black plastic piece) and modify the motor adapter (white plastic piece) so that it fits in the intended space reserved in the base, as we can see in Figure 62. Base with plastic axis, original piece for reference.. In our case, we had to cut the adapter using a pair of pliers and sand the rough edges using a file. Finally, we screwed the adapter to the base for the intended fixed position.

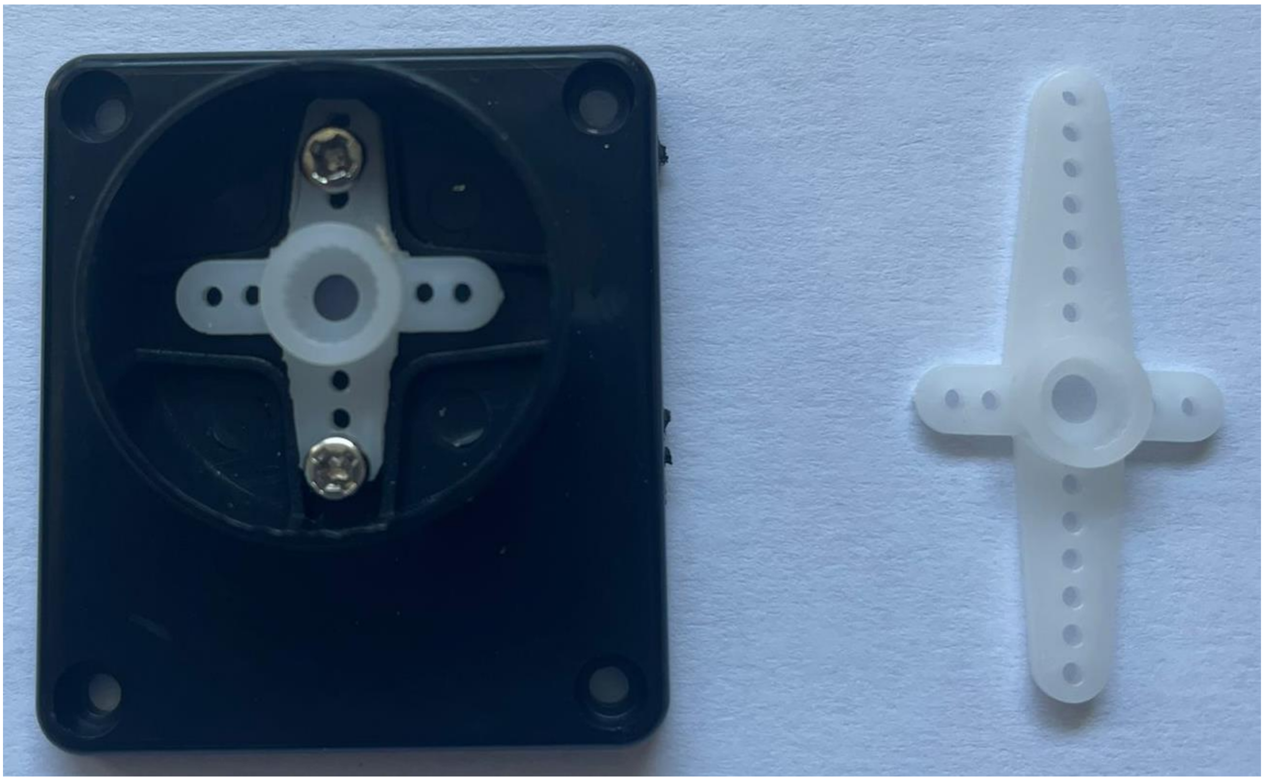


Figure 62. Base with plastic axis, original piece for reference.

Once we had our base almost ready, we had to assemble the SG90 DC motor to the pivotal platform that will be connected to our base. In our case, the great quality of Chinese manufacturing processes forced us to rasp the edges of our motor so that it could be fixed to the platform using Philips screws.

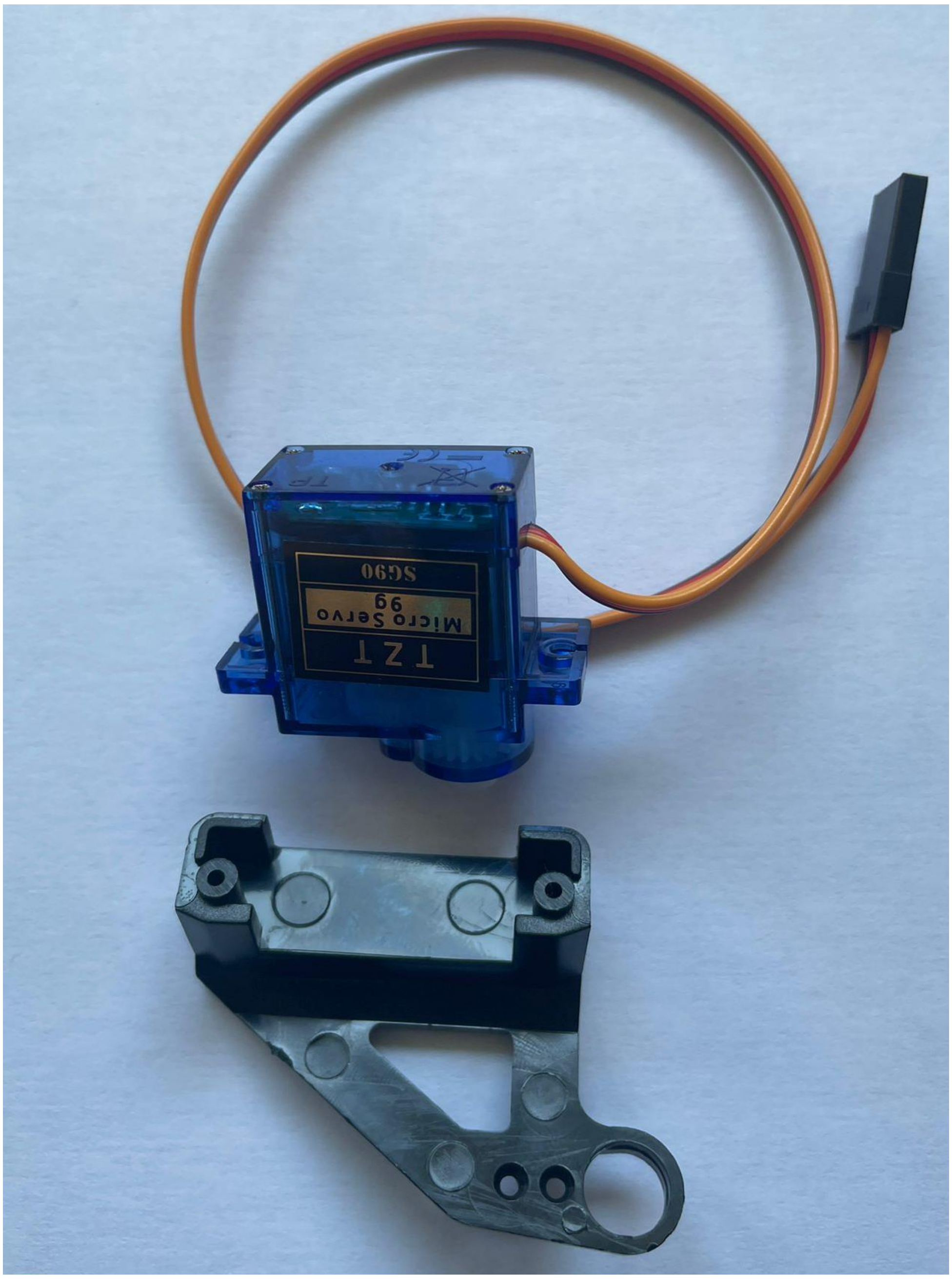


Figure 63. Pivotal platform and SG90 motor.

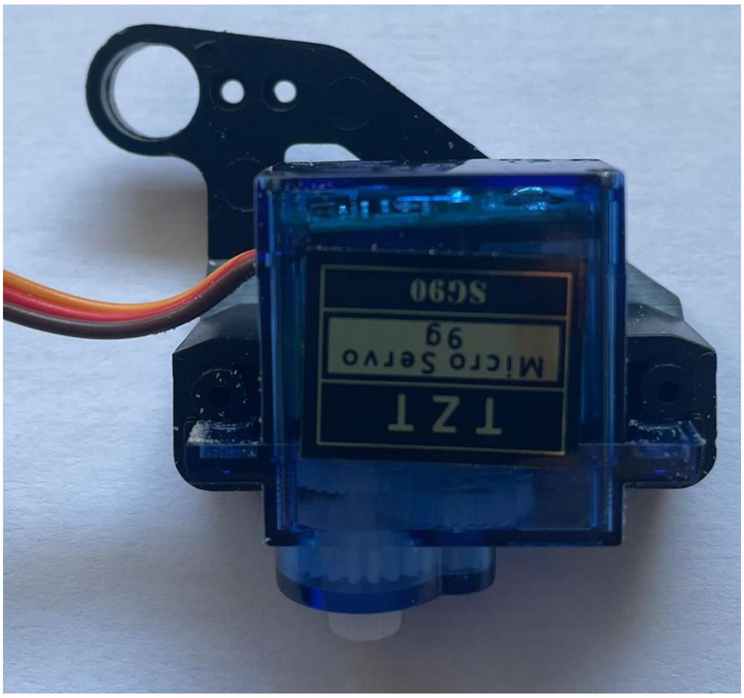


Figure 64. Rasped motor inside the first platform piece.

Figure 65. Fixed pivotal platform with motor inside.

Once we had the previous parts ready, we had to connect the adapter to the SG90 DC motor and fix it using a screw under the base, as we can see in Figure 66 and Figure 67.

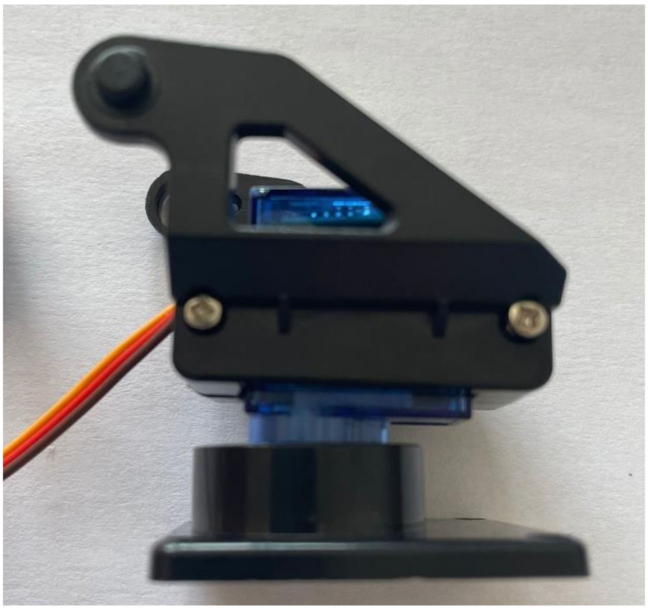


Figure 66. Base attached to the platform using the motor and axis.

Figure 67. Screw used to fix the motor in place.

In the next step, we fixed the second SG90 DC motor to the second platform. In our case, this motor is not intended for the normal configuration of the robot, but it is a nice addition to the robotic arm as a show-off of craftmanship. As we have a modular-designed robot, this second engine could be powered and controlled by the Arduino Nano. These steps are self-explanatory with the following images.

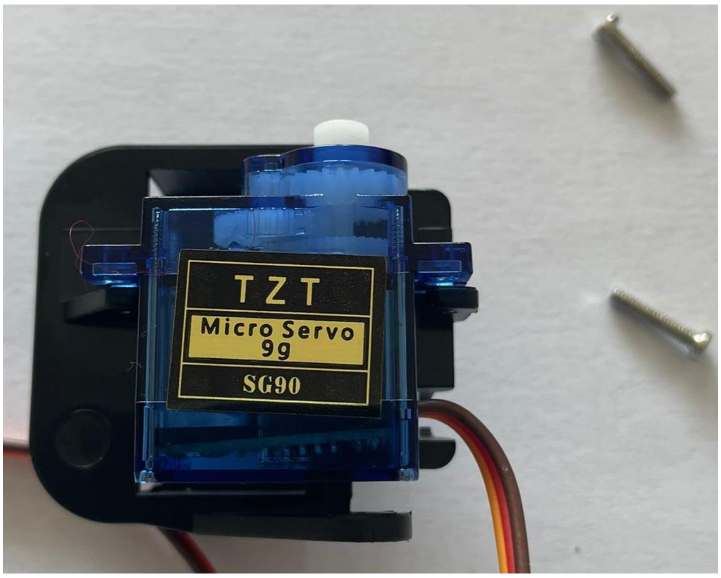


Figure 68. Second platform with motor.

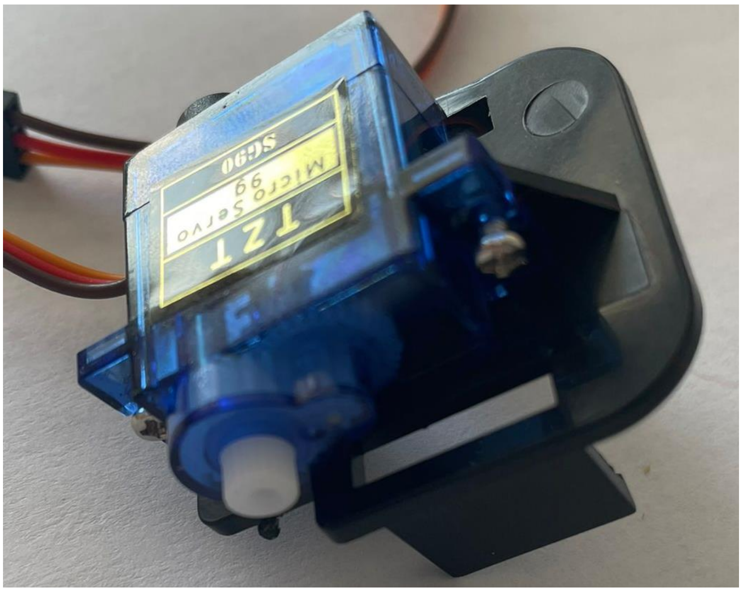


Figure 69. Second motor fixed by screws to platform.

