INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Robot

Carolina de Oliveira Neves

Final Project Report,
of the undergraduate degree in Engenharia Eletrónica e Telecomunicações e de Computadores
2019-2020 Spring Semester

Advisers : Doctor António Manuel Albuquerque Couto Pinto
Doctor Pedro Miguel Florindo Miguens Matutino

November, 2020

Acknowledgments

First, I would like to thank my advisers, Dr. António Manuel Albuquerque Couto Pinto and Dr. Pedro Miguel Florindo Miguens Matutino, for the availability and for all the help given throughout this PFC.

Thanks to my undergraduate colleagues who proved to be true journey companions and helped me overcome obstacles that allowed me to get here. Also to the teachers with whom I had the pleasure of working along this path.

Lastly a special thanks to my parents for their patience and support. Thank you for all the support given throughout my life and for guiding me to be the person I am today. Without a doubt, without you, I wouldn't be where I am.

Abstract

This project consists of the development of a robot from scratch. The design started with the selection of all its components throw the software implementation. To allow the establishment of a starting point and gathering the indispensable knowledge for the robot prototype development, initial research work has been carried out.

After researching and validating the solutions found, the prototype was assembled, and the required algorithms have been implemented. Finally, validation and functionalities check tests were performed to optimize the developed prototype. The autonomous movement of the robot, herein developed, is based on an anti-collision system, which allows the robot to navigate on a surface without human control.

The remote mode has been implemented through the design of a Web page connected to a server deployed into on the development board.

Keywords: Robot, motor, development board, ultrasonic sensor, control, Web aplication, ESP8266 NodeMCU.

Contents

List of Figures

List of Tables

Acronyms

PFC Course's Final Project. iii, 1
T Time/Period. ix, 9
IP Internet Protocol. ix, 21
IC Integrated Circuit. xi, 10, 11
UC Curricular Unity. 1
LEETC *Licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores*. 1
PWM Pulse Width Modulation. 9
DAC Digital to Analog Converter. 9
BEMF Back Electro Motive Force. 10
RF Radio Frequency. 10, 17
IoT Internet of Things. 12, 13
MD Maximum Distance. 19
d distance. 19
RD Right Distance. 19
LD Left Distance. 19
HTTP Hypertext Transfer Protocol. 21

1

Introduction

1.1 Motivation

This document was made within the scope of the Curricular Unity (UC) Course's Final Project (PFC) of the course *Licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores* (LEETC).

The motivation behind this project was based on the consolidation of the different areas studied throughout the LEETC course, as well as the exploration of the robotics area, since this is the student's area of interest.

This interest begins around 13 years of age the student enrolled in a project carryed out by IBM (EX.I.T.E 2012) with the goal of appeal girls to the Technologies and engineering areas. On this project she had her first contact with LEGO robots and block programming. It was also the project that motivated her to pursue the LEETC course. This project is in the robotics area combining the different aspects of the course, electronics, telecommunications and computers, consisting of the realization of an autonomous robot for domestic use.

1.2 Goals

As mentioned, this project is presented in the scope of the study and exploration of the area of robotics by developing an autonomous robot. This work is divided into four

phases:

1) the development of the robot is preceded by a period of research and study in which, among other requirements, the basis for the robot will be selected, including the wheels, the platform, the motor, the batteries to be used as the power source, the sensor and the development board;

2) definition of the robot's main specifications, where the sensors, actuators and other characteristics have to be defined;

3) the next phase, the platform, wheels and motor will be assembled, as well as the implementation of the driver and the microcontroller. The base movement of the robot will be initially controlled by command or directly by the computer;

4) concept and design the autonomous movement capability of the robot.

1.3 Document Organization

This document is divided into five chapters. The first one contains the motivation, organization and objectives and the structure of this document.

In the second chapter, the selection and requirements for the different elements necessary to the construction of the robot is carried out: robot structure, motor, controller, power supply, micro controller and sensor.

The third chapter describes the entire implementation of the robot. From its practical implementation, with the assembly of the different components, as well as the development of the software carried out for the operation of the same.

The fourth chapter presents the experimental validation of the robot's behavior, that is, the result of the implementations carried out in the previous chapter.

The conclusions are presented in the last chapter as well as some points of future work.

2

System Architecture

The development of a robotic equipment requires the study and establish the requirements of its elements and components, namely: *i*) the robot's structure; *ii*) motor; *iii*) controller; *iv*) power supply; *v*) sensor; *vi*) development board.

The Robot in here proposed is intended for domestic use, in particular with the aim of amusing (toy) and interacting with its user in order to reduce the isolation and making companionship. The mechanical/electronic specifications were defined considering this purpose, meaning that, the robot must be small in size, reasonably light and low cost.

2.1 Robot's Structure

The robot's structure often called a chassis, must meet the requirements in which this robot is designed, so three options were considered:

i) 2 drive wheels → a total of 3 wheels;

ii) 2 drive wheels → a total of 4 wheels;

iii) 4 drive wheels → a total of 4 wheels.

In order to better compare the different types of chassis with the respective number of wheels and cost (at the time of the execution of this project) this values are presented in a Table 2.1.

Table 2.1: List of the chassis and correspondent specifications.

	Wheels	Support Wheels	Weight (g)	Wheels Diameter (cm)	Price (€)
Smart Robot Car Chassis Kit	3	1	220	6,5	13,47
2WD Round Double Layer Chassis	4	2	290	6,5	15,56
Car Kit 01 Joy-iT	4	0	350	6,5	21,20

In order to obtain the best stability possible, the option chosen was a chassis with 4 wheels, reducing the probability of robot flipping. However, due to the intended purpose of the robot, there is no requirement for four wheels with traction, and therefore, a chassis with two wheels with traction and two spherical/support was used. In Figure 2.1 [11] it's possible to find the image of the chassis that was selected for the prototype.

Figure 2.1: Robot chassis (2 wheels with traction and 2 of support).

The wheels of this structure are placed crosswise, there is a support wheel in the front and one on the opposite side, at the back, which will allow very high speed changes in direction and with increased maneuverability.

2.2 Motor

The robot is powered by a continuous voltage from a battery, so a conventional DC motor (with brushes) must be used, Figure 2.2 [5]. These motors have the advantage of being inexpensive, easy to use, light, efficient and it exists a wide range to choose from. Despite requiring some brush maintenance and producing some noise, these are not penalizing factors in this project [6].

Figure 2.2: Conventional DC Cotor (with brushes).

There is a wide variety of applications for this type of engine and almost all share the same basic structure, shown in Figure 2.3 [6].

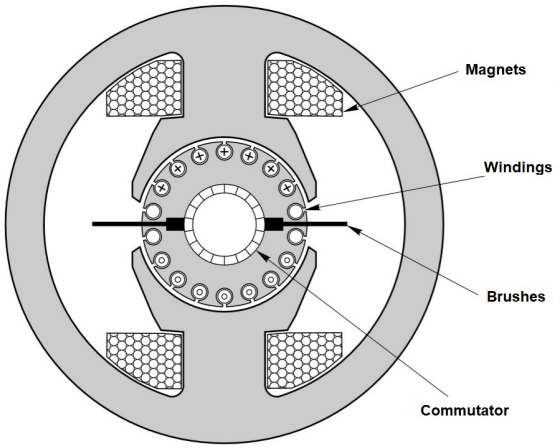


Figure 2.3: Layout of a conventional DC motor (with brushes).

It is possible to design a DC motor for any input voltage, however, it is rare to find motors with voltages below 6 V and above 700 V, for the reasons explained below. The lower limit (6 V) is due to the fact that the brushes have some resistance, creating an inevitable voltage drop, thereby losing a portion of the supply voltage. In relation to the upper limit (700 V), it becomes very expensive to isolate the switch segments to withstand higher voltages.

When electrical energy is supplied to the collector, through the brushes, the motor starts to rotate, providing mechanical power to its shaft [2]. That is, when the windings are subjected to an electric current, they generate their own magnetic field whose magnetic poles are attracted by the opposite magnetic poles, generated by the stator, causing the motor to rotate. With this rotation, the windings are constantly receiving electrical current in different sequences so that the poles generated do not overlap with those of the stator. This field exchange is called switching.

The Figure 2.4 [1] represents the principle of operation of the motor. The electric current comes through one of the brushes (+), enters the commutator plate and returns to the source through the other brush (-), with this the rotor makes its first half turn. In this half turn, the commutator plates exchange contacts with the brushes, which inverts the direction of the current in the rotor coil. This, together with the inertia of the movement, causes the motor to rotate continuously in the same direction.

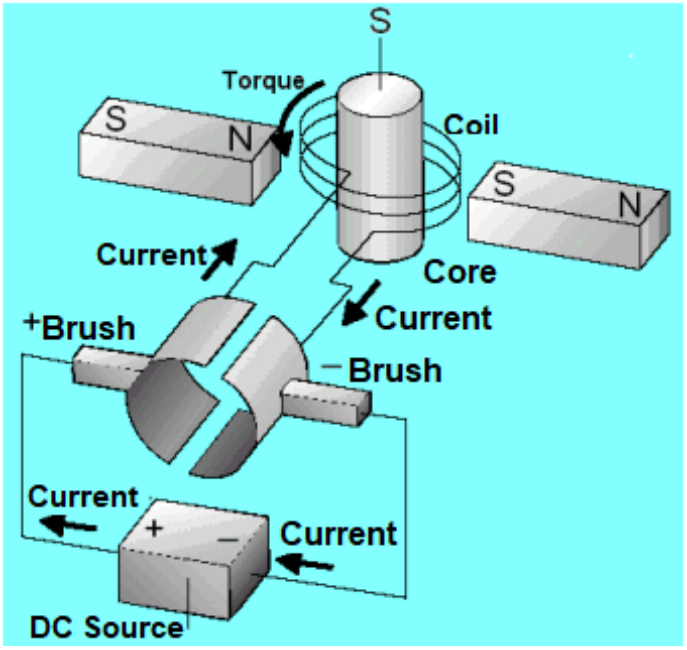


Figure 2.4: Layout of a conventional motor (DC with brushes).

[1] Image taken from [1]

In order to select the engine that best suits the requirements of this project, the most relevant specifications must be considered: *i*) torque; *ii*) speed; *iii*) precision; *iv*) DC voltage; *v*) cost; *vi*) current. Therefor for this project the most relevant specifications are torque and current.

Torque is the engine's ability to provide rotational force to its shaft. In a robot, the motor torque is transferred to the wheel that leads to its movement. The torque can be quantified in different units of measurement, here in kg.cm will be used, and can be calculated from the expression (2.1) [8], where: T - torque (kg.cm); C - friction coefficient; W - weight of the structure (kg); D - wheel diameter (cm).

$$T = 8 \times C \times W \times D \tag{2.1}$$

The total weight of the structure (W) is the sum of the weight of the chassis (290 g), the 2 engines (2×9.5 g) and the powerbank (61 g). Thus it is possible to obtain (2.2).

$$W = 290 + 2 \times 9,5 + 61 = 0,37 \approx 0,4\text{kg} \tag{2.2}$$

Considering $C = 0.03$, which normally varies between 0.001 and 0.03, $D = 6.5$ cm and $W \approx 0.4$ kg, was calculated using of (2.3) the necessary torque for this robot.

$$T = 8 \times 0,03 \times 0,4 \times 6,5 = 0,6243 \text{ kg.cm} \tag{2.3}$$

The calculation of the torque required to move the structure was performed considering only one engine, that is, when only one of the engines is running, making the robot rotate around the other wheel, a situation that is considered the most unfavorable. As two motors are used, along the robot's displacement the effort made by both will be less since both are contributing to the movement.

The motor that will be used is called micro motor, due to its small size. In order to obtain a higher torque, the motor dimension must be larger, however it is also possible to use motors of relatively small dimensions and still achieve high torques through the use of a gearbox and at the expense of reducing the rotation speed at the output.

