Self Navigating Robot Car with Object Detection

by

Buse Felamur

Engineering Project Report

Yeditepe University

Faculty of Engineering

Department of Computer Engineering

2019

Self Navigating Robot Car with Object Detection

APPROVED BY:

Assist.Prof.Dr. Dionysis GOULARAS ………………………………
(Supervisor)

Assist.Prof.Dr. Tacha SERIF ………………………………

Assist.Prof.Dr. Esin ONBASIOGLU ………………………………

DATE OF APPROVAL: 14/01/2019

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Assist Prof. Dr. Dionysis Goularas for his guidance and support throughout my project. Also I would like to thank Abdullah Yıldız for his help on my project.

ABSTRACT

Self Navigating Robot Car with Object Detection

Humanity builds machines so they do the dangerous or unwanted jobs instead of them. The aim of this project is to create an application that in collaboration with a small robot which will be able to navigate, recognize objects and have a specific behavior depending of the object it finds. This makes it safer for people to search and find specific objects if environment is unsafe for humans to enter or makes life of people easier if simply searching and finding is an undesirable job for people do manually.

In this project PC and Raspberry Pi connected to same network via Wi-Fi to communicate with each other. Raspberry Pi application uses Raspberry Camera to recognize objects and it frequently communicates with Arduino since it handles navigation with the help of ultrasonic distance sensor. After robot car achieves its goal, Raspberry Pi uses camera to send found objects photo to PC and causing PC to make a sound, indicating correct object is found.

ÖZET

Self Navigating Robot Car with Object Detection

İnsanlık, makineleri tehlikeli ve istenmeyen görevleri, kendimizin yerine yapmaları için yarattı. Bu projenin amacı; kendi yönünü bulabilen, cisimleri tanıyabilen ve tanıdığı cisimlerin ne olduğuna göre tepki veren küçük bir robot ile birlikte çalışabilen bir uygulama yaratmaktı. Bu robot eğer içinde olduğumuz ortam insanların girmesi için tehlikeliyse, cisimleri arama ve bulmayı daha güvenilir kılıyor yada eğer sadece bu arama ve bulma istenmeyen bir görevse, insanların yaşamını daha kolaylaştırıyor.

Bu projede, bilgisayar ve Raspberry Pi aynı Wi-Fi ağına bağlanarak iletişime geçiyorlar. Raspberry Pi, Raspberry Kamerasını kullanarak cisimleri tanıyor ve aynı zamanda Arduino ile sürekli iletişim kuruyor çünkü Arduino, robotun kendi yönünü bulmasını ses ötesi uzaklık sensorları ile sağlıyor. Robot amacına ulaştıktan sonra, Raspberry Pi kamerası ile cismin çekilen resmini bilgisayara gönderiyor ve bu yüzden bilgisayar doğru cismin bulunduğunu bildirmek için ses çıkarıyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

USB	Universal Serial Bus
PC	Personal Computer
GND	Ground
VCC	Voltage at the Common Collector
V	Volt
IN	Input
EN	Enable
HD	High Definition
TCP	Transmission Control Protocol

1. INTRODUCTION

Technological developments assisted people in their lives to make their life easier. Many people benefitted from this development to solve their daily life problems. There are multiple branches of technology, robotics being one of them.

Robotics branch uses technologies like computer systems, information processing and sensory feedback to develop machines that can take place of humans and act like a human would do. In this current time, robot are more easy to use and affordable. Robotics are being used in many fields ranging from medical to industrial, because of this.

Using robots are more efficient since they can work in every environment whether it is safe or not. They can do the threatening jobs like handling toxic substances or entering unsafe buildings. This can prevent humans from getting fatal wounds.

Another benefit for using robots are their capability. They can perform tasks more consistent and faster than humans. This increases production and saves time. In the medical field they can even perform tasks humans can't do. For example in surgeries, human hand can't fit and reach everywhere but an robot can.

Robots usually made from three main part. Controller, sensors and mechanical part. Mechanical parts are any gear that makes it move or interact with surrounding area like motors and wheels. Controller is acts as a brain and do tasks. Sensors used to tell robot of its surroundings. For example it can be a distance sensor.

Embedded system is a programmed controlling and operating system with real-time computing constraints. It also has an dedicated function within a larger mechanical system. Embedded systems are backbone of any robotic system as they are used to control sensors and actuators. They also communicate with higher levels such applications which can work with the robot itself. Modern embedded systems are usually based on microcontrollers but ordinary microprocessors are also commonly used. Figure 1.1 shows an embedded system example on a plug-in card. Microprocessor is an component that is used by a computer to do its work. Microcontrollers are used when something needs to be controlled and they are designed to perform specific tasks

Figure 1.1 Embedded System

1.1. Problem Statement

In this project, a robot car designed to navigate and find one of the predetermined objects. For robot part, Raspberry Pi 3 and Arduino used. Camera, distance sensor, motors and motor driver are connected to them. Robot navigates until its sensor detects an object in front of itself. Object is identified and compared to the ones it is looking for. If found correct object, photo of it sent to the user and alerted with an sound. If not found, robot turns to right or left and continues its way.

2. BACKGROUND

2.1. Related Works

In this century, there are lots of similar robots using the navigation and object detection technology.

2.1.1. Anki Vector

Vector is a toy home robot which is made to assist people in their homes. (Bohn) Vector spends most of its time investigating its surroundings. It can read a room. Vector uses an HD camera to see the area around itself. With the help of the camera it can identify people and objects. It can also navigate without bumping into things with the help of its sensors. It is also able to find its on charger by itself. It can also communicate with people around itself. Figure 2.1 shows a picture of it.

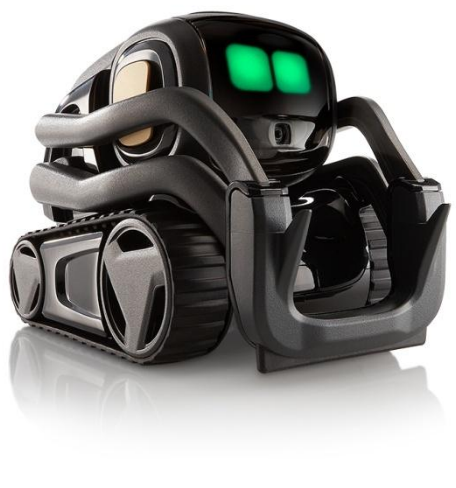


Figure 2.1 Anki Vector

2.1.2. Misty II

Misty is a personal programmable robot. It is very similar to Vector in terms of abilities and purpose. It is equipped with sensors and camera to assists people in their daily life. It can recognize people and objects. It can navigate without any problem. It can communicate with its user and since it is programmable it can do anything its user want it to do. Figure 2.2 shows a picture of it.

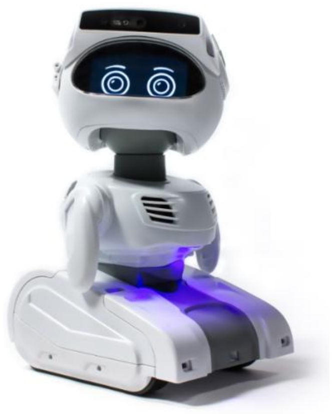


Figure 2.2 Misty II

2.1.3. IRobot Roomba 880

IRobot Roomba 880 is a robotic vacuum cleaner. It features multiple sensors so it can navigate around the house and clean it. With its sensors it can detect walls and obstacles so it doesn't bump into stuff or fall from stairs. This model also owns a camera to make navigation easier.

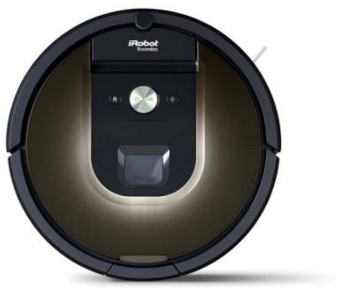


Figure 2.3 IRobot Roomba 880

3. ANALYSIS & DESIGN

This part will be about project requirements and the general design of the project. Requirements of this project are the followings:

1. Robot will navigate on its own.

2. Robot will detect objects and find out what they are.

3. Robot will send a picture to user when one of the predetermined object is found.

4. Robot will alert the user with a sound when object is found.

5. Robot will wait for start command from the user.

3.1. Overall Design

The purpose of this project is designing a robot car which navigates on its own and finds one of the predetermined objects. In Figure 3.1, system design can be seen. Robot part consists of Raspberry Pi (microcontroller), Arduino, motor driver, motors, camera and distance sensor. A PC is also needed for the companion application.

Robot navigates with the help of its sensor. Sensors calculate the distance between the car and the objects. When it is lower than a certain value it stops and checks the object in front of it. It uses camera to check it. If the object is not the one it is looking for it turns right or right and continues these steps until one of the objects is found. When it is found, car stops and send picture of the found object to users PC and alerts the user with an sound.

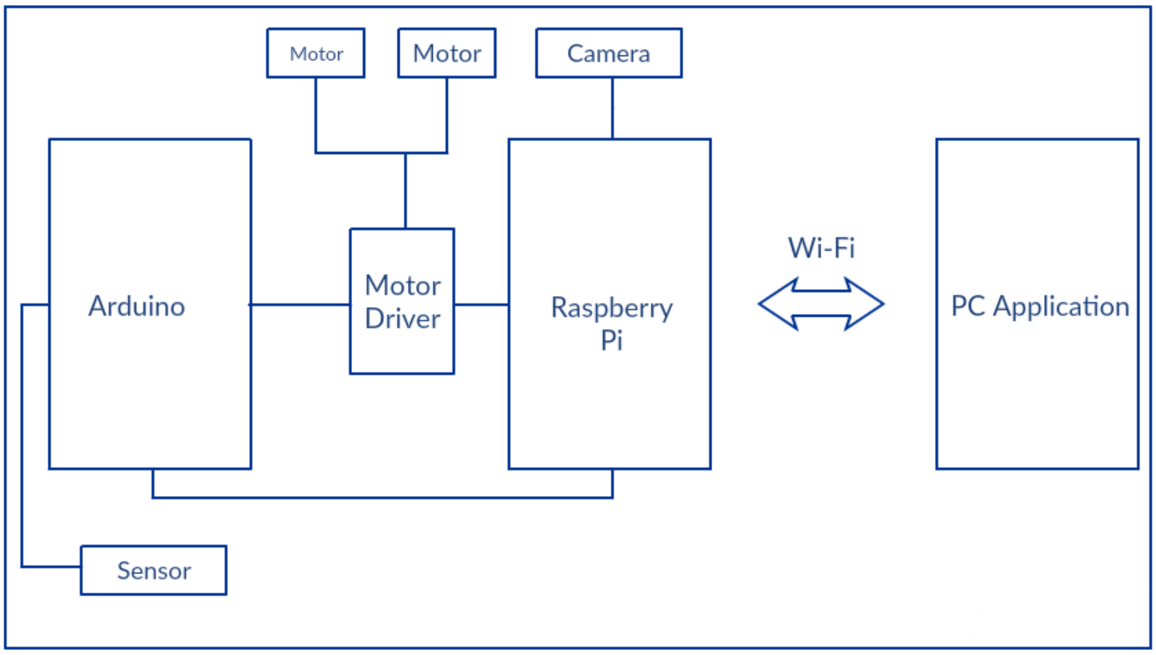


Figure 3.1 Overall System Design

3.2. System Requirements

Hardware requirements for this project are Raspberry Pi, Raspberry Pi Camera, HC-SR04 ultrasonic distance sensor, DC Motors, L293D Chip and Arduino Uno.

