EEL4709C
Final Project Report

Project Title:
Robotic Control and Computer Vision

Authors

Name: Jose Hernandez PID: [redacted]

Name: Andy Carvajal PID: [redacted]

Name: Sebastian Garcia PID: [redacted]

Name: Alejandro Perez PID: [redacted]

Project Objectives

As a primary objective the team decided to make a robotic car from scratch using a 3D printer to create mechanical parts. After the mechanical and electrical parts of the robot were assembled the main project objectives were:

-Use raspberry pi to control robotic car
-Search area for an object, be able to recognize and move towards it
-Use camera to id certain objects (OpenCV)
-Be able to control car over the network (using flask)
-Demonstrate the power efficiency of raspberry pi powering whole system with one Lithium-ion battery

Components Used

- Raspberry pi
- Raspberry pi camera version 2
- 3D printing material to create structure of car
- Plastic wheels
- Screws and nuts
- DC geared motors
- Fan
- Lithium-ion battery 21700 3.7V 5000 mA
- Motor driver DRV8834
- Capacitors
- Current sense resistors
- Wires
- Voltage depot
- On/off mechanical switch
- Voltage step-up

Picture or Schematic of the System

In the picture below is the complete wiring schematic of the robotic system. Four DC gear motors were use for actuation and a fan was used for cooling purposes. A motor driver was specifically designed for this robotic car using Texas Instruments' DRV8834 driver. This driver is powered with 6.5V from the corresponding voltage step up and has six signal connections to the Raspberry Pi. Two of these connections (yellow and blue) are the H-bridge connections which are controlled using PWM signals to control the speed of the motors. The orange and green signals control the direction of the spin of the motors using a digital high or low voltage. The red signal is an input signal into the Raspberry Pi which indicates overvoltage, overcurrent or overheat related issues in the motor driver by means of a low voltage. The purple signal enables or disables the whole operation of the motor driver with a low voltage. For optimal control of the driver, the controller and driver should share the same ground. To power the whole system a 21700 battery was used. To power the computer a 5V step up with 2.4A current regulator was used.

