Graduation Project Report

Vision-based Control of a Mobile 5 DOF Robot Arm for Pick and Place

Under the supervision of:
Dr. Mohamed Sallam

Presented by:
Hani Atef Abdallah Mohamed

July 2021

Jury

Acknowledgment

I am indebted to many people who have influenced me in this project. Their help and support led to the completion of this research paper.

I would like to express my particular gratitude and deep appreciation to Dr. Mohamed Sallam for helping and guiding me in my research paper.

I also would like to thank all the Professors and Doctors for all their effort and hard work through all the years.

Finally, I want to thank my family and all my friends for their support through all my years of study.

Hani Atef Abdallah Mohamed, July 2021

Abstract

Adding vision system to mobile robots allowed them to be autonomously used in several industrial and domestic applications. Robots with vision system are more like humans where they can recognize the persons and environment surrounding them and take the suitable action when needed. They can also navigate from one place to the other avoiding the different types of obstacles.

In this thesis, it is developed a mobile robot equipped with a vision system that can navigate in supermarkets, collect all the required products from the shelves and bring them back to a pick-up point. It consists of three main parts: **firstly** a carriage that move around and carry all the other robot components, **secondly** a 5DOF robot arm with a gripper that can pick the products from the shelves and place them in the basket, and **thirdly** a vision system that allows the robot to recognize the product and localize itself.

The experiments showed that the developed robot can give a satisfactory performance and could achieve the pre-programed task.

The main objectives of this project can be summarized as following:

- To modify an old robot arm and make it smarter.
- To make a moving robot arm.
- To allow the manipulator to work with objects of different orientations, whose position is not known beforehand.
- To be able to identify different objects and identify their location to decide a path for the robot arm to reach.
- To make this project fully autonomous

Table of Contents

List of Figures

List of Tables

Nomenclature

S1:	Switch number 1
S2:	Switch number 2
S3:	Switch number 3
S4:	Switch number 4
PWM:	Pulse Width Modulation
Vout:	Output Voltage
Vmax:	Max Voltage
D:	Distance
t:	time
C:	Speed of Sound
IR:	Infrared
UDP:	User Datagram Protocol

CHAPTER 1 INTRODUCTION

1.1 Background

The pick and place robot arm is a robot that can be programmed to pick an object from a certain location and place it at the desired location. These robots help to free up humans from repetitive tasks. The robot arm count on computer vision to pick and place different objects.

1.2 Literature Review

In 2016, Teodorescu et al. built a robot arm that uses a vision system to identify round objects that are randomly scattered on a table. The robot arm then uses the gripper to pick it and place it inside a basket as shown in Figure 1.

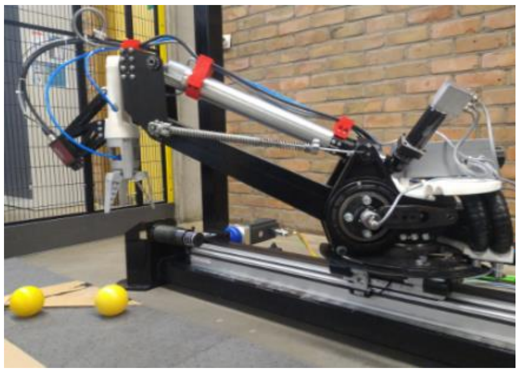


Figure 1: The 4-DOF robot, able to execute fast pick-and-place of round objects.

In 2017, Borkar & Andurkar designed a pick and place robot so that the user can fill the liquid in a bottle according to the volume occupied in the bottle, and the robot can pick and place using some mechanical equipment such as the gripper and robotic arm after the bottle is filled as shown in Figure 2.

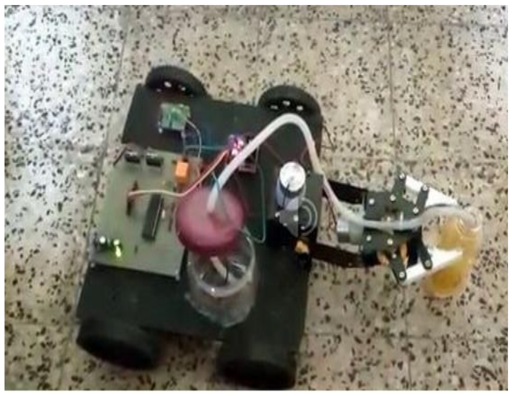


Figure 2: Photograph of Robot from top view

Younus et al. (2019) designed a Line following Robot that uses an infrared sensor (Figure 3) to follow and run over a predefined path. It will move in the direction specified by the user to navigate the robot through a black line drawn on a white surface.

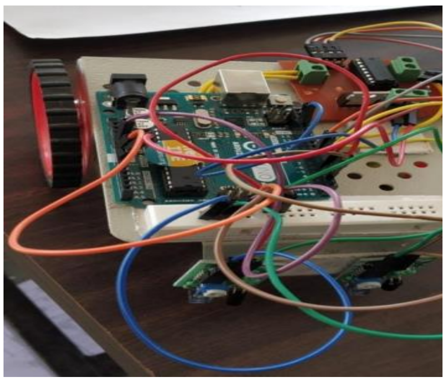


Figure 3: Line following robot.

Abdel Aziz (2020) designed and controlled a pick and place prototype with a gripper for the end effector. The robot arm can move in all directions (x,y,z) and can rotate around a vertical axis. His objective was to model, develop and control a 5 D.O.F pick and place robot prototype that has a gripper as an end effector as shown in Figure 4.

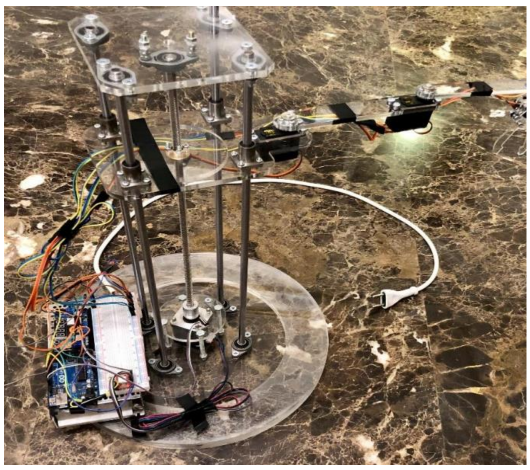


Figure 4: Scara robot

1.3 Thesis objectives

The main goal of this project is to model and automate a Vision-based robot manipulator for a grocery store.

The camera placed on top of the robot arm should be able to track, identify and decode any bar code, and comparing the decoded bar code with a list of given bar codes.

The camera and the robot arm are placed on top of a vehicle that will go to some specific location to grab the correct item or items from the grocery store.

1.4 Thesis organization

Chapter 1: This chapter will introduce the meaning and the functionality of a robot arm using computer vision to pick and place, a summary of previous research on this topic, and the objectives of the research.

Chapter 2: This chapter will explain how the vehicle and the robot arm are working, the concept of the vision system, and how all the subsystems are interconnected.

Chapter 3: This chapter will explain the functionality and the role of every subsystem, we will start by discussing in detail how the vehicle, robot arm, and the vision system work, then we will talk about the electrical system needed to build this project and finally we will explain the algorithm behind controlling the vehicle and the robot arm.

Chapter 4: This chapter will list a sample of raw data and observe the results of the vision system.

Chapter 5: This chapter will discuss the results to know if we achieved our goal or not, and then compare it with some previous work.

Chapter 6: This chapter will summarize all the ideas behind the building and controlling a mobile robot arm using computer vision, and the future work to improve this project.

CHAPTER 2 THEORETICAL APPROACH

This project consists of a pick and place robot in a movable system. The system is divided into multiple subsystems as following:

The first subsystem will be responsible for moving the whole system which is the vehicle (car) following a line going thru all the grocery stores. This subsystem includes 4 dc motors, an IR sensor to keep track of the position and the direction of the line, and an ultrasonic sensor to avoid any collision, controlled using an Arduino.

The second subsystem will direct the vehicle using a camera connected to a microcomputer to give commands to the first subsystem to go into the right path (section) since the grocery store is composed of multiple sections, every section has some category of items.

The third subsystem will be responsible for the pick and place robot arm, it consists of a second Raspberry pi camera that will start the scanning process. The Raspberry pi camera will be decoding and comparing all the scanned bar codes with the item list that it has, if the item is correct which means that it’s the item that we need, the Raspberry pi will give an order to the Arduino using serial communication to stop the first subsystem, then the robot arm will pick the item and then place it in a given position which will be our cart.

Finally, the robot arm will go to its initial position waiting for the next item and the first subsystem will start moving again, this process will be repeated until all the items are found.

CHAPTER 3 EXPERIMENTAL APPROACH

In this chapter, we will represent how we implemented each part of the system, the system is divided into 4 parts: robot arm, vehicle, vision system, and control system.

3.1 General description

As we can see in Figure 5, the mobile robot arm system consists of multiple subsystems:

The **first subsystem** is the vehicle which is colored in blue as shown in figure 5, which consists of a steel box that holds all the electrical system and the robot arm.

The **second subsystem** (colored in green) is responsible for guiding the vehicle to the right path using a vision system (Figure 5).

The **third subsystem** (circled in red) is the robot arm that consists of 4 servo motors to move all the links, a stepper motor to move the arm up and down, and the vision system (Figure 5).

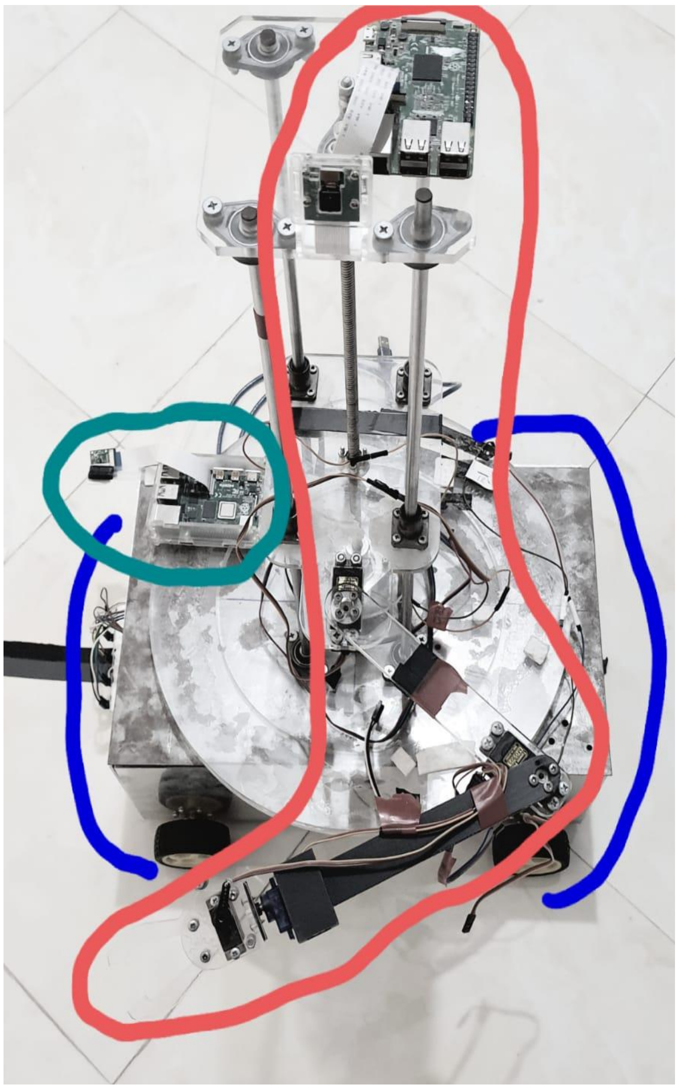


Figure 5: Overview of the mobile robot arm

3.2 Vehicle

Picking up a dc motor to drive the vehicle will depend on the weight of the robot arm, so, using a 12V dc motor (Figure 6) will be able to move the system at an acceptable speed.

The speed of the vehicle will be adjusted according to the camera mounted on top of the robot arm.

The vehicle will follow a line with the help of 5 infrared sensors as said earlier.

The dimensions of the vehicle are 400 * 250 * 150 mm.

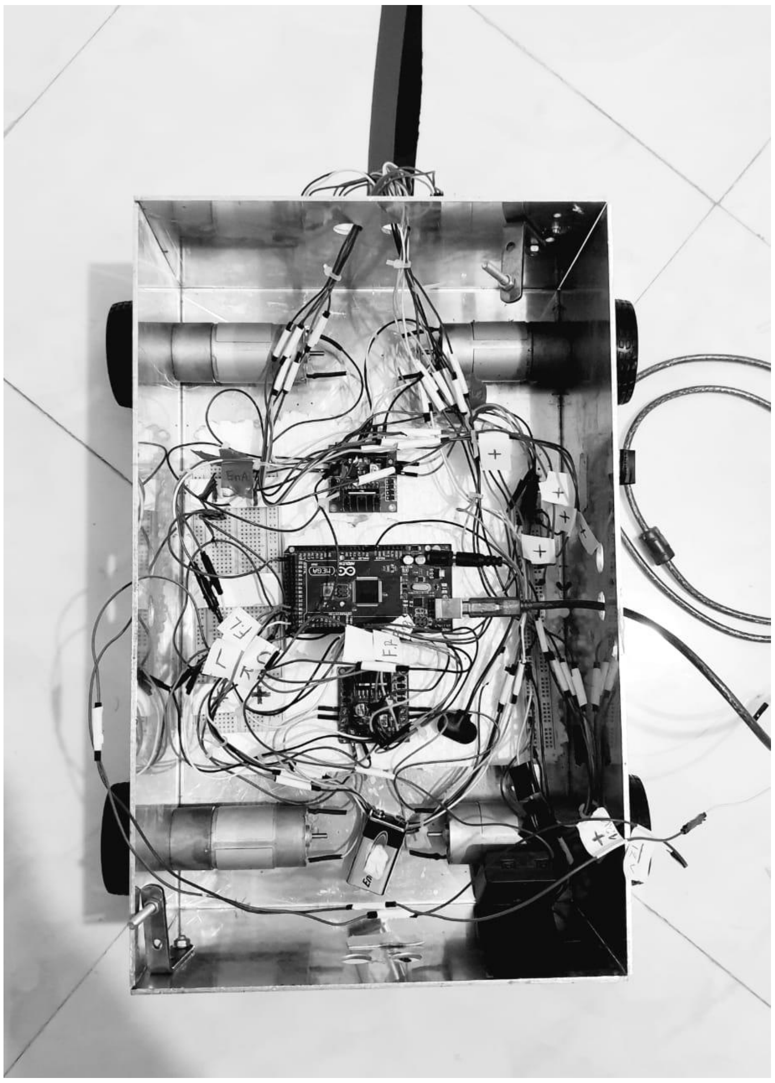


Figure 6: Top view of the vehicle

3.2.1 Electrical system of the vehicle

Table 1: Electrical components of the vehicle

List	**Description**
4x Dc motors	Used to move the vehicle
2x L298N driver	Used to control the speed of the motors
5x Infrared sensor	Used to track a black line so the vehicle can follow the line
4x Ultrasonic sensor	Used to measure the distance between the vehicle and any obstacle
Arduino mega	Microcontroller used to control all the system

3.2.1.1 Dc motor

A Dc motor stands for direct current motor which transfers electrical energy into mechanical energy.

The DC motor (Figure 7) uses direct current to convert electrical energy into mechanical rotation.

The DC motor powers the fixed rotor on the output shaft using the magnetic field generated by the generated current.