3.2.1. Raspberry Pi

Raspberry Pi is a small single-board computer. (Upton) It was developed by the Raspberry Pi Foundation. The idea to make this was to aid teaching students, beginner level computer science but when it became popular it end up being used for other uses, such as robotics. Raspberry Pi runs on its own operating system called Raspbian. Raspbian is actually Linux but it is made for Raspberry Pi.

There are multiple versions of the device and for this project I picked Raspberry Pi 3. This version is faster than its previous version and comes with an built in WIFI. We connect Raspberry Pi to WIFI and with the help of python sockets it can communicate the PC

Figure 3.2 Raspberry Pi 3

3.2.2. Raspberry Pi Camera

Raspberry Pi Camera is a camera designed to be attached to Raspberry Pi. (Upton) It's possible to take 3280 x 2464 pixels pictures and also supports multiple video resolutions. It connects to a small socket on top of Pi.

In this project Raspberry Pi Camera V2 is used. Pi camera is more compatible with Pi when compared to normal webcams. It is also in higher quality compared to previous Pi camera. Board is small and lightweight. This makes it perfect for this project since car needs to be mobile most of the time, less weight is important.

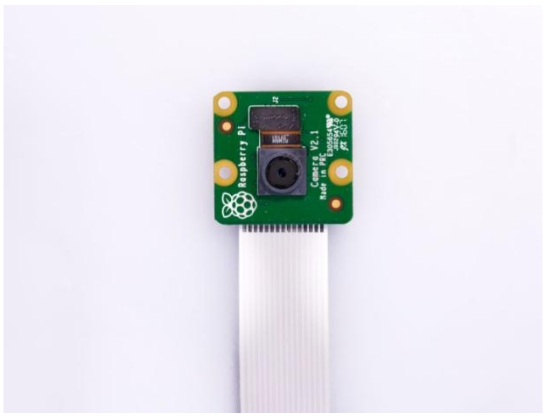


Figure 3.3 Raspberry Pi Camera V2

3.2.3. HC-SR04 Ultrasonic Distance Sensor

HC-SR04 sends 40 000 Hz ultrasound and this is used to detect objects in front of itself for this project. Ultrasound travels through the air and when it intersects with an object, it returns to the sensor. Distance can be calculated by travel time of ultrasound and speed of the sound.

HC-SR04 has VCC, Ground, Echo and Trig pins. VCC and Ground pins connected to 5V pin and ground pin of Raspberry Pi. Echo and Trig pins connected to Arduino Uno pins instead. Echo connected to 10 and Trig connected 9.

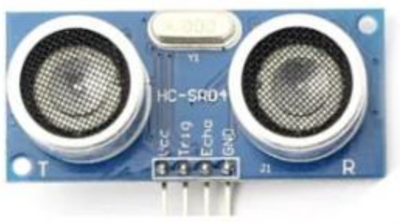


Figure 3.4 HC-SR04 Ultrasonic Distance Sensor

3.2.4. DC Motors

A DC motor is an rotary electrical machines which converts direct current electrical energy into mechanical energy. In this project it is used for navigation of wheels. There are two motors to move the car. DC motors are powered by 4 batteries, 6V total. DC motors are connected to pins of L293D Chip.

3.2.5. L293D Chip

L293D is an integrated circuit committed to controlling motors. (Badamasi) L293D has total of 16 pins, which mean it is possible to control two DC motors at the same time in any direction. The reason this motor driver is used because motors draws high amount of current. If connected directly to the circuit it may damage rest of the circuit because of its high current.

Figure 3.5 L293D Chip

3.2.6. Arduino Uno

Arduino is an open-source microcontroller board. (Kaura) The board is equipped with input and output pins. With these it can be connected other components such as microcontrollers and sensors. Arduino uses its own programming language and IDE. They can be used to set some instructions and make the board do what we want.

For this project Arduino Uno is selected. It can be seen in Figure 3.6. Its digital pins are connected to distance sensor and motor driver. By doing this, Arduino can control both of them. It can calculate the distance between sensor and an object. Then it can move the motors according to that. It also has a USB Port and it is connected to Raspberry Pi. This make Arduino and Raspberry communicate serially. Arduino is added to the project because it was hard for Raspberry Pi to handle all of the code by itself.

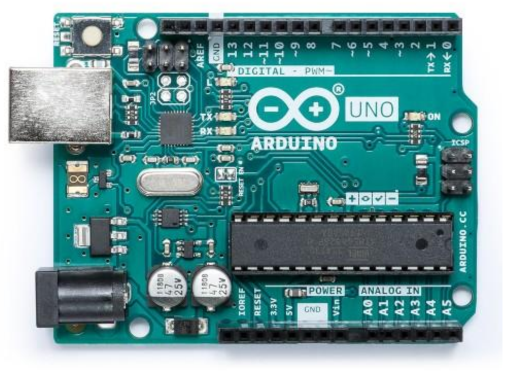


Figure 3.6 Arduino Uno

3.3. Pseudocode of Program

Algorithm 3.1 shows pseudocode of navigation part at Arduino. Arduino waits response from Raspberry Pi to start. If distance is not lower than 15 centimeter, robot moves forward. When it detects and object it communicates with Raspberry Pi. Depends on the response comes from Raspberry Pi, Arduino decides what to do.

Algorithm 3.1 Navigation in Arduino

```
WAIT response
CALCULATE distance
IF distance >= 15
    MOVE
ELSE
    STOP
    SEND message
    WAIT response
    IF response == f
        STOP
        WHILE 1
    IF response == n
        CALCULATE distance
        WHILE distance < 15
            SEND message
            WAIT response
            IF response == f
                STOP
                WHILE 1
            IF response == n
                STOP
                TURN
                WAIT 1500 milliseconds
                STOP
                CALCULATE distance
```

Algorithm 3.2 shows pseudocode of main in Raspberry Pi. It waits for start message from server to start.

Rest of the code runs when Arduino detects an object in front of itself. Depends on if object is the correct one, Raspberry Pi sends messages to other components.

Algorithm 3.2 Object Identification in Raspberry Pi

```
WHILE True
    IF x == 1
        GET response from server
        IF response == yes
            SEND message to Arduino
    WHILE 1
        GET message from Arduino
    IF message == 12
        CAPTURE image
        CREATE blob
        RUN Caffe model
        IF object is correct
            SEND message "f" to Arduino
            SEND photo to server
            CLOSE socket
        ELSE
            SEND message "n" to Arduino
```

3.4. Diagrams

3.4.1. Flowchart of Self Navigating Robot Car with Object Detection

Flowchart of Self Navigating Robot Car with Object Detection can be seen in the Figure 3.7 and Figure 3.8. First of all connections between server-client and Raspberry Pi - Arduino are done. After that navigation starts until robot detects an object in front on itself. This is handled on Arduino part. Object detection algorithm starts on Raspberry Pi. If correct object is not found, navigation continuous. If correct object is found, Raspberry Pi communicates with both PC and Arduino. Robot stops, PC receives an voice alert and a photo of the found object.

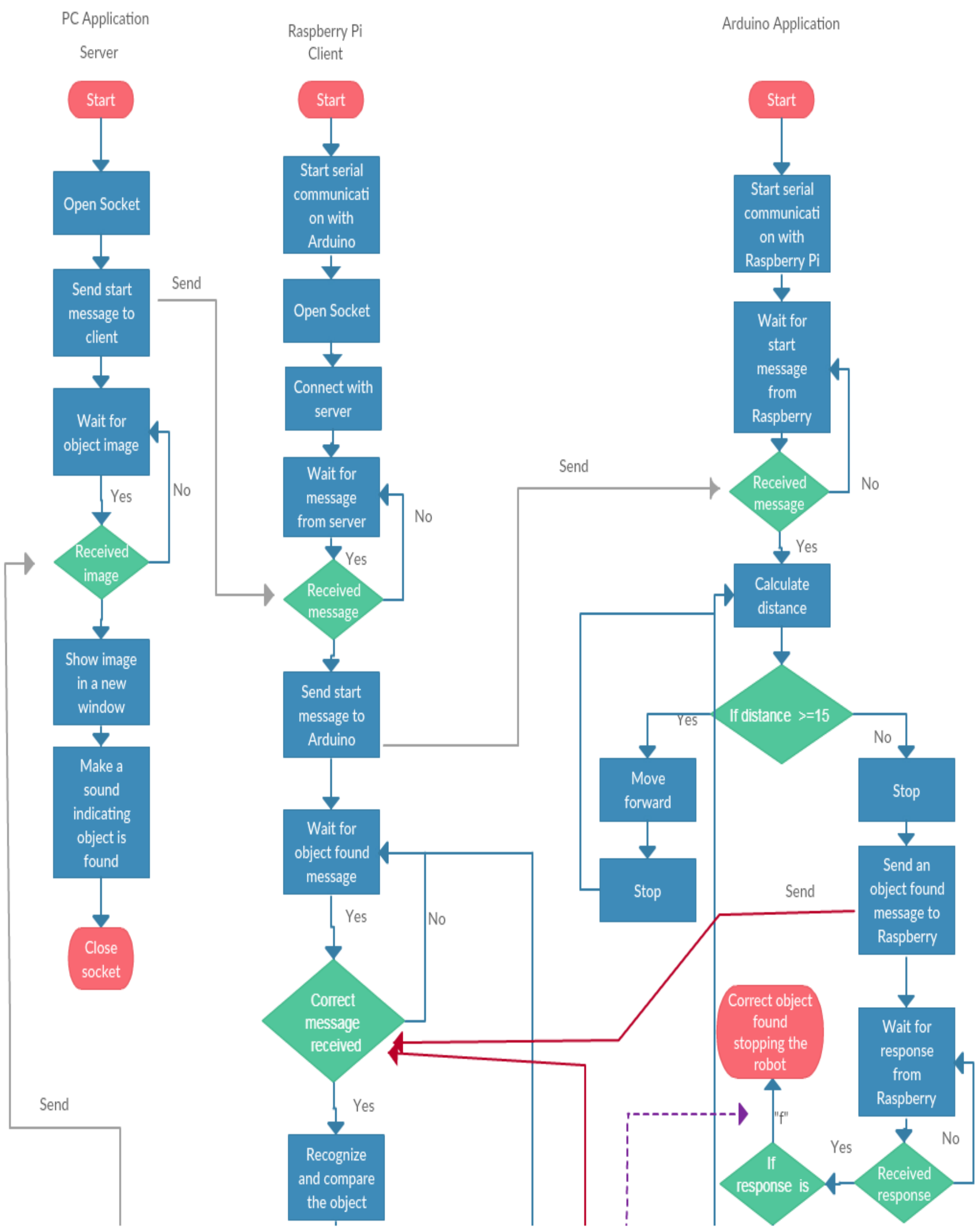
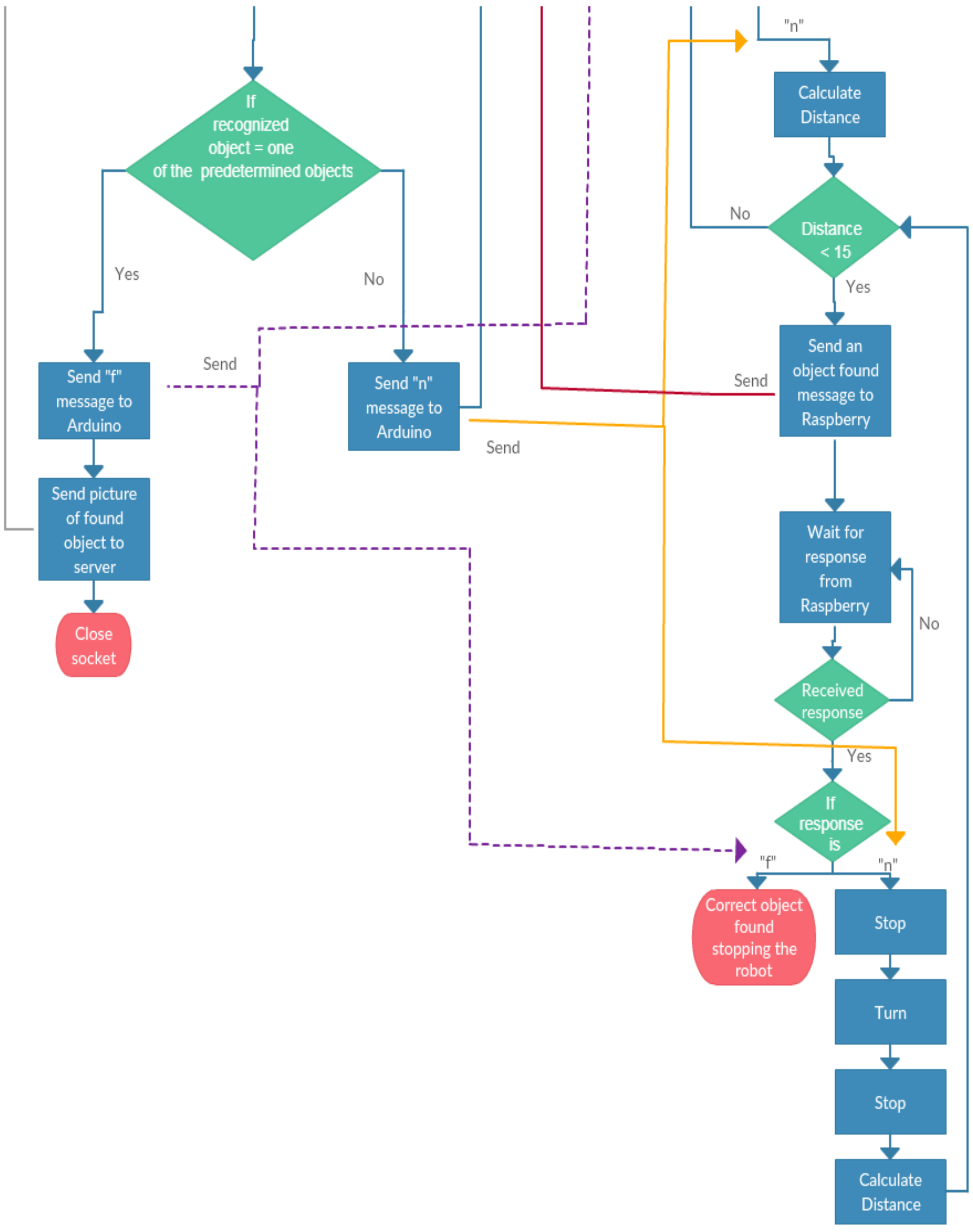

