

CMPSC 497: Special Topics (Spring 2024)

Final Project: Autonomous Mobile Robot Car

Jack Demtshuk

Jon Cunningham

Chris Cleus

Ishaq Halimi

Computer Science

Instructor

Dr. Vinayak Elangovan, Ph.D.

Program of Computer Science

Submitted On: 5/5/2025

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| 1 | INTRODUCTION | 3 |
| 2 | BACKGROUND | 3 |
| 3 | DEVELOPMENT | 4 |
| 4 | PERFORMANCE | 5 |
| 5 | CONCLUSION | 6 |
| 6 | REFERENCES | 6 |

* **INTRODUCTION**

This project involved programming and testing a robot car to make it run autonomously. The car is able to use its many sensors and additional hardware to navigate a custom-built track all on its own without user input. The car uses an IR sensor to detect what is and isn't track, an ultrasonic sensor to detect when an obstacle is too close, and an additional Raspberry Pi camera to identify road signs such as stop signs, yield signs, speed limit signs and perform corresponding actions when identified, such as stopping, slowing down, and speeding up. All of this is accomplished through autonomous communication between the robot car's Arduino, and the camera's Raspberry Pi.

* **BACKGROUND**

The Freenove robot car that we received came equipped with multiple different features that were all used in our project. The most important is the two-board Arduino that provides power and instructions to the car to allow it to perform its functions. The car also came equipped with two major sensors, the IR sensor and the ultrasonic sensor.

The IR sensor is located on the bottom of the car and has three different input pieces. These pieces each individually detect what is beneath the car as it drives. We used the sensors to allow the car to identify when it is and isn't on the track we designed. The car is able to sense the track, which is black with white outlines, using this IR sensor. When it detects black, the car will move, and when it detects white, it will not. When one piece detects white and the others detect black, this will inform the car to turn.

The ultrasonic sensor is located on the front of the car and is able to send an ultrasonic wave out and accept it's rebound back. Doing this allows the car to determine how far away an obstacle is from itself. We used this sensor to allow the car to have an "emergency brake" feature, that will cause it to automatically stop if it detects an obstacle is too close.

Our final and most important addition to the robot car is the Raspberry Pi and its camera. The Raspberry Pi is able to create a live feed of the connected camera, and using OpenCV, can analyze and identify signs that enter the camera's view. We programmed Raspberry Pi to be able to identify stop signs, yield signs and speed limit signs. Using the serial monitor of the Arduino, we can send messages from the Raspberry Pi to the Arduino, which will in turn allow the Arduino to command the robot car to perform certain actions when the Pi tells it to. This is the system and communication we established between the Arduino and the Pi in order to command the car to stop, slow down or speed up when corresponding road signs are identified by the camera.

* **DEVELOPMENT**

The development process of our project began as soon as we received the robot car. The first thing we did was analyze what materials we would need to complete our project, and we determined what Raspberry Pi camera we would need and asked the professor to order it for us. As we waited for the camera's delivery, we began work on preliminary code for the robot, to test its basic movement functions and operation.

Using the sample code provided by Freenove as a base, we created a basic functioning code to test the car's movement using an IR remote. This remote sent an IR signal to an input sensor on the top of the car, which would allow us to move the car manually using the remote. However, our project requires the car to be autonomous, and this was only for basic demonstration and testing. Unfortunately, as we began to develop the autonomous code for the car's Arduino, we ran into a problem.

For some reason that alluded to us for quite some time, the robot car which functioned perfectly fine with the IR remote before stopped working. Despite the fact that we could tell through LED lights and the serial monitor in the Arduino that the car was indeed receiving input from the remote, the car would not move at all. After trying everything we could, we ended up taking the entire car apart and testing the motors of each wheel individually, as this was the only cause of the problem we could think of. When we confirmed that the motors did indeed work correctly, we eventually solved the error, which ended up being a wiring issue that prevented the motors from receiving the necessary power.

When the Raspberry Pi camera finally came in, we began working with it in order to establish a live video feed. At first, we needed to use a keyboard and monitor in order to connect with the Pi, but eventually we managed to establish a wireless connection to a laptop. At first the camera's feed was rather rough, and would be zoomed in too far into a corner and not show a full view of what is in front of it. Eventually, we fixed this issue by messing with the camera's size settings and the system it was running through. Using OpenCV and training with Cascade files, we were able to train a program to identify a stop sign. We then used this same method to train the program to first identify right turn and left turn signs. Eventually, we determined that the turn signs were too complex, and the training data was not accurate enough, so we abandoned the turn signs in favor of a yield sign and a speed limit sign. We once again used the Cascade files to train the program to identify yield signs and speed limit signs.

With the basic programming of the Arduino and the Pi almost complete, we began to construct the track that the robot car would run on. We took black paper and white duct tape and created track shaped like the letter P. The car managed to run this track somewhat well, being able to identify what is and isn't track using it's IR sensor to see if it is on black paper or white tape. Unfortunately, the car was not able to follow this track completely accurately and suffered a lot of errors. We decided to start over with a new track, which was smaller and more manageable. While still shaped like the letter P, we decided to make the track more angular rather than curvy, as the first track was. This way, the car could make turns easier and quicker, as well as not accidentally veer off the track when it hits a turn too quickly.

While the basic Arduino code managed to run the car across this track well enough with minimal issue, when we began to implement the Arduino-Pi communication code, something went wrong. The car began to act awkwardly and seemingly made wrong turns and wrong decisions when presented with certain obstacles, such as turning when it detects something too close rather than stopping, or making left turns on a right turn when it had no problem doing so before. This was a confusing and shocking change in the car's behavior, as before this, the car seemed to be working perfectly fine. It was following the track correctly with only minimal problems and was successfully identifying and reacting to obstacles and road signs.

At this point, we decided the best way to go was to rewrite the Arduino code from scratch, as our current Arduino code did not manage to mesh well with the Raspberry Pi code. The new Arduino code was able to keep the same basic functionality of the autonomous robot car, while also successfully implementing the Raspberry Pi communication and serial monitor actions through sign identification. With the code completely finished, we simply set up the track with the signs and a second robot car as an obstacle, and the project was finished.

* **PERFORMANCE**

The robot car’s performance is rather well, although not entirely perfect. When the car runs along the track, it successfully follows the track by using its IR sensor to identify when to make turns as it hits the edges of the track. Sometimes, when the car hits an edge too close, it will cause it to make many small turns rather than a large turn and readjustment. This makes it so the car successfully turns, but it is rather jerky and slow. Sometimes, when the car hits the edge of the track at an awkward angle, it will cause the car to hit the edge with all three IR sensor nodes, and make it stop completely without turning.

When the camera manages to detect one of the road signs, it will send a serial monitor message to the car’s Arduino in order to perform corresponding actions. When a stop sign is detected by the camera, it will send “STOP” to the Arduino, which triggers the car to immediately stop, and pause for 3 seconds. After these 3 seconds are up, the Pi sends a “GO” message that causes the car to continue on as normal, just as a normal car would do at a stop sign. When the car identifies a yield sign, the Pi will send a “YIELD” message to the Arduino, which causes the car’s speed to drop by 10. Once the yield sign has left the viewpoint of the camera, the serial monitor receives a “SPEED UP” message which returns the car to normal speed. When the camera identifies a speed limit sign, it will receive a “SPEED LIMIT” message, which increases the car’s speed by 10. This increase in speed is a permanent increase that does not revert at any point, in order to show the difference between how the Arduino and Pi can interact.

We decided to use a second robot car as an obstacle for the main car to identify using its ultrasonic sensor. Demonstrating this by placing the spare car on the track in front of the main car, the main car’s ultrasonic sensor will identify when the obstacle is too close and will immediately stop in an “emergency brake” system. The car will not continue or resume movement until this obstacle leaves the minimum distance. Once this obstacle does leave the distance, which is done by controlling the second car with the IR remote, the car will resume normal function after 3 seconds.

* **CONCLUSION**

This project has been one that has challenged us to implement all of the skills and techniques that we had learned over the course of the semester and combine them all into one project. Using the Arduino, Raspberry Pi, OpenCV, and the way that all of these aspects communicate and interact with one another is a perfect way to summarize and demonstrate all that we have learned in this Special Topics course. This class and project were challenging, but very educational. It provided us with an insight into what kind of work computer scientists are tasked with, and the ways that they have to manage working on and handling them, both with time and teamwork.

* **REFERENCES**

[**https://github.com/Freenove/Freenove\_4WD\_Car\_Kit**](https://github.com/Freenove/Freenove_4WD_Car_Kit)

[**https://projects.raspberrypi.org/en/projects/getting-started-with-picamera**](https://projects.raspberrypi.org/en/projects/getting-started-with-picamera)

[**https://www.raspberrypi.com/documentation/computers/camera\_software.html**](https://www.raspberrypi.com/documentation/computers/camera_software.html)

[**https://chatgpt.com/**](https://chatgpt.com/)

[**https://www.mathworks.com/help/simulink/supportpkg/raspberrypi\_ref/detect-stop-signal-traffic-sign-using-raspberry-pi-and-simulink.html**](https://www.mathworks.com/help/simulink/supportpkg/raspberrypi_ref/detect-stop-signal-traffic-sign-using-raspberry-pi-and-simulink.html)

[**https://github.com/maurehur/Stop-Sign-detection\_OpenCV**](https://github.com/maurehur/Stop-Sign-detection_OpenCV)

[**https://freenove.com/tutorial**](https://nam10.safelinks.protection.outlook.com/?url=https%3A%2F%2Ffreenove.com%2Ftutorial&data=05%7C02%7Cjfd5838%40psu.edu%7Cb3ac12df0ad94a5e557608dd8ba14172%7C7cf48d453ddb4389a9c1c115526eb52e%7C0%7C0%7C638820246879129137%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=IZ853FrBMEg2oClaIgF2M1sMujnUQHnwLzhLE%2FrhXLA%3D&reserved=0)

[**https://forum.arduino.cc/t/freenove-tracking-line/856886**](https://forum.arduino.cc/t/freenove-tracking-line/856886)

[**https://www.geeksforgeeks.org/detect-an-object-with-opencv-python/**](https://www.geeksforgeeks.org/detect-an-object-with-opencv-python/)