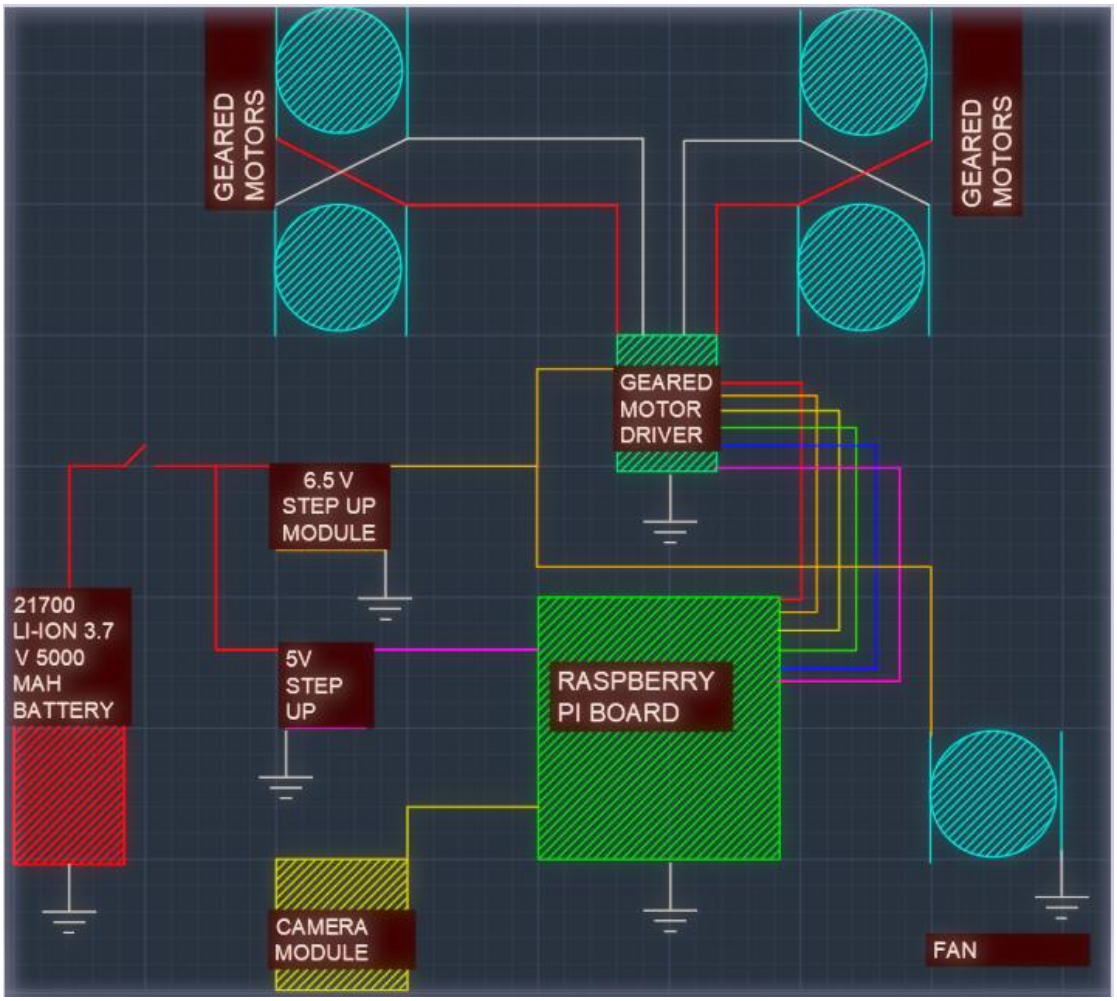


Figure 1. Electrical wiring diagram

The picture below shows the driver that was made specifically for this robotic system. Such board was designed using the Eagle software. It was decided to make this custom driver due to size constraints. Most motor drivels sold out there were a bit too big to fit into this small system. The capacitors in this driver were used for filtering purposes. R1 and R2 are current sense resistors to measure excess current being drawn by the motors. 0.5 Ohms resistors were used so that the tripping current to activate overcurrent warning is 1A for each set of motors (which never happens, therefore higher value resistors should have been used for this overcurrent feature to work).

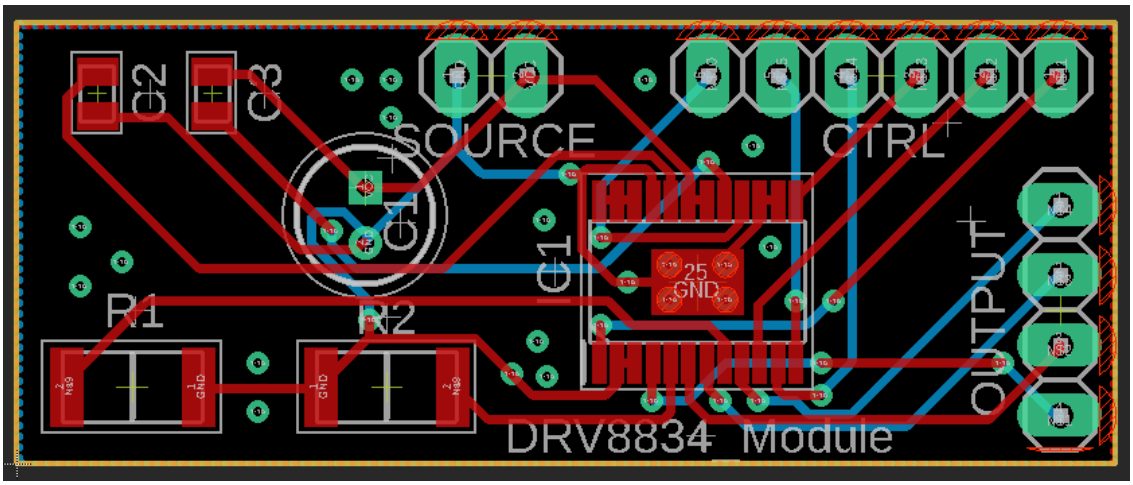


Figure 2. DRV8834 Motor Driver Module

Below is shown the mechanical design of the robotic vehicle. This design was made using Autodesk software called Fusion 360. This mechanical design is basically a kit which can be assembled using M3 and M2 screws and nuts. It includes different cases to protect the Raspberry Pi, the camera, and the battery. Furthermore, a suspension system was implemented using only printed parts. We included an opening in the top of the Raspberry Pi case to allow for cooling by using a fan. All these parts were designed from scratch to fit the custom needs of the project. All mechanical parts were manufactured with a thermoplastic named Polyethylene terephthalate glycol (PETG) using a 3D printer.

