Near East University

Department of Mechatronic Engineering

Capstone Design
MCT411

Name of Project: *Self driving car*

Group Name: *TEAM MAKA*

Group Members:

20177947	*Mustafa Khalifa*
20174217	*Basel Zeid*
20177390	*Michael Alshawareb*
20177376	*Mustapha Hamad*

Supervisor: *Prof. Dr. Bülent Bilgehan*

Semester: *Fall 2020_21*
Submission Date: *01/15/2021*

ABSTRACT

In addition to the different features, one of the most difficult components of creating an autonomous vehicle is to reach low costs and low power usage. The purpose of our project is to illustrate how we built a scaled down model of an autonomous vehicle using an a Pi-car to use Neural Networks and Computer Vision to incorporate an automated self-driving system. The suggested solution relies on the use of an autonomous vehicle in which the car detects lanes and road signs when moving on a predetermined road and tracks the surroundings with the aid of a pi-camera. This camera is required to provide the required data from the physical world. The data is stored and analyzed by Raspberry Pi. Then the raspberry-pi processes input images for target recognition (stop sign and right arrow) to avoiding possible collision and act accordingly. The neural network model runs on the raspberry pi will therefore make predictions for acceleration and directional control based entirely on input images in real time. Predictions are then sent to Arduino to monitor the acceleration of the car through the H-bridge. Almost all calculations are performed on raspberry pi, such as environmental awareness and decision-making.

Table of Contents

CHAPTER 1 – INTRODUCTION

1.1. Detailed definition of the project

Autonomous car and is also known as (driverless car, self-driving car, robotic car) is a robotic vehicle designed to transport between locations without any human involvement. About 1.24 million people are killed every year on the highways around the world. According to the BUET Injury Study Centre, the death toll is 10-12 thousand every year and many people are wounded or damaged, devastating so many lives and communities. The Autonomous Vehicle is the solution to this issue.

1.2. Significance of the project

If we want to talk about the safety matter, whatever we say humans are not that good at driving. Whatever we choose to believe the numbers are giving us the truth that we are the main cause of accidents because Around 1.25 million people are killed every year worldwide but unlike humans, the driverless cars can never drive drunk and will not be able to speed up, take careless risks or race with their friends in the traffic lights. Robots don't have feeling so they will never drive angry or competitive. In a brief, they are much safer than we will ever be.

1.3. Detailed project objectives

Nowadays, in most countries of the world, there are more crowds and vehicles than ever were and they're all in a traffic jam between where you are and where you need to go. Driverless vehicles will travel in convoys, inches apart without unnecessary brake pedals to filter back into traffic and build unexplained, futile hold-ups.

1.4. The goal of this project

In this project, the goal is to build an autonomous pi-car featuring Computer Vision applications using HAAR Cascade to detect stop and arrow signs. Using C++ on OpenCV, we managed to develop a code that regulates the navigation of the car to maintain its line of action within a roadway. The code applies Supervised Learning to classify and detect images to act accordingly depending on the gesture displayed on the pi-camera.

CHAPTER 2 – LITERATURE REVIEW

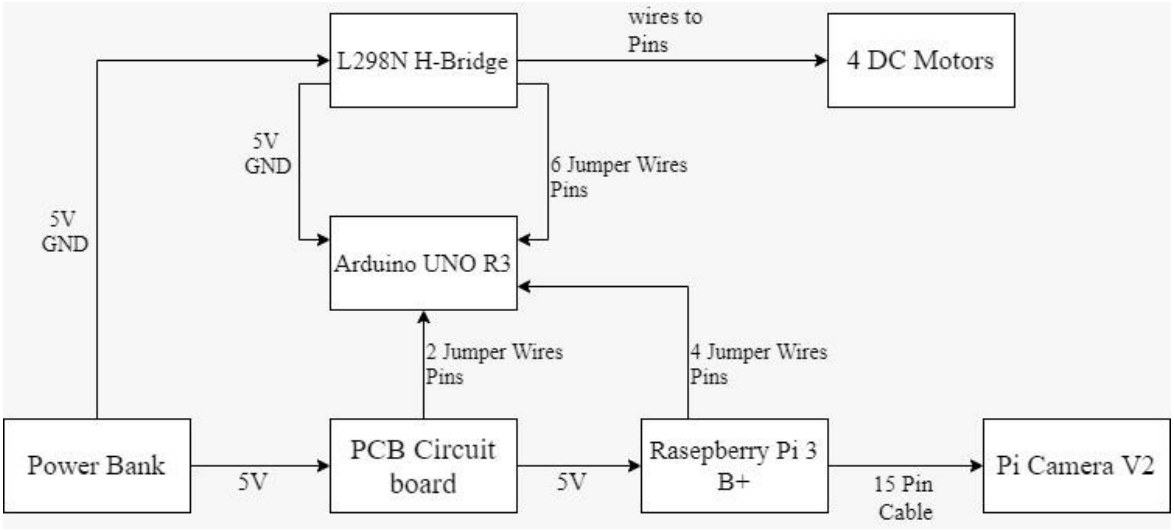
In seeking to clarify the progress of self-driving technology in recent years, it is essential to conduct a literature review to understand the diverse areas of implementation through which self-driving has developed, as well as to factors that must be considered gaps. Of course, several technological advancements that allow self-driving cars are attributed to software as well as algorithmic advancement. Incredible advancements have been made in computer learning to enhance the prospect of the environment, modern monitoring and setting up algorithms that makes driving simpler and easier, and the computing technology to model and process vast volumes of data in data centers have also been important contributors to self-driving vehicles. As a part of the accelerated growth regarding autonomous potentials, Google's autonomous vehicle initiative launched in 2009 to shift its own corporate agency – Waymo – under Google's origin corporation (Alphabet) in 2016. Waymo's autonomous vehicles possess a wide variety of innovations which allow the cars to feel the vehicle's environment, interpret and recognize what's going on area around the area of the vehicle as well as recognize the secure and effective steps for the vehicle to proceed.

We can classify Waymo's self-driving technologies into three main areas: sensing, processing, and embedded power. The sensors collect information about just the car's physical surroundings, location and climate. The sensors are transmitting their data to something like a high-performance computer. A processor fuses, analyses and interpretation of the data of the sensor, eventually creating the paths that the car would obey. The processor transfers these paths to integrated process control, that, in turn, interface with the motors of the vehicle to regulate steering, acceleration and throttle. (Daniel L. Rosenban, 2017)

Self-driving vehicle needs a range of technologies that need to be identified in order to be applied, such as Machine Awareness, Sensor Integration, Deep Learning, Route Mapping, Actuator. Computer’s visual perception helps recognize how devices can be enabled to extract knowledge from visual images or videos. From the technical point of view, it is used to simplify functions where humans imagine how the device would implement it. Sensor integration incorporates a number of sensory input or information obtained among multiple references so that the resultant data has fewer ambiguity about them than it would have been if all sources were employed separately. Deep learning is another aspect of ML which focuses on studying data models, like conflicting task-specific algorithms utilized. Track mapping for automated mobile robots, and so on. This helps robots to determine the quickest or best way from two points. Instead that optimal routes may be tracks which minimize the number of spinning, the number of stopping, or a particular implementation need. The actuator is necessary for shifting the mechanism. (Mochamad Vicky Ghani Aziz, Hilwadi Hindersah, Ary, 2017)

CHAPTER 3 -DESIGN and COMPONENTS

By examining how autonomous cars are designed, the materials used, and which materials they are made of; we can see how this important technology developed by time and get a sense of how it could advance in the near future.



