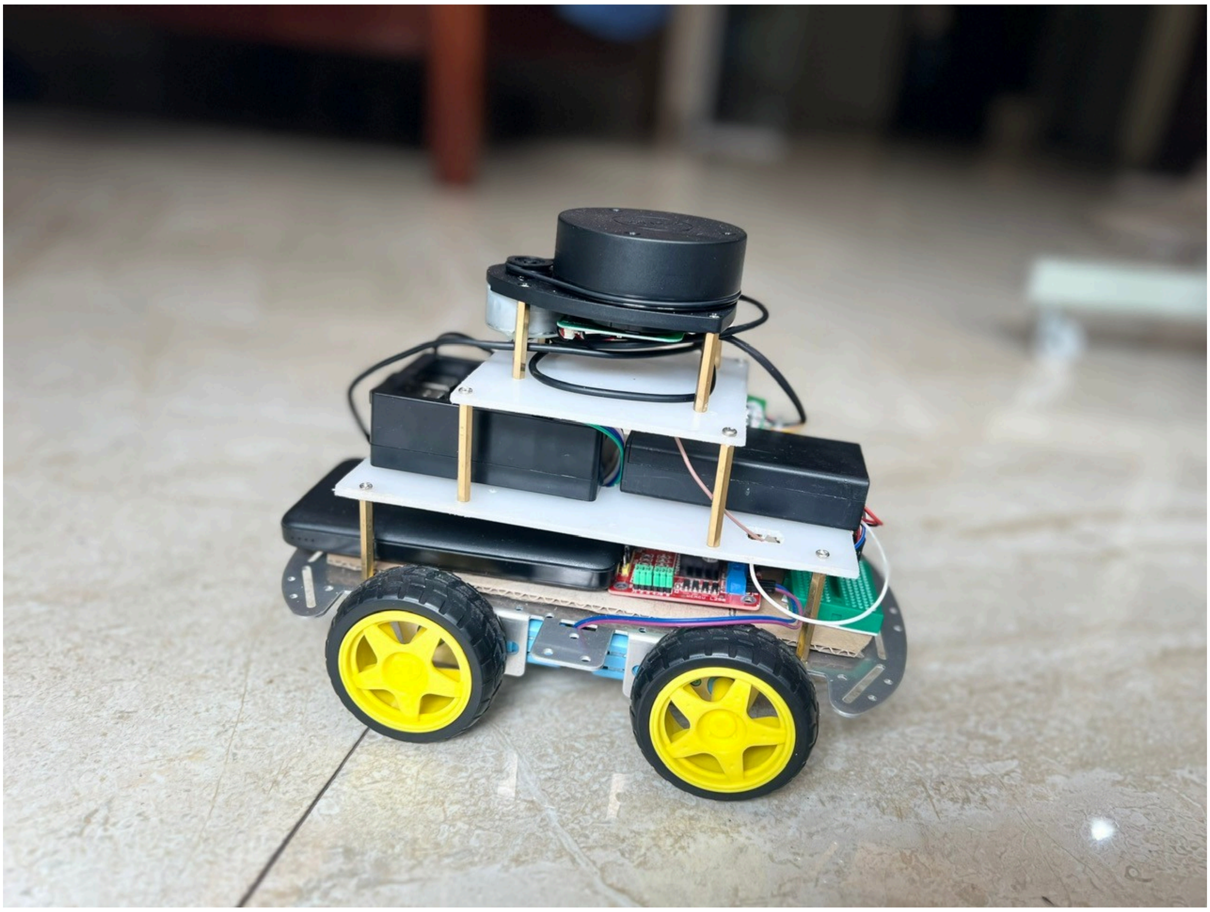
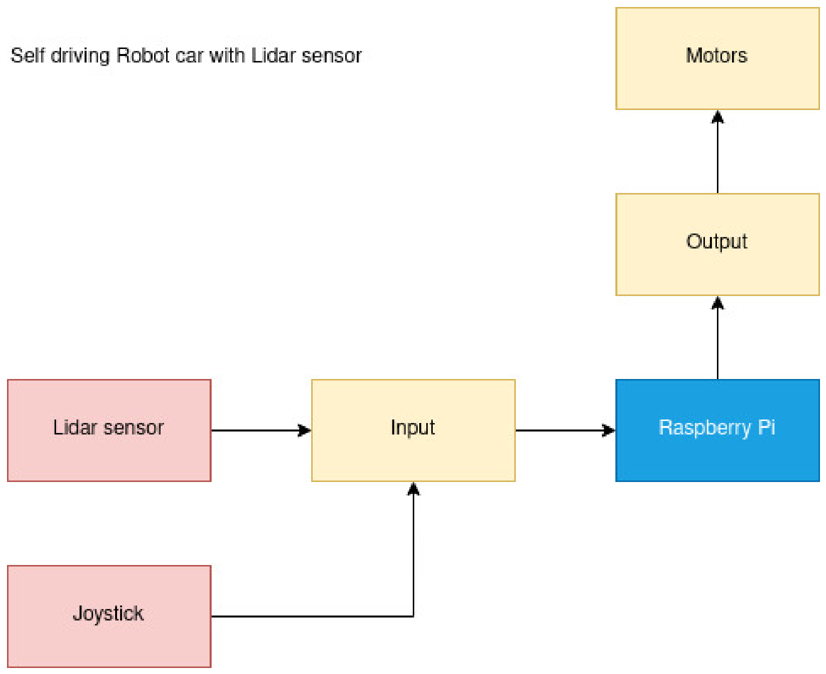
# Autonomous self driving car using Raspberry Pi and RPLidar