PC Application
Server
Start
Open Socket
Send start message to client
Send
Wait for object image
Yes
No
Received image
Show image in a new window
Make a sound indicating object is found
Close socket
Send
Raspberry Pi
Client
Start
Start serial communication with Arduino
Open Socket
Connect with server
Wait for message from server
No
Yes
Received message
Send start message to Arduino
Send
Wait for object found message
Yes
No
Correct message received
Yes
Recognize and compare the object
Arduino Application
Start
Start serial communication with Raspberry Pi
Wait for start message from Raspberry
Received message
No
Yes
Calculate distance
If distance >=15
Yes
No
Move forward
Stop
Stop
Send an object found message to Raspberry
Send
Wait for response from Raspberry
Received response
No
Yes
If response is
"f"
Correct object found stopping the robot

Figure 3.7 First part of flowchart of Self Navigating Robot Car with Object Detection



If recognized object = one of the predetermined objects
Yes
No
Send "f" message to Arduino
Send
Send "n" message to Arduino
Send
Send picture of found object to server
Close socket
"n"
Calculate Distance
No
Distance < 15
Yes
Send
Send an object found message to Raspberry
Wait for response from Raspberry
No
Received response
Yes
If response is
"f"
"n"
Correct object found stopping the robot
Stop
Turn
Stop
Calculate Distance

Figure 3.8 Second part of flowchart Self Navigating Robot Car with Object Detection

Use case diagram can be seen in the Figure 3.9. User starts the application. Robot starts to navigate and identify objects when detects one. If correct one is found application alert the user with sound and sends a picture of the found object.

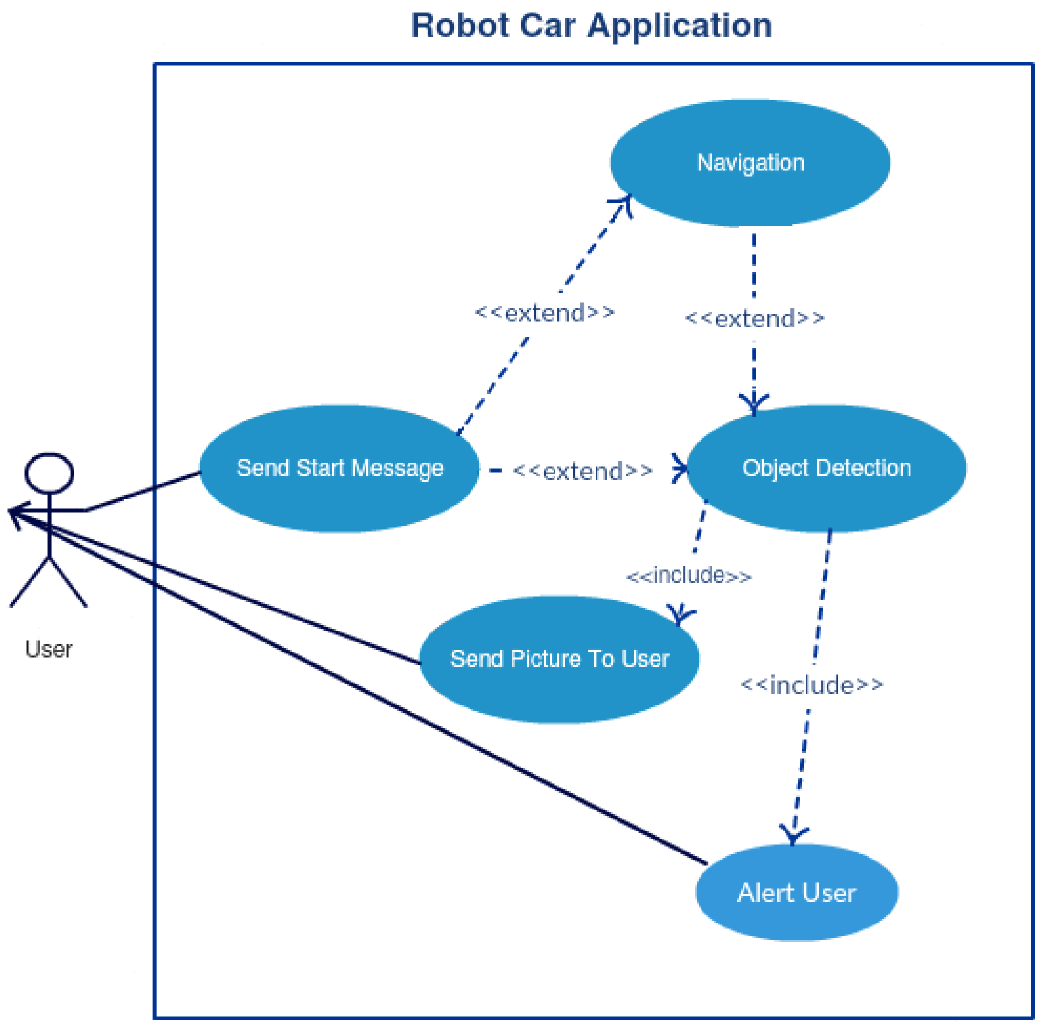


Figure 3.9 Diagram of Self Navigating Robot Car with Object Detection

4. IMPLEMENTATION

This section will be about implementation. There is going to be two part, hardware and then software. Software is for algorithms and code. Hardware part is for connections between components.

4.1. General System Design

Robot car mainly works on Raspberry Pi 3 and Arduino Uno. Both of them are microcontrollers connected to each other. PC works as an server and contains the companion application. Arduino are connected to motor driver which is needed for motors to work and distance sensor. Raspberry Pi on the other hand connected to camera and handles the object identification. When Arduino part detects an object, it communicates with Raspberry Pi part for identification. If correct object is found, user is alerted and a photo of object is sent to server by Raspberry Pi. Raspberry Pi is powered by a battery and rest of the components are also powered by the same source except the motors. Motors are powered by an four 1.5V batteries. In Figure 4.1 general system design can be seen. It is colour coded related which power source and which way in the circuit they are powered from.

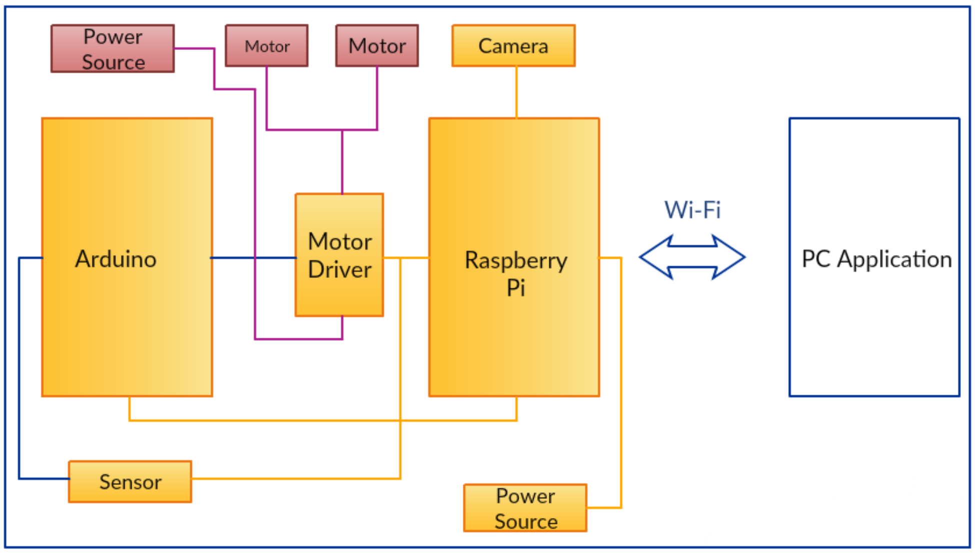


Figure 4.1 General System Design for Robot Car

4.2. Hardware System Design

In this project, used hardware parts are Raspberry Pi 3, HC-SR04 Ultrasonic Sensor, Arduino Uno, L293D and Pi Camera. In the following part, connections between components will be defined per component.