3.1Components

3.1.1 Raspberry Pi 3B+

Raspberry Pi 3B+ is mini personal. It is commonly used for actual image/video processing, IoT-based applications and robotics platforms. The official operating system available to use is the Raspbian OS. The operating system is stored in the SD card which is usually recommended to be more than 8 GB. For creating a program, the raspberry pi offers access to on-chip hardware, i.e., GPIOs. We can attach devices such as LEDs, motors, sensors, etc. by using GPIO and can manage them as well. (Raspberry pi Web site, n.d.)

Figure 1 Raspberry Pi

Specifications:

- The CPU type is ARM Cortex-A53 with a speed of 1.4 GHz
- The RAM size of this version is 1GB SRAM
- Wi-Fi Integrated in this Pi model is 2.4GHz and 5GHz
- Ethernet speed is around 300 MPs
- this model has PoE
- it has 4 USB port inputs and 1 HDMI port input
- SD micro card support included
- It can handle to a maximum of 5V/2.5A DC and a minimum of 4V/2.4A
- Operating temperature of this model can handle between 0 and 50°C
- The Quad core with 64-bit processor is 1.4GHz
- In Built LED

3.1.2 Arduino UNO R3

Arduino Uno R3 is one type of microcontroller board based on ATmega328P. It contains everything you need to keep the microcontroller; simply mount it to a PC with assistance Using .an AC-DC converter or battery to start the supply with a USB cable

The uno does not use the FTDI USB-to-Serial driver chip, but is distinct from all previous boards. Built as a USB-to-serial adapter, it features an ATmega16U2. This auxiliary microcontroller has a USB bootloader that can be reprogrammed by experienced users. (David Kushner, 2011)

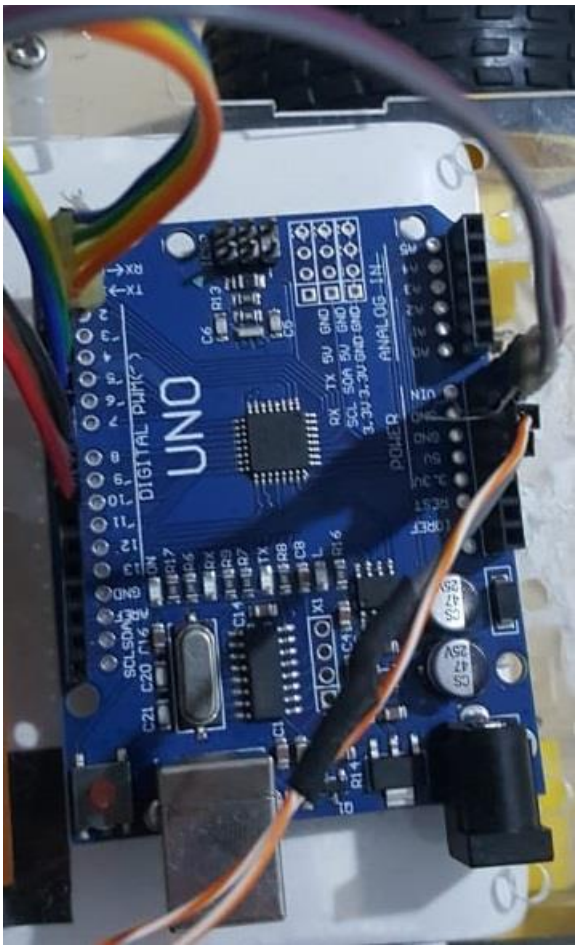


Figure 2 Arduino UNO

Specifications:

Built in ATmega328P microcontroller

Operational stress of 5V

I/P tension ranges between 7 to 12V

I/P voltage ranges between 6 and 20V

I/P pins of the analog is 6

Pins-14 are for optical input and output

The present DC is 20 mA for any I/O pin

3.3V Pin present is 50 mA

Boot loader uses Flash Drive -32 KB and 0.5 KB of memory

EEPROM of 1 KB

SRAM of 2 KB

The CLK has 16 MHz rpm

Built in LED

3.1.3 H-bridge

The L298N has the power to drive a pair of DC motors with a two-channel H-Bridge engine. This means that it can drive up to two engines independently making it suitable for two-wheel robot platforms.

The 3-pin 3,5-mm pitch screw terminals drive the motor driver assembly L298N. It is composed of motor (V), floor and 5V logical power supply pins (Vss).

The motor L298N IC has two input pins, in fact. Vss and Vs and Vs. The power for the driving of the 5 to 35V motors is supplied by the Vs pin H-Bridge. Vss is used for performing a 5 to 7V logic circuit. And both fall into a typical soil called 'GND.'

The module has a STMicroelectronics 78M05 5V on-board regulator. You can turn it on or off with a switch. (In-Depth: Interface L298N DC Motor Driver Module with Arduino, 2020)

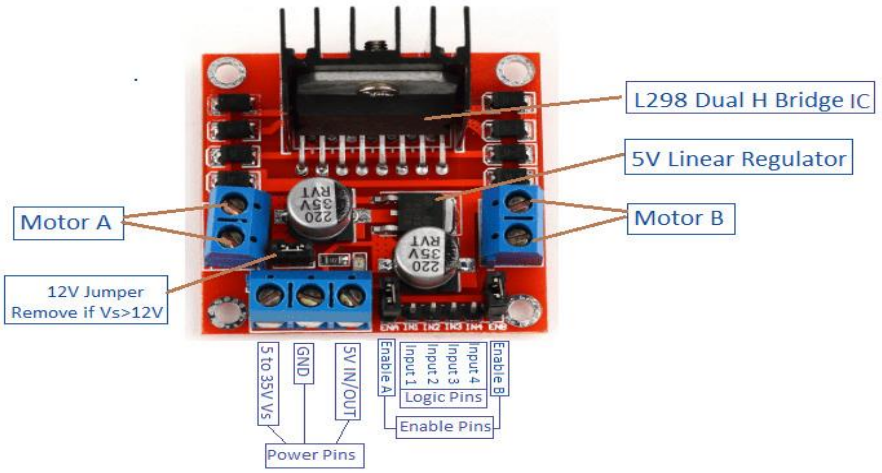


Figure 3 H-Bridge (MOTOR MODULE)

H-Bridge Pinout

VCC: Pin provides engine control. It can be from 5 to 35V everywhere. Remember to provide 2 additional volts from the real voltage of the generator while the 5V-EN jumper is in place such that optimum speed is achieved from the motor.

GND: It's a mutual pin ground.

V pin: Supplies control to the switches in the L298N IC for logic loops. The power pin is used for powering your Arduino if a 5V-EN jumper is in position. You need to attach the 5V-EN jumper on Arduino to the 5V pin.

ENA :The pins are used for regulating the speed of engine A. If the HIGH pin is pushed, the motor A will be reversed, pushing LOW will stop the engine. This pin will stop the engine A. If the jumper is removed and the pin attached to the PWM entry, the engine A speed is controlled.

IN1 & IN2: Motor A's spinning orientation is regulated by pins, while if one is HIGH and the other is LOW, the Motor A spins. The motor A is stopped when all inputs are high and low.

