Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

ROS2 Project: Robotic Car

Robotics 814
Report 1: Moving Without Control

Dylan Charl Eksteen
22623906

March 28, 2023

Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Private Bag X1, Matieland 7602, South Africa.

Plagiarism declaration

I have read and understand the Stellenbosch University Policy on Plagiarism and the definitions of plagiarism and self-plagiarism contained in the Policy [Plagiarism: The use of the ideas or material of others without acknowledgement, or the re-use of one's own previously evaluated or published material without acknowledgement or indication thereof (self-plagiarism or text-recycling)].

I also understand that direct translations are plagiarism, unless accompanied by an appropriate acknowledgement of the source. I also know that verbatim copy that has not been explicitly indicated as such, is plagiarism.

I know that plagiarism is a punishable offence and may be referred to the University's Central Disciplinary Committee (CDC) who has the authority to expel me for such an offence.

I know that plagiarism is harmful for the academic environment and that it has a negative impact on any profession.

Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully (acknowledged); further, all verbatim copies have been expressly indicated as such (e.g. through quotation marks) and the sources are cited fully.

I declare that, except where a source has been cited, the work contained in this assignment is my own work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/ assignment. I declare that have not allowed, and will not allow, anyone to use my work (in paper, graphics, electronic, verbal or any other format) with the intention of passing it off as his/her own work.

I know that a mark of zero may be awarded to assignments with plagiarism and also that no opportunity be given to submit an improved assignment.

Signature: .....

Name: ..... Student no: .....

Date: .....

Assignment Overview

This report is for the completion of the ROS2 project for Robotics 814. The project aimed to use ROS2 and python to make a robotic car drive in an s-shape.

This report is the first report of two that are required for the project and is limited to the uncontrolled movement of the car using ROS2. The second report covers the feedback-control implementation and s-shape movement of the car.

For the first part of the assignment, ROS had to be used to start the movement of the car by sending the same command to both motors. The subsequent movement of the robot is expected to not be in a straight line as there will be some variance in the motors. This movement should then be recorded in a way to clearly show that the car is not travelling in a straight line.

Submission Requirements

- A video of the robot moving.
- The code used for the project
- A short report that lists and explains key parts of the code.
- The ROS Computation Graph.

This report will thus list and explain key parts of the code, as well as a basic overview of the hardware implementation used for the project to serve as a reference for future work.

Software Overview

The code for the project was written in Python and interpreted using Python 3 interpreter. All of the information and instructions for the ROS part of the project are from an online course that was takes as part of the module. (Renard, 2023)

ROS Implementation

For the first part of the project, a single ROS2 node was used. Within this node, the following core functions are performed:

- The GPIO Pins are initialized.
- Timers are set to start and stop the car's movement.
- An PWM value is written to both motors when the movement starts.
- The PWM value of both motors is set to zero for movement to stop.
- The GPIO pins are cleared and the node destroys itself at the end of the program.

Figure 0.1 shows the ROS 2 graph of the single node that was used for the movement of the car.

Figure 0.1: ROS Graph of Simple Wheels Node

After the project was built using the *colcon* build tool, the node was started using the following command in the system terminal:

```
ros2 run my_py_pkg simple_wheels_node
```

Program Logic

General Overview

The program was written using object orientated programming (OOP) principles. Thus, all of the control logic could be placed in a single class and an instance of this class was created within the main loop as is shown in the following summary of the node structure.

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
import signal
import threading
import RPi.GPIO as GPIO

class WheelsNode(Node): ...

def main(args=None):
    rclpy.init(args=args)
    node = WheelsNode()
    rclpy.spin(node)

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

Class Structure

The WheelsNode is the class that is responsible for the simple movement of the wheels. The general structure of the class can be seen in the following overview of the methods that exist within the class:

```
class WheelsNode(Node):

    def __init__(self): ...

    def sigint_handeler(self, signum, frame): ...

    def cleanup(self): ...

    def start_wheels(self): ...

    def stop_wheels(self): ...
```

When the class instance is initialized, a timer is started with an initial delay of 10 seconds. When this timer expires, it calls the *start_wheels* method to start the movement of the wheels.

Upon starting the wheels, second timer with a delay of 12 seconds is started. This second timer will then call the *stop_wheels* method that stops the wheels and finally calls the *cleanup* method that clears the GPIO pins and destroys the ROS node.

The *sigint_handeler* method is called when the program is closed from the terminal. This method will then call the *cleanup* as before in order to properly exit the program and clear the GPIO pins before destroying the node.

Hardware Overview

This section is not inclued as part of the fulfillment of the assignment outcomes, but merely for work records and future reference.

The car that is used for the project is based on an inexpensive DIY robotic car kit. The components were all provided but needed to be assembled and connected. A Raspberry Pi 4B computer was provided with the necessary Ubuntu-based operating system and ROS2 software installed.

Materials Used

The following was used for the construction of the robotic car:

- Raspberry Pi 4B computer
- Battery/power bank
- DC-motor Driver
- IMU sensor module
- Jumper wires
- DIY car kit consisting of:
  - Chassis to mount all other components on
  - 2 x DC motors to power the movement of the car
  - 1 x Idle wheel for the front of the car
  - Mounts and fittings required to assemble the car

Hardware Configuration

The components were connected to the Raspberry Pi computer through the built-in GPIO pins. Table 1 below shows the pin configuration that was used for the project.

Table 1: Raspberry Pi GPIO Pin Setup

Pin Number	Connection	Function
16	Motor Driver: A1	Right Motor FWD
26	Motor Driver: A2	Right Motor BKW
5	Motor Driver: B1	Left Motor FWD
6	Motor Driver: B2	Left Motor BKW
5 V	Motor Driver: Vcc/Vm	5 V Power
GND	Motor Driver: GND	Ground
3	IMU sensor: SCL	I2C SCL
2	IMU sensor: SDA	I2C SDA
5 V	IMU sensor: Vcc	5 V Power
GND	IMU sensor: GND	Ground

Figure 2 and Figure 3 below show the car that was assembled and used for this project. Figure 2 clearly shows the Raspberry Pi, Motor Driver and Power Bank that were used. In Figure 3 the IMU sensor and two DC motors and Idle wheel can be seen mounted on the bottom of the car.

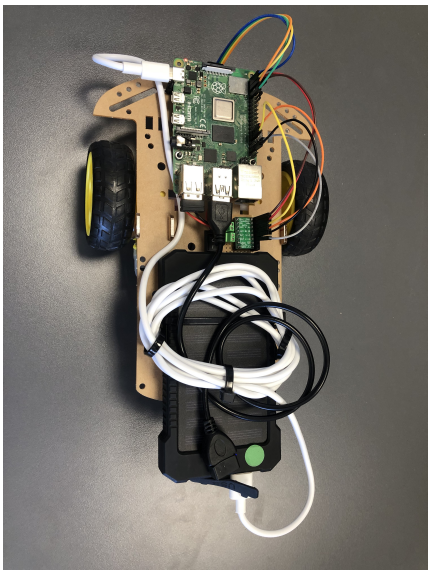

Figure 2: Top view of Car

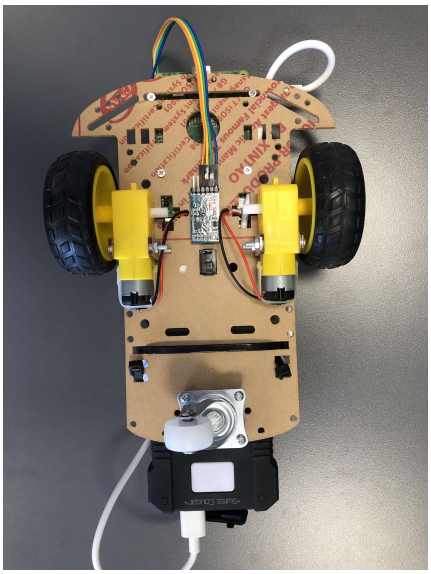

Figure 3: Bottom view of Car

Conclusions

The first part of the project serves to demonstrate the use of ROS and to set up a simple node to make the robot car drive forward. This was achieved with the method that was demonstrated in this report.

Upon starting the movement of the car, it would consistently pull to the left at a constant rate. This is, as mentioned, due to the inconsistency between the two DC motors. This inaccuracy proves the need for the feedback control that is to be implemented using the IMU sensor and is covered in Report 2 for this project.

List of references

Renard, E. (2023). Ros2 for beginners (ros foxy, humble - 2023) | udemy. Available at: https://www.udemy.com/course/ros2-for-beginners/learn/lecture/21305292?start=135#overview