4.2.1. Raspberry Pi 3 Connections

Raspberry Pi uses two pins and two ports to connect to rest of the parts. Connection to camera can be seen in the Figure 4.2.

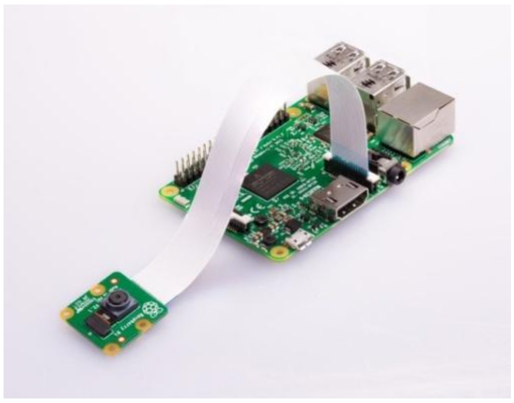


Figure 4.2 Raspberry Pi and Pi Camera Connection

Raspberry Pi needs to be connected to Arduino Uno with an USB cable so they can communicate serially. Rest of the connections are made by pin number 2 and 6. Pin 2 is 5V and pin 6 is Ground. Raspberry Pi also of course connected to an battery to power itself. In Figure 4.3 pinout of Raspberry Pi can be seen.

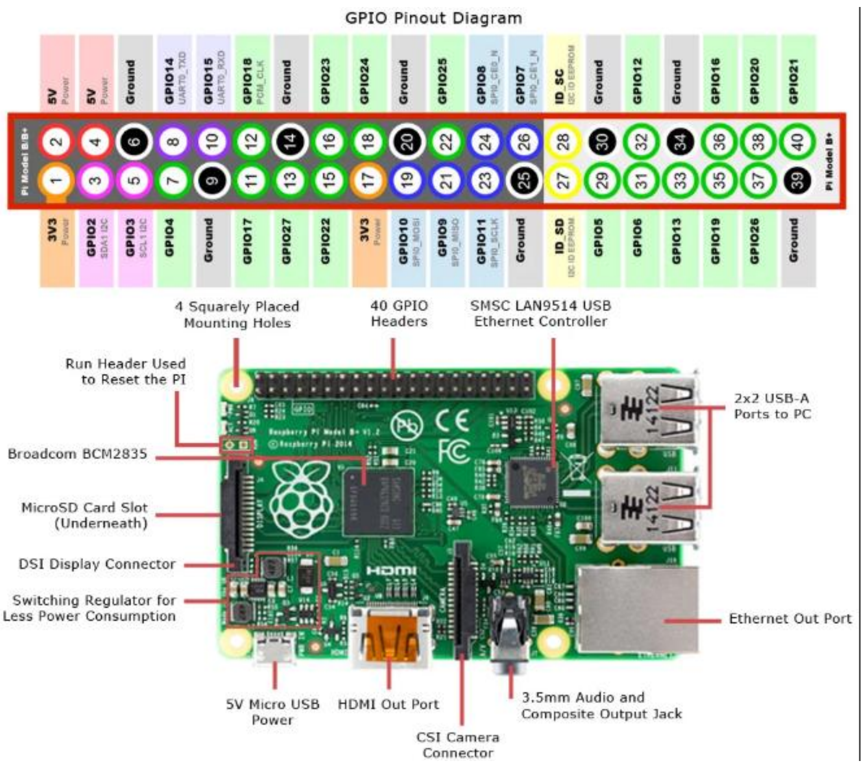


Figure 4.3 Raspberry Pi 3 Pinout

4.2.2. HC-SR04 Ultrasonic Distance Sensor Connections

Arduino Uno controls the navigation so TRIG and ECHO pins connected to Arduino. VCC and GND pins on the other hand connected to Raspberry Pi since it powers the sensor. Connections can be seen in the Table 4.1. and Figure 4.4.

Table 4.1 HC-SR04 Ultrasonic Distance Sensor Connections

Sensor Pin Names	Component Names	Pin of Components
VCC	Raspberry Pi	2
TRIG	Arduino Uno	9
ECHO	Arduino Uno	10
GND	Raspberry Pi	6

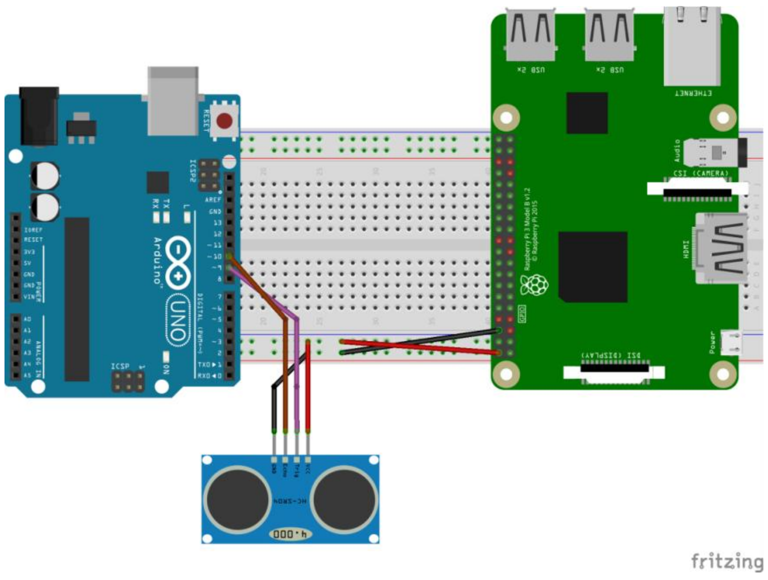


Figure 4.4 HC-SR04 Ultrasonic Sensor Distance Sensor Connections

4.2.3. Arduino Uno Connections

Arduino is the one responsible for navigation of the device so Pin 10 and Pin 9 connected to ECHO and TRIG pins of HC-SR04 Distance Sensor. Apart from the sensor, it also needs access to motors for navigation. Pin 4, 5, 6 and 7 connected to input pins 7, 2, 10, 15 of L293D Motor Driver. Arduino also connected to Raspberry Pi with an USB cable for serial communication. Connections excluding the USB cable can be seen in the Table 4.2 and Figure 4.5.

Table 4.2 Arduino Uno Connections

Arduino Uno Pins	Component Names	Pin of Components
10	HC-SR04 Distance Sensor	ECHO
9	HC-SR04 Distance Sensor	TRIG
7	L293D	15(IN4)
6	L293D	10(IN3)
5	L293D	2(IN1)
4	L293D	7(IN2)

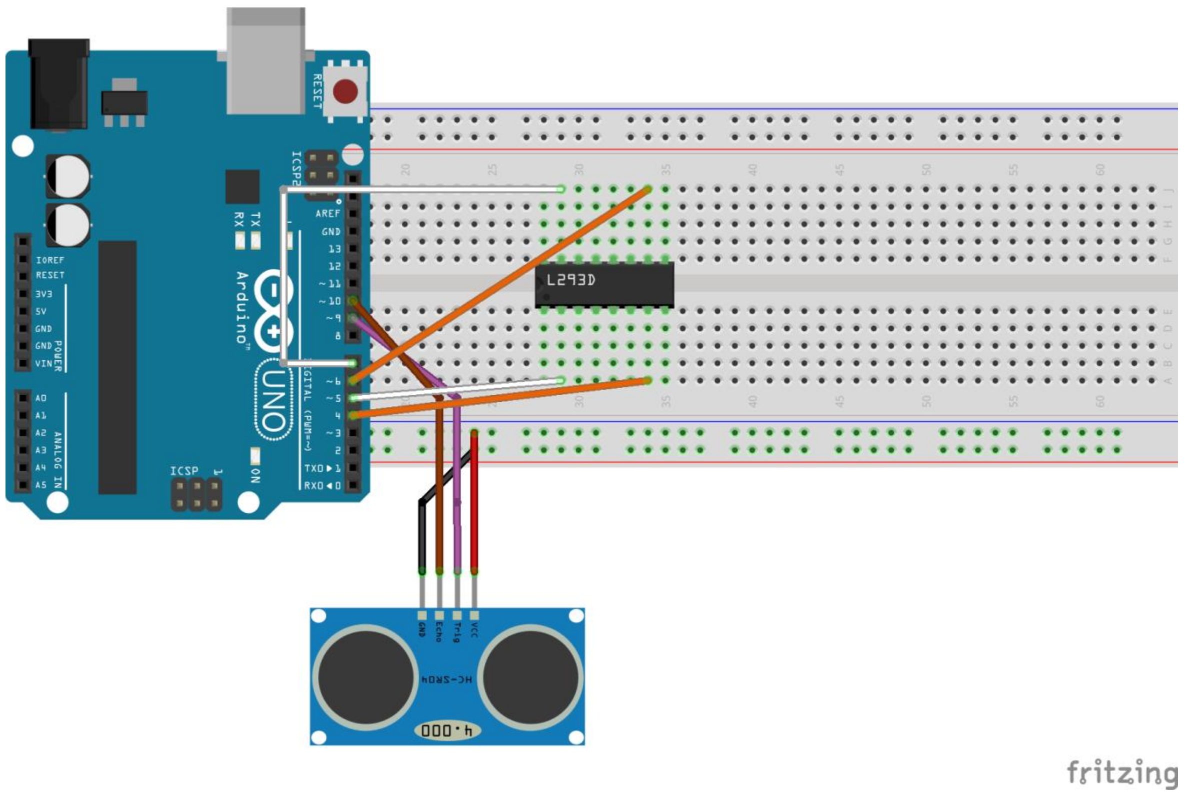


Figure 4.5 Arduino Uno Connections (excluding USB connection)

4.2.4. L293D Connections

As seen in Table 4.3, L293D connects to most of the components. OUT Pins are connected to both of the motors. IN pins are connected to Arduino so it can control the motors. Pin 8, the VCC2 connected to batteries. Power to move the motors come from these batteries. Half of the GND Pins connected to GND of batteries, other half connected to GND of Raspberry Pi because they are connected to the one that is closest. Pin 16, VCC1 is the one that powers the L293D. It should be connected to 5V source so it takes power from Raspberry Pi. When EN1 and EN2 is HIGH left and right part works so EN1 is connected to VCC Pin of Raspberry Pi. EN2 connected to EN1 so indirectly that is connected to the Raspberry Pi too. In Figure 4.6, wired connections of all of the system, including the L293D can be seen.

