Build a driver for a mobile robot Vector

Second year internship report
FISE 2024

University of Oviedo
Gijon-Spain

Martin PILON
ENSTA Bretagne - Autonomous Robotics
martin.pilon@ensta-bretagne.org

Supervised by

Prof. Juan Carlos ALVEREZ ALVEREZ
Universidad de Oviedo - Department Research Group on Multisensor Systems and Robotics
juan@uniovi.es

Acknowledgements

I would like to thank the University of Oviedo and especially Prof. Juan Carlos ALVAREZ ALVAREZ, head of the Department Research Group on Multisensor Systems and Robotics, for accepting my internship in this lab.

I am extremely grateful to Prof. Juan Carlos ALVAREZ ALVAREZ, for giving me this opportunity and welcoming me into the laboratory. He was very attentive and available throughout this internship, gave me lots of advice, which made this internship a very good experience and participated to my choice of professional orientations.

I express my deepest thank to ENSTA Bretagne for allowing students to perform this type of internship and training.

Abstract

As part of my engineering training, I did a 16-week engineering assistant internship from 28/04/23 to 21/08/23 at the Research Group on Multisensor Systems and Robotics department in the University of Oviedo. Supervised by Prof. Juan Carlos ALVAREZ ALVAREZ. I was given a mobile robot, Vector, to improve. To do this, global objectives were set for me.

- Create a driver to improve program structure
- Use of ROS 2 (Robot Operating System) to implement wireless communication

In the first month I studied the hardware of the mobile robot, to understand how communication was carried out between the various components so that I could implement it in the driver. During the next month, after understanding, we put it into practice and created the first driver to enable communication between a remote PC (Personal Computer) and the mobile robot's motors. Then a test node was added to test the driver. Finally, a LIDAR (Laser Imaging Detection And Ranging) sensor was subsequently added. The driver structure and the node test had to be modified to accept the addition of this LIDAR

Résumé

Dans le cadre de ma formation d'ingénieur, j'ai effectué un stage d'assistant ingénieur de 16 semaines du 28/04/23 au 21/08/23 au sein du groupe de recherche sur les systèmes multi capteurs et du département de robotique de l'Université d'Oviedo. Ce stage était supervisé par le professeur Juan Carlos ALVAREZ ALVAREZ. On m'a donné un robot mobile, Vector, à améliorer. Pour ce faire, des objectifs globaux m'ont été fixés.

- Création d'un driver pour améliorer la structure du programme
- Utilisation de ROS 2 (Robot Operating System) pour implémenter la communication sans fil.

Le premier mois, j'ai étudié le hardware du robot mobile, afin de comprendre comment la communication s'effectuait entre les différents composants pour pouvoir l'implémenter dans le driver. Le mois suivant, nous avons mis en pratique et créé le premier pilote pour permettre la communication entre un PC distant et les moteurs du robot mobile. Ensuite, un nœud de test a été ajouté pour tester le pilote. Enfin, un capteur LIDAR a été ajouté. La structure du pilote et le test du nœud ont dû être modifiés pour accepter l'ajout de ce LIDAR

Keywords

Mobile robot, Driver, ROS, LIDAR

Table of contents

Table of illustration

Table of Appendice

Abbreviation list

ENSTA Bretagne = Ecole Nationale Supérieure de Techniques Avancées Bretagne
ENSMM = Ecole Nationale Supérieure de Mécanique et des Microtechniques
ROS = Robot Operating System
PC = Personal Computer
LIDAR = Laser Imaging Detection And Ranging
CPR = Counts Per Revolution
LPR = Lines Per Revolution
DIP Switch = Dual In-line Package Switch
PID = Proportional Integral Derivative

1- Introduction

1-1 Organisation of the structure

The Robotics Laboratory in the Electrical, Electronics, Computer and Automation Engineering department of Oviedo University welcomes every year many students to work on high-tech robots or to carry out their own projects. For example, there are industrial robots such as the ABB-IRB 120 robot but also mobile robots. It is also one of the leading robotics laboratories in the region in terms of projects carried out in collaboration with companies such as MoviRobots.

1-2 Presentation of the internship subject

My internship topic is based on the second year internship of the year 2018-2019 of Pierre PICHOU, a student in ENSMM (Ecole Nationale Supérieure de Mécanique et des Microtechniques).

His objective was to build a mobile robot: Vector. The goal of Vector is having autonomous navigation in the natural environment. He is inspired by the end of career project of Olga CORDERO MORALES, professor at the University of Málaga (Spain). This project is focused on the development and programming of the Andábata vehicle. Andábata is a small 4-wheeled mobile robot (see Figure 1) for autonomous navigation in the natural environment. It is battery-powered and has a 3D laser range finder and a suspension system. Moreover, the robot has a mobile phone to retrieve information (GPS, gyroscope, live video).

Figure 1: The mobile robot Andábata

All of his objectives have been achieved. He made the data sheet of the robot's mechanical, electrical and electronic systems. A programme was also made to control the movement of the robot. He described some possibilities for improvement such as:

- Develop wireless control method
- Implement a 3D laser range finder for a better perception of the environment
- Implement a suspension system
- Develop an application on smartphone
- Add various sensors for a better rendering of information

The main goal of the internship is to improve the mobile robot Vector. The selected improvement criteria were the wireless control method and a better perception of the environment. But also a better organization of structure for the program.

1-3 Objectives

The ROS2 use for the wireless control was requested in order to use later this methodology in other mobile robots in the laboratory. For the program structure, we created a driver using ROS2 to communicate. Also, the Vector's environment was improved with a LIDAR.
Therefore, the main objectives during this internship were:

- Create a structure with a driver
- Add ROS2 on the mobile robot to implement wireless control
- Implement a LIDAR to the system for a better perception of the environment
- Make a simple controller of the mobile robot with ROS2 to control the robot

2- Hardware

The first part of the internship consists in understanding how hardware works and the next part is about what was used to improve the mobile robot.This understanding is necessary because the driver communicates between the main program and the hardware. Finally,to create a driver; it is necessary to understand how the hardware is working.

2-1 Mechanical system

2-1-1 Robot structure

The mobile robot ”Vector” consists of a mBase MR5-2D chassis made by MoviRobotics. This chassis has a robust aluminum structure. The mobile robot has a length of 0.53m, a width of 0.45m, a height of 0.31m and a mass of 26kg.

Figure 2: Photograph of the structure on Vector

2-1-2 Wheels

Vector is a tricycle-type robot (see Figure 3) consisting of two independent non-steerable driving wheels and placed on the same axis. Moreover a free steerable centering wheel, also called an "idler wheel", was placed on the longitudinal axis to provide vehicle stability. The movement of the robot was given by the rotation speed of the driving wheels and by the orientation of the free steerable wheel. Its center of rotation was located in the center of the two driving wheels. The two driving wheels are each driven by a DC motor associated with an encoder.

A tricycle-type robot has several advantages and disadvantages over other configurations:
- Simple and robust mechanical system
- Non-holonomic, which means that it can not be moved in a direction perpendicular to the drive wheels.

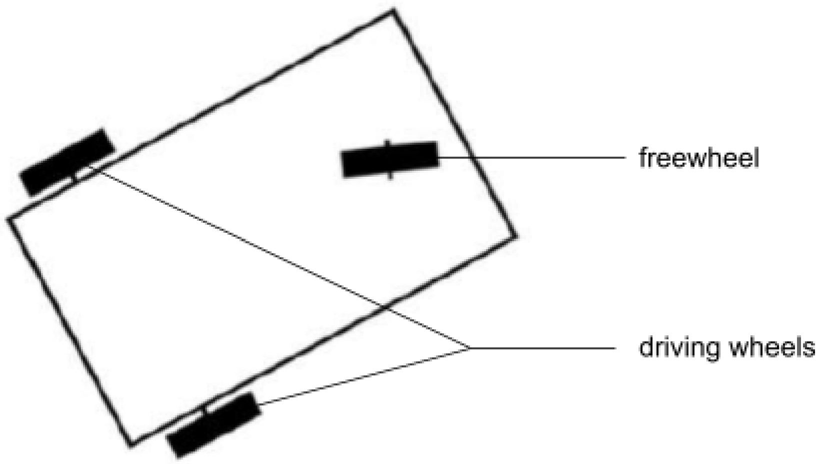


Figure 3: Vector wheel position diagram

The two-wheel drive had a diameter of 20 cm and consisted of an inner tube and an aluminum rim. The tires used had small irregularities allowing better driving in a natural environment.