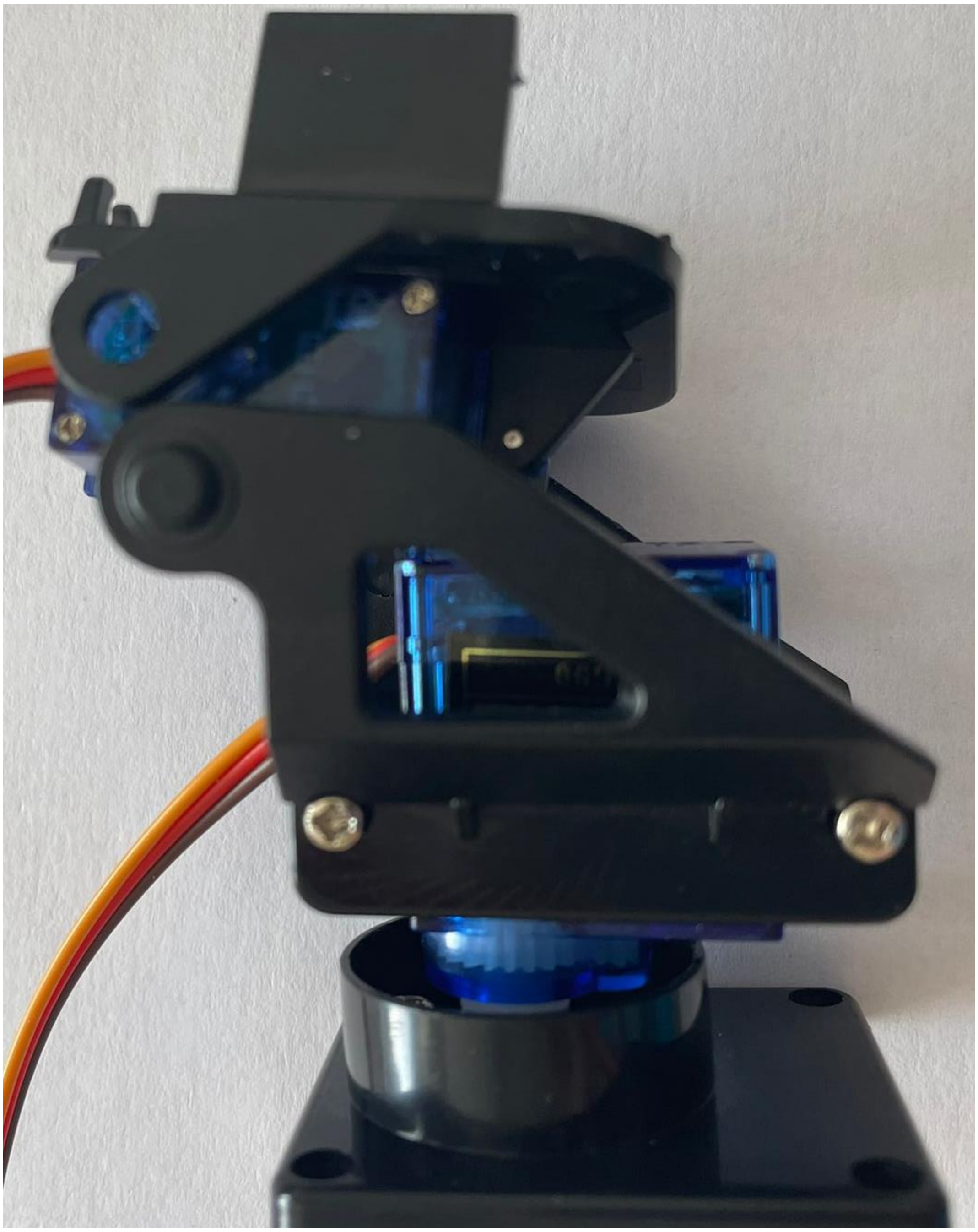


Figure 70. Second platform before fixing to first platform.

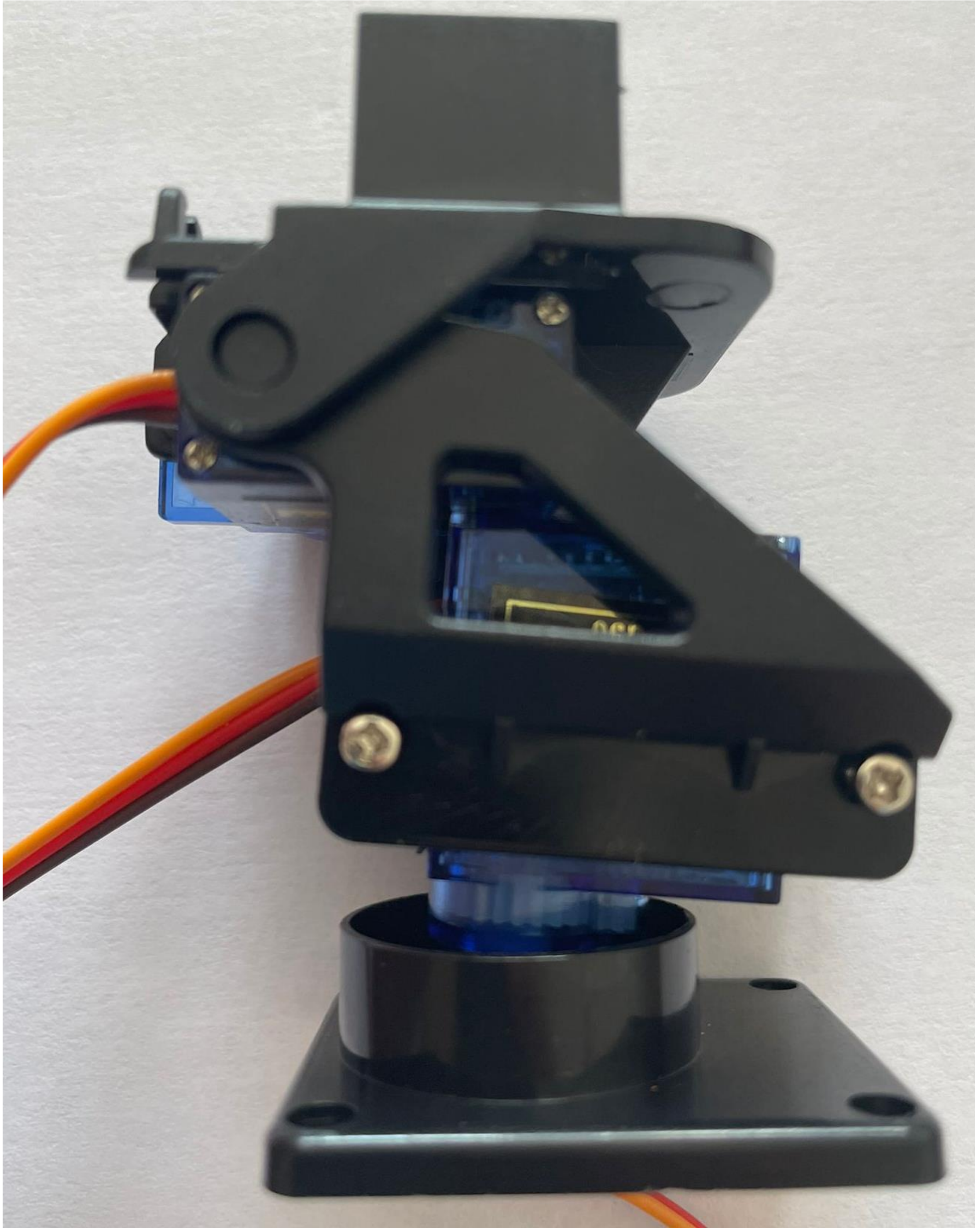


Figure 71. Both platforms attached mechanically.

Finally, we fixed the motor to the second platform; to achieve this goal, we had to once again perform a modification to the plastic adapter and screw it to the motor's gear in order to fix the second platform to the motor.

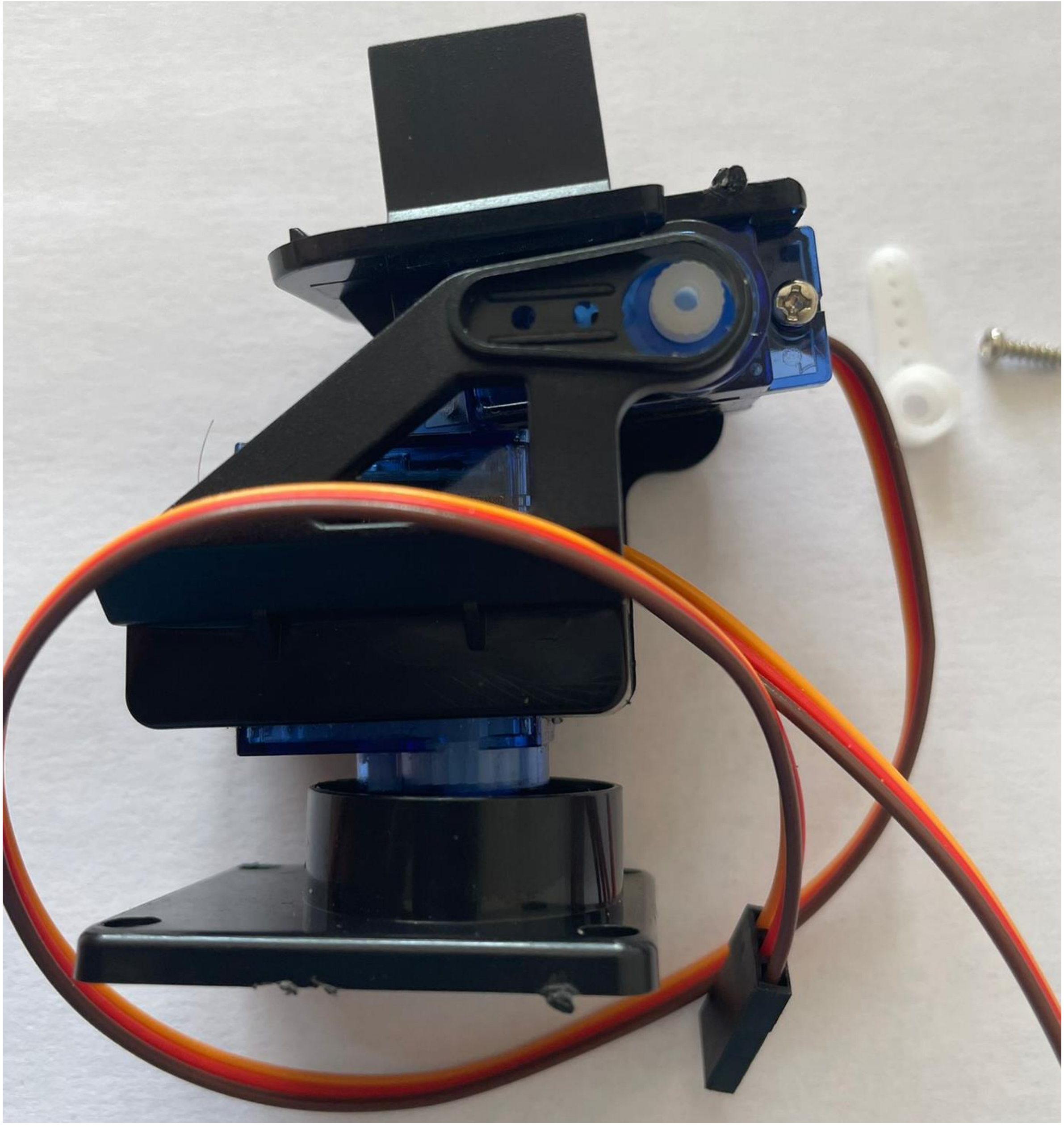


Figure 72. Second platform before adapting plastic axis.

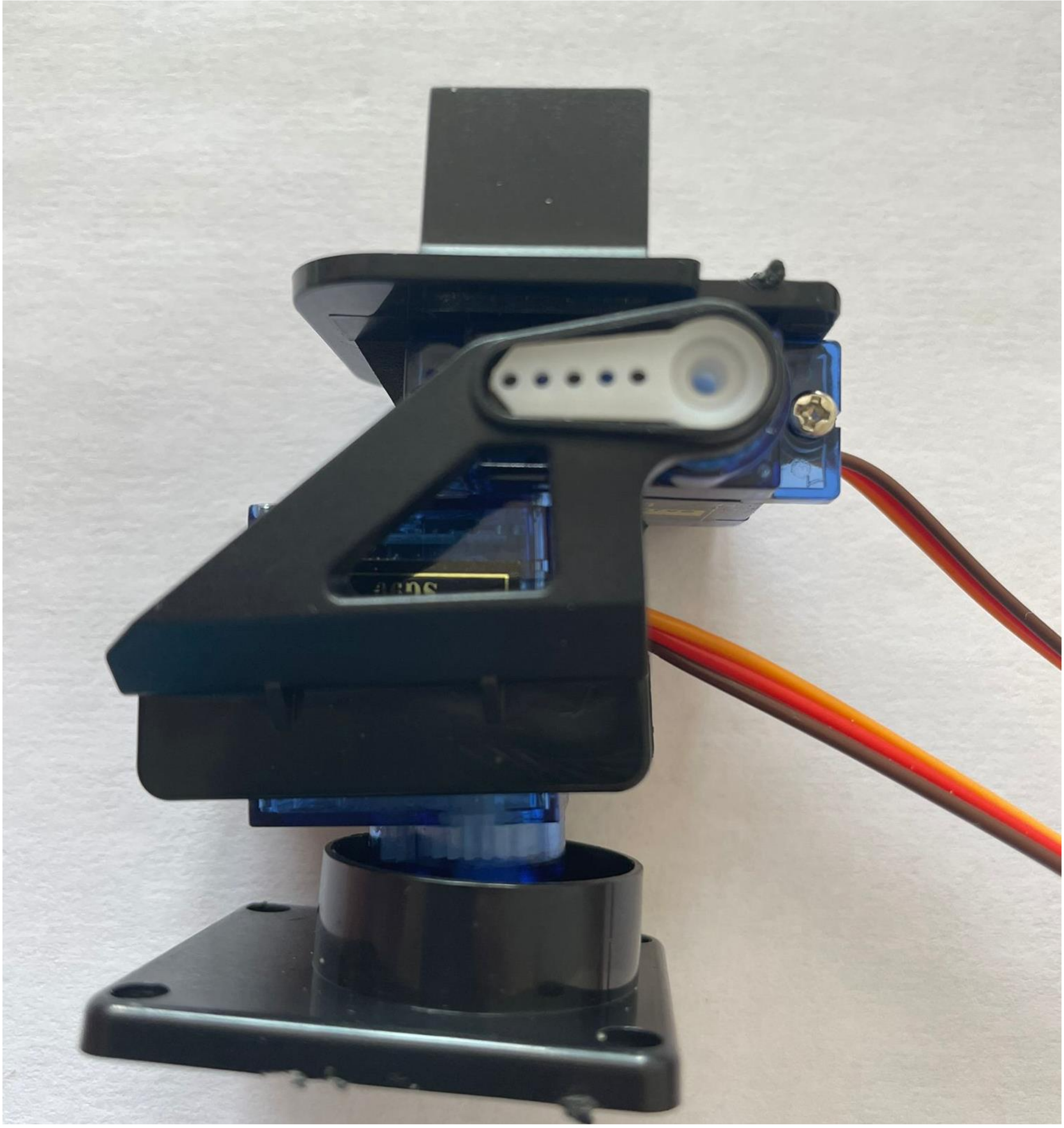


Figure 73. Plastic axis adapted.