The gearbox is a set of cogs or gears that allows lowering the rotation speed by increasing the torque. This reduction is achieved by using driving sprockets (controls the movement of the gear) with different sizes. In this case, the smaller gear runs at the speed provided to the motor it self and the other gear is larger in order to decrease the number of revolutions per minute (RPM), as shown in Figure 2.5 , thereby increasing the available torque.

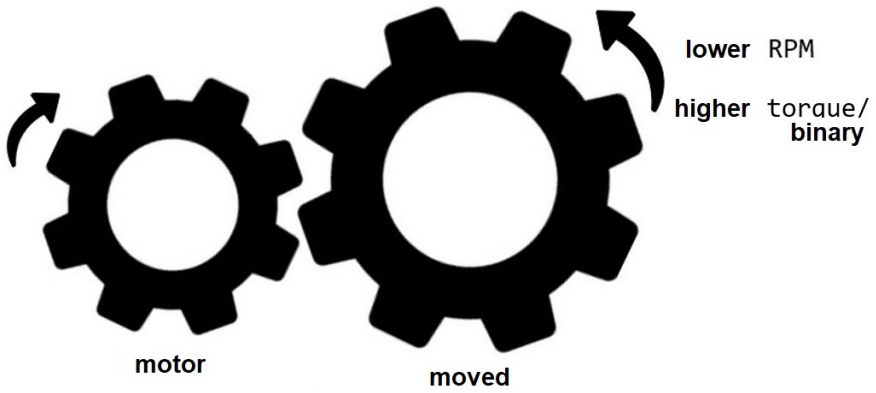


Figure 2.5: Operation of the gearbox.

The relation between torque and speed is given by Equation (2.4) [14], where: T - torque (N m); V - voltage (V); ω - rotational speed (rad s $^{-1}$); k - motor constant (V s); R - resistance (Ω) [3].

$$T = \frac{V - \omega \times k}{R} \times k \tag{2.4}$$

The table 2.2 contains a list of motors with the specifications considered: *i*) DC voltage; *ii*) current; *iii*) speed; *iv*) torque; *v*) price.

Table 2.2: List of engines and their specifications.

	Voltage DC (V)	Current (mA)	Speed (RPM)	Torque (kg.cm)	Weight (g)	Price (€)
1 (ref:096-7646)[7]	6	160	300	0,65	9,5	9,15
2 (ref:096-6472)[7]	6	900	4400	0,07	9,5	10,25
3 (ref:096-6473)[7]	6	350	130	0,79	9,5	10,05
4 (ref:096-6471)[7]	6	900	440	0,58	9,5	10,30
5 (ref:096-5766)[7]	6	200	6000	0,09	9,5	21,00
6 (ref:096-6474)[7]	6	900	733	0,39	9,5	12,70
7 (ref:PTR005066)[10]	12	60	6000	0,09	9,5	21,46

Considering the information present throughout this chapter and the objective of developing a robot for home use at low cost, engine 1 (ref: 096-7646) was chosen. This has a higher torque closer to the calculated value in (2.3) and a good speed ratio $\times$ torque, that is, the ability to move the robot structure at an acceptable speed will be low cost.

2.3 Motor Controller

The motor control is intended to control the motor rotation direction and speed. The direction is controlled by changing the polarity, while the speed is controlled by voltage regulation. The applied voltage can be obtained by changing the value of its amplitude or, as the motor reacts to the average value, it can be done by Pulse Width Modulation (PWM). The variation in amplitude requires the use of an Digital to Analog Converter (DAC), which makes the system more complex and more expensive. For this reason, in this project, the control is performed by PWM.

The PWM controls the *duty cycle*, that is the time that a signal is in the positive alternation during a Time/Period (T) and uses it to establish the speed (as depicted in Figure 2.6). Thus, we have:

- duty cycle = 0%: average voltage equal to 0 V, so the motor is stopped;
- duty cycle = 100%: average voltage equal to the motor supply voltage, so it will rotate with RPM equal to its maximum speed.

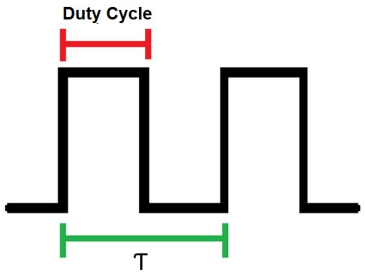


Figure 2.6: Duty cycle and period (T).

When a DC motor is connected to a battery, it rotates at a constant speed and in a single direction. The direction can be changed by reversing the connection of the motor terminals. To avoid doing this operation manually, a motor controller is used, which in this project will have an H bridge configuration.

An H bridge is a combination of 4 transistors positioned in the shape of the letter "H", as it can be seen in the electrical diagram depicted in Figure 2.7, each of which is at one end and the motor is in the middle. For the motor to work, a pair of transistors diagonally opposed (Q1 and Q4 or Q2 and Q3) are activated, which causes the current to flow from the highest potential (V_{bat}) to the lowest (GND) passing through the engine and causing it to turn. To reverse the motor rotation, these transistors are switched off

and the other pair of transistors (Q2 and Q3 or Q1 and Q4, respectively) are connected, causing the current in the motor to flow in the opposite direction, and consequently inverting the motor's rotation.

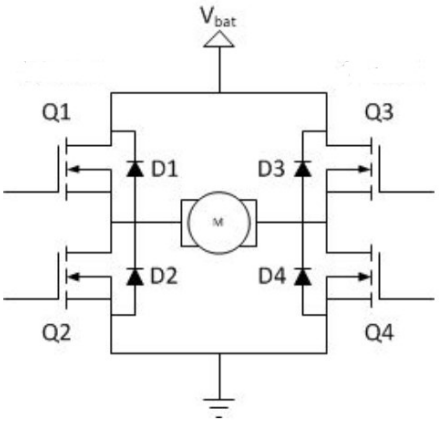


Figure 2.7: Diagram of the H bridge.

The transverse diodes for each transistor, D1 - D4 protect the transistors from current peaks generated by the motor, Back Electro Motive Force (BEMF), when they are switched off. A capacitor can be optionally placed in parallel with each diode in order to reduce the Radio Frequency (RF) radiation that is produced by the switch.

For the implementation of the H bridge, the integrated L923 [9] is selected, which, as it can be seen from the electrical diagram represented in Figure 2.8, is composed of two H bridges, allowing to control up to 2 motors with a single device.

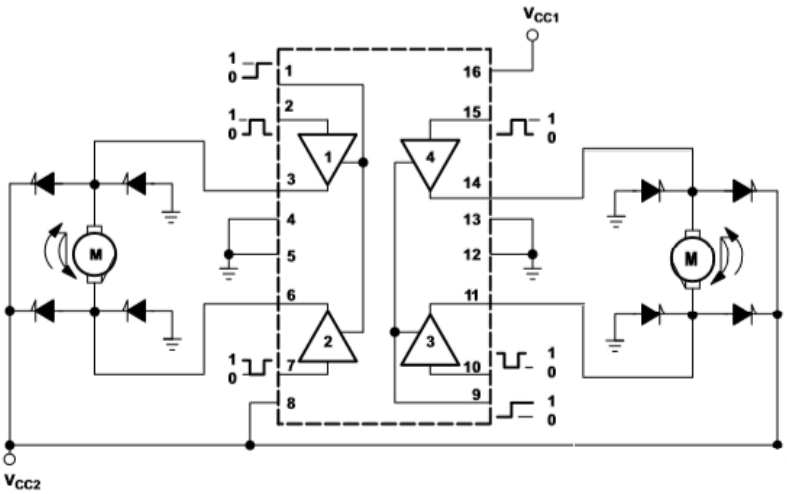


Figure 2.8: Application note for two motors of the L923

Table 2.3 contains the Integrated Circuit (IC) specifications.

Table 2.3: L923 IC specifications.

	Dynamic output range (V)	Price (€)	Current DC max. output (A)	Peak current per channel (A)
L923	4,5 - 36	2,80	1	2

2.4 Power Supply

A powerbank instead of a battery will be used as power supply. This has the advantage of having a better storage capacity, weight and price ratio, allowing for greater storage capacity at a lower weight and price.

The powerbank consists of a set of lithium-ion batteries and a controller that regulates the charge and discharge of the batteries using the input current to control the charging, controlling the heat, never up the temperature limit. They are also composed with a device that allows the measurement of the charge level.

A 5 VDC powerbank is considered as the first approach. However, an initial test will be carried out to check if it is possible to drive both motors directly without significant loss of torque, since the motors requires a continuous voltage of 6 V. Consequently this voltage does not allow to drive the motors a DC-DC converter will be necessary between the powerbank and the L923, in order to raise the voltage above 6 V.

The specifications of the powerbank used are summarized in Table 2.4.

Table 2.4: PowerBank specifications.

	Voltage DC (V)	Capacity (mAh)	Weight (g)	Price (€)
PowerBank USB	5	1000	61	6,70

2.5 Sensor

In order to implement autonomous movement of the robot, the decision-making capability is obtained by the HC-SR04 *ultrasonic sensor* [12] requires. This ultrasonic sensor uses sonar to determine the distance to an object. The transducer sends a high frequency signal, which upon reaching an object is reflected and the reflected wave is received by the sensor. The Figure 2.9 shows the operation of this type of sensors.

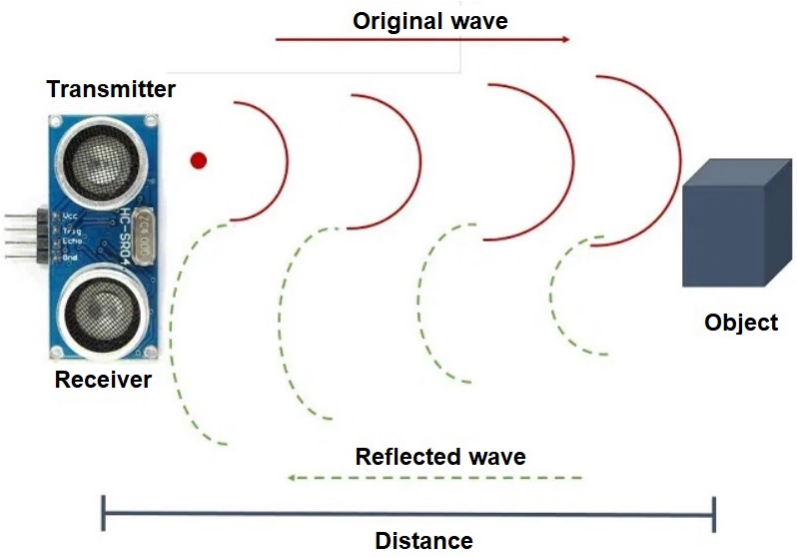


Figure 2.9: Operation of the ultrasonic sensor.

The difference between the times of emission and reception of the signal allows the calculation of the object's distance. In this project, the equation 2.5 was used to calculate the distance through the emission and reception times, where: D - distance (cm); t - duration (s); speed of sound = 0.0343 cm/μ s.

$$D = \frac{0,0343t}{2} \tag{2.5}$$

The relevant specifications of this sensor are summarized in Table 2.5 [13]. The sensor has a range between 3 cm and 400 cm with a resolution of 1 cm.

Table 2.5: Ultrasonic sensor specifications.

	Voltage DC (V)	Minimum reach (cm)	Maximum reach (cm)	Resol. (cm)	Pulse width (μs)	Price (€)
Sonar HC-SR04 ultrasonic sensor	5	3	400	1	10	2,10

2.6 Microcontroller

The development platform chosen for this project was the NodeMCU ESP8266-CH340 [4], based on the ESP8266 microcontroller with integrated Wi-Fi, specially developed for Internet of Things (IoT) applications, but very versatile and adaptable to the main

objectives of this project. The table 2.6 describes some of the most important specifications of the development board.

Table 2.6: Development board specifications.