The IN3 & IN4: pins control motor B's path. If one is HIGH and the other LOW, then the B"s motor spins. The engine B will stop if all inputs are HIGH or LOW.

ENB: pins are used to control engine speed B. When this pin is pulled HIGH(Maintaining the jumper in place), the motor B turns, and the motor fails if pulled Short. With this pin removed and attached to PWM, we can control the speed of Motor B.

OUT1 & OUT2: Engine A pins are connected.

OUT3 & OUT4: Engine attaches pins

3.1.4 Power Supply

Figure 4 Power Bank (Power Supply)

A power bank contains one or two integrated lithium batteries cells. They basically operate like normal battery-powered cells and can be charged using the available connection or a USB connection. We used this ANKER power bank because the Raspberry Pi needs of 5V – 2.5A of power and this model can give a maximum of 5V – 3A power that enough to run the raspberry pi, the DC motors and the other components. Below are the specifications of this power bank model.

Specifications:

battery with a capacity of 13000 mah (46.8 W/h)

1 USB input with 5V and 2A

2 micro-USB outputs with 5V and 3A maximum per each port

fast recharging when using a 2A USB charger

Measure of battery level

3.2 Connections and Design

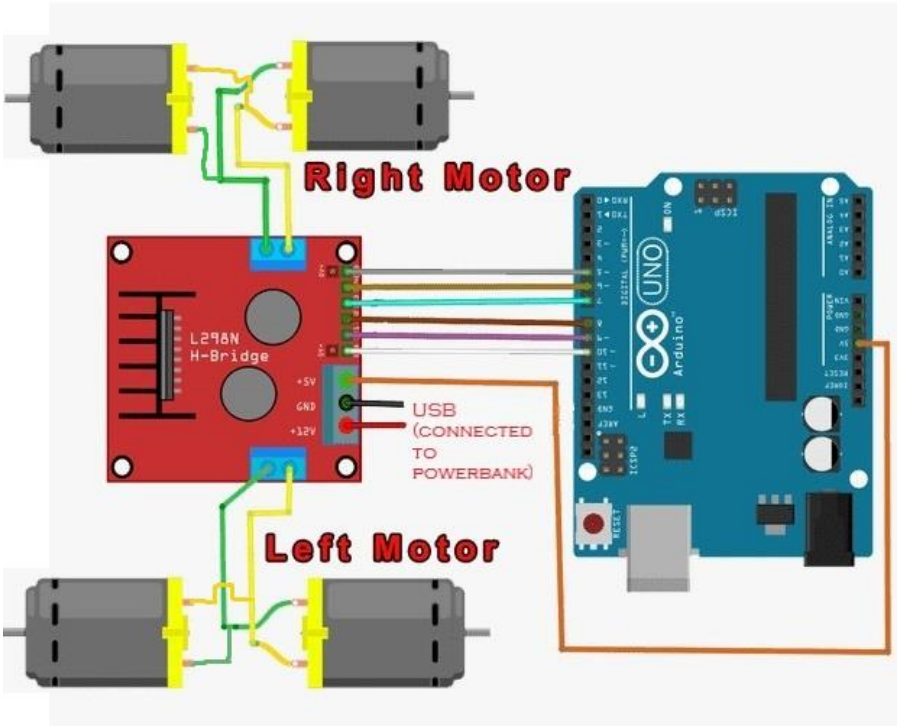


Figure 5 DC Motors and Arduino UNO Connections

Step by step (PART A):

1- We assembled two plates robot chassis with a width of 256 mm and a height of 150 mm for each plate with a free space around 50 mm in between.
2- We installed 4 motors to the lower part of the chassis. Two motors on each side. The motors on each side were connected in parallel as shown Figure 5.
3- The motors on the right are connected to the output pin on the right side of the H-bridge (placed in the free space of the chassis) and the same goes for the left side as shown in Figure 5.
4- The power bank was placed next to the H-bridge.
5- The H-bridge was connected to the power bank through the power supply pins using a USB cable. The USB had two wires (phase and GND) connecting the phase wire to VCC pin and the GND wire to the GND pin of the H-bridge.
6- Next, we placed the Arduino on top the chassis.

7- Then the wheels were placed to the motors.
8- From the +5V pin of the power supply, we connected a wire to the power pins of +5V pin of the Arduino. The purpose of this connection is to power up the Arduino to command the H-bridge using the code uploaded on Arduino board.
9- 4 F-M jumper wires from the Arduino's digital PWM pins ~9, 8,7, ~6 connected to the 4 inputs in the direction control pins of the H-bridge to navigate the car. Then, two F-M jumper wires were connected from the PWM pins: ~10, ~5 to the ENABLE (EN) inputs of the speed control pins.
    a. ENA, IN1 and IN2 pins connected to ~10, ~9, and 8 respectively used to control the left-side motors.
    b. ENB, IN3 and IN4 pins connected to ~5, ~6, and 7 respectively used to control the right-side motors (connections are as shown in Figure 5).

The connections of the PCB:

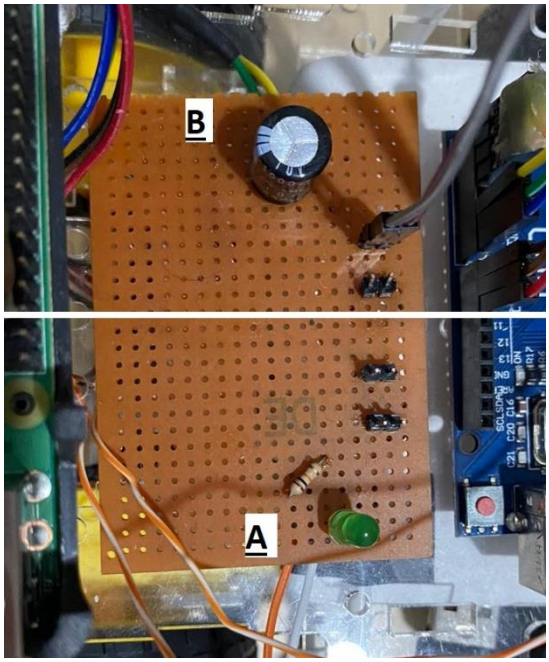


Figure 6 PCB Board

We soldered the following parts on the PCB board in parallel as shown in Figure 6

The Parts were:

1- Capacitor 1000 µF – 10V

2- 4x two male pins
3- Green LED and Resistor 1k Ω (Series Connection)

Step by step (PART B):

1- The PCB Board was placed on the top of the chassis.
2- We connected the USB cable to A and the USB Micro-B to B as shown in Figure 6.
3- We connected two F-M Jumper wires from the male pins on the board to VIN and GND power pins on the Arduino.