2-1-3 Motors and Encoders

Figure 4: GM9236S019 motor with E30 encoder

The mobile robot was equipped with two Pittman GM9236S019 brushed DC motors (see Figure 4). These motors were simple to control, their cost was low and their behavior is reliable even under restrictive conditions. However, they required regular maintenance: the brushes must be regularly cleaned, and have a low heat dissipation capacity. These two motors had a stator consisting of permanent magnets. They offered better performance, better power-to-volume ratio and better reliability compared to other DC motors. They had a nominal voltage of 12 V, a maximum efficiency of 84%, a maximum rotation speed of 236 rpm, a nominal torque of 1.1 Nm

The two motors of the robot were each equipped with an integrated encoder. The encoders transmitted square signals to calculate the angular position and the speed of wheel rotation The encoders used for the robot were incremental optical encoders E30 from the company Pittman (see Figure 4).

- The radius of the driving wheels: 100 mm;
- A wheel revolution in a straight line corresponds to 2π * 100 = 628 mm;
- The reduction ratio: 19.7;
- Encoder resolution: 125 LPR (500 CPR);
- A straight wheel circumference corresponds to 500 * 19.7 = 2463 LPR (9850 CPR);

One turn of the wheel was 628 mm traveled in a straight line was equivalent to 2463 LPR counted.

2-2 Electronic

2-2-1 Electrical system

At the beginning of the internship, there was one power source for the power electronics. An emergency button was installed for safety reasons. With the installation of the Raspberry Pi 4 B for the driver, another power source was necessary. A 5V 10000mAh battery had been added as a power source. An emergency button was not necessary for this power source, a simple switch was added. (see Figure 5)

The LIDAR was powered with USB. The USB between Raspberry Pi 4 B and the Sabertooth send only information and not power.

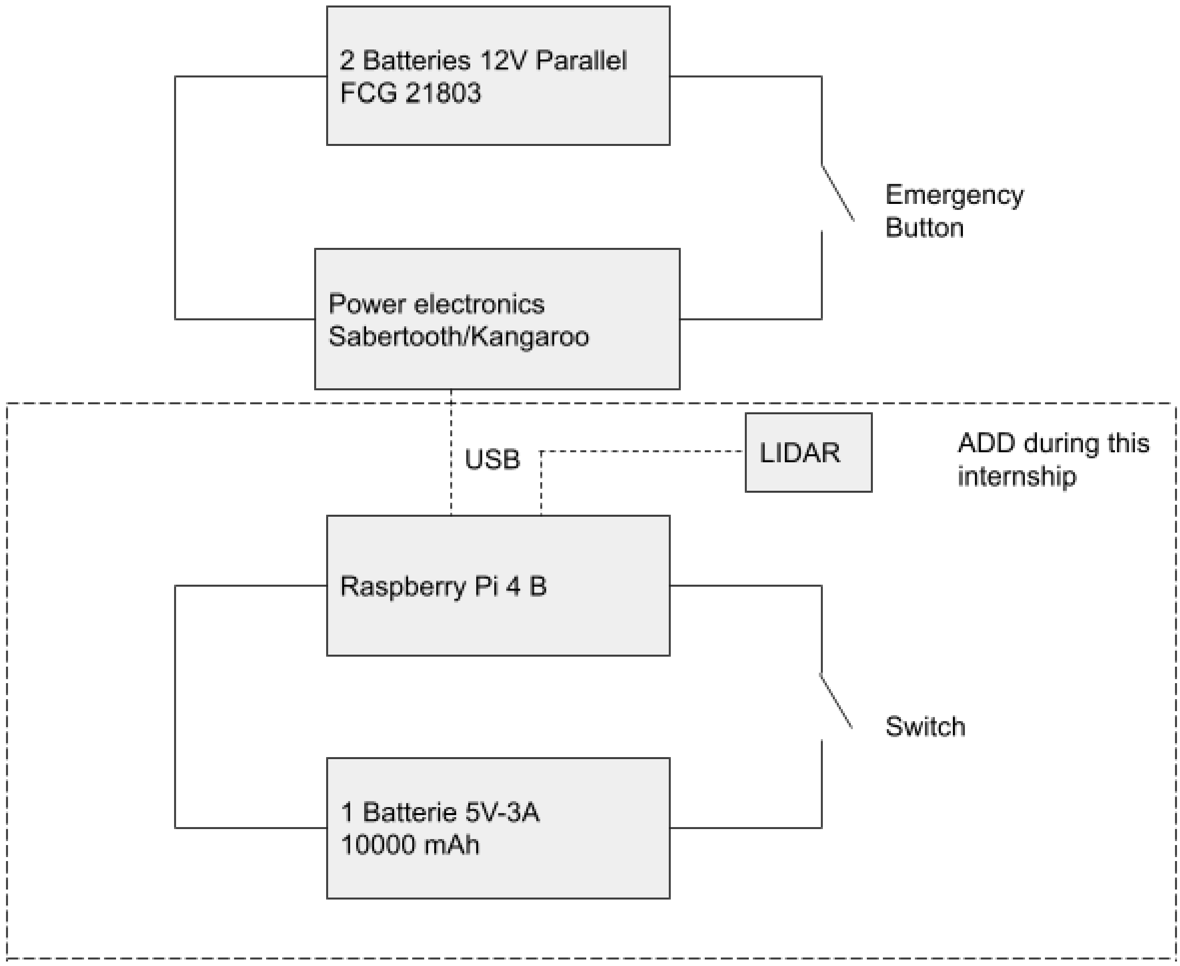


Figure 5: Diagram of electrical circuit

2-2-2 Power electronics

Electronic circuits of the mobile robot consisted of a Sabertooth 2x32 control board controlling the motors and a Kangaroo x2 module, that reads both incremental encoders, and can regulate the speed with feedback. As shown in Figure 6 below, the motors are connected to the Sabertooth 2x32 board. This card is connected to the Kangaroo x2 module which receives the commands sent with a serial-USB computer.

