CSC615-01 Group Term Project
Data Pi-rats:
Team Report

Team Members:

Elisa Chih

TJ Layug

Kenny Leong

Cameron Yee

Primary GitHub:
krleong: https://github.com/CSC615-2022-Fall/csc615-term-project-krleong

Task Description

In this project, our group was to build an RC car robot using hardware parts of our choice that was to move independently along a track course. The car incorporates motors and sensors to follow a black line of tape that indicates the path of the track. The car must finish the course, sticking to the path while avoiding/navigating around any potential obstacles encountered on the track.

Building the Robot

Parts Used

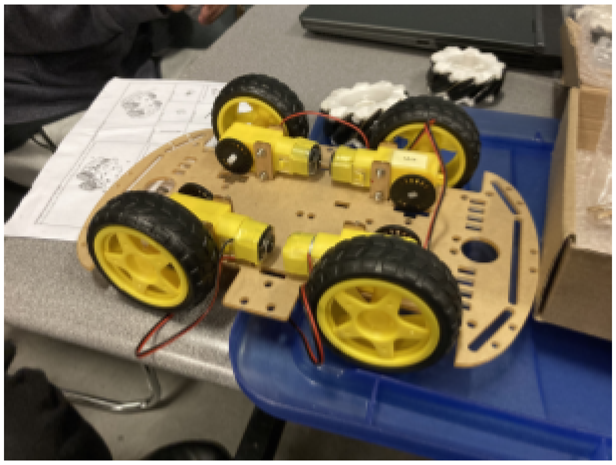
1. RC car chassis
2. Omnidirectional wheels x4
3. Line sensor with TCRT5000 reflective optical sensor x2
4. IR infrared obstacle avoidance sensor x3 (1 not in use)
5. Echo sensor x1
6. Raspberry Pi 4 Model B
7. GPIO extender
8. Waveshare Motor Driver HAT x 2
9. Portable charger

How the Bot was Built

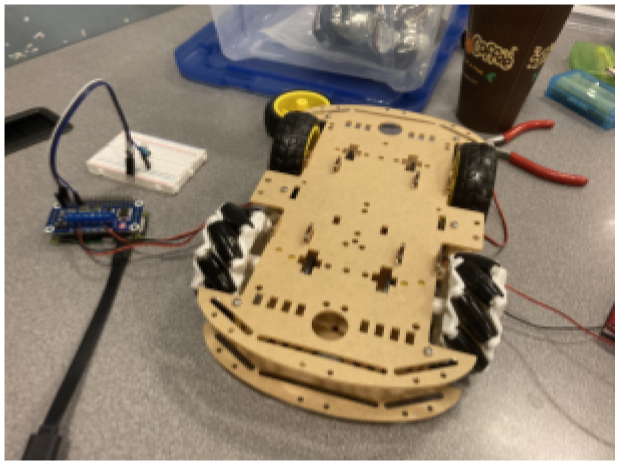
Our approach in simple terms:

1. Assemble chassis
2. Add wheels
3. Add motors and sensors
4. Program motors
5. Program sensors
6. Test
7. Fix bugs/shortcomings discovered from testing
8. Swap some obstacle sensors for an echo sensor for obstacle detection when drifting
9. More testing
10. More debugging
11. More testing
12. Finished product