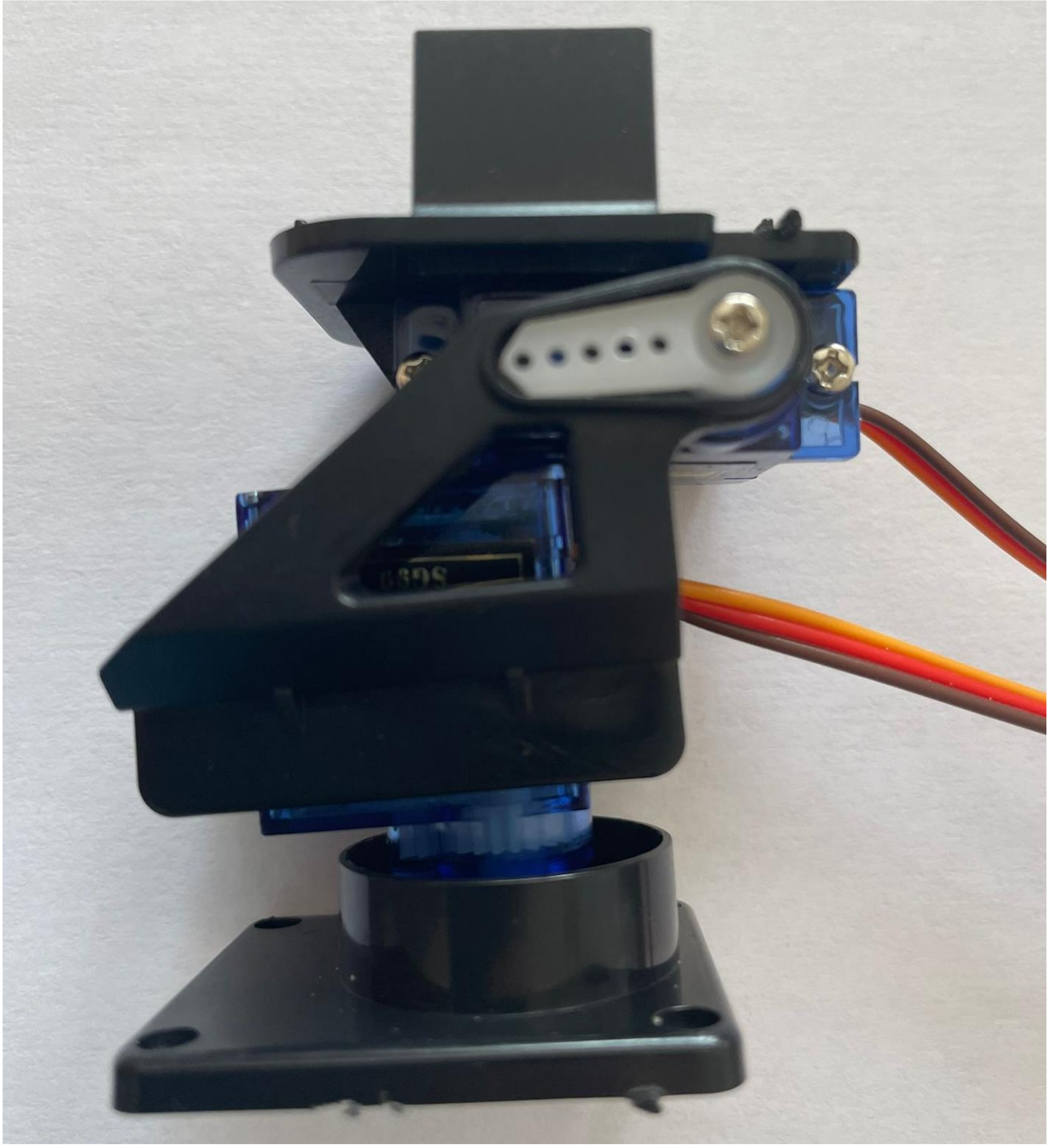


Figure 74. Robotic arm completed.

2.3.3 PCB assembly

In this section, we will describe how the prototype was finally prepared and assembled once we had all the modules and the PCB is in our hands.

As described in *Analysis & design of the robot*, JLCPCB sent us five copies of our designed PCB with the basic purchase. In our case, the PCB has no components mounted yet.

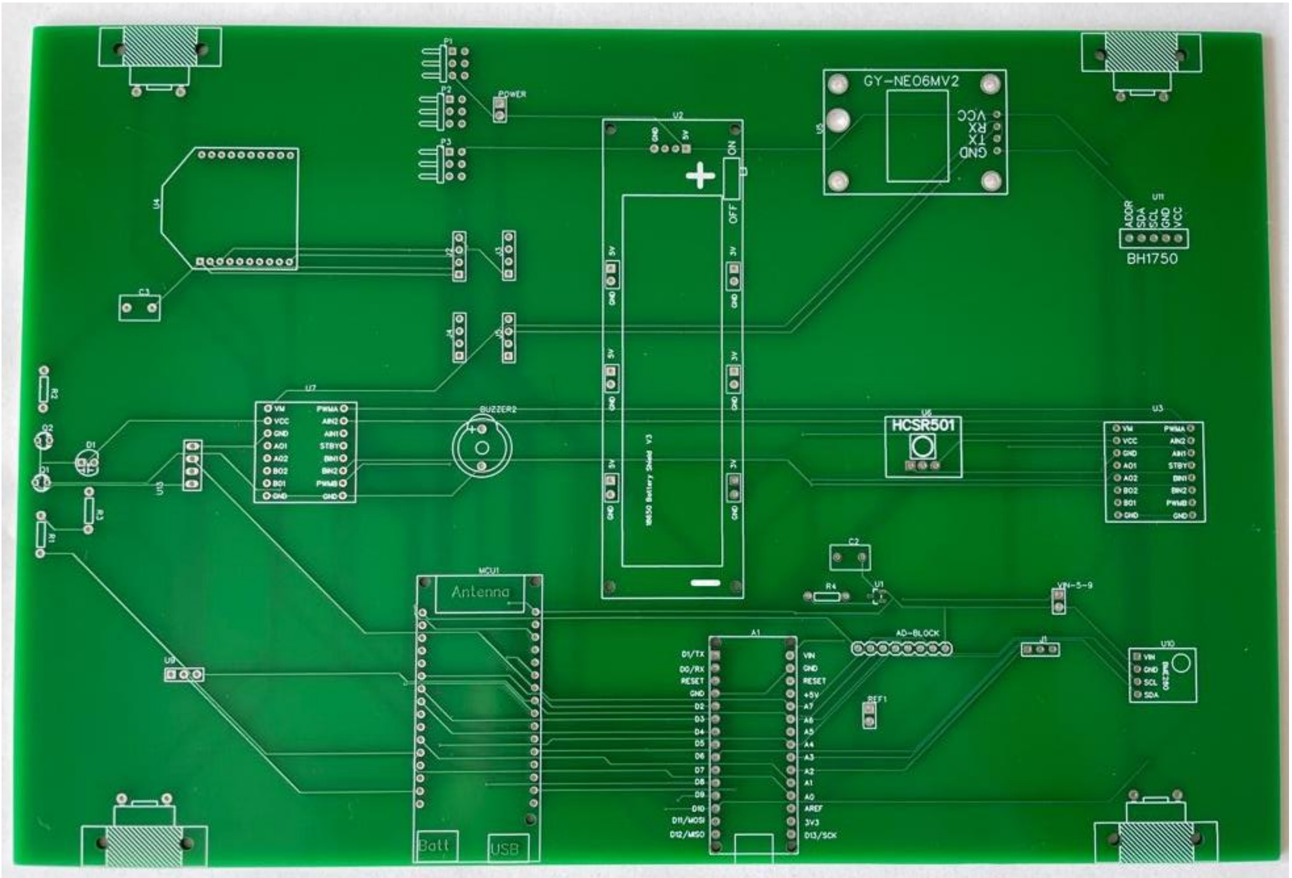


Figure 75. PCB top view.

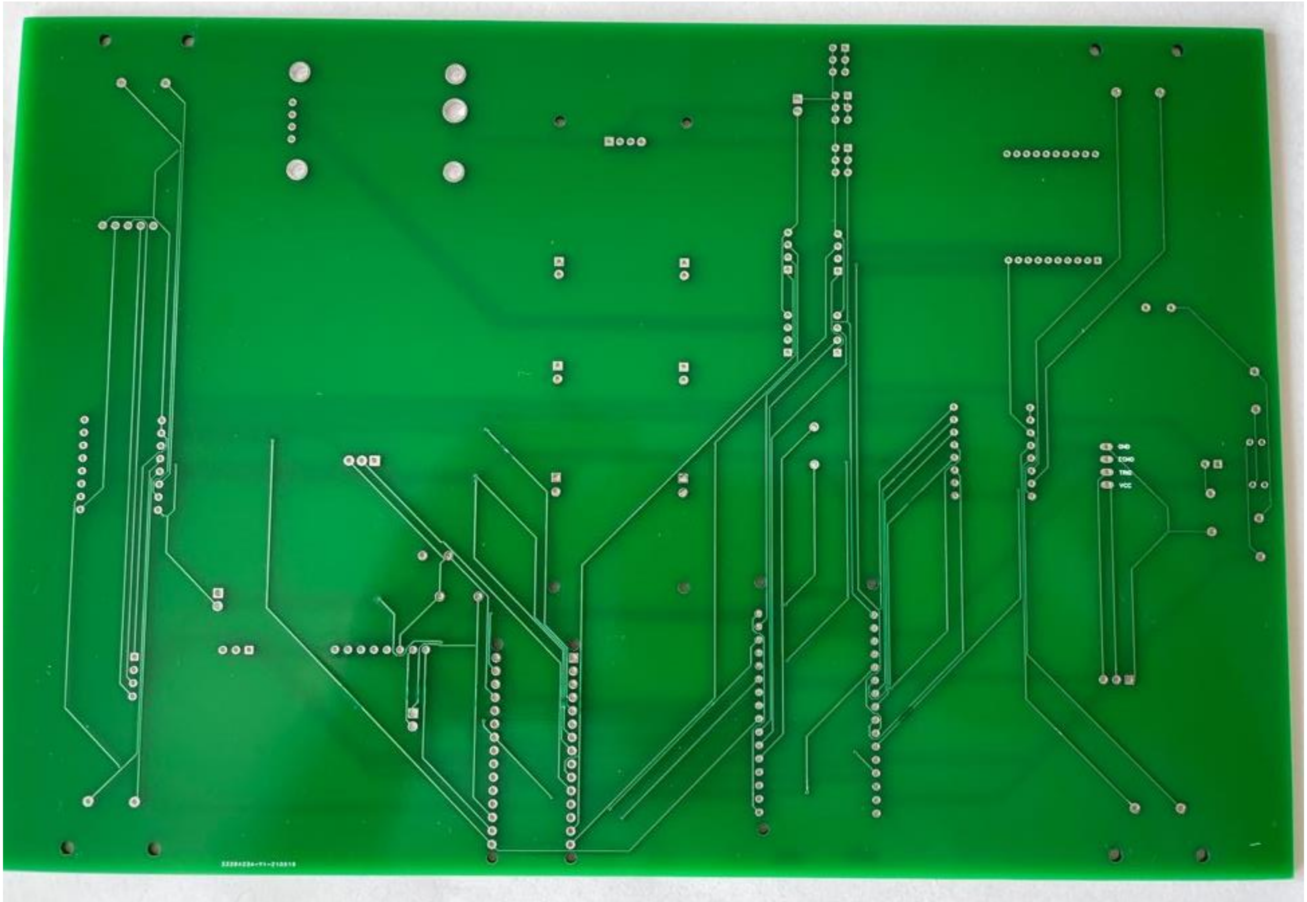


Figure 76. PCB bottom view.

Even though the PCB should be perfectly fine, it is a good practise to visually inspect all the tracks and pin holes in order to verify that everything is arranged as intended. Theoretically, one of the final steps in the production chain at JLCPCB is an electrical test, meaning that all connections should work properly; but, as we already know, many external factors could have messed with our PCB: the tracks could be damaged or not be thick enough to properly deliver the intended voltage to our components. Some of these external problems occurred during this project and are explained in this section.

Another very important factor is how we assemble the components. In our case, small components such as capacitors, resistors and the photodiodes are directly soldered to the PCB, while the rest of the components are placed in pin headers that are soldered to the intended positions. As we have already explained, this allows us to hot swap the most important components in order to adapt the available functionality, better storage or simply remove an already broken component that could potentially damage the robot.

We started the process by placing the photodiodes and the resistors needed for the line following functionality. We should consider that the diodes must be faced to the ground to achieve their expected properties. In the following images, we illustrate the step-by-step course of action. Firstly, place the components in their corresponding positions; then, solder them to the PCB and, finally, cut the loose ends with our pliers.

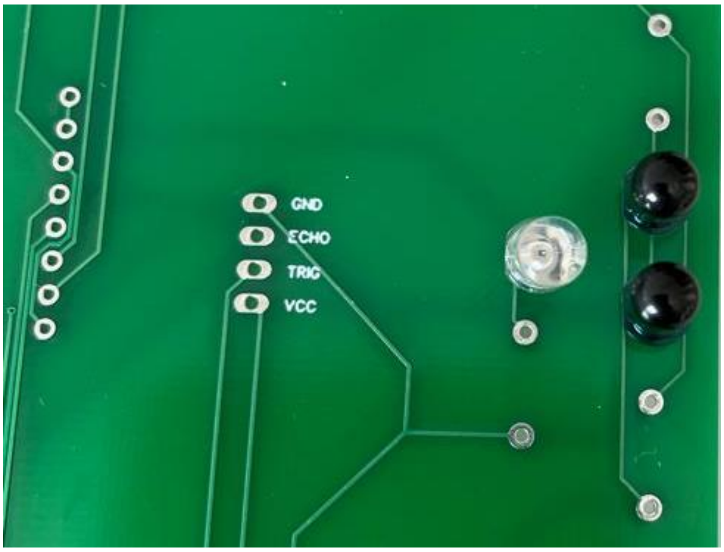


Figure 77. Photodiodes placed.