	Source(V)	Processing speed (MHz)	Flash memory (MB)	Price (€)
NodeMCU ESP8266-CH340	5	80 160	16	5,99

NodeMCU has 128 kB of RAM and 16MB of flash memory for storing data and programs. In addition, this development platform has built-in Wi-Fi, which allows the creation of a server and thus remote control of the robot.

This module has several applications, some of which are the most common: *i*) internet of things IoT; *ii*) automation and Control; *iii*) data transmission; *iv*) data processing; *v*) generate data; *vi*) connection to networks; *vii*) WebServer; *vii*) AcessPoint.

In Figure 2.10, the layout of the development board is shown, with all the functionality of the pins.

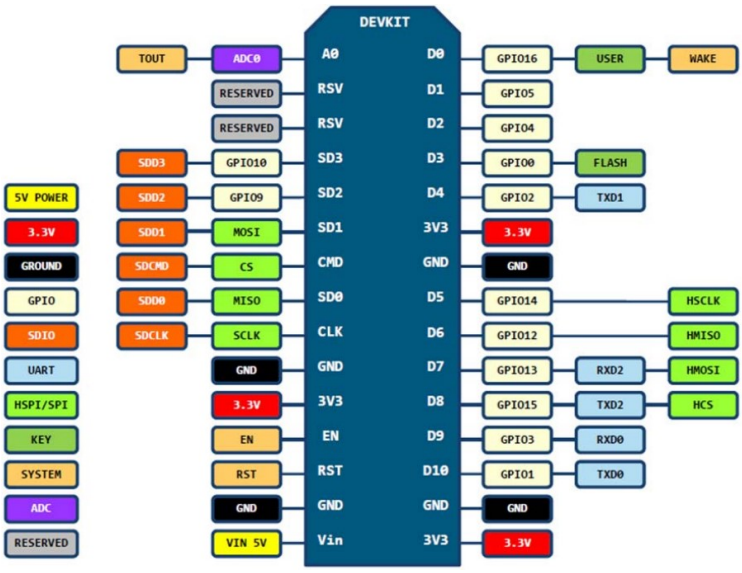


Figure 2.10: Diagram of the NodeMCU ESP8266-CH340 with pin definition.

Throughout this chapter, all the components used were described and a detailed study of each one, its characteristics and ways of functioning was carried out. In the next chapter, the implementations made with the selected components and using all the information and knowledge acquired during the elaboration of this phase of the project will be described.

3

Development

After presenting the context and goals of the project, in this chapter, the implementation details of all the modules developed are presented, including the assembly of the prototype. In Figure 3.1 is the representation of the block diagram, with the blocks positioned by level of abstraction, that is, from the most basic functionalities to the application of control of the entire system and interaction with the user, in the case of software, and the components and corresponding interfaces, in the case of hardware.

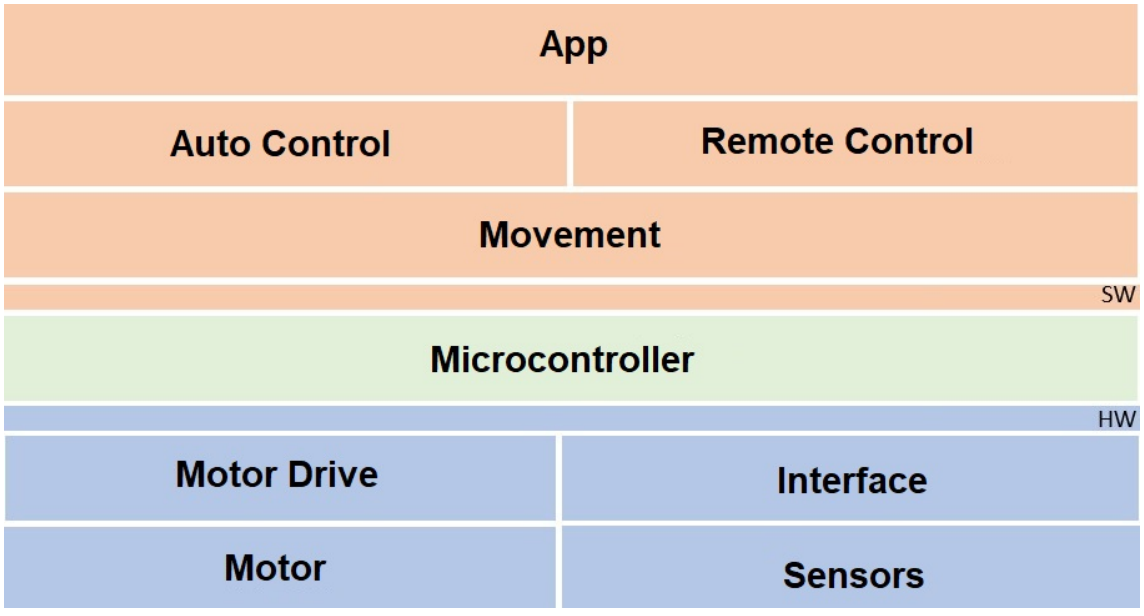


Figure 3.1: Block diagram

3.1 Assembly

In assembling the robot, all the components described in the previous chapter were used. This section describes the assemblies carried out as well as some changes and extra components that have been included.

The first phase consisted of assembling the chassis together with the engines. In Figure 3.2 is the assembly of the robot's skeleton. In order to connect the motors to the circuit, 2 wires were soldered to each motor, all of different colors to better distinguish the motor and its direction. Each motor terminal has been assigned a designation that reflects not only the motor to which it corresponds but also the color of the wire.

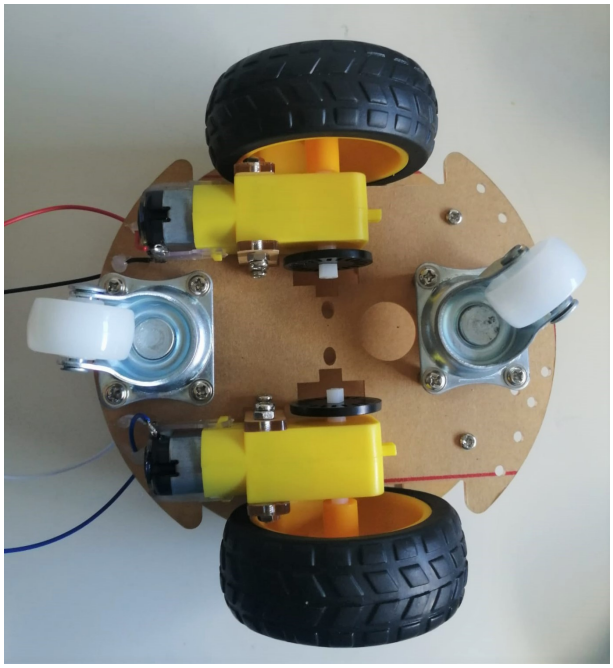


Figure 3.2: Skeleton or base of the robot.

The second phase consisted of assembling the components in breadboard following the electrical diagram represented in Figure 3.3. As depicted in the diagram, the circuit consists of three main components: the motor controller (v1); the ESP826 micro controller (v2) and the ultrasonic sensor (v3).

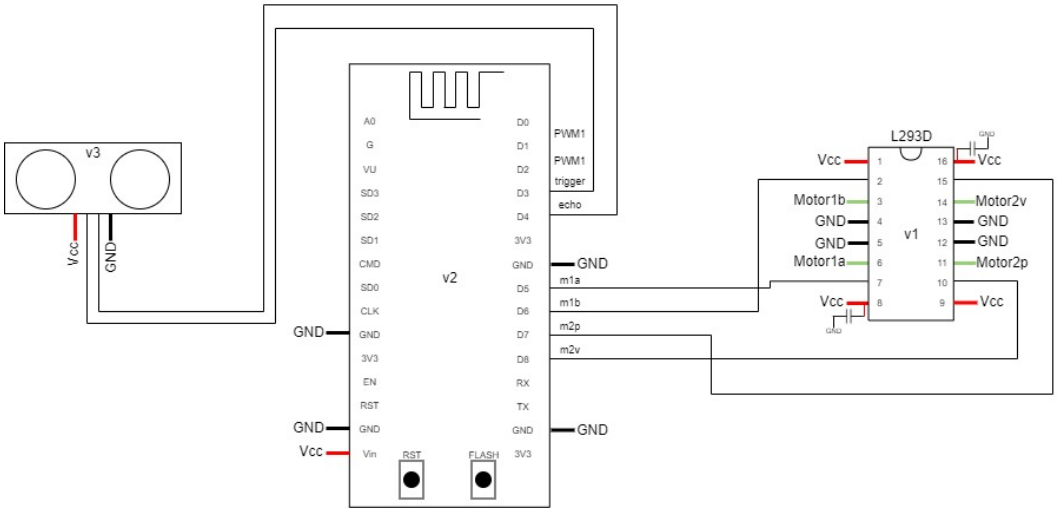


Figure 3.3: Electrical diagram of the proposed System.

To obtain a better functioning of the circuit, copper wires with a bigger diameter than those used in digital interconnections were used, because the section is as bigger as lower the resistance offered, supporting greater of electric currents. Two 100 nF filter capacitors were added to the motor controller feeds, in order to mitigate the noise Radio Frequency (RF) generated by the switching, avoiding the contamination of the power of the remaining circuit by the RF noise.

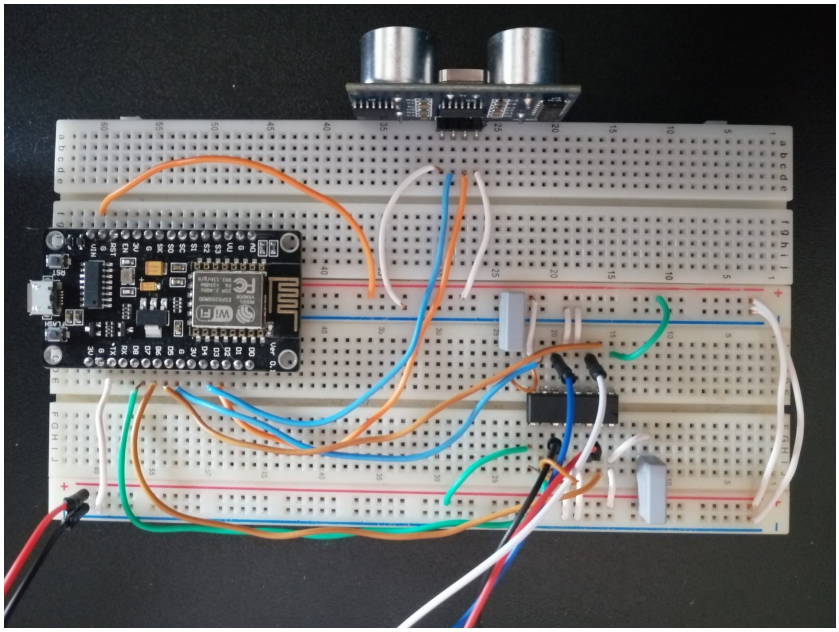


Figure 3.4: Assembly of the circuit in breadboard.

3.2 Implementation of interface systems

This section describes all the basic implementations, that is, all the modules that implements of the most basic functionalities of the robot: movement, decision-making ability.

The robot's navigation system was implemented in two distinct phases. In the first phase, the motor was controlled, where the basic functions of access to the controller were implemented. In the second phase, autonomous navigation of the robot was implemented through the use of the ultrasonic sensor.

3.2.1 Motor control

In order to define the movement of the robot, codes are assigned to each motor terminal, according to the movement intended to be performed, described in Table 3.1.

Table 3.1: Engine operating table.

	Index	**Left Motor a**	**Left Motor b**	**Right Motor a**	**Right Motor b**
Stop	0	0	0	0	0
Front	1	1	0	1	0
Back	2	0	1	0	1
Right	3	1	0	0	1
Left	4	0	1	1	0

To simplify and optimize the algorithm implementation, the table values were placed in two two-dimensional arrays, corresponding to each engine. Subsequently, and depending on the desired direction, the index corresponding to that direction is indicated.