Step by step (Part C):

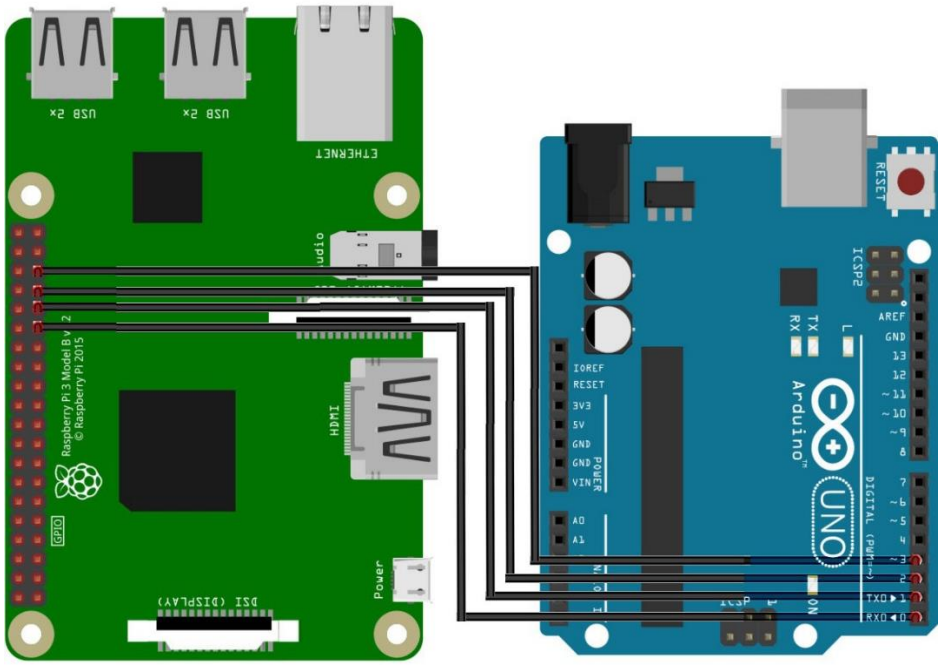


Figure 7 Raspberry pi connections

1- The raspberry was placed on the top of the chassis.
2- The camera was placed at the rear of the car.
3- The 15-pin flex cable connected to the camera from the raspberry pi camera port.
   a- 4 F-M jumper wires were connected from GPIO 5, 6, 13 and 19 pins (29, 31, 33 and 35 respectively) to PWM pins 0, 1, 2, ~3 as shown in Figure 7. The Arduino is a slave device that works according to the Pi's commands. It all depends on what the camera detects to send data to the Arduino to control the motors accordingly.

b- Pin 0 (RX receives data from the Pi)
c- Pin 1 (TX transmits data to the pi)

Final design:

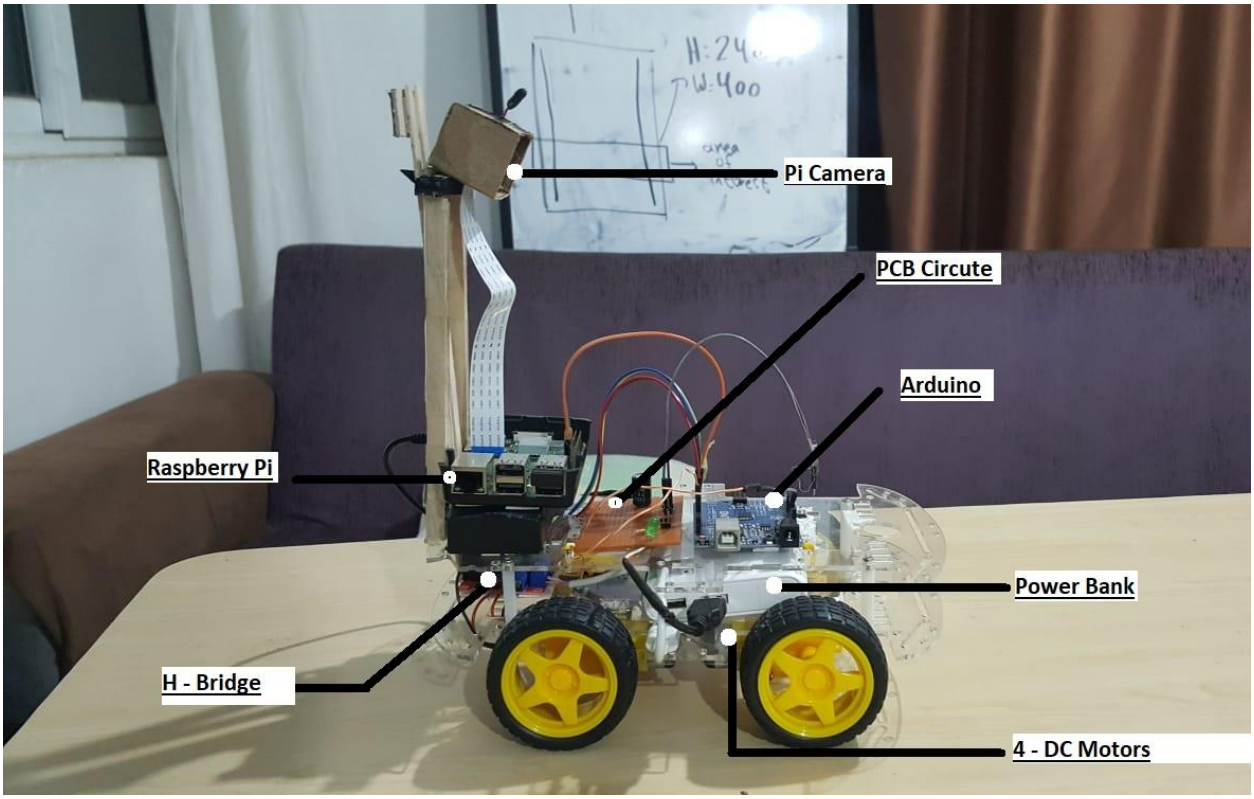


Figure 8 Final Design

Chapter 4 – COMPUTER VISION AND CAR MOVMENT

Computer vision technologies are currently one of the fastest expanding industries of automation including robotics, along with other related scientific as well as technical fields, including mechanics, intelligent transportation and communication, biomedical engineering and also the food industry. However, automation and robotics continue to become among the advancing fields of practical implementation for newly evolved artificially intelligent technologies, especially computer as well as machine vision algorithms. Computer Vision (CV) helps a Raspberry Pi to recognize objects in our project. In real world terms, this implies that now the Raspberry Pi can easily examine an image, search for objects of interest, but also identify faces and texts. (Simon Monk, 2016)

4.1 Track building

Building the track in a specific way was crucial in our project. We built a pitch-black track with white lines on the end to make it easier to differentiate between colors when using image processing. In digital imaging, the smallest bit of data in an image is a pixel. In color schemes, there are usually three-pixel components such as red, green and blue; each pixel has three components (RGB). This method is used to convert the original image to a quantitative value by converting it to a black and white pixmap image. The distance between the two white lines were approximately 30 cm and each line had a width of 1 cm. In the code, we taught the car to balance its course of action to stay between the two white lines.