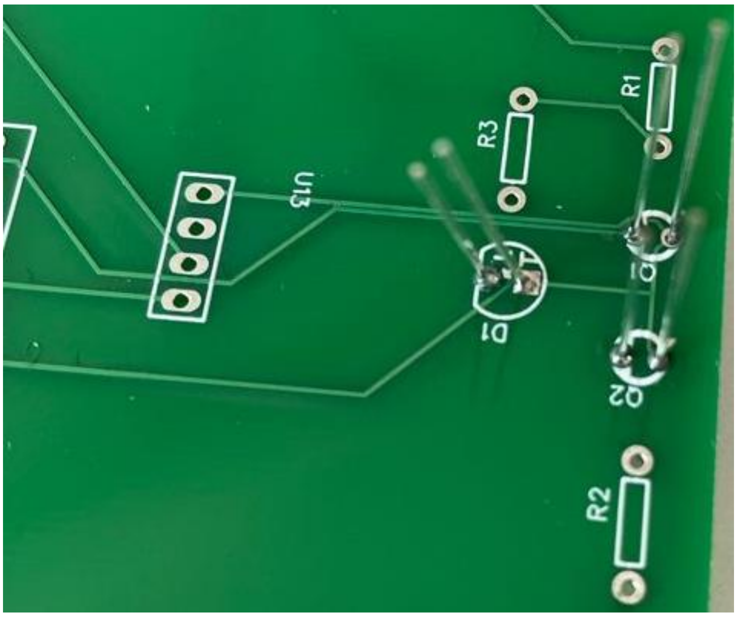


Figure 78. Soldered photodiodes.

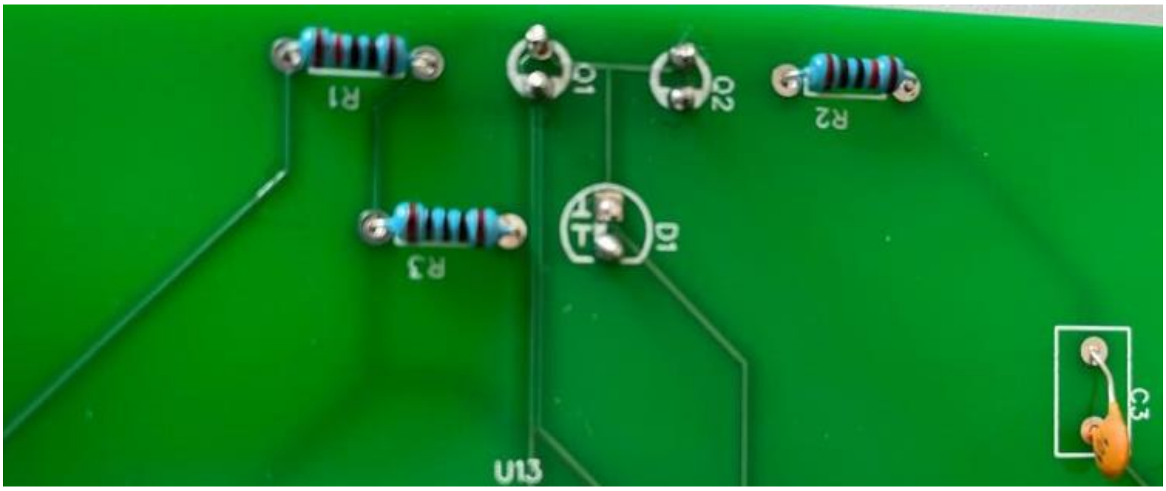


Figure 79. Completed soldering for the photodiodes circuit.

Once that we have one of the most difficult tasks done, we simply keep soldering the pin headers in a downwards movement from the photodiodes perspective.

This enables the righthanded to achieve a superior soldering position. If that is not your case, you may use a soldering path equal to your strong hand in order to obtain free space and movility while using the hot soldering tool.

To properly keep in place the headers while soldering, use a small but sufficient amount of tape in the opposite side of the PCB where you are going to solder.

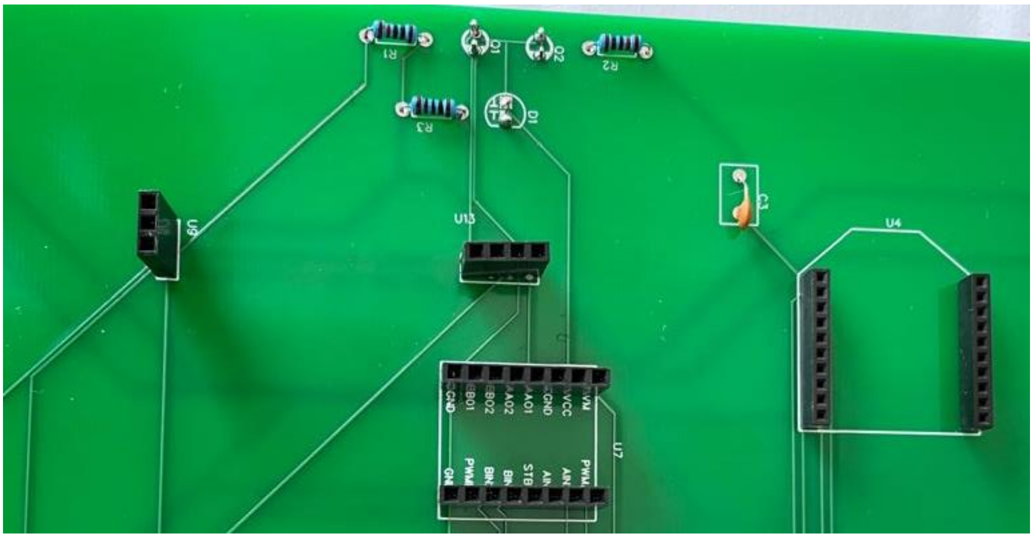


Figure 80. Female pin headers soldered in place.

At this point we encountered two problems: the buzzer and the level converters. Let us analyze the problems one by one.

The buzzer has only two pins, the pin that receives the control voltage and a ground pin; these pins do not use the standard 2.54mm spacing, so placing an individual pin header for each pin is quite difficult. Instead of cutting a pin strip and adapt it to our necessities, the buzzer is soldered directly to the PCB as it is not a very complex or critical module prone to failures, which may need a replacement.

The level converters were not found on the EasyEDA's component library as required, so we had to manually make up the spacing using headers and measuring by hand in the design process. Some corrections had to be done when placing the modules on the PCB. The real components were a bit wider than the PCB pin holes that were intended for them; to overcome this issue, simply place the headers in the pins of the module and then procede to adjust carefully the headers in order to solder them to the PCB.

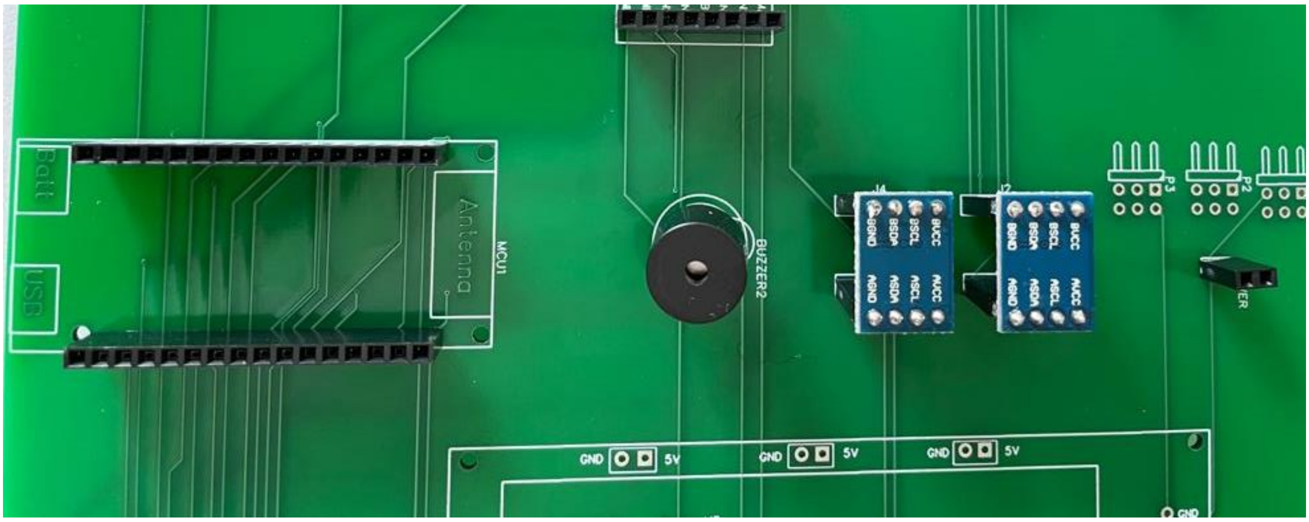


Figure 81. Soldered buzzer and adapted level converters.

We continued soldering headers, leaving the power unit for the final steps as this module only needs two wires soldered. The rest of the work consists of adjusting the elevation of the component, using screws as a guide rod and nuts.

This decision was taken because we had enough space available in our PCB to manoeuvre this part. In other designs, this possibility may not be feasible, so the default sequence of placements should be taken.

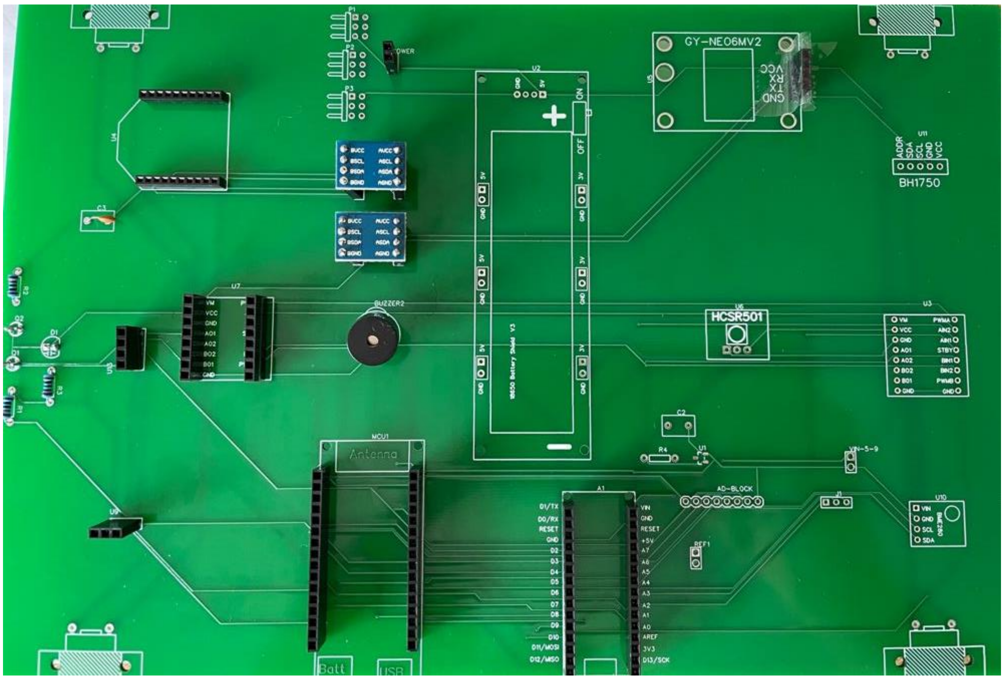


Figure 82. Soldered headers for microcontrollers.

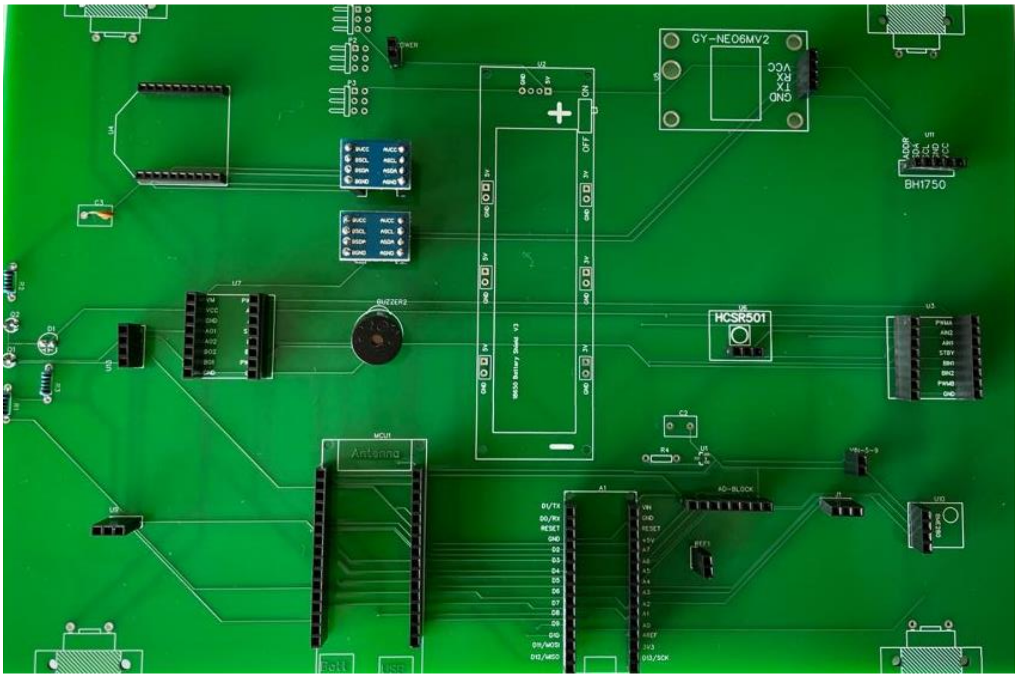


Figure 83. Soldered modules headers.

Once we had the headers in place, we centred our attention at the wheels. They are composed by the DC motor, the power wires, the plastic grip with bolts, nuts and of course the tyre itself.