3.2.2 Object detection and decision code

The robot's ability to bypass objects was implemented in two phases. First, an object detection function was implemented by reading the distance. Subsequently, a decision algorithm was created based on the flowchart presented in Figure 3.5.

As previously mentioned, the function `distance()` was developed, with the objective of calculating the distance the robot is from objects and the function `getDistance()` that returns this value.

Knowing that the diameter of the robot platform is 14 cm and that the breadboard is 16 cm long, the reference distance Maximum Distance (MD) was defined with a value that would allow the robot to rotate without hitting the object you are avoiding. So 15 cm was allocated as the ideal distance, enough to prevent the robot from crashing into objects in its blind spots.

After the distance (d) at which the object is found, it is compared with the reference distance MD and if it is higher, the robot continues to move forward. Otherwise, the robot stops and assesses the existence of objects on its right and on its left. In the `RIGHT_90` state, the robot rotates approximately 90º to its right and calculates the distance Right Distance (RD) by comparing it with MD and if it is higher, it means that there is no object, following in front. Otherwise, state `LEFT_180`, the robot rotates approximately 180º to the left and measures this distance Left Distance (LD), following the same logic as in previous decisions. If there is an obstacle in this direction, the robot rotates approximately 90º (state `LEFT_90`) to the left and moves forward, thus moving in the opposite direction to the initial one, this decision process is represented in the flowchart on Figure 3.5.

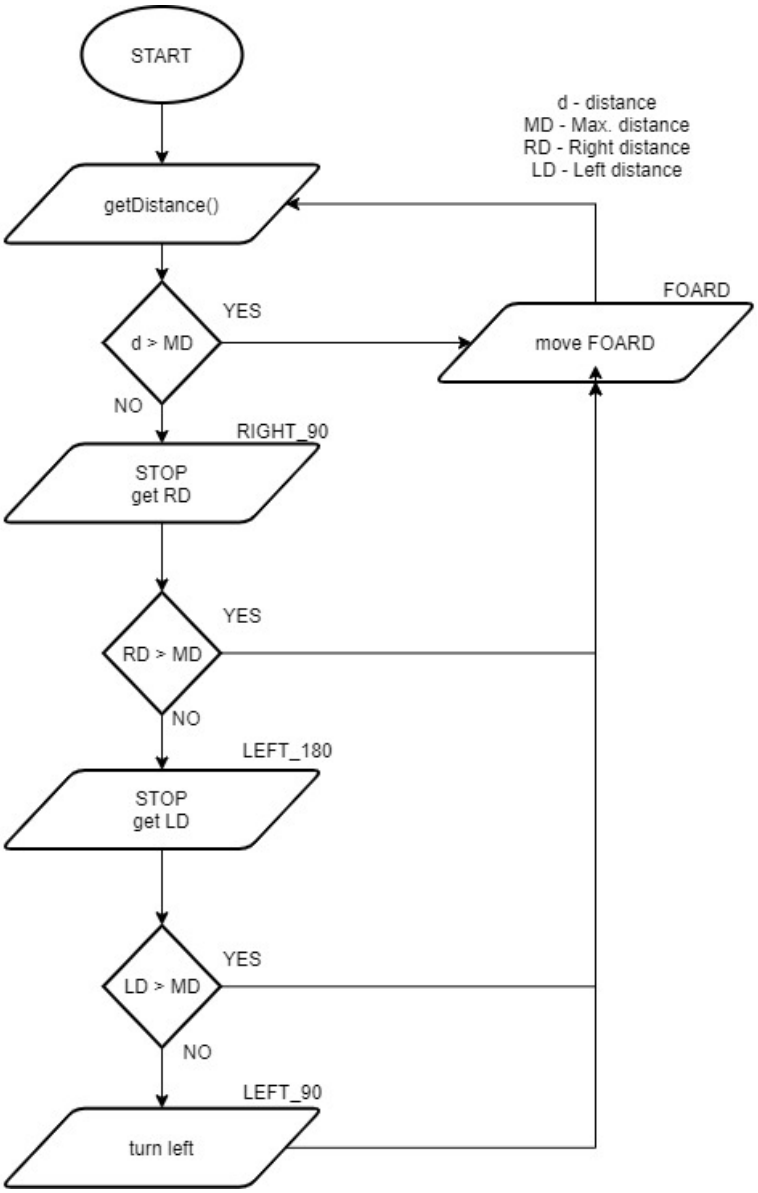


Figure 3.5: Flowchart of the robot displacement decision process.

3.3 Remote control

A remote control was implemented, using the development board's Wi-Fi.

The remote control is implemented through a Web page based on the languages `html`, `CSS` and `javascript`, with the buttons for controlling the basic characteristics of the robot's movement : walk forward, turn right, turn left, walk back and stop. And also the ability to switch to autonomous mode (`AUTO`). The application was developed in a very simple and intuitive way.

First, access credentials are provided to *internet* in order to connect the development

board to the *Wi-Fi* network. With this connection a IP is assigned to the card, which can be obtained through the command `WiFi.localIP()`.

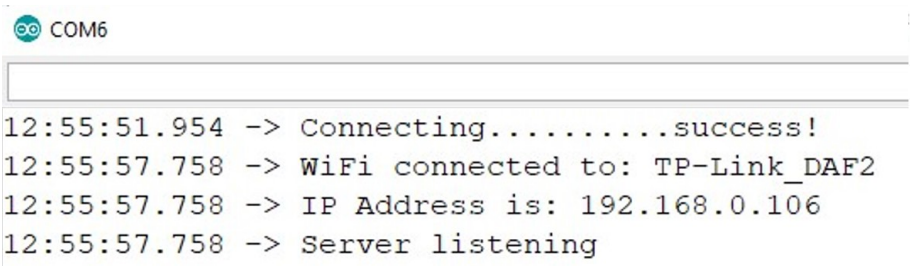


Figure 3.6: Command line with *prints* that return IP.

The Web page is made available abroad by implementing a Web Server on the development platform, using the `server.on()` method to link the `handle` function to the route and its second argument specifies the HTTP method allowed on the route, in this case `HTTP_GET`.

Finally, each button on the Web page is associated with a `id`. When pressed, each button evokes the function `control_btn()` that receives the parameter `id` by parameter. In the `.c` file to access the server, the command `server.arg()` is used, which takes the name of the file `html` as a parameter and returns a *string* with the button triggered, that is, the *id*.

In order to control whether the robot is supposed to be in automatic or manual mode, a flag `control_type` has been created which, by default, is set to 0, which means that the robot is is in manual mode, and is set to 1 when automatic mode is activated. When any of the buttons on the remote are pressed, the robot automatically switches to remote mode. In Figure 3.7 the *Web* page is illustrated.

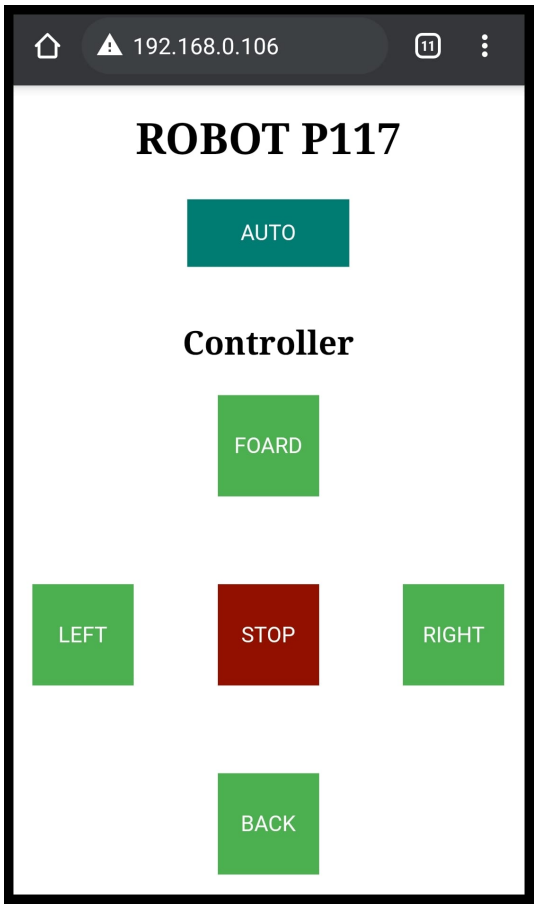


Figure 3.7: Robot control web page.

Throughout this chapter, all implementations, assemblies and algorithms performed with detailed descriptions of them were presented. In the next chapter, several tests are described based on the implementations of this chapter and the final prototype will be also presented.

4

Experimental Evaluation

This chapter presents the results of the tests performed in order to validate the developed system.

In the first section, engines tests were performed, where it was investigated whether the powerbank it was able to supply the current required to start both engines. In the following section, the distances calculated by the ultrasonic sensor were adjusted using consecutive measurements and the comparison between the measurements calculated by the sonar and the measurements with a ruler. Finally, the final prototype is depicted.

4.1 Powerbank and engines

When using the robot it was noticed that sometimes it only started with a manual impulse, for this reason tests were performed on the power supply and motors in order to ascertain the cause of the problem and a possible solution.

We started by measuring the voltage (V) and current (mA) values required to start each motor, individually and simultaneously, and also the minimum values supported with the motors running. The Table 4.1 summarizes the values found with these measurements. When analyzing the table, it is possible to conclude that the right motor and the left motor start, approximately, with the same values of voltage and current, about 1.5 V and stabilize the current at 70 mA, respectively. In addition, when both motors are connected to the power supply, although the starting voltage is the same, they need twice the current value to start, about 138.1 mA.

During this test it was possible to verify that when the two motors are connected to the power supply, there is a peak current, which the powerbank cannot instantly supply. Consequently, there are situations in which the engines cannot start and need a manual boost. For this reason, the initial current peak was analyzed in order to try to understand if the powerbank had the capacity to supply the motors with current.

Table 4.1: Voltage and current values required for starting and minimum supported.

	Left Engine		**Right Engine**		**2 Engines at the same time**	
	Startup	**Red.**	**Startup**	**Red.**	**Startup**	**Red.**
Voltage (V)	1,5	0,7	1,4	0,8	1,7	1,1
Current (mA)	70,9	58,0	68,6	58,9	138,1	126,4

Then, the values of the current as a function of the voltage were recorded, varying between the minimum value required for starting and the 5 V supplied by the voltage source. In the Figure 4.1 shows the graph of the characteristic of the recorded values.

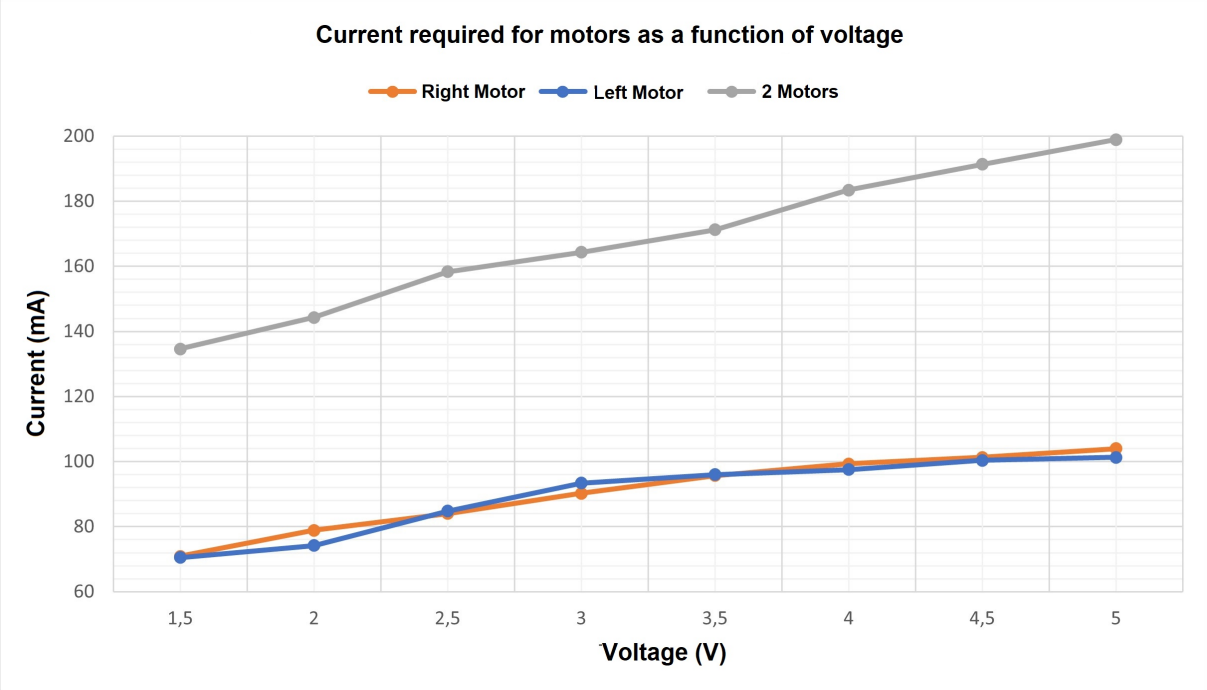


Figure 4.1: Values of the required current depending on the voltage values.

