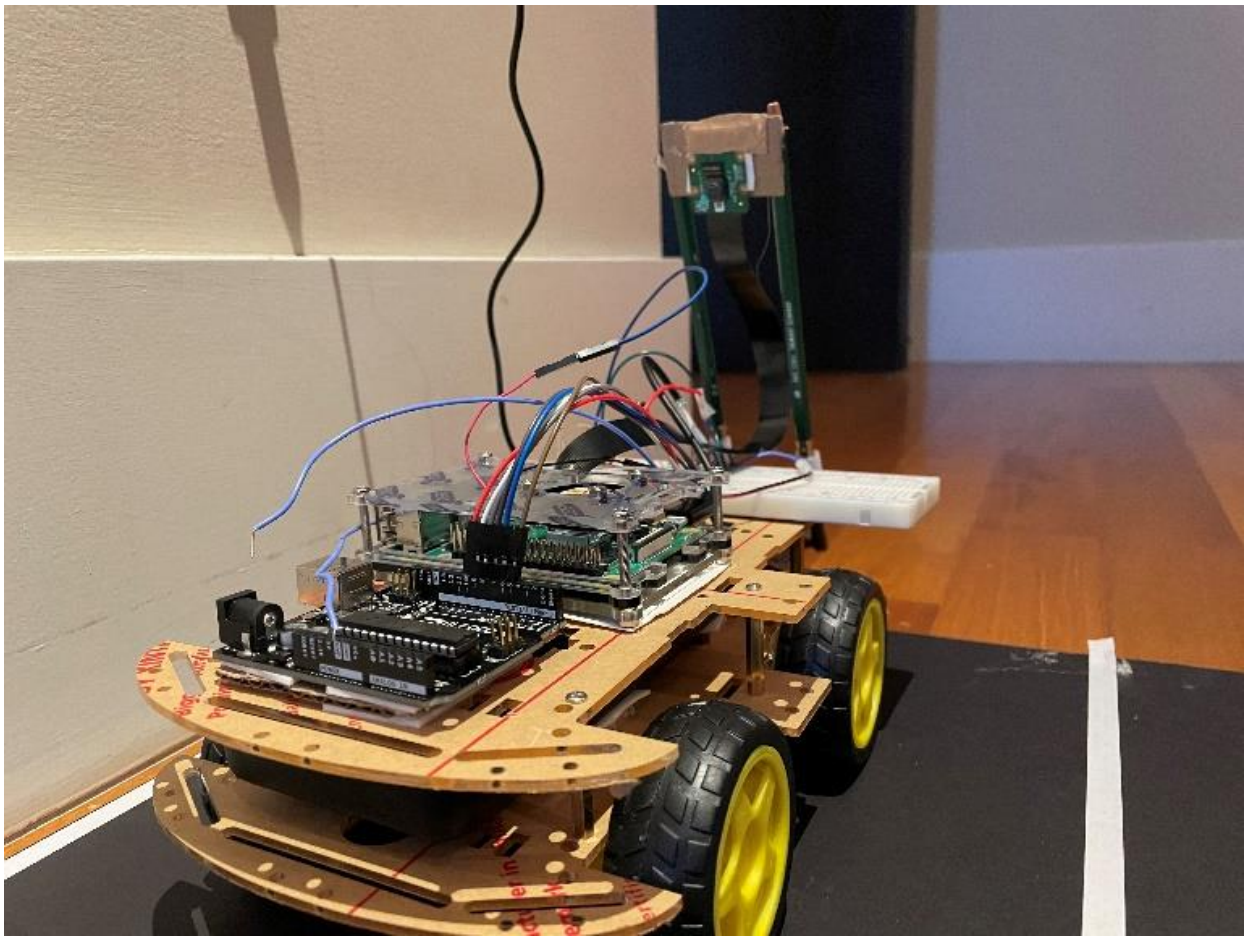
Self-Guided Tutorial for Self-Driving Cars

Using Raspberry Pi and Arduino Uno
Step by Step Guide



By Yunchu Feng

Safety Instructions

Overview

This guide servers as a reference to those who wants to apply their theorical knowledge to the field.

The materials in this guide must be studied and used with caution due to their dangerous nature. For example, soldering gears can get to extreme temperatures and cost burn damage.

Raspberry Pi is the primary microprocessor used in this guide; Arduino will be used to support it.