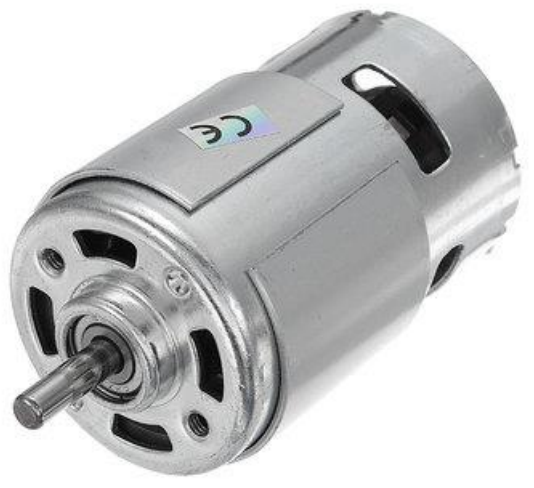


Figure 7: Dc motor

Technical Specifications:

- Rated Voltage: 24V
- No Load Speed: 8300 RPM
- Rated Torque: 4.0kg-cm/4000RPM
- Motor Size: Diameter 42mm, Length: 77mm
- Shaft: Diameter 5mm, Length: 16mm

3.2.1.2 L298N driver

An H bridge is a simple circuit that lets us control a motor bi-directionally. It consists of 4 switches as shown in Figure 8.

H-bridge is a series of transistors (4 transistors) that will allow current to be routed depending on several pins being High or Low through the motor in either direction.

If we close some switches together, we will be able to control the direction of the motor. There is a lot of ways to close switches but there is just a couple of useful ways.

Table 2: State of the switches

Closed Switches	Motor Direction
S1 and S4	Forward
S2 and S3	Reverse

In this project, a dual H-bridge (Figure 9) will be used, which works with the same methodology.

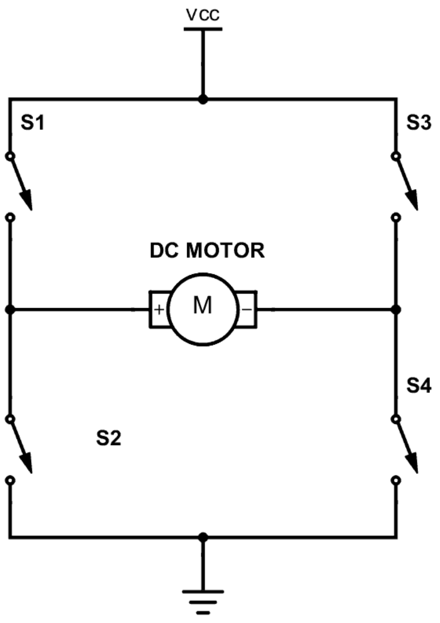


Figure 8: H-Bridge circuit

Technical Specifications:

- Double H bridge Drive chip: L298N
- Logical Voltage: 5V
- Drive Voltage: 5V - 35V
- Logical current: 0-36mA
- Driver Current: 2A (Max single bridge)
- Max power: 25W
- Dimension: 43 * 43 * 26 mm
- Weight: 26 g

Figure 9: Dual H-Bridge Driver

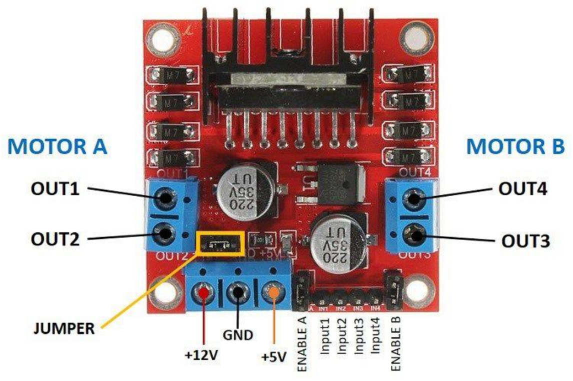


Figure 10: Dual H-Bridge driver pins

To control the speed of the 2 motors, we need to connect the Enable A and Enable B pins (Figure 10) to the Arduino to a PWM (Pulse width modulation) capable digital outputs.

The pulse width modulation signal is a digital rectangular wave.

PWM is a technique to lower the power provided by an electrical signal, effectively cutting it up into discrete parts.

PWM controls the speed of a motor by giving a series of ON and OFF pulses and changing the duty cycle.

The duty cycle describes the time when the signal is high in our wave, so the output voltage will be equal to the duty cycle multiplied by the max voltage (Figure 11).

Vout = Vmax * DT

The duty cycle ranges between 0% (output voltage is equal to 0) and 100% (output voltage is equal to the max voltage).

In Arduino, the duty cycle of 0% to 100% will correspond to a value of 0 to 255.

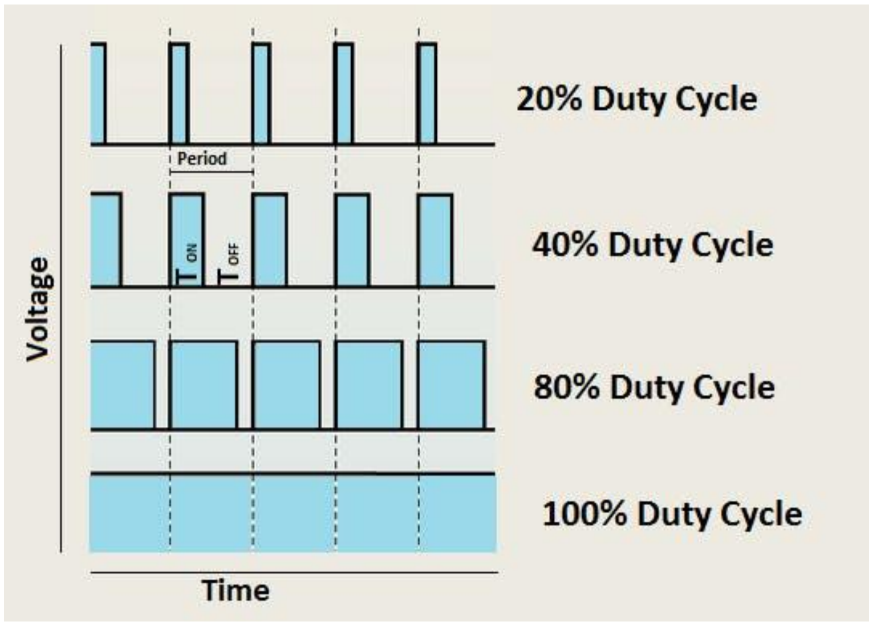


Figure 11: Duty cycle

3.2.1.3 Infrared sensor

Infrared sensors (Figure 12) are used to detect the position of the vehicle to be able to follow a line. IR sensor is widely used for the line following robot.

The sensor emits infrared light on a surface if the IR light is reflected by the surface and the sensor picks it up then it's a light-colored surface if the IR light is not reflected then it's a dark surface. This allows the sensor to detect a black line on a light-colored surface.

Figure 12: Infrared sensor

In this project, we will use 5 IR sensors to control the vehicle.

There are multiple ways to use the IR sensor to drive the vehicle, saying that we will use 2 methods and compare them.

The first method is the classic control algorithm which consists of multiple conditions, depending on the output of the IR sensor.

The second method is using a PID algorithm to control the speed and directions of the motors.

PID stands for Proportional, Integral, and Derivative.

The Proportional value for the line follower robot will be proportional to the position of the vehicle with respect to the line, which is the error (how far the vehicle is from the line), if the vehicle is exactly on the line the proportional value will be 0. The error will be multiplied by the proportional constant gain Kp to adjust the position of the vehicle (Figure 13).

The Integral is the area under the curve meaning that the integral value is the sum of the proportional values (errors) over the time from the start of the vehicle run. The integral gain Ki will be multiplied by the accumulated errors (Figure 13).

The Derivative is the rate of change of the proportional value. The derivative value will be the difference between the error and the previous error multiplied by the derivative gain Kd (Figure 13).

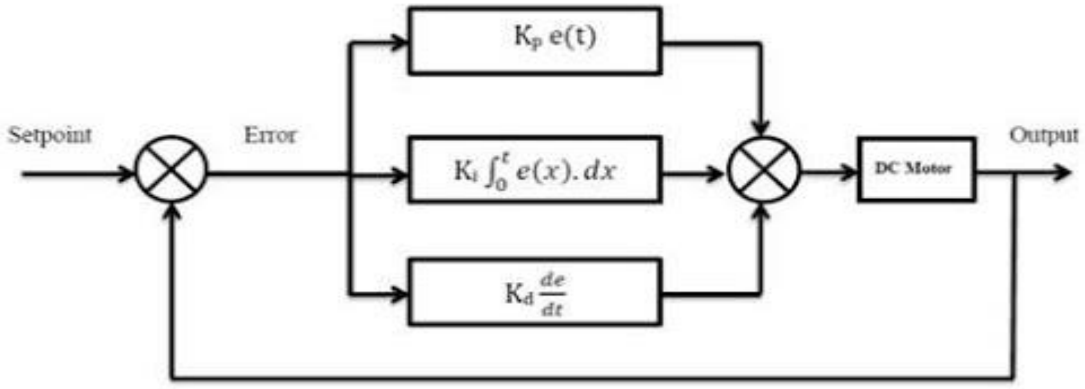


Figure 13: PID controller

Trial and error is the most used technique to get the PID constants(Kp, Ki, and Kd), so we have to tune them until we see the best result.

3.2.1.4 Ultrasonic sensor

An ultrasonic sensor (Figure 15) measures the distance between the object and the sensor using ultrasonic sound waves.

The ultrasonic sensor consists of 2 main components, a transmitter that emits the sound using piezoelectric crystals and a receiver that encounters the sound after it has travelled to and from the object (Figure 14).

In our case, we will use the ultrasonic sensor to protect the movable system from any collision.

So, to Calculate the distance between the sensor and the object, the ultrasonic sensor measures the time it takes the sound to travel from the transmitter back to the receiver.

$D = 0.5\, t * C$

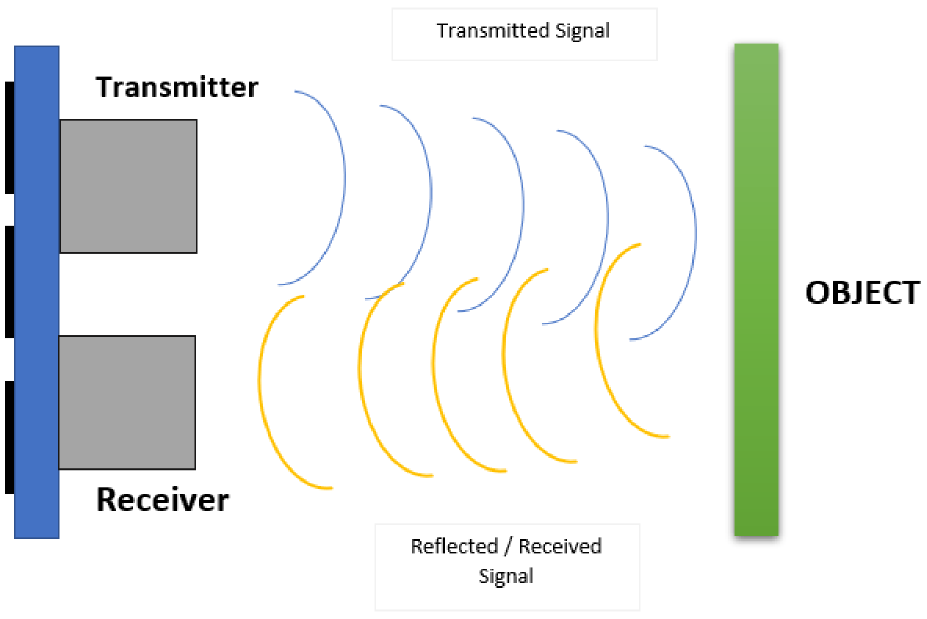


Figure 14: Transmitter/Receiver ultrasonic sensor

Technical Specifications:

- Operating Voltage: 5V
- Operating Current: 15 mA
- Operating Frequency: 40 kHz
- Farthest Range: 4 m
- Nearest Range: 2 cm
- Measuring Angle: 15 Degree
- Input Trigger Signal: 10 us TTL pulse
- Output Echo Signal: Output TTL level signal
- Dimension: 45 * 20 * 15 mm

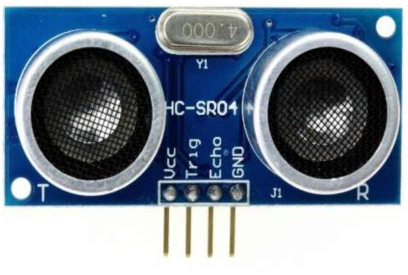


Figure 15: Ultrasonic sensor

3.2.1.5 Arduino Mega

Arduino Mega (Figure 16) is a programmable microcontroller based on the ATmega2560 used widely mainly because it's inexpensive and its simplicity in usage.

It has 54 digital pins(input/output) of which 15 PWM pins, and 16 analog input.

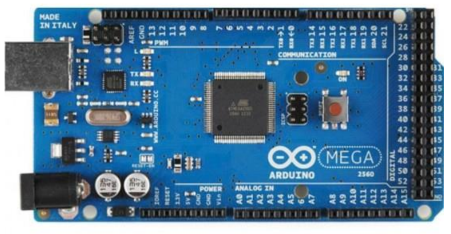


Figure 16: Arduino mega

3.2.2 Wiring

The wiring of every electrical component will be shown independently to facilitate the understanding of the cabling.

3.2.2.1 L298N driver with the dc motor

Dual H bridge driver (L298n) with 2 dc motors wirings, the same wiring will be for the 2 other dc motors (Figure 17).

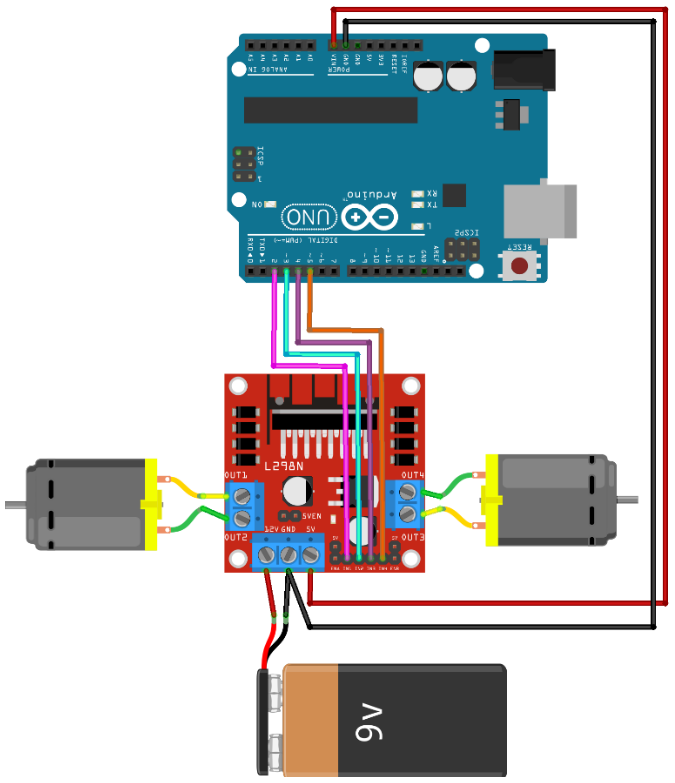


Figure 17: Wiring of 2 dc motors

3.2.2.2 IR sensor

The same wiring will be for the 5 infrared sensors (Figure 18).