It can be seen from the graph in the Figure 4.1 that there is a slight discrepancy between the values of the two engines. The right motor has different current values for the same voltage than the left motor. When observing the mechanical operation of the two

motors, it is possible to notice that the right motor has a lower rotation compared to the left motor, which allows us to state that the right motor needs more current to run at the same speed. It is for this reason that, when the robot is moving, it descends to the right instead of moving in a straight line. This problem can be corrected by implementing a negative feedback mechanism by obtaining the number of revolutions for each wheel and using this value to make the alignment. Another solution is to add a variable resistance in series with the motor that allows to compensate for this difference and to turn the motor off briefly to control the situation. Figure 4.1 can also confirm that when connecting both motors to the power supply, double the current is required for them to move.

Since the peak current at start-up is too narrow, the multimeter cannot be used, so a resistance of value 1 has been connected 1 Ω and 1 W, in series with the power supply and connecting the oscilloscope in parallel with the resistance, as shown in the diagram of Figure 4.2.

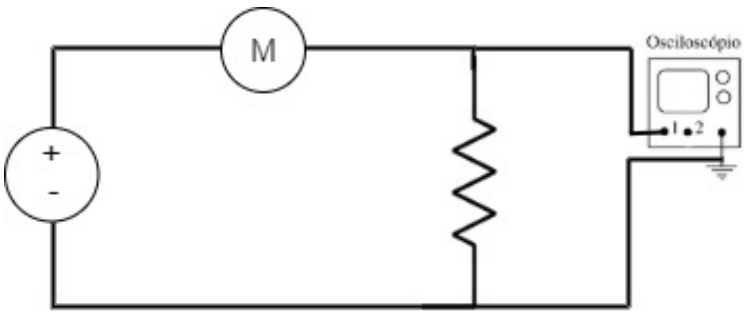


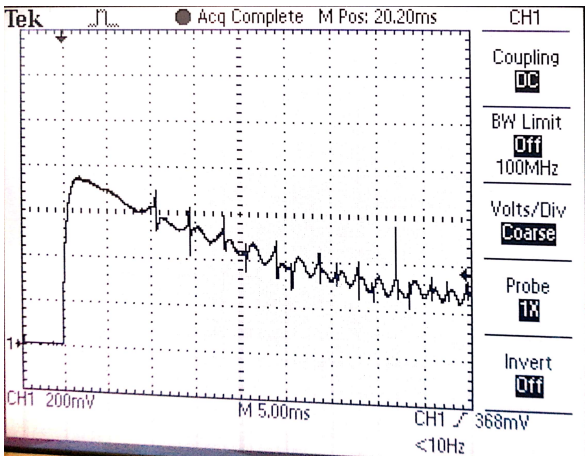
Figure 4.2: Electrical circuit of the assembly performed to observe the peak current.

This test was carried out first with a bench power supply, and only then with the powerbank, so that, since the bench power supply has no current limitation, to understand the current that the required at start-up and be able to compare these results with the results obtained with the powerbank.

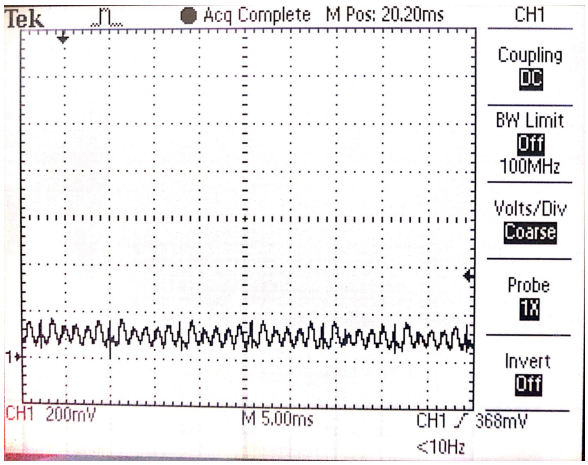
In Figure 4.3 it is possible to observe the results of the tests carried out on the motors with the bench power supply. Figure 4.3 a), c), e) e 4.4 a), c), e) have the graph obtained on the oscilloscope for the peak current values, necessary for the circuit to function and the Figure 4.3 b), d), f) e 4.4 b), d), f) illustrate with the behavior of the voltage over time with the motors connected to the source.

This experiment was carried out with the robot facing upwards (with the wheels facing upwards), thus removing all the extra friction created at the begin of the start-up. This fact leads to the fact that the current required is somewhat higher than the values

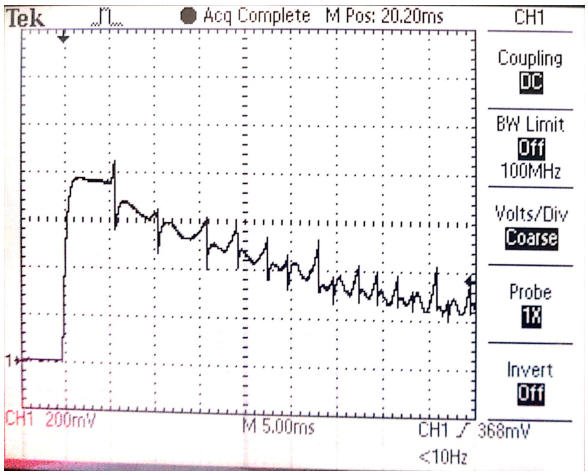
obtained.



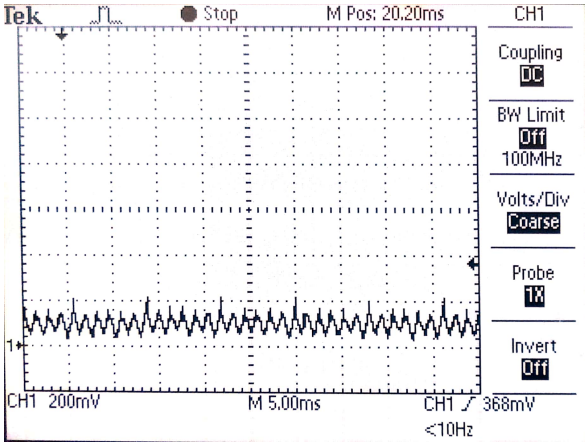
(a) Transient left engine



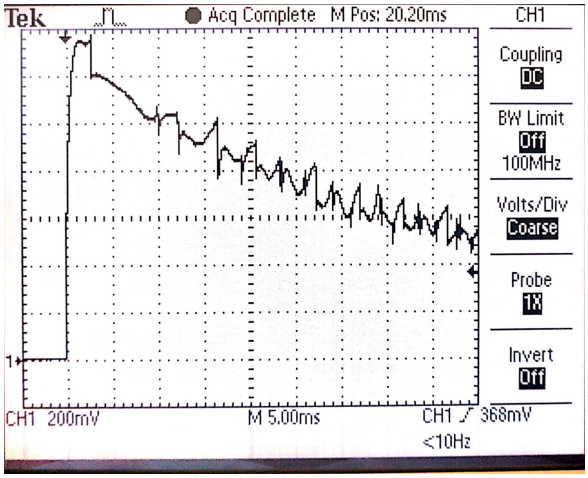
(b) Permanent left motor



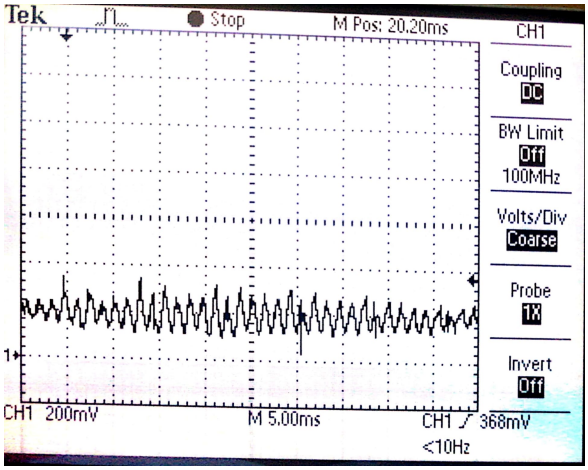
(c) Transient right engine



(d) Permanent right engine



(e) 2 Transient engines



(f) 2 Permanent engines

Figure 4.3: Peak current analysis using a bench power supply.

Through the voltage values presented in the figures and knowing the value of the

resistance used, it is possible to calculate the value of the electric current using Ohm's law (4.1).

$$V = I \times R \tag{4.1}$$

As it is possible to observe by the Figure 4.3a when connecting the left motor directly to the source, there is an initial peak of approximately 800 mV, which results in a current of 800 mA. When the right motor was connected to the circuit, a peak current of 800 mA was also obtained.

In Figure 4.3b it is possible to confirm the results obtained in the test performed previously (Figure 4.1). With 5 V of power the left motor consumes about 100 mA. The same is true for the right motor (Figure 4.3d).

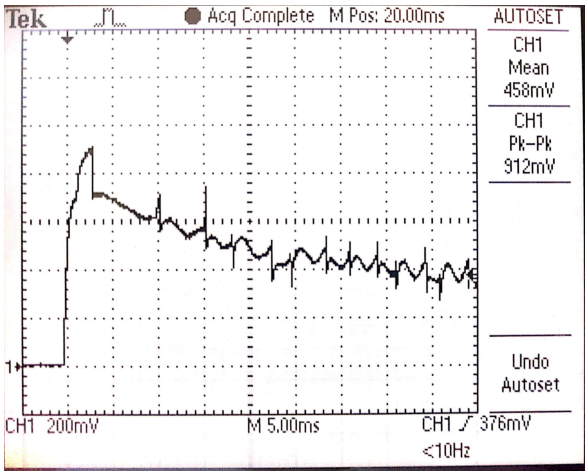
To end the experiment with the bench power supply, both engines were connected to the circuit and the transient of the same was recorded (Figure 4.3e) and their behavior over time (Figure 4.3f). By the same logic, presented above, it is possible to verify that the motors simultaneously require a current peak of 1.4 A and that, as mentioned before, they operate in a steady state with a current of 200 mA, a value consistent with the value displayed on the graph of Figure 4.1.

After checking the current required by the motors, using the bench voltage generator, the same test was carried out, but using a powerbank, in order to fulfill the objective of this test: check if the powerbank has the capacity to supply the necessary electric current when starting the engines simultaneously. The results of this test are in the Figure 4.4.