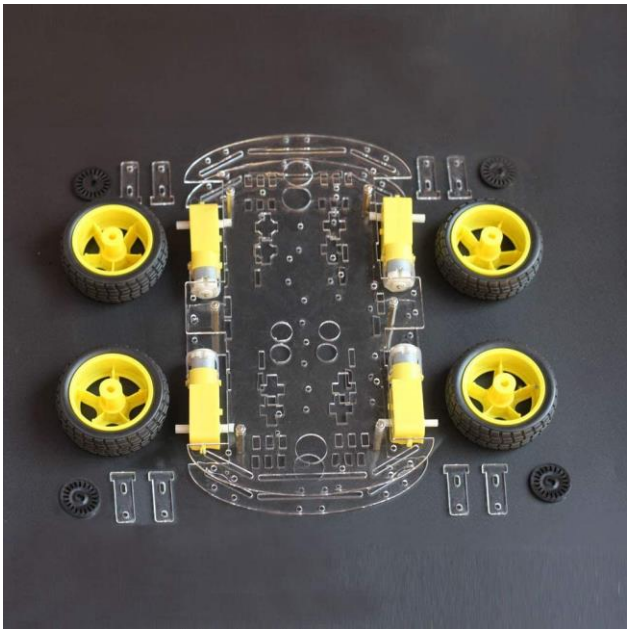
This guide will cover most, not all, of the problems that could happen during your install and coding. If an error occurs, please read the guide carefully and see if you have missed anything. Doing research along with the guide is advised.

The electrical installation and maintenance work on the car might cause friction and even fire, processed with caution.

Hope you the best in acquiring your own self-driving car

Materials

(Links will be provided)



- The first item needed for this project is a ***car frame powered by 4 motors*** with rated voltage of 3~6V and current: 150mA +/- 10%. If you want more zesty looking cars, 3D printing is also an option.
- Second item is a ***motor drive controller*** that will control the voltage of the motor.
- ***Power bank*** is also a very important item. Recommended stable voltage of 5V.
- Most important item the Raspberry Pi. I recommend using ***Raspberry Pi 4***, since is the one I used in this guide.
- A compatible ***camera*** for your Pi.
- We are using an initiator and target set ups, so we also need an ***Arduino Uno*** to help us. Initiator being our Pi, and target being the Arduino.
- Lastly ***A 8GB+ SD card*** to store the OS for your Pi.
- ***Soldering kit***.
- There are also other miscellaneous items need depending on your situation.

Car Frame and Motors:
https://www.amazon.com/gp/product/B06VTP8XBQ/ref=ox_sc_act_title_1?smid=AZ7YXD2PTX165&psc=1

Motor Dive Controller:
https://www.amazon.com/gp/product/B014KMHSW6/ref=ox_sc_act_title_1?smid=A12MRQC2NA7LMA&psc=1

Power Bank:

https://www.amazon.com/gp/product/B07K13SSVS/ref=ox_sc_act_title_1?smid=A1KL49E6INLWJH&psc=1

Raspberry Pi:

https://www.amazon.com/gp/product/B01CMC50S0/ref=ox_sc_act_title_6?smid=AAU5UPIIBDRLP&psc=1

Camera:

https://www.amazon.com/gp/product/B01ER2SKFS/ref=ox_sc_act_title_2?smid=AYY3UP8OL9FV8&psc=1

Arduino:
https://www.amazon.com/gp/product/B01EWOE0UU/ref=ox_sc_act_title_4?smid=A2WWHQ25ENKVJ1&psc=1

SD Card:

https://www.amazon.com/Sandisk-Ultra-Micro-UHS-I-Adapter/dp/B073K14CVB/ref=sr_1_2?keywords=16GB+Class+10&qid=1578502735&s=electronics&sr=1-2

Soldering Kit:

https://www.amazon.com/s?k=soldering+kit&ref=nb_sb_noss_2

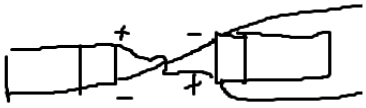
Miscellaneous (may or may not needed depending on your build):

Soldering Wires, Longer Raspberry Pi Camera cable, USB to USB cables, batteries, and batteries holders.

Installation

1. The First thing after getting your Kit is you should wire the top/bottom left and top/bottom right motors together to make them controlled by one wire. This step will require soldering, for more information on soldering please go to the soldering guide page. The reason we do this is because our Motor Dive Controller (H-Bridge), figure 2, only takes in two motor inputs. To counter this, we must combine the four motor inputs into two motor inputs. Meaning combine the two on the left and the two on the right. Circuit refers to figure 1.

