Project Report

Embedded Linux System Report

Omar Farag Rashed 1061007

Supervised By: Eng. Tasnim Basmaji

Submitted: June 2, 2021

Contents

List of Figures

List of Tables

Abstract

In this report I aim to create an embedded system that will control the robot to go through a parking plot to check if he has paid Mawaqif's fee through using computer vision to read the license plate. To conduct this project I need a Raspberry Pi, Arduino UNO, Pi Camera, and a Line Follower. The Raspberry pi and the Ardunio UNO communicated to each other through I2C where the Raspberry pi will let the Ardunio UNO know when to stop. The Raspberry Pi will extract the license plate using tesseract Optical Character Recognition then the Ardunio UNO will be able to control the robot using the reading from the Line follower.

1 Introduction

Mawaqif's staff have a hard time working in this country specially because of the constant heat during day time, so imagine having a robot that will do that job for them which is the goal of this project. By making use of both Raspberry Pi and Arduino UNO I will be able to control both the robot and detect the car's license plate.

1.1 Motivation

We hope to help Mawaqif's staff working out during the heat of the day time by preventing them from getting illnesses with this project as it can allow the robot to take over during the day time where the heat is it's highest and allow these Mawaqif's workers to takeover during the night.

1.2 Problem Statement

Since UAE will always have a hot weather because it was built over a desert people working out doors have to suffer from this heat which can cause sun burn, exhaustion, and other illnesses, and Mawaqif's staff face this problem everyday and due to exhaustion can make mistakes when issuing fines.

1.3 Literature Review

The first paper talks about the development of a hand held embedded system device that will automatically recognize license plate which is aimed to help security employees at the entrances by helping them to automatically check if the vehicle is authorized or not from a provisioned database with a list of authorized vehicles. The device is centered around the raspberry pi which works with the help the optical character recognition to detect the license plate numbers [1].

For the second paper it talks about toll booth robot that was developed in India that can read the car's license plate number by converting it to ASCII characters then check the database for the owner information and automatically deduct the toll cost and notifying him by a message to the phone number registered in the database which finally going to open the toll gate and the car through [2].

2 Design

2.1 Requirements Constraints, and Considerations

Requirements:

- Raspberry Pi 4
- Arduino UNO
- Pi Camera
- Line Follower
- Power Bank
- Male to Female Wires
- Microhdmi to hdmi

For this project I made the assumption that the robot will take around 5 seconds to go from one license plate to another.

2.2 Design Process

2.2.1 Physical Connections

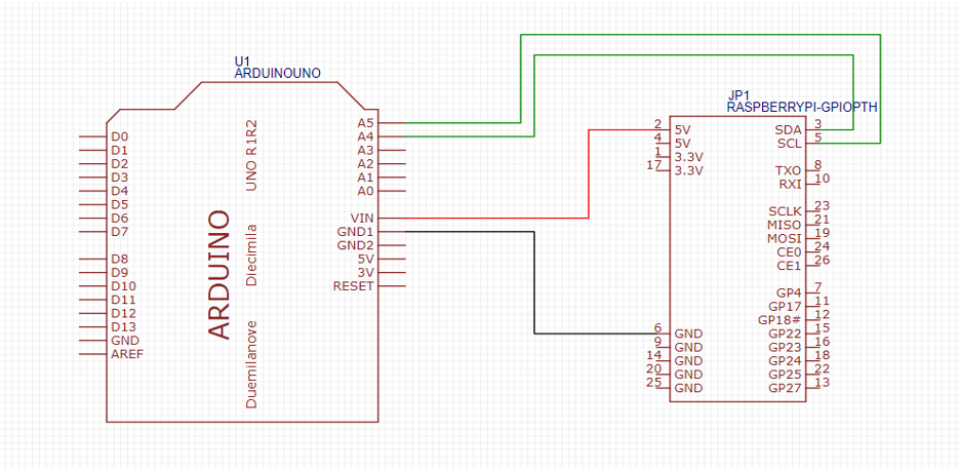

Figure 1: Wire Connections Required to Establish I2C Connection Between Raspberry Pi and Arduino UNO

In order to establish I2C communication between the Arduino UNO and the Raspberry Pi you will need to connect the Analog pin A4 from the Arduino UNO to the SCL pin of the Raspberry Pi and the Analog pin A5 to the SDA pin of the Raspberry Pi. After that I need to have them share the same ground so I will connect the ground pins to each other.