Firstly, we attached the motor to the plastic grip and then screwed it to the PCB. You should obtain something like Figure 84. Fixed DC motor.:

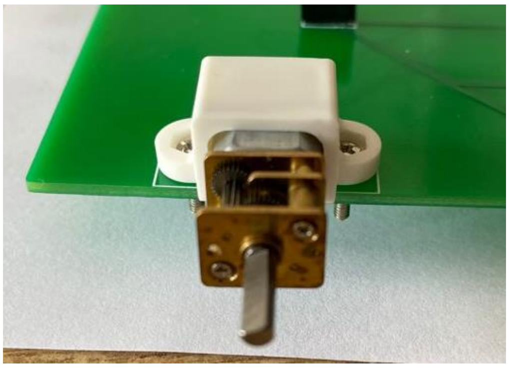


Figure 84. Fixed DC motor.

In our case, our wheels are large enough to obtain the sufficient ground clearance to properly function. If that is not the experienced situation, you may have various equally functional options: cut the rod of the screw and fine the sharp edges of the cut rod, use a screw with smaller length or use bigger wheels that could be attached to the DC motor.

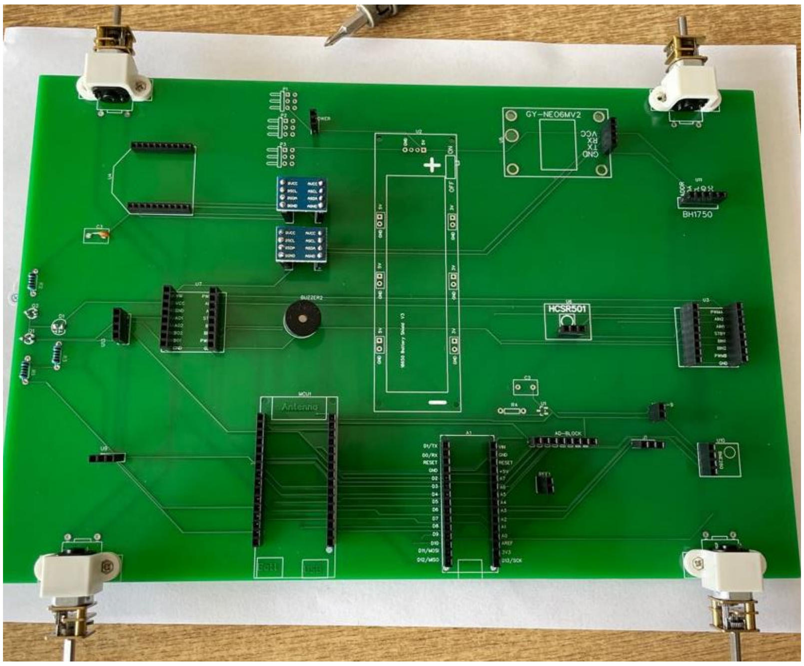


Figure 85. Traction system overview.

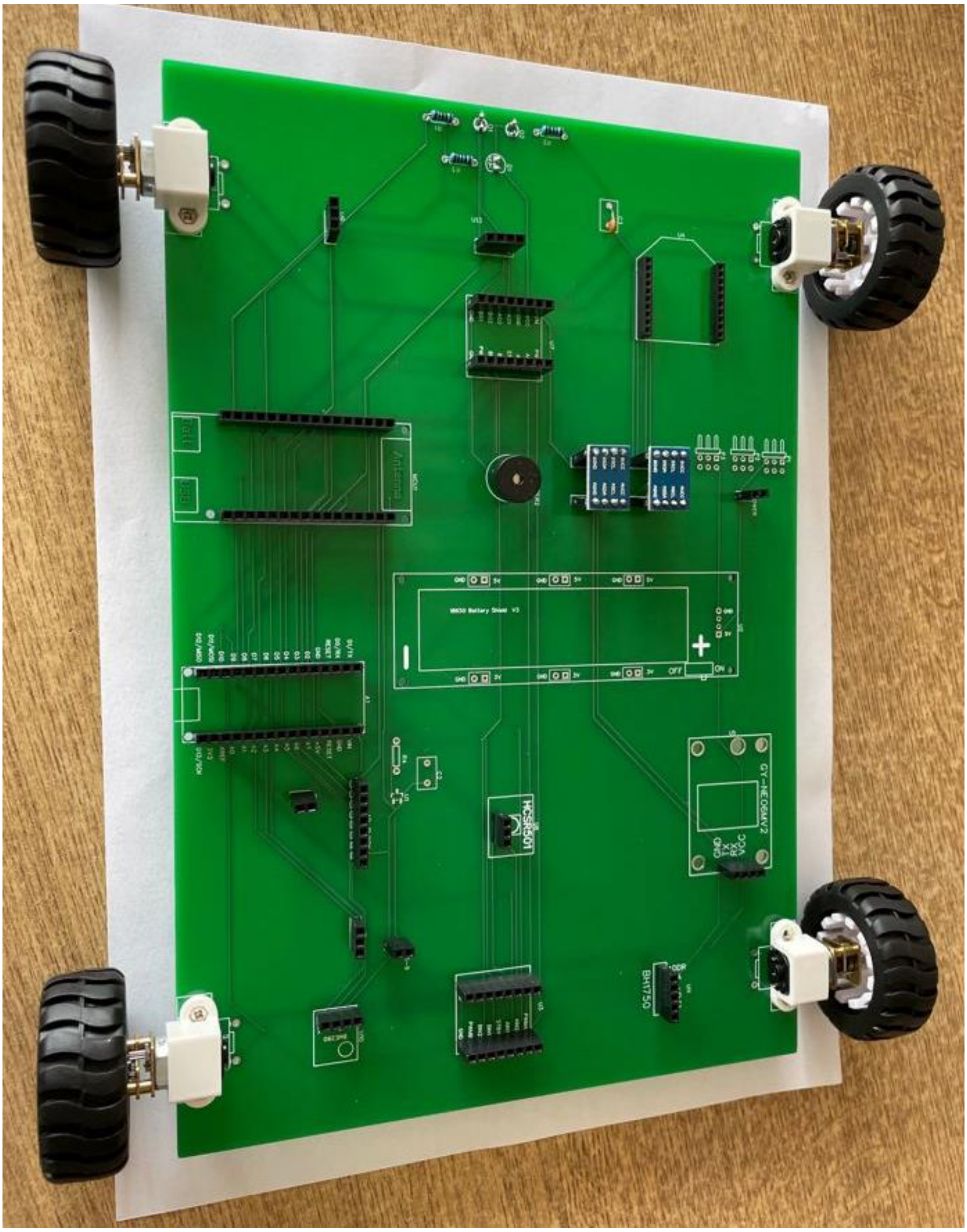


Figure 86. Wheels attached to motors.

At this point we just had to connect the motor pins to the PCB holes. In order to have the maximum space available, we should firstly use our pliers to attach one end of the wire to the desired pin. The process is simple: pass the straight wire through the pin hole, bend the wire in a 90º angle and then, using the pliers, roll the wire to obtain a result similar to the following Figure 87. DC motor connection process.:

Figure 87. DC motor connection process.

At this point, we had to solder the wires in the PCB holes and cut the excess of wire.

The final part for completing the prototype was placing the power unit. We passed its wires through the previously drilled holes of the PCB to the backside. We used three nuts for each screw, twelve in total: the first one, utilized to fix the crew to the PCB, and the other two, which elevate and keep in place the battery shield. Finally, the +5V and GND wires were soldered in place to the PCB.

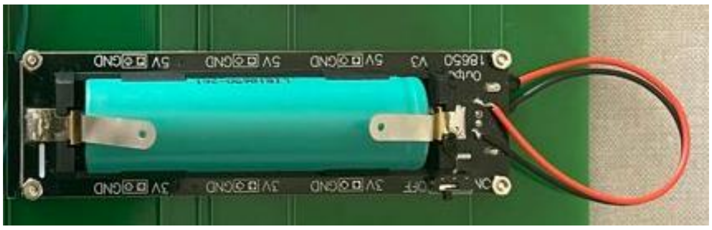


Figure 88. Fixed battery shield with battery.

Once everything was connected, we loaded a small script to our desired microcontroller and did a test run of the components.

In our case, the left front wheel did not move, so we had to start our electrical "debugging" phase. We took out the components from the robot and manually connected everything to a spare protoboard.

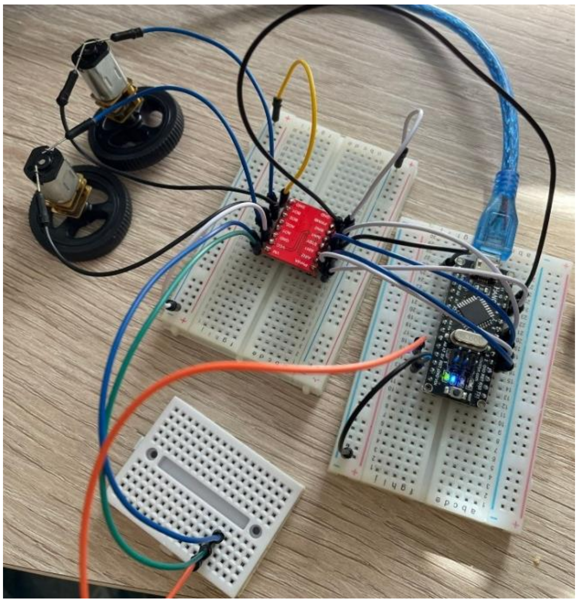


Figure 89. Motors and drivers test.

As no critical error occurred during this test, we used our multimeter and found out that the PCB track was damaged. There was not any visible damage to the PCB, but the problem persisted. As we designed a modular and redundant traction system, a straightforward fix was possible with no major concern. We merely connected the front left wheel to its rear counterpart, as they work in conjunction with the same signal.

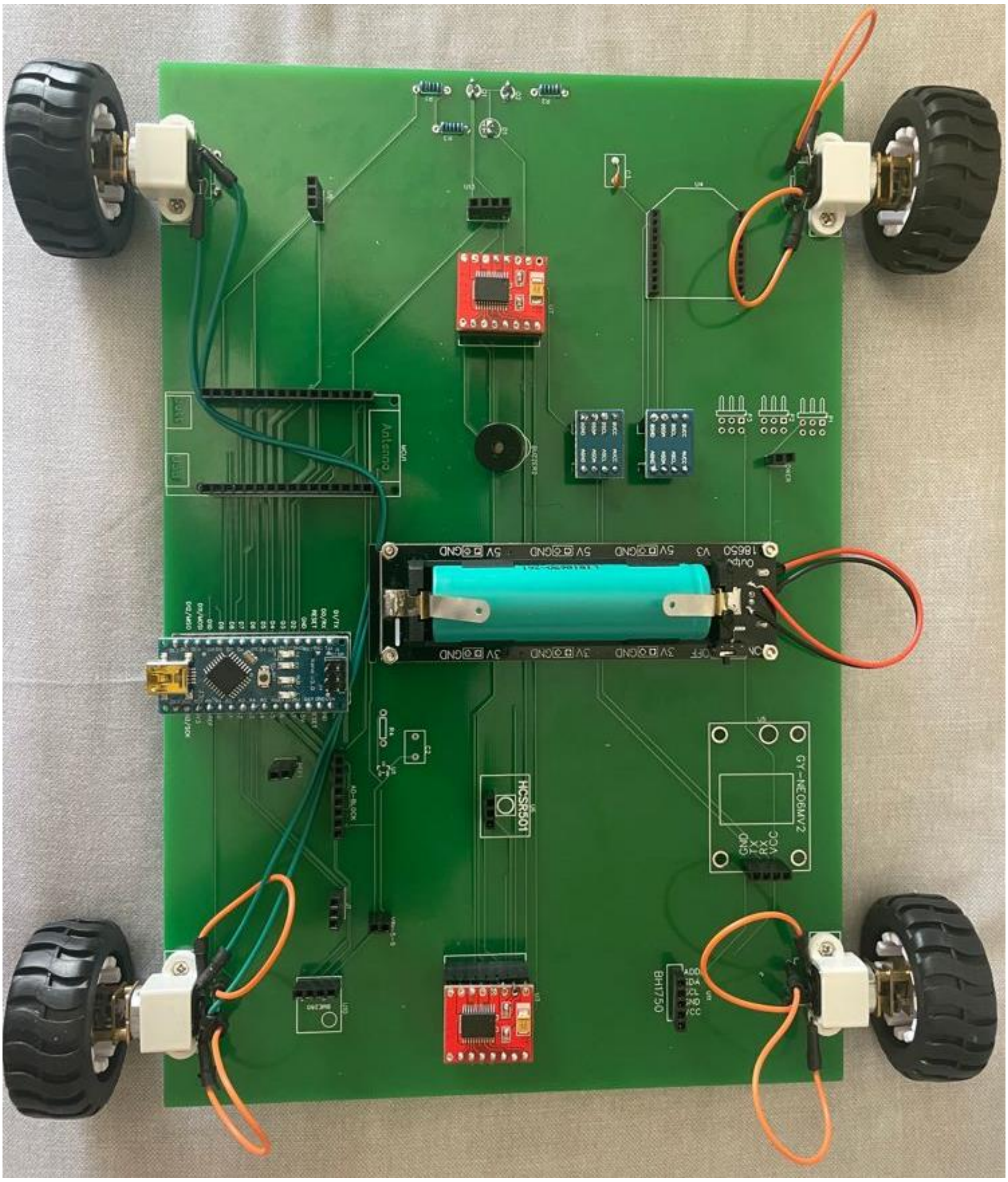


Figure 90. Assembled robot.

2.4 Design of the robot practices

In this section, we are going to suggest some laboratory sessions which aim to teach the capabilities of modern microcontrollers. Students will be able to learn all the skills required by the course by following this set of exercises and explanations.

2.4.1 Practice 1: Introduction

In this lab session, we are going to learn the fundamentals of our robot, its basic configuration and capabilities.

2.4.1.1 Objectives

- Describe the robot's hardware and controls.
- Describe how the DC motors are controlled (H-Bridge).
- Describe how the line following functionality is achieved.
- Describe the buzzer and how it works.
- Practical exercises.

2.4.1.2 Introduction to robot's hardware and controls

The basic configuration of the robot includes: chassis (PCB), microcontroller (Arduino Nano or ESP32), buzzer, two wheel drivers, two phototransistors alongside an infrared led, and a Li-Ion rechargeable battery. The robot could be also configured with various modules to alter its functionalities: GPS, barometric pressure sensor, light sensor, XBee communication module, movement sensor, robotic arm, and distance sensor.