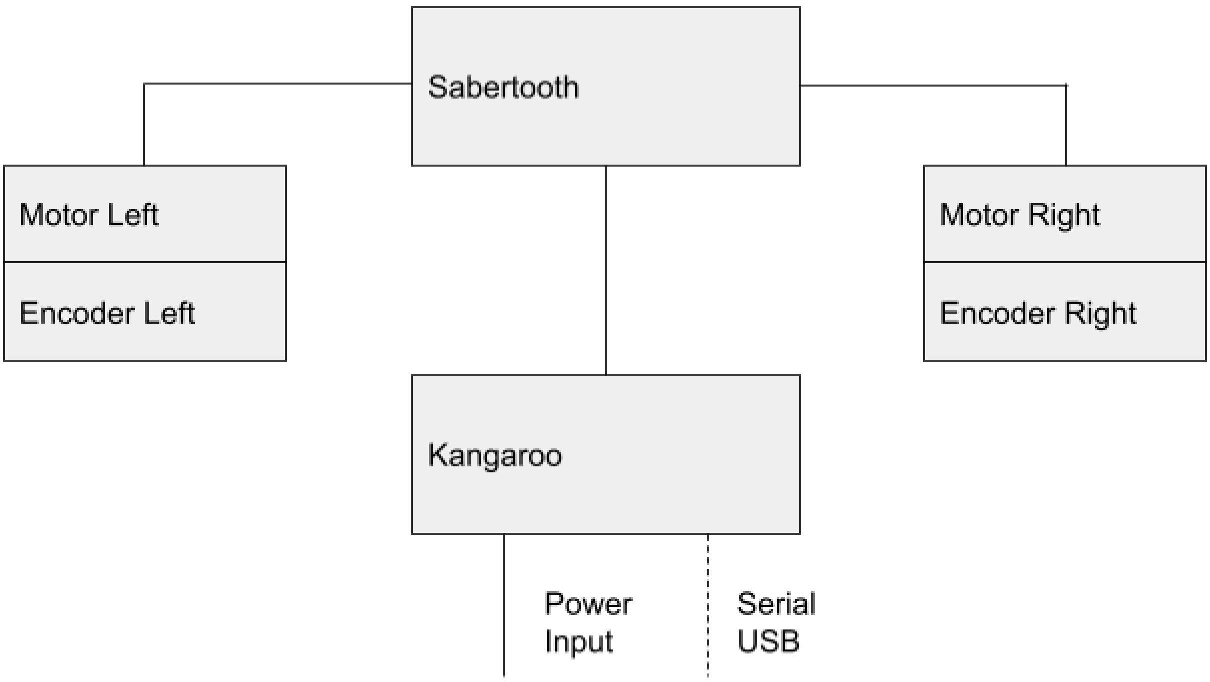


Figure 6: Diagram simplified of the power electronic

2-2-2 a) Sabertooth 2x32

The Dimension Engineering Sabertooth 2x32 board was a two-way control board capable of providing 32A to two motors with peaks of up to 64A per motor. There were four different control modes.

The main features of the Sabertooth 2x32 card are:

- Wide power supply range from 6 V to 33.6 V;
- Thermal protection and overload protection;
- A mode of protection of the batteries;
- Four modes of control possible;
- A configurable current limit.

The control mode is chosen using DIP switches located on the control board (see Figure 7). Switches 1 and 2 select the control mode, switch 3 configures the system power and switches 4, 5 and 6 select different configuration options. The four control modes available are:

- Analogical: a voltage was used to control the motors. For example, this voltage can be generated by a potentiometer e. The voltage range was 0 V to 5 V. This is the simplest use.
- Radio: R / C pulses were used to send commands to the motors. These signals were generated by radio transmitters / receivers or by a microcontroller.
- Series: a microcontroller was used to control the Sabertooth 2x32 card.
- USB: commands were received via a serial-USB port on the card (see Figure 7).

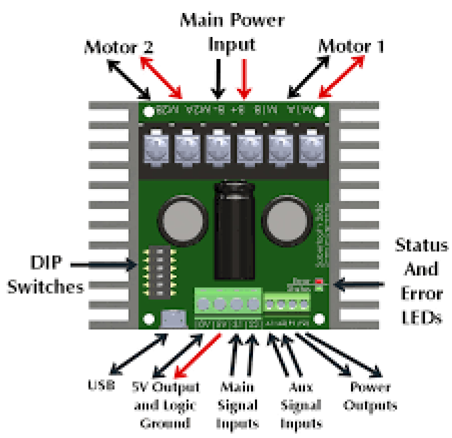


Figure 7: Sabertooth 2x32 control card connections

In our project the information was sent by USB to the Kangaroo 2x module. We have to set the Sabertooth 2x32 on DIP switches. Figure 8 shows possible choices for DIP switches.

The first two switches to select the control mode were ON and OFF respectively to select the USB mode. Switch 3 was in the OFF position to select a battery power supply. The switch 4 is in the ON position. Switch 5 in the OFF position allowed both the card to receive serial-USB commands but also to relay these commands to other modules such as the Kangaroo x2 module. Finally, the switch 6 was in the ON position, because there is an emergency stop button being located on the electrical circuit. The following combination is then obtained: ON / OFF / OFF / ON / OFF / ON.

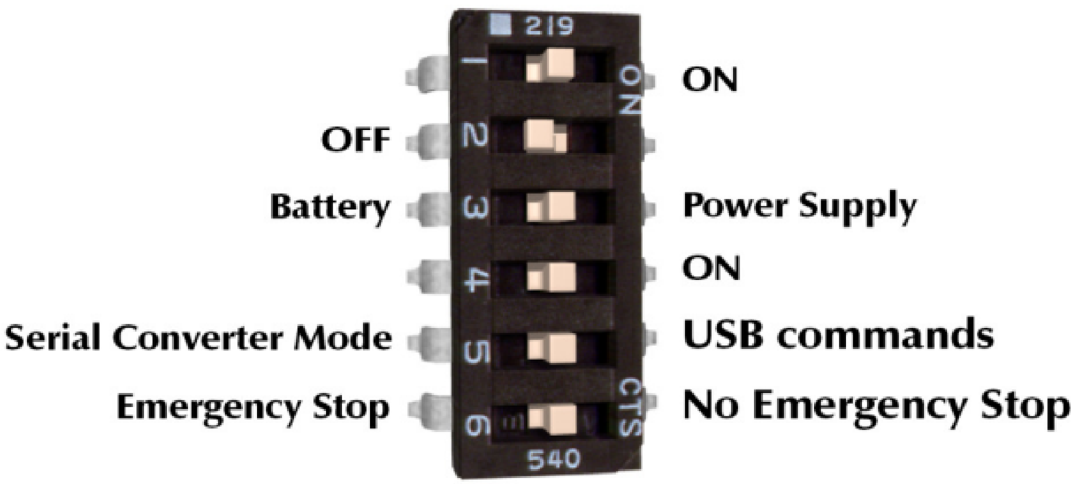


Figure 8: Choice of DIP switches on the Sabertooth 2x32 control board

2-2-2 b) Kangaroo x2

Dimension Engineering's Kangaroo x2 module read both encoders and added feedback loops to control the speed and position of the mobile robot's wheels in a closed loop. It connected directly to the Sabertooth 2x32 board via the S1 and S2, 0V and 5V serial ports (see Figure 7 and 9)

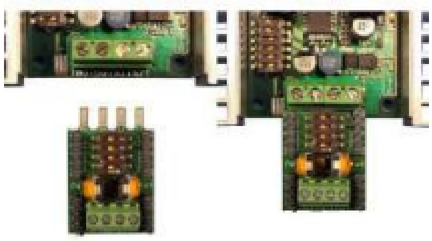


Figure 9: Separate Kangaroo x2 module connected to Sabertooth 2x32 board

Like the Sabertooth 2x32 a DIP switches have to be configured (see Figure 10). In our case ON / ON / OFF / OFF.

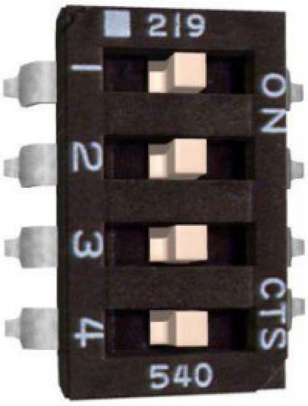
OFF setting		**ON setting**
1 off: Analog input. Connect 0-5V analog signals to the S1 and S2 inputs.		**1 on: Digital input** Connect TTL serial, TX to S1 and RX to S2, or R/C servo signals to S1 and S2
2 off: Analog feedback Connect a 0-5V signal to Feedback Input A		**2 on: Quadrature feedback** Connect an encoder to Feedback Inputs A and B
3 off: Velocity control Motor speed and direction are controlled by the input signal		**3 on: Position control** Motor position is controlled by the input signal
4 off: Mixed mode The outputs are mixed together for differential drive mobile robots		**4 on: Independent mode** The outputs are independent. S1 controls motor 1 and S2 controls motor 2.

Figure 10: Choice of DIP switches on the Kangaroo x2 board

2-2-3 Raspberry Pi 4 B

The Raspberry Pi 4 B is a nanocomputer that can connect to a monitor, a keyboard/mouse set and has Wi-Fi, Bluetooth and Ethernet interfaces. It was supplied with a case and a power supply.

In our case the Raspberry Pi 4 B was used as the driver processor. These USB ports were used to communicate with the Kangaroo and the Sabertooth 2x32 and the LIDAR (see Figure 5). Wi-Fiwas used to communicate with an external PC containing the mobile robot control program. A monitor, a keyboard and a mouse were added for card parameterization, then removed during final installation in the mobile robot Vector.