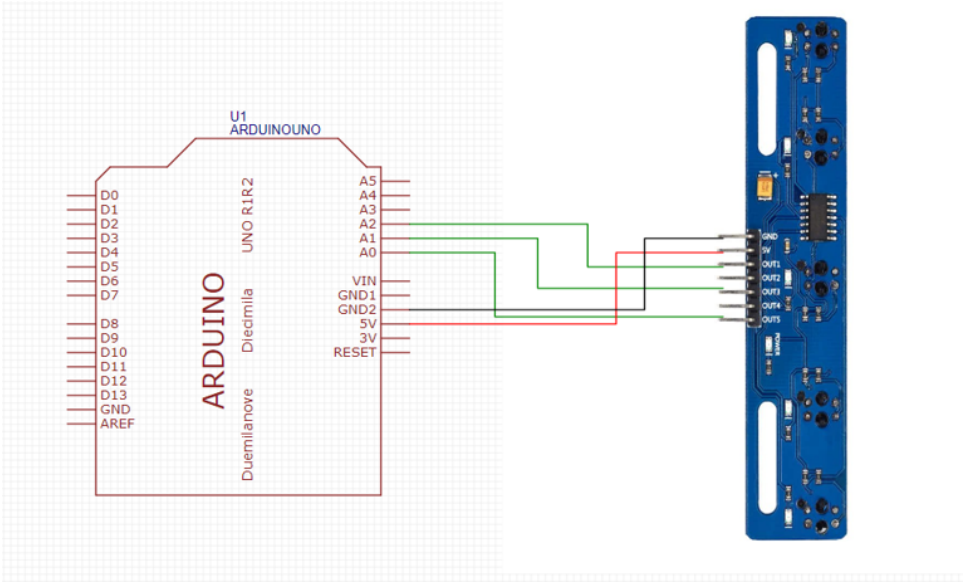

Figure 2: Wire Connections Required to Connect the Line Follower with Arduino UNO

As shown in the figure 2 you can see that I used three of the fiver available sensors of the line follower and that is because they require to be connected to an Analog pin and the Ardunio UNO has only 6 Analog pings and we need 2 of them for the I2C communication. We need to power the Line follower using the 5V pin and ground it using the ground pin then connect OUT1 with A2 and OUT3 with A1 and finally, OUT5 with A0.

Figure 3: Image of a Pi Camera Connected to Raspberry Pi 3

It is really simple to connect the Pi Camera to the Raspberry Pi since all you have to do is pull the black lid then insert the Pi Camera strip with golden side facing away from the Ethernet Plug.

2.2.2 Software Configuration

You need to configure the Raspberry Pi in order to make the Pi Camera Function and the I2C Communication to work.

In order to enter the configuration screen you need to enter the following command in the terminal:

```
1 sudo raspi-config
```

After running that command you will need to enable the following interfaces:

