Automatic and Semi-Automatic car

Project Work
submitted in partial fulfillment of the
requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted By
Aanchal Mishra (1413101002)
Abhay Kumar (1413101004)
Vijay Pratap Pandey (1413101314)

Under the supervision of
C. Ramesh Kumar
Assistant Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY
GREATER NOIDA – 201306
MAY 2018

Certificate

I hereby certify that the work which is being presented in the project entitled, **"Automatic and Semi-Automatic car"**, in partial fulfillment of the requirements for the award of degree of Bachelor in Technology in Computer Science and Engineering submitted in School of Computing Science and Engineering of Galgotias University, Gr. Noida, is an authentic record of my own work carried out under the supervision of C. Ramesh Kumar and refers other researcher's works which are duly listed in the reference section.

The matter presented in this project has not been submitted for the award of any other degree of this or any other university.

Aanchal Mishra
Abhay Kumar
Vijay Pratap Pandey

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

C. Ramesh Kumar
School of Computing Science and Engineering
Galgotias University Gr. Noida

Countersigned by

Dr. R. Senthil Kumar
Professor & Dean
School of Computing Science & Engineering
Galgotias University
Gr. Noida

ABSTRACT

Autonomous car (driverless car or self-driving car) is a vehicle which runs without the interventions of humans.
It senses the environment and operates accordingly. The decision making ability it gain by learning (deep learning). The more it will learn from its previous input the better will it become. Car can be moving in four fundamental directions (Forward, backward, left and right).

We use many technology to make self-driving car possible such as Radar, laser light, GPS, odometry, and computer vision.
The radar is to sense the nearby object, might be a man or a wall or some other kind of obstacles.

The laser light is a beam of concentrated light in a direction can be used to locate other car or some other moving object.
GPS is to find the location and odometry is to find the difference in its projection.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

An autonomous car (also known as a driverless car or self-driving car, robotic car) and Unmanned Ground Vehicle is a vehicle that is capable of sensing its environment and navigating without human input. Many such systems are evolving, but as of 2017 no cars permitted on public roads were fully autonomous. They all require a human at the wheel who must be ready to take control at any time. Autonomous cars use a variety of techniques to detect their surroundings, such as radar, laser light, GPS, odometry and computer vision. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage. Autonomous cars must have control systems that are capable of analysing sensory data to distinguish between different cars on the road.

Demonstration systems date to the 1920s and 1930s. The first attempts at truly autonomous cars appeared in the 1980s, with Carnegie Mellon University's Navlab and ALV projects in 1984 and Mercedes-Benz and Bundeswehr University Munich's Eureka Prometheus Projectin 1987. A major milestone was achieved in 1995, when CMU's NavLab 5 completed the first autonomous long distance drive. Of the 2,849 miles between Pittsburgh, PA and San Diego, CA, 2,797 miles were autonomous (98.2%), completed with an average speed of 63.8 miles per hour (102.7 km/h). Since then, companies and research organizations have developed working prototypes.

The potential benefits of autonomous cars include reduced mobility and infrastructure costs, increased safety, increased mobility, increased customer satisfaction and reduced crime. Specifically a significant reduction in traffic collisions, the resulting injuries; and related costs, including less need for insurance. Autonomous cars are predicted to increase traffic flow; provided enhanced mobility for children, the elderly, disabled and the poor; relieve travellers from driving and navigation chores; lower fuel consumption; significantly reduce needs for parking space; reduce crime; and facilitate business models for mobility as a service, especially via the sharing economy.

In spite of the various potential benefits to increased vehicle automation, there are unresolved problems, such as safety, technology issues, disputes concerning liability, resistance by individuals to forfeiting control of their cars, customer concern about the safety of driverless cars, implementation of a legal framework and establishment of government regulations; risk of loss of privacy and security concerns, such as hackers or terrorism; concern about the resulting loss of driving-related jobs in the road transport industry; and risk of increased suburbanization as travel becomes less costly and time-consuming. Many of these issues arise because autonomous objects, for the first time, would allow computers to roam freely, with many related safety and security concerns.