2.4.1.3 Motor controls

In order to control the rotational direction of a DC motor, you must be able to change the polarity of the voltage applied. An H-Bridge allows us to change the polarity of a DC motor.

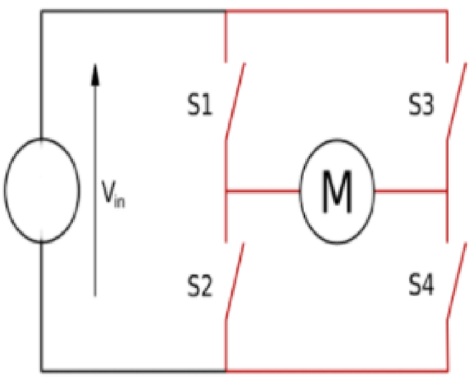


Figure 91. H-Bridge circuit.

The H-Bridge is composed by 4 switches(transistors): S1, S2, S3, S4. If S1 and S2 are closed, the DC motor will turn in one direction; if S3 and S4 are closed, the polarity reverses and the turn direction is changed. This H-Bridge configuration is usually integrated in a more complex component such as the TB6612FNG. Our driver can handle two DC motors (two H-Bridges).

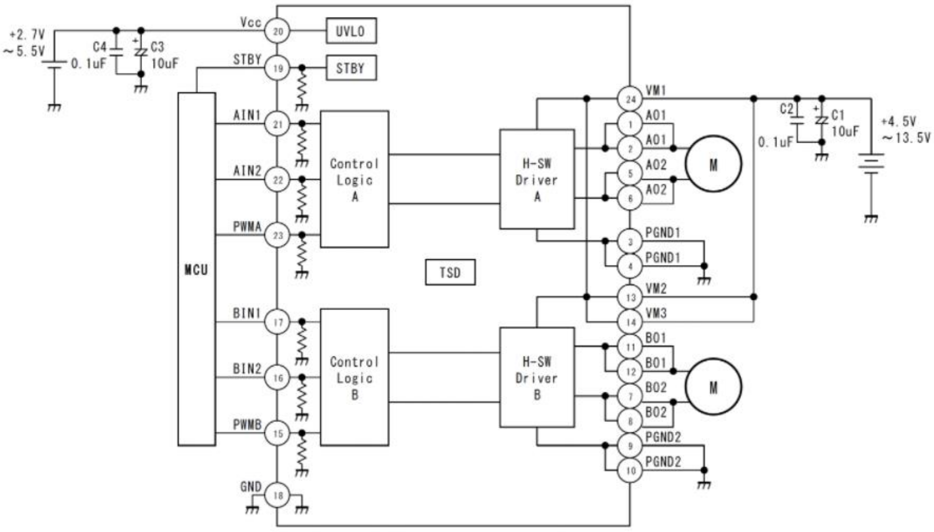


Figure 92. Wheel driver internal circuit.

In our robot we have two drivers. Theoretically, we could handle 4 wheels independently, but the configuration of the microcontrollers only allocates controls for one driver. The signals are the same for both drivers.

Table 9. Control signals of wheel driver. describes the control signals for the TB6612FNG driver:

Signal	Functionality
AIN1=1, AIN2=0	Right motor moves forward
AIN1=0, AIN2=1	Right motor moves backwards
AIN1=AIN2	Halts right motor
PWMA	Turning speed of right motor
BIN1=1, BIN2=0	Left motor moves forward
BIN1=0, BIN2=1	Left motor moves backwards
BIN1=BIN2	Halts left motor
PWMB	Turning speed of left motor
STBY=0	Enables motors
STBY=1	Disables motors

Table 9. Control signals of wheel driver.

Table 10. Microcontrollers and driver connections. describes how the microcontrollers and the drivers are connected:

Driver Pin	Arduino Nano Pin	ESP 32 Pin
AIN1	D2	32
AIN2	D3	33
BIN1	D4	25
BIN2	D5	26
PWMA	D9	22
PWMB	D10	21
STBY	D13	5

Table 10. Microcontrollers and driver connections.

Here is a code example using the Arduino Nano microcontroller:

```
const int AIN1 =  2; // control pins right motor
const int AIN2 =  3; //

const int BIN1 =  4; // control pins left motor
const int BIN2 =  5; //

const int PWMA =  9; // right motor speed control pin PWM
const int PWMB=  10; // left motor speed control pin PWM

const int STBY =  13; // enabling motors pin (Standby)
```

We can rotate the robot by simply providing different directions to each motor or by applying different turning speeds.

2.4.1.4 Track detection

It is based on an optical reflexion sensor. The circuit is composed by an infrared diode (D1) and two phototransistors (Q1 and Q2). The circuit is placed in the front end of the robot.

When a phototransistor is placed above a surface that reflects the light emitted by the infrared diode, the terminals of the phototransistor short-circuit and a logic "1" signal is sent to the corresponding pin of the microcontroller (left or right). If the phototransistor does not detect any light, the terminals are cut and, by doing so, a logic "0" signal is sent.

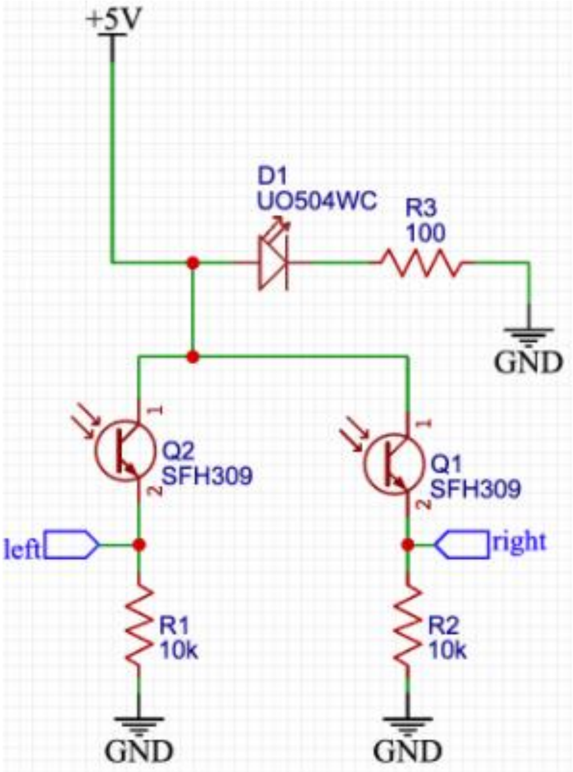


Figure 93. Track detection electrical circuit.

Table 11. Microcontrollers and track detection signals connections. Left and right signals for the microcontrollers:

	Arduino Nano Pin	ESP 32 Pin
Left	A0	14
Right	A1	12

Table 11. Microcontrollers and track detection signals connections.

Code example for the Arduino nano:

```
const int PhotoSensorLeft = A0; //  pin connected to left optical sensor
const int PhotoSensorRight = A1; // pin connected to right optical sensor

// constants
const int WHITE = 0;
const int BLACK = 1;
```

2.4.1.5 Buzzer

The buzzer allows the robot to emit sound. It has a control pin:

	Arduino Nano Pin	ESP 32 Pin
Buzzer	D7	4

Table 12. Microcontrollers and buzzer connections.

2.4.1.6 Exercises

- Write a small script that allows the robot to move forwards, backwards, turn left and turn right. The duration of each movement must be 3 seconds. Use the TB6612FNG driver to achieve this goal.
- Modify the previous script to gradually change the speed of the motors.
- Write a program that allows the robot to keep inside a black track. You should consider that every time one of the photoreceptors is situated outside the track, the robot must correct its direction to recover the expected path.
- Modify the previous program to make the robot capable of emitting a sound when it is outside the track (like the lane control system of some modern cars).

2.4.2 Practice 2: Collision detection

In this lab session, we are going to modify the robot and use the HC-SR04 and learn its fundamentals.

2.4.2.1 Objectives

- Learn the basics of the HC-SR04 distance sensor.
- Practical exercises.

2.4.2.2 Distance sensor

The HC-SR04 is a distance sensor based on ultrasounds. It can detect the distance of objects from the sensor in a range between 2 cm and 400 cm.

Figure 94. HC-SR04, distance sensor.

The sensor is composed by two special transductors: emitter and receptor. The emitter sends an ultrasound that will bounce back from the nearest object, returning to the receptor.

The Figure 95. Timing diagram of working distance sensor. shows the signals required by the sensor. The sensor will emit an ultrasound if it receives a 10 µs pulse in its “trigger” pin. At that moment, an 8-cycle burst of ultrasound (at 40 kHz) is sent while the “echo” output is placed at logic ª1ª. The logic one is held during the sound transmission and its return.

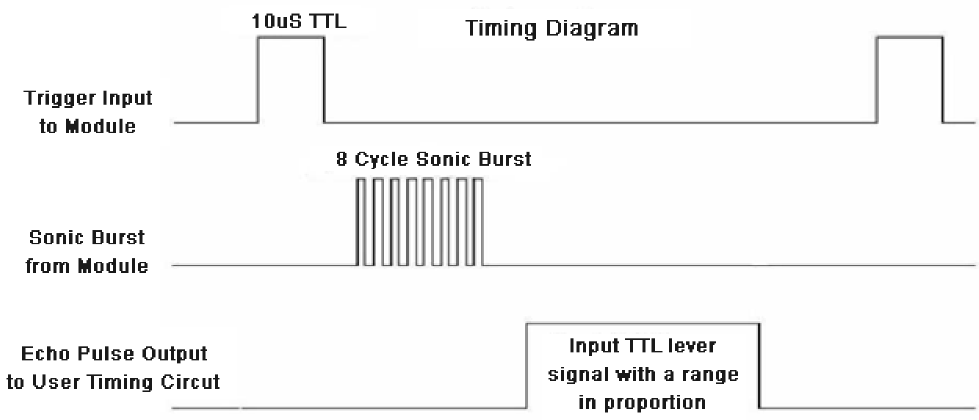


Figure 95. Timing diagram of working distance sensor.

Measuring the length of the “echo” signal at logical “1” and knowing the speed of sound (340 m/s or 0,034 cm/µs) we can determine the distance covered by the ultrasound wave. We will use the following formula (time measured in µs and distance in cm):

distance = sound_speed × time_spent_by_echo_at_“1” × 1/2 = 0,034 × time_spent_by_echo_at_“1” × 0,5

The result must be divided by two since we must consider the time spent arriving to the object and the returning time.

HC-SR04	Arduino Nano Pin	ESP 32 Pin
Trig	A6	18
Echo	A7	17

Table 13. Microcontrollers and distance sensor connections.

2.4.2.3 Exercises

- Modify the previous lab session code to consider possible obstacles in the track. You will need to add the new feature to the line-following source code and connect the module to the robot.

2.4.3 Practice 3: Remote control

In this lab session, we are going to adapt our robot so that it can be remotely controlled using a joystick. For this session we will also need: the remote-control shield, Arduino Nano, 2 Digi Xbee and the robotic arm.

2.4.3.1 Objectives

- Learning the basics of wireless serial communication using Xbee.
- Introduction to remote control.
- Practical exercises.

2.4.3.2 Introduction to Xbee

The Xbee modules of this session are already configured to work on transparent mode in order to stablish a point-to-point communication. In this mode, the Xbee modules connect through wireless communication. All data is sent by the serial input pin (Din pin). Data is transmitted immediately upon arrival to the Xbee which will receive this data and transmit it by its output pin (Dout pin). Both Xbees must be on the same network. If those are not configured, make use of XCTU and an USB adapter to manually configure both modules using a computer.

The goal is to create a remote controller using a special Arduino shield, a Xbee module and a joystick. The robot must have the receptor Xbee module installed. The Figure 96. Remote control and robot communication diagram. summarises this concept:

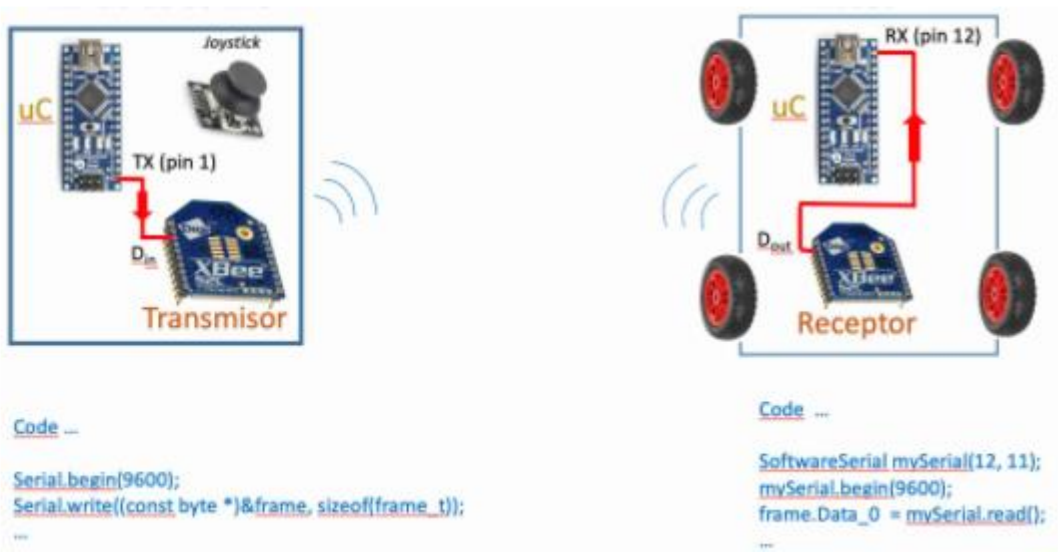


Figure 96. Remote control and robot communication diagram.

2.4.3.3 Remote control programming

The remote-control hardware consists of: Arduino shield, Arduino Nano, Xbee module and a joystick. The shield helps in the assembly of the components involved.

The wireless shield has a small switch labelled as "Serial". This switch allows us to upload code to the Arduino Nano (OFF position) and enable the serial communication with the Xbee (ON position).

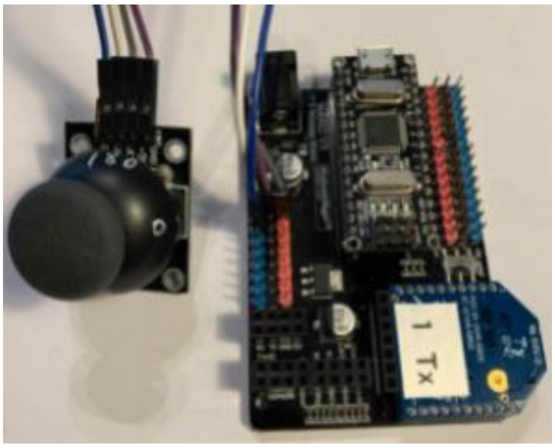


Figure 97. Remote control.