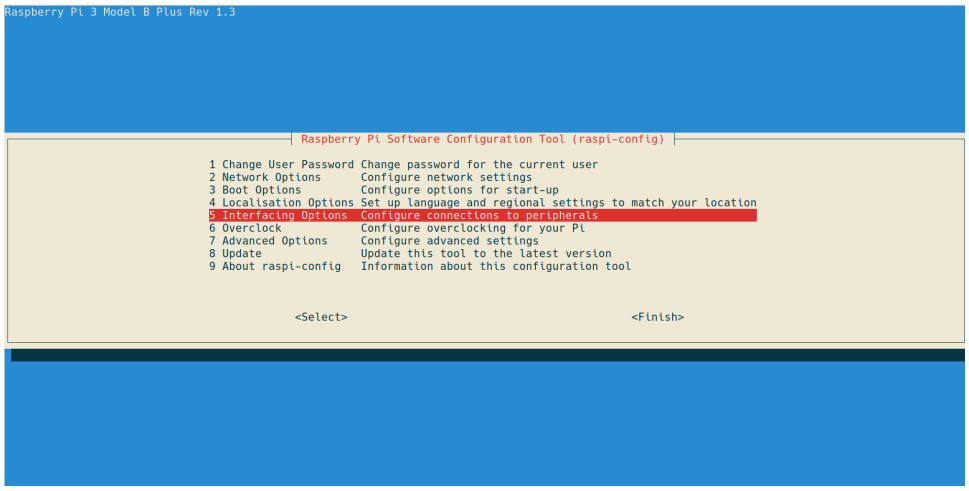

Figure 4: Main Configuration Screen of Raspberry Pi

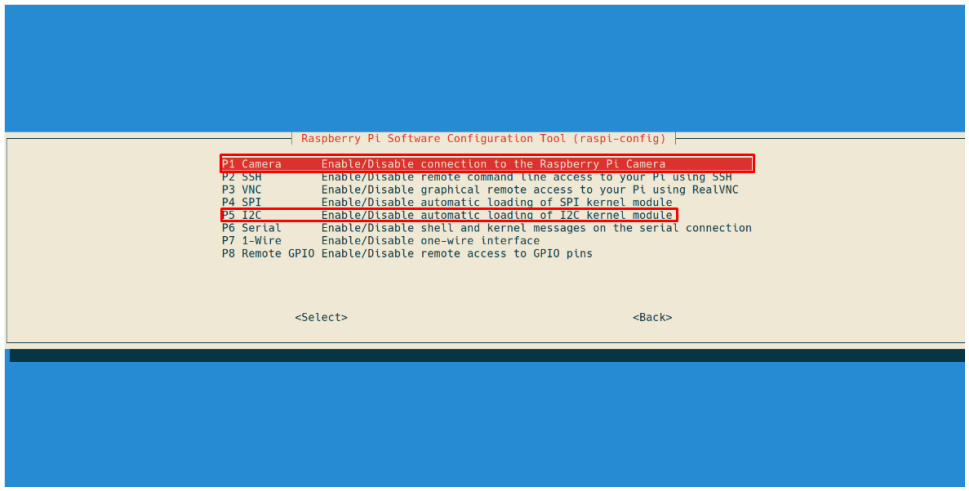

Figure 5: Interface Configuration Screen of Raspberry Pi

After enabling both the Pi Camera and I2C Communication as shown in Figure 5 you will need to run the following command to reboot the Raspberry Pi and make these changes take effect.

```
reboot
```

2.2.3 Code

Raspberry Pi's Code:

```
import cv2
import pytesseract
import imutils
from picamera.array import PiRGBArray
from picamera import PiCamera
from time import sleep
from smbus import SMBus

addr = 0x8
bus = SMBus(1)
sleep(1)

camera = PiCamera()
camera.resolution=(640,480)
rawCapture= PiRGBArray(camera, size=(640, 480))
for frame in camera.capture_continuous(rawCapture, format="bgr",
    use_video_port=True):
    image = frame.array
    cv2.imshow("frame", image)
    key = cv2.waitKey(1)&0xff
    sleep(5)
    bus.write_byte(addr, 0x73)
    rawCapture.truncate(0)
    image = imutils.resize(image, width=900)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.GaussianBlur(gray, (3,3), 0)
    text = pytesseract.image_to_string(thresh, lang='eng',
        config='--oem 2 --psm 6')
    words = text.split(" ")
    for item in words:
    if(item.isnumeric()):
        if(len(item) == 2):
            TwoDigitNumber = item
        elif (len(item) == 5):
            FiveDigitNumber = item
    print(TwoDigitNumber, FiveDigitNumber)
    cv2.imshow("frame", image)
```

First we set the address for I2C Communication as 8 and initialise the bus and create a delay of 1 second for the initialisation of the bus then I initialise the camera and set the resolution to 640x480 (WidthxHeight). After that we capture the camera in an array and loop through it saving the frame's array in a variable called image. Then we show the image we captured in a window called frame and give it a delay of 5 seconds since we assume thats the amount it will take to reach from one car to another. After 5 seconds have passed I will send a byte of the decimal number 73 which represents the char 's' and now for processing the image to increase its accuracy I first resize the image to higher width and change its colors to gray since that will make it easier to read the characters in the image then add a blur and finally get the reading from using the built-in function of pytesseract with the inputs of the image, the language set to English and setting the used engine as 2 and the text as block. After that I split the text and check each word if it is numeric and if its length is 2 or 5 then I save it in a variable and finally display it along with the frame.

Arduino UNO's Code:

```
#include <Wire.h>

//Sensor Connection
  const int left_sensor = A2;
  const int middle_sensor = A1;
  const int most_right_sensor = A0;
  int most_left_sensor_state;
  int middle_sensor_state;
  int most_right_sensor_state;

void setup() {
  Wire.begin(0x8);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
}

void receiveEvent(int howMany){
  while(Wire.available()) {
    char c = Wire.read();
    if( c == 's'){
```

```
      Serial.println("Stopping Vehicle for 5 seconds");
      delay(5000);
      Serial.println("Starting Vehicle");
    }
  }
}

void loop() {
  most_left_sensor_state = analogRead(sensor1);
  middle_sensor_state = analogRead(sensor3);
  most_right_sensor_state = analogRead(sensor5);
  if(most_left_sensor_state < 500 && most_right_sensor_state > 500){
    Serial.println("Need to go left");

  } else if ( most_left_sensor_state > 500 &&
      most_right_sensor_state > 500 && middle_sensor_state < 500){
    Serial.println("Going on the right track");

  } else if (most_right_sensor_state < 500 &&
      most_left_sensor_state > 500){
     Serial.println("Need to go right");
  }
}
```

First I point out the pins used and set the address for I2C communication as 8 and make it when message received through I2C it will trigger receiveEvent function. The receiveEvent will loop through the message recieved and covert them to a char which if they equal s then it will stop the vehicle for 5 seconds and start it again. For the loop I analogRead the sensors from the line follower and the number returned will be from 0 to 1000 with 0 representing that there is a black line under it while 1000 shows that there is no black line under it. If the most left sensor has black line under it and the most right doesn't then it will need to go to the left to correct its path and same for the opposite. If the middle sensor has black line under it while other two doesn't then it means its at the correct path and will not change its trajectory.

3 Experimental Testing and Results

3.1 Results

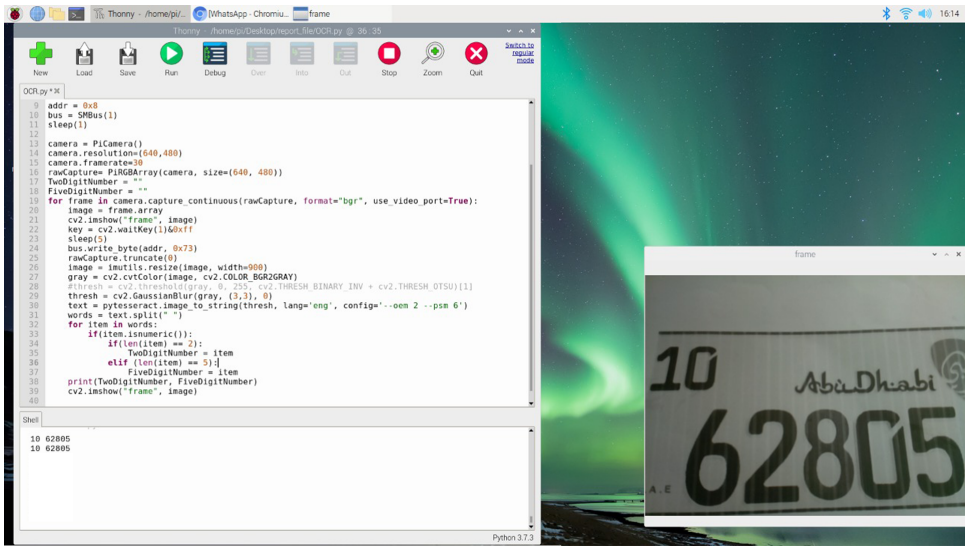

Figure 6: Example of License Plate Reading Using Raspberry Pi and Pi Camera

In the Figure 6 you can see that Raspberry Pi was able to detect both the Category Number and the five digit number from the License plate image printed on an A4 Paper.

3.2 Results

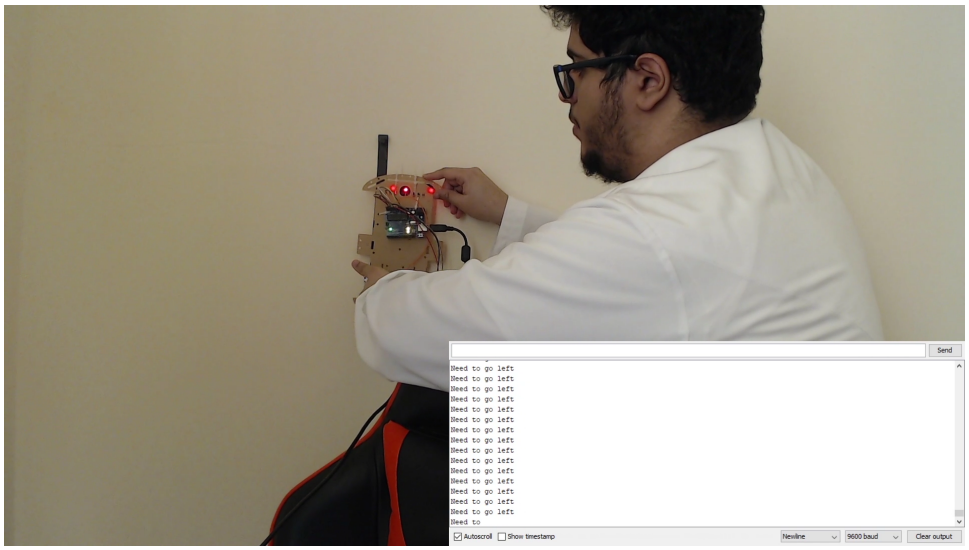

Figure 7: Arduino UNO Outputting "Need to go left"

In the Figure 7 you can see that Arduino UNO is printing "Need to go left" since the most left sensor is only one with black line under it.

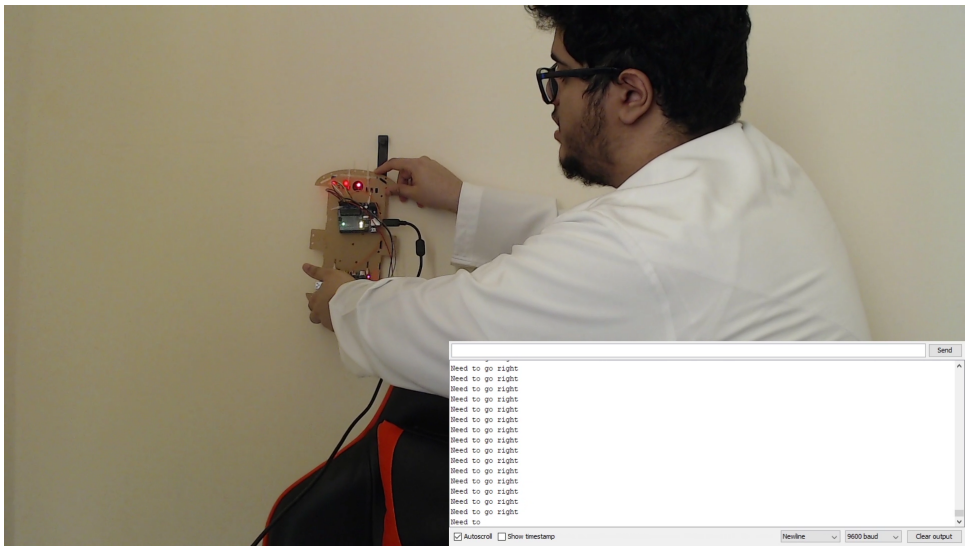

Figure 8: Arduino UNO Outputting "Need to go right"

In the Figure 8 you can see that Arduino UNO is printing "Need to go right" since the right sensor is only one with black line under it.

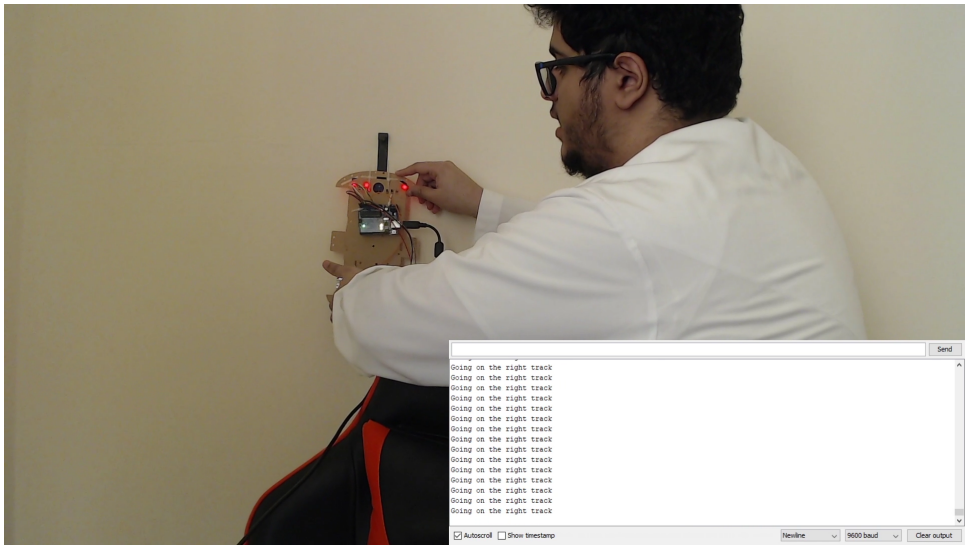

Figure 9: Arduino UNO Outputting "Going on the right track"

In the Figure 9 you can see that Arduino UNO is printing "Going on the right track" since the middle sensor is only one with black line under it.

4 Conclusion

4.1 Summary

In Summery I was able to detect the license plate numbers and communicate with Arduino UNO to control the vehicle and check if the vehicle moving on the correct path using a black line on the ground and line follower sensor.

4.2 Future Improvements and Takeaways

It is possible to improve this Embedded System by making it automatically detect license plates instead of assuming that the cars are 5 seconds apart.

4.3 Impact Statement

We hope that by implementing this project, we will be able to protect Mawaqif's employees who work out during the day from becoming ill, as the robot will be able to take over during the day when the heat is at its peak, and these Mawaqif employees will be able to take over at night when the heat is at its lowest.

References

[1] J. Raju, C. V. Raghu, S. N. George and T. S. Bindiya, "Development of a Hand held device for Automatic License Plate Recognition," in The Institute of Electrical and Electronics Engineers, Inc. (IEEE) Conference Proceedings, Piscataway, 2020.

[2] M. M. Desai and J. J. Patoliya, "Smart toll collection system using embedded Linux environment," in The Institute of Electrical and Electronics Engineers, Inc. (IEEE) Conference Proceedings, Piscataway, 2017.