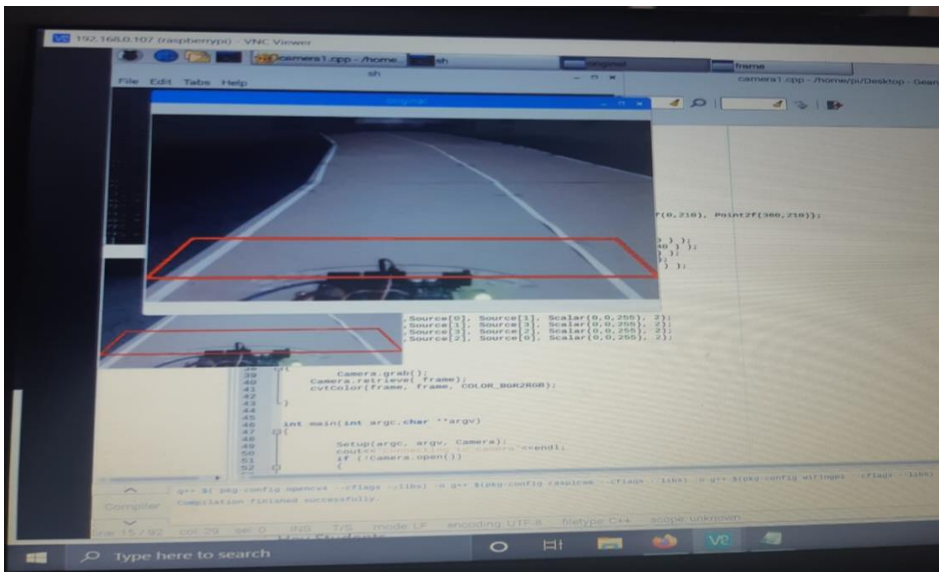


Figure 9 Lane of Interest

4.2 Cascade Training and Testing

We chose Cascade for the training component here in our project: Cascade is a learning technique that uses cascade function from many positive as well as negative pictures. The artifacts in many other pictures are then observed. Then it uses all this to try to detect objects in any image but first we need to have a collection of positive samples (with items that we want to recognize) as well as negative images, including anything we do not want to detect for the training. We manually prepared a series of negative samples, while the program opencv create samples produces a collection of positive samples.

Negative Samples

Negative Samples of the objects we would like to detect are drawn from abstract pictures. This negative image that are used to produce the samples should be described as an absolute as well as relative image file with one direction per section. Background samples are often known as negative samples as well as reference images. The images mentioned can vary in scale. That being said, each image should have a scale equal to or higher than that of the target scaling of the training window. provided that such pictures are used to test a particular negative image in multiple picture samples of that size. An example of such a negative description in Figure 10

Figure 10 Negative image sample

Positive Samples

They are used to determine what the system might specifically look for when attempting to locate the items. We can try to generate from one single sample image all the positive needed but we supplied the cascade with all the positive images needed and used the tool to cut them in the right dimensions and resized them (400x240). An example of this in figure 11.

Figure 11 Positive Image sample

Testing

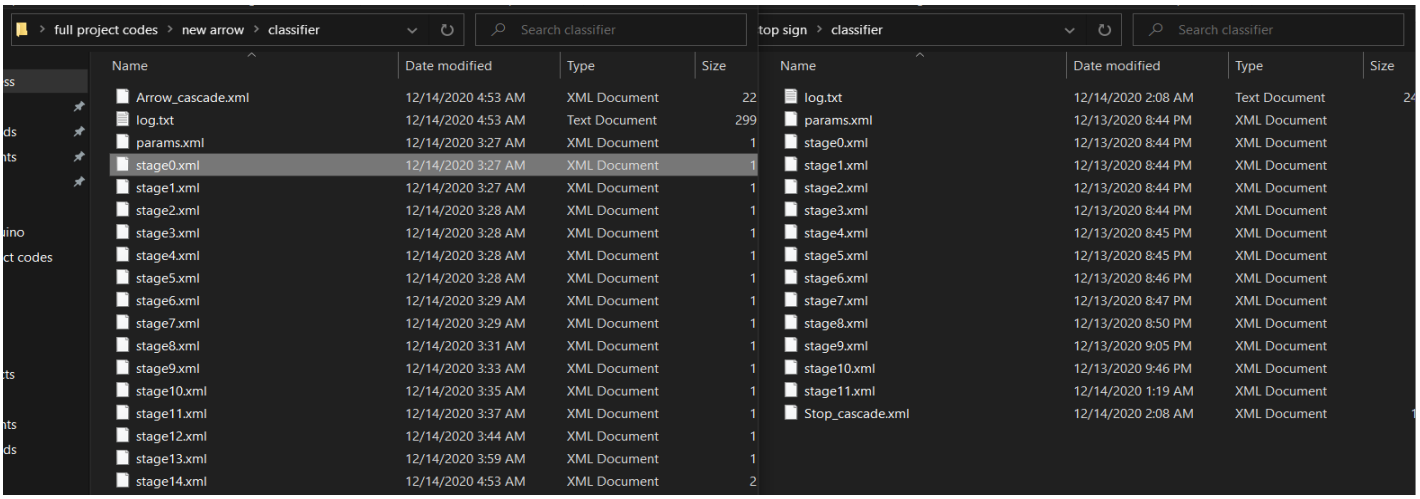


Figure 12 Cascade Training results

The next stage is the actual teaching of the reinforced cascade for our classifier on the basis of previously arranged positive and negative datasets. After the training is completed, a folder named “classifier” will be ready to use. In this folder, it will show the number of stages that were done in the process according to the assigned numbers of stages, width, height and type of the feature used. As shown in Figure 12, two .xml documents were made, one for the Stop sign and the other for the Arrow sign. We will use these .xml files to check the accuracy of its detection by testing the positive images we have in our Cascade Trainer GUI. In Figure 13, it shows a sample of the detected photos after it gets detected successfully. These .xml files are the results of our training that we will use in our code in the raspberry pi.

Figure 13 Cascade testing results

4.3 Car movement

In this part we used the Arduino IDE software to write a code that can control the movement of our car by uploading it to the Arduino microcontroller board. Using specific functions provided in this software and applying them in raspberry code; we will control the movement of our car.

Functions used:

1- Forward: move the car forward.
2- Backward: move the car backward.
3- Left 1,2 and 3: move the car left depending on different angles of turning.
4- Right 1,2 and 3: move the car right depending on different angles of turning.
5- Stop: stops the movement of the car.
6- Stop End: stops the car by detecting a Lane End code provided in raspberry code.
7- Arrow: after detecting an arrow sign, the car will apply the code provided in this function depending on the distance between the arrow sign and the car.

Stop sign: different function applied in ELSE IF loop to let the car stop for a certain period of time.

CHAPTER 5 – COST ANALYSIS

In this project, we adopted a systemic procedure by evaluating the advantages as well as disadvantages of our methods to find alternatives that provide the optimal components required in our project to produce the best outcomes while maintaining savings. Here in this figure, we show a list of the components used together with the cost.

Components - Bill of quantity			
Item	Description	Quantity	Cost (TL)
1	4WD robot Chassis	1	63.08
2	L298N H-bridge motor module	1	13.95
3	Raspberry pi Camera V2	1	298.01
4	Male-Female Pin Jumper Cables	40	4.82
5	Arduino Uno R3 Clone	1	28.75
6	Male Pin Headers 1x40	4	3.98
7	PCB Board	1	2.36
8	1000 µF Capacitor - 10 V	4	2.3
9	LED (RED,GREEN,YELLOW)	50	26.76
10	Raspberry pi 3b+	1	273.16
11	Electric Tape	4	10
12	Paperboard	5	50
13	Electric Soldering Iron	1	37.29
	Total		814.46

Figure 14 Bill of Quantity.

The raspberry pi 3B+, L298N H-bridge motor module, Raspberry pi Camera V2 and Arduino Uno R3 Clone are considered to be indirect costs; meaning that the costs are fixed everywhere. We used the Arduino Uno R3 Clone instead of the original as it is more cost-efficient while achieve the desired results. The rest are classified as direct costs as their prices aren't constant and vary depending on the seller.