# 1. Introduction

## 1.1. The robot



### 1.2. Logical components




## 2. Hardware components

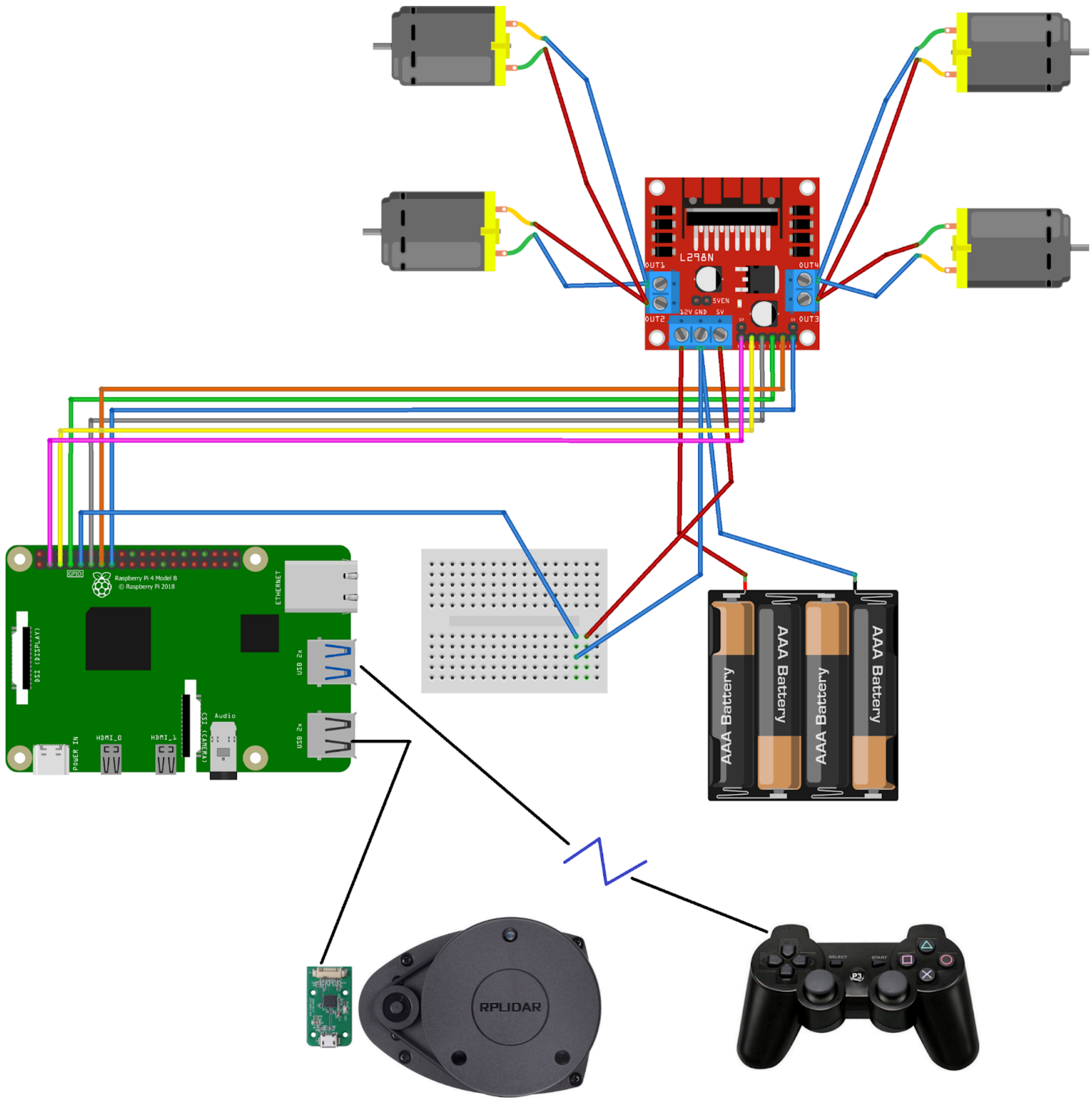
### 2.1. Electronic components list

| STT | Name | Quantity | Note |
|---|---|---|---|
| 1 | Raspberry Pi 4B  | 1 | |
| 2 | Raspberry Pi 4 case  | 1 | |
| 3 | RPLidar A1M8 | 1 | |

| | | | |
|---|---|---|---|
| |  | | |
| 4 | L298N motor driver<br> | 1 | Prefer the v1 as it has leds indicating signals. |
| 5 | Reducer Motor<br> | 4 | |
| 6 | Robot chassis<br> | 1 | |
| 7 | Copper cylinder<br> | * | |
| 8 | PVC plastic<br> | | |
| 9 | Battery for motor 18650 | 2 | |

| | | | |
|---|---|---|---|
| |  | | |
| 10 | Battery case  | | To power the motor separately from Raspberry Pi |