CHAPTER 2

PROJECT REQUIREMENT SPECIFICATION

Language used - python

Tools used -

Raspberry Pi,

H-Bridge – L293D

Sensors – Ultra Sonic Sensor

Servo Motor

Processor: - Quad Core processor

Ram:-1 Gb

Processor Speed:-1.2 GHz

System Type:-64 bit OS, x64-based processor

2.1 **COMPONENTS DESCRIPTION**

1) **Raspberry Pi**

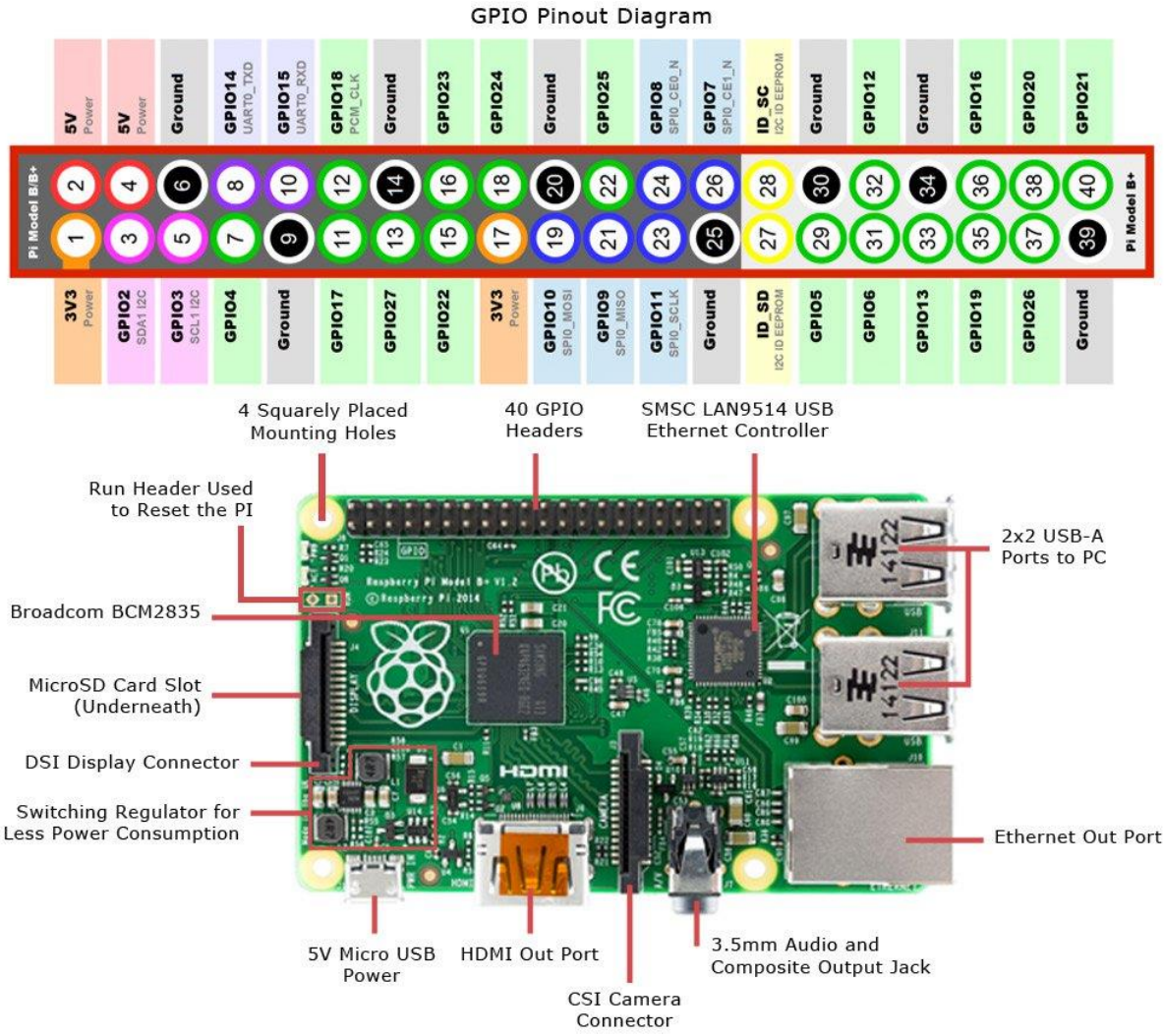
In this proposed system we have used Raspberry Pi as the controller of coordinator node. Rpi is the small, inexpensive minicomputer. It continuously collects the information send by sensor nodes via ZigBee, and processing large quantities of data timely and available for users to view. It is the core of the system.