Figure 1



The reason we want to use a (H-Bridge) is because how the motor speed can be controlled by voltage inputs. High voltage means faster speed; low voltage means slower wheel rotation speed. H-Bridge will server as a controller that can lower the voltage going into a motor, therefore, slowing it down. To make our car turn left, all we need to do is slow the wheel and the left side of the car and speed up the wheels on the right. How much you speed up or slow down the wheels will cause different sharpness of turn.

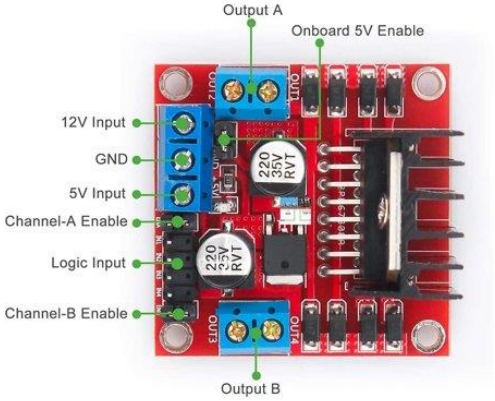


Figure 2

A connected figure of motors and H-bridge, figure 3.

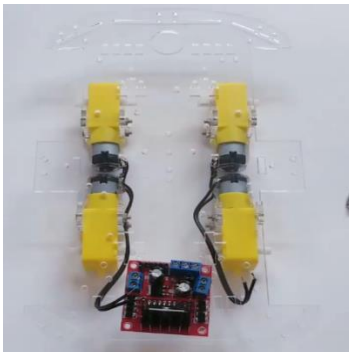


Figure 3

H-Bridge Guide for Detailed Explanation:

https://www.youtube.com/watch?v=dyZolgNOomk&ab_channel=CircuitMagic

2. The power source we are using here is a 5V power bank with at least two voltage output, since we have more than 2 devices that need voltage, we want to make a circuit board. Here soldering of wirings might be necessary to complete the board.

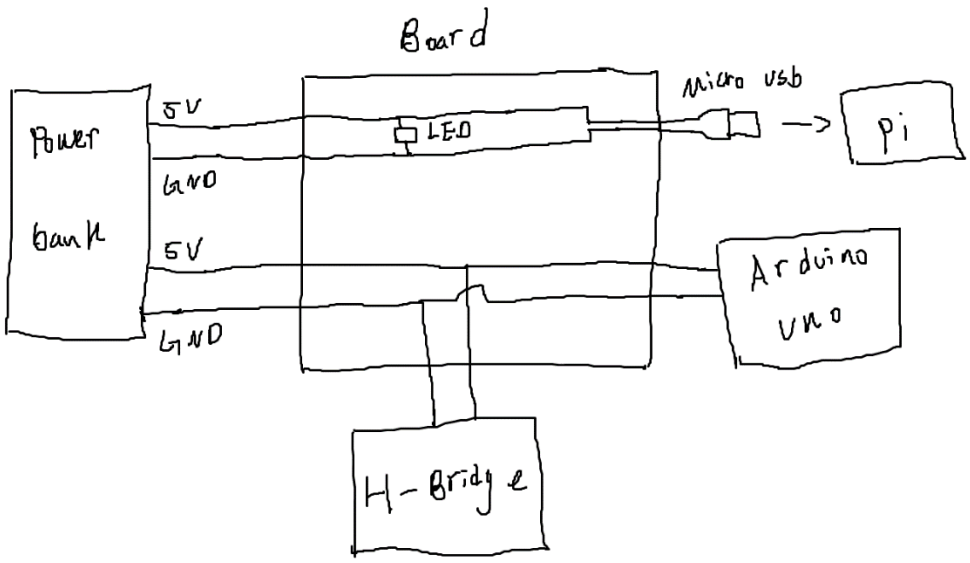


Figure 4

Complete circuit please refer to figure 4. We want to mount the power bank somewhere on the car where activation is easy, also leave rooms for other components. Make sure when connecting the wires to connect the Arduino and the H-Bridge together, as different voltage will mess up the speed. **WARNING**, if you encounter sudden voltage drop on your power bank either later or now, you might want to attach four 1.5V batteries to the board, refer to figure 5.