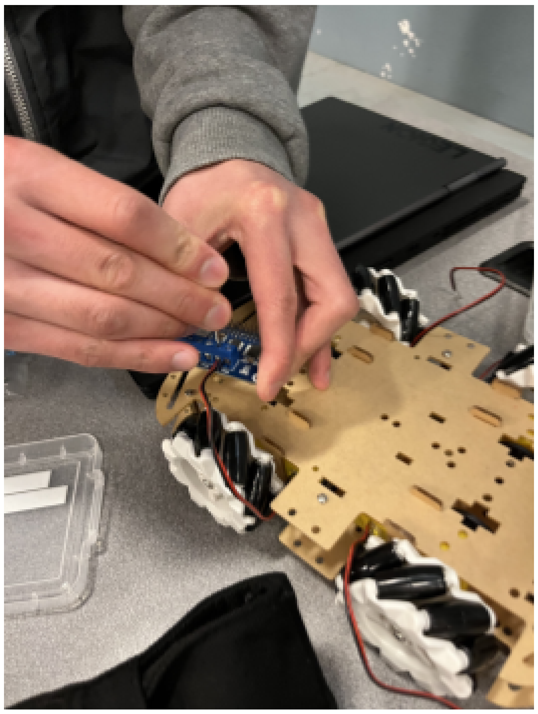
Pictures Documenting Building Process



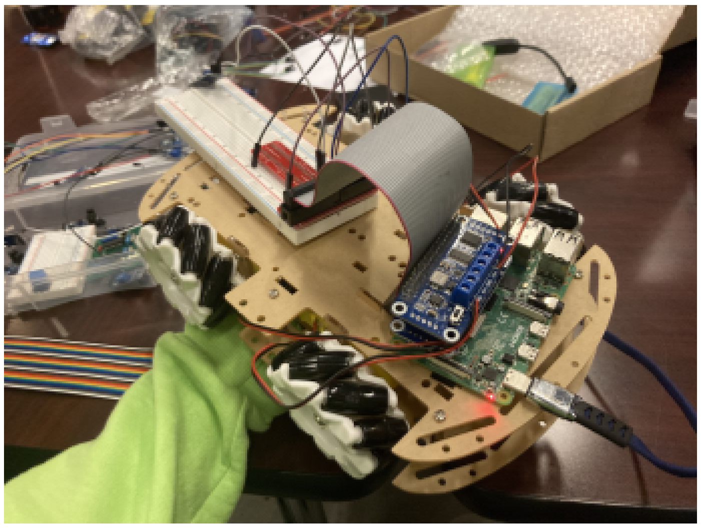
Motors mounted to the chassis with original wheels connected



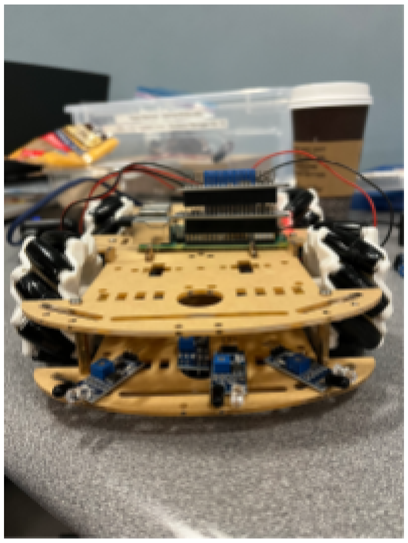
Chassis top attached and changing out the stock wheels to omnidirectional