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, some accessories have been included in several official and unofficial bundles.

Detail about Raspberry pi 3 B+

System-on-chip	-	Broadcom BCM2837B0
CPU	-	1.4 GHz 64/32-bit quad-core ARM Cortex-A53
Memory	-	1 GB LPDDR2 RAM at 900 MHz
Storage	-	MicroSDHC slot
Graphics	-	Broadcom Video Core IV 300 MHz/400 MHz
Power	-	1.5 W (average when idle) to 6.7 W (maximum under stress)

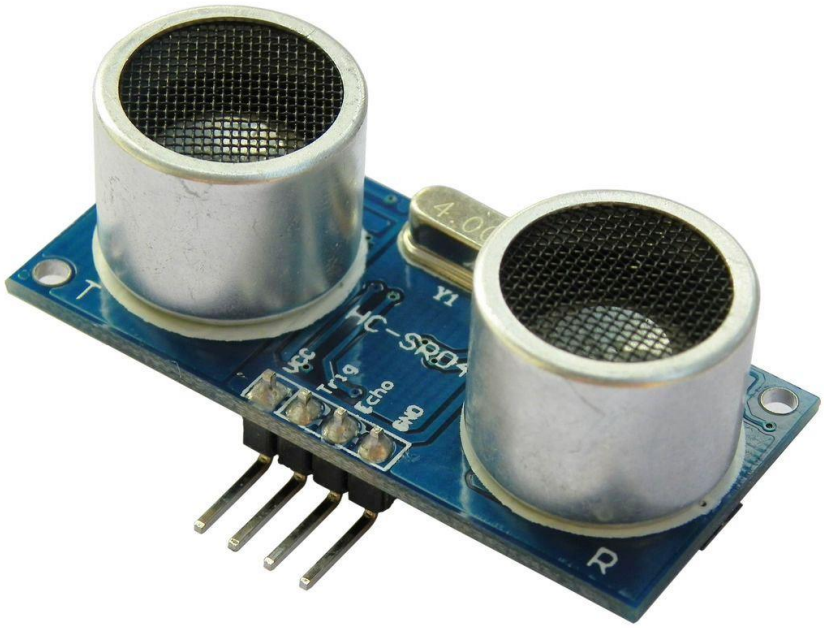
Figure of raspberry pi 3 Model B+ and its GPIO



2) **Sensors-: Ultra Sonic Sensor**

A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.