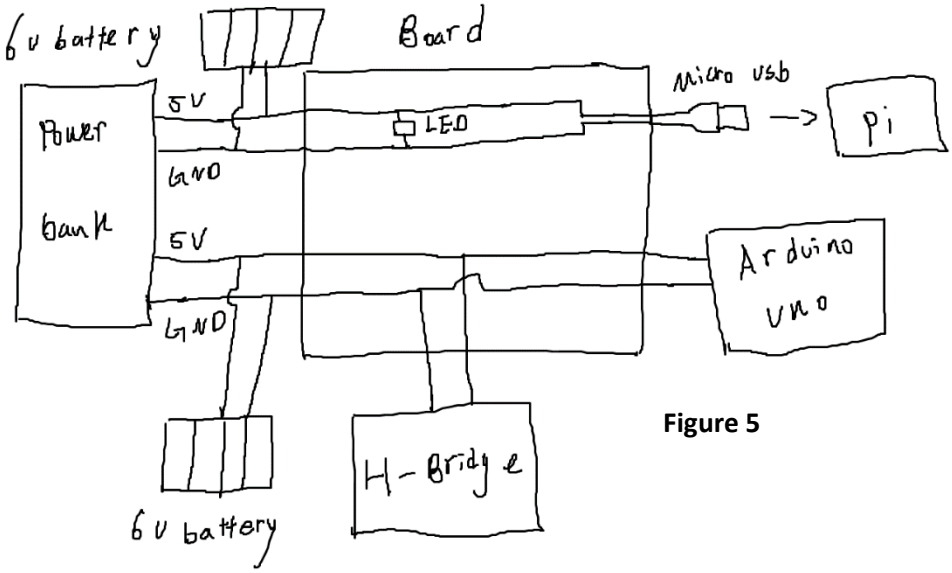


Figure 5

The reason why we need to attach addition batteries is because even though the power bank will output 5V when tested, there will be sudden voltage drop across the entire board when you start your Pi, run heavy codes, or controlling multiple motors at once.

3. Next, we will attach Arduino to our board according to figure 4. WARNING, make sure you are not mixing the wires inputs in the Arduino, short circuits might happen. Please study and read the Arduino pinout, figure 6. For an even more detailed explanation visit the website link: https://www.arduino.cc/en/Guide/ArduinoUno.

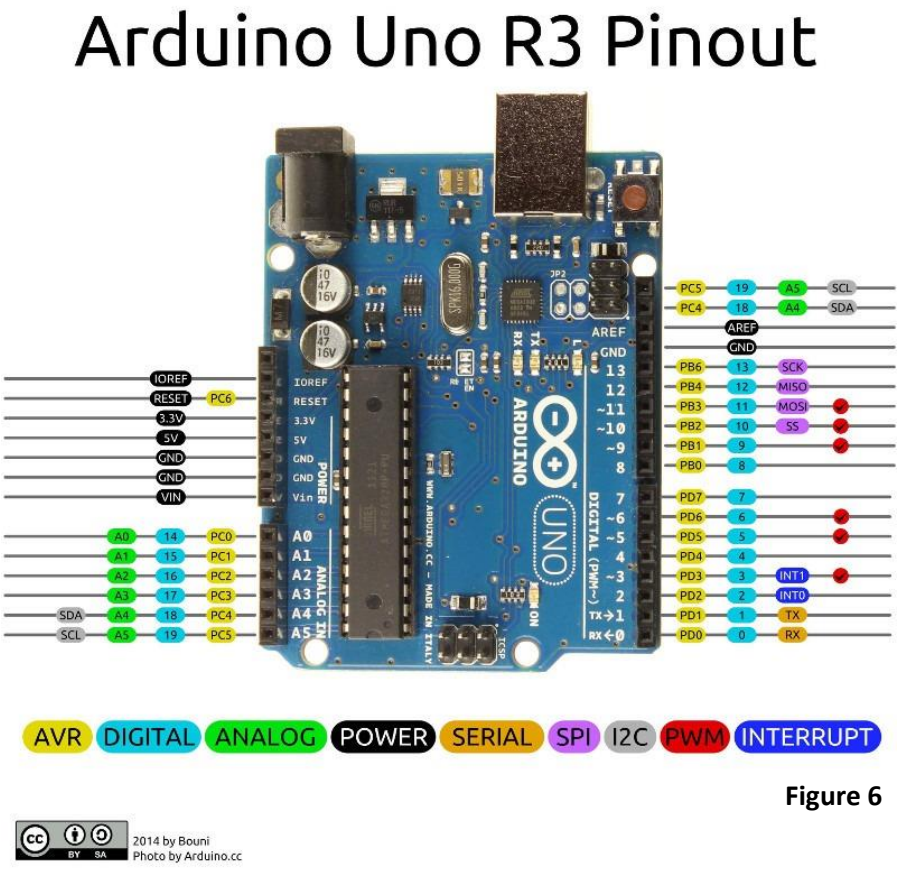


Figure 6

For a detailed explanation visit: Finished build should look something like this:

4. Now go ahead and connect your Pi to the power source too, Pi will take in the Micro-USB as its input voltage, you might need to solder your own wire for this step. After the Pi is powered connect your Pi with your Arduino. Below is the wiring for your Pi, for this project I chose pin 21-24 from the Pi and input 5-10 as the input from the Pi and output to the H-bridge. Refer to figure 6 and 7 for detailed pin layout.