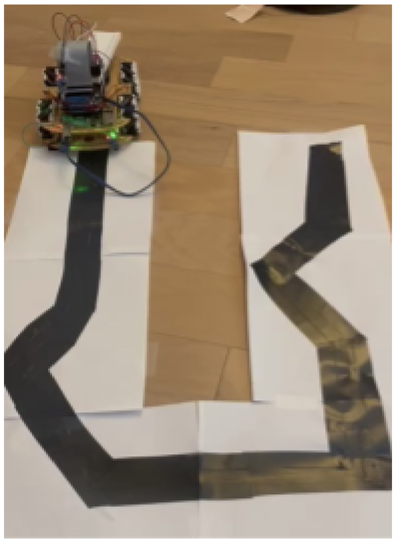


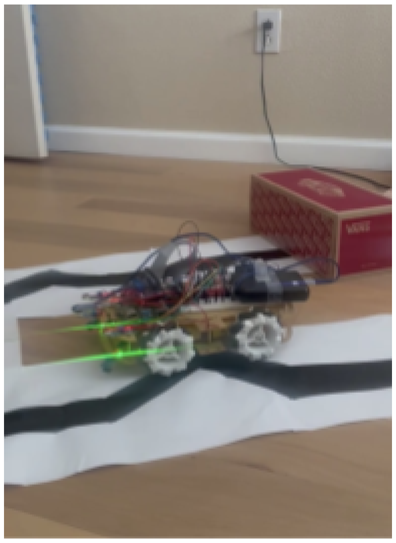
Wiring up the motors to a motor HAT



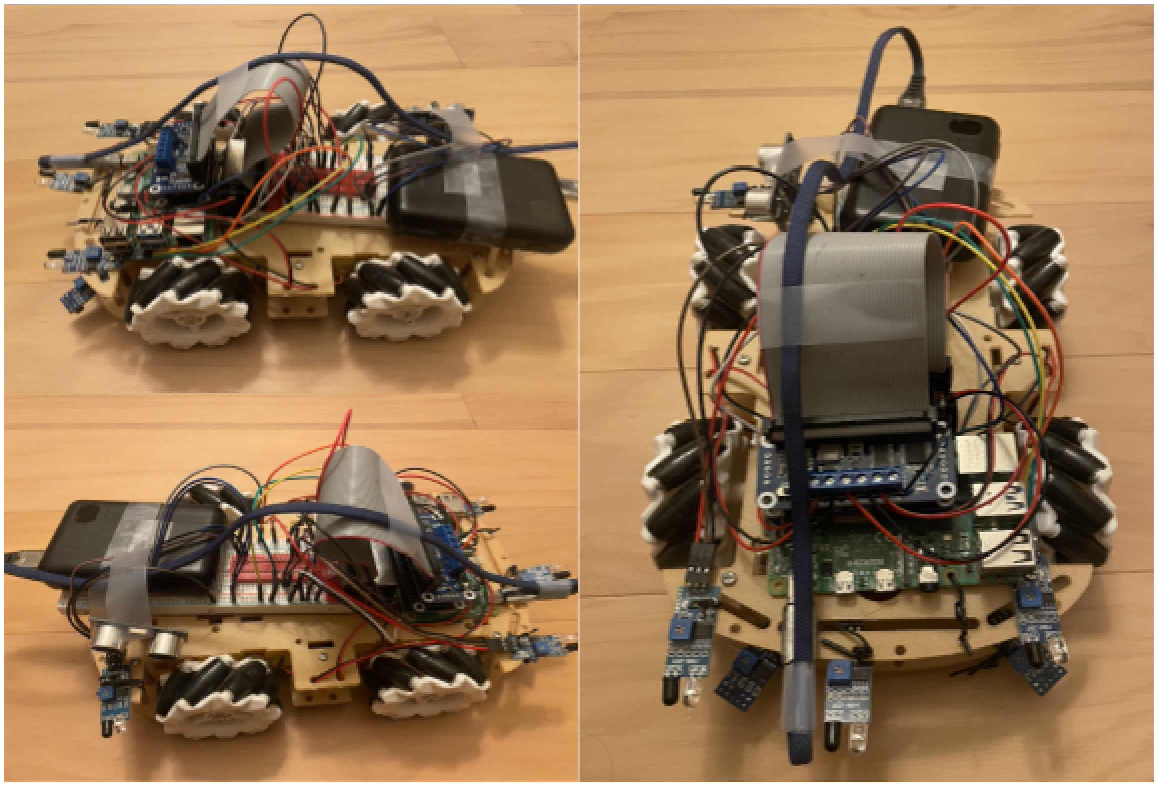
All four motors hooked up to Pi motor hats via GPIO extender