Picture of Ultrasonic sensor

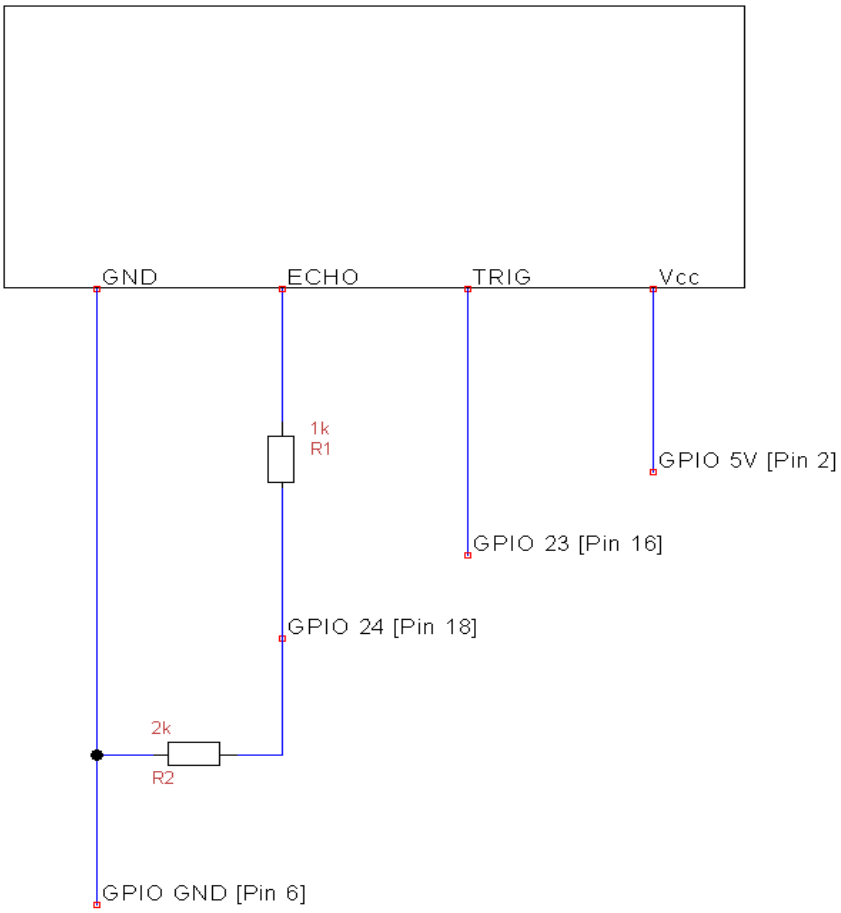


The HC-SR04 Ultrasonic sensor we'll be using in this tutorial for the Raspberry Pi has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc). We power the module using Vcc, ground it using GND, and use our Raspberry Pi to send an input signal to TRIG, which triggers the sensor to send an ultrasonic pulse. The pulse waves bounce off any nearby objects and some are reflected back to the sensor. The sensor detects these return waves and measures the time between the trigger and returned pulse, and then sends a 5V signal on the ECHO pin.

ECHO will be “low” (0V) until the sensor is triggered when it receives the echo pulse. Once a return pulse has been located ECHO is set “high” (5V) for the duration of that pulse. Pulse duration is the full time between the sensor outputting an ultrasonic pulse, and the return pulse being detected by the sensor receiver. Our Python script must therefore measure the pulse duration and then calculate distance from this.

IMPORTANT. The sensor output signal (ECHO) on the HC-SR04 is rated at 5V. However, the input pin on the Raspberry Pi GPIO is rated at 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage your GPIO pins, which is something we want to avoid! We’ll need to use a small voltage divider circuit, consisting of two resistors, to lower the sensor output voltage to something our Raspberry Pi can handle.

