FACULTAT
DE CIÈNCIES, TECNOLOGIA
I ENGINYERIES
UVIC | UVIC·UCC

Technical document

FOLLOW-ME ROBOT

Ferran Huéscar, Eloi Garcia, Jordi Rovira & Jana Bover

Degree in Mechatronics Engineering

Subject: Integrated Project II

Vic, June of 2023

Summary

Title:Follow-me robot

Authors: Ferran Huéscar, Eloi Garcia, Jordi Rovira i Jana Bover

Subject: Integrated Projects II

Date: Vic, June of 2023

Keywords: Autonomously, Assistance, Sensors, Wireless

The main idea of this project is to create an advanced follow-me robot that can autonomously track and follow a person, providing uninterrupted assistance and support. With the ability to detect and follow individuals using sophisticated sensors, our robot adapts to their movements and pace without the need for constant manual control. This allows users to focus on their primary objectives while the robot autonomously accompanies them in their activities.

In addition to its core tracking capability, the robot will be equipped to carry objects, alleviating the burden on individuals and further optimizing tasks. Whether it's retrieving tools, transporting goods, or providing on-the-go assistance, our robot will be a valuable companion in various professional and personal settings.

Prioritizing convenience and practicality, our follow-me robot will operate wirelessly with a long-lasting battery. This freedom from cables and power outlets ensures maximum flexibility and mobility, enabling the robot to move and follow people in a wide range of environments, including offices, warehouses, and even outdoor spaces.

Table of Contents

List of tables

List of Figures

1. Introduction / State of the art – Benchmarking

Follower robots have emerged as versatile devices equipped with sensors to detect and track individuals or objects. These intelligent robots can be programmed to navigate specific paths while maintaining a constant distance from their target. Their applications span across diverse fields, including surveillance, logistics, inspection, and assisting people with limited mobility.

As follower robots continue to gain popularity due to their interactive capabilities, there arises an exciting opportunity to design a robot that not only tracks a person in real-time but also assists in carrying heavy objects like boxes. Achieving this requires a combination of cutting-edge sensors, algorithms, motors, acceleration mechanisms, and sophisticated software to interpret sensor data and control the robot's movements efficiently. By undertaking such a project with meticulous planning and the right tools, the creation of a person-following, box-carrying robot can yield substantial rewards and hold immense potential for practical applications.

2. Objectives

The main goal of our project is to create an advanced follower robot that can autonomously track and follow a person, providing seamless assistance and support.

But also there are specific objectives of this assignment:

1. Design and build a robot that can follow a person autonomously.
2. Integrate sensors to allow the robot to detect and follow a person.
3. Develop a control system to allow the robot to follow a person smoothly and smoothly.
4. Add additional functionality to the robot, such as the ability to carry objects.
5. Wireless robot using a battery.

Finally, in the management section we aim to structure, follow and organize the following tasks:

		FEBRUARY															MARCH																							
TASKS	ASSIGN MENT	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Milestone 0 document	Group	🟥	🟥																																					
First design	Ferran		🟥	🟥																																				
Presentation milestone 0	Group	🟥	🟥	🟥																																				
Shopping list	Group				🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																									
Navigation research	Jordi	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																															
Motors calculations	Eloi			🟥	🟥	🟥	🟥	🟥	🟥																															
Electronics research	Jana			🟥	🟥	🟥	🟥	🟥	🟥																															
Preliminary design	Ferran																		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥														
Prototipe	Eloi																									🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥				
Hardware test	Jana													🟥	🟥	🟥	🟥	🟥																						
Final code	Jordi																					🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥									
Mid-term presentation	Group																																			🟥	🟥	🟥	🟥	🟥
Final design	Ferran																																						🟥	🟥
GitHub	Group	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	

Table 1. Planning of tasks throughout the project

3. Development of the project

Once we discussed the project objectives and what we were willing to achieve, we began searching for possible solutions and subsequently creating initial sketches and diagrams. Below is a detailed explanation of all the solutions obtained and descriptions of the designs: the mechanical design, the electrical design, and all software control. Finally, we present the tests performed until achieving proper functioning.

3.1. Analysis of the problem and possible solutions.

The main problem we aimed to solve was to create a robot that could follow a person or object autonomously, using sensors and control algorithms. After conducting some research, we identified some possible solutions to achieve this goal, which included:

1. Ultrasonic Sensors that can detect obstacles and measure distances using ultrasonic waves. By using these sensors, the robot could follow a person or object without colliding with other obstacles.

2. Infrared Sensors that can detect heat signatures and can be used to detect humans or animals, which would allow the robot to follow a person or pet.

3. Vision-Based Sensors that can detect visual cues, such as colour or shapes, and could be used to track a person or object based on these visual cues.

4. GPS and Navigation so the robot could follow a person or object autonomously without the need for sensors. However, this solution would only work in outdoor environments where GPS signals are available.

After analyzing these solutions, we decided to use a LIDAR. The LIDAR sensors would be used to detect obstacles and p t collisions and to track a person or object.

Additionally, we decided to use two wheels behind to provide the robot with mobility and maneuverability. The robot's dimensions were 200cm x 150cm, and it weights approximately 1.5kg. The robot is powered by a 12V DC motor and had a battery life of approximately 2 hours.

For the programming part, we used Python, which allow us to control de LIDAR. We also use Arduino to control the motors and other components of the robot.

Overall, the communication of the LIDAR, along with the two wheels drive system, allows to the robot to follow a person or object autonomously while avoiding obstacles.

3.2. Description of the design.

One of the key factors for the successful development of this robot has been the utilization of 3D printing as the primary manufacturing method. 3D printing is a technology that allows the creation of three-dimensional objects by layering materials, providing great versatility in terms of design and production. Thanks to this groundbreaking technique, we have been able to develop a robot with specific features and functionalities tailored for the transportation of small to medium-sized loads.

3D printing has also proven to be a highly efficient method in terms of costs and production times. By avoiding traditional manufacturing processes and associated expenses, we have been able to significantly accelerate the development of our robot, enabling rapid iterations in design and bringing the product to market more efficiently.

Firstly, taking into account the initial specifications, which include two drive wheels, two caster wheels, a platform for weight transportation, and the necessary capacity to accommodate all electronic components, we created the following preliminary layout.

Below is a development of the conceptual design of the robot.

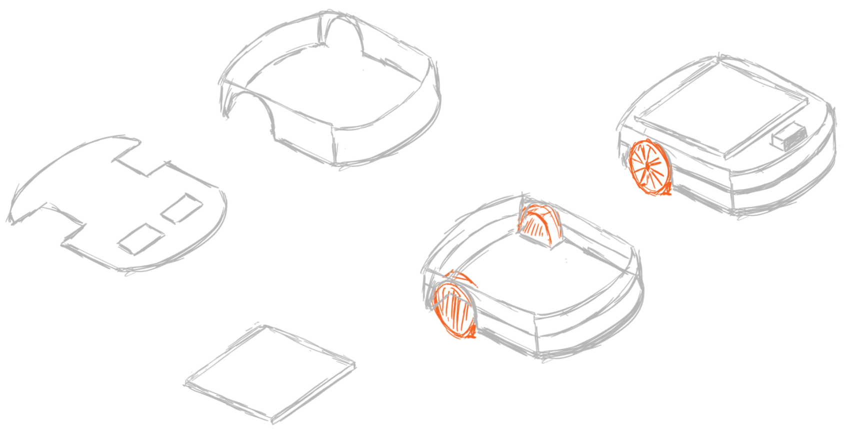


Fig 1. First conceptual design of the follow-me robot

Once the shape and aesthetics of the robot were determined, we decided to transfer it to the computer to begin deciding on the overall dimensions of the robot.