Raspberry Pi 4 B ran from a micro-SD card and worked with a Linux or Windows 10 IoT based operating system. Due to the desire to use the ROS 2 humble version, a version of Ubuntu 22.04 was installed on the Raspberry Pi 4 B.

2-3 Sensor LIDAR

In the mobile robot vector there was only one sensor, FHL-LD19 Lidar of the company Youyeetoo. This sensor was a LIDAR : “Laser Imaging Detection And Ranging”. It was added during this internship to improve the perception of the environment of the mobile robot.

It was composed of a laser ranging core,an angle measurement unit and a motor drive. The LD19 ranging core can measure 4,500 times per second. Each time the distance was measured, the LD19 emitted an infrared laser forward.he laser was reflected to the single photon receiving unit after encountering the target object. Thanks to this, the time when the laser was emitted and the time when the single-photon receiving unit captured the laser were obtained. The time difference between the two was the time of flight of light. The time of flight can be combined with the speed of light to calculate the distance. After obtaining the distance data, the LD19 combined the angle values measured by the angle measurement unit to form point cloud data, and then send the point cloud data to the external interface through wireless communication (see Figure 11).

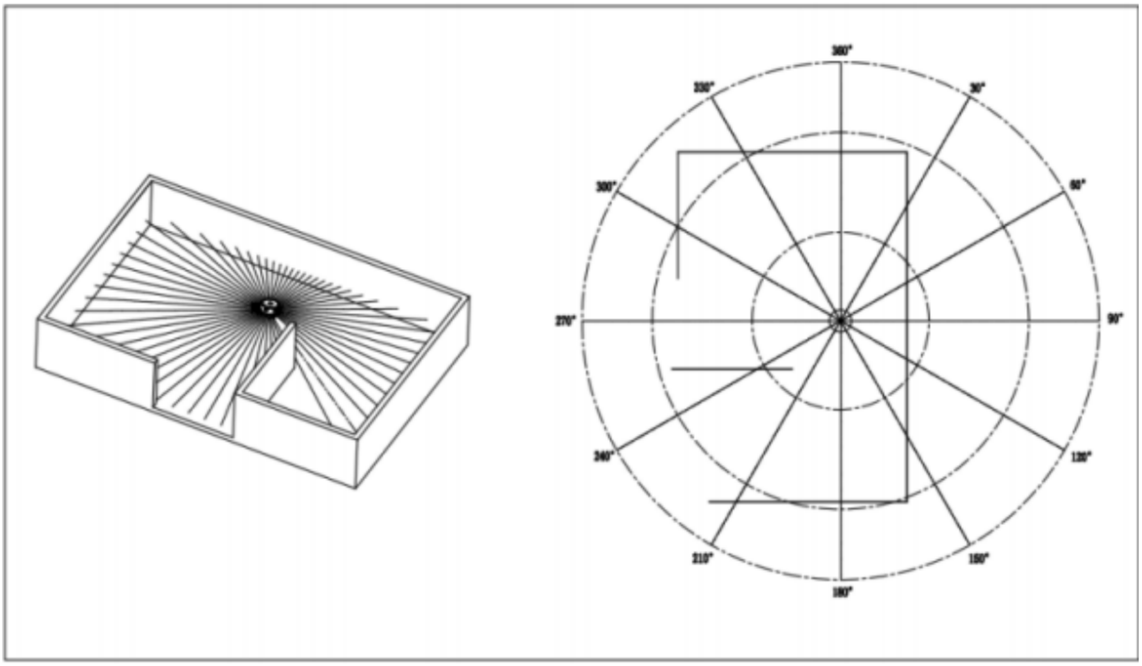


Figure 11: Environmental scan in a cloud data

3- Communication

After the work done on the hardware in order to understand the functioning of each component independent of one another, understanding how the hardware communicates with each other is important to write the driver. Three communications caught our attention:

- The communications with the Raspberry Pi. Because these were the ones that our driver will have to manage in order to transmit the information.
- The communication between the Kangaroo - Raspberry Pi
- he communication between the LIDAR - Raspberry Pi

The last two communication systems were the first that we analyzed. Their installation was easy to set up, connect these components via USB to the Raspberry Pi and authorize the transmission of information from a Raspberry Pi terminal. The last communication that interests us is between the PC and the Raspberry Pi. The latter will be done via a Wi-fi network that we will have to install in the laboratory.

3-1 Communication between Sabertooth/Kangaroo and Raspberry Pi 4

This communication was one of the communications between the driver and the hardware. It's made with a USB “simple” serial. It would send commands for motors and get information from the encoders (see Figure 12).

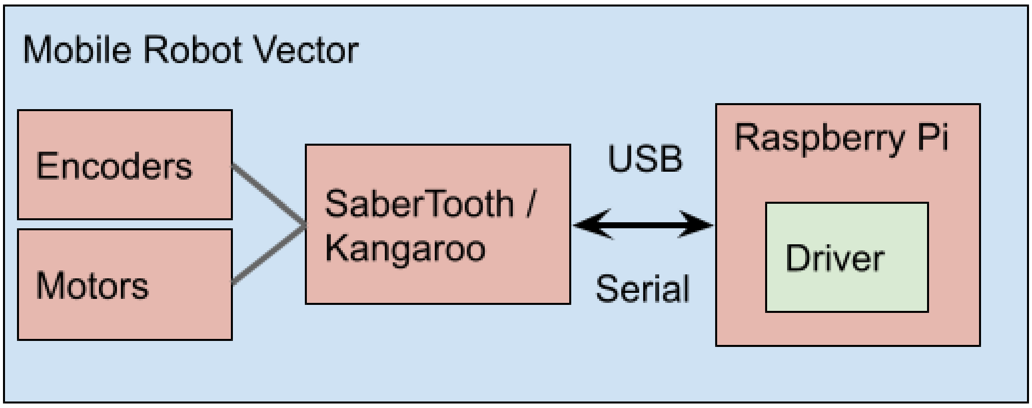


Figure 12: Communication between Hardware in Vector

We send a simple serial command with the USB encode in utf-8. The default serial settings to use this mode were 9600 baud, 8N1. All commands followed the same format. Spaces were ignored and can be added for readability. All commands consisted of a channel number, followed by a comma, the command and a newline. The channel number was 1 or 2 for the motor 1 or 2, in our case the motor 1 was the left motor and the motor 2 the right one. A list of commands possible was given by the constructor below.

Table 1: Motion command

Command	Result
p	Position command. The motor will go to the specified position, in units
s	Speed command. The motor will go at the specified speed, in units per second
powerdown	Power down. This command will turn off the motor and control system. You can still read position and speed with the motor powered off. This is used to allow the system to freewheel or to save power.

Table 2: Readback Commands

Command	Result
getp	Get position. Returns the channel number, followed by a comma, followed by a capital P if the move is completed or a lowercase p if the move is still going on, followed by the position in units (plain text) followed by a return and a newline
gets	Get speed. Returns the channel number, followed by a comma, followed by a capital S if the device is up to max speed or a lowercase s if the device is still accelerating, followed by the position in units (plain text) followed by a return and a newline

example of what we could send for the kangaroo x2:

1,start
2,start
1,s100
2,s100
1,gets
2,gets
1,powerdown
2,powerdown

3-2 Communication between LD19 and Raspberry Pi 4 B

Communication between the LD19 and the Raspberry Pi 4 B was a one-way communication. The LIDAR starts to send information on the USB without sending any commands. the packet format sending is show on Figure 13

Table 3: packet format

Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	1 Byte	LSB	MSB	LSB	MSB	......	LSB	MSB	LSB	MSB	1 Byte

Table 4: description of the packet

Value	Description
Header	The length is 1 Byte, and the value is fixed at 0x54, indicating the beginning of the data packet
VerLen	The length is 1 Byte, the upper three bits indicate the packet type, which is currently fixed at 1, and the lower five bits indicate the number of measurement points in a packet, which is currently fixed at 12, so the byte value is fixed at 0x2C
Speed	The length is 2 Byte, the unit is degrees per second, indicating the speed of the lidar
Start angle	The length is 2 Bytes, and the unit is 0.01 degrees, indicating the starting angle of the data packet point
Data	Indicates measurement data, a measurement data length is 3 bytes, please refer to the next section for detailed analysis
End angle	The length is 2 Bytes, and the unit is 0.01 degrees, indicating the end angle of the data packet point
Timestamp	The length is 2 Bytes, the unit is milliseconds, and the maximum is 30000. When it reaches 30000, it will be counted again, indicating the timestamp value of the data packet
CRC Check	The length is 1 Byte, obtained from the verification of all the previous data except itself. For the CRC verification method, see the following content for details

3-3 Communication between Raspberry Pi 4 and Computer

The communication between the raspberry pi and the computer was made by Wi-Fi. For wifi communication between the PC and the mobile robot a wifi network was necessary. Thanks to a wifi Routeur (Dlink) connected to the university network, a subnetwork of the latter was created.

