WHEELED ROBOT ROS IMPLEMENTATION

By Jaime Bohorquez

Introduction

The Tesla electric cars were the inspiration for the development of this wheeled robot. I felt that this project was a nice way to implement a similar system in a much more manageable scale.

The system which more closely resembles a standard remote-control car than a traditional robot, is built using robotic software architectures along with the respective hardware. This separates the system from your standard remote-control car since this system can be scaled to do just about anything.

I wanted to implement a teleoperated robot chassis as it seemed the ideal project for a couple of reasons. I wanted to first get familiar with ROS and implementing the Python programming language in a real-world setting. Second, I wanted to learn more about the hardware and mechanical parts needed to get a robot built. Lastly, I wanted to have a project that was scalable for future additions, which in this case there are many.

Method - Software

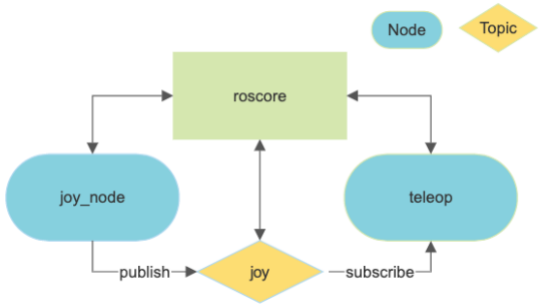
To implement the software side of the robot, I used a Raspberry Pi 4B+ running Ubuntu server 18.04.5. On this platform, ROS Melodic Morenia was used as the architecture. Since the robot wasn't very complex, it took the usage of just 2 nodes and a topic to communicate between the two nodes.

The first node was the joy_node which is an included node in the ROS built in library. The function of this node is to read joystick data (in which case we used an Xbox One Controller paired via Bluetooth) and published said data in a formatted manner to the joy topic.

The second node is the teleoperation node which I implemented using Python. This node subscribes to the joy topic and gets the joystick data.

The data received from the node, which is represented using a floating point between zero and one, gets processed into a PWM signal of a 1 to 2 millisecond activation regions of a 20-millisecond pulse wave. One millisecond would send the robot fully in reverse, 2 milliseconds would send the robot fully forward, and in between we are able to vary the voltage percentage sent. At 1.5 milliseconds our robot would rest as the output signal would be zero. The PWM signal then gets sent to the motor controllers of the robot.

Diagram A – ROS Node and Topic Setup



Method - Hardware

In terms of physical parts, I tried to use parts of electronics I had laying around and bought the rest online.

To meet the power demands, the robot is powered using a Vex 7.2 Volt battery (rated at 3,000 mAh) which is able to supply the drive and turning motors of the robot. There is a secondary 5 Volt battery that powers the Raspberry Pi 4B+.

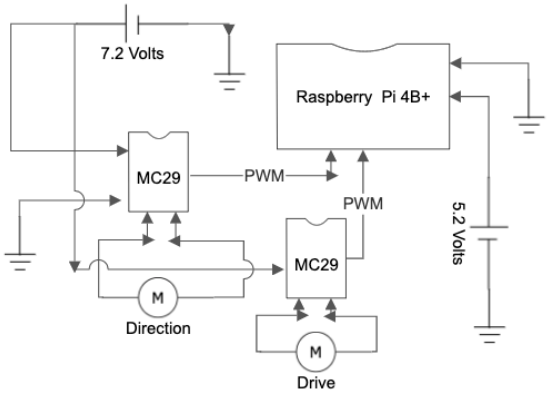
To process the computer output (PWM signal) and power, I used two Vex Motor Controller 29 boards (MC29). These were rated for 7.2 Volts at a maximum of 5A, which does not exceed the power demands of the motors.

The motor used for guiding the robot in the correct direction is a standard hobby DC motor, while the drive motor is a more powerful 7.2 Volt rated DC motor.

For robot turning the original plan was to use a servo which are much more accurate in terms of getting a

correct angle for the robot to turn. However, these never arrived, and I had to settle for using a standard DC motor.

Diagram B - Robot Schematic



Note: Please ignore arrow ends in the wire connections, they do not have special meaning.

On the mechanical side of the system, I was able to mostly reuse the parts of an old remote-controlled car I had laying around. These included the transmission that converts the drive motor speed into more torque to be able to drive the robot. The robot steering and general chassis structure was also inherited.

Discussion

Before talking about robot operation, it is important that most of this can be automated using a startup script. Sadly, I was unable to get the script to start up automatically.

To utilize our robot system, we first need to connect our robot batteries. Once we do that the robot starts automatically. When the computer is on, we then can connect by connecting a display to the robot or most commonly, by connecting remotely using SSH.

Once we are online, we run the scripts that are shown on the video to enable Bluetooth and the GPIO header of the Raspberry Pi. We then are able to turn on our Xbox controller that pairs automatically. Lastly, we start roscore and the rest of the necessary nodes.

Our system is setup so that we can drive our robot just like any other car on a video game. We have the right throttle for acceleration, the left throttle for braking and reverse, and the upper leftmost joystick to turn the robot.

The main issue with this robot system was the intervals of time where lag between joystick input and robot action was very noticeable and something that is very undesirable on a real time system such as this.

Although the problem was relatively rare, I explored likely reasons that could be attributed to the lag. These included too much data being passed through the topic and connection problems or delays in the Bluetooth. However, I was unable to pinpoint this lag problem.

Another issue worth mentioning was the noticeable inaccuracy of the steering due to the fact that the voltage decreased as the robot was being used therefore causing changing steering angles as the robot was used.

Conclusion

Being able to get this system working was something that I doubted could be completed since a lot of these things were very new to me. Now that the robot is functional, I am able to see that there are many ways to improve this system.

In summary, what I was able to do in this project was to create a robot platform that can in the future be equipped with a variety of other robot subsystems to make it more functional.

One route I would like to explore more with this robot is the ability to make the robot autonomous. I personally thought that making the robot be able to drive itself through sidewalks would add a lot of functionality to the robot. One of the ways I would implement this idea would be using stereo vision along with a Neural Network so that the robot is able to learn to be autonomous.

This robot could also see the implementation of a robotic arm to be able to pick up objects and transport them around. Even potentially of doing both the arm and being able to automate the arm and the robot driving.

Overall, I am hoping to improve this robot system over the winter break to learn more about other robotic system; as well as applying more robotic automation algorithms to this robot.