Therefore, we used the SketchUp program to create the first 3D schematic of our robot. Some changes were made to the second drawing, such as reducing the platform for weight loading since the LIDAR took up a significant portion of the upper platform of the robot. Additionally, we added a groove at the front of the robot. The resulting design is shown below:

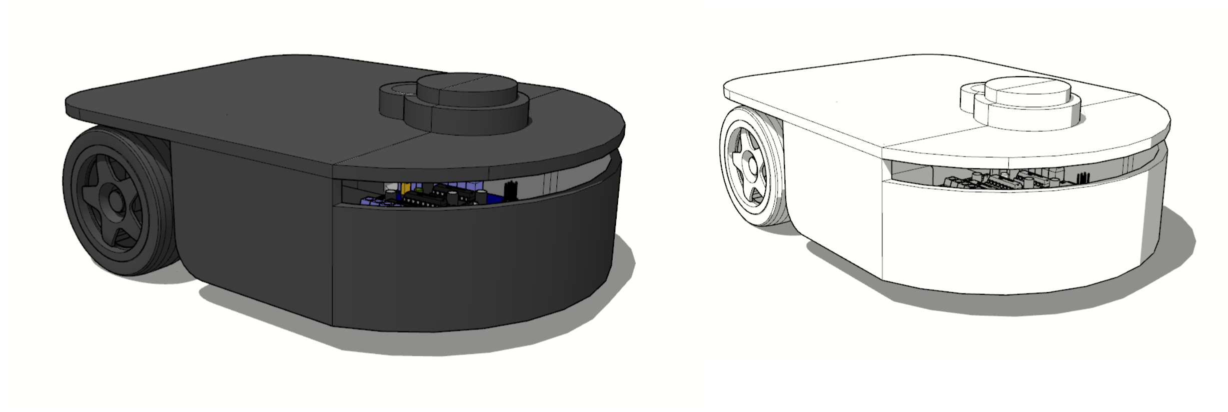


Fig 2&3. First 3D schematics of the robot with Sketchup

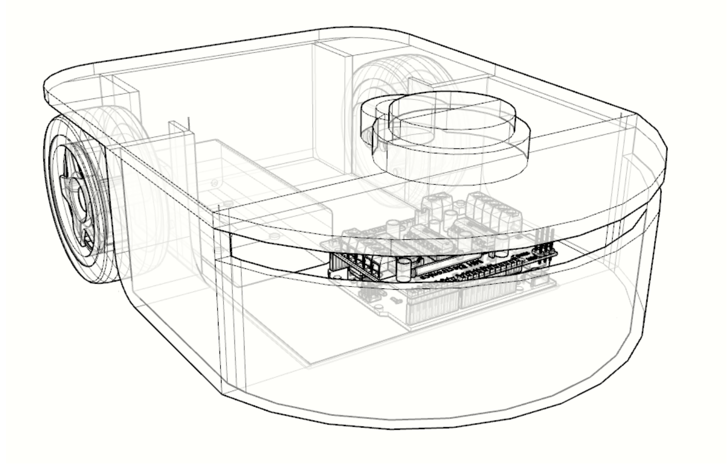


Fig 4. X-Ray view of the first 3D schematics

Finally, once we have determined the electronic components which motors go, the battery, the drivers, or the diameters of the wheels, we have been able to adapt the design based on the items purchased.

Some of the features that we have implemented in this latest design are supports for the lidar sensor, supports for the connection plate, joints between platforms, or holes to connect the battery externally.

To make the final result in 3D we have used the PTC Creo program since it is easy to use, versatile and all the members of the group can use it without accessibility problems. On the other hand, the final 2D plans have also been made with the same program, which has meant speed when generating them.

Below we show the final renders implemented with the PTC Creo program.