This network allowed the wireless transmission of commands but also to connect me to the Raspberry Pi terminal from an external PC in order to configure the mobile robot. However, to do this you must know the IP address of the Raspberry Pi on the network. To facilitate connection, a fixed address had been configured on the Raspberry PI 4: 192.168.10.102.

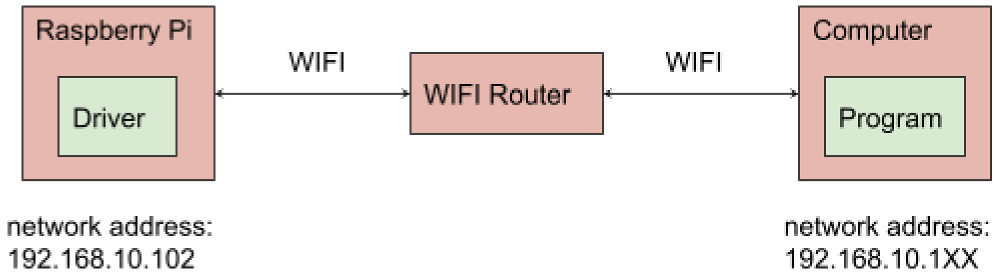


Figure 13: Communication with a network

4- Program

The first part of this internship consisted of understanding the hardware and resulting with the creation of the mode motors. This step was essential to start Vector programmation.. Once this node was completed, a test node was created in order to remove errors in the motor node. After adding the LIDAR to the robot, the LD19 node created by the LIDAR manufacturer was installed on the robot. The test node had been modified accordingly in order to also test the LD19 node. The last part of the internship was focused on the Launch folder created to facilitate the launch of the different nodes.

The folder of Vector was composed of three parts (see Figure 14).

- There was the folder “doc” with all documents of the mobile robot vector.
- There is the folder driver where there are two packages ROS 2.
- The last folder is the launch file to launch the driver and the different packages of the mobile robot.

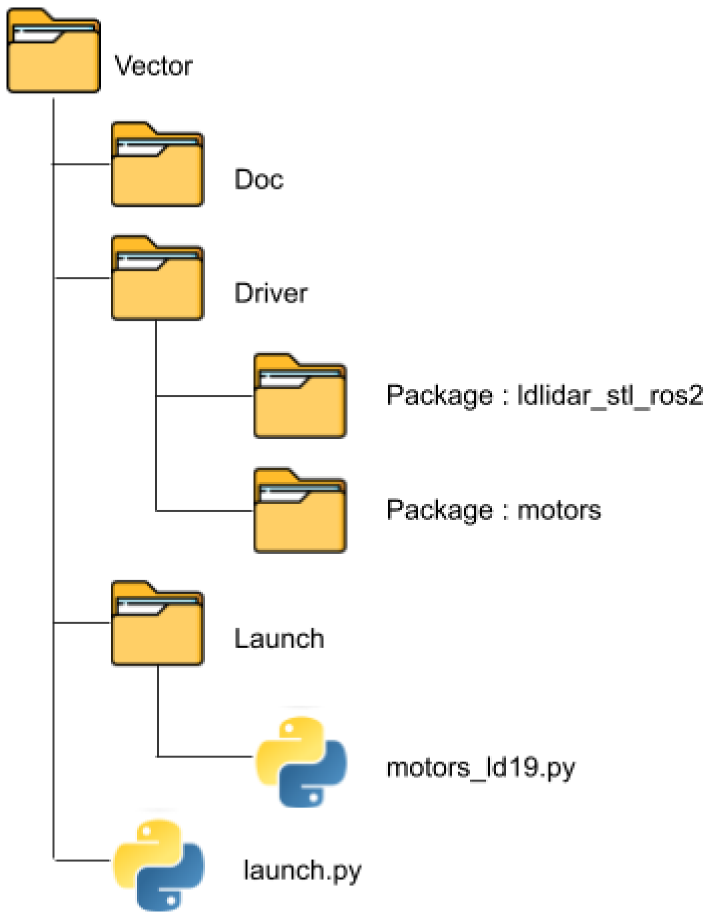


FIgure 14: Package and files organization of the mobile robot Vector

To work on this project, all files were on a GitLab. Gitlab is a platform for hosting and managing projects. To update the program, we made the change on a computer, we pushed the modification, we pulled it on the mobile robot Raspberry Pi 4, and we compiled it directly on the mobile robot.

To launch the mobile robot the file launch.py must be launched. This program authorized USB ports to send and receive information. This program also ran a ROS 2 launch file, motors_ld19.py, which launched the LD19 and motors nodes.

4-1 Driver

Initially, the robot control and the sending of the simplified serials necessary for the Kangaroo were in the same program. However, this program structure was risky. If there was a programming error or any other bug in the robot control part , the program loop would not stop and the robot would also stop working.
Adding a driver separated the two programs. A simple program "without" bugs and on the mobile robot, the driver, served as a communication gateway. And we were able to code a more complex program, requiring more resources, which can be run in parallel on a remote and more powerful system.

Thus, the driver should only be used to process and transmit information from a higher level program to the hardware (see Figure 15).

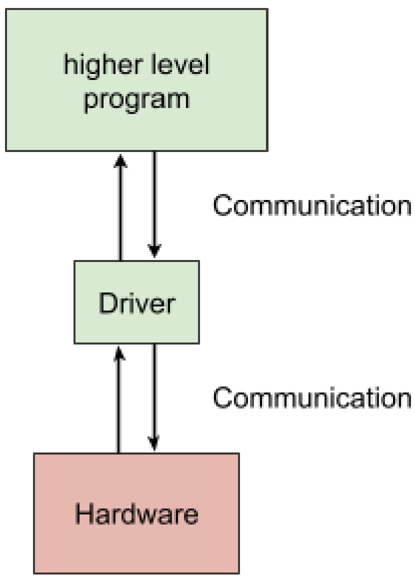


Figure 15: Simplified driver communication diagram

ROS 2 works with nodes and topics. Nodes are nodes where information is intercepted and processed by programs. Topics are the information channels where information can pass between nodes.

Our driver was composed of 2 packages ROS 2, the node LD19 and the node motors (see Figure 16). Each package was a node ROS2 which was launched when the mobile robot started. Each Node was independent and was able to work without the other. This had the main advantage : if one of the two nodes crashed the other continued to function normally. The node LD19 sent information on the topic scan and the node motors read information on topics cmd_motor_left and cmd_motor_right.

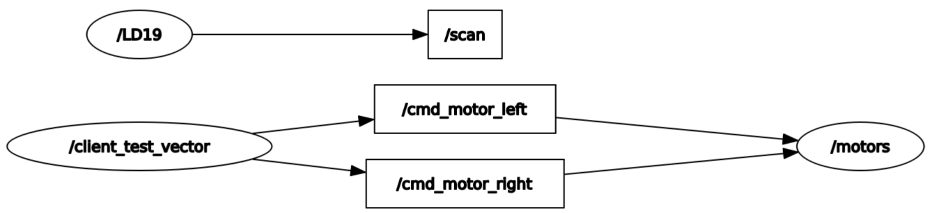


Figure 16: ROS 2 Graph

5-1 Node Motors

This node made the communication between the Kangaroo/sabertooth and other nodes. Therefore, this node had two goals: send the command to the motors and get the information from the encoders.