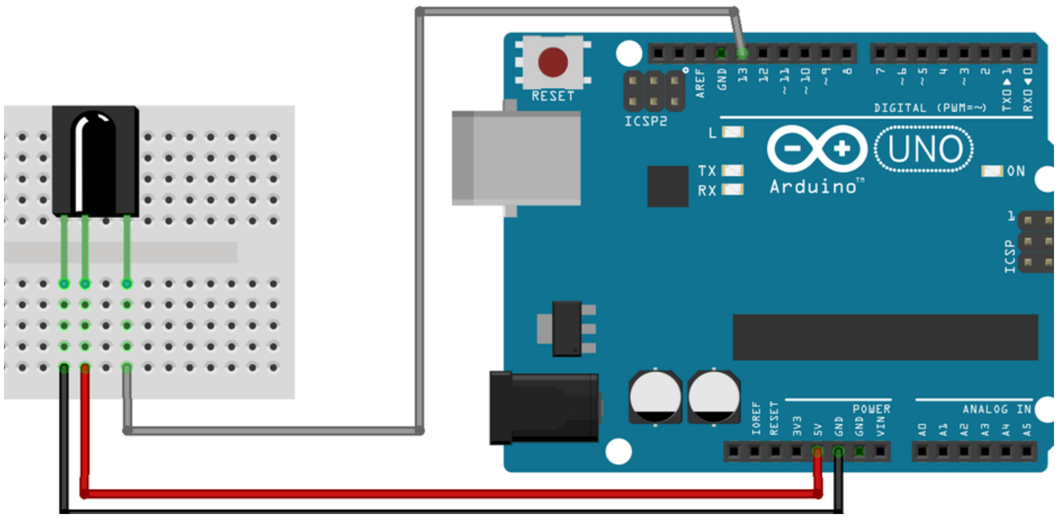


Figure 18: Wiring of IR sensor

3.2.2.3 Ultrasonic sensor

The same wiring will be for all the ultrasonic sensors (Figure 19).

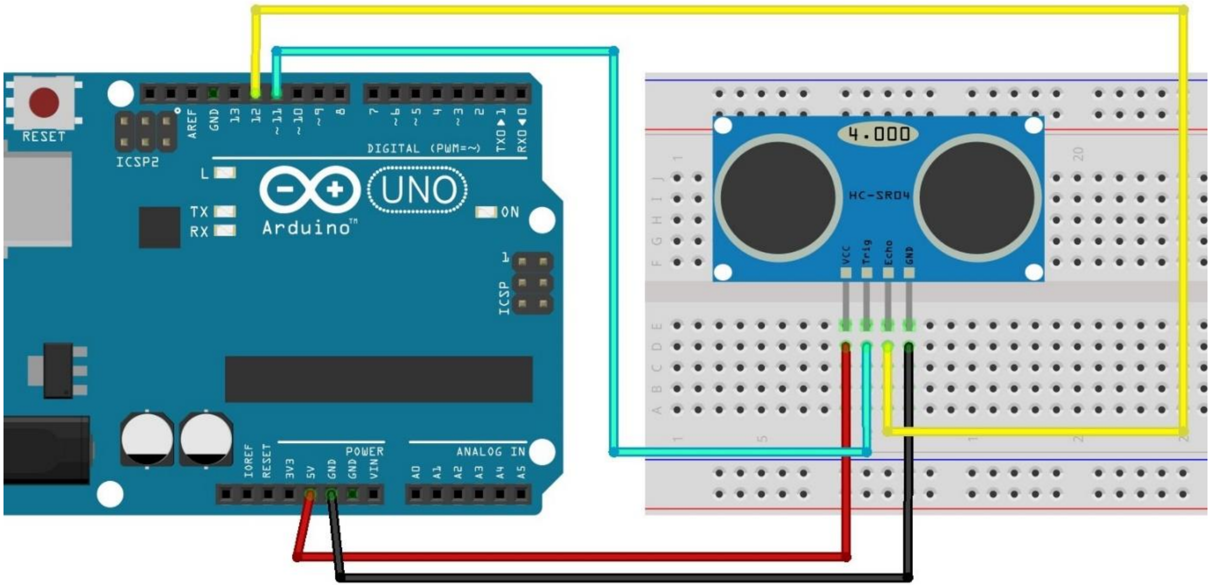


Figure 19: Wiring of ultrasonic sensor

3.3 Vision system

In this project, we will use the raspberry pi with a camera to control the robot arm. A Raspberry Pi is a credit card-sized microcomputer that can be plugged into a monitor and uses a mouse and a keyboard. A Raspberry Pi is used as a normal computer, it can be programmed to do some tasks and it has general-purpose input/output pins and ports on which we can connect multiple things (Figure 20).

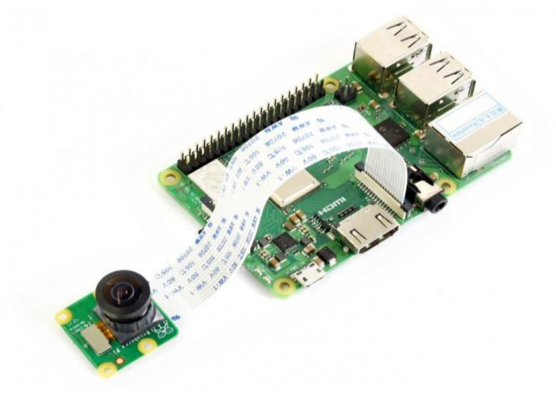


Figure 20: Raspberry pi with a camera

The Camera attached with the raspberry pi will be the eyes and the brain of the robot arm, they will be processing a video means performing operations on video frame by frame. A frame is nothing more than a specific instance of the video at a single moment. Even in a second, we can have multiple frames. Frames can be considered like images.

Therefore, any operation that we can perform on the image can also be performed on the frame.

Image processing is the analysis and processing of digital images.

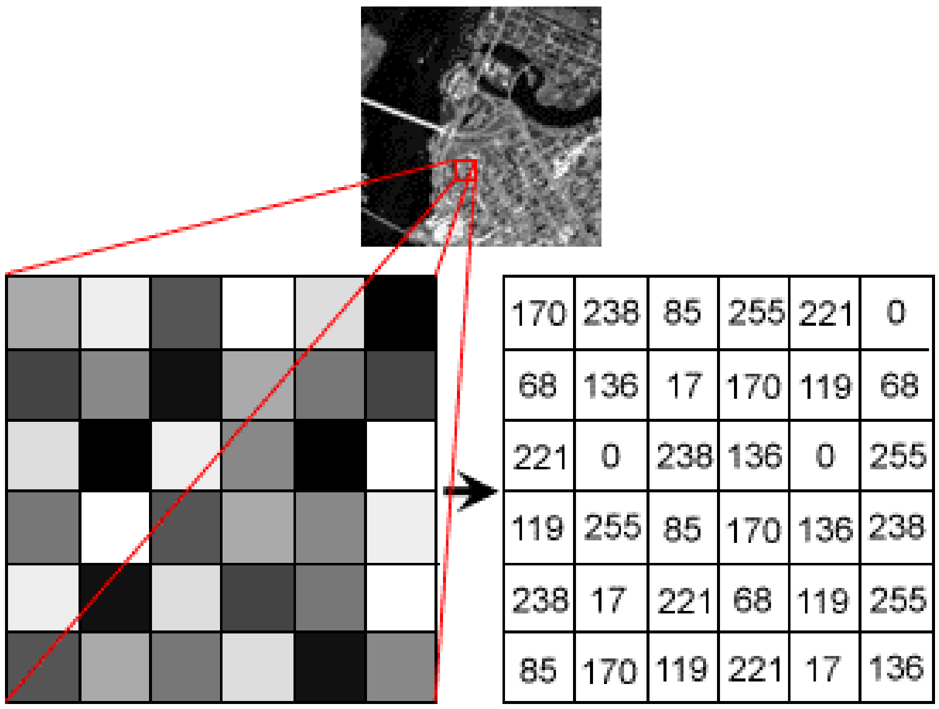
An image can be defined as a two-dimensional function f(x,y)(two-dimensional matrix), where x and y are the plane coordinates.

An image is made up of pixels. Every pixel contains a value depending on the color of the image.

For a binary image, each pixel has a value of 1 or 0 which is black or white.

For a grayscale image as shown in Figure 21, each pixel represents the amount of light or the brightness that it has (shades of gray), it ranges from 0 to 255, 0 is taken to be black and 255 for white.

In colored images, every pixel is represented with 3 values for the 3 color channels red, green, and blue (RGB) each value ranging from0 to 255, combined to give a color for the pixel.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Figure 21: Gray Scale image

2 types of cameras can be used in this project, a normal webcam which will be connected to the raspberry pi through a USB port, and the raspberry pi camera module which will be connected throughout the Camera port in the raspberry pi.

3.4 Robot arm

We used the project of another student who made a robot that performs the required function in the system, but with different axes of motion, we modified the mechanical design to be able to work in our system. Therefore, we will not go into more details on the design of the robot arm, but we will show the design before modifying it (as shown in Figures 22 and 24), and after the modification as shown in Figures 22 and 25. (For more details about the mechanical design, the research paper of this student will be in the last reference).

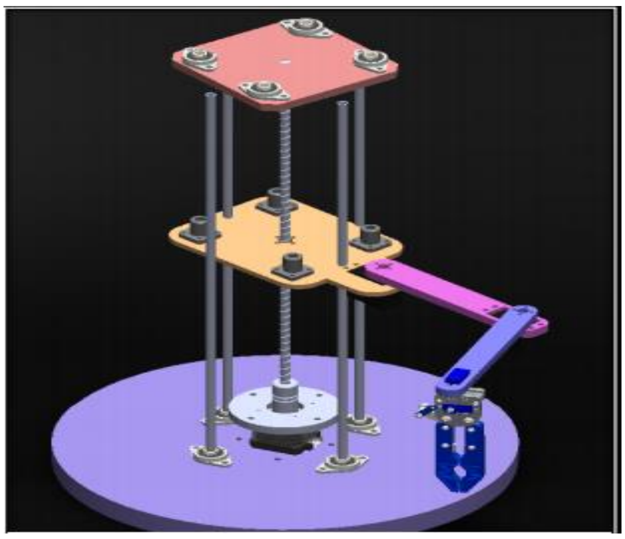


Figure 23: Robot arm before modifying the link

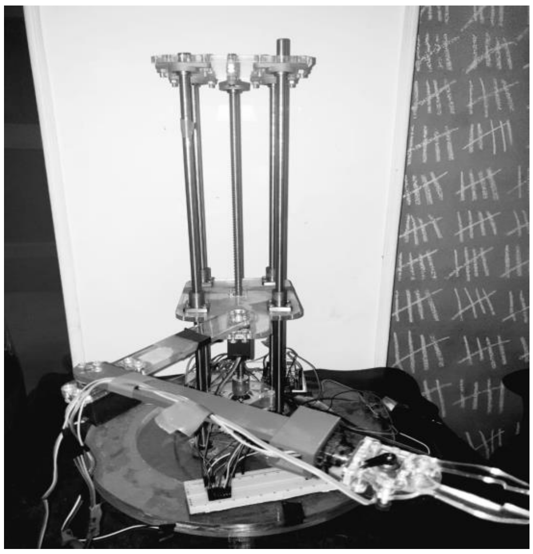


Figure 22: Robot arm after modifying the link

3.4.1 Link before and after

The newly designed link (Figure 25) will change the axe of motion of the gripper. This link is connected to a joint, the joints are the movable components of the robot that cause a relative motion between the links. The figures 24 and 25 show the difference between the old design of the link and the new one.

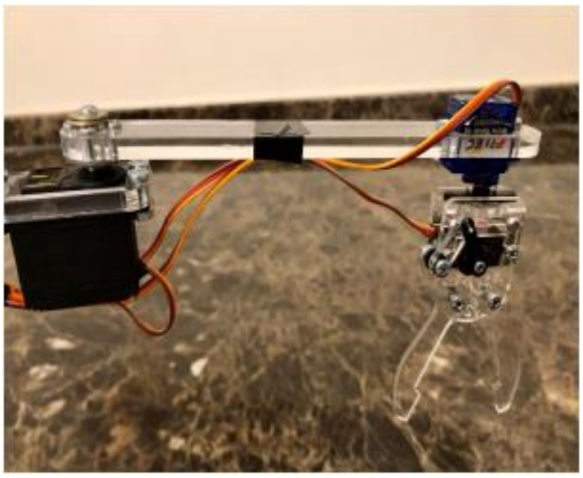


Figure 24: link before the changes

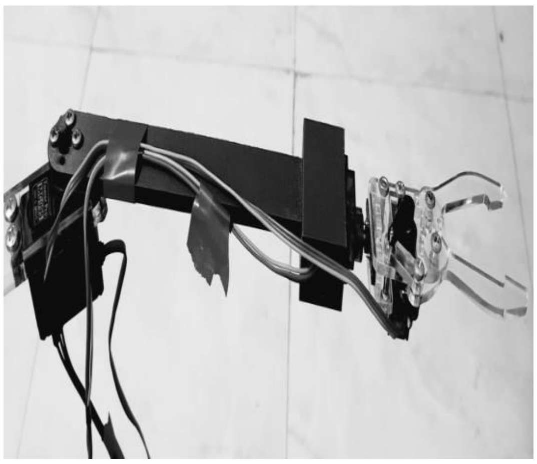


Figure 25: New design of the link

3.4.2 Link dimensions

a- Left side view of the link (Figure 26).

b- Top view of the link (Figure 26).

c- Right side view of the link (Figure 26).

d- Front view of the link (Figure 26).