Circuit diagram to connect Ultrasonic sensor to pi



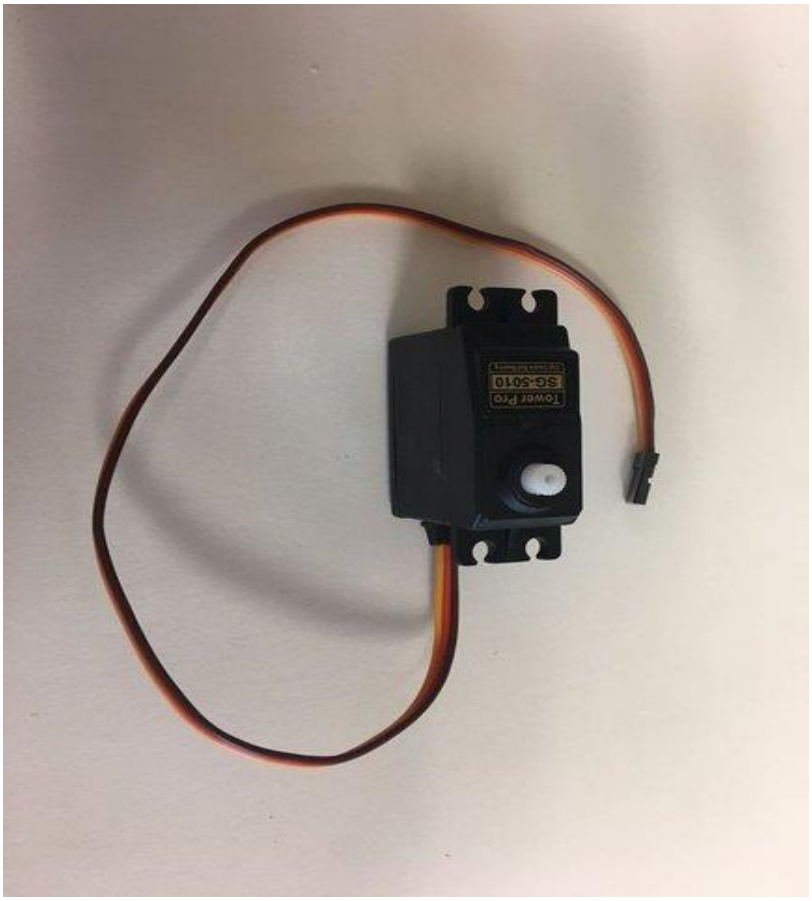
3) **Servo Motor**

A servo motor is a type of DC motor that, upon receiving a signal of a certain frequency, can rotate itself to any angle from 0-180 degrees. Its 90 degree position is generally referred to as 'neutral' position, because it can rotate equally in either direction from that point.

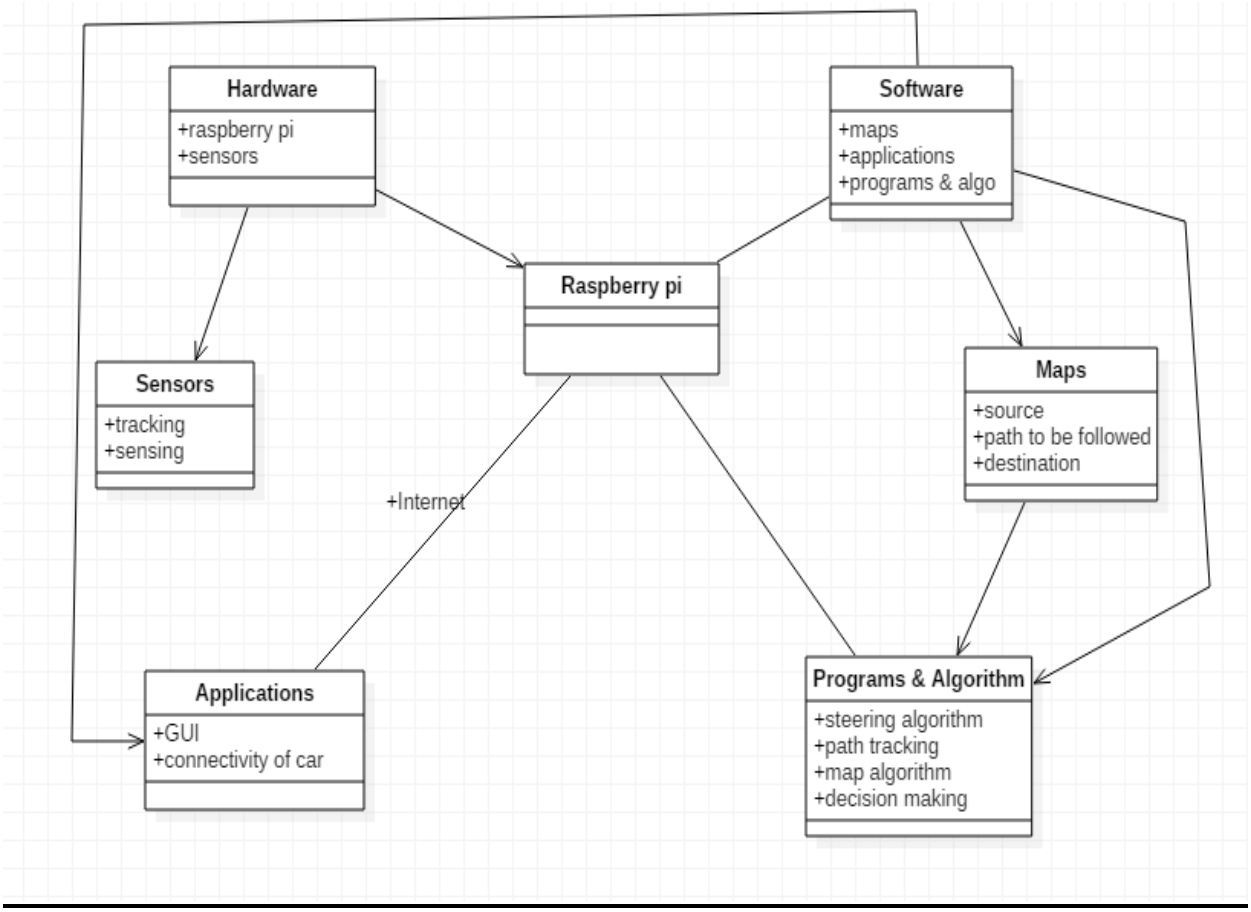
The way a servo motor reads the information it's being sent is by using an electrical signal called PWM. PWM stands for "Pulse Width Modulation". That just means sending ON electrical signals for a certain amount of time, followed by an OFF period, repeated hundreds of times a second. The amount of time the signal is on sets the angle the servo motor will rotate to. In most servos, the expected frequency is 50Hz, or 3000 cycles per minute. Servos will set to 0 degrees if given a signal of .5 ms, 90 when given 1.5 ms, and 180 when given 2.5ms pulses. This translates to about 2.5-12.5% duty in a 50Hz PWM cycle.