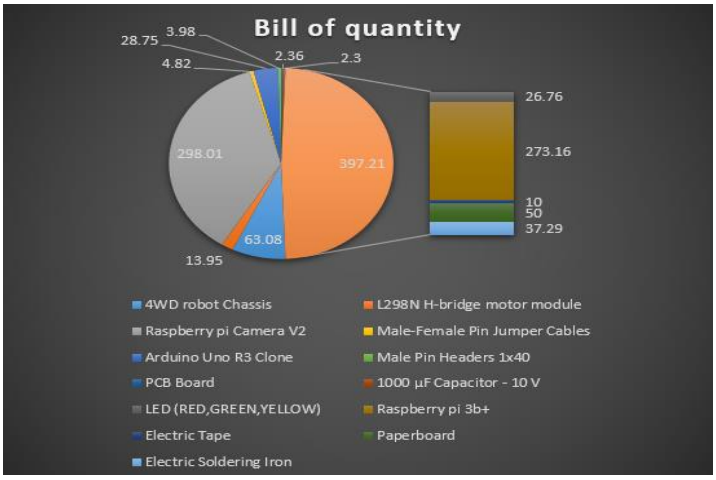


Figure 15 Bill of Quantity (Chart)

CHAPTER 6 – FINAL OUTCOMES

After designing the track and placing the signs in their assigned spots, we ran the codes on both the Arduino and Raspberry Pi to practically test the car. The car detected the signs successfully and followed the course of action as instructed in the code.

6.1 Achieved Results

The car moved in a stable manner between the two lines as instructed in the code. We used two functions: LaneFinder and LaneCenter for that purpose. As a result, if the car strayed away from the center, it will automatically adjust itself back. While car is moving, it will state the angle of movement by showing the stats on the screen stating whether its slightly moving to the left or right as shown in Figure 16. The pi will send commands to the Arduino to activate the code based on the values of the LaneFinder function.

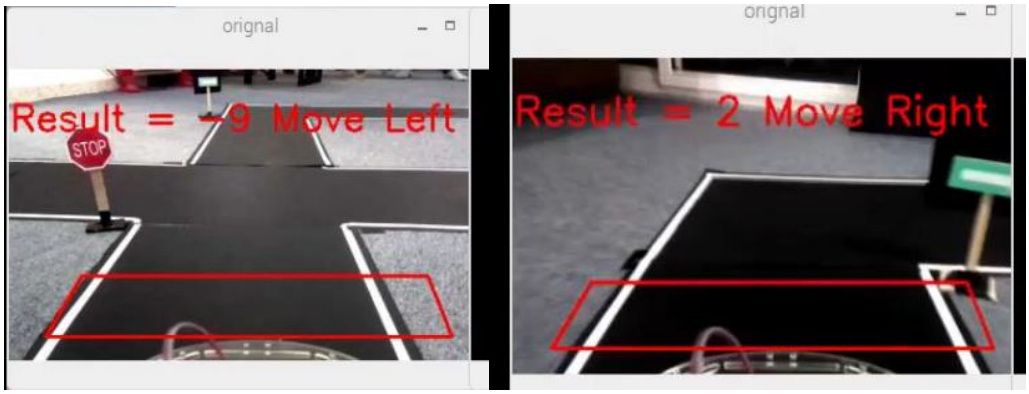


Figure 16 Center Adjustment

Based on the cascade training, the car was able to detect the signs successfully while moving. Every time the car detects a sign, it calculates the distance from the vehicle's perspective to the sign by using the functions: Stop_detection, Arrow_detection. We calculated the distance using the formula y=mx+b. The car was programmed to stop when the distance is less than a certain range (In our code it was 30cm).

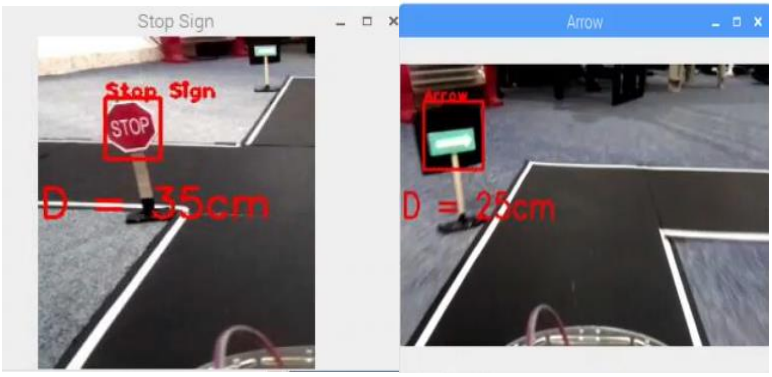


Figure 17 Sign Detections

The Histrogram function measures the noise in the background. The noise is considered as white pixels, so as the noise increased the number of white pixels displayed increases. That is displayed on the Terminal window as numbers. The variable LaneEnd was defined in that function to run when the car exceeds a specific number of noise to stop for a certain amount of time (In our code it was 6000).

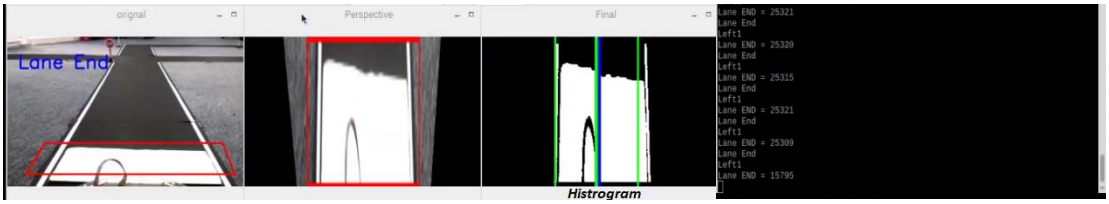


Figure 18 LaneEnd Detection

6.2 Problems Encountered

- The power supply wasn't providing enough voltage to power the pi at its optimal performance when the power bank wasn't fully charged or about to die.
- Due to high usage in CPU the temperature increased above it's limit which led to sudden reboot.
- The IP address kept changing kept changing because we had two internet modems, as a result the connection to the VNC viewer kept getting interrupted.
- Due to the change in light environment, the brightness wasn't stable which made the detection and staying on track more difficult.
- As the code had a lot of lines, the computational power of the pi affected the fps ratio which caused a lag in the camera. That made it harder for the camera to detect signs accurately.
- We used images with high resolution in our training which required a more powerful CPU than the one in the raspberry pi.

6.3 Future Enhancements

- Use a more stable power supply with a higher voltage output. As the project used a capacitor, it wouldn't matter if the voltage got above the required level because it will regulate the power going to the raspberry pi. This way we will avoid sudden reboot when the power supply level is low.
- Use a better version of raspberry pi to handle the pressure on the CPU (Ex: Raspberry pi 4 has a better CPU power and more RAMs)
- Optimize the code to lessen the number of lines to decrease the load on the CPU.
- Install a fan on the pi board to avoid overheat.
- Use a more stable IP address.

- Train the program with low resolution levels for a faster training process and performance.
- Using more images of varying intensities of light.
- Choose a particular location and try to maintain the brightness.

CHAPTER 7 – CONCLUSION

A low-cost prototype is built of a self-driving car platform and all features are seen effectively. The car is able to accurately monitor the lane with Pi camera and the signs are identified and the car takes decisions using image recognition techniques to interact with traffic laws in actual environments. Car can distinguish between actual objects as well as correctly respond to the directions issued and successfully identify and resolve obstacles.