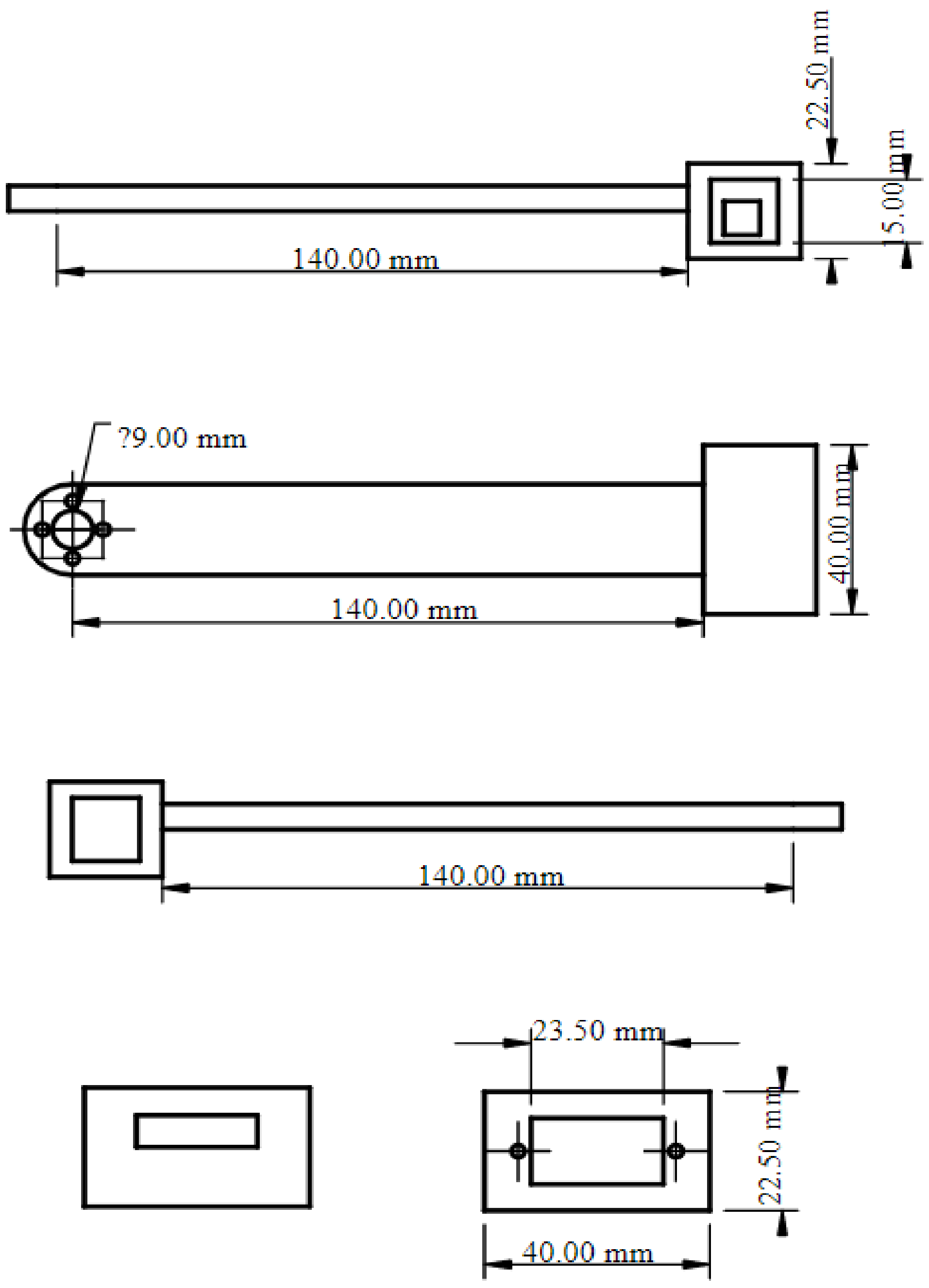


Figure 26: Dimensions of the new link

Overview of the link as shown in Figure 27

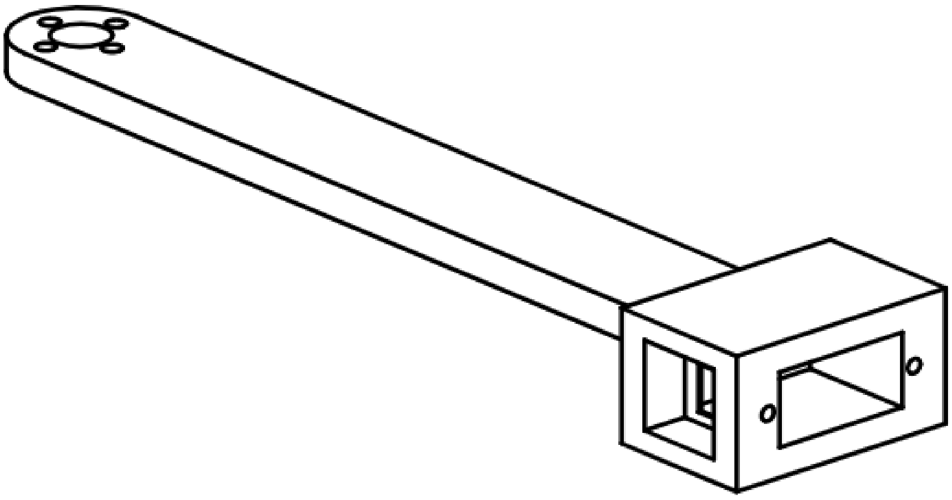


Figure 27: Overview of the link

The robot arm works with 5 motors: a Stepper Motor to move the arm in Z axe as shown in Figure 31, 2 Servo MG995 motor to move the arm in X and Y axes as shown in Figure 28, a micro servo SG90 (Figure 29) to rotate the gripper if needed, and a Micro servo motor SG90S to close and open the gripper as shown in Figure 30.

3.4.3 Electrical system for the robot arm

Table 3:Electrical components of the robot arm

List	Description
2x MG995 servo motor	Used to rotate the arm 180 degrees.
SG90 micro servo motor	Used to rotate the gripper 180 degrees.
MG90S micro servo motor	Used to open and close the gripper.
Stepper motor	Used to lift the robot arm up and down.
A4988 driver	Used to control the stepper motor

3.4.3.1 Servo motor

A servo motor is using a closed-loop mechanism that has position feedback so it can be able to control its rotational speed and position. The servo motor receives a control signal that represents the desire output position of the links.

In this project, 3 types of servo motors will be connected to the robot arm, every type has its own role in the robot arm.

3.4.3.1.1 MG995 motor

The two MG995 servo motors will be connected to the links of the robot arm to rotate them from 0 degrees to 180 (Figure 28).

Technical specifications(MG995):

- Operating voltage range: 4.8 V to 7.2 V
- Operating speed: 0.2 s/60 degrees (4.8 V), 0.16 s/60 degrees (6 V)
- Stall torque: 9.4kg/cm (4.8 V), 11kg/cm (6 V)
- Rotational degree: 180º
- Operating temperature range: 0ºC to +55ºC
- Weight: 55 g
- Dimension: 40.7 * 19.7 * 42.9mm

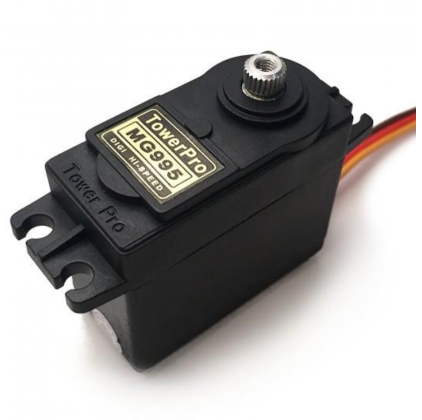


Figure 28: MG995 Servo motor

3.4.3.1.2 SG90 micro servo motor

The SG90 micro servo motor will be connected to the last link of the robot arm to rotate the gripper 180 degrees (Figure 29).

Technical specifications(Micro servo motor SG90):

- Operating voltage: 4.8 V to 6 V
- No load speed: 0.12s/60 degrees (4.8 V)
- Stall torque: 1.8 kg/cm (4.8 V)
- Rotational degree: 180º
- Operating temperature range: -30ºC to +60ºC
- Weight: 9 g
- Dimension: 22 * 12.5 * 29.5 mm

Figure 29: SG90 Micro servo motor

3.4.3.1.2 MG90S micro servo motor

The MG90S micro servo will be connected to the gripper and the gripper will be linked to the SG90 micro servo to allow us to open and close the gripper so it can pick anything (Figure 30).

Technical specifications(Micro servo MG90S):

- Operating Voltage: 4.8V to 6V
- Operating speed: is 0.1s/60° (4.8V), 0.08sec/60degree(6v)
- Stall Torque: 1.8 kg/cm (4.8V), 2.2 kg/cm (6V)
- Rotational degree: 180°
- Weight of motor: 13.4gm
- Dimension: 22.8 * 12.2 * 28.5 mm

Figure 30: MG90S micro servo

3.4.3.2 Stepper motor

A full rotation in a stepper motor (Figure 32) is divided into several steps. The stepper motor consists of 2 main parts, the rotor, and stator as shown in Figure 31.

The rotor is a rotating shaft and the stator consists of multiple coils, arranged in groups called Phase, if we energize each phase in sequence, the motor will rotate, one step at a time.

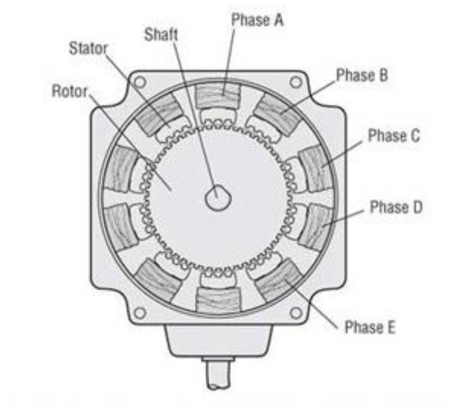


Figure 31: Stepper motor from the inside

Using a computer, we can control the steps and get very precise positioning or speed.

To calculate the height that we want to achieve, we need to know the specifications of the stepper motor, in our case, the step angle is equal to 1.8 degrees (200 steps per revolution)

Technical specifications:

- Motor type: Bipolar Stepper
- Step Angle: 1.8 degrees
- Holding Torque: 40 Ncm
- Operating Voltage: 12V
- Frame size: 42 * 42 mm
- Body Length: 40 mm
- Shaft Diameter: 5 mm
- Number of Leads: 4
- Lead Length: 400 mm
- Weight: 280g

Figure 32: Stepper motor

3.4.3.3 A4988 driver

The A4988 is a micro-stepping driver used to control stepper motors in 2 directions (Figure 33). This driver has 2 pins for control, one to control the direction of the rotation(clockwise or counterclockwise) and the other one used to control the steps.

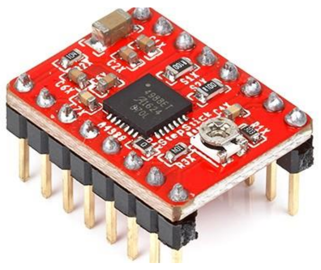


Figure 33: Stepper motor driver (A4988)

Technical Specifications:

- Minimum operating voltage: 8 V
- Maximum operating voltage: 35 V
- Continuous current per phase: 1 A
- Maximum current per phase: 2 A
- Minimum logic voltage: 3 V
- Maximum logic voltage: 5.5 V
- Size: 15.5 * 20.5 mm
- Weight: 1.3 g

3.4.4 Wiring

The wiring of every electrical component will be shown independently to facilitate the understanding of the cabling.

3.4.4.1 Servo motor

The same wiring will be for all the servo motors as shown in 34.

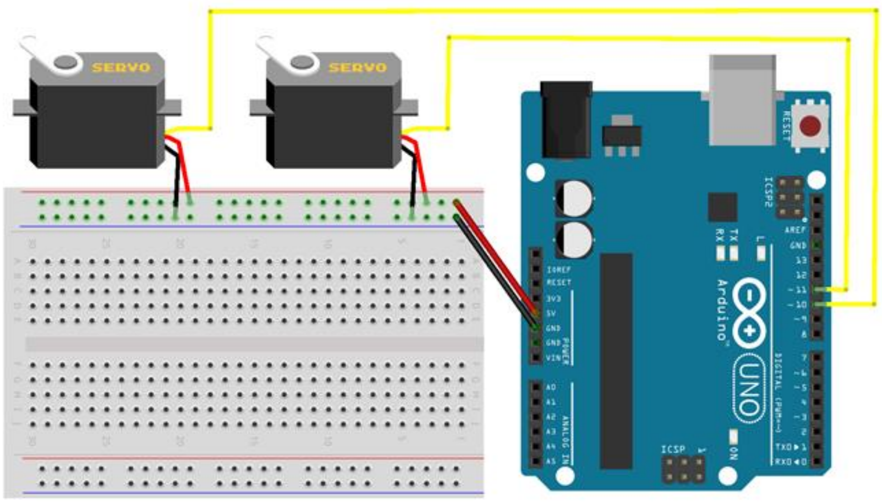


Figure 34: Wiring of the servo motors

3.4.4.2 Stepper motor with the A4988 driver

Wiring of a stepper motor with a driver as shown in Figure 35.

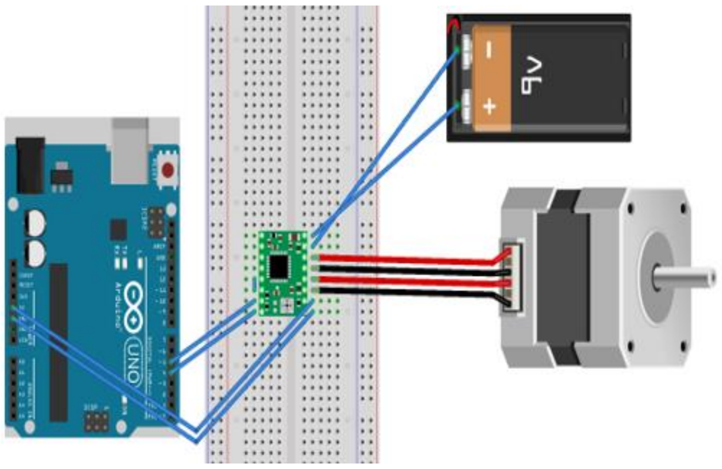


Figure 35: Wiring of the stepper motor

3.5 Wiring of the whole system

Wiring of all the subsystems togother (Figure 36)

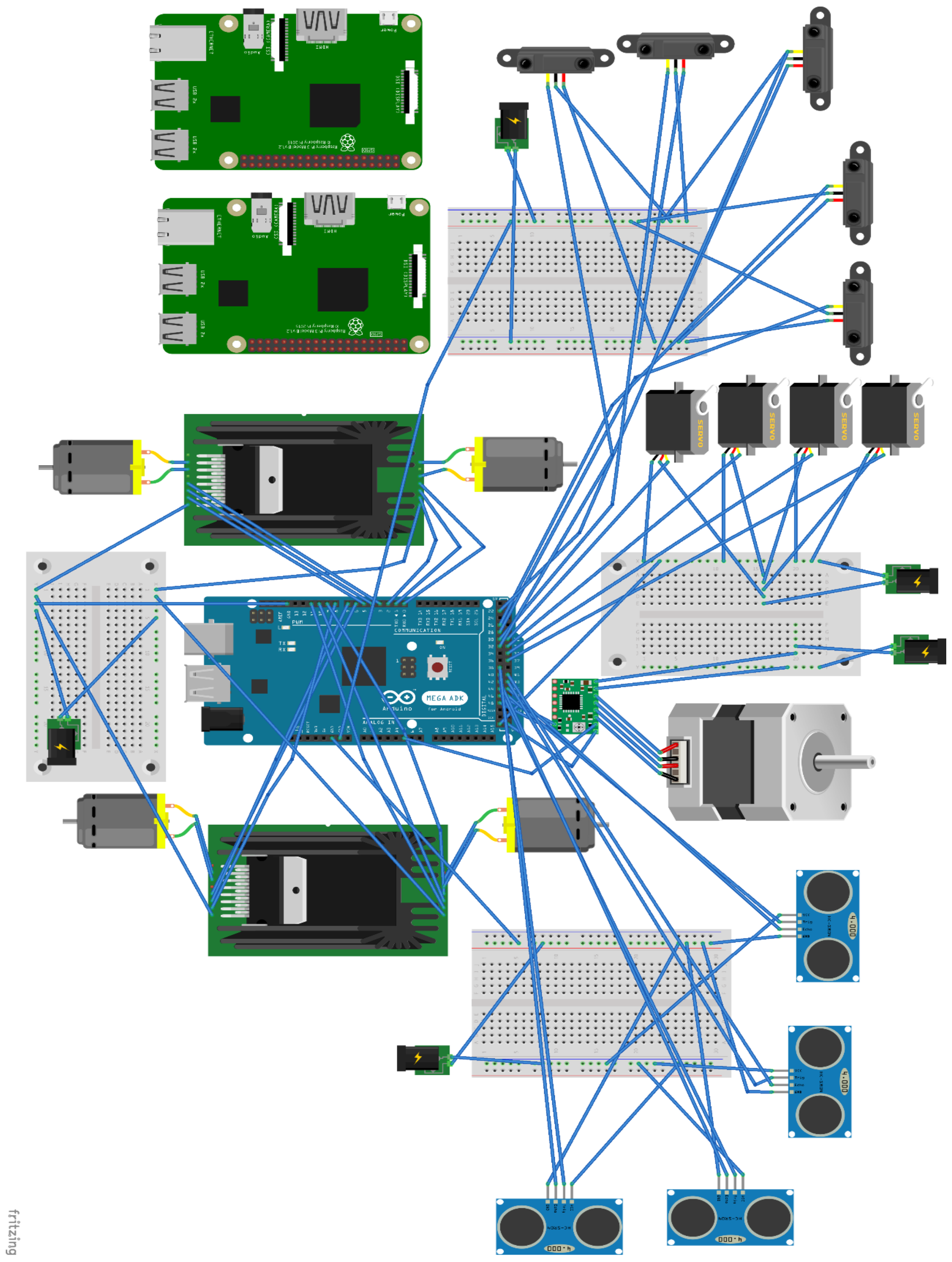


Figure 36: Wiring of the whole system

3.6 Control and algorithm

First thing we will have a list of items that we want the camera to identify and the robot arm to pick it and place it. As we know, the grocery store will be divided into multiple sections.

After knowing the items and the sections, the vehicle will start moving toward the sections using 5 infrared sensors to guide it.

To guide the vehicle to enter the correct section, another Raspberry pi camera will be connected to the system, with that being said, we have 2 raspberry pis, the main one which will be connected to the robot arm and the second raspberry pi which will guide the robot to the right path (section).

When the vehicle enters the section, the main Raspberry pi camera will start the scanning process.

The main Raspberry pi camera is decoding and comparing all the scanned bar codes with the item list that it has, if the item is correct, the Raspberry pi will give an order to the Arduino using serial communication to stop the 4 dc motors so the vehicle will stop and then the robot arm will start moving toward the item to pick it and then release the item in a given position which will be our cart, finally the robot arm will go to its initial position waiting for the next item and of course the vehicle will start moving again, this process will be repeated until all the item are found.

We will start by programming the 2 Raspberry pis by using Python as a programming language and then we will program the Arduino mega using the Arduino IDE to control everything based on some commands from the Raspberry pi.

The Raspberry pi will process all the data or frames coming from the camera and then it will decide whether to give the commands to the Arduino mega or not.

We will use Serial Communication to send data or commands from the Raspberry pi to the Arduino mega.

The main raspberry pi and the second raspberry will communicate over a network using UDP (User Datagram Protocol) to send data.

We will start by explaining the code of the Raspberry pi and then the Arduino mega.

3.6.1 Libraries used for the Raspberry pi

- OpenCV
- Pyzbar
- Pyserial
- Socket
- Picamera

3.6.2 Libraries used for the Arduino mega

- Servo
- Serial

3.6.3 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library mainly aimed at real-time computer vision (Figure 37).

Figure 37: OpenCv logo

Open cv is used in many applications such as:

- Face recognition
- Automated inspection and surveillance
- Number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Defect detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- Object recognition
- Medical image analysis

3.6.4 OpenCV Functions Documentation

- rectangle(): Draws a simple, thick, or filled up-right rectangle.
  Parameters: frame,point1,point2,color,thickness
- putText():Draws a text string.
  Parameters: frame,text,posLeft,font,fontScale,color,thickness
- line(): Draws a line segment connecting two points.
  Parameters: frame,point1,point2,color,thickness
- imshow(): Displays an image in the specified window.
  Parameters: name,frame
- moveWindow(): Moves the window to the specified position.
  Parameters: name,posX,posY
- waitKey(): Waits for a pressed key.
- release(): Closes video file or capturing device.
- destroyAllWindows(): Destroys all windows.

3.6.6 Pyzbar

Pyzbar a pure Python library that reads one-dimensional barcodes and QR codes using the zbar library, pyzbar is an open-source software suite for reading bar codes from various sources, such as video streams, image files, and raw intensity sensors.

3.6.7 Pyzbar documentation

We will use the decode function, we will pass it as a parameter the frame.

We will get:

```
Decoded (

        Data=b'pepsi', type='CODE128',

        rect=Rect(left=60,top=420,width=320,height=50),

        polygon=[

            Point(x=3,y=1), Point(x=3,y=50), Point(x=350,y=60),

            Point(x=350,y=0)

          ]  )
```

As we can see, the decode method will return some values, in our case we will use the data which will be the decoded barcode, and the position of the barcode so we can draw a rectangle surrounding it.

3.6.8 pyserial

PySerial is a library that provides support for serial connections. To establish a connection, we need to install the python module called pyserial. To start the connection, we need to know which port the Arduino is using. After knowing the port, we will use it in the python code to initiate the serial connection.

We will choose a 9600 baud rate.

3.6.9 Socket

Socket programming is a method that connects multiple systems on a network to communicate with each other. It is mostly used to create a client-server environment. The 2 raspberry pis will use UDP (User Datagram Protocol) as a communication protocol (Figure 38).

UDP stands for User Datagram Protocol is a communication protocol used to transfer data across a network.

The main raspberry pi will be the server and the second raspberry pi will be the client.

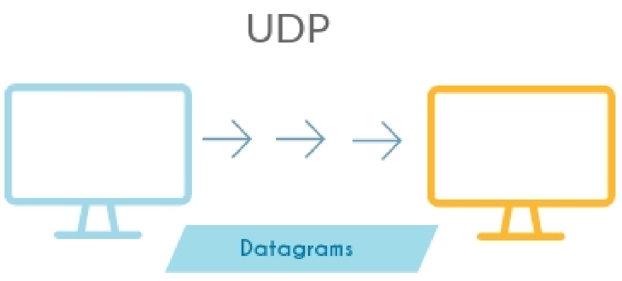


Figure 38: UDP communication

3.6.9 Algorithm

As mentioned before, this project has 3 subsystems, the vision system (the main raspberry pi with the camera) with the robot arm, the second raspberry pi with the camera to guide the vehicle to the right path, and finally the line following vehicle.

3.6.9.1 Robot arm algorithm

The main Raspberry Pi will be at the top of the robot arm so it could be able to scan 2 items at a time.

Knowing which item is correct and which is not is based on comparing the decoded barcode with a list of given items, if the item scanned is matching, now we have 2 option its either 1 barcode is correct or both. If both barcodes are matching, the raspberry pi will send 2 commands to the Arduino using serial communication, it depends on which barcode was decoded first. The second option is that 1 of the 2 barcodes is matching, in this case, the raspberry pi will know its position is either mid-top or mid-bottom and then send the command to the Arduino based on its position (Figure 39).

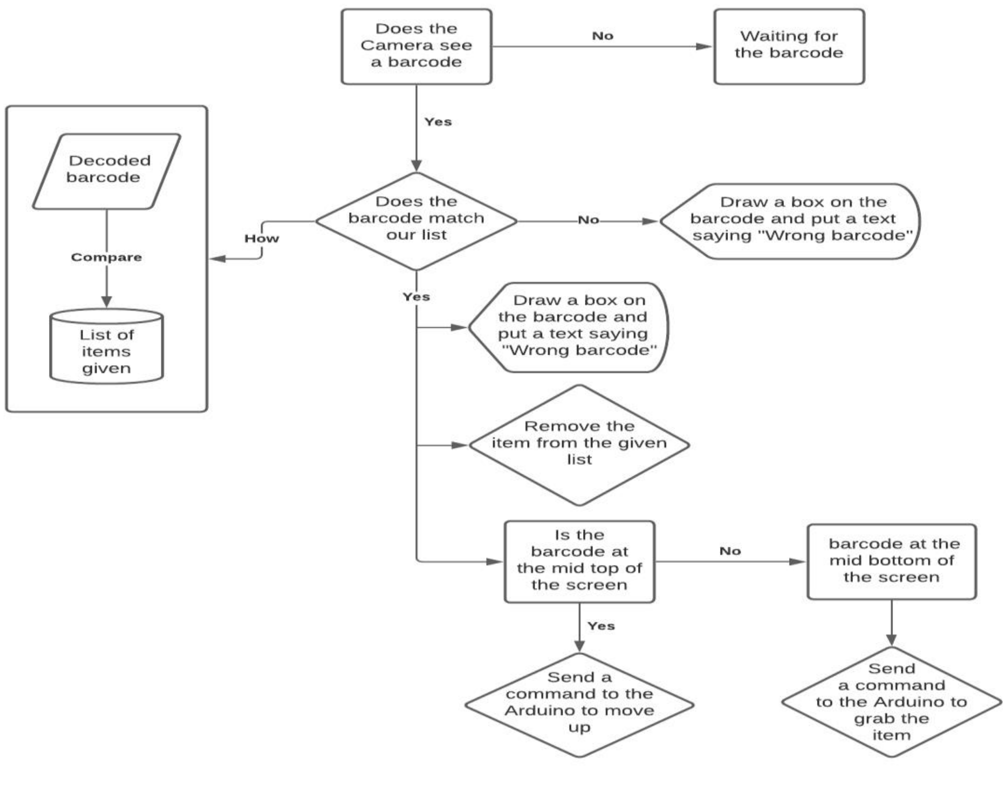
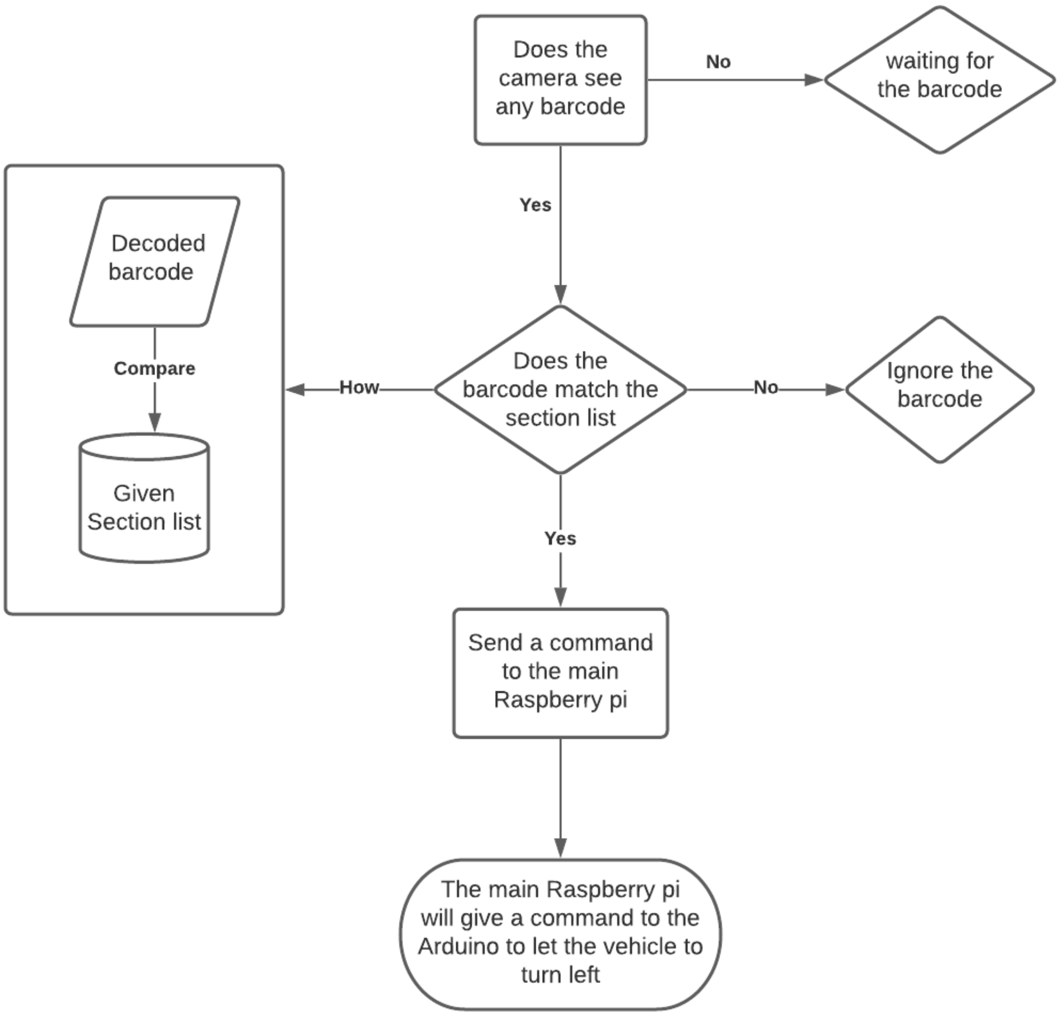

Does the Camera see a barcode
No
Waiting for the barcode
Yes
Decoded barcode
Compare
List of items given
How
Does the barcode match our list
No
Draw a box on the barcode and put a text saying "Wrong barcode"
Yes
Draw a box on the barcode and put a text saying "Wrong barcode"
Remove the item from the given list
Is the barcode at the mid top of the screen
No
barcode at the mid bottom of the screen
Yes
Send a command to the Arduino to move up
Send a command to the Arduino to grab the item

Figure 39: Flow chart diagram explaining the vision system of the robot arm algorithm

3.6.9.2 Section selection algorithm

In a grocery store, there are many sections with different item categories, for that we need to guide and let the robot know the right section or sections without wasting time searching in all the sections. So when the second raspberry pi camera sees the right section barcode, it will send a command to the main raspberry, and then the main raspberry pi will send a command to the Arduino to turn left (Figure 40).



Does the camera see any barcode
No
waiting for the barcode
Yes
Decoded barcode
Compare
Given Section list
How
Does the barcode match the section list
No
Ignore the barcode
Yes
Send a command to the main Raspberry pi
The main Raspberry pi will give a command to the Arduino to let the vehicle to turn left

Figure 40:Flow chart diagram explaining the section selection algorithm

3.6.9.3 Line following algorithm

As we said before, there is 2 algorithm that we can use to let the vehicle follow the line, first, we have the classic algorithm which is based on stopping the motors completely to make a turn (Figure 41), secondly, we have the PID algorithm which will adjust the speed of the motors to make a turn (Figure 42).

As we said we have 5 IR sensors:

Number 1: Far-Left IR sensor

Number 2: Left IR sensor

Number 3: Center IR sensor

Number 4: Right IR sensor

Number 5: Far-Right IR sensor

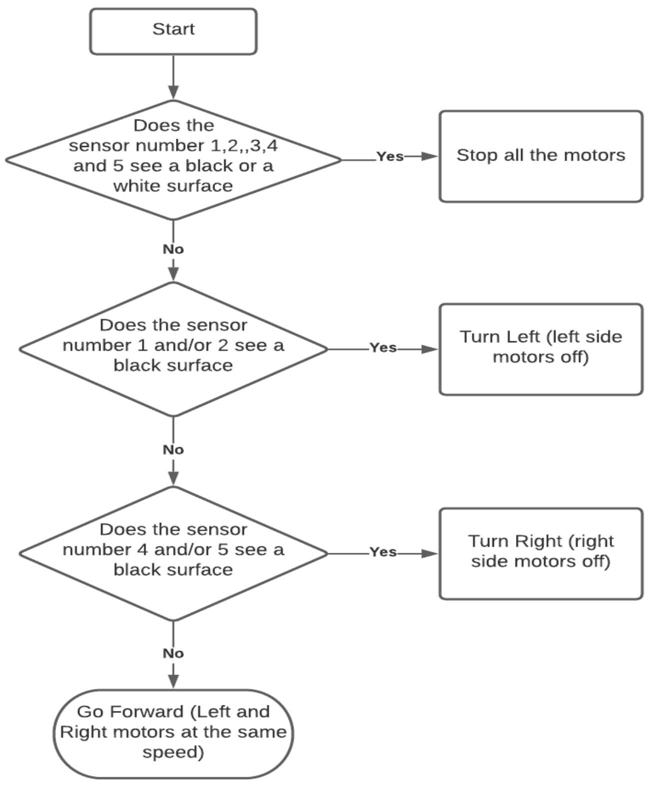

Start
Does the sensor number 1,2,,3,4 and 5 see a black or a white surface
Yes
Stop all the motors
No
Does the sensor number 1 and/or 2 see a black surface
Yes
Turn Left (left side motors off)
No
Does the sensor number 4 and/or 5 see a black surface
Yes
Turn Right (right side motors off)
No
Go Forward (Left and Right motors at the same speed)

Figure 41: Flow chart diagram explaining the IR classic algorithm

PD algorithm for line following (Figure 42)

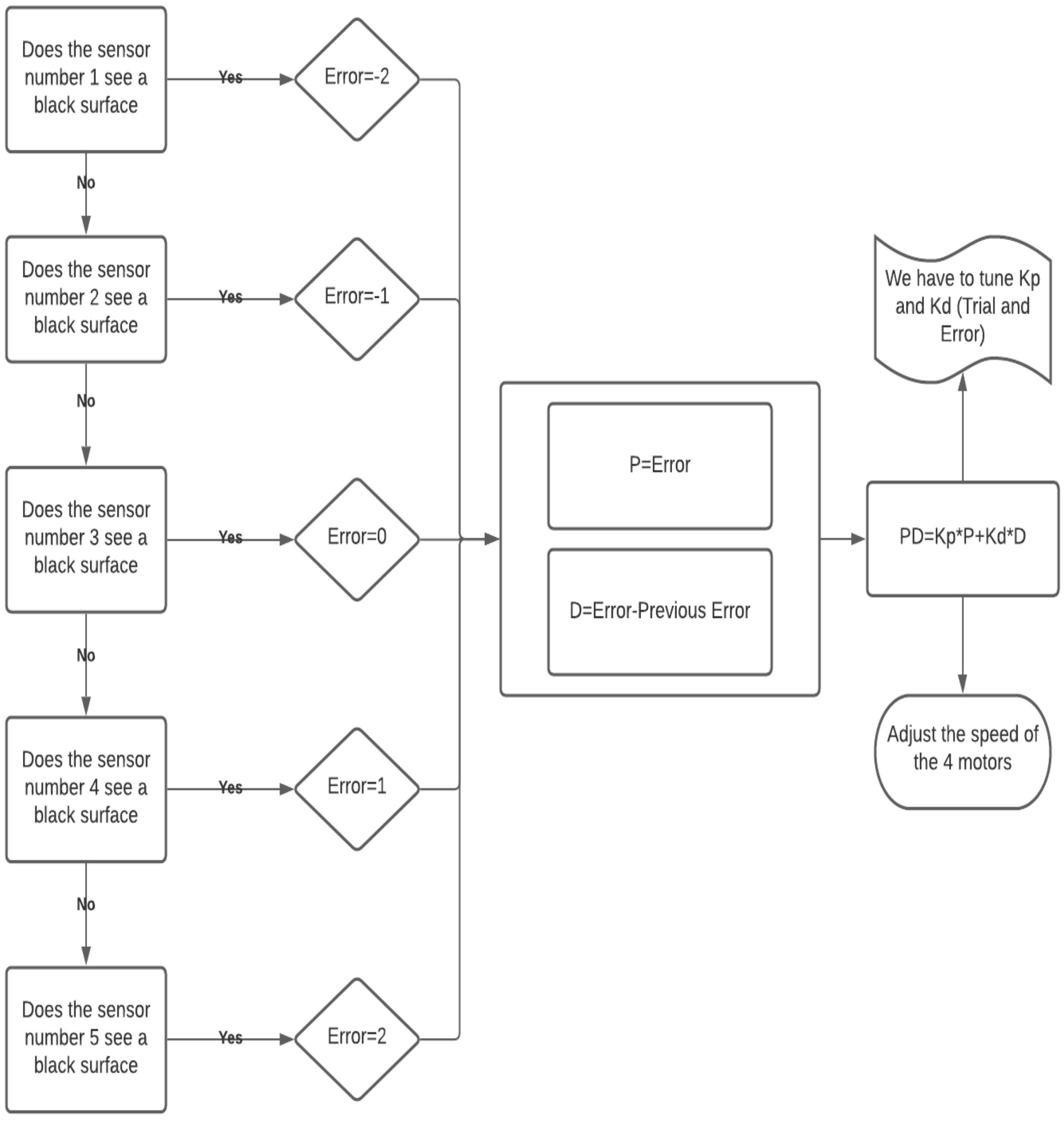

Does the sensor number 1 see a black surface
Yes
Error=-2
No
Does the sensor number 2 see a black surface
Yes
Error=-1
No
Does the sensor number 3 see a black surface
Yes
Error=0
No
Does the sensor number 4 see a black surface
Yes
Error=1
No
Does the sensor number 5 see a black surface
Yes
Error=2
P=Error
D=Error-Previous Error
PD=Kp*P+Kd*D
We have to tune Kp and Kd (Trial and Error)
Adjust the speed of the 4 motors

Figure 42: Flow chart diagram explaining the PD algorithm

The role of the Arduino is to control all the motor and the sensors (4 dc motor, stepper motor, 4 servo motors, infrared sensors, and ultrasonic sensors). The Arduino will control the vehicle not depending on the commands sent by the raspberry pi, the raspberry pi can just stop the vehicle for a moment until the robot arm picks and place the item. On the other hand, the robot arm will not move without a command from the raspberry pi (Figure 43).

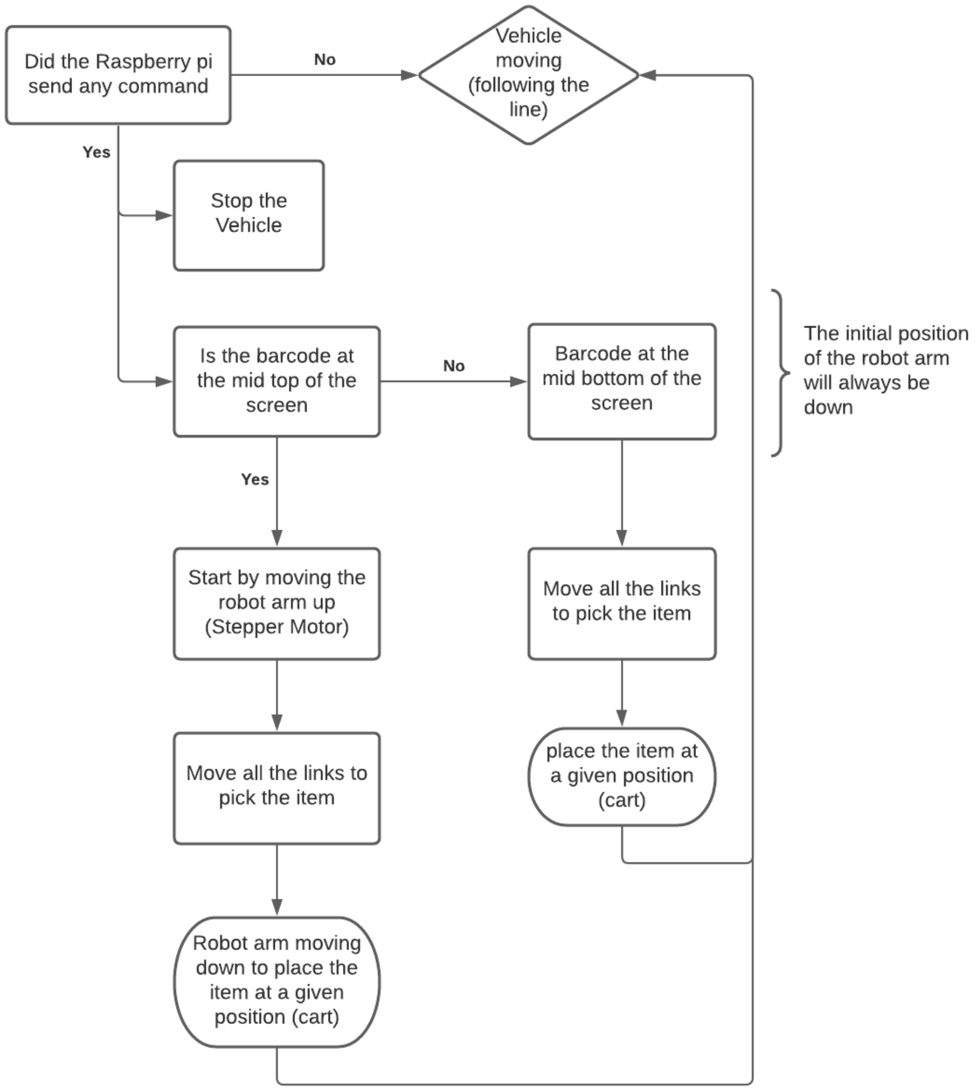

Did the Raspberry pi send any command
No
Vehicle moving (following the line)
Yes
Stop the Vehicle
Is the barcode at the mid top of the screen
No
Barcode at the mid bottom of the screen
The initial position of the robot arm will always be down
Yes
Start by moving the robot arm up (Stepper Motor)
Move all the links to pick the item
Move all the links to pick the item
place the item at a given position (cart)
Robot arm moving down to place the item at a given position (cart)

Figure 43: Flow chart diagram explaining the Arduino algorithm

For more details about all the algorithms, all the source codes will be available in Appendix 1

CHAPTER 4 RESULTS

The 2 raspberry pis working with the same methodology, with that being said we will review only the results of the main raspberry pi (robot arm).

To get the item that we want, we need to write them in a list (text file) to be able to read them in the program and compare them with the decoded barcodes (Figure 44).

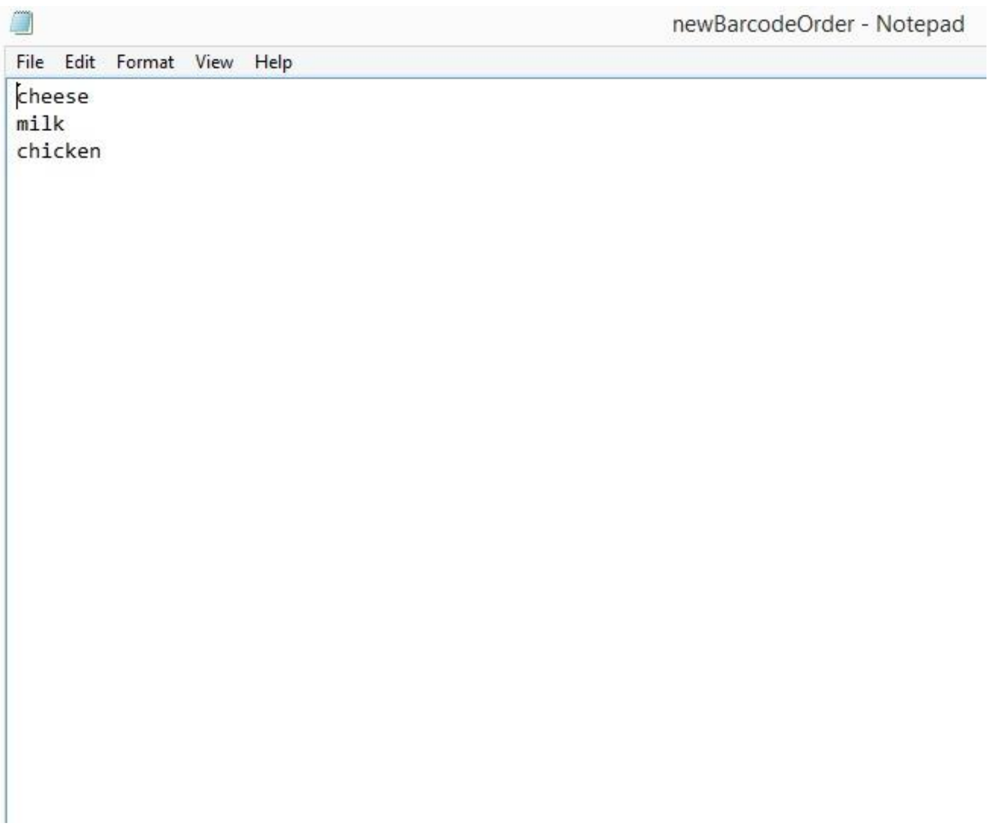
First, when the script is running, a message will be printed saying that the system is starting to make sure that the flow of the program is correct, now using the list of items, we can observe the names of the wish list items, the total amount of items, the product found (it will be equal to 0 at the start of the program) and how many items are left (Figure 45).

As previously mentioned, the camera can scan 2 barcodes at a time, with that being said, 3 cases can occur, first case scenario that the 2 barcodes are correct (matching) (Figure 46), the second case is that one of the barcodes is correct (Figure 48) and lastly, the 2 barcodes are not matching (wrong barcode) (Figure 51).

To demonstrate the results of this project, we will take an example of a couple of items (Figure 44).

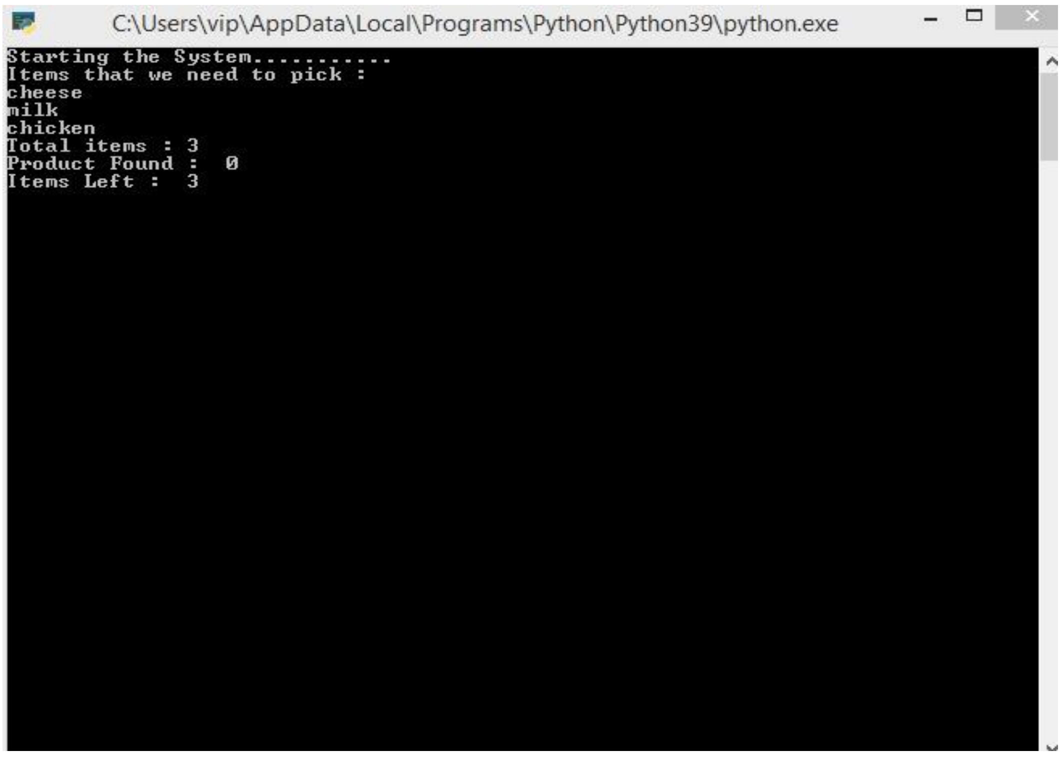
Our item list will contain:

- cheese
- milk
- chicken



```
cheese
milk
chicken
```

Figure 44: List of items


C:\Users\vip\AppData\Local\Programs\Python\Python39\python.exe

```
Starting the System...........
Items that we need to pick :
cheese
milk
chicken
Total items : 3
Product Found :  0
Items Left :  3
```

Figure 45: Starting the system

As shown in Figure 32, the camera was able to decode the 2 barcodes, draw a box colored in green surrounding them saying “CORRECT ITEM” (Cheese and milk) (Figure 46) and inform us of the name of the item found, the position of the 2 barcodes, the command sent to the Arduino (we printed the steps of the process just for demonstration it has nothing to do with the Arduino) and the updated products found and the items left (Figure 47).