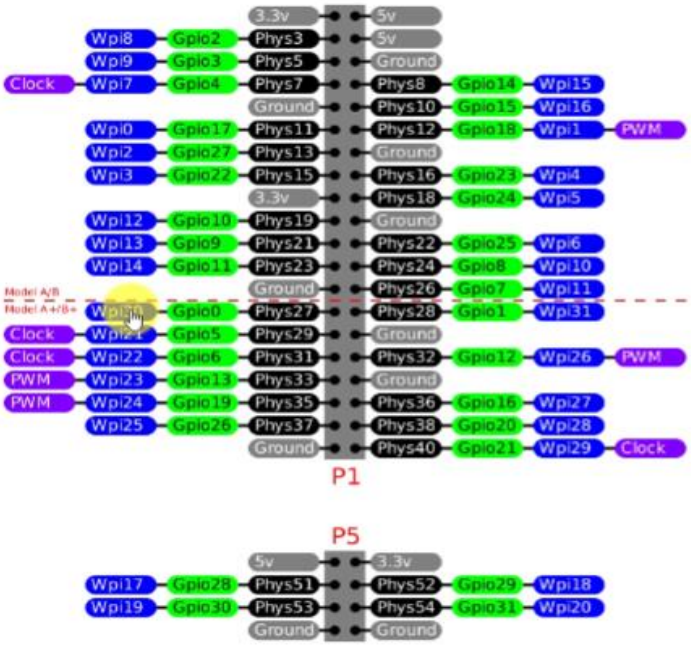


Figure 7

Finished version of the layout below, figure 8.

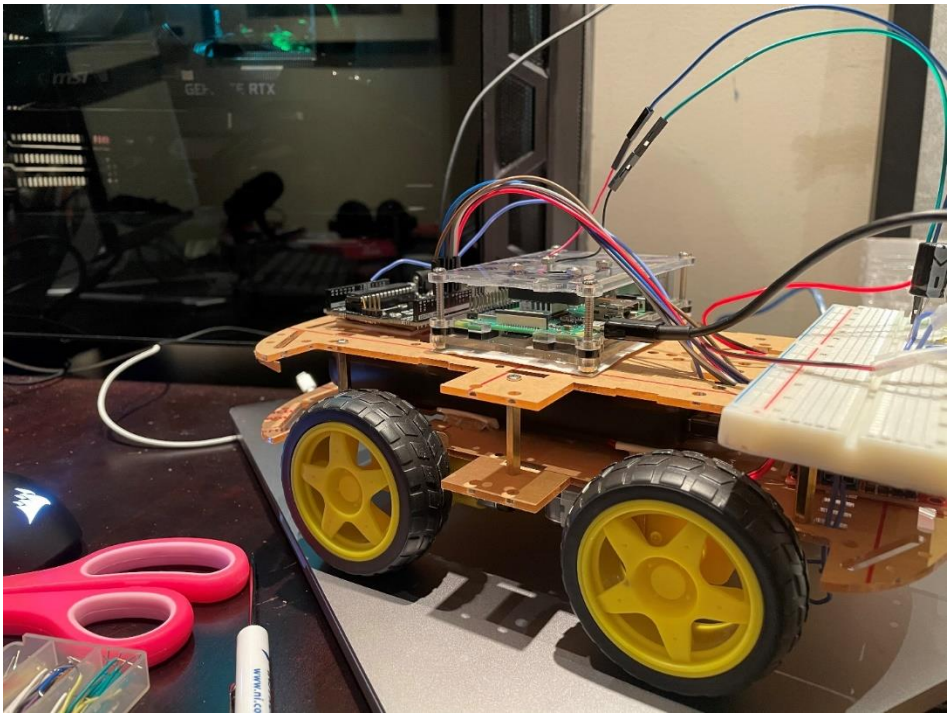


Figure 8

5. The last step is to connect your camera to the Pi, there should be a port on the Pi for camera insert, figure 9. You want to install your Pi camera somewhere it can see its surroundings. This is very helpful as a wider view will help later in the steps. Example camera placement figure 10. The camera can only take picture and not videos, so we must constantly take pictures, depending on the camera you got, 10 picture every second is required to form a video.