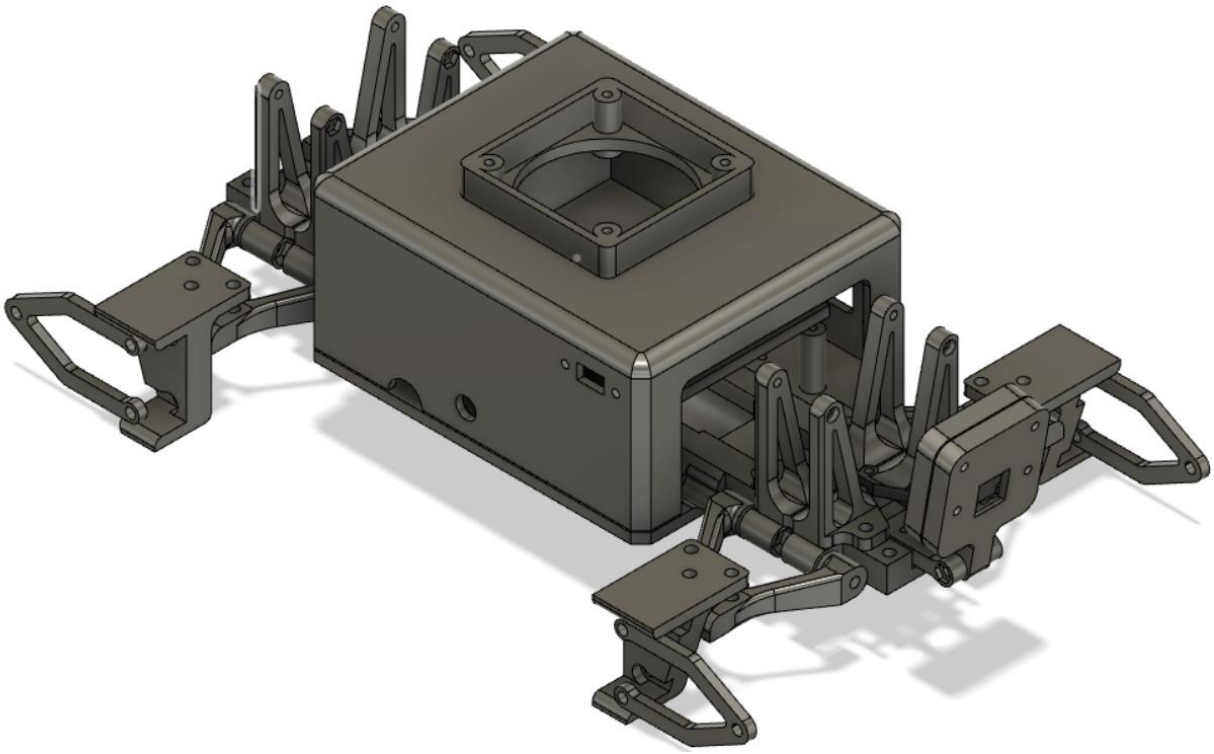


Figure 3. Mechanical Design

Computer Program

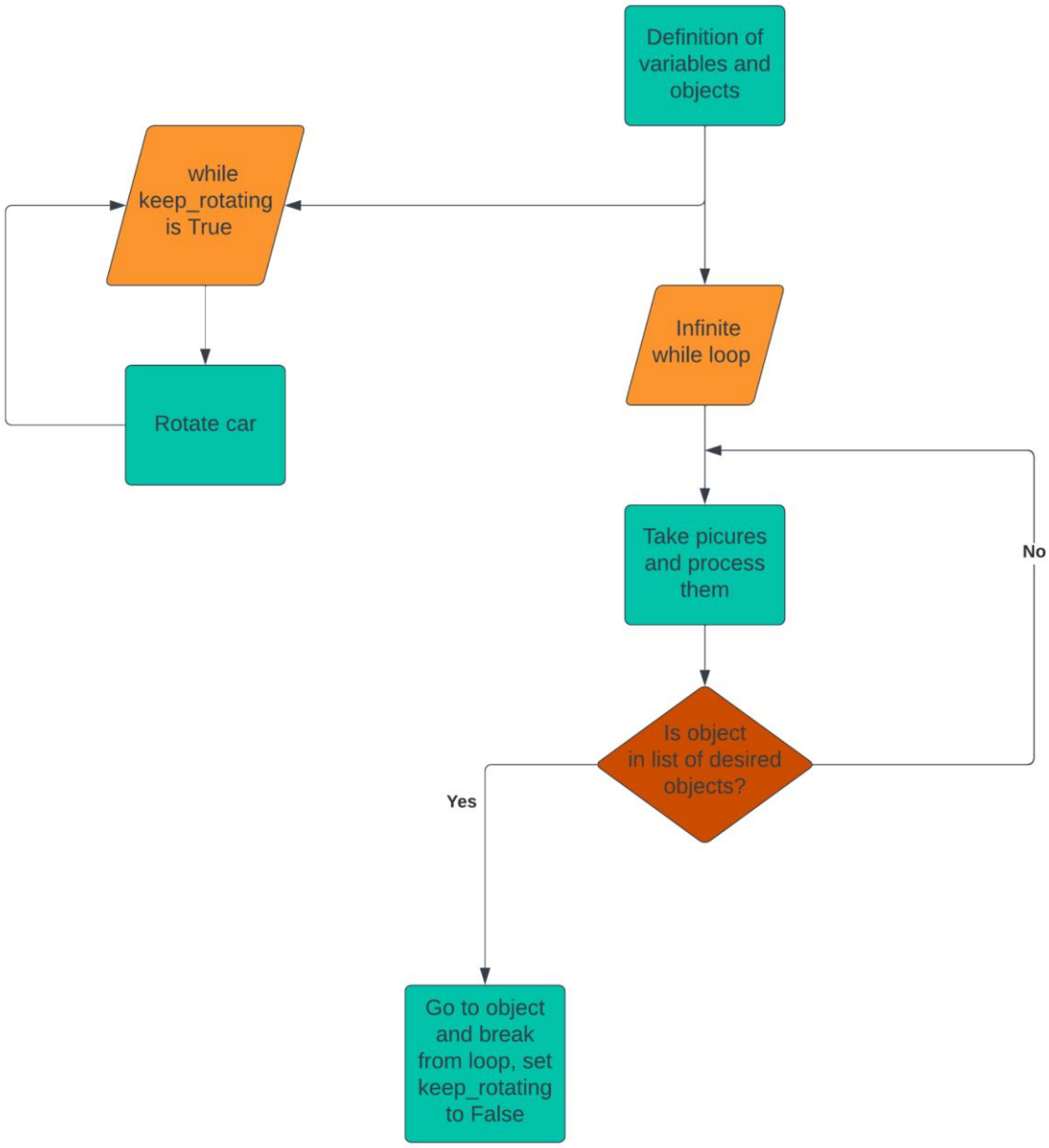


Figure 4. Object Recognition Algorithm Diagram
Code: https://github.com/Scoobylolo/Pi-Car-Code