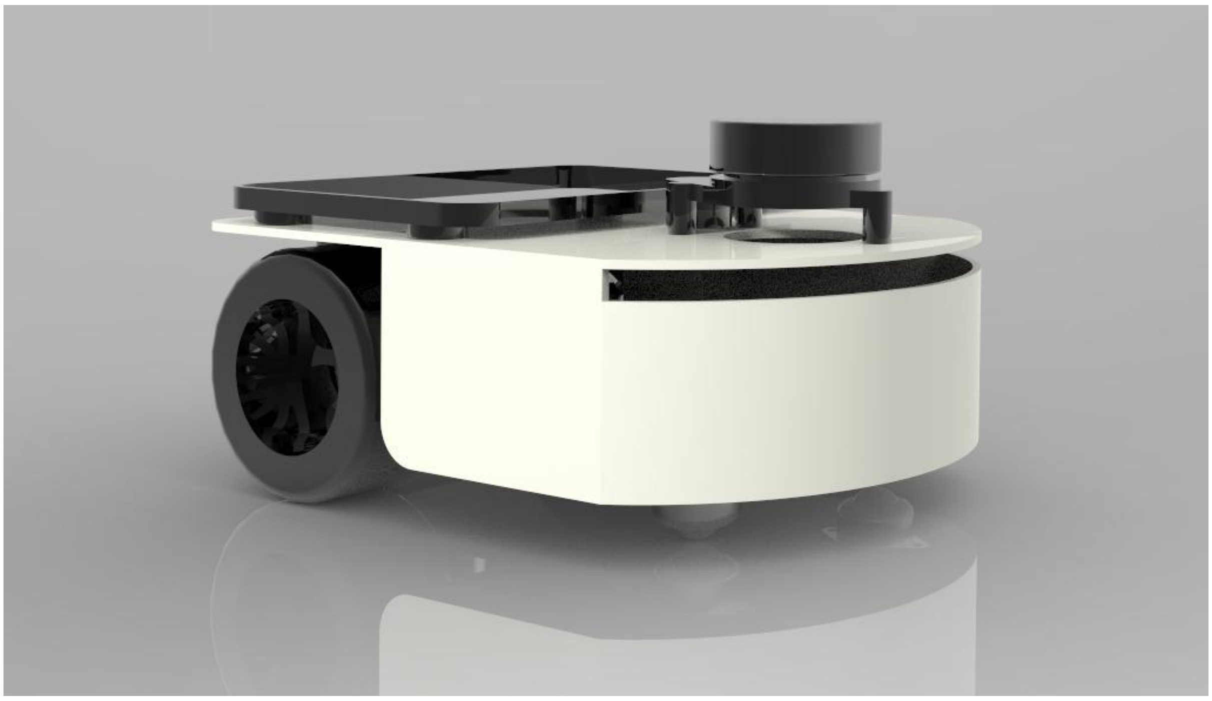


Fig 5. Render of the main view of the robot

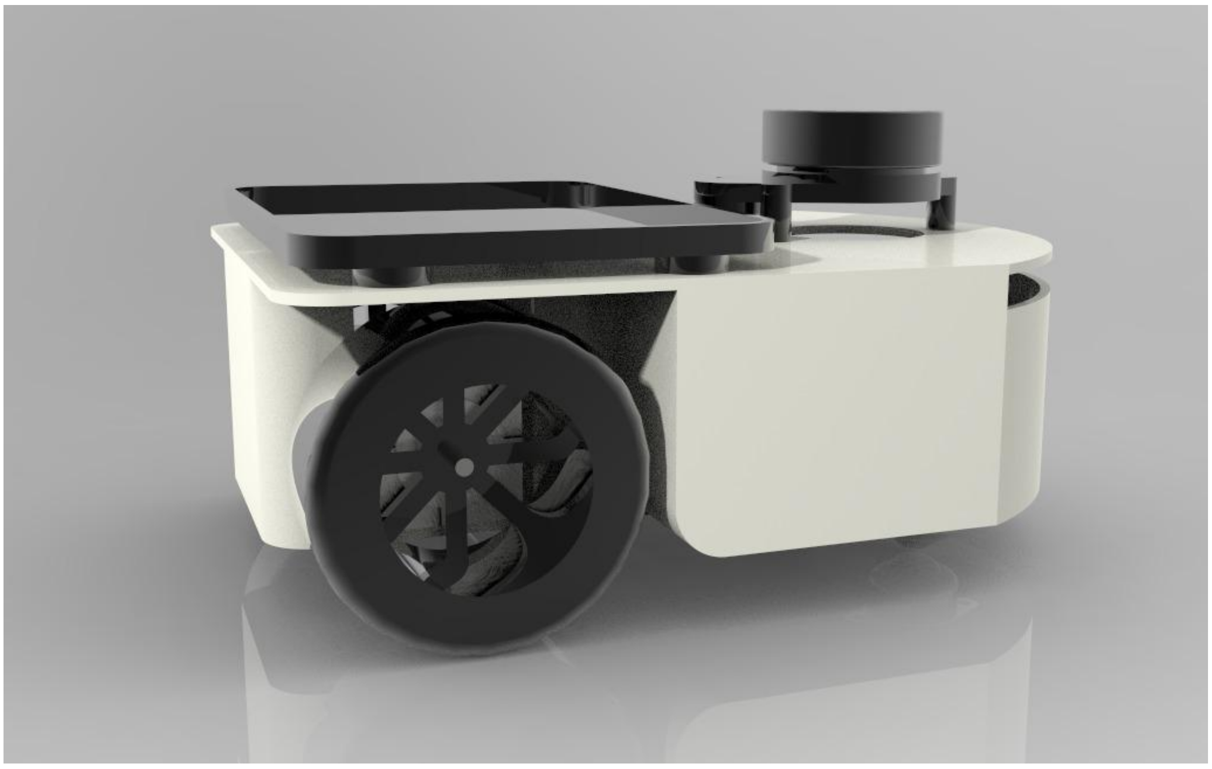


Fig 6. Render of robot profile view

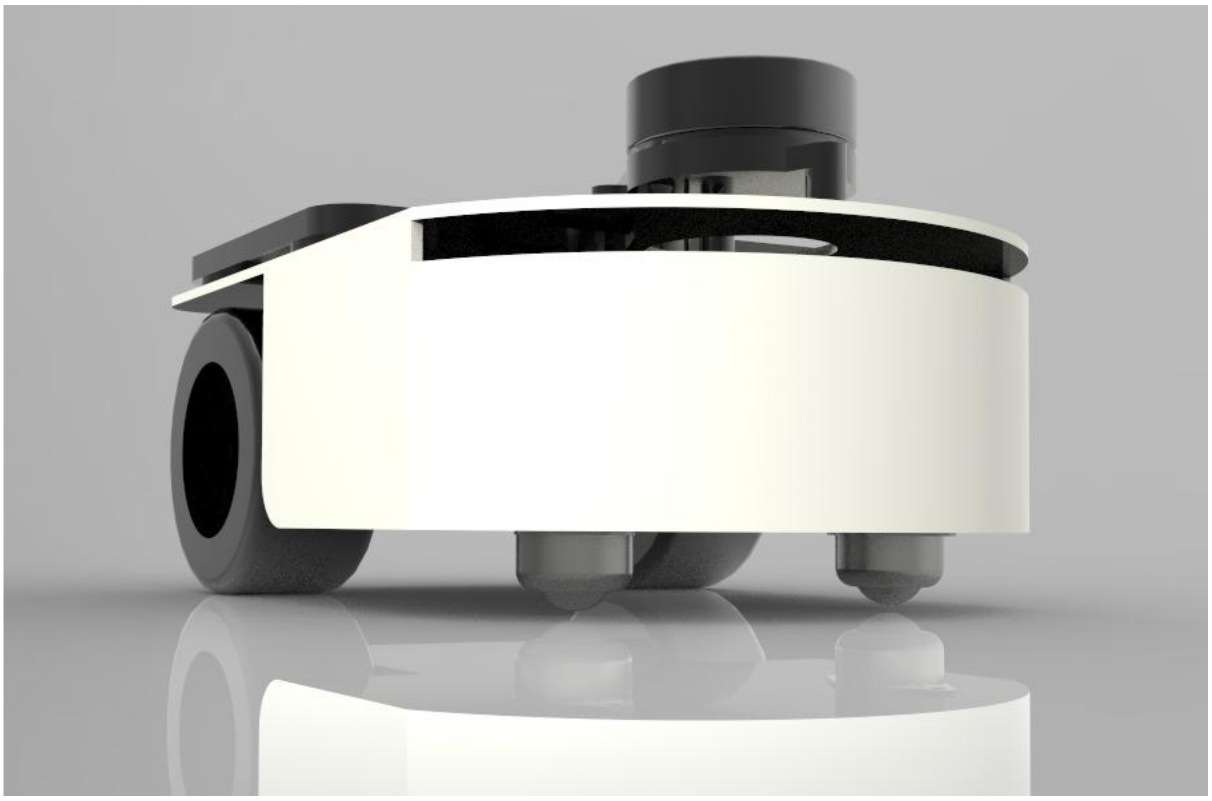


Fig 7. Render of robot front view

3.2.1. Mechanical design

In order to determine the mechanical specifications of the motors we studied all the factors included on the equation of making a follow-me robot run.

- Angular Velocity Calculation:

To determine the required angular velocity (ω) at which the motor shaft should rotate, we used the formula:

$$\mathbf{v = \omega * R}$$

where v is the desired linear velocity of the wheels and R is the radius of the wheels. Assuming the chosen wheels have a radius of approximately 35 mm, we can substitute this value into the formula to calculate the angular velocity.

- RPM Calculation:

To determine the minimum and maximum RPM (revolutions per minute) of the wheels, we need to convert the angular velocity from rad/s to RPM. The conversion formula is:

$$\mathbf{RPM = (\omega * 60) / (2\pi)}$$

By substituting the calculated angular velocity into this formula, we can find the RPM of the wheels. This information is useful for determining the specifications of the motors required.

- Motor Torque Calculation:

Motor torque is influenced by various factors, including the weight of the robot, the number of contact points touching the ground, and the friction of the surface. To calculate the total torque needed to start moving the robot from standstill for each wheel, we used the following formula:

Torque = (0.105 Nm * 4 wheels) / (2 driven)

This formula assumes that each wheel requires the same torque, and only two wheels are driven. By using this formula, we can determine the minimum torque that each motor should have.

- Power Calculation:

The power required by each wheel can be calculated using the following formula:

Power = (1.5 W * 4 wheels) / (2 driven)

This formula distributes the total power equally between the two driven wheels. By applying this formula, we can determine the minimum power rating that each motor should possess.

In summary, the motor calculations involve determining the angular velocity based on the wheel radius, calculating the RPM of the wheels, estimating the required torque based on the robot's weight, contact points, and friction, and finally, determining the minimum power rating for the motors. These calculations provide essential information for selecting the motors that are suitable for the given specifications and requirements of the robot.

3.2.2. Electrical design.

Below is the complete electrical schematic including the final PCB. It showcases all the research done for each component to ensure its proper functioning.

Our project includes an ESP32, two drivers to control the stepper motors, a LIDAR, a buck converter, and a battery supply involves several components and considerations.

The ESP32 is a popular microcontroller that can handle multiple tasks simultaneously and supports Wi-Fi and Bluetooth connectivity, It can be programmed using the Arduino IDE or other programming languages. In our project, to control the microcontroller we program it with Python.

The motor drivers are essential components for controlling the motors of the robot. They are responsible for providing the necessary power and control to the motors. There are various types of motor drivers available, such as H-bridge and L298N motor drivers. We use the DRV8825 driver.

LIDAR (Light Detection and Ranging) is a remote sensing technology that uses laser pulses to measure distance and create 3D maps of the environment. It can be used for obstacle avoidance and mapping of the surrounding area.

We also implement a buck converter, and it's a DC-DC converter that steps down the voltage from the battery supply to a lower voltage that is suitable for the microcontroller and other components. It is essential for providing a stable and regulated voltage to the microcontroller and other components.

The battery supply is another essential component for powering the robot. It is important to choose the right type of battery that can provide enough power for the motors and other components and has sufficient capacity for the desired operating time.

When designing the electrical system, it is important to carefully consider the power requirements of each component and choose appropriate voltage regulators and other components to ensure that each component receives the correct voltage and current. Proper wiring and component selection are also crucial to prevent shorts or other electrical problems; That is why for the components that need 12V, we used a bigger wire.

In summary, after researching and understanding the datasheets of all the components, we carefully made the pcb with the proper wires and the proper components.