The Table 14. Remote control connections. recaps the connections between the shield and the joystick:

Wireless Shield	Joystick	Description
A0	VRx	Left - right turns
A1	VRy	Forward – backwards
GND	GND	Ground
Vcc	Vcc	+5V
2	Sw	Joystick button

Table 14. Remote control connections.

To upload any program and that involves the Xbee to the Arduino shield and check what is sending follow these steps:

- Put the switch in the ON position to upload the program to the microcontroller.
- Switch to OFF position.
- Open XCTU program in your computer.
- Enter the serial console: Tools -> Serial Console.
- Configure the baud rate in the "Configure" menu.
- Open the terminal using the "Open" option.

The following code snippet shows how to stablish the communication using the Xbee:

```
/*TX Transmitter  */
// frame format (Block of data that will be sent)
struct __attribute__((packed)) frame_t {
byte Data_eje_X;
byte Data_eje_Y;
// ... other data
};
frame_t frame;
void setup(){
// Configures Arduino's serial port to communicate with the Xbee
        Serial.begin(9600);
}
void loop(){
   // frame to transmit
   frame.Data_eje_X  = ;  // data sent (byte)
   frame.Data_eje_Y  = ;  // data sent (byte)
   // transmitting ...
   // Write the binary data to the serial port
   Serial.write((const byte *)&frame, sizeof(frame_t));
   Serial.flush();        //  Wait for data to be sent
   delay(25) ;
}
```

2.4.3.4 Robot control programming

The Xbee module of the robot is connected to the 12th pin in the case of the Arduino, 16th in the ESP32. The robot in this practice session will not send any data, only receiving data from the remote control. The microcontroller has a base RX0 default pin for serial communication; this hardware serial port will not be available, so we must declare a new port via software. We can manually codify this new port using the "SoftwareSerial.h" library.

The following code allows an Arduino Nano to receive data from an Xbee:

```
/* RX receiver example */
#include <SoftwareSerial.h>
// frame format
struct __attribute__((packed)) frame_t {
byte Data_0;
byte Data_1;
// ... other data
};
frame_t frame;
SoftwareSerial mySerial(12, 11);// serial reception uses Arduino Nano
pin 12
// pin RX-mySerial (12) is connected to Xbee pin DOUT  (2)
// pin TX-mySerial (11) is connected to Xbee pin DIN   (3)not mandtory
void setup() {
        mySerial.begin(9600);  //initialize software serial port
        Serial.begin(9600);    //initialize UART for debugging
}
void loop(){
if (mySerial.available() > 1){ // check if there is data in the buffer
        // read the incoming bytes:
        frame.Data_0  = mySerial.read(); // frame read
        frame.Data_1 = mySerial.read();
        // debug on console
        Serial.write(frame.Data_0);
        Serial.write(frame.Data_1);
    }
}
```

2.4.3.5 Exercises

- Extend the code for the wheel test from *Practice 1: Introduction* session. Now the robot must move using the information provided by the joystick of the remote control.
- Connect the robotic arm and create a program that allows for arm rotation using the remote control.

2.4.4 Practice 4: GPS

In this lab session, we are going to learn about the NEO6MV2 GPS module. We will require a protoboard as well as the robot itself.

2.4.4.1 Objectives

- Learn the basics of the NEO6MV2 module.
- Practical exercises.

2.4.4.2 GPS module

The NEO6MV2 it is a serial GPS - Global Positioning System - module. This module is able to locate itself on Earth by measuring the distance to various satellites (triangulation). It has an external ceramic antenna which is responsible of the communication with the satellites. In order to properly operate this module, it must be powered externally using +5V; this means that we cannot use the microcontroller to power it. The robot has the necessary features to adequately power this module. Another consideration is that the module takes some time to obtain the first bunch of data from the satellites, since it may take at least 5 or 10 minutes before the data received is complete.

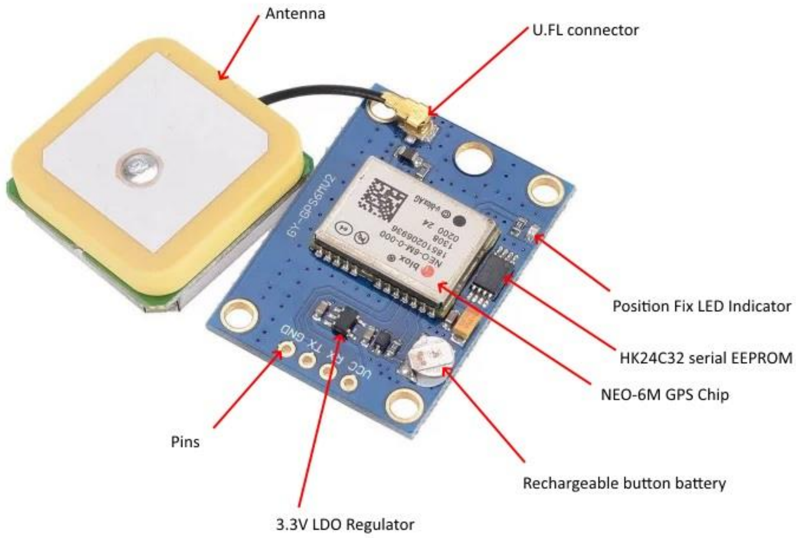


Figure 98. GPS module main parts.

To write programs that use the information obtained by this module we will require to:

- Create our own serial port.
- Use “TinyGPS.h” library.

The following code allows to display the information obtained in the serial console:

```
#include <SoftwareSerial.h> // Own software serial port
#include <TinyGPS.h> // Library to handle the data provided
TinyGPS gps;
SoftwareSerial serialgps(11,12);
char data= ' '; // Read from module
void setup() {
  Serial.begin(115200); // Microcontroller serial port
  serialgps.begin(9600); // Serial module port
}
void loop() {
  if(serialgps.available()){  // Checks data available
    data = serialgps.read(); // Obtains the data
    Serial.print(data); // Display data to the serial console
    if(gps.encode(data)){ // Library interprets data
      // Calls  tiny GPS library
    }
    delay(100);
  }
}
```

2.4.4.3 Exercises

- Using the module outside the robot, write a small program that prints in the serial console: latitude, longitude, date, altitude, speed, and number of satellites.
- Using the robot, write a small program that orders the robot to move to a specific geographical location.

2.4.5 Practice 5: Light control

In this lab session, we are going to control our robot using the BH1750 light sensor and a flashlight -or any light source-.

2.4.5.1 Objectives

- Learn the fundamentals of the BH1750 light sensor.
- Practical exercises.

2.4.5.2 Light sensor

The BH1750 has a light meter which can determine lux. Lux is a SI derived unit which measures the luminous flux per unit of area. For example, a dark night has a total of 0.001-0.02 lux whereas a clear sunny day has 100000 lx. This sensor is a I2C module, which means it can work simultaneously alongside other sensors using the same pins. To utilize this module, we will require the "BH1750.h" library.

A I2C module has two main pins: SDA (Serial Data Line) where the data is transferred, and SCL (Serial Clock Line) which controls the transmission bit by bit of the SDA. Some modules have an extra pin ADDR which controls the reference of the module. This I2C (Inter-Integrated Circuit) allows for multiple modules to use the same serial communication bus.

Usually, the libraries handle the address of the modules in the I2C bus, so you do not have to consider such details while programming the master unit (microcontroller).

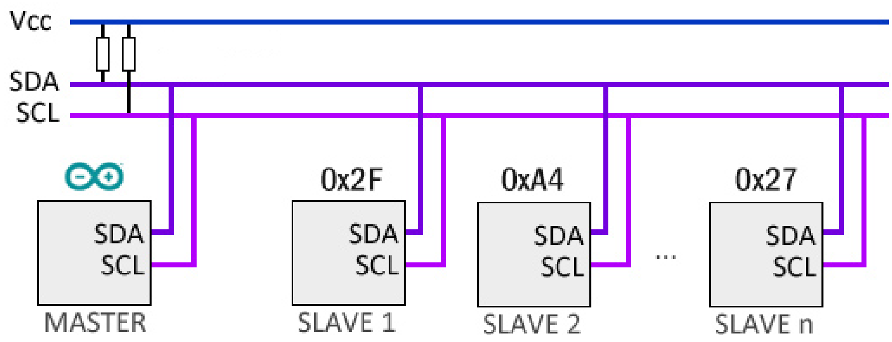


Figure 99. Master/slave I2C communication.

Table 15. Microcontrollers and GPS module connections. describes the pin-out of the module for the Arduino and the ESP32:

BH1750	Arduino Nano Pin	ESP 32 Pin
SDA	A4	13
SCL	A5	15

Table 15. Microcontrollers and GPS module connections.

The following code prepares the basics of this module and library:

```
#include <BH1750.h> // Library for the module, 0x_23 address by default
BH1750 luxometer; // luxometer object
// Resolution modes of the sensor
// BH1750::CONTINUOUS_HIGH_RES_MODE
// BH1750::CONTINUOUS_HIGH_RES_MODE_2
// BH1750::CONTINUOUS_LOW_RES_MODE
// BH1750::ONE_TIME_HIGH_RES_MODE
// BH1750::ONE_TIME_HIGH_RES_MODE_2
// BH1750::ONE_TIME_LOW_RES_MODE
void setup() {
   Serial.begin(9600);
   Serial.println("Initializing sensor...");
   luxometer.begin(BH1750::CONTINUOUS_HIGH_RES_MODE); // Initializing
}
void loop() {
   uint16_t lux = luxometer.readLightLevel(); // Read of BH1750
   Serial.print("Light level:  ");
   Serial.print(lux);
   Serial.println(" lx");
   delay(500);
}
```

2.4.5.3 Exercises

- Write a small program that makes the robot move to a brighter position in the room. You may use a flashlight to control the movement.
- Write a small program that makes the robot move to a darker position. Take into consideration table legs to avoid potential damage to the robot.

2.5 Effort measurement and estimation

In this annexe, we are going to show the complete WBS (Work Breakdown Structure) where the elemental tasks that were done to achieve the final solution will be enumerated, prioritized, and scheduled.

Table 16. Work Breakdown Structure of the project. will be considered for calculating the final budget described in *Budget* section.

Index	Name of task	Time spent	Start date	End date
1	**Project**	**1576 hours**	**15/01/21**	**05/04/21**
1.1	**Design of the robot**	**19 hours**	**15/01/21**	**19/01/21**
1.1.1	Analysis of legacy robot	5 hours	15/01/21	15/01/21
1.1.1.1	Components	1 hour	15/01/21	15/01/21
1.1.1.2	Electrical Schema	3 hours	15/01/21	15/01/21
1.1.1.3	PCB Schema	1 hour	15/01/21	15/01/21
1.1.2	EasyEDA familiarization	3 hours	15/01/21	15/01/21
1.1.3	Component selection	3 hours	18/01/21	18/01/21
1.1.4	New robot design	8 hours	18/01/21	19/01/21
1.1.4.1	Electrical Schema	6 hours	18/01/21	19/01/21
1.1.4.2	PCB Schema	2 hours	19/01/21	19/01/21
1.2	**Hardware acquisition**	**1528 hours**	**19/01/21**	**29/03/21**
1.2.1	Modules	1455 hours	19/01/21	25/03/21
1.2.1.1	Buying modules	3 hours	19/01/21	19/01/21
1.2.1.2	Waiting for modules	1440 hours	20/01/21	22/03/21
1.2.1.3	Modules testing	12 hours	23/03/21	25/03/21
1.2.2	PCB	73 hours	25/03/21	29/03/21
1.2.2.1	Ordering PCB	1 hour	25/03/21	25/03/21
1.2.2.2	Waiting for PCB	72 hours	25/03/21	29/03/21
1.3	**Robot Assembly**	**14 hours**	**30/03/21**	**31/03/21**
1.4	**Lab session elaboration**	**15 hours**	**03/04/21**	**05/04/21**
1.4.1	Introduction	3 hours	03/04/21	03/04/21
1.4.2	Collision detection	2 hours	03/04/21	03/04/21
1.4.3	Remote control	5 hours	03/04/21	05/04/21
1.4.4	GPS	3 hours	05/04/21	05/04/21
1.4.5	Light Control	2 hours	05/04/21	05/04/21

Table 16. Work Breakdown Structure of the project.

3. Requirements

In this section, we are going to use an adaptation of the ISO/IEC/IEEE 29148:2011 to specify the system requirements. In this case, we will only take into consideration the items that apply to the project; criteria such as logical requirements for database systems are outside of the scope of this project and will be omitted.

3.1 Functional requirements

Functional requirements describe the functionality constraints of the project.

FR 1. The robot must be capable of performing movement.
- **FR 1.1.** The robot must be capable of moving forward.
- **FR 1.2.** The robot must be capable of moving backwards.
- **FR 1.3.** The robot must be capable of turning clockwise.
- **FR 1.4.** The robot must be capable of turning counterclockwise.

FR 2. The robot must be capable of following line paths.
- **FR 2.1.** If the left photodiode detects an off-track reading, it must alter its route to the right.
- **FR 2.2.** If the right photodiode detects an off-track reading, it must alter its route to the left.
- **FR 2.3.** If both photodiodes detect an in-track reading, it must maintain its forward course.
- **FR 2.4.** If both photodiodes detect an off-track reading, it must:
  - FR 2.4.1. Halt its current movement.
  - FR 2.4.2. Initiate a retreat to a position between track boundaries.

FR 3. The robot must be capable of being controlled remotely.

FR 4. The robot must be capable of having obstacle detection.

FR 5. The robot must be capable of communicating via WIFI.

FR 6. The robot must be capable of measuring the following:
- **FR 6.1.** Forward distance.
- **FR 6.2.** Movement.
- **FR 6.3.** Temperature.
- **FR 6.4.** Pressure.
- **FR 6.5.** Humidity.
- **FR 6.6.** Luminosity.

FR 7. The robot must be capable of being geolocalized.

FR 8. The robot must be capable of emitting sound.

FR 9. The power supply must be controlled by a switch.

FR 10. The microcontroller must be capable of handling a robotic arm.
- **FR 10.1.** The robotic arm must be capable of rotating in the horizontal axis.
- **FR 10.2.** The robotic arm must be capable of rotating in the vertical axis.