Table 4.3 L293D Connections

L293D Pins	Component Names	Pin of Components
1(EN1)	Raspberry Pi - L293D	2 - 9
2(IN1)	Arduino Uno	5
3(OUT1)	Motor 1	-
4(GND)	Raspberry Pi	6 (GND)
5(GND)	Raspberry Pi	6 (GND)
6(OUT2)	Motor 1	-
7(IN2)	Arduino Uno	4
8(VCC2)	Battery	VCC
9(EN2)	L293D	1
10(IN3)	Arduino Uno	6
11(OUT3)	Motor 2	-
12(GND)	Battery	GND
13(GND)	Battery	GND
14(OUT4)	Motor 2	-
15(IN4)	Arduino Uno	7
16(VCC1)	Raspberry Pi	2

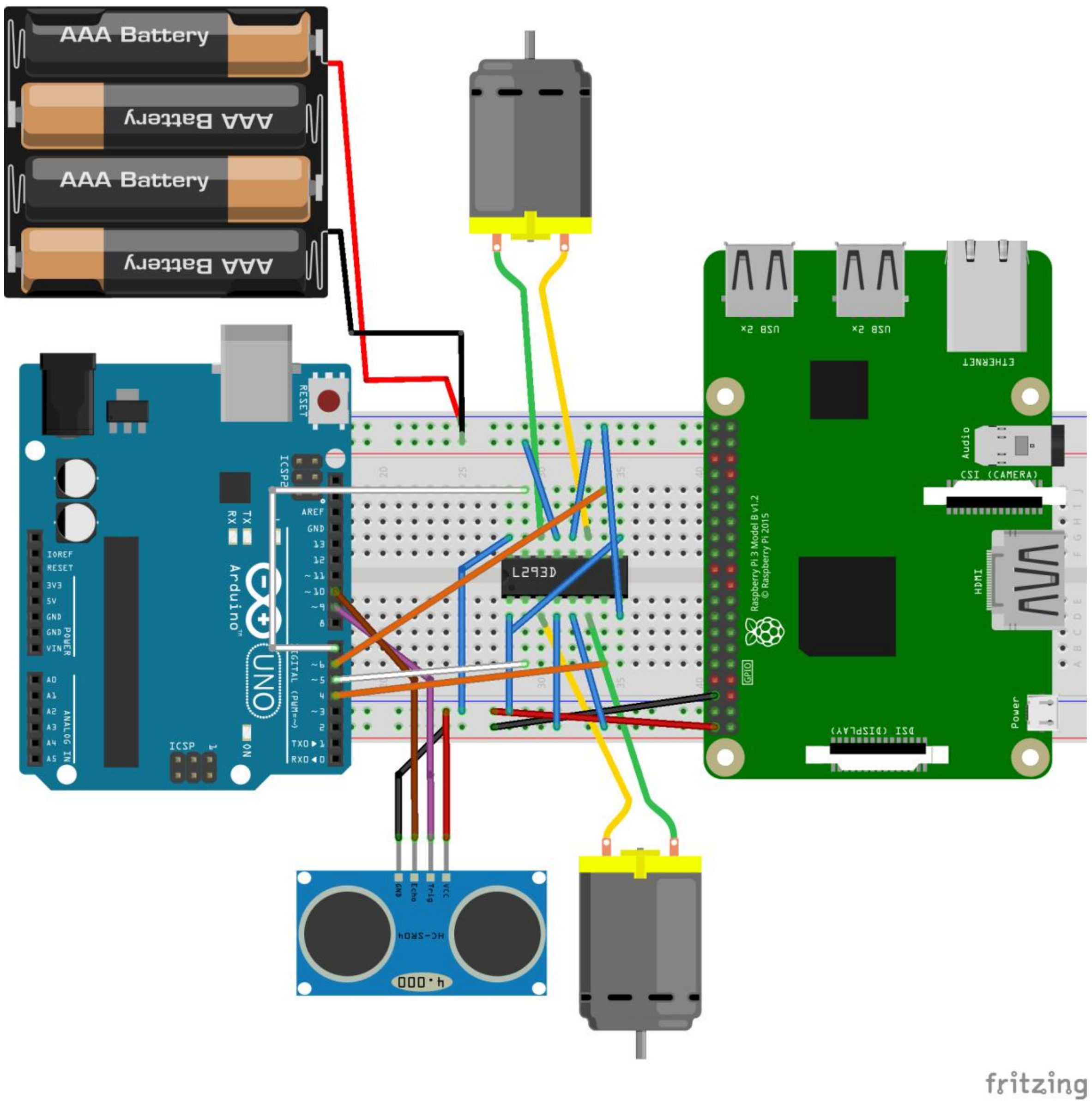


Figure 4.6 Connections of All System

4.3. Software Implementation Design

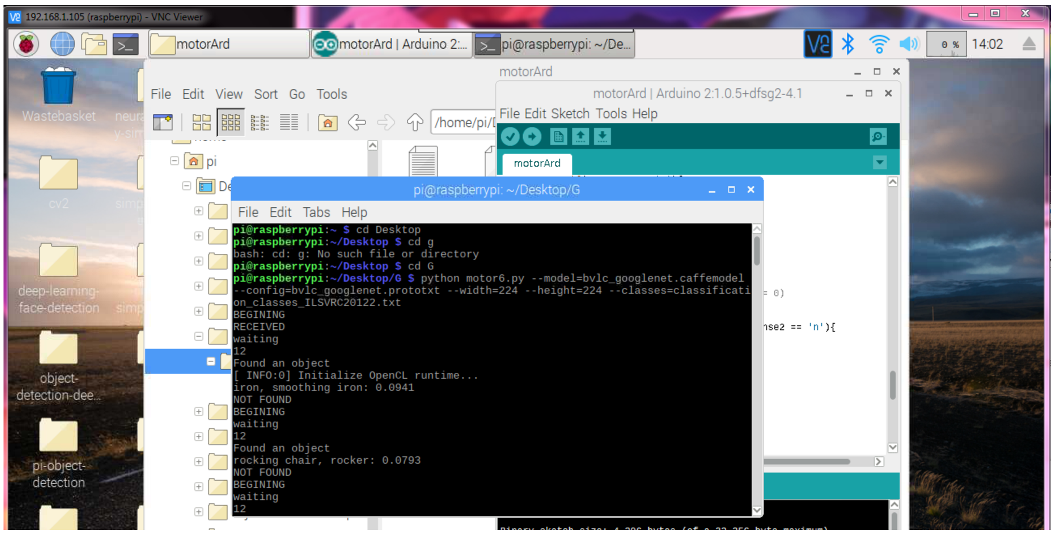
In this section, software on Raspberry Pi, Arduino and PC will be defined.

4.3.1. Python

Python is an interpreted, high-level, general-purpose programming language which is made by Guido van Rossum and it is released in 1991. (Van Rossum) Python is easy to learn and use. It emphasizes one code readability, especially by using whitespace. Python used for both code in Raspberry Pi and PC side. The main reason was to be able to use python sockets.

4.3.2. VNC Viewer

Raspberry Pi does not have a screen so a way to view it was needed. VNC Viewer works on a client-server model. It is installed on the local computer and connects to the server component on a remote computer. Server than sends a duplicate of the remote computer's display screen to the viewer. This how Raspberry Pi was viewed in this project.


192.168.1.105 (raspberrypi) - VNC Viewer
motorArd
motorArd | Arduino 2:...
pi@raspberrypi: ~/De...
14:02
Wastebasket
cv2
deep-learning-face-detection
object-detection-dee...
pi-object-detection
File Edit View Sort Go Tools
/home/pi/
pi
motorArd | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
motorArd
pi@raspberrypi: ~/Desktop/G
File Edit Tabs Help

```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ cd g
bash: cd: g: No such file or directory
pi@raspberrypi:~/Desktop $ cd G
pi@raspberrypi:~/Desktop/G $ python motor6.py --model=bvlc_googlenet.caffemodel
--config=bvlc_googlenet.prototxt --width=224 --height=224 --classes=classificati
on_classes_ILSVRC20122.txt
BEGINING
RECEIVED
waiting
12
Found an object
[ INFO:0] Initialize OpenCL runtime...
iron, smoothing iron: 0.0941
NOT FOUND
BEGINING
waiting
12
Found an object
rocking chair, rocker: 0.0793
NOT FOUND
BEGINING
waiting
12
```

Figure 4.7 VNC Viewer

4.3.3. PC-Raspberry Pi 3 Socket Connection

In this project, socket programming is used to communicate between Raspberry Pi and PC. Sockets are the endpoints of a two-way communications links between two programs running on a network. (Lutz) They have port numbers so TCP layer can understand where is that data supposed to go. Sockets are able to communicate within a process, between processes on the same machine, or between processes on different continents. Sockets are used in client-server models. Figure 4.8 shows the client-server model. In this model, server listens to the socket and waits for an connection request from the client. When a client sends a request and gets accepted, it creates a socket. After successfully creating that socket, now both sides can communicate both ways. For this project PC is an server and Raspberry is the client side.

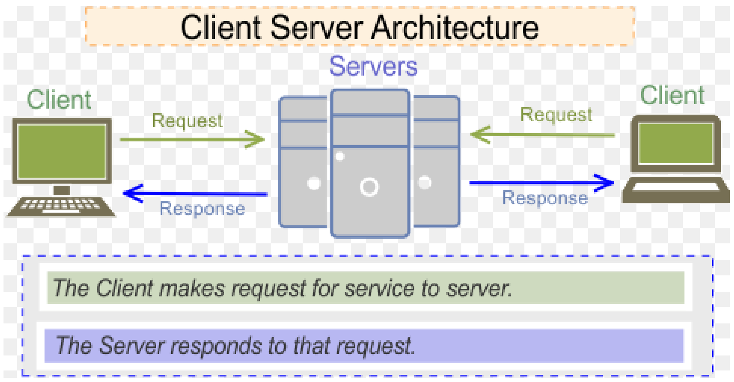

Client Server Architecture
Servers
Client
Request
Response
Client
Request
Response
The Client makes request for service to server.
The Server responds to that request.

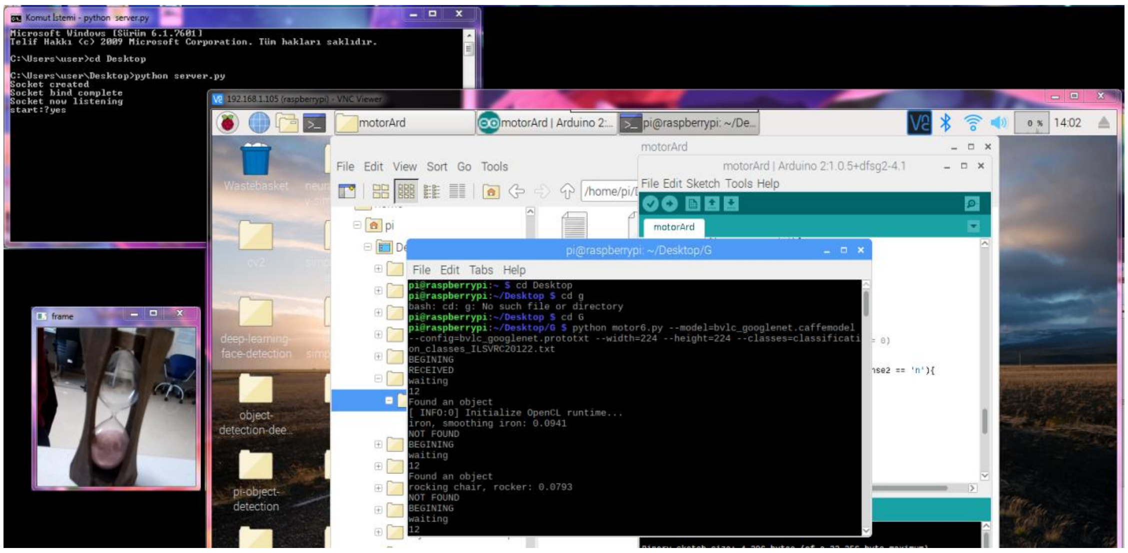
Figure 4.8 Client-Server model

On the server, first step is to define port and host. It is important to define theme correctly or communication won't work. Then socket is created with the *socket()* function. After that socket is bind with *bind()* function. Next step is accepting connections with the *accept()* function. At this step server waits until a client connects. (Algorithm 4.1) After client connects, server sends a message with *sendmsg()* function, giving permission for Raspberry Pi and Arduino to do their work. This function asks for an input and sends it with *sendall()*. Client doesn't sent anything again until robot finds the correct object. That time client sends a picture of found object in numpy array. Server unpacks it and displays the image. It also gives an voice alert indicating object is found. After completing these actions, server closes its socket.