Mostly with proposed model which decreases the human job in running the automobile, we addressed the issue of non-autonomous automobiles (Human error). In addition, this project can give an idea on how the autonomous can be much better than just a typical user. Since results is higher and more reliable, the proposed solution will overcome the fundamental human mistake that actually occurs.

CHAPTER 8 – REFERENCES

References

(n.d.). Retrieved from Raspberry pi Web site: https://www.raspberrypi.org/about/

Daniel L. Rosenban. (2017). “Inside Waymo’s Self-Driving Car: My Favorite Transistor. *Symposium on VLSI Circuits Digest of Technical Paper 2017*.

David Kushner. (2011). The Making of Arduino. *IEEE SPECTRUM*.

In-Depth: Interface L298N DC Motor Driver Module with Arduino. (2020, December 27).

Mochamad Vicky Ghani Aziz, Hilwadi Hindersah, Ary. (2017). Implementation of Vehicle Detection Algorithm for Self-Driving Car on Toll Road Cipularang using Python Language. *4th international conference on Electric Vehicular Technology 2017.*

Simon Monk. (2016). Computer vision with the Raspberry Pi. In O'REILLY, *Raspberry Pi Cookbook.*

CHAPTER 9 – APPENDIX

9.1 Appendix 1 / C++ code

```
#include <opencv2/opencv.hpp>
#include <raspicam_cv.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <wiringPi.h>

using namespace std;
using namespace cv;
using namespace raspicam;

// Image Processing variables
Mat frame, Matrix, framePers, frameGray, frameThresh, frameEdge, frameFinal,
frameFinalDuplicate, frameFinalDuplicate1;
Mat ROILane, ROILaneEnd;
int LeftLanePos, RightLanePos, frameCenter, laneCenter, Result, laneEnd;

RaspiCam_Cv Camera;

stringstream ss;


vector<int> histrogramLane;
vector<int> histrogramLaneEnd;

Point2f Source[] = {Point2f(60,170),Point2f(345,170),Point2f(30,220),
Point2f(365,220)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),Point2f(100,240),
Point2f(280,240)};


//Machine Learning variables
CascadeClassifier Stop_Cascade, Arrow_Cascade;
Mat frame_Stop, RoI_Stop, gray_Stop, frame_Arrow, RoI_Arrow, gray_Arrow;
vector<Rect> Stop, Arrow;
int dist_Stop, dist_Arrow;

 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
```

```
    Camera.set ( CAP_PROP_FRAME_WIDTH,   ( "-w",argc,argv,400 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,   ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,60) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
    Camera.set ( CAP_PROP_SATURATION,   ( "-sa",argc,argv,50 ) );
    Camera.set ( CAP_PROP_GAIN,   ( "-g",argc,argv ,50) );
    Camera.set ( CAP_PROP_FPS,   ( "-fps",argc,argv,0));

}

void Capture()
{
    Camera.grab();
    Camera.retrieve( frame);
    cvtColor(frame, frame_Stop, COLOR_BGR2RGB);
    cvtColor(frame, frame_Arrow, COLOR_BGR2RGB);
    cvtColor(frame, frame, COLOR_BGR2RGB);



}

void Perspective()
{
        line(frame,Source[0], Source[1], Scalar(0,0,255), 2);
        line(frame,Source[1], Source[3], Scalar(0,0,255), 2);
        line(frame,Source[3], Source[2], Scalar(0,0,255), 2);
        line(frame,Source[2], Source[0], Scalar(0,0,255), 2);


        Matrix = getPerspectiveTransform(Source, Destination);
        warpPerspective(frame, framePers, Matrix, Size(400,240));
}

void Threshold()
{
        cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
        inRange(frameGray, 240, 255, frameThresh);
        Canny(frameGray,frameEdge, 900, 900, 3, false);
        add(frameThresh, frameEdge, frameFinal);
        cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
        cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);   //used
in histrogram function only
        cvtColor(frameFinal, frameFinalDuplicate1, COLOR_RGB2BGR);   //used
in histrogram function only
```

```
}

void Histrogram()
{
    histrogramLane.resize(400);
    histrogramLane.clear();

    for(int i=0; i<400; i++)       //frame.size().width = 400
    {
        ROILane = frameFinalDuplicate(Rect(i,140,1,100));
        divide(255, ROILane, ROILane);
        histrogramLane.push_back((int)(sum(ROILane)[0]));
    }

        histrogramLaneEnd.resize(400);
    histrogramLaneEnd.clear();
        for (int i = 0; i < 400; i++)
        {
                ROILaneEnd = frameFinalDuplicate1(Rect(i, 0, 1, 240));
                divide(255, ROILaneEnd, ROILaneEnd);
                histrogramLaneEnd.push_back((int)(sum(ROILaneEnd)[0]));


        }
           laneEnd = sum(histrogramLaneEnd)[0];
           cout<<"Lane END = "<<laneEnd<<endl;
}

void LaneFinder()
{
    vector<int>:: iterator LeftPtr;
    LeftPtr = max_element(histrogramLane.begin(), histrogramLane.begin() +
150);
    LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

    vector<int>:: iterator RightPtr;
    RightPtr = max_element(histrogramLane.begin() +250,
histrogramLane.end());
    RightLanePos = distance(histrogramLane.begin(), RightPtr);

    line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos, 240),
Scalar(0, 255,0), 2);
    line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLanePos, 240),
Scalar(0,255,0), 2);
```

```
}

void LaneCenter()
{
    laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
    frameCenter = 188;

    line(frameFinal, Point2f(laneCenter,0), Point2f(laneCenter,240),
Scalar(0,255,0), 3);
    line(frameFinal, Point2f(frameCenter,0), Point2f(frameCenter,240),
Scalar(255,0,0), 3);

    Result = laneCenter-frameCenter;
}

void Stop_detection()
{
    if(!Stop_Cascade.load("//home//pi/Desktop//MACHINE
LEARNING//Stop_cascade.xml"))
    {
        printf("Unable to open stop cascade file");
    }
    RoI_Stop = frame_Stop(Rect(0,0,200,240));
    cvtColor(RoI_Stop, gray_Stop, COLOR_RGB2GRAY);
    equalizeHist(gray_Stop, gray_Stop);
    Stop_Cascade.detectMultiScale(gray_Stop, Stop);

    for(int i=0; i<Stop.size(); i++)
    {
        Point P1(Stop[i].x, Stop[i].y);
        Point P2(Stop[i].x + Stop[i].width, Stop[i].x + Stop[i].height);

        rectangle(RoI_Stop, P1, P2, Scalar(0, 0, 255), 2);
        putText(RoI_Stop, "Stop Sign", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0,
0, 255, 255), 2);
        dist_Stop = (-0.91)*(P2.x-P1.x) + 71.86;

       ss.str(" ");
       ss.clear();
       ss<<"D = "<<dist_Stop<<"cm";
       putText(RoI_Stop, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);
    }
}
```

```
void Arrow_detection()
{
    if(!Arrow_Cascade.load("//home//pi/Desktop//MACHINE
LEARNING//Arrow_cascade.xml"))
    {
        printf("Unable to open Arrow cascade file");
    }
    RoI_Arrow = frame_Arrow(Rect(0,0,200,240));
    cvtColor(RoI_Arrow, gray_Arrow, COLOR_RGB2GRAY);
    equalizeHist(gray_Arrow, gray_Arrow);
    Arrow_Cascade.detectMultiScale(gray_Arrow, Arrow);

    for(int i=0; i<Arrow.size(); i++)
    {
        Point P1(Arrow[i].x, Arrow[i].y);
        Point P2(Arrow[i].x + Arrow[i].width, Arrow[i].x + Arrow[i].height);

        rectangle(RoI_Arrow, P1, P2, Scalar(0, 0, 255), 2);
        putText(RoI_Arrow, "Arrow", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0,
255, 255), 2);
        dist_Arrow = (-0.75)*(P2.x-P1.x) + 73.5;

       ss.str(" ");
       ss.clear();
       ss<<"D = "<< dist_Arrow <<"cm";
       putText(RoI_Arrow, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255),
2);
    }
}

int main(int argc,char **argv)
{

    wiringPiSetup();
    pinMode(21, OUTPUT);
    pinMode(22, OUTPUT);
    pinMode(23, OUTPUT);
    pinMode(24, OUTPUT);

    Setup(argc, argv, Camera);
    cout<<"Connecting to camera"<<endl;
    if (!Camera.open())
    {
```

```
    cout<<"Failed to Connect"<<endl;
}

    cout<<"Camera Id = "<<Camera.getId()<<endl;


while(1)
{

auto start = std::chrono::system_clock::now();

Capture();
Perspective();
Threshold();
Histrogram();
LaneFinder();
LaneCenter();
Stop_detection();
Arrow_detection();


 if (dist_Stop > 5 && dist_Stop < 30)
{
    digitalWrite(21, 0);
    digitalWrite(22, 0);    //decimal = 8
    digitalWrite(23, 0);
    digitalWrite(24, 1);
    cout<<"Stop Sign"<<endl;
    dist_Stop = 0;

    goto Stop_Sign;
}


 if (dist_Arrow > 5 && dist_Arrow < 30)
{
    digitalWrite(21, 1);
    digitalWrite(22, 0);    //decimal = 9
    digitalWrite(23, 0);
    digitalWrite(24, 1);
    cout<<"Arrow"<<endl;
    dist_Arrow = 0;

    goto Arrow;
```

```
}


if (laneEnd > 6000)
{
    digitalWrite(21, 1);
    digitalWrite(22, 1);    //decimal = 7
    digitalWrite(23, 1);
    digitalWrite(24, 0);
    cout<<"Lane End"<<endl;
}


if (Result == 0)
{
    digitalWrite(21, 0);
    digitalWrite(22, 0);    //decimal = 0
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    cout<<"Forward"<<endl;
}


else if (Result >0 && Result <10)
{
    digitalWrite(21, 1);
    digitalWrite(22, 0);    //decimal = 1
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    cout<<"Right1"<<endl;
}

    else if (Result >=10 && Result <20)
{
    digitalWrite(21, 0);
    digitalWrite(22, 1);    //decimal = 2
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    cout<<"Right2"<<endl;
}

    else if (Result >20)
{
    digitalWrite(21, 1);
    digitalWrite(22, 1);    //decimal = 3
```

```
        digitalWrite(23, 0);
        digitalWrite(24, 0);
        cout<<"Right3"<<endl;
    }

        else if (Result <0 && Result >-10)
    {
        digitalWrite(21, 0);
        digitalWrite(22, 0);    //decimal = 4
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left1"<<endl;
    }

        else if (Result <=-10 && Result >-20)
    {
        digitalWrite(21, 1);
        digitalWrite(22, 0);    //decimal = 5
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left2"<<endl;
    }

        else if (Result <-20)
    {
        digitalWrite(21, 0);
        digitalWrite(22, 1);    //decimal = 6
        digitalWrite(23, 1);
        digitalWrite(24, 0);
        cout<<"Left3"<<endl;
    }

     Stop_Sign:
     Arrow:


   if (laneEnd > 7000)
    {
       ss.str(" ");
       ss.clear();
       ss<<" Lane End";
       putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(255,0,0), 2);

     }
```

```
else if (Result == 0)
{
   ss.str(" ");
   ss.clear();
   ss<<"Result = "<<Result<<" Move Forward";
   putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

 }

else if (Result > 0)
{
   ss.str(" ");
   ss.clear();
   ss<<"Result = "<<Result<<" Move Right";
   putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

 }

 else if (Result < 0)
{
   ss.str(" ");
   ss.clear();
   ss<<"Result = "<<Result<<" Move Left";
   putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

 }


namedWindow("orignal", WINDOW_KEEPRATIO);
moveWindow("orignal", 0, 100);
resizeWindow("orignal", 420, 300);
imshow("orignal", frame);

namedWindow("Perspective", WINDOW_KEEPRATIO);
moveWindow("Perspective", 400, 100);
resizeWindow("Perspective", 420, 300);
imshow("Perspective", framePers);

namedWindow("Final", WINDOW_KEEPRATIO);
moveWindow("Final", 800, 100);
resizeWindow("Final", 420, 300);
imshow("Final", frameFinal);

namedWindow("Stop Sign", WINDOW_KEEPRATIO);
moveWindow("Stop Sign", 1200, 100);
```

```
    resizeWindow("Stop Sign", 420, 300);
    imshow("Stop Sign", RoI_Stop);

    namedWindow("Arrow", WINDOW_KEEPRATIO);
    moveWindow("Arrow", 1200, 450);
    resizeWindow("Arrow", 420, 300);
    imshow("Arrow", RoI_Arrow);

    waitKey(1);
    auto end = std::chrono::system_clock::now();
    std::chrono::duration<double> elapsed_seconds = end-start;

   float t = elapsed_seconds.count();
   int FPS = 1/t;
   // cout<<"FPS = "<<FPS<<endl;

    }


    return 0;

}
```