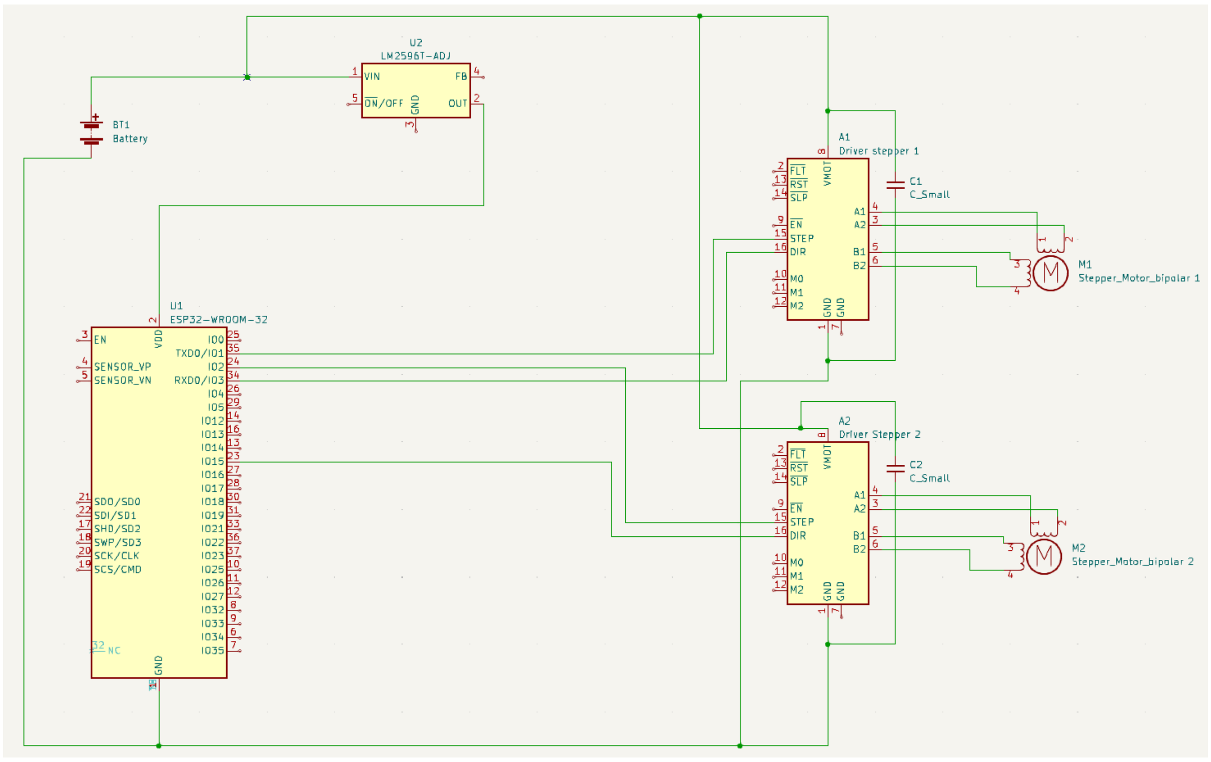


Fig 8. Electrical schematics for the connections design

3.2.3. Control and software.

The purpose of this section is to provide a comprehensive overview of the control and software aspects of a robotics project aimed at developing a robot that follows a user using a LiDAR sensor. The LiDAR data is decoded using Python, while the movement of the robot is controlled by two stepper motors, which are controlled via an Arduino microcontroller with an ESP32. This document will outline the system architecture, software components, and their functionalities.

- System Architecture

The control and software system for the LiDAR-based robot comprises the following components:

- LiDAR Sensor

Type: Triangulation lidar LD14P

The LiDAR sensor scans the environment and generates distance measurements, the raw data is sent to the Python program for decoding.

- Python Program

The program reads the raw byte stream from the LiDAR sensor and it decodes the byte stream according to the sensor's communication protocol and establishes a serial communication link with the Arduino microcontroller using the appropriate port.

It formats the decoded data and sends it as a serialized message via the serial connection. The decoded data represents distance measurements and angle to define a certain position from the LiDAR sensor.

Taking advantage of the possibilities offered by Python libraries, we utilize the decryption process to also display what the sensor sees.

Dependencies: pyserial, scipy, numpy, matplotlib

- Arduino with ESP32

The Arduino microcontroller, equipped with an ESP32 module, receives the serialized message from the Python program via the serial port and determines the appropriate movement commands based on the received data.

The program adjusts the motor movements to follow the nearest detected object.

3.3. Functional testing of the prototype

This section outlines the steps involved in functional testing and highlights any challenges or modifications encountered during the process.

Testing Procedure

- Data Decoding and Visualization

Initially, the LiDAR sensor data is decoded and visualized using Python and its libraries. The Python program reads the raw byte stream from the LiDAR sensor, decodes the data, and provides a graphical representation of the sensor readings. This step helps verify the proper functioning of the LiDAR sensor and the accuracy of the decoding process.

- Integration with ESP32 via Serial Communication

To streamline the control process and leverage the existing Python code, a decision was made to communicate the LiDAR sensor with the ESP32 microcontroller via the serial port from a computer. This required adapting an FTDI (USB-to-Serial) module to establish the serial communication between the computer and the LiDAR sensor.

- Challenges and Modifications

During the functional testing of the prototype, the following challenges were encountered:

- LiDAR Sensor Board Failure

In the initial stages of testing, the board that came with the LiDAR sensor suffered a failure. To continue with the project, an alternative solution was devised by adapting an FTDI module for serial communication. This modification enabled the team to proceed with the programming and testing of the robot.

- Limited Time for ESP32 Dual-Core Programming

Given the project timeline, it was not possible to fully utilize the dual-core capability of the ESP32 microcontroller. It was initially intended to program the microcontroller to operate with two loops simultaneously, utilizing the two cores for improved wheel movement during turns. Unfortunately, due to time constraints, this enhancement could not be implemented. However, it remains a potential improvement for future iterations of the project.

4. Cost of the project

Calculations of the costs of the project:

Below is a graph of the hours spent on the project based on the work sector. And the detail of the sections is:

- Mechanical: design, calculations, drawings, 3D printer, assembly, manufacturing processes, list of mechanical materials.
- Electrical: electronic design, electrical schematics, electronic components test, assembly, list of electronic materials.
- Programming: code, control system, software tests.
- Management: meetings, plannings, milestones, presentations, final documentation, video, GitHub, website.

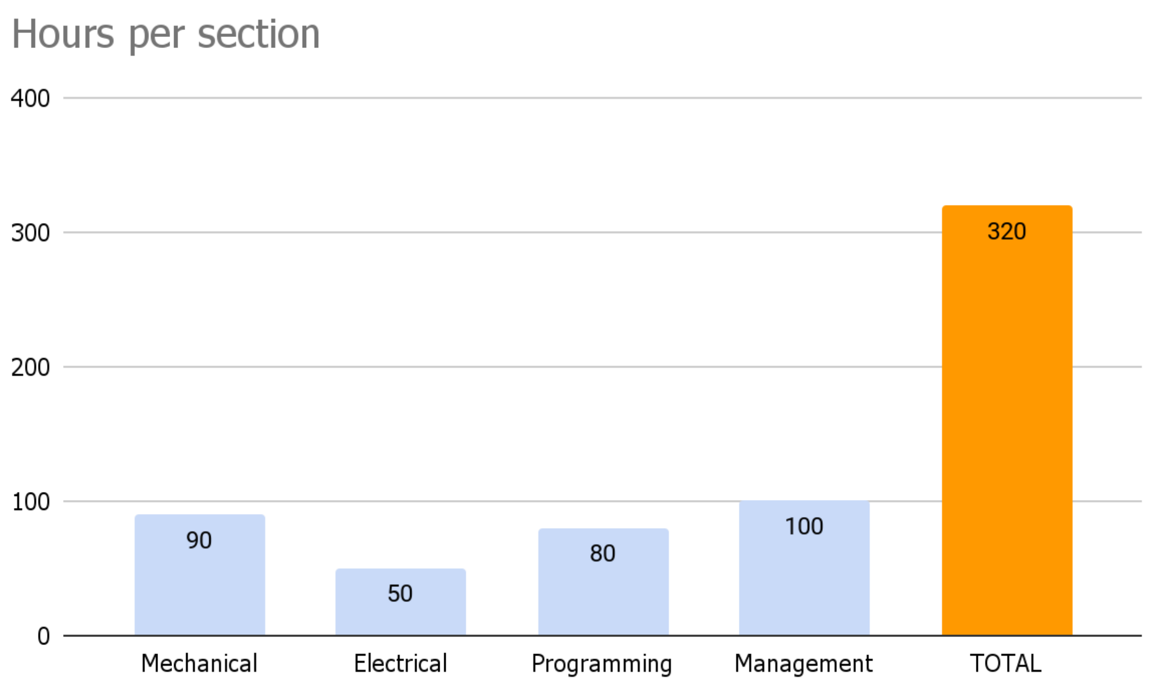


Table 3. Hours per section to calculate the final budget

On the other hand, the hours spent by each member of the group are difficult to define, even so we have distributed the work equally, therefore it is approximately 150 hours for each one.