There were two topics and one service. The two topics were: cmd_motor_left and cmd_motor_right. There were commands for each motor. They send a command of speed in turns per second of the wheel. To do this, the ratio described in part 2-1-3 Motors and Encoders was necessary. The message sent in these topics was a float64. Therefore for each message the subscriber sent a simplified serial to the Kangaroo see in 3-1 Communication between Sabertooth/Kangaroo and Raspberry Pi 4.

The node motors was the server of the service. Clients send a bool, if it was true the node sent back a string with the information of the encoder position.

5-2 Node LD19

This node sent information about the LIDAR. This node had been made by the company of the LIDAR Youyeetoo in ROS 2. The ROS 2 function package of this product supported the use of the ROS 2 Foxy (Ubuntu 20.04) version environment. In our case we used this node in ROS 2 Humble (Ubuntu 22.04) version. No error was detected so the node provided by the manufacturer was not updated. This node had one topic: scan. This topic send:

- angle_min (float)

This float was the minimum angle of the LIDAR

- angle_max (float)

This float iwas the maximum angle of the LIDAR

- angle_increment (float)

This float is incrementation of angle between each measurement

- time_increment (float)

- scan_time (float)

- range_min (float)

This float was the minimum range measured

- range_max (float)

This float was the maximum range measured

- ranges (sequence<float>)

The list of ranges. This list started with the range of the minimum angle and ended with the range of the maximum angle.

- intensities (sequence<float>).

The list of intensities. This list started with the intensity of the minimum angle and ended with the intensity of the maximum angle.

Youyeetoo also allowed us to test the LIDAR interface on Rviz2 in order to display the point clouds given by the LIDAR. Therefore we can see the scan of our laboratory (see Figure 17). The small red line represents the positioning of the LIDAR on the scan. The points are the values of the ranges list. The range between blue and red represents the intensity of the received signal. The more blue it is, the lower the intensity, the more it is red, the higher the intensity.

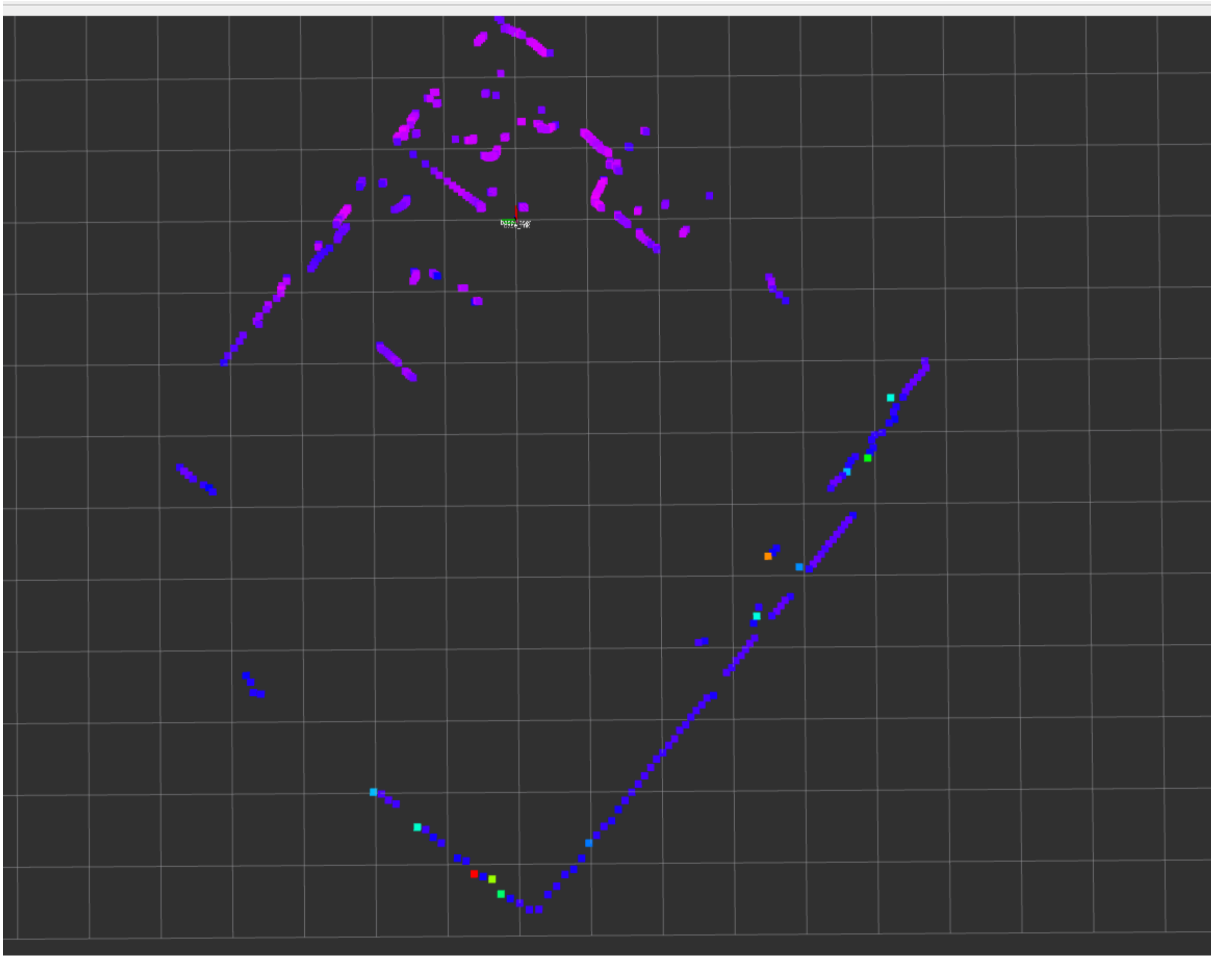


Figure 17: Scan on Rviz 2 of the laboratory

4-2 Node Test

This node was on a PC distant from the mobile robot. The goal of this node was to test the node motor and the node LD19. The node read the keyboard of the computer, send information on the topic cmd_motor_left and the topic cmd_motor_right and read the topic scan. The nod read the following keys:

- Arrow Forward set both command motors on 1 turn/sec
- Arrow Back set both command motors on -1 turn/sec
- Arrow Right set the command of the right motors on 1 turn/sec and set the command of the left motors on -1 turn/sec
- Arrow Left set the command of the right motors on -1 turn/sec and set the command of the left motors on 1 turn/sec

The node also created a screen where it puts the cloud of data of the topic scan (see Figure 18). The red dot was the LIDAR and black dots were measurements.

FIgure 18: Scan on the node test of the laboratory

5- Results

5-1 Culture and communication

During my 16-week internship in Spain, my immersion in Spanish culture showed me a new way of working. My supervisor Juan Carlos ALVAREZ ALVAREZ gave me the confidence and total autonomy to carry out the necessary modifications to the mobile robot, Vector. His aim was not to set precise objectives and a precise work methodology, but to get results. That's why every week we had a meeting to show how the project was progressing.This autonomy enabled me to concentrate on the areas of improvement I felt Vector needed.

These meetings were conducted in English, even though occasional words and phrases were expressed to facilitate communication. Living in Spain has also enabled me to improve my Spanish.

Towards the end of the internship, my supervisor Juan Carlos ALVAREZ ALVAREZ asked me to prepare a presentation of Vector and more specifically a presentation of ROS 2. Indeed, ROS 2 was rarely used in the various projects at the laboratory. This presentation was given in English to doctoral students and colleagues working in the laboratory.

5-3 Work Done

By the end of August 2023, the objective of the internship had been met. The driver was installed in Vector. A simple launch.py runs the driver. With this driver, we can control the rotation of the motors, get their speed, and get the cloud data of the LIDAR. A node test was also available to test Vector and to communicate with the driver. The documentation will make it easy for someone to use the driver and improve it.

5-4 Future work

Numerous modifications and improvements can now be made on Vector. The next improvements that could be interesting to make are:

- Cleanly attach the LIDAR to the outside of Vector: unfortunately the LIDAR was not fixed.
- Increase the amount of information the driver can send to the user. Some information might be interesting to extract from Vector, such as battery level or even internal temperature.
- Sensor additions such as sensors. We only have LIDAR as a sensor. We could add a central inertial unit or even a satellite positioning system.