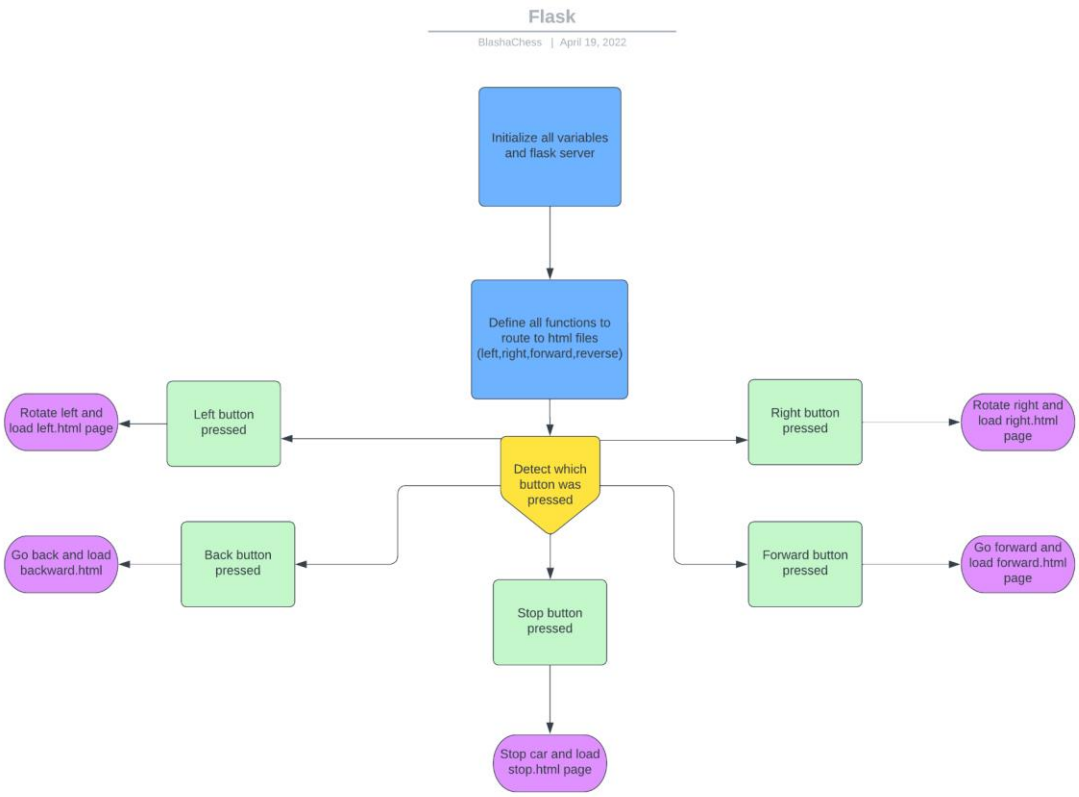


Figure 5. Network Remote Control Algorithm Diagram
Code: https://github.com/aale24/Flask-Car-Controller-Raspberri-Pi

Troubleshooting

- Overheating of the Raspberry Pi due to image processing. After several minutes the computer got hot, and the performance was reduced
- Redesign of the motor driver module was necessary to allow proper air flow to reach the Raspberry Pi by adding a fan on top
- Slow execution of the program due to motor control delays so we needed to add multithreading using python Threading library
- Plastic geared motors were first used; however, these were replaced due to gears slipping when the wheels were in high friction (soft) surfaces
- Originally, it was planned to power the motors directly from the 3.7V battery. Nevertheless, this proved ineffective due to a voltage drop of ~0.5 V across the motor driver. Therefore, a voltage step-up module from 3.7 to 6.5 V was used. We decided to power the motors independently from the voltage step up and current regulator module

that was used to power the Raspberry Pi. This was done to ensure steady power supply to the computer.

- Inaccurate object identification is a tough problem we encountered. We have learned that the images used to train an object identification model should match different positions of the object, different light conditions, and even different backgrounds to be more accurate.