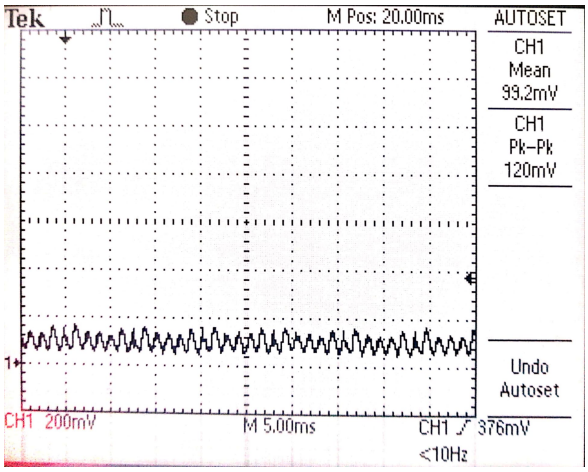
In general, it is possible to see that the values obtained by the motors connected individually are similar to those obtained with the bench power supply. Which allows us to conclude that the powerbank is able to deliver the peak 800 mA required by the engines (Figure 4.4a e 4.4c).The values of the operating current in steady state Figure 4.4b and 4.4d) are also consistent with those obtained previously (Figure 4.1), with 100 mA.

Finally, the two motors were connected simultaneously to the circuit powered by the powerbank resulting, as expected, in a current saturation. Since the powerbank indicates a maximum current of 1000 mA at its output (Table 2.4),it is possible to predict that it will not be possible to provide the necessary 1.4 A. In the Figure 4.4e this forecast is confirmed, presenting a peak 1.2 V, equivalent to 1.2 A.

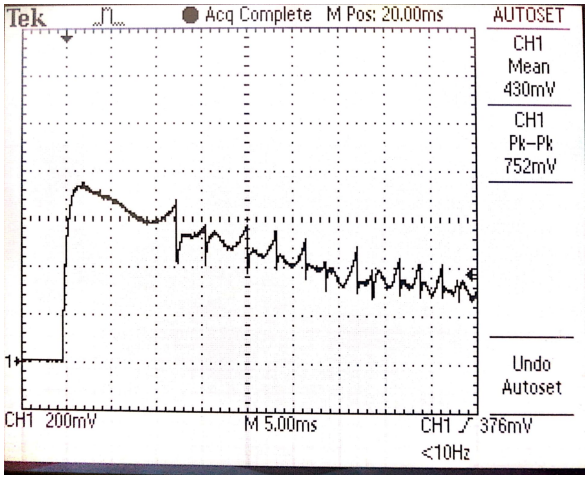
In Figure 4.4f shows the current drawn by the motor over time, the graph allows us to conclude that the current has an average value of approximately 200 mA, as expected (Figure 4.1).



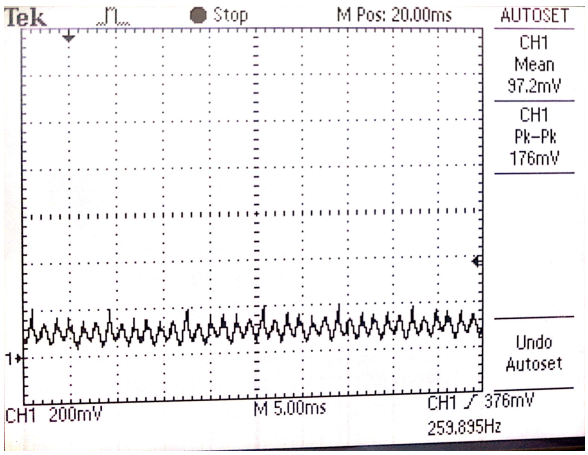
(a) Transient left engine



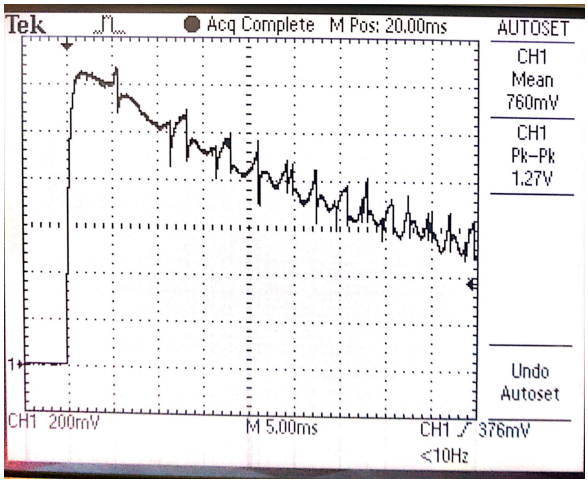
(b) Permanent left motor



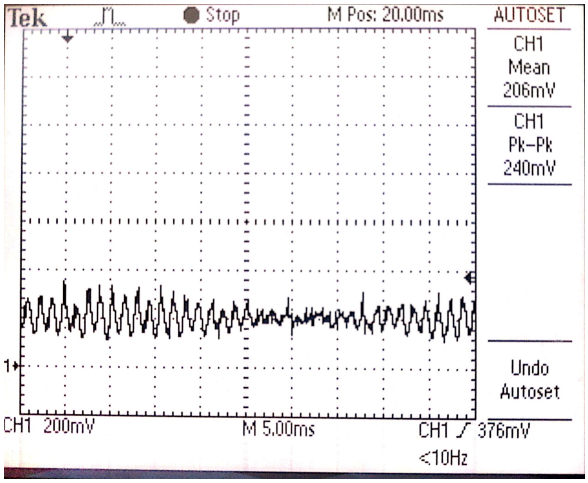
(c) Transient right engine



(d) Permanent right engine



(e) 2 Transient engines



(f) 2 Permanent engines

Figure 4.4: Peak current analysis using the powerbank

With these results it is possible to conclude that the powerbank is capable of supplying current to each engine, individually, and at startup however the same does not happen

when the engines are started simultaneously. In this case, the powerbank is able to supply the necessary current in a steady state, and is only able to operate both engines as long as the start is out of phase.

The solution found to solve the problem was changing the engine start code. A waiting time has been added between starting the motors, so that the current peaks of each motor are not required at the same time, thus guaranteeing the current needed to start each one. This problem was corrected by inserting a mismatch between the motor connection.

4.2 Ultrasonic sensor and distance evaluation

In order to validate and characterize the relative and absolute errors of the ultrasonic sensor, the distances obtained in the ultrasonic sensor were compared with the actual values measured with a ruler.

Table 4.2 summarizes the actual values with the respective values obtained, as well as the calculation of errors.

The absolute error was calculated using the formula (4.2) and the relative error was calculated using the formula (4.3).

$$EA = medido - real \tag{4.2}$$

$$ER = \frac{EA}{calculado} \tag{4.3}$$

The measurements depicted in the Table 4.2 were performed several times. These values were not presented because the results were always identical. This fact allows us to state that the errors presented are systematic and constant, which means that the value, in module, of the absolute error, can be corrected algebraically, thus compensating the systematic error.

Table 4.2: Distances measured by the ultrasonic sensor compared to actual distances.

Real (cm)	Measured (cm)	Absolute error (cm)	Relative error (%)
3	2,0	-1,0	50
4	3,0	-1,0	33
5	4,0	-1,0	25
6	5,5	-0,5	9
7	6,0	-1,0	17
8	7,0	-1,0	14
9	8,0	-1,0	13
10	9,5	-0,5	5
11	10,0	-1,0	10
15	14,0	-1,0	7
20	19,0	-1,0	5

As mentioned in the second chapter, the sensor has a 3 cm blind zone, which means that it can only make correct measurements of distances greater than 3 cm. At Figure 4.5 it is possible to observe the absolute error graph as a function of distance. When analyzing the graph it is possible to see how the error remains for the different distances. It is this fact that makes the error systematic and, as mentioned earlier, can be mitigated.

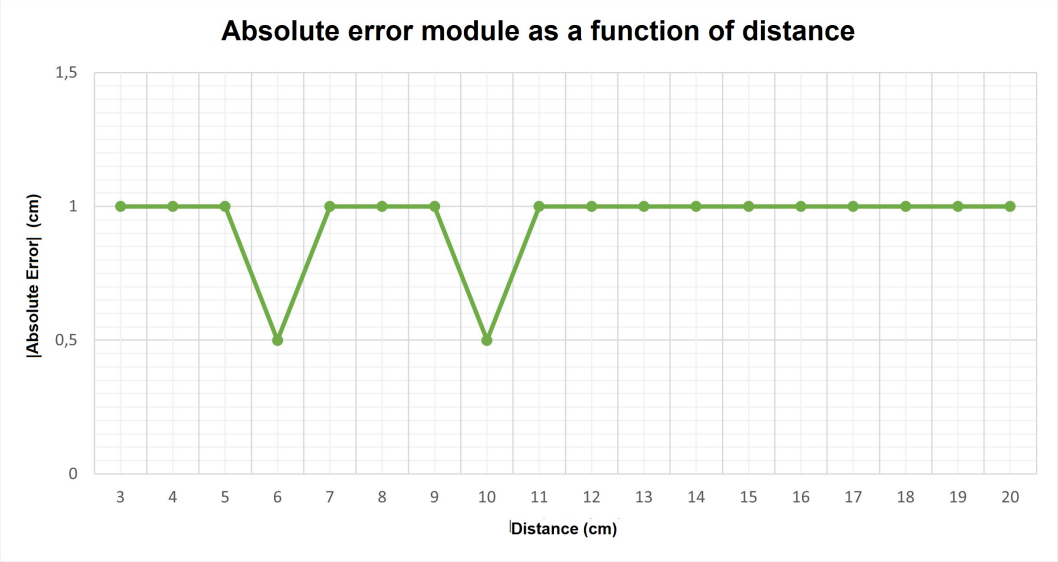


Figure 4.5: Graph of the absolute error module as a function of the actual distance.

In the previous sections, belonging to this chapter, several tests were carried out in

order to validate the system developed in this project and also allow to optimize some algorithms and correct some errors. In the next section, the solutions implemented in order to optimize the robot will be presented.

4.3 Robot prototype

As mentioned earlier, this chapter will present the changes implemented. We start by transferring the entire project previously implemented on breadboard to a perforated printed circuit assembly board, where all the components were welded. Resulting on the final prototype on Figure 4.6.

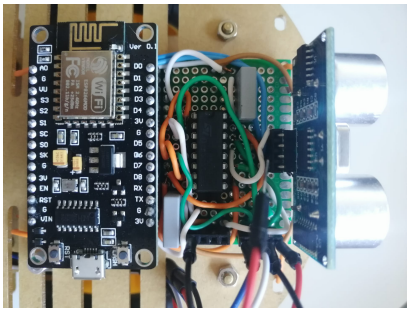


Figure 4.6: Prototype of the robot's control system.

Table 4.3: Distances measured by the ultrasonic sensor compared to real distances, after correcting the algorithm.

Real (cm)	Measured (cm)	Absolute error (cm)	Relative error (%)
3	3,0	0	0
4	4,0	0	0
5	5,0	0	0
6	6,0	0	0
7	7,0	0	0
8	8,0	0	0
9	9,0	0	0
10	10,5	0,5	4
11	11,0	0	0
15	15,0	0	0
20	20,0	0	0

One of the corrections that was made was related to the distance, in which the value of the absolute error was added to the value of the distance obtained by the sonar.

To validate this change, the same test was repeated and the values indicated in the Table were obtained 4.3. Analyzing the values it is possible to notice that the measurements obtained by the sensor coincide with the measurements obtained with the ruler.

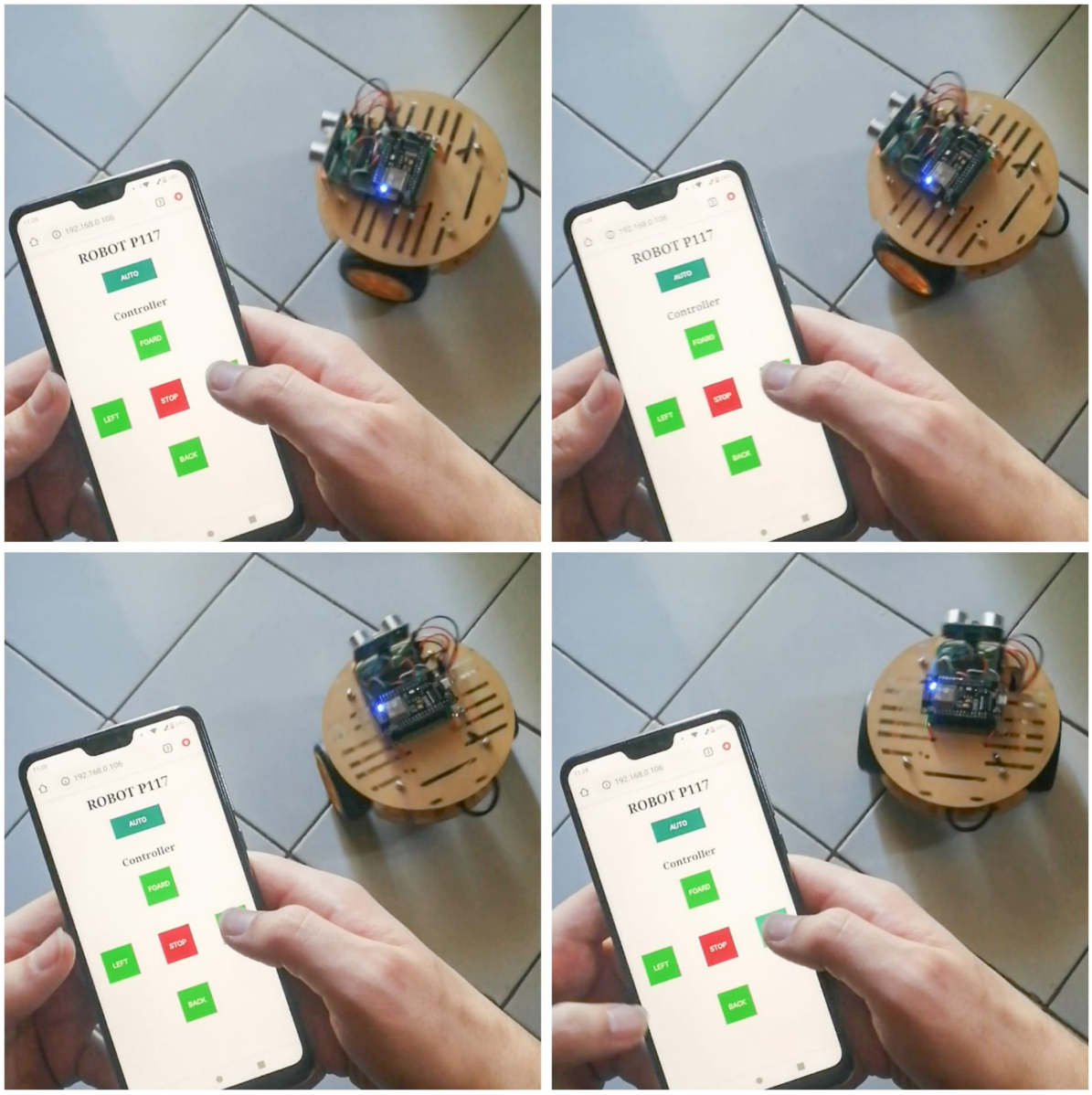


Figure 4.7: Series of photographs illustrating the application's control of the robot.

It was also possible to observe that by placing a delay in the connection of the motors, that is, not being activated simultaneously, it solved the problem of the robot not being able to start without receiving a manual impulse. In Figure 4.7 it is possible to observe

a series of 4 images that aim to illustrate the movement performed by the robot when it is being controlled by the application. In this case, it is possible to see the movement of turning clockwise.

Finally, it presents itself in the Figure 4.8 the prototype developed throughout this project.

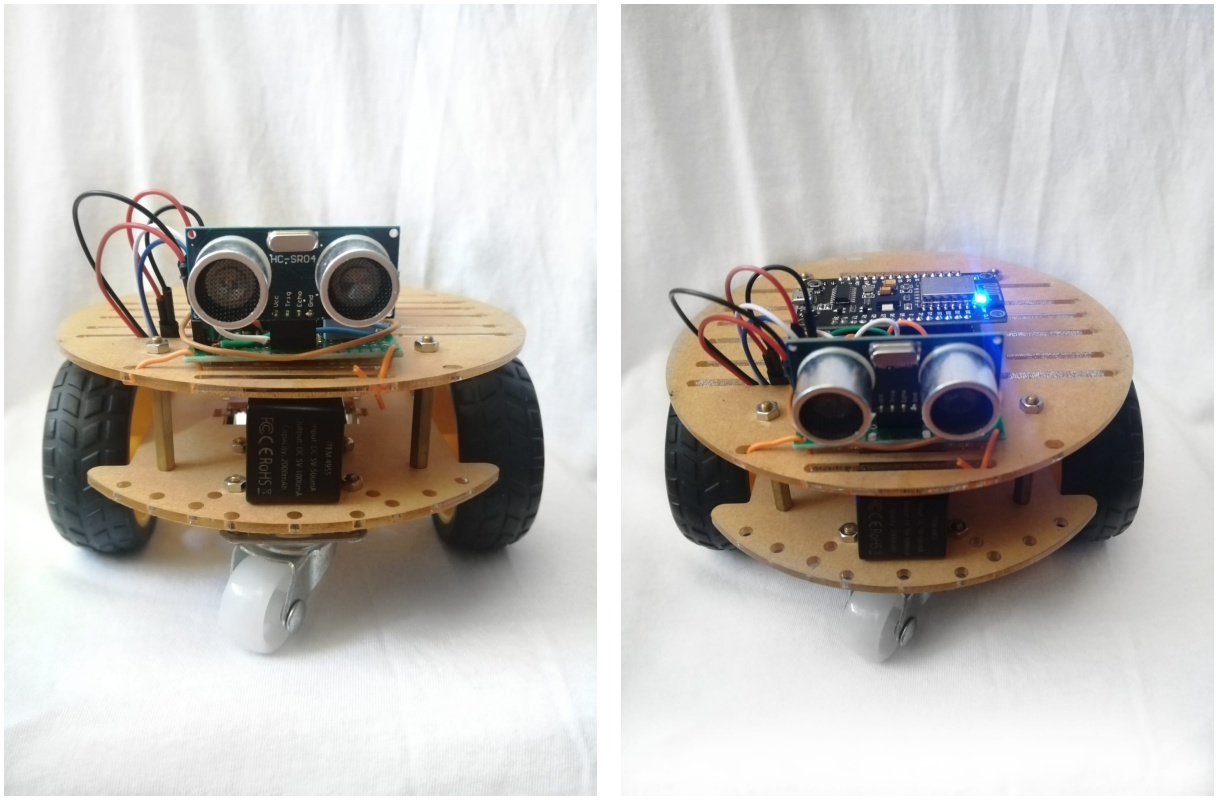


Figure 4.8: Robot prototype.

5

Conclusions and Future Work

This chapter presents a summary of the work carried out within the scope of this final course's project and the relevant conclusions resulting from it, concluding with suggestions for future work.

The project underwent some changes due to the restrictions suffered during the school semester (COVID-19). However, it is possible to state that the project has fulfilled the main goals proposed. The area of robotics was explored through a detailed study of the different components necessary for the development of the robot as well as its characteristics and ways of implementation. This research and the work carried out resulted in a prototype that performs the basic functions intended.

Given the area and nature of the project, it offers several possibilities for expansion and future work. However, some possible future implementations are: the use of a step up for speed control, since with the current assembly the motor controller cannot change the speed supplied to the motors. Also the use of an infrared sensor to detect small objects such as cables and stairs thus avoiding falling or getting stuck.

References

[1] elearning, *Tipos de motores*, https://elearning.iefp.pt/pluginfile.php/47163/mod_resource/content/0/CD-Rom/Estudo/Electr_nica_e_Electricidade_N_vel_III/G_-_M_quinas_El_ctricas/frame_3.htm, [*Online*: acedido em Maio de 2020].

[2] José Vagos Carreira Matias, *Máquinas Eléctricas de Corrente Contínua*. Didáctica Editora, 2005.

[3] Mechatronics Principle and Applications, *Onwubolu*.

[4] Handson Technology, *ESP8266 NodeMCU WiFi Devkit*, https://www.handsontec.com/pdf_learn/esp8266-V10.pdf, [*Online*: acedido em Agosto de 2020].

[5] Future Electronics, *Future Electronics*, https://store.fut-electronics.com/products/micro-metal-gear-dc-motor-120-rpm, [*Online*: acedido em Maio de 2020].

[6] Austin Hughes, *Electric Motors and Drives*. Newnes, 2006.

[7] Mouser.pt, *Mouser*, https://mauser.pt/catalog/, [*Online*: acedido em Novembro de 2020].

[8] Instructables Circuits, *Complete Motor Guide for Robotics*, https://www.instructables.com/id/Complete-Motor-Guide-for-Robotics/, [*Online*: acedido em Maio de 2020].

[9] Texas Instruments, *L293x Quadruple Half-H Drivers*, https://www.ti.com/product/L293D, [*Online*: acedido em Maio de 2020], 2016.

[10] Mouser.pt, *PTRobotics*, https://www.ptrobotics.com/, [*Online*: acedido em Novembro de 2020].

[11] PTRobotics, *Chassi*, https://www.ptrobotics.com/chassis/7353-2wd-round-double-layer-chassis-smart-car-kit.html, [*Online*: acedido em Maio de 2020].

[12] ELEC Freaks, *Ultrasonic Ranging Module HC - SR04*, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf, [*Online*: acedido em Novembro de 2020].

[13] Mouser.pt, *Sensor ultrassónico (HC-SR04) compatível com Arduino*, https://mauser.pt/catalog/product_info.php?cPath=1667_2669_2677&products_id=096-6220, [*Online*: acedido em Novembro de 2020].

[14] Motion Control tips, *The Torque Equation and the Relationship with DC Motors*, https://www.motioncontroltips.com/torque-equation/, [*Online*: acedido em Maio de 2020].

Code writen on for the project

A.1 Principal File - P117

```
#include <ESP8266WebServer.h>

/*********************************
 *       MACROS AND VARIABLES
 *********************************/
#define TRIGGER 0 // gpio sonar trigger
#define ECHO 2 // gpio sonar echo

// Description of all the states and motions
#define STOP 0
#define FRONT 1
#define BACK 2
#define RIGHT 3
#define LEFT 4
#define FRONT_STOP 11
#define ROTATE_RIGHT_90_START 5
#define ROTATE_RIGHT_90_EXEC 6
#define ROTATE_LEFT_90_START 7
#define ROTATE_LEFT_90_EXEC 8
#define ROTATE_LEFT_180_START 9
#define ROTATE_LEFT_180_EXEC 10

// remote or auto control flag
```

```
24 #define AUTO 1
25 #define REMOTE 0
26
27 #define DISTANCE_TO_DODGE 15 // the maximum distance in cm
28
29 #define NUMBER_OF_MOTORS 2 // number of motors
30
31 //uint8_t pwm1 = 5; //Nodemcu PWM pin
32 //uint8_t pwm2 = 4; //Nodemcu PWM pin
33 int motor[] = {14, 12, 13, 15}; // motors pins
34
35 //                              {{  STOP  },{  FOARD  }, {  BACK  }, {
       RIGHT  }, {  LEFT  }}
36 int direction__motor1[7][2] = {{LOW, LOW},{HIGH, LOW}, {LOW, HIGH}, {HIGH,
       LOW}, {LOW, HIGH}}; // LEFT MOTOR
37 int direction__motor2[7][2] = {{LOW, LOW},{HIGH, LOW}, {LOW, HIGH}, {LOW,
       HIGH}, {HIGH, LOW}}; // RIGHT MOTOR
38 int decision = 0
39 ; // this variable stores the movement
40
41
42 ESP8266WebServer server(80); // this is the server port
43
44
45 /*********************************
46  *             CODE
47  *********************************/
48
49 void setup() {
50   Serial.begin(9600);
51   Serial.println();
52
53   // Declaring Sonar pins
54   pinMode(TRIGGER, OUTPUT);
55   pinMode(ECHO, INPUT);
56
57   //Declaring l293d control pins as Output (motor pins)
58   for (int i = 0; i < 4; i++){
59       pinMode(motor[i], OUTPUT);
60   }
61   serverSetUp(); // this funtion does the server setup
62 }
63
64 void loop() {
65   server.handleClient(); //Handling of incoming client requests
```

```
66
67    if (getTypeControl() == AUTO){ // avaliates if it's on auto mode
68      distance();
69      decision = movementDecision();
70    }
71    rolling(decision, HIGH); // implements the movement
72  }
```