Detailed guide on Raspberry Pi camera module

https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

Figure 9

Now everything is finished with the hardware side it is time to dive into the codes. We will first go into the Arduino code, how to control the car to go forward, backward, turn left, and turn right. Next, we will go into setting up the right Pi environment, meaning getting OpenCV, and flashing your Pi. Lastly, we will go into how the software work and how to use the AI.

Good Luck!

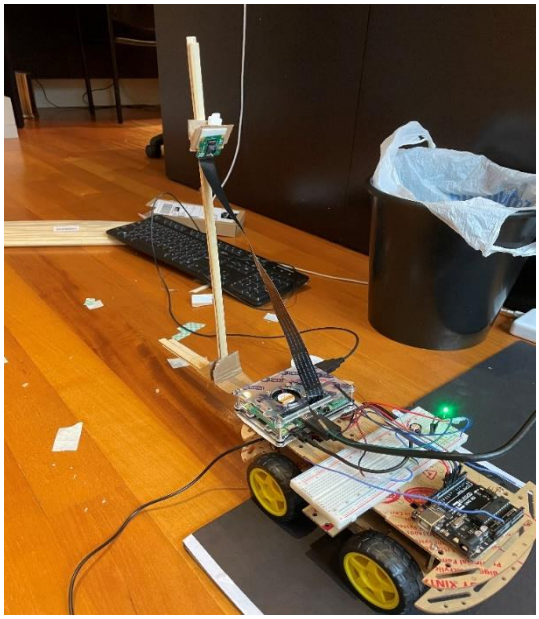


Figure 10

Soldering

This is an optional step that will help you build the right circuits. We will go over how to solder two wires together. **WARNING,** this step is very dangerous! Please solder somewhere save and use the right equipment's.

Figure 11

Figure 11 shows a typical soldering machine you can find online or in a ECE lab. First you want to heat up your solder, and then apply soldering wiring to the tip of the pen. Then, connect the two wirings you wish to solder and touch the wires with your heated pen. You will need to apply more solder wire one the pen is in contact with the wires. Figure 12 shows a perfect solder. Make sure to use lead free solder wires, as some gas might be released in the process, masks are advised.

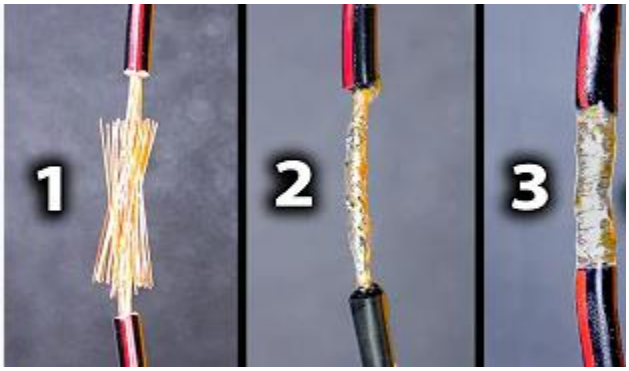


Figure 12

1. First, we need to set up the Arduino by defining all the pins we are going to use. Check figure 13 for detailed code

```
const int EnableL = 5;
const int HighL = 6;        // LEFT SIDE MOTOR
const int LowL =7;

const int EnableR = 10;
const int HighR = 8;        //RIGHT SIDE MOTOR
const int LowR =9;

const int D0 = 0;        //Raspberry pin 21    LSB
const int D1 = 1;        //Raspberry pin 22
const int D2 = 2;        //Raspberry pin 23
const int D3 = 3;        //Raspberry pin 24    MSB

int a,b,c,d,data;
```

Figure 13.

This is just the basic initializing when it comes to Arduino, you want to do this in the global space since they are going to be used in multiple different functions. The best compiler the one Arduino offer themselves. Link for download: https://www.arduino.cc/en/software.