IR and Line sensors added to front bumper


Initial testing of our car on a homemade track


Testing drifting when obstacle is encountered

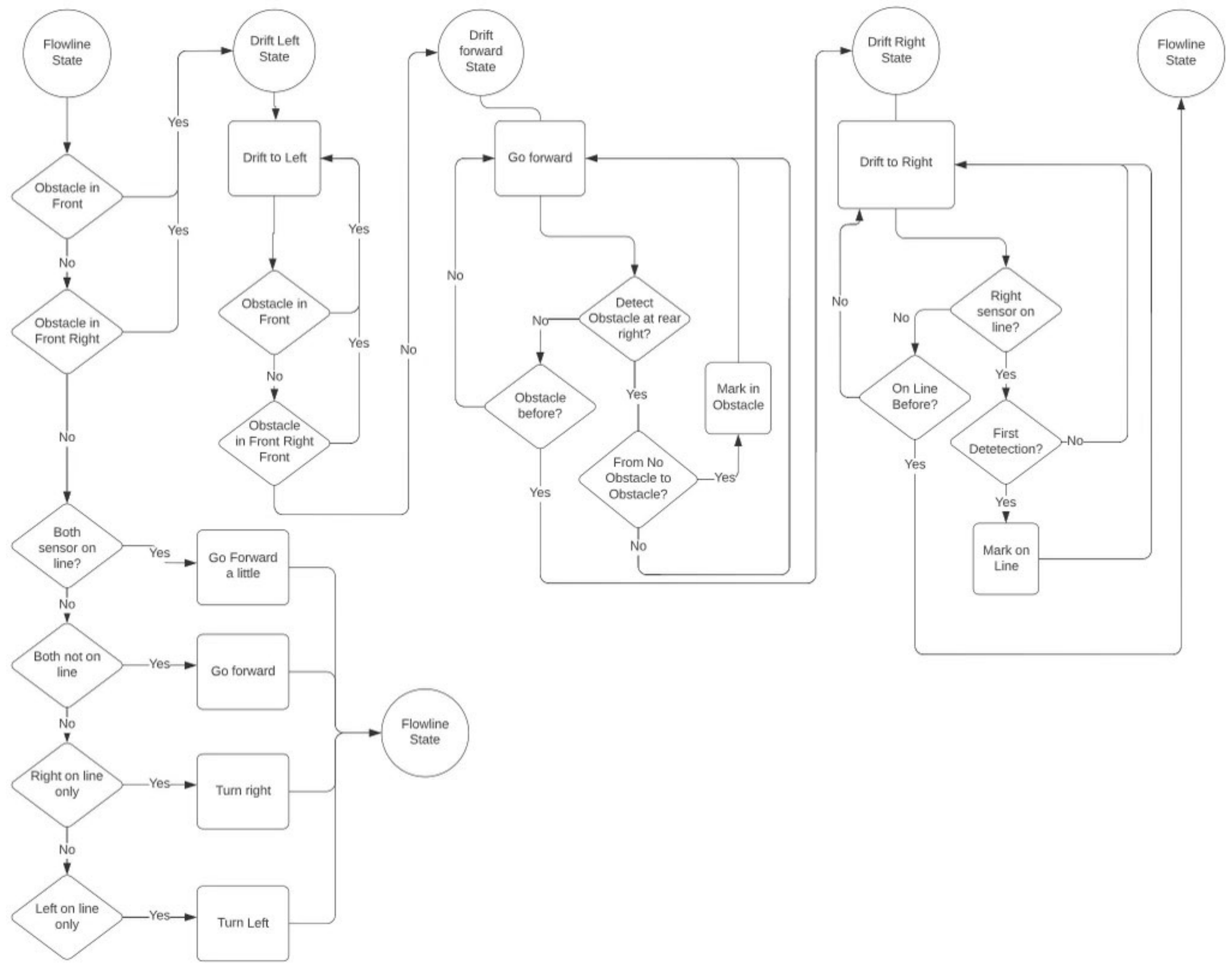
Finished Product



Libraries/Resources Used

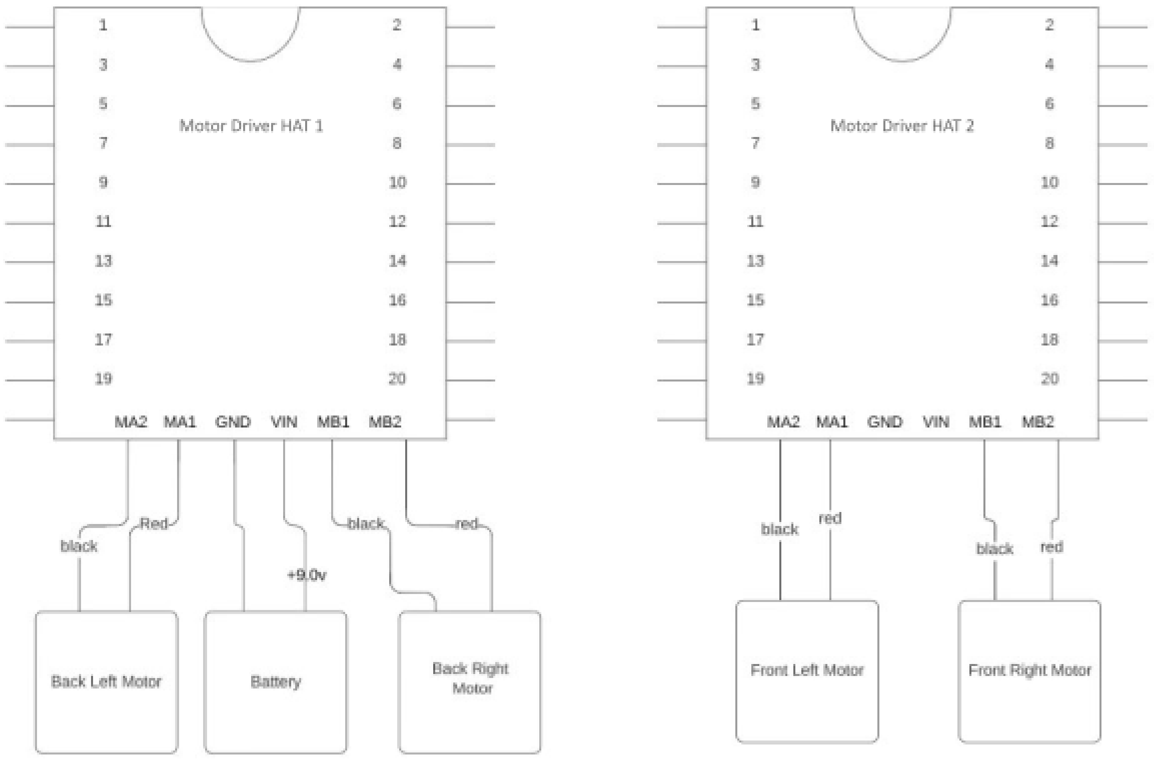
1. PiGPIO Library - https://abyz.me.uk/rpi/pigpio/
2. Waveshare PCA9685 motor driver - https://www.waveshare.com/wiki/Motor_Driver_HAT
3. Raspberry Pi GPIO Direct register access - https://elinux.org/RPi_GPIO_Code_Samples#Direct_register_access

Flowchart of Our Code/"What the car was thinking"