The vehicle will stop to pick and place the 2 items.

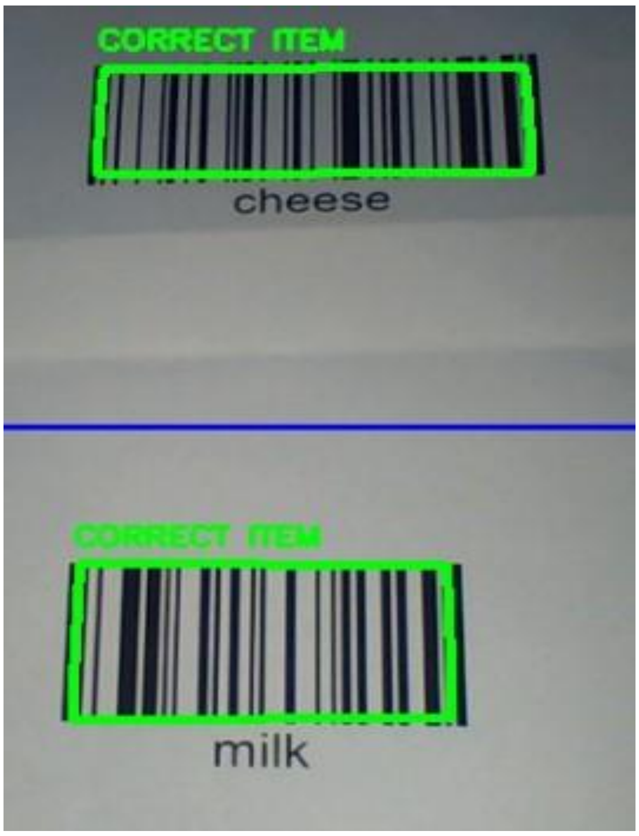
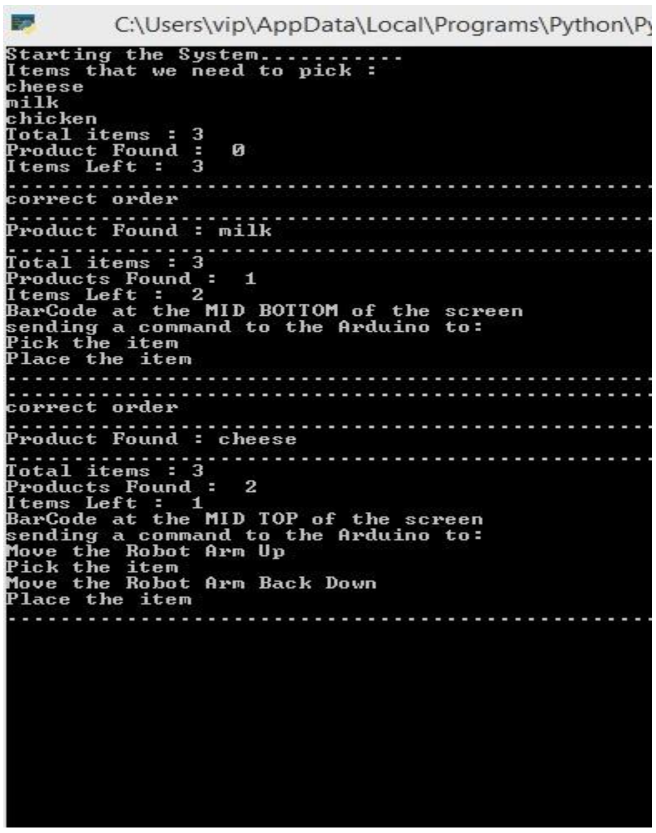


Figure 46: 2 matching items

The print statement when the camera sees two matching barcodes at a time.



```
C:\Users\vip\AppData\Local\Programs\Python\Py
Starting the System...........
Items that we need to pick :
cheese
milk
chicken
Total items : 3
Product Found :  0
Items Left :  3
------------------------------------------------
correct order
------------------------------------------------
Product Found : milk
------------------------------------------------
Total items : 3
Products Found :  1
Items Left :  2
BarCode at the MID BOTTOM of the screen
sending a command to the Arduino to:
Pick the item
Place the item
------------------------------------------------
------------------------------------------------
correct order
------------------------------------------------
Product Found : cheese
------------------------------------------------
Total items : 3
Products Found :  2
Items Left :  1
BarCode at the MID TOP of the screen
sending a command to the Arduino to:
Move the Robot Arm Up
Pick the item
Move the Robot Arm Back Down
Place the item
------------------------------------------------
```

Figure 47: The print statement for the matching barcodes

The same operation will take place for the second case which is that we have only one matching barcode, but the box drawn on the wrong item will be colored in red and the text that will appear on the screen will be “WRONG ITEM”.(Figure 48)

As shown in Figure 49, the program neglected the wrong barcode and did not take any action.

The vehicle will stop to only pick and place one item.

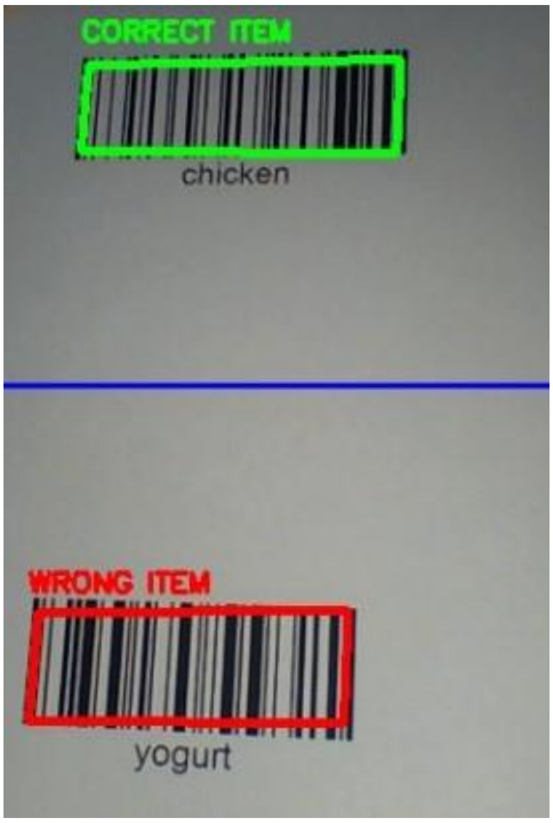
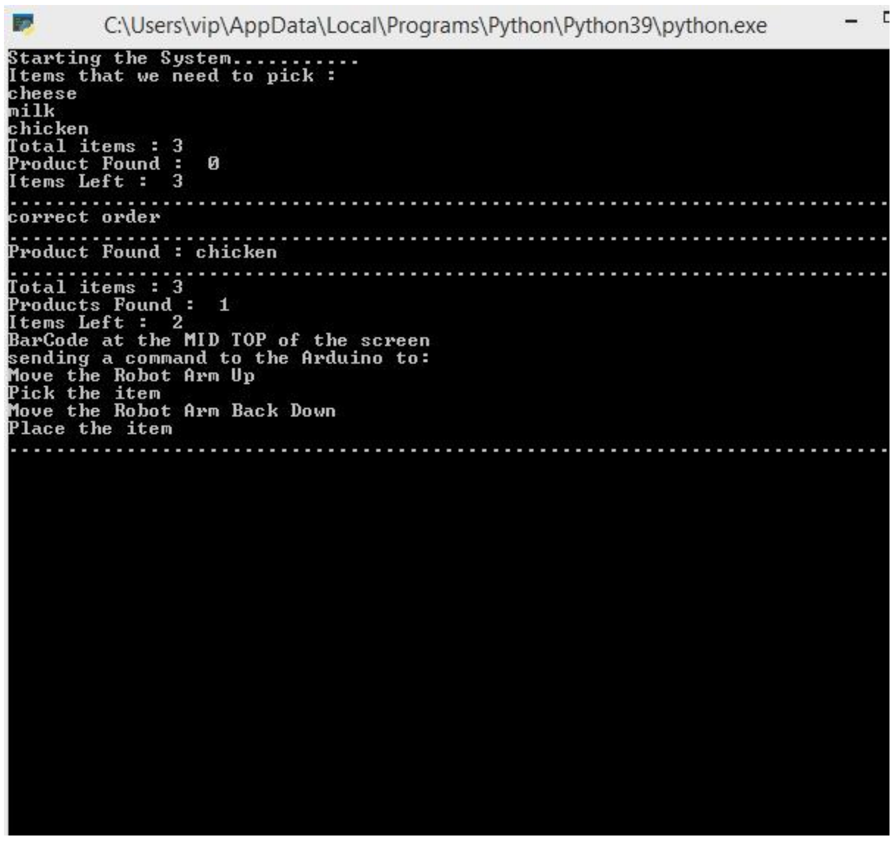


Figure 48: One matching Barcode

The print statement when the camera sees one matching barcode.



```
C:\Users\vip\AppData\Local\Programs\Python\Python39\python.exe
Starting the System...........
Items that we need to pick :
cheese
milk
chicken
Total items : 3
Product Found :  0
Items Left :  3
--------------------------------------------------------------------------
correct order
--------------------------------------------------------------------------
Product Found : chicken
--------------------------------------------------------------------------
Total items : 3
Products Found :  1
Items Left :  2
BarCode at the MID TOP of the screen
sending a command to the Arduino to:
Move the Robot Arm Up
Pick the item
Move the Robot Arm Back Down
Place the item
--------------------------------------------------------------------------
```

Figure 49: The print statement for one matching barcode

The last case is that the 2 barcodes are not matching, a box will be drawn on both and a text will be shown on the screen (Figure 51), but the program will not take any action (Figure 50) and the vehicle will not stop.

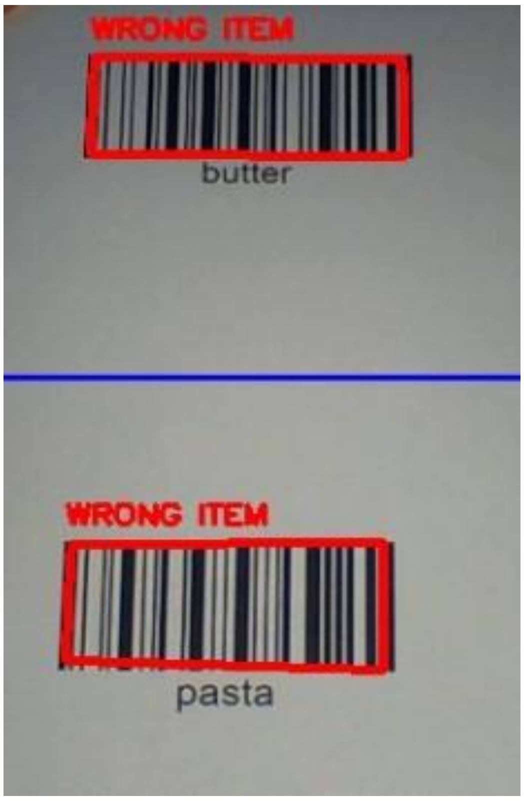
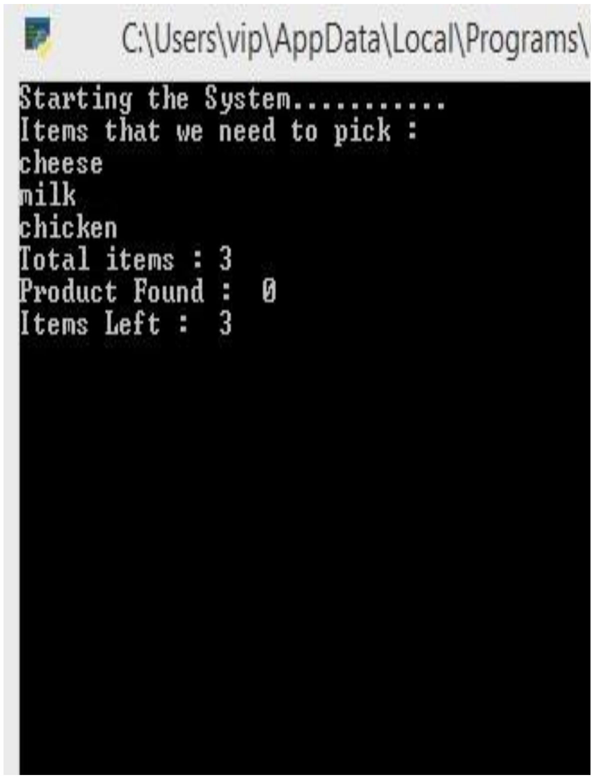


Figure 51: Zero matching barcode



```
C:\Users\vip\AppData\Local\Programs\
Starting the System...........
Items that we need to pick :
cheese
milk
chicken
Total items : 3
Product Found :  0
Items Left :  3
```

Figure 50: The print statement for zero matching barcode

CHAPTER 5 DISCUSSION

Thankfully we achieved what we wanted by controlling a movable pick and place robot arm using computer vision to be able to scan, compare between barcodes, and taking decisions 100% autonomously without the intervention of humans. Besides that, we were capable of connecting a line following vehicle to the robot arm to create a fully functional movable pick and place robot arm system.

We effectively improved the project of another student and put it as a part of this project, the improvement achieves was by modifying the design to change the axe of motion and adding a camera device connected to a raspberry pi to make the robot arm smarter and independent.

Then place the modified robot arm into another system that consists of a vehicle with some sensors and connect them using an Arduino and raspberry pi to form an integrated system capable of achieving the goal, finally, we showed how we transformed a stationary project with a limited job to a movable system that can be used in a lot of applications.

CHAPTER 6 CONCLUSION AND FUTURE WORK

In conclusion, we can say that we presented a vision-based robot manipulator that can scan and compare different bar codes, knowing which item is correct and which is not, placed on top of a vehicle that follows a path, to obtain a movable pick and place robot arm.

As a future scope, we can try to create a website that can send the list of items directly instead of writing it manually to the Raspberry pi so everything will be automated and finally we will try a build a self-driving vehicle to avoid any collision if there is more than one vehicle.

REFERENCES

Borkar, V. & Andurkar, G.K. (2017). Development of pick and place for industrial applications. *International Research Journal of Engineering and Technology (IRJET), 4*(9), 347-356.