Finally, to calculate the total budget of the project we have taken into account all the materials from the shopping list and the man-hours price given a standard engineer hour price.

Below is a table of all prices and the total budget:

Description	Quantity	Unit Price	Amount
Stepper motor	2	20,99€	41,98€
ESP32	1	11,99€	11,99€
Pack 5 Converter	1	10,99€	10,99€
Laser Lidar	1	86,9€	86,9€
Pack 5 Drivers	1	25,99€	25,99€
Pack 2 free wheels	1	8€	8€
Pack 2 Wheels	1	21,79€	21,79€
Battery	1	23,5€	23,5€
Battery connector	1	3,99€	3,99€
Bolts (M3x8mm)	10	0€	0€
Bolts (D2.2x1cm)	4	0€	0€
SUBTOTAL			265,38€
Mechanical hours	90	15€	1350€
Electrical hours	50	15€	750€
Programming hours	80	15€	1200€
Management hours	100	15€	1500€
SUBTOTAL			4800€
TOTAL			5065,38€

Table 2. Table of all prices and the total budget of the project

5. Conclusions

In conclusion, the project aims to create an advanced follower robot that can autonomously track and follow a person, while also having the ability to carry objects. The project involves several components, including mechanical design, electrical design, control and software development.

The mechanical design utilizes 3D printing to create the robot, allowing for customization and rapid iterations in design. And the electrical design includes components such as an ESP32 microcontroller, motor drivers, LIDAR sensor, buck converter, and battery supply.

The control and software development involve programming the ESP32 with Python, decoding the data from the LIDAR sensor, and controlling the robot's movement based on the received data.

The project follows a structured development process with tasks planned and assigned to team members.It also includes cost analysis, with tables indicating the budget and hours allocated for each section. Functional testing of the prototype is conducted to ensure proper functioning and address any challenges or modifications.

Overall, the project demonstrates a comprehensive approach to designing and developing a follower robot with advanced capabilities. The combination of mechanical design, electrical components, and software control showcases a systematic and well-structured approach to achieving the project's objectives.

6. Bibliography

Bibliography must be numbered. All bibliography entries must be cited in the main text of the document.

1. Website link of Stepper motors:

https://www.amazon.es/Creality-Stepper-Impresora-Grados-Ender-3/dp/B091D37BM2/ref=sr_1_6?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=3BOATU7NHV788&keywords=motor+stepper+0.3+Nm&qid=1677768062&sprefix=motor+stepper+0.3+nm%2Caps%2C109&sr=8-6

2. Website link of ESP32:

https://www.amazon.es/HiLetgo-ESP-wroom-32-Desarrollo-procesador-microcontrolador/dp/B0718T232Z/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=9RVQ223A4XKW&keywords=esp32&qid=1677771664&sprefix=esp32%2Caps%2C104&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

3. Website link of Converter:

https://www.amazon.es/Yizhet-Convertidor-Regulador-Ajustable-Alimentaci%C3%B3n/dp/B0823P6PW6/ref=mp_s_a_1_5?crid=3AT2SIKO8O52N&keywords=regulador+buck+arduino&qid=1677769183&sprefix=regulador+buck+arduino%2Caps%2C94&sr=8-5

4. Website link of Laser Lidar:

https://botland.store/laser-scanners-lidar/22016-lidar-ld14-360-degree-laser-scanner-8m-range-waveshare-22415.html

5. Website link of Drivers

https://www.amazon.es/WJMY-Controlador-Stepper-Disipador-Machines/dp/B08R64Y7V7/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2GAYE59PGVMAS&keywords=driver+a4988&qid=1677771455&s=industrial&sprefix=driver+a4988%2Cindustrial%2C141&sr=1-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

6. Website link of Free wheels

https://fadisel.com/ca/roda-boja-o-roda-robotica/1055-roda-boja-acer-o-25-mm.html

7. Website link of Motor wheels

https://www.amazon.es/Dilwe-Neum%C3%A1ticos-Coche-Rueda-Pl%C3%A1stico/dp/B07DZS4T6J/ref=sr_1_4?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=219KN94OJQ348&keywords=neum%C3%A1tico+de+goma+rc+90mm&qid=1680867321&s=toys&sprefix=neumatico+de+goma+rc+90mm%2Ctoys%2C86&sr=1-4

8. Website link of Battery

https://www.amazon.es/OVONIC-bater%C3%ADa-Paquetes-2200mAh-Evader/dp/B07LFS8ZL6/ref=pd_lpo_2?pd_rd_w=Yt2Us&content-id=amzn1.sym.d229992b-4342-448b-b0e5-b51a45434c54&pf_rd_p=d229992b-4342-448b-b0e5-b51a45434c54&pf_rd_r=CZDJ0SFPD0V45FAPBGV0&pd_rd_wg=PJnMN&pd_rd_r=9b209e05-6037-44a2-8e7b-06f2960ba08c&pd_rd_i=B07LFS8ZL6&psc=1

9. Website link of Battery connector

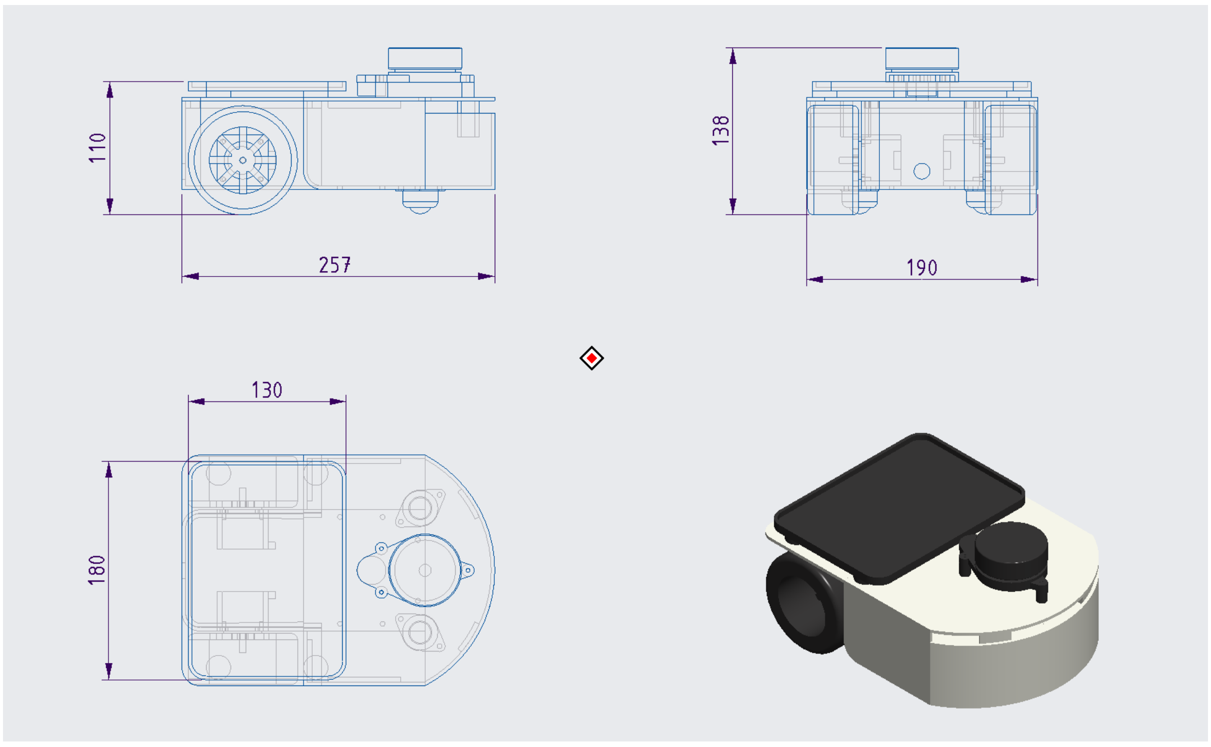
https://www.amazon.es/Conectores-bater%C3%ADa-Battery-Vehicle-Hembra/dp/B074RGT76J/ref=sr_1_2_sspa?keywords=conector+xt60&qid=1677772424&sr=8-2-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