- Had to disconnect the fan from the raspberry pi because it was drawing too much power. Instead, we connected the fan directly to the voltage step up
- Implemented multithreading where one program drives the car and the other program detects the object because there was a problem where the movement of the car was ahead of the detection of the object
- Originally wanted to use TensorFlow but could not get it to work properly so we ended up using OpenCV
- Usage of HDMI while the Raspberry Pi runs on the battery caused low voltage warning to pop up. Thus, it's better to access the Pi using VNC and/or SSH instead.

Recommendations and Conclusions

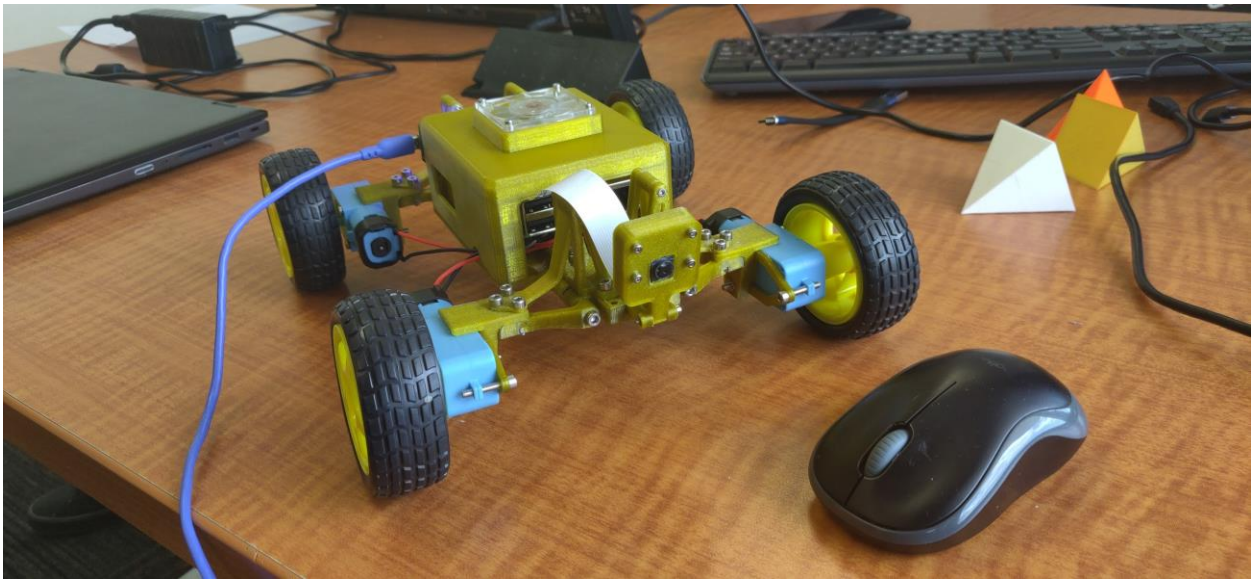
Overall, this project was and continues to be a challenge as we further work on it. Nevertheless, these challenges taught us about mechanical and circuit design, robotic control, object identification, server creation and of course all this while working with the Raspberry Pi and thus becoming more knowledgeable of its operative system. Despite the hurdles faced, we were able to put together a robotic car which is controlled by a Raspberry Pi. This robotic car has two functionalities, it could be remotely controlled through the network, or it could search for a specific object in a small area without obstacles. In its current state, this robot is unable to avoid obstacles due to the challenges related with distance estimation using a single camera. Further work that could be done to improve this project include:

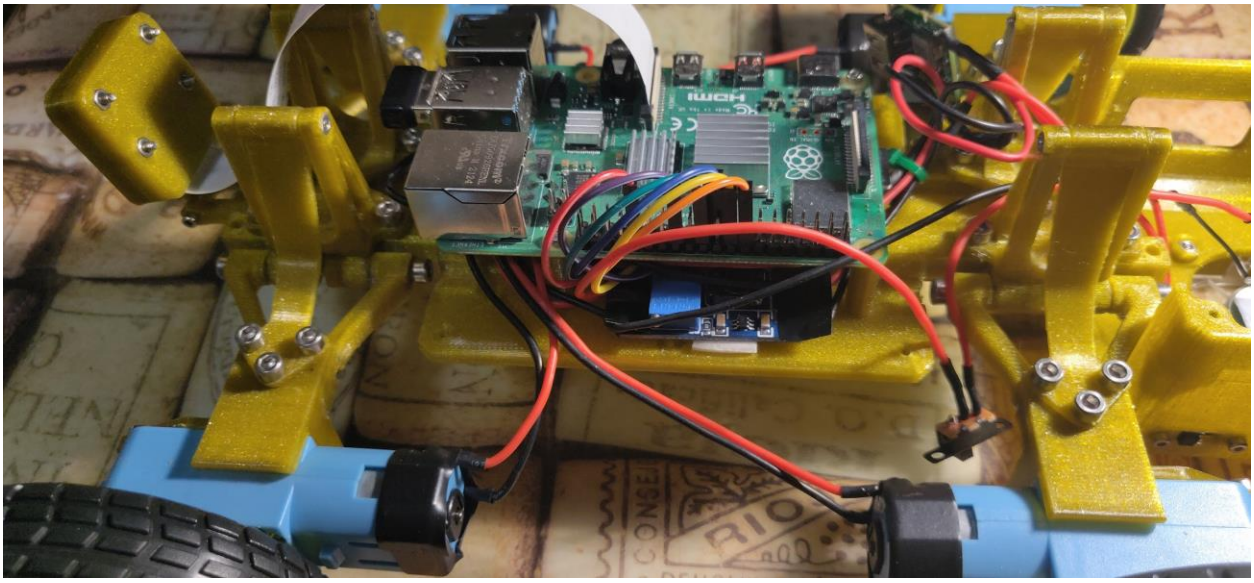
- Improve by making an integrated circuit board that would connect all modules directly to the Raspberry Pi
- Adding an LED in the front of the vehicle to have lighting in low light areas to be able to id an object
- Improve material of the car to make it more sturdy
- Add a servo that can tilt the camera up/down to increase area of vision
- Improve camera resolution to be able to recognize objects beyond 10ft
- Create a new model that allows for identification of different hand gestures
- Better battery (10000 mA)
- Improve tires for better traction and maneuverability
- Add obstacle distance estimation and avoidance algorithm. Other sensors could be used to aid with distance estimation.
- Addition of an IMU will further help with navigation

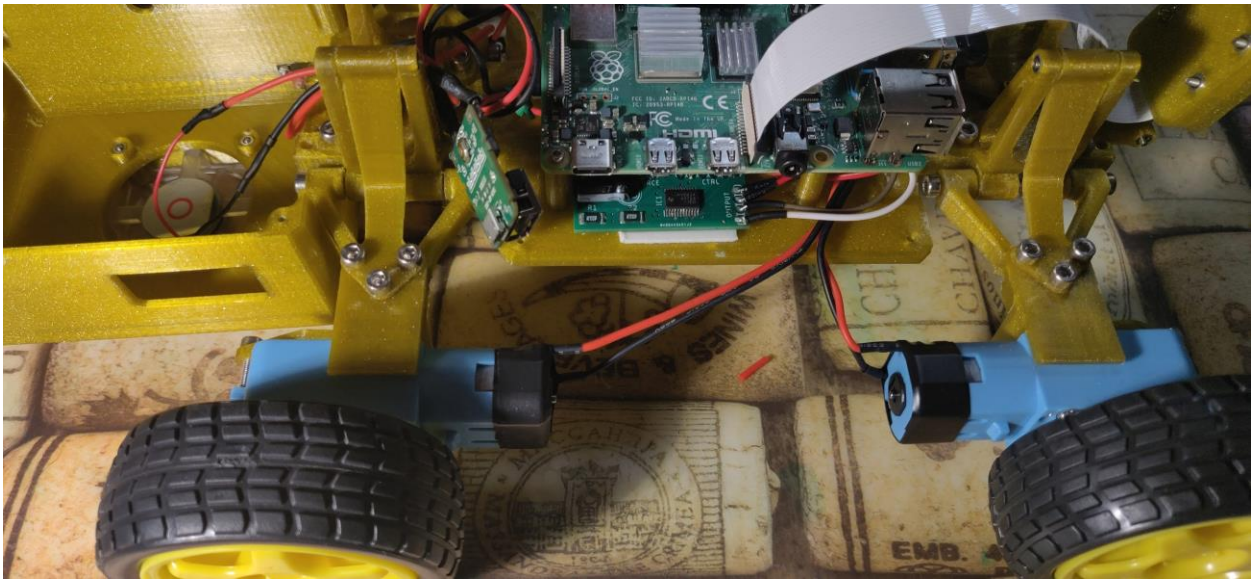
References

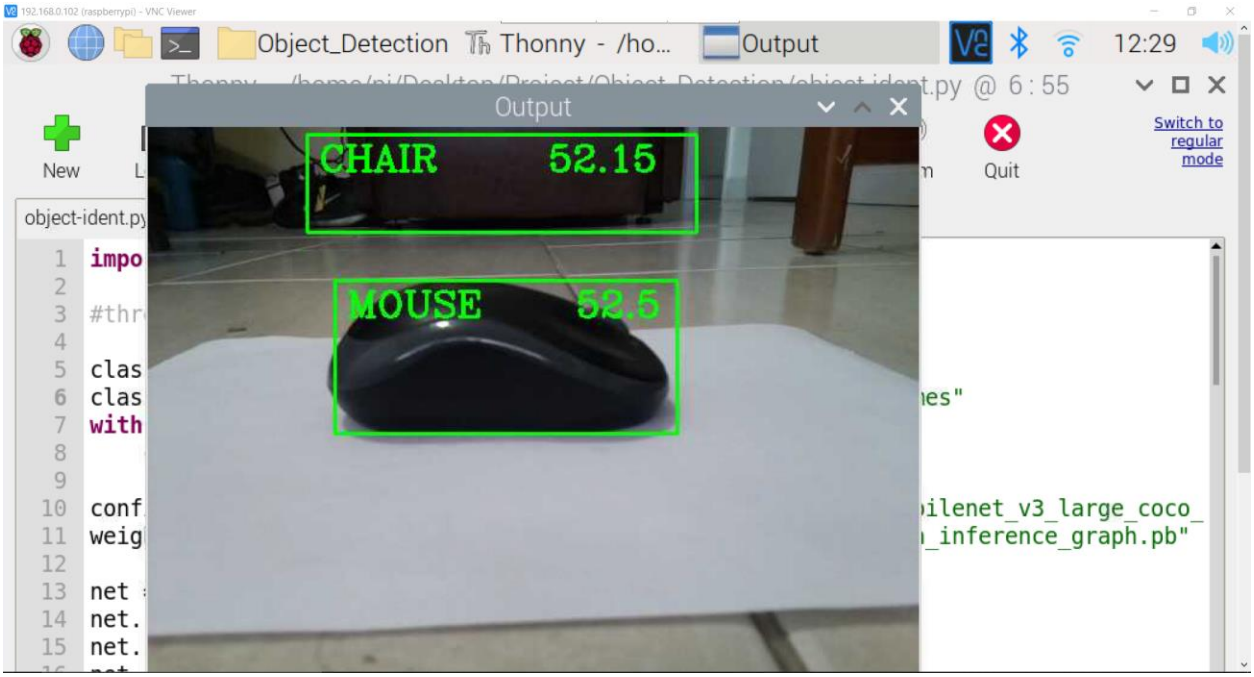
Original code to create flask server was retrieved from Eben Kouao's repository, below his repository:
https://github.com/EbenKouao/pi-camera-stream-flask
Original code to access the camera and identify objects using open CV was retrieved from Core Electronics and was created by Tim, link below:
https://core-electronics.com.au/guides/face-identify-raspberry-pi/
Design specifications for the motor driver circuit were derived from the DRV8834 datasheet by Texas Instruments, link below:
https://www.ti.com/lit/ds/symlink/drv8834.pdf?ts=1650634728664

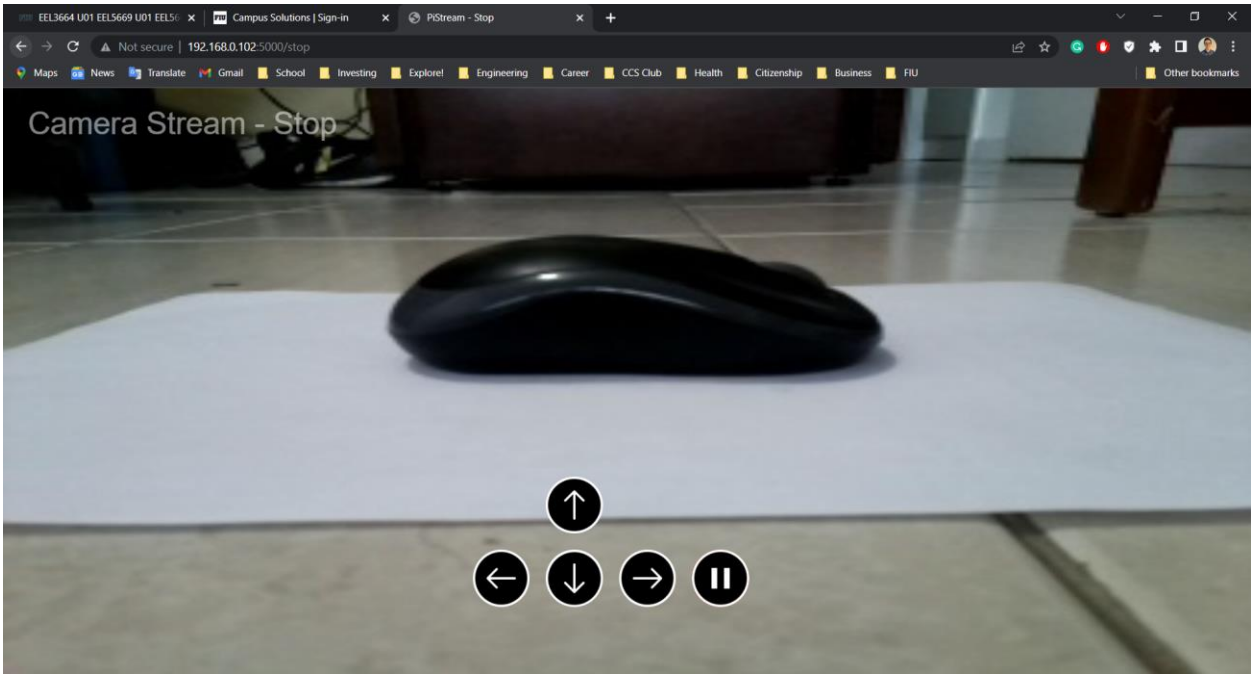
Gallery








Object_Detection
Thonny - /ho...
Output
12:29
Output
CHAIR 52.15
MOUSE 52.5
New
Quit
Switch to regular mode
object-ident.py
impo
#thr
clas
clas
with
conf
weig
net
net.
net.
es"
ilenet_v3_large_coco_
_inference_graph.pb"


EEL3664 U01 EEL5669 U01 EEL5
Campus Solutions | Sign-in
PiStream - Stop
Not secure | 192.168.0.102:5000/stop
Maps
News
Translate
Gmail
School
Investing
Explore!
Engineering
Career
CCS Club
Health
Citizenship
Business
FIU
Other bookmarks
Camera Stream - Stop