2. Next we need to set up the outputs for the Arduino, then we need to constantly get our data. The data function will constantly read the volage input from there inputs, pin 21-24, from the Pi. This will control our Arduino.

```
void setup() {

pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);


pinMode(EnableR, OUTPUT);
pinMode(HighR, OUTPUT);
pinMode(LowR, OUTPUT);

pinMode(D0, INPUT_PULLUP);
pinMode(D1, INPUT_PULLUP);
pinMode(D2, INPUT_PULLUP);
pinMode(D3, INPUT_PULLUP);
```

Figure 14.

```
void Data()
{
   a = digitalRead(D0);
   b = digitalRead(D1);
   c = digitalRead(D2);
   d = digitalRead(D3);

   data = 8*d+4*c+2*b+a;
}
```

Figure 15.

3. Next the functions for going forward and backward are define below. These functions will be controlled by the signals passed by the Pi. If everything is clear it will go forward. The function digitalWrite will set the pin of a specific output to be low/high, and analogWrite will send a signal to the H-bridge which will control the car to move. The 255 is the speed index of the car. Detailed function explanation visit: https://www.arduino.cc/reference/en/. You have to figure out yourself on how to do the right and left turn because the function will vary due to the motors.

```
void Backward()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,255);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,255);

}
```

```
void Forward()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}
```

4. Assuming you have the functions for all the operations for the Arduino, we now will write the loop to control the Arduino which will control the H-bridge.

```
void loop()
{
  Data();
  if(data==0)
   {
     Forward();
   }

  else if(data==1)
   {
     Right1();
   }

  else if(data==2)
   {
     Left1();
   }

  else if (data>3)
   {
     Stop();
   }



}
```

The Loop function is a special function, acting as the Main () function in C/C++. The loop function will, as you guessed it, constantly run-in loop.

Every time the loop is ran it will check the input by running the Data function, and if the data we got is 0 it means we can go forward. If we get 1 or 2, we must turn right or left. If the data is bigger than 3 than it means our car was ran out of the tracks and we must stop it.

This concludes the Arduino section of the guide, now we will go into coding the Pi.

If you have trouble doing this part download the material from "Final" folder in my GitHub repository.

Raspberry Pi

In this section we will go into how to setup our Pi, how to code our software, how to setup the environment for our Pi. This is going to be a long and boring process.

If you have trouble with this part, please go to "Final" where I have a flash file for you to download. This file has all the environment set up and ready to use. The password and username should all be in the folder.

1. First thing you want to insert the SD card into your PC and connect your Pi with your computer. This is to flash your Pi, and to do so we must download the image from link: https://www.raspberrypi.org/software/operating-systems/.
   Recommend getting the: "Raspberry Pi OS with desktop and recommended software". After getting the torrent we must flash the image to our Pi using Etcher: https://www.balena.io/etcher/. Once everything is flash you can get you SD card from your computer and insert it into your Pi. Figure 16 for Etcher.

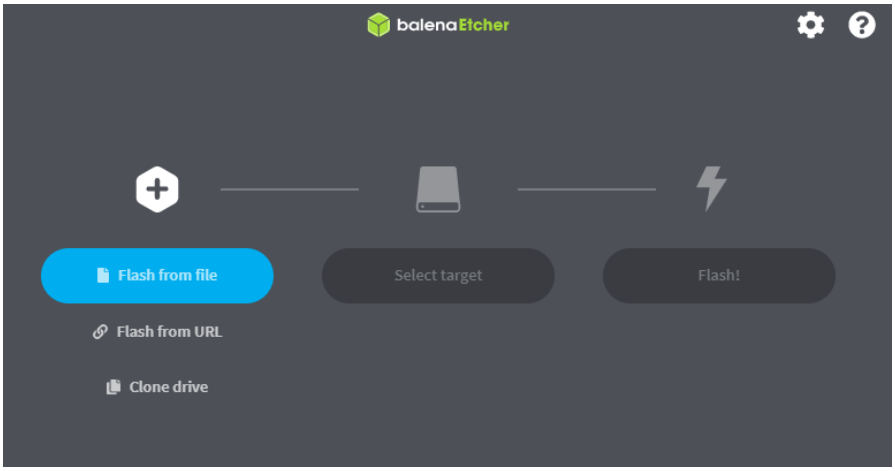


Figure 16.

2. Micro-processor is just like a minicomputer, you must have a monitor for it. There is a HTML output, find a scree and connect it. If you have everything flashed correctly you will be able to boot into your Pi and connect to WIFI or Ethernet, figure 17.

Figure 17.

3. At this point your Pi is connect to WIFI, and you can now download the right environment to program. Now begin the very boring process. Go to my GitHub and open the environment text file to find all the command need to run. You want to run this command in your terminal. The main things we are downloading are OpenCV, which is the library need to do our image processing. Camera library, which is needed to run the camera connect to our Pi. Wiring Pi, which is needed to make our Pi talk to the Arduino Uno.