Kumar, S. S. (2015). Design of Pick and Place Robot. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 4*(6), 4887-4898. DOI: 10.15662/ijareeie.2015.0406112

Teodorescu, C.S., Vandenplas, S., Depraetere, B., Anthonis, J., Steinhauser, A., & Swevers, J. (2016). A fast pick-and-place prototype robot: design and control. *IEEE Multi-Conference on Systems and Control 2016 At Buenos Aires, Argentina, 1*-13. DOI:10.1109/CCA.2016.7588005

Younus, M., Gadekar, P. & Walse, A. (2019). Line Follower Using Arduino And Its Applications. *International Journal of Applied Engineering Research, 14*(13), 156-161.

YOUSSEF, A. (2020). Development and Control of a 5-DOF Pick and Place Robot. 1-53. https://ufe.edu.eg/

Appendix 1

The main raspberry pi code:

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
from pyzbar.pyzbar import decode
import numpy as np
import serial
import socket
camera=PiCamera()
camera.resolution=(640,480)
port=serial.Serial('/dev/ttyACM0',9600)
UDP_IP=”192.168.0.4”
UDP_PORT=5005
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind((UDP_IP,UDP_PORT))
camera.framerate=32
rawCapture=PiRGBArray(camera,size=(640,480))
time.sleep(0.1)
with open('/home/pi/Desktop/order') as f:
    order1=f.read().splitlines()
size=len(order1)
listY=[0]*size
listI=[0]
ProductFound=0
TotalOrder=size
ItemsLeft=TotalOrder-ProductFound
print('Total items :',TotalOrder)
print('Product Found : ',ProductFound)
print('Items Left : ',ItemsLeft)

for image in camera.capture_continuous(rawCapture,format="bgr",use_video_port=True):
    frame=image.array
    barc=decode(frame)
    font=cv2.FONT_HERSHEY_SIMPLEX
    datas, addr=sock.recvfrom(1024)
    datas=int(datas)
     if datas == 2:
        port.write(b”12”)//turn left

    if listI[0]==0:
       port.write(b'0')
       listI[0]=1
       print(listI)

    for barcode in barc:
       BarCdata=barcode.data.decode('utf-8')
       #print(BarCdata)
       x,y,w,h=barcode.rect
       #print(listY)
       for i in range(size):
             if ((BarCdata in order1[i]) and (listY[i]== 0) and 300<x<400):
                    print('..............................................................................')
                    print('correct order')
                    print('..............................................................................')
                    print('..............................................................................')
                    listY[i]=1
                    ProductFound+=1
                    TotalOrder=size
                    ItemsLeft=TotalOrder-ProductFound
                    print('Total items :',TotalOrder)
```

```
            print('Product Found : ',ProductFound)
            print('Items Left : ',ItemsLeft)



            if (y<240 and 300<x<400):
                print('BarCode up')
                port.write(b'2')
                print('...............................................................')
            elif(y>240 and 300<x<400):
                print('Barcode Down')
                port.write(b'1')
                print(x)
                print(y)
                print('...............................................................')
            else:
                print('out............................................................')

        if BarCdata in order1:
            #print('right order(bar code)')
            scanning='right order'
            color=(0,255,0)
        else:
            #print('wrong order(bar code)')
            scanning='wrong order'
            color=(0,0,255)

       cv2.rectangle(frame,(x,y),(x+w,y+h),color,4)
        cv2.putText(frame,scanning,(x,y-10),font,0.5,color,2)
        cv2.circle(frame,(x,y),5,(0,0,255),-1)
    cv2.line(frame,(0,240),(640,240),(255,0,0),2)
    cv2.line(frame,(300,0),(300,480),(0,0,255),2)
    cv2.line(frame,(400,0),(400,480),(0,0,255),2)
```

```
   #cv2.circle(frame,(x,y),5,(0,0,255),-1)
   #cv2.circle(frame,(100,200),30,(0,255,0),-1)
    cv2.imshow('laptop cam',frame)
    cv2.moveWindow('laptop cam',0,0)

    key=cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)

    if key == ord('q'):
         break

cv2.destroyAllWindows()
```

The second raspberry pi code:

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
from pyzbar.pyzbar import decode
import numpy as np
import serial
import socket
camera=PiCamera()
camera.resolution=(640,480)
port=serial.Serial('/dev/ttyACM0',9600)
UDP_IP=”192.168.0.4”
UDP_PORT=5005
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind((UDP_IP,UDP_PORT))
turnleft=”2”
```

```
camera.framerate=32
rawCapture=PiRGBArray(camera,size=(640,480))
time.sleep(0.1)
with open('/home/pi/Desktop/section') as f:
   section=f.read().splitlines()
size=len(section)
listY=[0]*size
listI=[0]


for image in camera.capture_continuous(rawCapture,format="bgr",use_video_port=True):
   frame=image.array
   barc=decode(frame)
   font=cv2.FONT_HERSHEY_SIMPLE

   for barcode in barc:
      BarCdata=barcode.data.decode('utf-8')
      x,y,w,h=barcode.rect
      for i in range(size):
            if ((BarCdata in order1[i]) and (listY[i]== 0) ):
                  print('...............................................................................')
                  listY[i]=1
                sock.sendto(turnleft,(UDP_IP,UDP_PORT))

      if BarCdata in order1:
          scanning='right section'
          color=(0,255,0)
      else:
          scanning='wrong section'
          color=(0,0,255)

     cv2.rectangle(frame,(x,y),(x+w,y+h),color,4)
```

```
        cv2.putText(frame,scanning,(x,y-10),font,0.5,color,2)
        cv2.circle(frame,(x,y),5,(0,0,255),-1)

    cv2.imshow('laptop cam',frame)
    cv2.moveWindow('laptop cam',0,0)

    key=cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)

    if key == ord('q'):
        break

cv2.destroyAllWindows()
```

Arduino mega code:

```
#include <Servo.h>
Servo servo;
Servo servo1;
Servo servo2;
Servo servo3;
int posServo = 0;
int posServo1=0;
int posServo2=0;
int posServo3=0;
const int dirpin=28;
const int steppin=30;
const int stepsPerRevolution=200;
int stepsFor10cmUp=0;
int stepsFor10cmDown=0;
int cyclesUp=6;
int cyclesDown=6;
int dataSerial;
int trig=24;
int echo=26;
long duration;
int distance;
int FrontRightMotor1=2;
int FrontRightMotor2=3;

int FrontLeftMotor1=5;
int FrontLeftMotor2=6;



int BackRightMotor1=11;
int BackRightMotor2=12;

int BackLeftMotor1=8;
int BackLeftMotor2=9;


int PwmEnaAFRM=4;
int PwmEnaBFLM=7;
int PwmEnaABRM=13;
int PwmEnaBBLM=10;

int fr;
int r;
int c;
int l;
int fl;
int pinfr=22;
int pinr=24;
int pinc=26;
int pinl=28;
int pinfl=30;
int maxSpeedleft=45;
int maxSpeedright=55;
float kp=45;
float kd=25;
float error=0,P=0,D=0,PD_value=0;
float prev_error=0;

void moveDown(){

  digitalWrite(dirpin,LOW);
  for (int cyc=0; cyc <= cyclesDown; cyc++){
    for(int x=0;x<stepsPerRevolution;x++){
      digitalWrite(steppin,HIGH);
      delayMicroseconds(1000);
      digitalWrite(steppin,LOW);
      delayMicroseconds(1000);

      }
    delay(1000);
    }
  }

void moveUp(){

  digitalWrite(dirpin,HIGH);

  for (int cyc=0; cyc <= cyclesUp; cyc++){
    for(int x=0;x<stepsPerRevolution;x++){
      digitalWrite(steppin,HIGH);
      delayMicroseconds(1000);
      digitalWrite(steppin,LOW);
      delayMicroseconds(1000);

      }
    delay(1000);
    }
  }

void From0to90Servo(){
  for (posServo = 0; posServo <= 90; posServo += 1) {
    // in steps of 1 degree
    servo.write(posServo);
    delay(25);
  }
  //servo.detach();
}

void From90to0Servo(){
  for (posServo = 90; posServo >= 0; posServo -= 1) {
    servo.write(posServo);
    delay(25);
  }
  }

 void From90to180Servo(){
  for (posServo = 90; posServo <= 180; posServo += 1) {
    // in steps of 1 degree
    servo.write(posServo);
    delay(25);
  }
  //servo.detach();
}

void From180to90Servo(){
  for (posServo = 180; posServo >= 90; posServo -= 1) {
    servo.write(posServo);
    delay(25);
  }
  //servo.detach();
  }

 void From0to90Servo1(){
  for (posServo1 = 0; posServo1 <= 90; posServo1 += 1) {
    // in steps of 1 degree
    servo1.write(posServo1);
    delay(25);
  }
}

void From90to0Servo1(){
  for (posServo1 = 90; posServo1 >= 0; posServo1 -= 1) {
    servo1.write(posServo1);
    delay(25);
  }
  }
 void From90to180Servo1(){
  for (posServo1 = 90; posServo1 <= 180; posServo1 += 1) {
    // in steps of 1 degree
    servo1.write(posServo1);
    delay(25);
  }
}

void From180to90Servo1(){
  for (posServo1 = 180; posServo1 >= 90; posServo1 -= 1) {
    servo1.write(posServo1);
    delay(25);
  }
  }

void From180to0Servo(){
  for (posServo = 180; posServo >= 0; posServo -= 1) {
    servo.write(posServo);
    delay(25);
  }
  //servo.detach();
  }
void From180to0Servo1(){
  for (posServo1 = 180; posServo1 >= 0; posServo1 -= 1) {
    servo1.write(posServo1);
    delay(25);
  }
  }
void stopp(){
  digitalWrite(FrontRightMotor1,LOW);
  //digitalWrite(FrontRightMotor2,LOW);
  digitalWrite(FrontLeftMotor1,LOW);
  //digitalWrite(FrontLeftMotor2,HIGH);
  digitalWrite(BackRightMotor1,LOW);
  //digitalWrite(BackRightMotor2,HIGH);
  //digitalWrite(BackLeftMotor1,HIGH);
  digitalWrite(BackLeftMotor2,LOW);



}



void setup() {
  // put your setup code here, to run once:

  servo.attach(32);
  servo1.attach(34);
  servo2.attach(38);
  servo3.attach(40);
  pinMode(steppin,OUTPUT);
  pinMode(dirpin,OUTPUT);
  Serial.begin(9600);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
   pinMode(pinfr,INPUT);
  pinMode(pinr,INPUT);
  pinMode(pinc,INPUT);
  pinMode(pinl,INPUT);
  pinMode(pinfl,INPUT);
  pinMode(FrontRightMotor1,OUTPUT);
  pinMode(BackRightMotor1,OUTPUT);
  pinMode(FrontLeftMotor1,OUTPUT);
  pinMode(BackLeftMotor1,OUTPUT);
  pinMode(FrontRightMotor2,OUTPUT);
  pinMode(BackRightMotor2,OUTPUT);
  pinMode(FrontLeftMotor2,OUTPUT);
  pinMode(BackLeftMotor2,OUTPUT);
  pinMode(PwmEnaAFRM,OUTPUT);
  pinMode(PwmEnaBFLM,OUTPUT);
  pinMode(PwmEnaABRM,OUTPUT);
  pinMode(PwmEnaBBLM,OUTPUT);


}


void forward(){

  digitalWrite(FrontRightMotor1,HIGH);
  //digitalWrite(FrontRightMotor2,HIGH);
  digitalWrite(FrontLeftMotor1,HIGH);
  //digitalWrite(FrontLeftMotor2,HIGH);
  digitalWrite(BackRightMotor1,HIGH);
  //digitalWrite(BackRightMotor2,HIGH);
  //digitalWrite(BackLeftMotor1,HIGH);
  digitalWrite(BackLeftMotor2,HIGH);


}


void loop() {
  digitalWrite(trig,LOW);
  delayMicroseconds(2);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  duration=pulseIn(echo,HIGH);
  distance=duration*0.034/2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(10);
   fr=digitalRead(pinfr);
  r=digitalRead(pinr);
  c=digitalRead(pinc);
  l=digitalRead(pinl);
  fl=digitalRead(pinfl);


  if(distance>5){
      if (fr==0 && r==0 && c == 1 && l==0 && fl==0){
    error=0;}
  else if (fr==0 && r==1 && c == 0 && l==0 && fl==0){
    error=1;}
  else if (fr==1 && r==0 && c == 0 && l==0 && fl==0){
```

```
  error=2;}

else if (fr==1 && r==1 && c == 0 && l==0 && fl==0){
  error=2;}



else if (fr==0 && r==1 && c == 1 && l==0 && fl==0){
  error=1;}

else if (fr==0 && r==0 && c == 0 && l==1 && fl==0){
  error=-1;}
else if (fr==0 && r==0 && c == 0 && l==0 && fl==1){
  error=-2;}

else if (fr==0 && r==0 && c == 1 && l==1 && fl==0){
  error=-1;}
else if (fr==0 && r==0 && c == 0 && l==0 && fl==0){
  error=0;}

else if (fr==0 && r==0 && c == 0 && l==1 && fl==1){
  error=-2;}

else if (fr==1 && r==1 && c == 0 && l==0 && fl==0){
  error=2;}


else if (fr==0 && r==0 && c == 0 && l==1 && fl==1){
  error=-2;}
//forward();
P=error;
D=error - prev_error;

PD_value=(kp*P)+(kd*D);
```

```
 error=prev_error;

 int left_motor_speed = maxSpeedleft + PD_value;
 int right_motor_speed = maxSpeedright - PD_value;



 left_motor_speed = constrain(left_motor_speed, 0, 255);
 right_motor_speed = constrain(right_motor_speed, 0, 255);

 analogWrite(PwmEnaAFRM, right_motor_speed);
 analogWrite(PwmEnaABRM, right_motor_speed);
 analogWrite(PwmEnaBFLM, left_motor_speed);
 analogWrite(PwmEnaBBLM, left_motor_speed);

 forward();



 }
 else{

     stopp();



   }

while(Serial.available()){

 dataSerial=Serial.read();
 if(dataSerial == '0'){ /// starting position
     servo.write(0);
     delay(2000);
     servo1.write(0);
```

```
    delay(2000);



  }
else if (dataSerial == '1'){// matching correct bar code
    stopp();
    delay(2000);
    From0to90Servo();
    delay(2000);
    From0to90Servo1();
    delay(2000);
    servo3.write(60)
    delay(2000);
    servo.write(30);
    servo2.write(25);
    delay(2000);
    servo2.write(160);
    delay(2000);
    From90to180Servo();
    delay(2000);
    From90to180Servo1();
    delay(2000);
     servo3.write(30);
     delay(2000);
     servo3.write(30)
    servo2.write(25);
    delay(2000);
    servo2.write(160);
    delay(2000);
    From180to0Servo();
    delay(2000);
    From180to0Servo1();
```

```
   delay(2000);
   }

else if (dataSerial == '2'){// correct match but the upper barcode
   stopp();
   moveUp();
   delay(2000);
   From0to90Servo();
   delay(2000);
   From0to90Servo1();
   delay(2000);
   servo3.write(60)
   delay(2000);
   servo3.write(30)
   servo2.write(25);
   delay(2000);
   servo2.write(160);
   delay(2000);
   From90to180Servo();
   delay(2000);
   From90to180Servo1();
   delay(2000);
   moveDown();
   servo3.write(60)
   delay(2000);
   servo3.write(30)
   servo2.write(25);
   delay(2000);
   servo2.write(160);
   delay(2000);
   From180to0Servo();
   delay(2000);
```

```
      From180to0Servo1();
      delay(2000);

      }

   }

 }
```