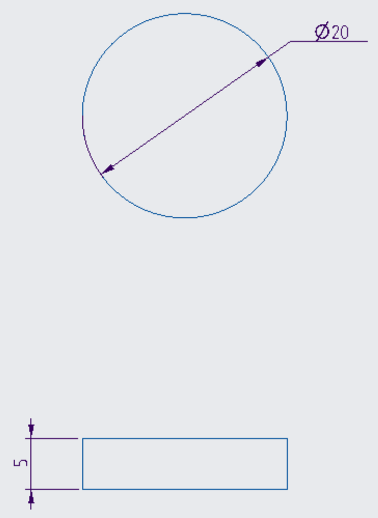
10. Instructables web

https://www.instructables.com/Follow-Me-Robot/

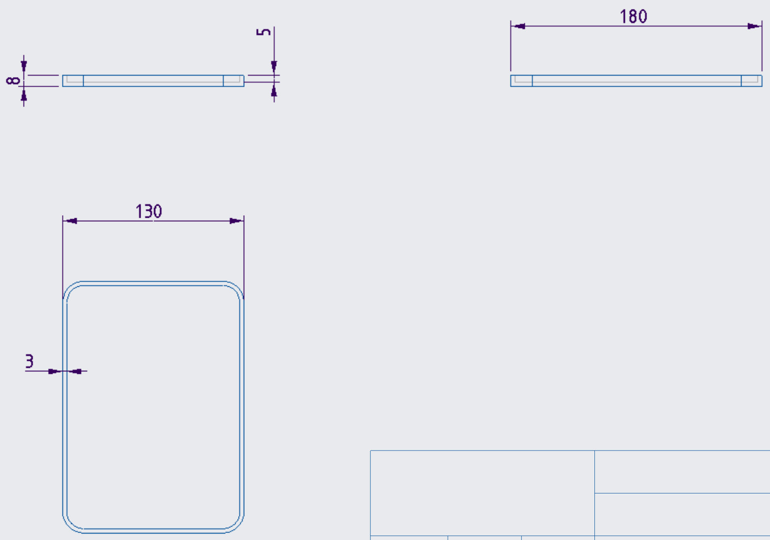
7. Annexes

2D PTC CREO drawings

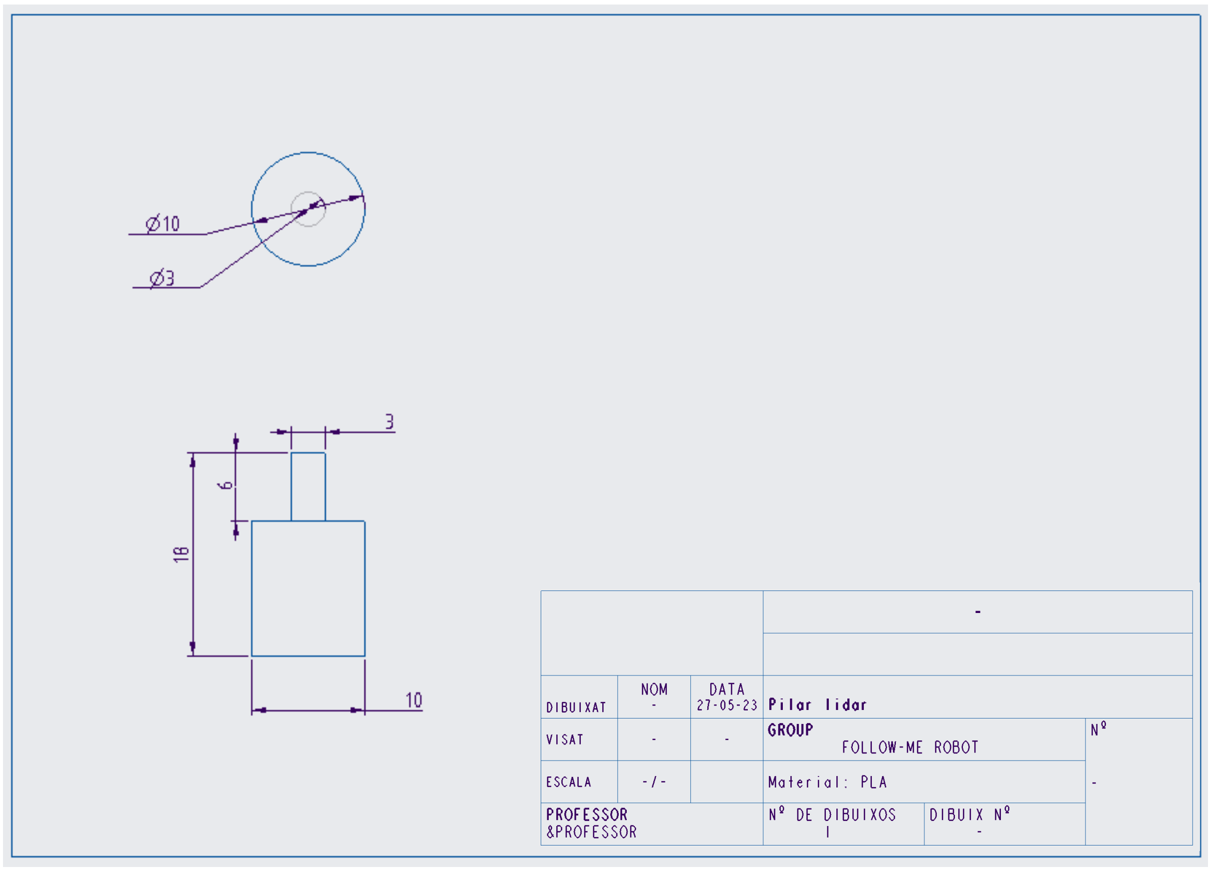




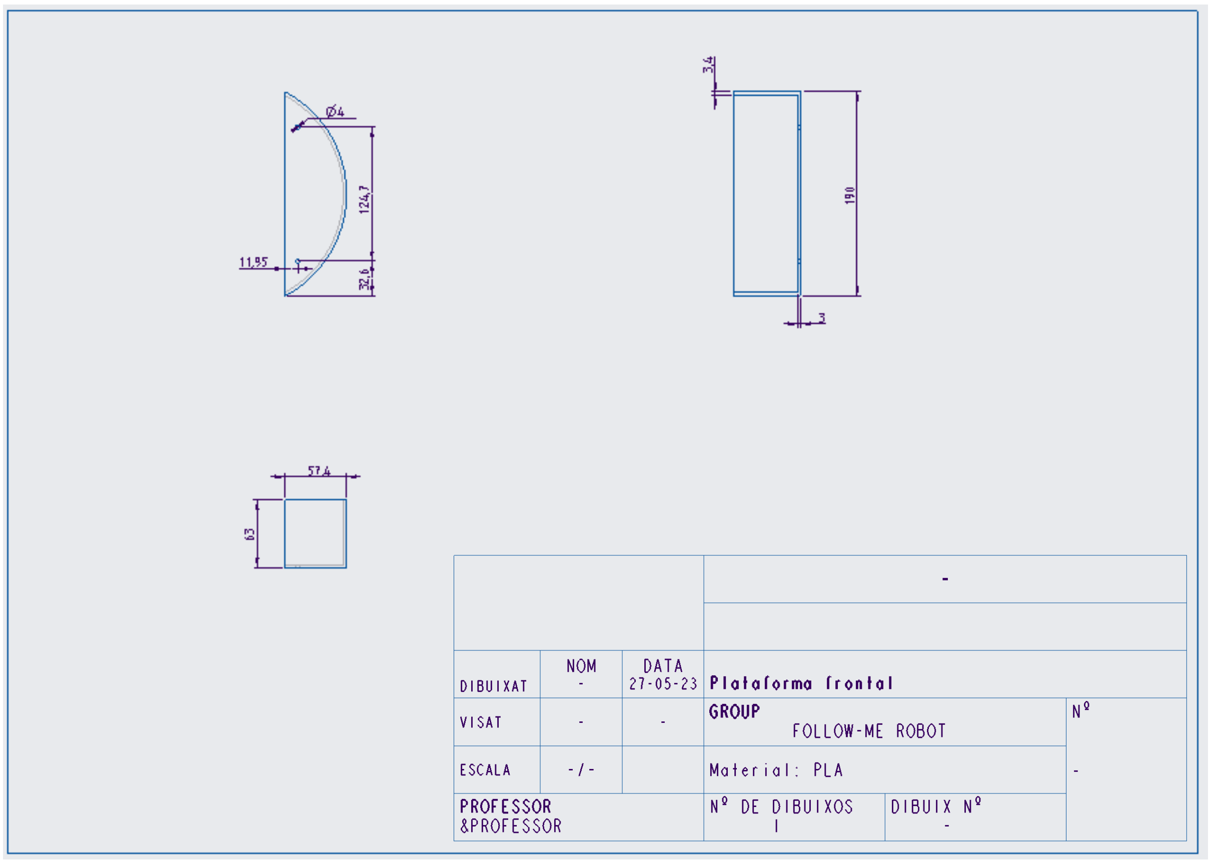
			-		
DIBUIXAT	NOM -	DATA 27-05-23	Pilar safata		
VISAT	-	-	GROUP FOLLOW-ME ROBOT		Nº
ESCALA	-/-		Material: PLA		-
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS I	DIBUIX Nº -	



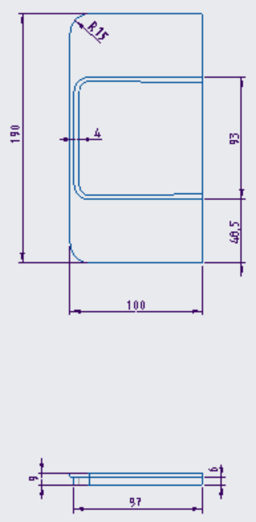
			-		
DIBUIXAT	NOM -	DATA 27-05-23	Safata		
VISAT	-	-	GROUP FOLLOW-ME ROBOT		Nº
ESCALA	-/-		Material: PLA		-
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS I	DIBUIX Nº -	



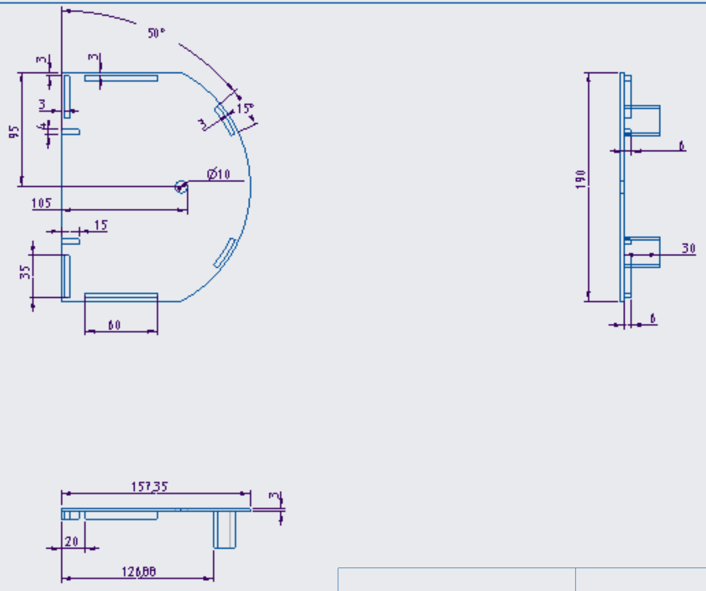
			-	
DIBUIXAT	NOM -	DATA 27-05-23	Pilar lidar	
VISAT	-	-	GROUP FOLLOW-ME ROBOT	Nº
ESCALA	-/-		Material: PLA	-
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS I	DIBUIX Nº -



			-	
DIBUIXAT	NOM -	DATA 27-05-23	Plataforma frontal	
VISAT	-	-	GROUP FOLLOW-ME ROBOT	Nº
ESCALA	-/-		Material: PLA	-
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS I	DIBUIX Nº -



			-		
DIBUIXAT	NOM -	DATA 27-05-23	**Tapa frontal**		
VISAT	-	-	**GROUP** FOLLOW-ME ROBOT		Nº -
ESCALA	-/-		Material: PLA		
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS 1	DIBUIX Nº -	



			-		
DIBUIXAT	NOM -	DATA 27-05-23	**Tapa lidar**		
VISAT	-	-	**GROUP** FOLLOW-ME ROBOT		Nº -
ESCALA	-/-		Material: PLA		
PROFESSOR &PROFESSOR			Nº DE DIBUIXOS 1	DIBUIX Nº -	

Python program

```
from pylab import *
import serial, time

ion()

com = 'COM7'



class DataFrame:#With this class I decode the LiDAR info

    def __init__(self, verlen, speed, start_angle,points, end_angle, crc):
        self.verlen = verlen #Value set to 0x2C(1Byte) packet type (I don't use it)
        self.speed = speed #Rotation speed (2Byte)
        self.start_angle = start_angle #Start angle of the data packet (2Bytes) 0.01degree
        self.end_angle = end_angle #Angle of the end of the data packet (2Bytes) 0.01degree
        self.points = points #DATA (3Bytes)
        self.crc = crc #crc (I don't use it)

#int.from_bytes: Create an integer from bytes

    @classmethod
    def from_bytes(cls,buffer):#Here I create the data buffer
        verlen = buffer[0] #I don't use it
        speed = int.from_bytes(buffer[1:3], byteorder='little') #Speed (2Byte 2nd and 3rd)
         start_angle = int.from_bytes(buffer[3:5], byteorder='little')*0.01 #Start angle (2Byte 3rd and
4rd)
        i = 5 #I do this to index the values in the buffer skipping the previous ones
        points = [] #Here I add the distance and intensity values
        for x in range(12):
            dist = int.from_bytes(buffer[i:i+2], byteorder='little') #2 bytes of distance (5th and 6th)
            i += 2
            intensity = int.from_bytes(buffer[i:i+1], byteorder='little') #1 Byte of intensity (7th)
            i += 1
            points.append((dist,intensity)) #I add them to the 'points' list


        end_angle = int.from_bytes(buffer[i:i+2], byteorder='little')*0.01 #Final angle (8th)
        i += 1

        crc = int.from_bytes(buffer[i:i+1], byteorder='little') #I don't use it

        return cls(verlen, speed, start_angle, points, end_angle, crc)


dt = 35 # +- dt (angle)

def process_data(tt,dd): #This function restricts the range of angles

    #return array([0]),array([0])

    i  = logical_and(tt>(180-dt),tt<(180+dt)) #The range of angles is from 145° to 215°

    tt = tt[i] #theta
    dd = dd[i] #distance
    i = nonzero(dd) #Filter to remove errors (value 0)
    tt = tt[i] #theta
    dd = dd[i] #distance

    if tt.size<10:
        return tt,dd #Filter if I have removed most items

    i = argsort(dd) #Sort the values from smallest to largest
    tt = tt[i] #theta
    dd = dd[i] #distance

    N = 6
    ttav = sum(tt[:N])/N
    ddav= sum(dd[:N])/N

    #Calculate the byte buffer of the message
    buff = int(ttav*100).to_bytes(2,"little")
    buff += int(ddav).to_bytes(2,"little")
    arduino.write(buff)
    arduino.flush()

    #print ('angle ',ttav,'distancia ', ddav)

    return tt[:N],dd[:N]



#Read the serial port of the LiDAR
ser = serial.Serial(com,
                    baudrate=115200,
                    parity=serial.PARITY_NONE,
                    stopbits=1,
                    bytesize=8,
                    xonxoff=0, rtscts=0)

#Serial port of the esp32
arduino = serial.Serial("COM6", 115200)

time.sleep(2)





fig = figure() #Graph the position (This part is for the tests)
ax = fig.add_subplot(111)

tt = array([], dtype=float)
dd = array([], dtype=float)
xx = array([0])
yy = array([0])

line, = ax.plot(xx,yy,'k.', markersize=1)
line_data, = ax.plot(xx,yy,'r.', markersize=5)
ax.plot([-4000*cos(dt*pi/180),0,-4000*cos(-dt*pi/180)],[-4000*sin(dt*pi/180),0,-4000*sin(-dt*pi
/180)])
ax.set_xlim(-4000,4000)
ax.set_ylim(-4000,4000)

while True:

    nbytes = arduino.inWaiting()#read arduino serial port
    if nbytes:#if there is something follow
        txt = arduino.read(nbytes)
        print(txt)#.decode("ascii"), end='') #Decodific ascii

    # wait for start of frame (x54)
    b = ser.read(1) #Read LiDAR serial port (1 byte)
    if b!= b'\x54':
        continue

    # read frame
    buffer = ser.read(46) #46 which is the length of the message

    # decode frame.
    frame = DataFrame.from_bytes(buffer) #the buffer is the 46 bytes of the message

     if frame.end_angle < frame.start_angle: #Here it corrects the error caused when a string of
values passes through 0°
        frame.end_angle += 360


     theta = linspace(frame.start_angle, frame.end_angle, 12)#12 points in each dataframe (creates
a vector of angles)
    pts = array(frame.points, dtype=float)
    dist = pts[:,0]


    # Concatenate data making sure we don't loose the information
    # about when new scan starts.
    if tt.size>0 and tt[-1]>theta[0]:
        theta += 360
    tt = concatenate((tt,theta))
    dd = concatenate((dd,dist))

    # check if full scan acquired.
    if tt[-1] >= 360:
        i = tt<360
        fulltt = tt[i]
        fulldd = dd[i]

        i = tt>=360
        tt = tt[i]-360
        dd = dd[i]

    # process tt and dd in range +-35degrees from 180

        ptt,pdd = process_data(fulltt,fulldd)

        xx = fulldd*cos(-fulltt*pi/180)
        yy = fulldd*sin(-fulltt*pi/180)
        line.set_xdata(xx)
        line.set_ydata(yy)

        xx = pdd*cos(-ptt*pi/180)
        yy = pdd*sin(-ptt*pi/180)
        line_data.set_xdata(xx)
        line_data.set_ydata(yy)

        fig.canvas.draw()
        fig.canvas.flush_events()
```