3.2 Non-functional requirements

The non-functional requirements describe project constraints such as reliability, security, or usability. Not to mention, they also define properties or general attributes in addition to establishing restrictions of the final product or the development process and describing the system's operative circumstances such as portability or compatibility.

3.2.1 Documentative requirements

NFDR 1. The student guide must provide the fundamentals and basics of the listed modules.
NFDR 2. The student guide must follow the style of the one currently in use by the course.
NFDR 3. The student guide must provide practical exercises for each session.
NFDR 4. The student guide must define the following sessions:
- **NFDR 4.1.** Introduction to the hardware.
- **NFDR 4.2.** Collision detection.
- **NFDR 4.3.** Remote control.
- **NFDR 4.4.** GPS.
- **NFDR 4.5.** Light control.

3.2.2 Technological requirements

NFTR 1. The robot will be powered by an 18650 battery.
NFTR 2. The PCB must:
- **NFTR 2.1.** Be the chassis of the robot.
- **NFTR 2.2.** Have a contrast between the base colour and its tracks.
- **NFTR 2.3.** Be prepared for the microcontrollers:
  - **NFTR 2.3.1.** Arduino Nano.
  - **NFTR 2.3.2.** ESP 32 LOLIN.
- **NFTR 2.4.** Connect the microcontrollers with:
  - **NFTR 2.4.1.** 18650 Battery shield.
    - **NFTR 2.4.1.1.** 4 separators.
    - **NFTR 2.4.1.2.** 4 nuts.
    - **NFTR 2.4.1.3.** 1 18650-battery.
  - **NFTR 2.4.2.** HC-SR501, infrared movement module.
  - **NFTR 2.4.3.** HC-SR04, distance sensor.
  - **NFTR 2.4.4.** BME280, barometric pressure sensor.
  - **NFTR 2.4.5.** BH1750, light intensity sensor.
  - **NFTR 2.4.6.** NEO6MV2, GPS module.
    - **NFTR 2.4.6.1.** 4 screws.
    - **NFTR 2.4.6.2.** 4 nuts.
  - **NFTR 2.4.7.** Buzzer.

NFTR 2.4.8. XBEE, serial communication receiver.

NFTR 2.4.8.1. The XBEE requires a capacitor connected in parallel to the VCC to reduce noise.

NFTR 2.4.9. The receiver must transfer the information sent by the remote controller to the microcontroller.

NFTR 2.4.10. Robotic arm.

NFTR 2.4.10.1. The robotic arm movement is guaranteed by a SG90 DC motor.

NFTR 2.4.10.2. The robotic arm must be fixed to the PCB.

NFTR 2.4.11. Power train containing:

NFTR 2.4.11.1. 4 DC motors.

NFTR 2.4.11.2. 1 wheel attached to each motor, 4 in total.

NFTR 2.4.11.3. 1 motor fixing for each motor, 4 in total.

NFTR 2.4.11.4. 2 screws for each fixing, 8 in total.

NFTR 2.4.11.5. 2 nuts for each screw, 8 in total.

NFTR 2.4.11.6. 1 motor driver for controlling each motor, 2 in total.

NFTR 2.4.12. Level converter between XBEE and Arduino Nano.

NFTR 2.4.13. Level converter between ESP32 and 5V signals from modules.

NFTR 2.4.14. Photodiodes circuit that contains:

NFTR 2.4.14.1. 1 left photodiode receiver.

NFTR 2.4.14.2. 1 right photodiode receiver.

NFTR 2.4.14.3. 1 centre photodiode emitter.

NFTR 2.4.14.4. 1 10kΩ resistor connected in series after the receiver, 2 in total.

NFTR 2.4.14.5. 1 100 Ω resistor connected in series after the emitter.

NFTR 2.4.15. There must be available connectors for:

NFTR 2.4.15.1. Free pins in the Arduino Nano.

NFTR 2.4.15.2. Battery shield ground.

NFTR 2.4.15.3. Battery shield 5V.

NFTR 2.4.16. The 3V supply is obtained from:

NFTR 2.4.16.1. Arduino Nano 3V pin.

NFTR 2.4.16.2. ESP32 3V pin.

NFTR 2.4.17. The Arduino Nano is powered by the 5V pin.

NFTR 2.4.18. The ESP32 is powered by the USB/5V pin.

NFTR 3. The robot will provide open connections for other desired modules.

4. Budget

In this section, we are going to analyse the budget of the project. We can differentiate two parts: the cost budget and the client budget.

4.1 Cost budget

Role	Units	Salary/hour(€)	Taxes	Cost per hour
Software Eng.	1	10,5	29.9%	13.64€

Table 17. Worker roles and costs.

Index	Name of task	Quantity	Unit	Cost	Subtotal(3)	Subtotal(2)	Subtotal(1)
1	**Design of the robot**	**32**	**Hours**				**259.16€**
1.1	Analysis of legacy robot	5	Hours			68.20€	
1.1.1	Components	1	Hour	13.64€	13.64€		
1.1.2	Electrical Schema	3	Hours	13.64€	40.92€		
1.1.3	PCB Schema	1	Hour	13.64€	13.64€		
1.2	EasyEDA familiarization	3	Hours	13.64€		40.92€	
1.3	Component selection	3	Hours	13.64€		40.92€	
1.4	New robot design	8	Hours	13.64€		109.12€	
1.4.1	Electrical Schema	6	Hours	13.64€	81.84€		
1.4.2	PCB Schema	2	Hours	13.64€	27.28€		
2	**Hardware acquisition**						**566.38€**
2.1	Acquisition	16	Hours			218.24€	
2.1.1	Acquisition of modules	3	Hours	13.64€	40.92€		
2.1.2	Modules testing	12	Hours	13.64€	163.68€		
2.1.1	Ordering PCB	1	Hour	13.64€	13.64€		
2.2	Acquired hardware					348.14€	
2.2.1	Nano shield	1	Unit	4.48€	4.48€		
2.2.2	SG90	2	Units	3.22€	6.44€		
2.2.3	Photodiode	3	Packs	2.16€	6.50€		
2.2.4	Female pin header	9	Packs	4.31€	38.82€		
2.2.5	Xbee	2	Units	40.35€	80.70€		
2.2.6	Battery shield	1	Unit	4.40€	4.40€		
2.2.7	Joystick	1	Unit	0.50€	0.50€		

2.2.8	TB6612FNG	2	Units	1.19€	2.38€		
2.2.9	Buzzer	1	Pack	1.11€	1.11€		
2.2.10	HC-SR04	1	Pack	22.54€	22.54€		
2.2.11	BME280	1	Unit	3.10€	3.10€		
2.2.12	DC Motor	4	Units	5.40€	21.60€		
2.2.13	Wheel	4	Units	1.95€	7.80€		
2.2.14	Arduino Nano	2	Units	7.40€	14.80€		
2.2.15	ESP32	1	Unit	6.99€	6.99€		
2.2.16	Level converter	2	Units	1.49€	2.98€		
2.2.17	GPS module	1	Unit	37.30€	37.30€		
2.2.18	Light sensor	1	Pack	13.68€	13.68€		
2.2.19	Movement sensor	1	Pack	12.67€	12.67€		
2.2.20	Battery	1	Unit	8.00€	8.00€		
2.2.21	PCB	5	Units	10.27€	51.35€		
3	**Robot Assembly**	**14**	**Hours**				**190.96€**
4	**Lab session elaboration**	**15**	**Hours**				**204.60€**
4.1	Introduction	3	Hours	13.64€		40.92€	
4.2	Collision detection	2	Hours	13.64€		27.28€	
4.3	Remote control	5	Hours	13.64€		68.20€	
4.4	GPS	3	Hours	13.64€		40.92€	
4.5	Light Control	2	Hours	13.64€		27.28€	
TOTAL							**1221.10€**

Table 18. Project direct costs.

In order to calculate the indirect cost, we will require the following auxiliary tables (Table 19. Amortizations.,Table 20. Electrical consumption.):

Product	Total cost(€)	Amortization period	Unit	Monthly cost
Custom built PC	1200€	4	Years	25.00€
Soldering Iron	20€	4	Years	0.42€
Soldering Stand	20€	4	Years	0.42€
Multimeter	80€	10	Years	0.66€

Table 19. Amortizations.

There will be no “other cost” calculations because of the lack of presential work. Due to the current circumstances, all work has been done by telematic means.

Tool	Hours	Power consumption (W)	Price KwH	KwH/month	Cost
Custom built PC	63	65	0.24€	15.60	3.74€
Soldering Iron	14	60	0.24€	14.40	3.46€
				Monthly base electrical fee	20.00€
TOTAL				**30**	**27,20€**

Table 20. Electrical consumption.

L1	L2	L3	Description	Qty.	Unit	Monthly Cost	Subtotal-3	Subtotal-2	Subtotal-1
1			**Project**						**101.50€**
	1		Hardware					101.50€	
		1	Custom built PC	1	Unit	25.00€	100.00€		
		2	Soldering Iron	1	Unit	0.42€	0.42€		
		3	Soldering Stand	1	Unit	0.42€	0.42€		
		4	Multimeter	1	Unit	0.66€	0.66€		
2			**Company**						**228.80€**
	1		Electricity	4	Months	27.20€		108.80€	
	2		Internet	4	Months	30.00€		120.00€	
TOTAL									**330.30€**

Table 21. Indirect costs.

Code	Item	Total
1	Design of the robot	259.61€
2	Hardware acquisition	566.38€
3	Robot assembly	190.96€
4	Lab session elaboration	204.60€
5	Indirect costs	330.30€
TOTAL COST		**1551.85€**

Table 22. Costs summary.

4.1.1 Client budget calculation

Profit margin	25%
Project total cost	1551.85€
Profit	387.93€
To compensate	718.27€
Total of items suitable for applying compensation	872.96€
Increase rate	82.28%
Total with profit included	**1949.20€**

Table 23. Client budget calculation.

4.2 Client budget

Code	Item	Price	Total
1	Design of a robot for teaching microcontrollers		1949.20€
1.1	Design of the robot	473.22€	
1.2	Hardware acquisition	754.95€	
1.3	Robot assembly	348.08€	
1.4	Lab session elaboration	372.95€	
TOTAL COST			**1949.20€**

Table 24. Client budget.

5. Conclusions

We have designed and built a functional piece of hardware. The robot satisfies the goals and requirements proposed. The blueprints and instructions provided in this document allows a person with little knowledge in electronics and programming to develop a similar hardware solution. The modularity of this robot tolerates rapid fixes, as it also uses female headers for the components. The knowledge provided makes relatively easy the conversion of the diagrams to other types of "robot vehicles" such as a small drone.

The final hardware could be used in future courses of Computer Electronics. The suggested laboratory practice sessions which have been provided here offer a basic-mid level experience similar to that of the current sessions in the course. Nevertheless, the designed hardware allows for much more complex or advanced functionalities that may be put in use, such as the Wi-Fi capabilities. For example, a discarded practice suggestion was using the ESP32 to read some sensor -such as the GPS module- and send the data to a local network to consumed by a web application.

This product is not only a multipurpose teaching tool. Some improvements were proposed in this document that might lead to future thesis or future laboratory sessions that use this robot as a base hardware. An interesting proposal could be using a smartphone as a remote controller while utilizing a HC-06 Bluetooth module in the robot which receives the signal from the smartphone.

EasyEDA has demonstrated itself as an outstanding development tool for electronical circuits and PCBs. The platform also allows for pre-built modules that even though they may increase the final price, they may help a developer with little knowledge or skill in soldering to achieve a professional finish.

There are plenty of available resources online that may help a novice developer to understand and make informed decisions of hardware design and construction. There is a similar quantity of viable solutions for software and troubleshooting about microcontrollers and modules.

As a summary, the final product satisfies the requirements described in the *Requirements* section; such requirements were later contemplated in our design phase - *Analysis & design of the robot* - . In this document, we have also provided an in-depth description of the manufacturing and assembly of the robot as well as some interesting considerations about the tools required and hardware modules (as regarded in *Prototype construction).* Complementary laboratory sessions have been suggested as well, inspired by their predecessors and adapted to the needs of the Computer Electronics Course as illustrated in *Design of the robot practices.* Although the final product is capable of various advanced features, the learning curve is thoroughly adjusted to the experience level of second course students; therefore, providing a challenging, but instructive experience when taken as a whole.

6. Bibliography

AENOR. (2002). *Criterios generales para la elaboración de documentos.* Madrid: AENOR.

AENOR. (2007). *Criterios generales para la elaboración de proyectos de sistemas de información.* Madrid: AENOR.

Antón, J. C. (2019, April 14). Guía de Prácticas de Laboratorio de Tecnología Electrónica de Computadores.

Arduino. (n.d.). *Arduino.cc.* Retrieved from https://www.arduino.cc

Components101. (2018, March 12). *Arduino Nano.* Retrieved from Components101: https://components101.com/microcontrollers/arduino-nano

Components101. (2018, October 19). *ESP32.* Retrieved from Components101: https://components101.com/microcontrollers/esp32-devkitc

DIGI. (2020). *Digi XBee S2C 802.15.4 RF Modules Datasheet.* Retrieved from Digi.com: https://www.digi.com/resources/library/data-sheets/ds_xbee-s2c-802-15-4

ELEC Freaks. (n.d.). *HC-SR04.* Retrieved from Sparkfun: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Electroschematics. (2020). *18650 battery shield V3.* Retrieved from Electroschematics: https://www.electroschematics.com/battery-shield/

Imperial College London. (n.d.). *SG90.* Retrieved from Imperial College London: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

ISO/IEC/IEEE. (2011). *Systems and software engineering — Life cycle processes — Requirements engineering.* Retrieved from ISO/IEC/IEEE: https://www.iso.org/standard/45171.html

Margolis, M. (2012). *Arduino Cookbook.* O'Reilly.

MOUSER Electronics. (2014). *BH1750.* Retrieved from MOUSER Electronics: https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf

MOUSER Electronics. (2018, September). *BME280.* Retrieved from Mouser Electronics: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf

MPJA. (n.d.). *HC-SR501.* Retrieved from MPJA: https://www.mpja.com/download/31227sc.pdf

Open Impulse. (n.d.). *GY-NEO6MV2 GPS Module.* Retrieved from Open Impulse: https://www.openimpulse.com/blog/products-page/product-category/gy-neo6mv2-gps-module/

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* Project Management Institute.

Toshiba. (2007, June 30). *TB6612FNG.* Retrieved from Sparkfun: https://www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf