CSC 615 Term Project Report

RASPBERRY ROLLERS

Primary Github: gsnilloC

Members:

Collins Gichohi

Diego Flores

Yakoub Alkabsh

Mohammad Dahbour

Task Description

Our task was to design and build a small autonomous car controlled by code on a Raspberry Pi. Using different sensors this car can navigate a course by following a dark line on the ground. Additionally, it has the ability to detect obstacles, maneuver around them, and then realign itself to continue following the line.

What Worked Well

Despite the challenges, many aspects of our project worked well. One of the major successes was our team's collaboration. Everyone in the group contributed ideas and solutions, which made troubleshooting and problem-solving much more effective. The communication and cooperation among team members was outstanding, helping us to quickly address any issues that arose.

Another success was the car's ability to follow the line consistently once we balanced the weight properly and adjusted the sensors. The single motor hat setup with the ball bearing for the rear wheel proved to be a great solution, providing stability and agility. The modifications we made to improve turning, such as adjusting the sensor placement and programming the opposite wheel to turn in the opposite direction, greatly enhanced the car's agility, performance, and most importantly consistency.

Additionally, our approach to handling obstacles worked well. By making the turns slower and softer, we were able to keep better track of obstacles and guide the car back to the line smoothly. Once we resolved the thread management issues for the obstacle sensor, the car was able to detect and navigate around obstacles effectively.

Issues

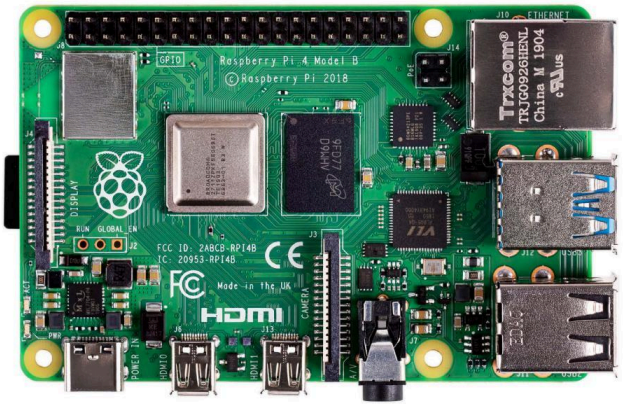
During the development of our car, we ran into several issues that required some problem-solving. At first, we had problems with using multiple motor hats, which seemed too complicated for us so we decided to simplify by using just one motor hat and added a ball bearing to the rear wheel to keep it stable. Finding the right weight distribution was also tricky; if the weight wasn't balanced correctly, the car would stray off its path, so we had to carefully adjust it for consistent performance.

Turning was another big challenge. The car often over-adjusted during turns, causing the sensors to lose the line. We fixed this by moving the sensors closer together at a specific angle. The car also struggled with sharp turns, so we made it more agile by programming the opposite wheel to turn in the opposite direction. When we added

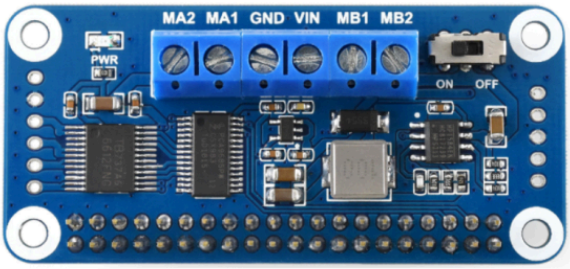
obstacle detection, we initially had trouble managing the threads, which caused synchronization issues. The issue with the threads ending up being an infinite while loop, once we added some conditions that will end the loop the threads worked as intended. The threads helped to guide the car around obstacles and back to the line, we made the turns slower and smoother, allowing better control and tracking. These adjustments helped our car reliably follow the line and navigate around obstacles.

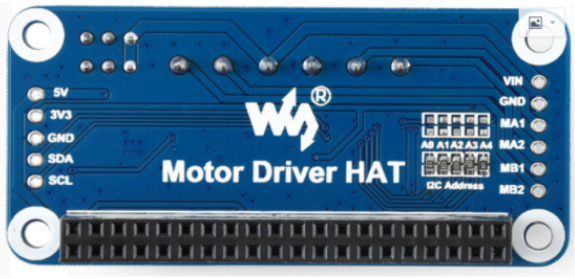
Parts / Sensors Used

- Raspberry Pi

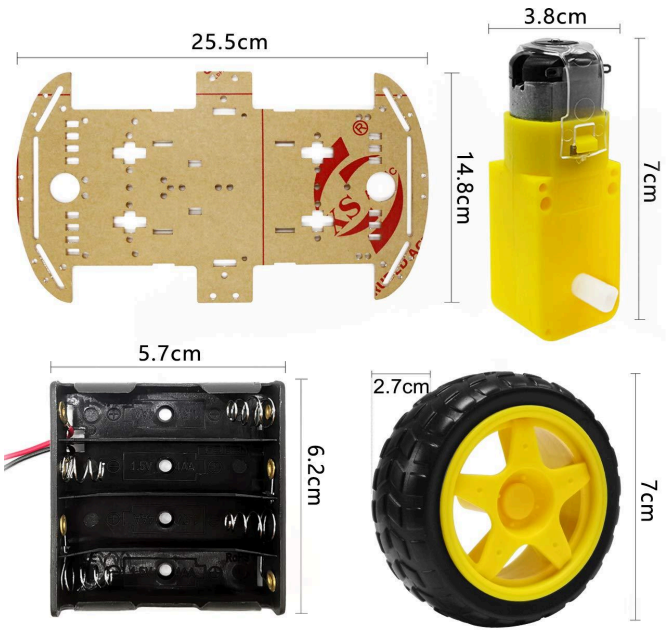


- Waveshare Motor Hat

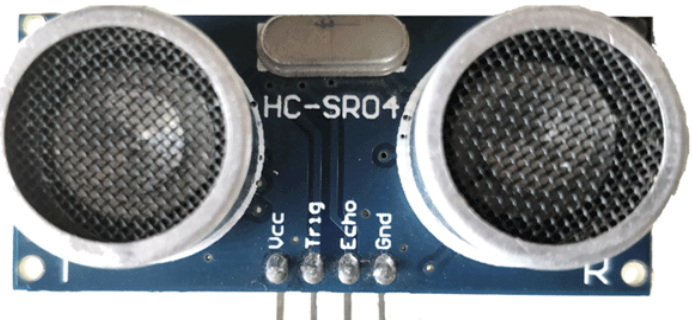




- Car Kit and Motor

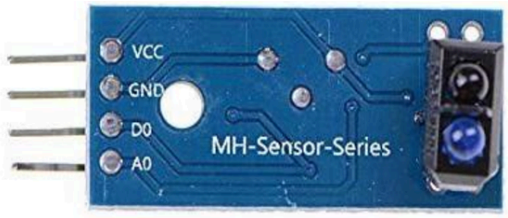


- HC-SR04 Echo Sensor

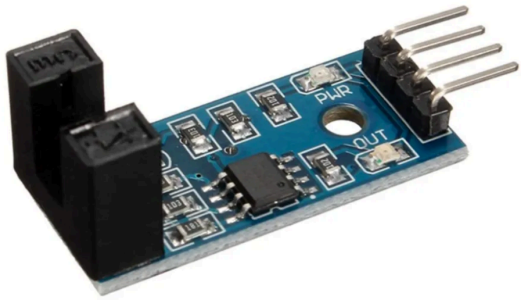


z

- Line Sensor(TCRT 5000)



- Wheel Speed Sensor



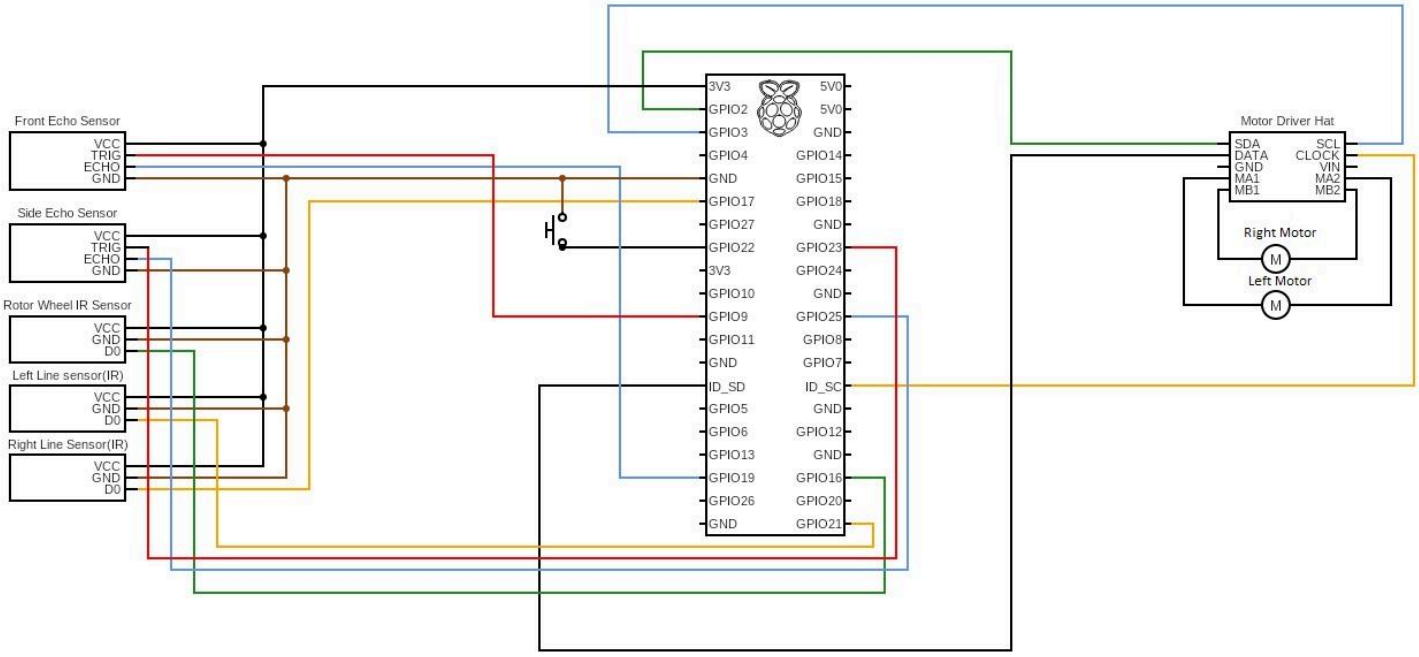
Pin Assignments

- Left Line Sensor: 21
- Right Line Sensor: 17
- Front Obstacle Echo: 19
- Front Obstacle Trig: 9
- Side Obstacle Echo: 8
- Side Obstacle Trig: 23
- Wheel Speed Sensor 16
- Button: 22

Libraries and Softwares Used

- PiGPIO
- PCA9685

Hardware Diagram



A bigger version of the diagram can be found on GitHub.

Flowchart of your Code

The flowchart can be found in README.md on GitHub.

Building the Car: