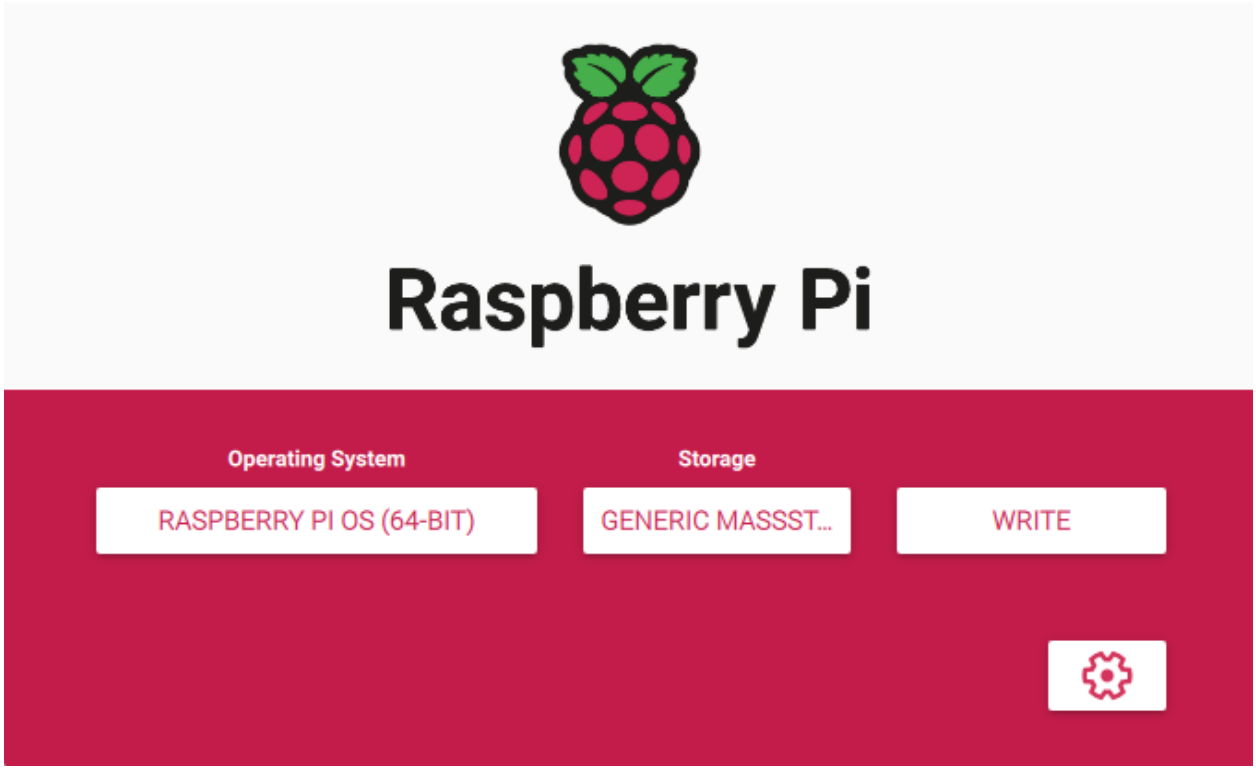
# Ball Tracking Robot




## Raspberry Pi Setup Summary

First I tried to set up the Raspberry Pi on the 64 bit OS. I tried to set up the camera with the raspberry Pi but it didn't work because there was new camera software installed. When I disabled the libcamera, the VNC would not work. Hence I re-setup the Raspberry Pi with the 32-bit os and this worked. Then I tried to reinstall opencv but it would not work. Hence I installed a new camera module named Picamera to get the output from the camera since the open-cv video capture does not work. After doing all of this I'm able to get the camera output working.

## Specifics for setting up the Pi

1. Use raspberry Pi imager to write os and settings onto a sd card. In the settings, turn on SSH, set the hostname, set a username and password and allow it to connect to the correct Wi-Fi.
2. Then insert the sd card into the Raspberry Pi and boot up the Pi.
3. Once the Pi is booted up try to ssh into the Pi using terminal:
   Command: ssh pi@raspberrypi.local
4. Run sudo raspi-config and go into interface options. Then enable VNC and change the resolution to a relatively large number.
5. Open the vnc viewer and connect to raspberry Pi through the search bar.

6. Then connect VS Code to ssh through the ssh extension.
7. Then connect the terminals through the button in the bottom left corner.
8. Cd into the correct directory.

---

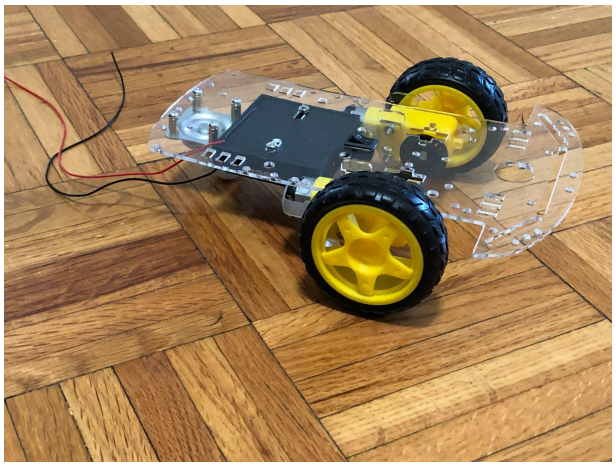
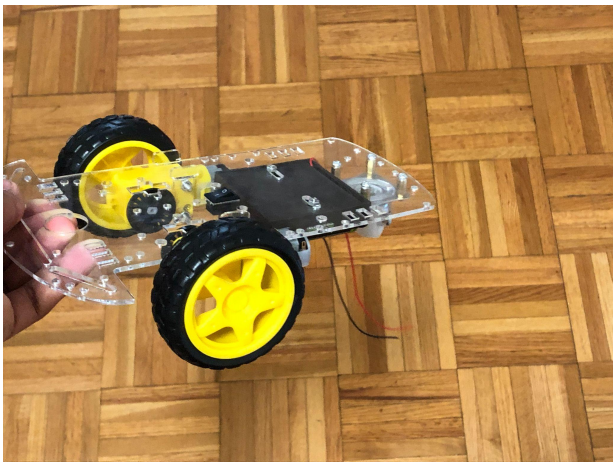
## Camera

I then connected the ArduCam to the Raspberry Pi through a slot on the Pi. The blue tape on the camera cable must face the black clip on the connector.

---

## Robot Assembly (Part 1)

I assembled the kit that I received which included the hardware that was needed to build the robot. The only thing that I had to be careful about was to ensure that the copper tabs on the motors faced outwards.

---

## Software Creation Process

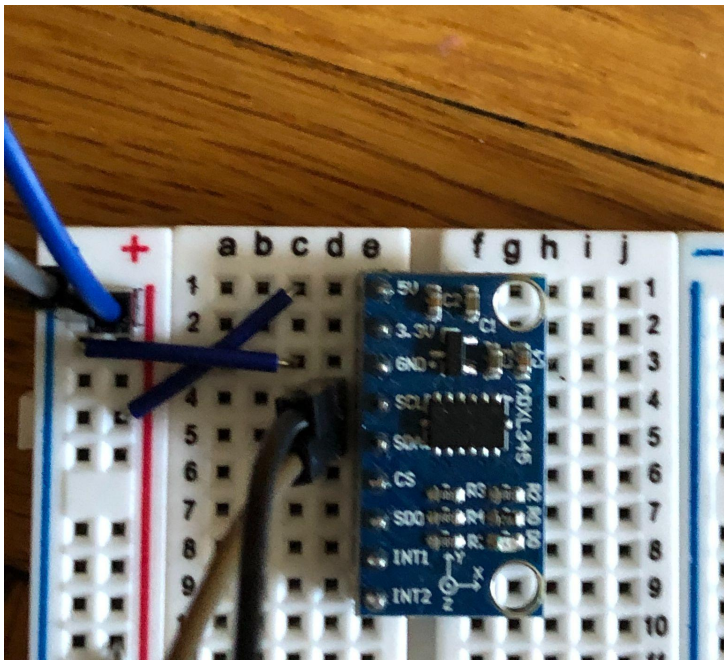
First I tried to get the contours in an image and then tried to filter out the ball. However it didn't work because I was unable to filter out the ball. So I tried a different way of generating the binary image that was being used to generate the contours, but I was still unable to filter out the ball. For the third and final iteration of the software, I decided to use image masking to create the binary image. To do this, I first changed the camera configuration from stream_configuration to still_configuration. This allowed us to take images in RGB instead of YUV420 as it was in the stream_configuration. RGB images are just easier to deal with and manipulate. Once I got the image I converted it to BGR which I used to convert the image to HSV. Once the image was in HSV I set up lower and upper HSV color bounds which allowed us to filter a specific color out of the image. This is called image masking. This generated a binary image that replaced all the

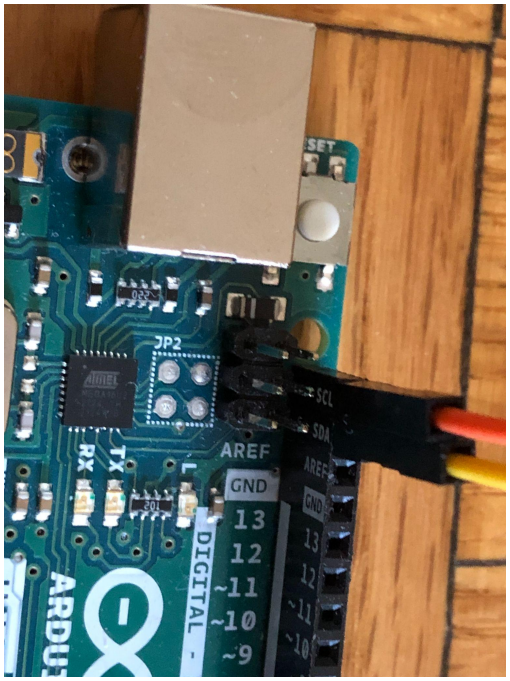
colors I wanted to detect with white and the rest with black. Once I had that image I counted the number of white pixels to detect if the ball was actually in the image and then I took the average location of all the white pixels to compute the center of the ball. We then wrote software that the robot could use to move towards the ball. The software included different functions that could be used to move in different directions and the software would come up with a sequence of movements that would move it closer to the ball. Finally, we wrote software that would allow the accelerometer to detect tilt and for the Arduino to process it and tell the bluetooth module to send a unique command to the Raspberry Pi. Upon receiving the command the Raspberry Pi would make the robot move in the appropriate direction. Plus, we made it so that if the accelerometer were to be flat, then the robot would automatically go into ball detection mode. Link to the github repository for the code.

---

## Setting up accelerometer and Arduino

I decided to set up my accelerometer with the Arduino. First I installed the Arduino IDE so that I could write and upload code onto the Arduino. Then I had to install the Adafruit ADXL345 library. I then installed the example code for bluetooth modules from the Arduino IDE. Once I had that properly set that up I calibrated the accelerometer offsets so that the accelerometer would accurately collect data. Finally, I calibrated the motions that the accelerometer would detect and then what it would do for each respective motion.

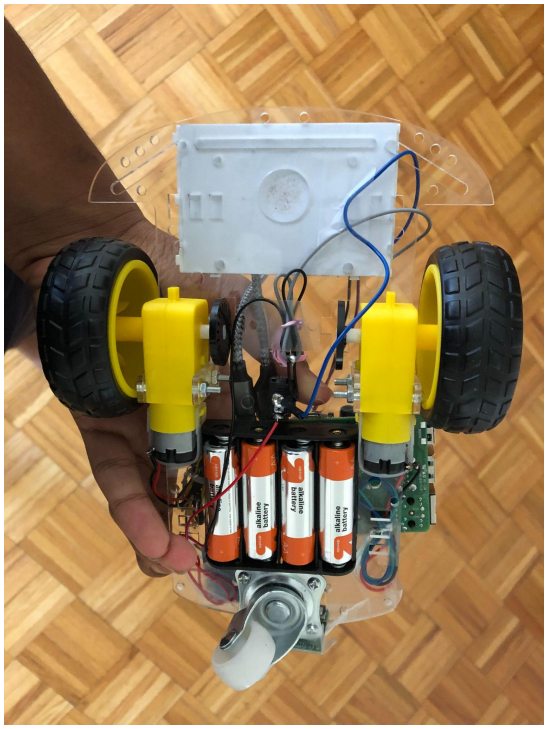


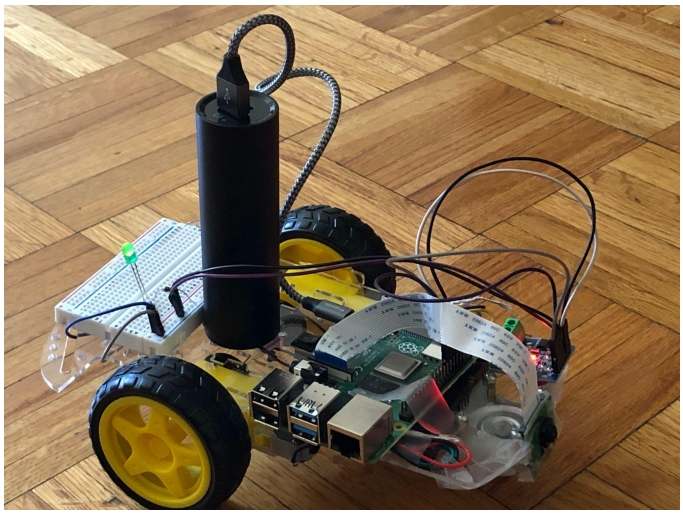




---

## Robot Assembly (Part 2)

First I took a wire from the battery pack and then sautered it onto one of the terminals on the switch. Then I took the second wire from the battery pack and sautered it to a jumper wire. This wire was used to transfer power from the battery pack to the power on the breadboard which is located on top of the robot. I then connected the GND and VCC on the H-bridge onto the breadboard. Then I took some jumper cables and sautered them together to make them longer. I used these elongated jumper wires to connect the H-bridge to the motors. Once this was done I connected the other motor pins on the H-bridge to the GPIO pins on the  Raspberry Pi. This setup allowed us to control the motors through the Pi. Finally, I taped some of the wires and the raspberry pi onto the robot. Also, I now started to power the Pi using a battery bank instead of a wall plug.





---

## Setting up the bluetooth module

To set up the bluetooth module, I started with getting the bluetooth module on the breadboard with the accelerometer and then properly wiring it to the Arduino. Once I had that set up, I had to configure the bluetooth module in software. Once I had it configured I then enabled bluetooth on the Raspberry Pi and then connected the bluetooth module to the Raspberry Pi. I then used a library called pySerial to collect the information that the Arduino was sending. Once we had that data we used it to make the robot move according to the gestures made by the human. Then, every time I need to connect the 2 devices I have to run these set of commands:

1. Bluetoothctl
2. Open new terminal
3. rfcomm bind rfcomm0 00:14:03:05:0A:F0