9.2 Appendix 2 / Arduino 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;


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);


}

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

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

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

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

}


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

```
  analogWrite(EnableL,255);

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

}

void Stop()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,0);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,0);

}

void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,100);

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

}

void Left2()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,30);

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

}
```

```
void Left3()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,10);

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

}

void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,100);   //200

}
void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,30);    //160

}

void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,10);    //100
```

```
}

void Stop_end()
{

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,0);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,0);
  delay(20000);

}

void Arrow()
{

  analogWrite(EnableL, 0);            //stop
  analogWrite(EnableR, 0);
  delay(300);


  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);            //forward
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(1100);

  analogWrite(EnableL, 0);            //stop
  analogWrite(EnableR, 0);
  delay(300);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);          //right
  digitalWrite(LowR, LOW);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(1000);

  analogWrite(EnableL, 0);            //stop
```

```
  analogWrite(EnableR, 0);
  delay(300);

}


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

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

  else if(data==2)
   {
     Right2();
   }

  else if(data==3)
   {
     Right3();
   }

  else if(data==4)
   {
     Left1();
   }

  else if(data==5)
   {
     Left2();
   }

  else if(data==6)
   {
     Left3();
   }

  else if (data==7)
```

```
    {
      Stop_end();
    }


  else if (data==8)
    {
       analogWrite(EnableL, 0);
       analogWrite(EnableR, 0);
       delay(7000);                          // stop sign detection

       analogWrite(EnableL, 150);
       analogWrite(EnableR, 150);
       delay(1000);
    }

   else if(data==9)
    {
      Arrow();
    }


      else if(data>9)
    {
      Stop();
    }



}
```