6- Conclusion

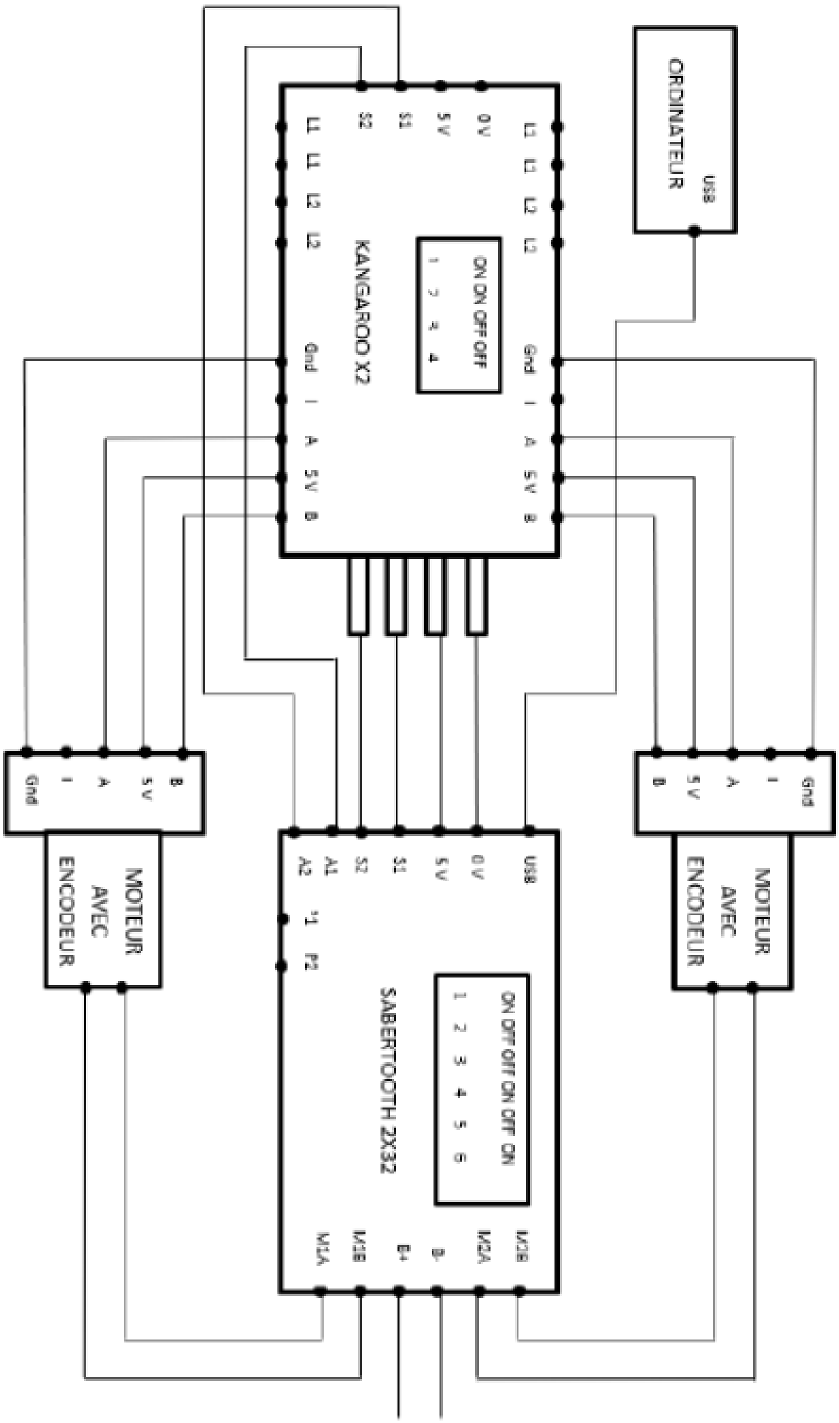
During my 16-week internship, I was able to advance and improve an existing project, but I also had to understand how the robot worked and how the hardware was organized. This gave me a global view of the project. This overview was motivating, especially as I had autonomy over the project. Thus showing that understanding the project can be a motivating factor in a company.

The project was finalized, and the initial goals, such as the addition of the driver with ROS 2 and the new LIDAR sensor, have been achieved. However, there is still room for improvement for Vector, for example we could add new sensors and new functions to the driver.

This internship put my knowledge of ROS 2 into practice, learning about actuators, pre-actuators and an electronic system. I learned how to manipulate and parameterize a Raspberry Pi and a LIDAR,all this independently. The knowledge I acquired and used during this internship was unique, and fits in perfectly with my curriculum and career plan.

Appendice

Appendix 1: Diagram Complet of the power electronic



Appendix 2: node motors.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32
from std_msgs.msg import Int32
from std_srvs.srv import SetBool

import serial
import serial.tools.list_ports
import pysabertooth

class motors(Node):

        '''
        The motors is heritance of Node.
        '''

        def __init__(self):
        print("Motors Initialisation")
        super().__init__('motors')

        # with the id_saber, find the right port to use
        id_saber = 513
        port_sabertooth = self.find_port(id_saber)

        if port_sabertooth is not None:
        self.sabertooth = pysabertooth.Sabertooth(port_sabertooth, baudrate=9600)
        self.kangaroo = serial.Serial(port_sabertooth,9600)
        print('temperature [C]: {}'.format(self.sabertooth.textGet(b'm2:gett')))
        print('battery [mV]: {}'.format(self.sabertooth.textGet(b'm2:getb')))
        self.init_kangaroo()
        else:
        print("port is not founding")

        # subscriber of the left motor command in speed
        self.sub_cmd_motor_left = self.create_subscription(
        Float32,
        'cmd_motor_left',
        self.cmd_motor_left,
        10)

        self.sub_cmd_motor_right = self.create_subscription(
        Float32,
        'cmd_motor_right',
        self.cmd_motor_right,
        10)

        self.srv = self.create_service(SetBool, 'ask_encodors', self.send_encodors)


        def find_port(self,identifiant):
        ports = list(serial.tools.list_ports.comports())
        for port in ports:
        print(port.pid)
        if identifiant == port.pid:
        return port.device
        return None

        def init_kangaroo(self):
        print("START CONNECTING WITH KANGAROO")
```

```
self.kangaroo.write('1,start\n'.encode())
self.kangaroo.write('2,start\n'.encode())

self.kangaroo.write('1,getp\n'.encode())
line = self.kangaroo.readline()
print(int(line.decode()[3:]))

self.kangaroo.write('2,getp\n'.encode())
line = self.kangaroo.readline()
print(int(line.decode()[3:]))

print("END INITIALISATION KANGAROO")

def send_encodors(self, request, response):
if request.data:

self.kangaroo.write('1,getp\n'.encode())
line = (self.kangaroo.readline()).decode()
enc_left=int(line[3:])

self.kangaroo.write('2,getp\n'.encode())
line = (self.kangaroo.readline()).decode()
enc_right=int(line[3:])

response.success = True
response.message = f"{enc_left},{enc_right}"
else:
response.success = False
response.message = "0,0"

return response

def cmd_motor_left(self, msg):
print("cmd left : ",float(msg.data))
commande = str(f'1,s{1000*int(msg.data*2.5)}\n')
self.kangaroo.write(commande.encode())

def cmd_motor_right(self, msg):
print("cmd right : ",float(msg.data))
commande = str(f'2,s{1000*int(msg.data*2.5)}\n')
self.kangaroo.write(commande.encode())