Arduino program

```
const int BUFFER_SIZE = 4; //for the received data
char buf[BUFFER_SIZE];


const int DIR_MOTOR1 = 5; //pins for the stepper motors
const int STEP_MOTOR1 = 18;
const int DIR_MOTOR2 = 0;
const int STEP_MOTOR2 = 4;

unsigned int dist; //received data distance

float theta; //received data angle

int pulse1 = 0; //that variable is to control the amount of steps each motor do on a turn

unsigned long previousMillis1 = 0;  //will store last time
unsigned long currentMillis1 = 0;   //will store current time

unsigned long previousMillis2 = 0;  //will store last time
unsigned long currentMillis2 = 0;   //will store current time



void setup() {
  Serial.begin(115200);
  Serial.setTimeout(10);
  //Motor 1
  pinMode(STEP_MOTOR1, OUTPUT); //step motor1 and his direction
  pinMode(DIR_MOTOR1, OUTPUT);

  //Motor 2
  pinMode(STEP_MOTOR2, OUTPUT);//step motor2 and his direction
  pinMode(DIR_MOTOR2, OUTPUT);

  digitalWrite(DIR_MOTOR1, HIGH);
  digitalWrite(DIR_MOTOR2, LOW);
}


unsigned long stepper_time = 0;
//unsigned long stepper_time1 = 0;
//unsigned long stepper_time2 = 0;
float stepper_freq = 0;
//float stepper_freq_1 =0;
//float stepper_freq_2 =0;
```

```
void loop() {

  if (Serial.available() > 0) {

    int rlen = Serial.readBytes(buf, BUFFER_SIZE); //read bytes from serial port
    if (rlen != BUFFER_SIZE) {
      return;
    }

    // prints the received data
    //Serial.print("I received: ");
    int th = buf[0] + (buf[1] << 8); //0 - 1 for distance
    dist = buf[2] + (buf[3] << 8); // 2 - 3 for angle

    theta = ((float)th) / 100; //scale the angle units

    //Serial.print(dist);
  }

  long set_pt = 0; //that's for the proportional speed - distance
  long err = dist - set_pt;
  if (err < 0) {
    stepper_freq = 0;
  } else if (err > 2000) {
    stepper_freq = 0;
  } else {
    stepper_freq = (float)err / 2000000.0;
  }

  stepper_freq = stepper_freq < 0.3e-3 ? 0 : stepper_freq; // below 0.3e-3 stepper_freq = 0
   stepper_freq = stepper_freq > 0.7e-3 ? 0.7e-3 : stepper_freq; //over 0.7e-3 stepper freq =
0.7e-3

  stepper_time = 1.0 / stepper_freq;

  if (stepper_freq > 0.0) {

//Here are the motor controls,
//it can be seen that there is an offset, so the center of the sensor is not 180°,
//and thats why the left and right turns are 4 degrees out of 180° +- 10°

    if (theta < 179.0) { //turn left

      currentMillis1 = micros();
      if (currentMillis1 - previousMillis1 >= stepper_time) {
        digitalWrite(STEP_MOTOR2, HIGH);
        pulse1 += 1;//to do the turn i restrict the half of the steps from one motor
        if (pulse1 >= 2) {
          digitalWrite(STEP_MOTOR1, HIGH);
        }
        previousMillis1 = currentMillis1;
      }

      currentMillis2 = micros();
      if (currentMillis2 - previousMillis2 >= stepper_time*2) {
        digitalWrite(STEP_MOTOR2, LOW);
        if (pulse1 >= 3) {
          digitalWrite(STEP_MOTOR1, LOW);
          pulse1 = 0;
        }
        previousMillis2 = currentMillis2;


      }
    }

    else if (theta > 189.0) { //turn right
       currentMillis1 = micros();
      if (currentMillis1 - previousMillis1 >= stepper_time) {
        digitalWrite(STEP_MOTOR1, HIGH);
        pulse1 += 1;//to do the turn i restrict the half of the steps from one motor
        if (pulse1 >= 2) {
          digitalWrite(STEP_MOTOR2, HIGH);
        }
        previousMillis1 = currentMillis1;
      }

      currentMillis2 = micros();
      if (currentMillis2 - previousMillis2 >= stepper_time*2) {
        digitalWrite(STEP_MOTOR1, LOW);
        if (pulse1 >= 3) {
          digitalWrite(STEP_MOTOR2, LOW);
          pulse1 = 0;
        }
        previousMillis2 = currentMillis2;

      }
    }


    else { //go straight
      currentMillis1 = micros();
      if (currentMillis1 - previousMillis1 >= stepper_time) {
        digitalWrite(STEP_MOTOR1, HIGH);
        digitalWrite(STEP_MOTOR2, HIGH);
        previousMillis1 = currentMillis1;
      }

      currentMillis2 = micros();
      if (currentMillis2 - previousMillis2 >= stepper_time*2) {
        digitalWrite(STEP_MOTOR1, LOW);
        digitalWrite(STEP_MOTOR2, LOW);
        previousMillis2 = currentMillis2;
      }
    }

    //digitalWrite(STEP_MOTOR1, HIGH);
    //digitalWrite(STEP_MOTOR2, HIGH);
    //delayMicroseconds(stepper_time);

    //digitalWrite(STEP_MOTOR1, LOW);
    //digitalWrite(STEP_MOTOR2, LOW);
    //delayMicroseconds(stepper_time);
  }
}
```