Algorithm 4.1 The Socket Connection Between Raspberry Pi Server and PC Server

```
SET HOST '192.168.1.102'
SET PORT 8080
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
conn, addr = s.accept()
```

On the client side, a socket is created with the port and IP numbers. After connecting to the server, client wait for the start message. It receives it with *recv()* function. They don't interact again until object is found. Client sends image with *sendall()* function. At the end it also closes its socket. In Figure 4.9 a successfully completed program can be seen. Picture arrived without an problem.



```
Microsoft Windows [Sürüm 6.1.7601]
Telif Hakkı (c) 2009 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\user>cd Desktop

C:\Users\user\Desktop>python server.py
Socket created
Socket bind complete
Socket now listening
start:?yes
```

```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ cd g
bash: cd: g: No such file or directory
pi@raspberrypi:~/Desktop $ cd G
pi@raspberrypi:~/Desktop/G $ python motor6.py --model=bvlc_googlenet.caffemodel
--config=bvlc_googlenet.prototxt --width=224 --height=224 --classes=classificati
on_classes_ILSVRC20122.txt
BEGINING
RECEIVED
waiting
12
Found an object
[ INFO:0] Initialize OpenCL runtime...
iron, smoothing iron: 0.0941
NOT FOUND
BEGINING
waiting
12
Found an object
rocking chair, rocker: 0.0793
NOT FOUND
BEGINING
waiting
12
```

Komut İstemi - python server.py
192.168.1.105 (raspberrypi) - VNC Viewer
motorArd
motorArd | Arduino 2:...
pi@raspberrypi: ~/De...
14:02
Wastebasket
cv2
deep-learning-face-detection
object-detection-dee
pi-object-detection
File Edit View Sort Go Tools
/home/pi/
pi
motorArd | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
pi@raspberrypi: ~/Desktop/G
File Edit Tabs Help
frame

Figure 4.9 Successfully Completed Program Output

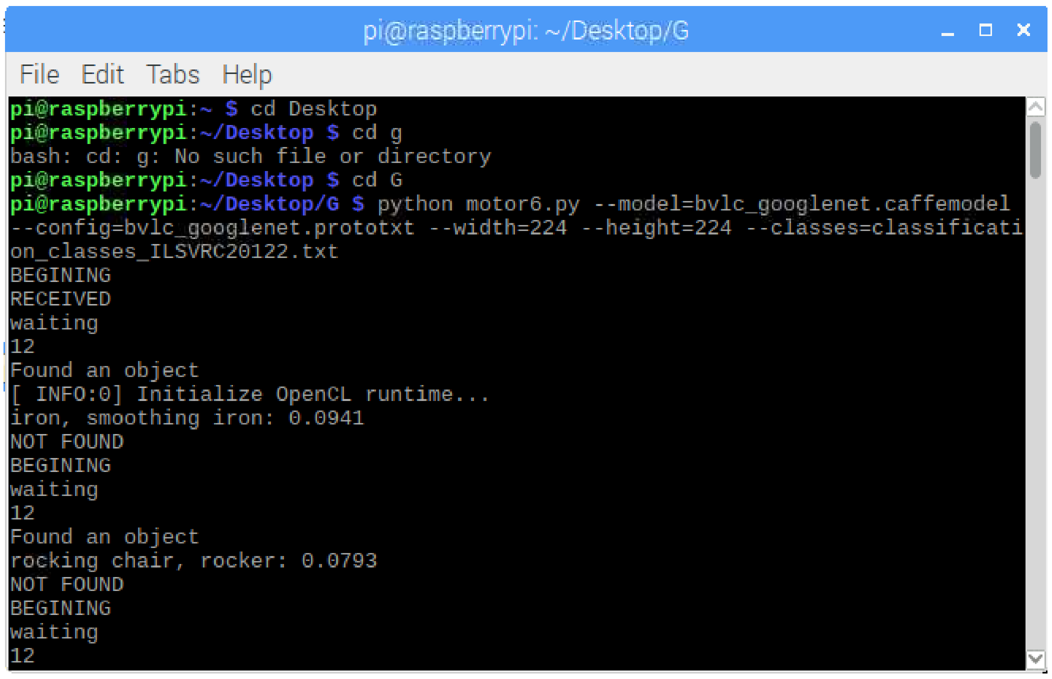
4.3.4. Raspberry Pi 3 Implementation

For this part, OpenCV v3 is installed on the Raspberry Pi 3 so it can do the object detection. OpenCV is an open source computer vision and machine learning software library which uses computer vision and machine learning to identify objects. (Bradski) OpenCV originally written in C++ but later developed for other languages like Python. OpenCV supports multiple deep learning frameworks. In this project Caffe model is picked.

Caffe model is a open source deep learning framework developed by Berkeley University. (Jia) It is written in C++ but with an Python interface. This framework supports lots of different kind of deep learning architectures aimed towards image segmentation and classification.

In object classification, every object class has its own unique features which helps the program to classify the class. All circles are round can be given as an example. These features are used for identification. For example if program is looking for squares, it looks for objects which are perpendicular at corners and have equal side lengths.

In the project, when the program wants to identify an object, it first of all takes a picture of said object with *capture()* function. 4D blob is created with *blobFromImage()* function from that frame then the Caffe model runs. It returns the class with the highest score. Before sending the object found information, Raspberry prints the class Id and confidence score (Figure 4.10). Confidence score is like an probability of accuracy of that identification. It ranges from 0 to 1. 1 is the highest score and this means algorithm is 100% sure that it identified the object correctly.



```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ cd g
bash: cd: g: No such file or directory
pi@raspberrypi:~/Desktop $ cd G
pi@raspberrypi:~/Desktop/G $ python motor6.py --model=bvlc_googlenet.caffemodel
--config=bvlc_googlenet.prototxt --width=224 --height=224 --classes=classificati
on_classes_ILSVRC20122.txt
BEGINING
RECEIVED
waiting
12
Found an object
[ INFO:0] Initialize OpenCL runtime...
iron, smoothing iron: 0.0941
NOT FOUND
BEGINING
waiting
12
Found an object
rocking chair, rocker: 0.0793
NOT FOUND
BEGINING
waiting
12
```

Figure 4.10 Object Id and Confidence Score

Algorithm 4.2 Initialize the Pi Camera and Identify the Object

```
camera = PiCamera()
SET capture height 224
SET capture width 224
DO capture the frame
DO frame to blob
DO Caffe model
```

4.3.5. Arduino Uno Implementation

Arduino Uno uses its own language but the language itself is just set of C/C++ functions. (About) A program is called a sketch in Arduino. In every sketch there needs to be setup and loop functions. At the beginning setup function is called. It is called only once. After setup function, loop function starts running continuously.

First of all, pin numbers and variables are defined. Both motor and distance sensor are connected to Arduino. In *setup()* function all of the pins are set as outputs and inputs. It then starts serial communication. After that *loop()* function starts to run. In that function it waits until a response from Raspberry Pi comes. When response comes it continues running and *calculatedistance()* runs to calculate the distance. It can be seen in Algorithm 4.1. In *calculatedistance()* function, it cleans the TRIG pin by setting it to LOW so it can return more accurate number. Later TRIG is set to HIGH for 10 micro seconds to be set to LOW again after stated duration. After that ECHO pin is read which returns the travel time of the sound wave in microseconds which is called an pulse with the *pulseIn().* (Language Reference) HIGH parameter in this functions means it will wait for pin to go HIGH then wait for it to go to LOW. This is how time is calculated. This also causes the number to be doubled because it also counts the time to both travel and bounce back to sensor so after multiplying with the speed of sound which is 0.034, number is divided into 2. Functions returns the distance.

Algorithm 4.3 Calculate Distance

```
SET trigPin to LOW
WAIT 2 microseconds
SET trigPin to HIGH
WAIT 10 microseconds
duration = pulseIn(echoPin , HIGH)
distance = duration *0.034/2
```

Back at the *loop()* function it checks the returned distance value. If it bigger or equal than 15 it goes into *move()* function. In the *move()* function it moves the wheels for 200 milliseconds by setting them HIGH and then stops. It is also checks the distance and stops if it is smaller than 15.

Back at the *loop()* function if distance is smaller than 15, it starts the *stop()* function. This function just stops the motors by setting them as LOW. After stopping Arduino sends an message to Raspberry Pi indicating that it found an object. It waits until any kind of responds returns. If the object is one of the predetermined objects it stops the motors. If not it calculates the distance again. As long as the distance is lower than 15 it sends a message again and waits until a response comes. If correct object is found it stops the motors. If not it continuously turns and check the distance in front of itself. When distance is bigger than 15 it breaks out of the while loop and moves. In the *turn()* function, it stops the motors and decides on a random number between 1 and 10. Depends if it is odd or even, wheels return to right or left. They turn for 600 milliseconds.