| 11 | Power bank  | | To power the Raspberry Pi 4 |
| 12 | 3.3V to 5V logic converter  | | This module is optional |

### 2.2. The Wiring




# 3. Software

## 3.1. Controlling motors

L298N allows 2 channels output with PWM control. The input signals are: ena, in1, in2, enb, in3, in4.

Because Raspberry Pi pins operate at 3.3V while the L298N driver operates at 5V:

- We could either use a logic converter
- Or wire the GND of L298N module with GND on Raspberry Pi in order for the HIGH/LOW threshold to work for 3.3V.

## 3.2. Modules

### 3.2.1. Motor controller

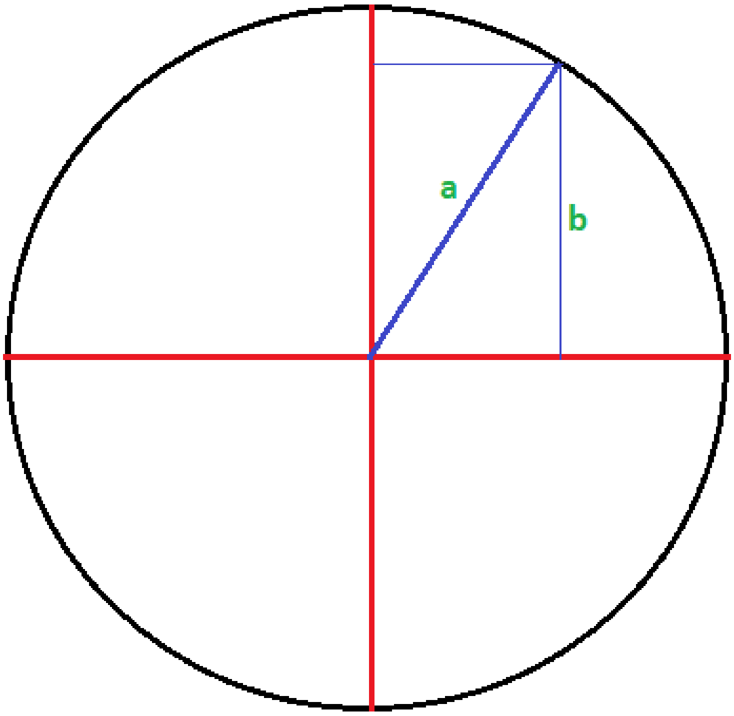
I would prefer the L298N v1 because it has leds to indicate HIGH/LOW signal of corresponding in1, in2, in3, in4.