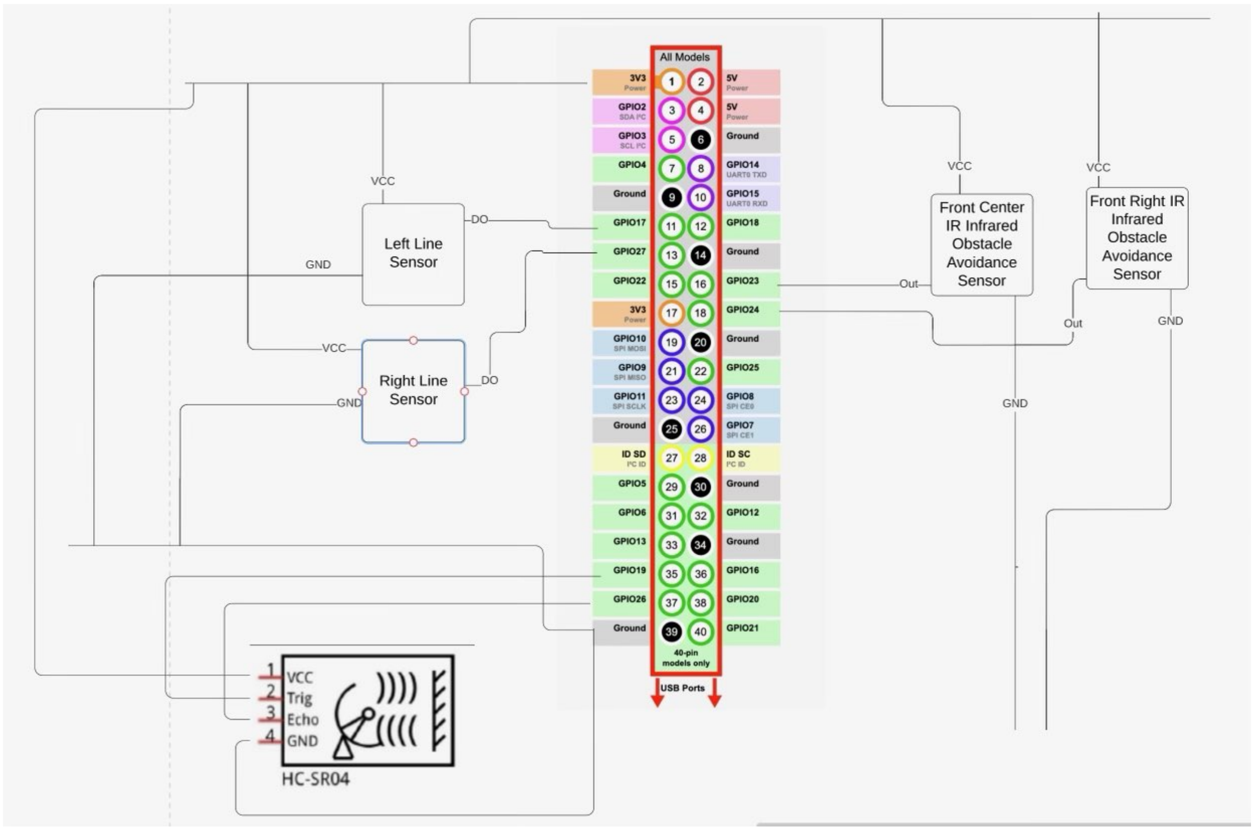


Pin Assignments Used

1.	OBSTACLE_RR_GPIO	21
2.	OBSTACLE_CENTER_GPIO	23
3.	OBSTACLE_RF_GPIO	24
4.	LINE_LEFT_GPIO	17
5.	LINE_RIGHT_GPIO	27
6.	REAR_RIGHT_TRIGGER_GPIO	19
7.	REAR_RIGHT_ECHO_GPIO	26



Hardware Diagram



What Worked Well

- Most of the code was adapted from previous assignments, saving a lot of time.
- We drastically reduced the number of sensors used in our final design compared to our initial design.
- We were excited to get the omnidirectional wheels to work since we thought those were cool and wanted to incorporate them into our car from the start.
- Despite difficulties in planning additional time outside of class to work on our car, we were able to work remotely and get it done by leveraging effective team communication.
- It works!

What Were Issues

1. The Line and IR sensors weren't reading properly when using the code in the program made for the car, which consists of a combination of parts of previous assignments for their respective hardware components. However, the sensors worked when testing the sensors using purely the code from Assignment 4 (Follow the Line but Stop). So we

modified the code to use the PiGPIO Library instead of direct register access, which we believe was the problem.
2. Some of the motors were put on the chassis the wrong way and in the wrong orientation. So instead of taking the entire chassis apart and flipping the motors, we simply rewired them since some wheels were spinning in the wrong direction.
3. The front and back wheels were put on backward, preventing the car from drifting properly. Drifting worked once the wheels' directions were corrected.
4. The car had difficulties making 90-degree turns. When trying to make the turn, both Line Sensors would be on the line, causing the car to be stuck in a state of correcting itself in opposing directions. A fix was made where the car automatically moves a small distance forward to be in a position with the sensors off the line.
5. The car had difficulties when encountering large obstacles with its single Obstacle sensor. To help with this, we added an Echo Sensor which has a farther detection range than the Obstacle Sensor sensors. This extended range allowed for better obstacle detection and helped the car drift avoid obstacles successfully, especially when the car would go crooked when determining if the car could drift back onto the line or not.
6. Shortcoming: The car is only able to drift and detect obstacles to the left.
7. It was difficult to plan a time for everyone to meet at the same time and work on the car outside of class due to schedule conflicts and other courses. This led to teammates taking the kit home on their own and working on it by themselves until reconvening with the rest of the group after the work had been completed.

Appendix

Code files used in our project:

1. main.c
    a. Contains the main driving logic for the sensors and car as a whole
2. main.h
    a. Includes C library dependencies
3. motorcontroller.c
    a. Contains the main driving logic for the motors
4. motorcontroller.h
    a. Includes motor driver files and DEV_config files
5. Library files for Motor HATs
6. Makefile