4.4. Appearance of Robot Car

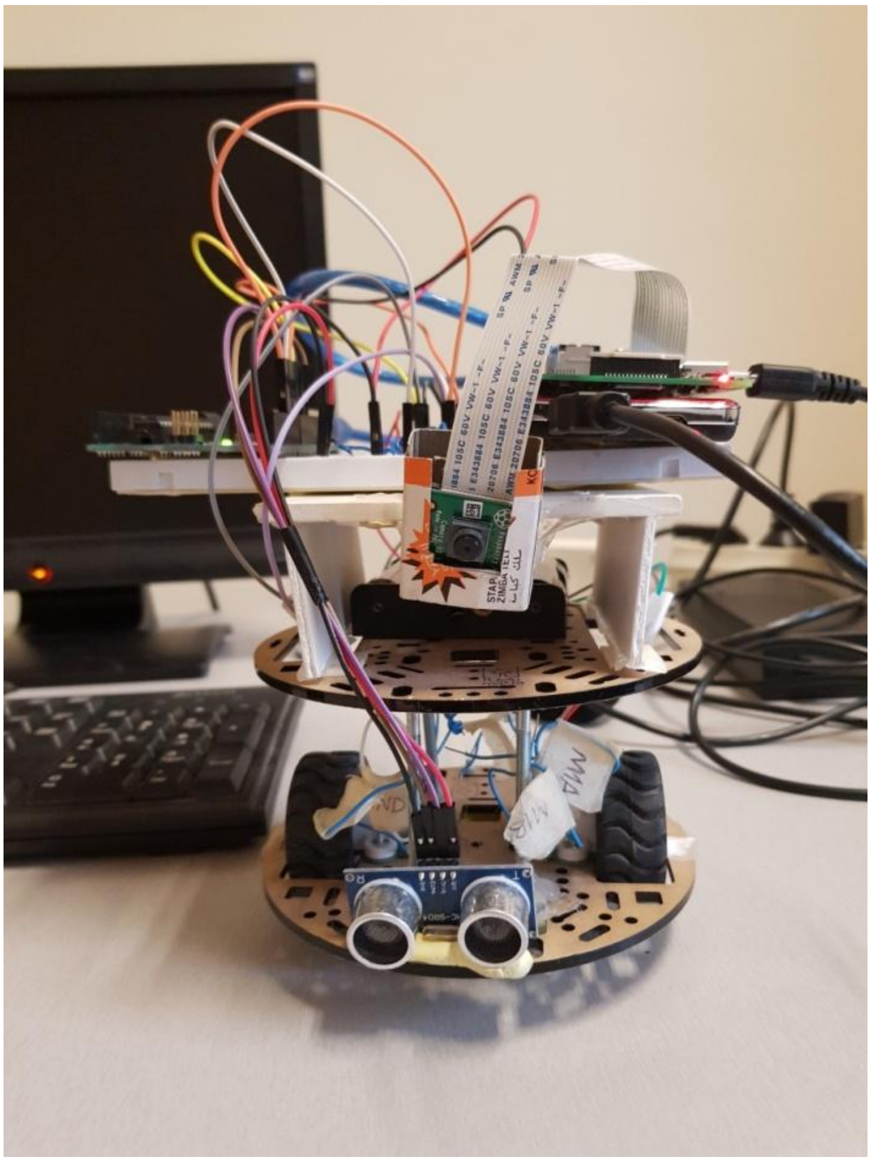


Figure 4.11 Picture From Front of the Robot

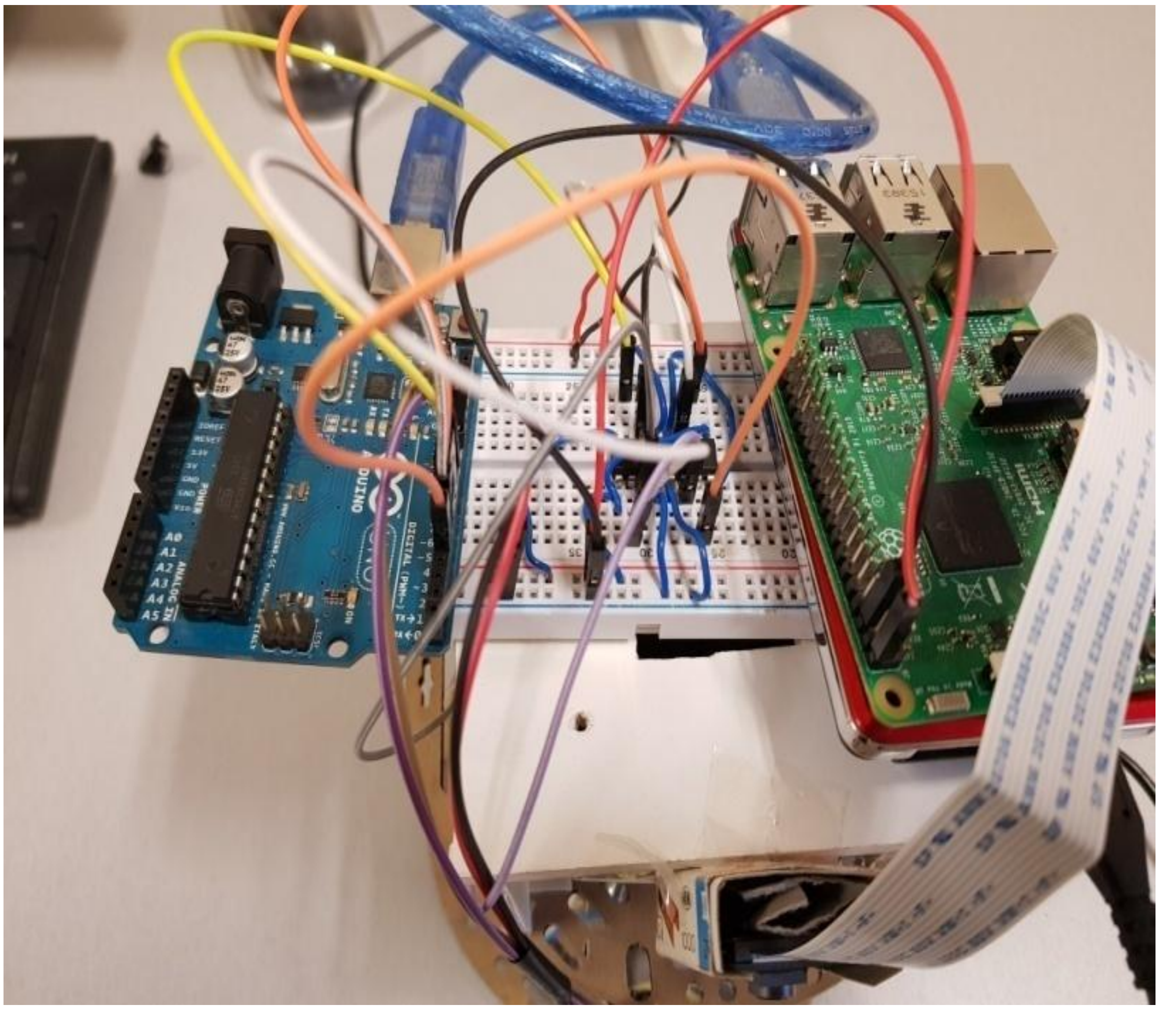


Figure 4.12 Picture from Above of the Robot

5. TESTS & RESULTS

The height of the object and the distance of the object to the sensor and the camera affects how robot works. Robot needs to find and detect objects correctly This is an crucial part of this project so factors that it gets affected by are tested.

5.1. Detection Confidence with Different Heights

Detection accuracy is most important part of this project. Robot needs to detect objects correctly to work. All three of the predetermined objects put 10 centimeter distance to the camera and tested multiple times to see their confidence numbers. Confidence number means how much confident is the application that it detected the object correctly. Camera is 12 centimeter high from the ground. Height of the objects are 20.4 centimeter for water bottle, 13.2 centimeter for hourglass and 3 centimeter for mouse. Figure 5.1 shows the detection accuracy of these objects at 10 centimeter distance. There isn't much difference but hourglass is slightly more accurate than other two. The possible reason for this is height of hourglass is closer to height of the camera.

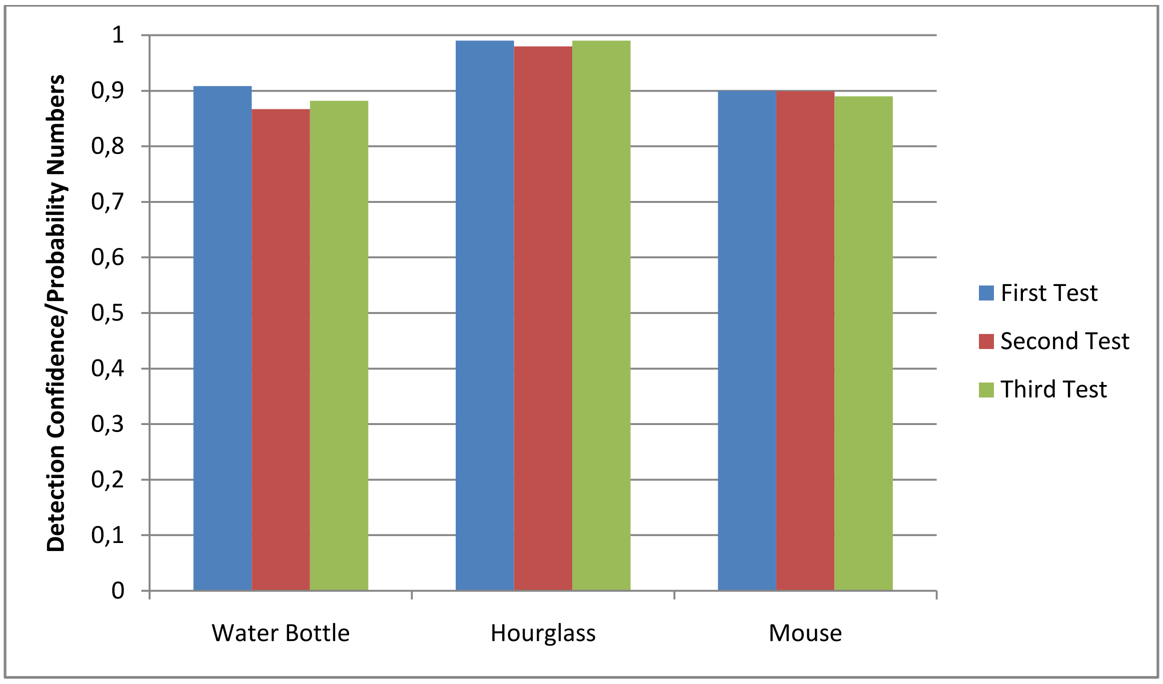


Figure 5.1 Detection Confidence of Objects at 10 centimeter distance

5.2. Distance Error Rate

Distance sensor is the most import part of how robot navigates. Before adding this sensor to the project. It is ability to calculate distance is calculated. Three of the predetermined objects and a box used for this test.