Murtaza, Youtube How to run robot motors

### 3.2.2. Remote controller using joystick

Depending on the joystick model that we have, control input can be varied. Use the page https://hardwaretester.com/gamepad to test and check input of the joystick.

Note that depends on the driver available to your platform (Raspberry Pi OS, or Windows etc) the input can be different.




**b = a x cosα**
**tanα = x/y**
**α = tan-1(x/y)**
**b = a x cos(tan-1(x/y))**

Check out this Youtube video from Murtaza on controlling motor with joystick:
Murtaza, Youtube How to run joystick with Raspberry Pi

### 3.2.3. Rplidar

Read the manual, SDK documentation from Semantec:

- Data is sent via serial communication, data is sent into the serial buffer.
- If buffer become full, data lagging is impacting our TURN outcome (ie outcome might not corresponding to the data frame recorded)
- To avoid data lagging, we need real-time or near real-time data, to clear the serial buffer if it reaches tolerant capacity.

Python RPLidar library:
Github, Robotica/RPLidar
Customized code, https://github.com/andygiangnh/iot/tree/master/RobotCar/ML

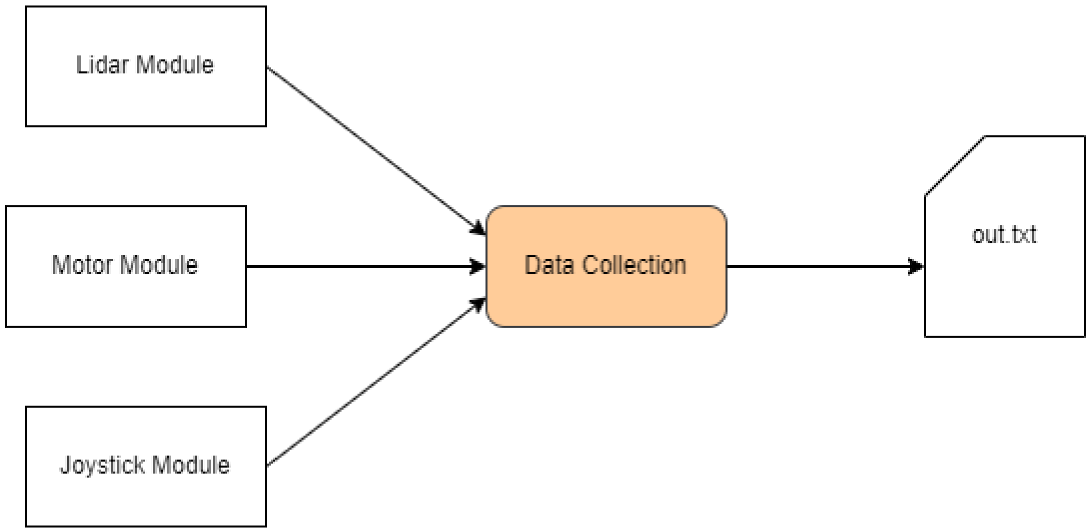
**Structure of data collection:**
=> use 360 data points corresponding to the angle our lidar rotates.

Choose a appropriate Python library:
=> modify the library to suit our requirements.

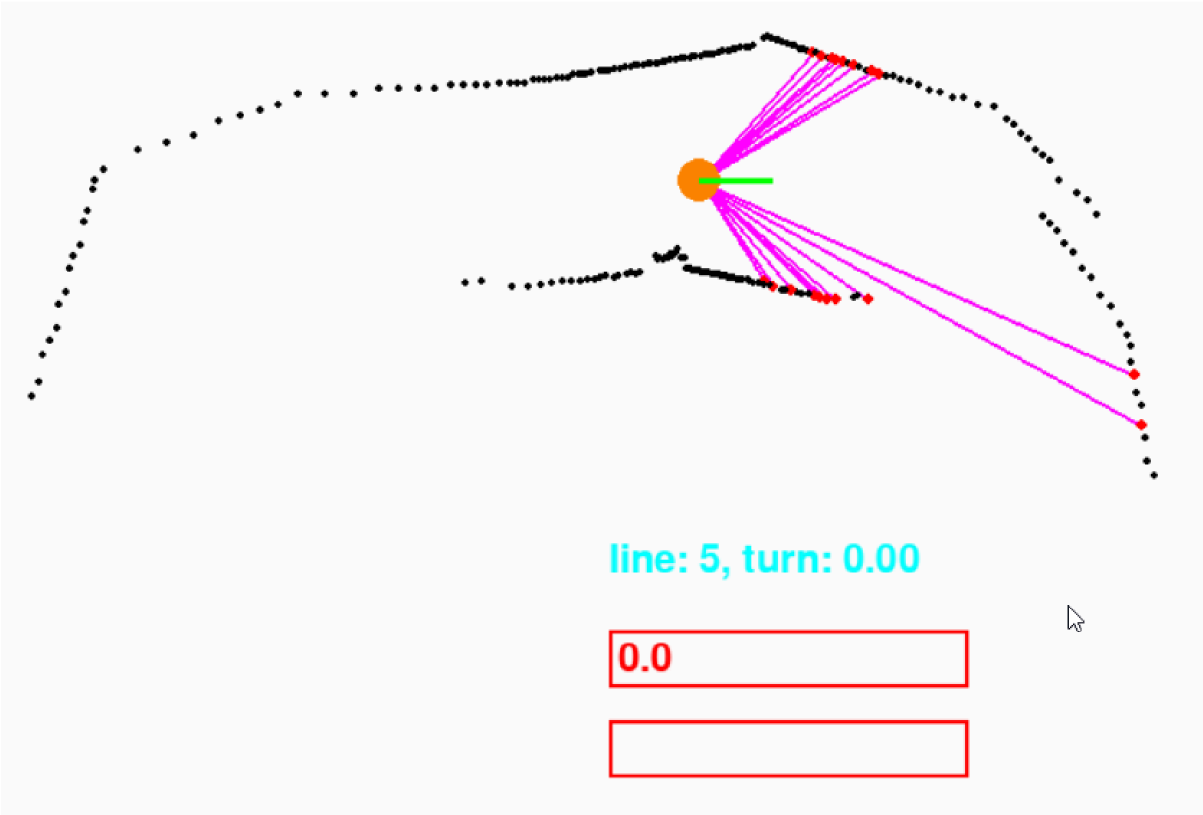
## 3.3. Data collection

Solution to use the lidar data to drive our autonomous car.

- Write CSV format of lidar data with the column index as angle (in degree)
- The last column is the TURN value



## 3.4. Data visualization



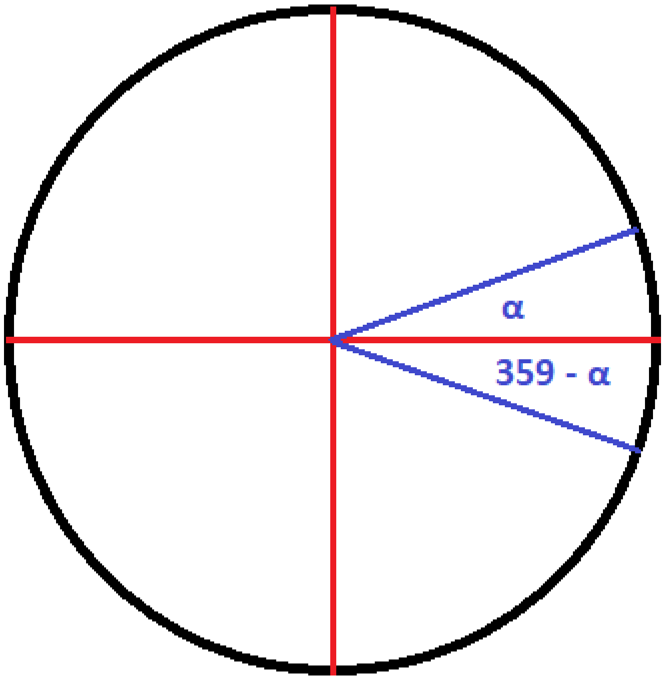

The original visualizer program is written by Nikodem Bartnik, checkout [his video here]