At this point you should have the right environment and the compiler working for your Pi. Next, we will discuss the code and the AI concept that goes into the programming.

1. The concept we are using to find our lane and to travel is call edge detection. As an ECE student you probably learned this in your ECE 2026 class. By multiplying a matrix by [1, -1] you can see where there is raise and where there is a fall. By tunning these numbers, you can find out where the image separations start. This will make everything that is not an edge into all black or white. We will have a clear image edge for use later. This is done because images are just a matrix of pixels where each pixel represent a number. By passing this matrix through a filter we can get the edge. Figure 18.

Figure 18.

All we need to do now is have a track that our camera can pick up easily and find out the edges of it.

2. What I did here was taking a section in front of the car and made it into a rectangle image, then I ran the Canny Edge Detection what is given to us by OpenCV. We can produce an image with black and white (Left). Figure 19. I grabbed the edges of the tracks and turned them into edge just like figure 18 did. Then I filled these lines with white fillings to make it more visible for us.

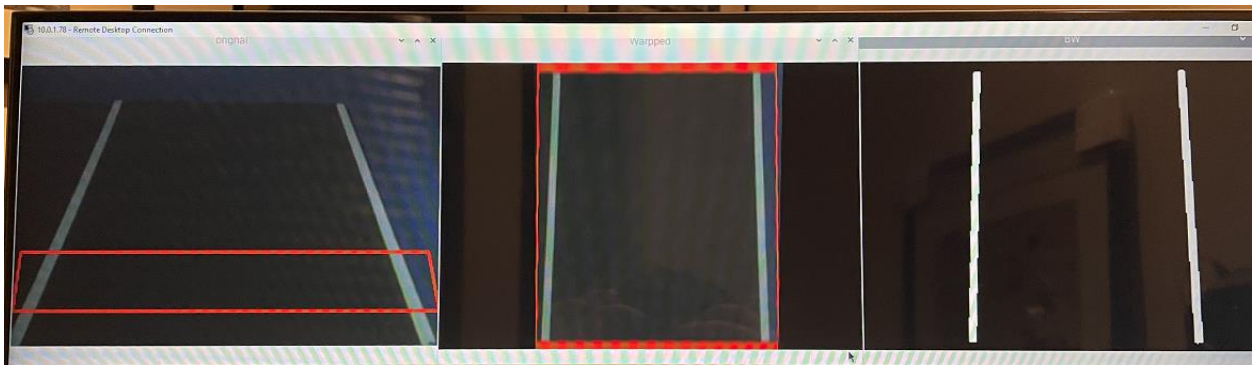


Figure 19.

3. All we must do now is add a Frame Center, which will always be in the middle of the camera, and a Lane Center, which is the average of the two lane we got. If the Frame Center and the Lane Center are aligned that means we can go forward, if they are not, we need to turn left or right. The only problem with this technique is that it can only manage smooth tracks and anything with sharp edge it will fail to respond. Figure 20. We will send our data to an Arduino, where everything is aligned, we will send 0, if we need to go left or right, we will send a 1 or 2. Everything will stop moving we send 3 meaning the lanes disappeared.

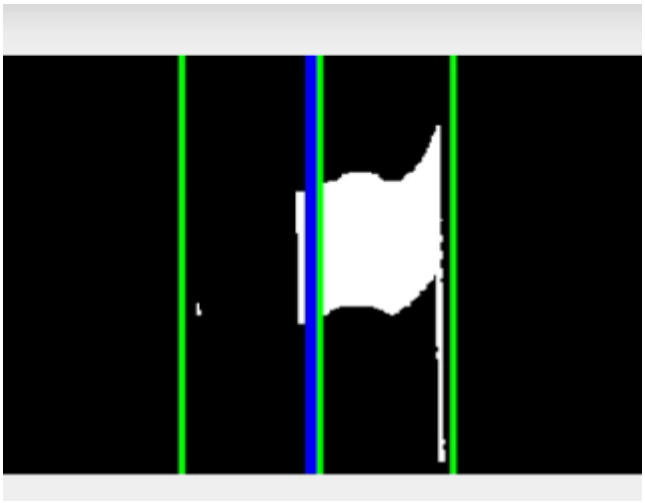


Figure 20.

For a detailed explanation on OpenCV please visit: https://opencv.org/. PLEASE GO LIKE AND SUB TO THEM!