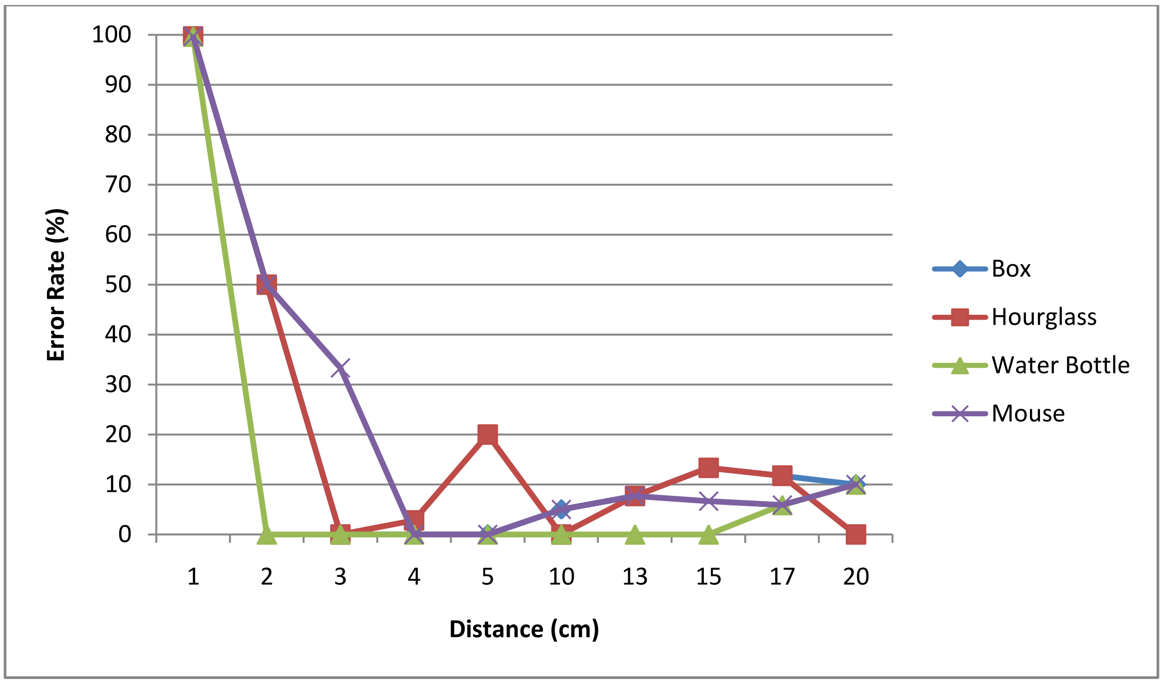


Figure 5.2 Distance Error Rate

5.3. Effects of Different Distances on Detection Confidence

Distance of the objects to the camera is also an factor which effects the detection confidence. All three of the predetermined objects put in front of the camera at different distances and tested multiple times. Camera is 12 centimeter high from the ground and height of the objects are still same. Figure 5.3 shows the detection confidence of these objects at different distances. At 3 centimeter and lower none of the objects are detected correctly at all.

Hourglass can be detected at more ranges than rest of the objects, between 4 to 20 centimeters. Furthermore, confidence is higher than rest of the objects too. Hourglass being similar in height to height of the camera shows its effect more here but after 15 centimeters, confidence starts to decrease for hourglass too. Water bottle and mouse shows similar amount of range and confidence level. Water bottle ranges from 5 to 15 and mouse ranges from 10 to 20 but overall its range is 10 centimeter. It is lower than 16 centimeter range of hourglass. This was an expected result considering their heights. Mouse needs to be farther away from the camera to be detected. Lower than 10 centimeter, object doesn't get detected correctly at all. This is opposite for water bottle. It needs to be closer to the camera to detected. Higher than 15 centimeter and it doesn't get detected too.

As an conclusion, it is certain that every object have their best distance to be detected with an high confidence. When that object is moved away from that distance, confidence starts to decrease. It can decrease enough to not detect the object correctly at all. Height of the object affects the optimal distance. Taller objects needs to be closer and smaller objects needs to be farther away to get to that optimal distance. Object at similar height to the camera doesn't get affected by this but after a while, as the object moves away, its confidence level start to decrease too. In this project optimal distance to detect object seems to be 10 centimeters. All three can be detected at that distance.

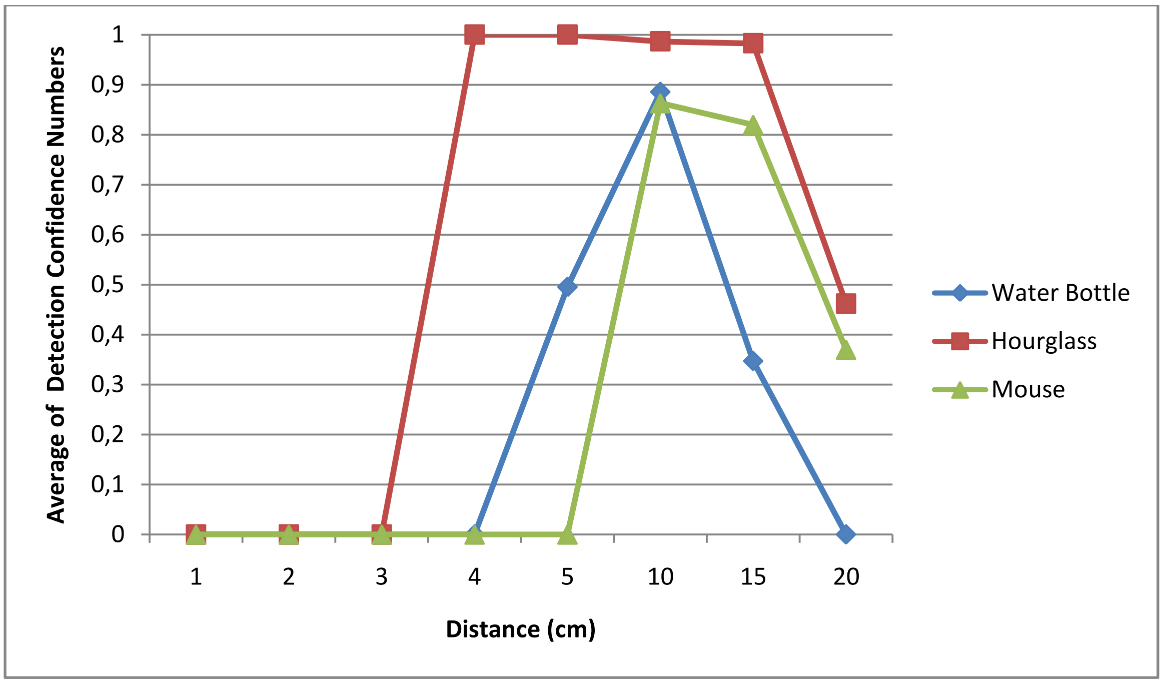


Figure 5.3 Average of Detection Confidence of Objects at Multiple Distances

5.4. Factors Effecting Time to Detect an Object and Move on

Robot car is an real time device so it needs to detect quickly and move on to next step as soon as possible. To test this an object which is not one of the predetermined objects, put in front of the robot. Robot is immobilized by putting an platform under it but wheels can still turn. Time to detect the object and the time for wheels to turn again calculated. Figure 5.4 shows the results. According to the graphic detection is short but despite that total time to move on to next step is almost three times more. This means detection is quick but transfer of this data between Raspberry Pi and Arduino is slow. The reason for this might just being a slow connection between the two. Arduino programmed to not make motors move until data from Raspberry Pi arrived correctly.

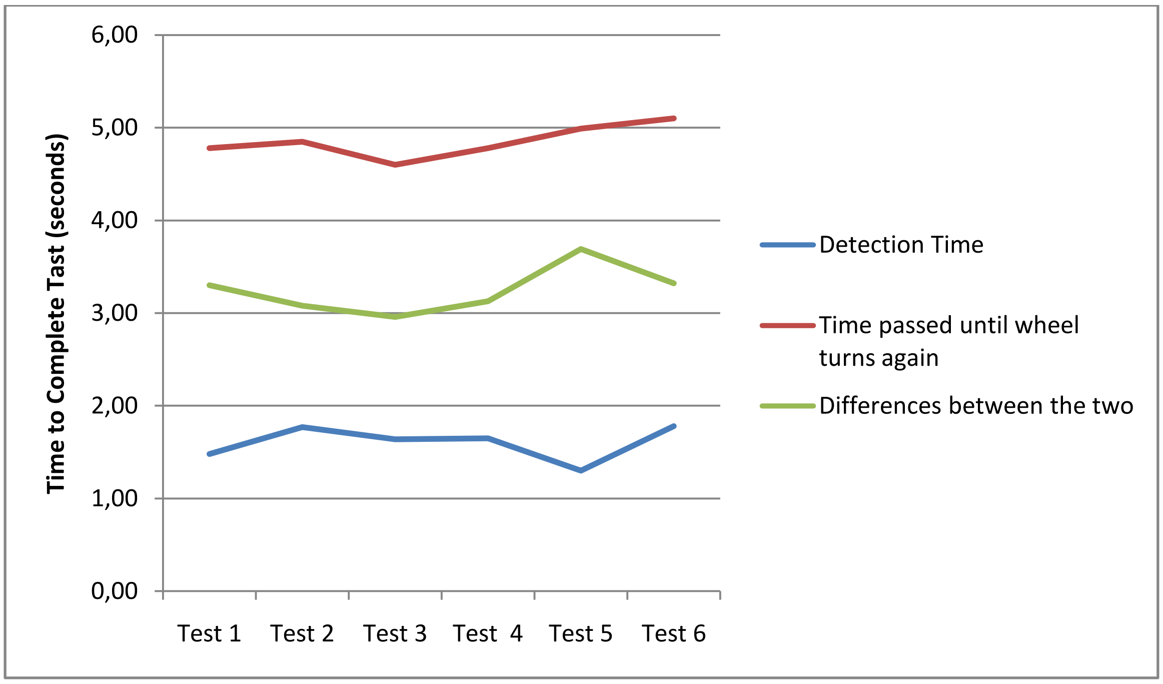


Figure 5.4 Time to Detect Objects and Time Between Wheel Turns

6. CONCLUSION

Useful products with the ability of searching and finding objects have been produced and continue to be produced. This project was designed to make easy the lives of people which can't or won't do this manually for reasons like safety or lack of time. For this project, Raspberry Pi and Arduino used for the robot car. A PC is needed to interact with the robot. On the Raspberry Pi, a Raspberry Camera was used to identify objects. Arduino on the other hand, handles navigation with distance sensors to detect objects on its path. PC works as an server and in the end, a photo of found object reaches it and alerts the user.

To conclude, Self Navigation Robot Car with Object Detection succeeded in its purpose. It can search and identify objects without a problem. This project can be improved with some changes for the future works. Adding a better power source can make the robot last longer. Improving distance sensors for more correct distance calculation and more having range. Improving quality of camera returns more clear picture making it to easier to detect. Overall, robot body can be made and covered with stronger materials. Making robot more durable, gives it a better chance at surviving an unsafe environment.

Bibliography

1. Bohn, Dieter. “The New Anki Vector Robot Is Smart Enough to Just Hang Out.” The Verge, 8 Aug. 2018, www.theverge.com/2018/8/8/17661902/anki-vector-home-robot-voice-assistant-ai.

2. Upton, E., & Halfacree, G. Raspberry Pi user guide. John Wiley & Sons., 2014.

3. Badamasi, Y. A. The working principle of an Arduino. In Electronics, computer and computation (icecco) 11th international conference on (pp. 1-4). 2014, September.

4. Kaura, H. K., Honrao, V., Patil, S., & Shetty, P. Gesture controlled robot using image processing. Int. J. Adv. Res. Artif. Intell, 2(5), 69-77. 2013.

5. Van Rossum, G., & Drake, F. L.. The python language reference manual. Network Theory Ltd.., 2011.

6. Lutz, M. Programming python. O'Reilly Media, Inc., 2001.

7. Bradski, G., & Kaehler, A. Learning OpenCV: Computer vision with the OpenCV library. . " O'Reilly Media, Inc."., 2008.

8. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678). 2014.

9.“Frequently Asked Questions.” .Arduino. 2018 Arduino.

www.arduino.cc/en/Main/FAQ Last accessed in: 2018/12/30

10. “Arduino Language Reference.” .Arduino. 2018 Arduino.

www.arduino.cc/en/Reference Last accessed in: 2018/12/30