The data visualization program helps to:

- Show the surrounding obstacle to our robot
- Display the direction which robot is heading
- Display the TURN value which robot reacts to lidar data (required)
- Display the SPEED value which robot reacts to lidar data (optional)
- Allow to remove/modify the data point which is bad (this is because of manual data collection, driving does not always reflect good decisions).
- Create new data points by allowing the user to move the Robot position in any data frame then add corresponding SPEED, TURN (future improvement).

## 3.5. Train Machine Learning Model

- Data cleaning using the visualizer.py app
  Click on first text field to update the Turn value then press “Enter”
- Data augmentation

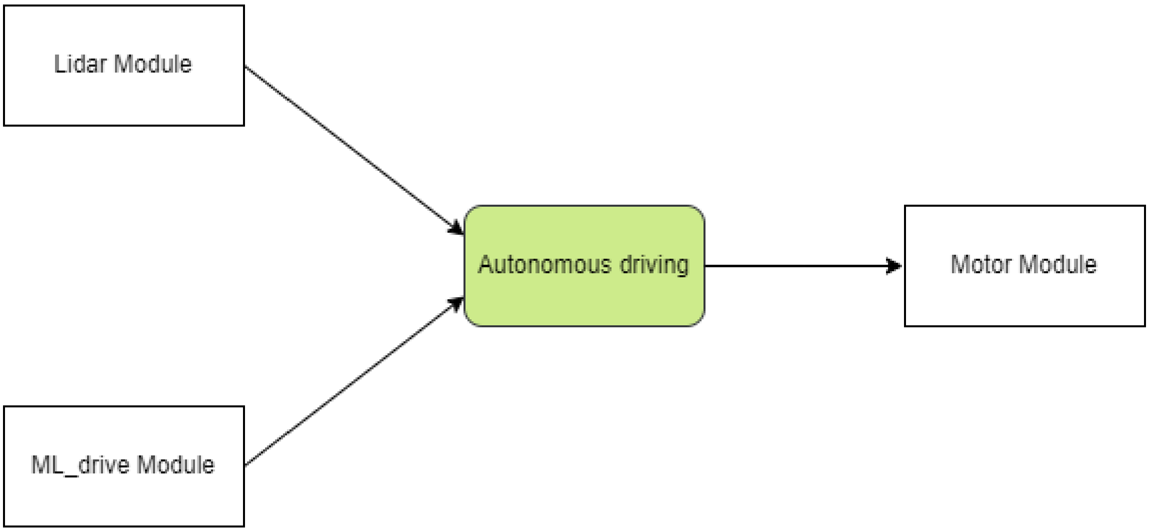



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| data | 0 | 1 | 2 | 3 | 4 | ... | 359 |
| augmented | 359 | 358 | 357 | 356 | 355 | ... | 0 |

Reverse the data, then negate the Turn value

Use Scikit-learn to train a regression model to predict TURN.
Develop a Jupyter notebook using Jupyter lab environment:
Regression model of target continuous TURN value (-1 to 1)
Loss function Mean Square Error
Calculate model accuracy with R square
Save the model using Pickle
Develop a Python module for self driving decision by loading the regression model and predict TURN by supplying Lidar Data from sensor.

### 3.6. Autonomous driving




### 3.7. Future improvement

- Can use multi-targets regression model to predict both SPEED and TURN.
- Using simulation to train
- Use reinforcement learning instead of regression ML model (develop reward function)
- Use ROS 2 to develop the software
- Combine Lidar with Computer Vision using Raspberry Pi Camera module: 2 separate ML models, the predicted output is sent REAL-TIME to controller Unit (node), it is up to controller unit to decide ACTUAL actuator outcome (signal to drive motors). The controller unit itself can be another ML model.

# 4. Demo

https://youtube.com/shorts/2FhzQcxH8CI?feature=share

# 5. Reference

Murtaza, Youtube How to run robot motors
Murtaza, Youtube How to run joystick with Raspberry Pi
Github, Robotica/RPLidar