def main(args=None):
    rclpy.init(args=args)
    node = motors()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Appendix 3: motor_ld19.py

```
#!/usr/bin/env python3
from launch import LaunchDescription
from launch_ros.actions import Node

'''
Parameter Description:
---
- Set laser scan directon:
  1. Set counterclockwise, example: {'laser_scan_dir': True}
  2. Set clockwise,                 example: {'laser_scan_dir': False}
- Angle crop setting, Mask data within the set angle range:
  1. Enable angle crop fuction:
        1.1. enable angle crop,  example: {'enable_angle_crop_func': True}
        1.2. disable angle crop, example: {'enable_angle_crop_func': False}
  2. Angle cropping interval setting:
  - The distance and intensity data within the set angle range will be set to 0.
  - angle >= 'angle_crop_min' and angle <= 'angle_crop_max' which is [angle_crop_min, angle_crop_max], unit is degress.
        example:
        {'angle_crop_min': 135.0}
        {'angle_crop_max': 225.0}
        which is [135.0, 225.0], angle unit is degress.
'''

def generate_launch_description():
  # LDROBOT LiDAR publisher node
  ldlidar_node = Node(
        package='ldlidar_stl_ros2',
        executable='ldlidar_stl_ros2_node',
        name='LD19',
        output='screen',
        parameters=[
        {'product_name': 'LDLiDAR_LD19'},
        {'topic_name': 'scan'},
        {'frame_id': 'base_laser'},
        {'port_name': '/dev/ttyUSB0'},
        {'port_baudrate': 230400},
        {'laser_scan_dir': True},
        {'enable_angle_crop_func': False},
        {'angle_crop_min': 135.0},
        {'angle_crop_max': 225.0}
        ]
  )
  # motors publisher and server node
  motor_node = Node(
        package='driver',
        executable='motors',
        name='motors'
  )

  # base_link to base_laser tf node
  base_link_to_laser_tf_node = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        name='base_link_to_base_laser_ld19',
        arguments=['0','0','0.18','0','0','0','base_link','base_laser']
  )


  # Define LaunchDescription variable
```

```
ld = LaunchDescription()

ld.add_action(ldlidar_node)
ld.add_action(motor_node)
ld.add_action(base_link_to_laser_tf_node)

return ld
```

Appendix 4: launch.py

```
import subprocess

subprocess.call("sudo chmod 666 /dev/ttyACM*", shell=True)
subprocess.call("sudo chmod 777 /dev/ttyUSB*", shell=True)
subprocess.call("ros2 launch launch/motor_ld19.py", shell=True)
```

Appendix 5: node node_test.py

```
import sys
import pygame
import numpy as np

import rclpy
from rclpy.node import Node
from std_srvs.srv import SetBool
from std_msgs.msg import Float32
from sensor_msgs.msg import LaserScan

class Node_test(Node):

        def __init__(self):
        super().__init__('client_test_vector')
        self.cli_vector = self.create_client(SetBool, 'ask_encodors')
        # while not self.cli_vector.wait_for_service(timeout_sec=10.0):
        #             self.get_logger().info('service not available, waiting again...')
        # self.req = SetBool.Request()

        self.pub_cmd_left = self.create_publisher(Float32, 'cmd_motor_left', 10)
        self.pub_cmd_right = self.create_publisher(Float32, 'cmd_motor_right', 10)
        self.sub_ld = self.create_subscription(
        LaserScan,
        '/scan',
        self.sub_lidar,
        10)
        self.vector_ld=[]

        def send_request(self):
        self.req.data = True
        self.future = self.cli_vector.call_async(self.req)
        rclpy.spin_until_future_complete(self, self.future)
        return self.future.result()

        def send_cmd(self, value_left,value_right):
        msg = Float32()
        msg.data = value_right
        self.pub_cmd_right.publish(msg)
        msg.data = value_left
        self.pub_cmd_left.publish(msg)


        def send_cmd_right(self, value):
        msg = Float32()
        msg.data = value
        self.pub_cmd_right.publish(msg)

        def sub_lidar(self,msg):
        coef = 50
        ranges=msg.ranges
        n=len(ranges)
        vector=[]
        for i in range(len(ranges)):

        if np.isnan(ranges[i]):
        vector.append([0,0])
        else:
        vector.append([int(coef*ranges[i]*np.cos(2*np.pi*i/n+np.pi/2)),int(-coef*ranges[i]*np.sin(2*np.pi*i/n+np.pi/2))])
```

```
        self.vector_ld = vector

def main(args=None):
        rclpy.init(args=args)
        pygame.init()

        node_test = Node_test()

        SIZE = [1000, 1000]
        screen = pygame.display.set_mode(SIZE)

        clock = pygame.time.Clock()
        global go_up,go_right,go_none,go_down,go_left
        go_up = False
        go_left = False
        go_right = False
        go_down = False
        go_none = True

        white=pygame.Color(255,255,255)
        black=pygame.Color(0,0,0)
        red=pygame.Color(255,0,0)


        while rclpy.ok():
        rclpy.spin_once(node_test)

        motor_left=0.0
        motor_right=0.0

        for event in pygame.event.get():
        if event.type == pygame.QUIT:
        running = False

        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
        if not go_left:
        print("LEFT")

        go_up = False
        go_left = True
        go_right = False
        go_down = False
        go_none = False

        motor_left = -1.0
        motor_right = 1.0
        node_test.send_cmd(motor_left,motor_right)

        elif keys[pygame.K_RIGHT]:
        if not go_right:
        print("RIGHT")

        go_up = False
        go_left = False
        go_right = True
        go_down = False
        go_none = False

        motor_left = 1.0
```

```
motor_right = -1.0
node_test.send_cmd(motor_left,motor_right)

elif keys[pygame.K_UP]:
if not go_up:
print("UP")

go_up = True
go_left = False
go_right = False
go_down = False
go_none = False

motor_left = 1.0
motor_right = 1.0
node_test.send_cmd(motor_left,motor_right)

elif keys[pygame.K_DOWN]:
if not go_down:
print("DOWN")

go_up = False
go_left = False
go_right = False
go_down = True
go_none = False

motor_left = -1.0
motor_right = -1.0
node_test.send_cmd(motor_left,motor_right)

else:
if not go_none:

go_up = False
go_left = False
go_right = False
go_down = False
go_none = True

node_test.send_cmd(motor_left,motor_right)


screen.fill(white)

for point in node_test.vector_ld:
point[0]+=SIZE[0]//2
point[1]+=SIZE[1]//2
pygame.draw.circle(screen,black,point,5)

pygame.draw.circle(screen,red,(SIZE[0]//2,SIZE[1]//2),5)
pygame.display.update()

clock.tick(30)  # Limite la boucle principale à 60 FPS

node_test.destroy_node()
rclpy.shutdown()


if __name__ == '__main__':
    main()
```

Appendix 6: Assessment report

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – *Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE*
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Universidad de Oviedo

Adresse / *Address* Ed. Torres Quevedo (Departamental Oeste) bloque 2, Laboratorio de Robótica, 2.B.02

Tél / *Phone (including country and area code)* +34 985 182 068

Nom du superviseur / *Name of internship supervisor* Juan Carlos Alvarez Alvarez

Fonction / *Function* Group Coordinator Associate Professor

Adresse e-mail / *E-mail address* juan@uniovi.es

Nom du stagiaire accueilli / *Name of intern* Pilon Martin

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
Please attribute a mark from ***A (excellent)*** *to* ***F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? *Was the initial contract carried out to your satisfaction?* A B C D E F

- Manquait-il au stagiaire des connaissances ? *Was the intern lacking skills?* ☐ oui/*yes* ☒ non/*no*

  Si oui, lesquelles ? / *If so, which skills?* ______

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)* A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*
He integrated in the lab and gave a nice presentation to the group about the work he was doing.

ENSTA BRETAGNE

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* He reported periodically about his work in the lab.

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*
He worked in a open project, and was able to solve the problems that emerged.

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* He was able to communicate well enough in english.

OPINION GLOBALE / *OVERALL ASSESSMENT*

- La valeur technique du stagiaire était : A B C D E F
  Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

- Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?
  Would you be willing to host another intern next year? [X] oui/*yes* [ ] non/*no*

Fait à ______, le ______
In Gijon, on 21/08

Juan Carlos Alvarez

UNIVERSIDAD DE OVIEDO
Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas

Signature Entreprise / *Company stamp* ______ Signature stagiaire / *Intern's signature*

Merci pour votre coopération
We thank you very much for your cooperation