A.2 motor

```
1   /*******************************************************
2    * rolling - this funcion recives the direction that the
3    * motors are suposed to rotate and the velocity (so it
4    * can be altered with PWM in the future.
5    *******************************************************/
6   void rolling(int diretion, bool velocity){
7       switch(diretion){
8         case ROTATE_RIGHT_90_START:
9           diretion = RIGHT;
10          break;
11        case ROTATE_RIGHT_90_EXEC:
12          diretion = STOP;
13          break;
14        case ROTATE_LEFT_180_START:
15          diretion = LEFT;
16          break;
17        case ROTATE_LEFT_180_EXEC:
18          diretion = STOP;
19          break;
20        case ROTATE_LEFT_90_START:
21          diretion = LEFT;
22          break;
23        case FRONT_STOP:
24          diretion = STOP;
25          break;
26      }
27
28      for(int i = 0; i < NUMBER_OF_MOTORS; i++){
29        motorSetUp(diretion, i, velocity);
30        delay(100);
31      }
```

```
32      //digitalWrite(pwm1, velocity); //pwm output
33      //digitalWrite(pwm2, velocity); //pwm output
34  }
35
36  /*****************************************************
37   * motorSetUp - it recives the direction the motor is
38   * suposed to rotate(direction); the number of the motor
39   * implement (number_of_motor) and the velocity (so it
40   * can be altered with PWM in the future (velocity).
41   *
42   * This puts the digital value in each terminal of the
43   * motor passed by parameter.
44   *****************************************************/
45  void motorSetUp(int diretion, int number_of_motor, bool velocity){
46    if (number_of_motor == 0){ //inicia o primeiro motor
47      for(int i = 0; i < 2; i++){ // the 2 are the number of terminals the
      motor has
48        digitalWrite(motor[i], direction__motor1[diretion][i]);
49      }
50    }
51    if (number_of_motor == 1){ //inicia o segundo motor
52      for(int i = 0; i < 2; i++){ // the 2 are the number of terminals the
      motor has
53        digitalWrite(motor[i+2], direction__motor2[diretion][i]); // +2 is so
       itimplements the right motor on the array
54      }
55    }
56  }
```

A.3 decision

```
1   #define DEGRES_90 500 // time to turn aprox. 90 degrees
2
3   int state; // this variable keeps the state of the decision
4   unsigned long millisTemp; // this variable stores the initial time for the
      timer
5
6   /*****************************************************************
7    * movementDecision - it implements the flowchart created
8    * it returns the decision that was made.
9    *
```

```
 * This function considers the possibility that objects can no apear
 * from nowhere. Whitch means that if the robot took a certain path
 * it can go on that same path again.
 ****************************************************************/
int movementDecision(){
  switch(state){
    case STOP: //turns right to avaluate the distance
      state = FRONT;
      break;
    case FRONT: // when it first finds an object
      if (getDistance() > DISTANCE_TO_DODGE){ //object not found
        state = FRONT;
      }
      else // object found
        state = FRONT_STOP;
      break;

    case FRONT_STOP: //turns right to avaluate the distance
      millisTemp = millis() + DEGRES_90; //starts timer
      state = ROTATE_RIGHT_90_START;
      break;

    case ROTATE_RIGHT_90_START: //turns right to avaluate the distance
      if(millis() > millisTemp){
        if (getDistance() > DISTANCE_TO_DODGE){ //object not found
          state = FRONT;
        }
        else // object found
          state = ROTATE_RIGHT_90_EXEC;
      }
      break;

    case ROTATE_RIGHT_90_EXEC: //turns left to avaluate the distance
      millisTemp = millis() + DEGRES_90*2; // starts timer to turn 180 = 90
    *2
      state = ROTATE_LEFT_180_START;
      break;

    case ROTATE_LEFT_180_START: //turns right to avaluate the distance
      if(millis() > millisTemp)
        if (getDistance() > DISTANCE_TO_DODGE){ //object not found
          state = FRONT;
        }
        else // object found
          state = ROTATE_LEFT_180_EXEC;
```

```
54        break;
55
56      case ROTATE_LEFT_180_EXEC: //turns left to avaluate the distance
57        millisTemp = millis() + DEGRES_90; // starts timer to turn 90
58        state = ROTATE_LEFT_90_START;
59        break;
60
61      case ROTATE_LEFT_90_START: //turns right to avaluate the distance
62        if(millis() > millisTemp){
63          state = FRONT;
64        }
65        break;
66    }
67    return state;
68  }
```

A.4 distance

```
1   #define SPEED_OF_SOUND 0.0343 // speed sound = 343 m/s and 0.0343 cm/us
2   #define COMPENSATION_ERROR 1 // the error correction found during the tests
3
4   long  distanceValue; // this variable keeps the distance calculated
5
6
7   /*****************************************************
8    * getDistance - it returns the distance of the object
9    * standing in the way.
10   *****************************************************/
11  long getDistance (){
12    return distanceValue + COMPENSATION_ERROR; //the distance calculated plus
        the compensation
13  }
14
15  /*****************************************************
16   * distance - it sends and recives a pulse through the
17   * sensor to calculate the distance of the object.
18   * And it stores the distance in the variable distanceValue
19   *****************************************************/
20  void distance(){
21    long duration; // a local variable that stores the duration os the pulse
22    digitalWrite(TRIGGER, LOW);
```

```
23    delayMicroseconds(2);
24
25    digitalWrite(TRIGGER, HIGH); // sends the pulse
26    delayMicroseconds(10);
27
28    digitalWrite(TRIGGER, LOW);
29
30    duration = pulseIn(ECHO, HIGH); // returns the length of the pulse in
        microseconds
31    distanceValue = (duration * SPEED_OF_SOUND) / 2; // stores the distance
        -> speed sound in cm/s; it's devided by 2 because it's the 2 ways the
        pulse makes
32
33  }
```

A.5 servidor

```
1   /*********************************
2    *       MACROS AND VARIABLES
3    *********************************/
4   #include "web_page.h"
5
6   String accessName = "p107";
7   // wifi's pass and id
8   //const char *ssid = "TP-Link_DAF2";
9   //const char *password = "37481735";
10  const char *ssid = "NOS-BED0"; //192.168.1.7
11  const char *password = "5ab74797f6aa";
12
13  String id_btn; // it stores the id of the button pressed
14  int  control_type = 0; // flag that says if it's on auto or remote mode
15
16  /*********************************
17   *            CODE
18   *********************************/
19
20  /*****************************************************
21   * serverSetUp - it implements the server
22   *****************************************************/
23  void serverSetUp(){
24    // Connect to WiFi network
```

```
25    WiFi.begin(ssid, password); // it gives the id and pass to the board
26
27    Serial.print("Connecting");
28    while (WiFi.status() != WL_CONNECTED){ // it checks the status
29      delay(500);
30      Serial.print(".");
31    }
32
33    Serial.println("success!"); // it conected succesfully
34    Serial.print("WiFi connected to: ");
35    Serial.println(WiFi.SSID()); // prints the id
36    Serial.print("IP Address is: ");
37    Serial.println(WiFi.localIP()); // Will print IP address
38
39    server.on("/", handleIndex); //Handle Index page
40    server.on("/control", HTTP_GET, remoteControl); // it sets the hangle,
       the request http GET and it gives the fuction to be redirected to
41
42    server.begin(); //Start the server
43    Serial.println("Server listening"); //waits for a client
44  }
45
46  /*****************************************************
47   * handleIndex - it return the OK whet it recives the get
48   * request.
49   *****************************************************/
50  void handleIndex(){
51    server.send(200, "text/html", control_page); // it returs the OK
52  }
53
54  /*****************************************************
55   * remoteControl - this function  controls the type of
56   * mode (remote or auto). It gets the pressed button and
57   * avaluates the button assigning it a decision
58   *****************************************************/
59  void remoteControl(){
60    id_btn = server.arg("control");//recive the parameter of function and
       returns teh pressed button
61    // it assigns a value to the flag
62    if(id_btn == "AUTO")
63      control_type = AUTO;
64    else if(id_btn != "AUTO")
65      control_type = REMOTE;
66    // it assigns a value to the decision
67    if (control_type == REMOTE){
```

```
68      if(id_btn == "FWD")
69        decision = FRONT;
70      else if(id_btn == "LFT")
71        decision = LEFT;
72      else if(id_btn == "STP")
73        decision = STOP;
74      else if(id_btn == "RGT")
75        decision = RIGHT;
76      else if(id_btn == "REV")
77        decision = BACK;
78    }
79    server.send(200, "OK"); // sends the OK
80  }
81
82  /*********************************************************
83   * getTypeControl - it returns the mode type
84   *********************************************************/
85  int getTypeControl(){
86    return control_type;
87  }
```

A.6 web_page

```
1   const char control_page[]  PROGMEM = R"html_file(
2   <!DOCTYPE html>
3   <html>
4       <script>
5           function control_btn(btn_id) {
6               var xhttp = new XMLHttpRequest(); // it creats a new http
        request
7               xhttp.open("GET", "control?control=" + btn_id, true); //
        exposes a function to be loaded when the server recieves the get request
8               xhttp.send(); // http://192.168.0.106/control?control=btn_id
9           }
10      </script>
11    <head>
12      <title>Controlo do robo</title>
13      <style>
14        .button {
15          height: 50px;
16          width: 120px;
```

```
17          border: none;
18          color: white;
19          text-align: center;
20          text-decoration: none;
21          display: inline-block;
22          font-size: 16px;
23          margin: 4px 2px;
24          cursor: pointer;
25          background-color: #017c72;
26        }
27        .button_quad {
28          height: 75px;
29          width: 75px;
30          border: none;
31          color: white;
32          text-align: center;
33          text-decoration: none;
34          display: inline-block;
35          font-size: 16px;
36          margin: 4px 2px;
37          cursor: pointer;
38          background-color: #4CAF50;
39        }
40        .button_STOP {
41          height: 75px;
42          width: 75px;
43          border: none;
44          color: white;
45          text-align: center;
46          text-decoration: none;
47          display: inline-block;
48          font-size: 16px;
49          margin: 4px 2px;
50          cursor: pointer;
51          background-color: #911000;
52        }
53        .right {
54            float: right;
55        }
56      </style>
57    </head>
58    <body>
59
60      <div style="text-align:center">
61        <h1>ROBOT P117</h1>
```

```
62
63        <button id="AUTO" onclick="control_btn(this.id)" class="button"> AUTO
      </button>
64
65        <br><br>
66
67        <h2>Controller</h2>
68        <button id="FWD" onclick="control_btn(this.id)" class = "button_quad"
      >FOARD</button>
69        <br><br><br><br>
70        <img hspace="10" style="padding-left: 5px">
71        <button id="LFT" onclick="control_btn(this.id)" class = "button_quad"
     >LEFT</button>
72        <img hspace="20" style="padding-left: 10px">
73        <button id="STP" onclick="control_btn(this.id)" class = "button_STOP"
     >STOP</button>
74        <img hspace="20" style="padding-left: 10px">
75        <button id="RGT" onclick="control_btn(this.id)" class = "button_quad"
     >RIGHT</button>
76        <img hspace="10" style="padding-left: 5px">
77        <br><br><br><br>
78        <button id="REV" onclick="control_btn(this.id)" class = "button_quad
     button_quad">BACK</button>
79
80      </div>
81    </body>
82  </html>)html_file";
```