We'll be sending PWM signals from one GPIO pin on the RPi, and powering it from the GPIO board, so three wires will run from the servo to the RPi.

Picture of Servo motor



2.2 UML REPRESENTATION



CHAPTER 3

CURRENT STATUS OF THE PROJECT

We have almost completed our project but till now our car is running only on decision algorithm till now we have not implemented any machine learning module, this is because of the absence of the much label i.e. we are deciding the route of the car by only one input i.e. the input from the ultrasonic sensor, so we can't just apply the machine learning or better to say regression algorithm to it.

We are studying different IEEE papers and gathering the knowledge of Machine Learning, Raspberry Pi circuit programming using python and Configuration of Raspberry pi.

CHAPTER 4

Coding and final build

4.1 Code

```
import pygame
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(22,GPIO.OUT) #tyre no 1 near Hbridge
GPIO.setup(27,GPIO.OUT) #tyre no 2
GPIO.setup(10,GPIO.OUT)
GPIO.setup(9,GPIO.OUT)
GPIO.setup(21,GPIO.OUT) #lights
TRIG = 23
ECHO = 24
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
servo_pin = 2     # Initializing the GPIO 02 for servo motor

GPIO.setup(servo_pin, GPIO.OUT)     # Declaring GPIO 02 as output pin
p = GPIO.PWM(servo_pin, 50)     # Created PWM channel at 50Hz frequency

def music(s):
    pygame.mixer.init()
    pygame.mixer.music.load(s)
    pygame.mixer.music.play()

def l():
```

```
    GPIO.output(21,GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(21,GPIO.LOW)

def Ultra():
    GPIO.output(TRIG, False)
    # "Waiting For Sensor To Settle"

    time.sleep(0.25)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO)==0:
        pulse_start = time.time()

    while GPIO.input(ECHO)==1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance

def Servo():
    p.start(2.5)
    p.ChangeDutyCycle(2.5)
    time.sleep(1)
    if Ultra()<20.00:
        p.ChangeDutyCycle(7.5)
        return 1
    p.ChangeDutyCycle(7.5)
    time.sleep(1)
    p.ChangeDutyCycle(12.5)
```

```
    time.sleep(1)
    if Ultra()<20.00:
        p.ChangeDutyCycle(7.5)
        return 2
    p.ChangeDutyCycle(7.5)
    time.sleep(1)
    return 0

def Left():
    GPIO.output(27,GPIO.LOW)
    GPIO.output(22,GPIO.HIGH)
    GPIO.output(10,GPIO.LOW)
    GPIO.output(9,GPIO.LOW)

def Right():
    GPIO.output(22,GPIO.LOW)
    GPIO.output(27,GPIO.HIGH)
    GPIO.output(10,GPIO.LOW)
    GPIO.output(9,GPIO.LOW)

def fn():
                GPIO.output(27,GPIO.HIGH)
                GPIO.output(22,GPIO.HIGH)
                GPIO.output(10,GPIO.LOW)
                GPIO.output(9,GPIO.LOW)
                time.sleep(1)

                if Ultra()<20.00:
                    music("Obstacle.wav")
                    time.sleep(0.25)
                    GPIO.output(27,GPIO.LOW)
                    GPIO.output(22,GPIO.LOW)
                    GPIO.output(10,GPIO.LOW)
                    GPIO.output(9,GPIO.LOW)
```

```
                s=Servo()
                if s==1:
                    for l in range(1):
                        Left()
                        time.sleep(1)
                elif s==2:
                    for l in range(1):
                        Right()
                        time.sleep(1)
                elif s==0:
                    Right()
                    time.sleep(1)
                    for l in range(1):
                        GPIO.output(27,GPIO.HIGH)
                        GPIO.output(22,GPIO.HIGH)
                        GPIO.output(10,GPIO.LOW)
                        GPIO.output(9,GPIO.LOW)
                        time.sleep(1)
                    Left()
                    time.sleep(1)
                    for l in range(1):
                        GPIO.output(27,GPIO.HIGH)
                        GPIO.output(22,GPIO.HIGH)
                        GPIO.output(10,GPIO.LOW)
                        GPIO.output(9,GPIO.LOW)
                        time.sleep(1)
                    Right()
                    time.sleep(0.50)

try:
    list=[[0,0,0],[0,0,5],[1,0.2,8],[0,0,10],[2,1,18]] #Sublist -1st element represent
direction 0-straight 1-left 2-right,
                                        #2nd element represent time to turn, 3rd represent
time to reach
```

```
    time.sleep(3)
    music("Source.wav")

    for i in range(1,len(list)):
            k=list[i][2]-list[i-1][2]

            if list[i][0]==1:
               music("Left.wav")
               time.sleep(1)
               Left()
               time.sleep(list[i][1])

            elif list[i][0]==2:
               music("Right.wav")
               time.sleep(1)
               Right()
               time.sleep(list[i][1])

            for j in range(k):
               l()
               time.sleep(0.25)
               fn()
    time.sleep(2)
    music("Dest.wav")


# If Keyborad Interrupt (CTRL+C) is pressed
except KeyboardInterrupt:
    pass   # Go to next line
GPIO.cleanup()            # Make all GPIO pins LOW
```

4.2 Comments and Description of code

The car has following functionality

1. It can move forward

   To move forward we set GPIO 23 and 27 at hight and GPIO 9 and 10 at low , so to make a voltage change such that it can drive our car in forward direction. We can also vary the speed of car by applying a sleep function and altering low and high on GPIO

```
def Forward():
   GPIO.output(22,GPIO. HIGH)
   GPIO.output(27,GPIO.HIGH)
   GPIO.output(10,GPIO.LOW)
   GPIO.output(9,GPIO.LOW)
```

   This is the function we are using to move forward

2. It can turn right

   To move right we stop the movement of the left wheel and continue the movement of right wheel. It is achieved by making the both GPIO of left wheel at same potential that is either set both GPIO at high or set both GPIO at low, and making no change of the right gpio.

```
def Right():
   GPIO.output(22,GPIO.LOW)
   GPIO.output(27,GPIO.HIGH)
   GPIO.output(10,GPIO.LOW)
   GPIO.output(9,GPIO.LOW)
```

   This is the function we are using to move right

3. It can turn left

To move left we stop the movement of the right wheel and continue the movement of left wheel. It is achieved by making the both GPIO of right wheel at same potential that is either set both GPIO at high or set both GPIO at low, and making no change of the left gpio.

```
def  Left():
   GPIO.output(27,GPIO.LOW)
   GPIO.output(22,GPIO.HIGH)
   GPIO.output(10,GPIO.LOW)
   GPIO.output(9,GPIO.LOW)
```

This is the function we are using to move left

4. It can stop

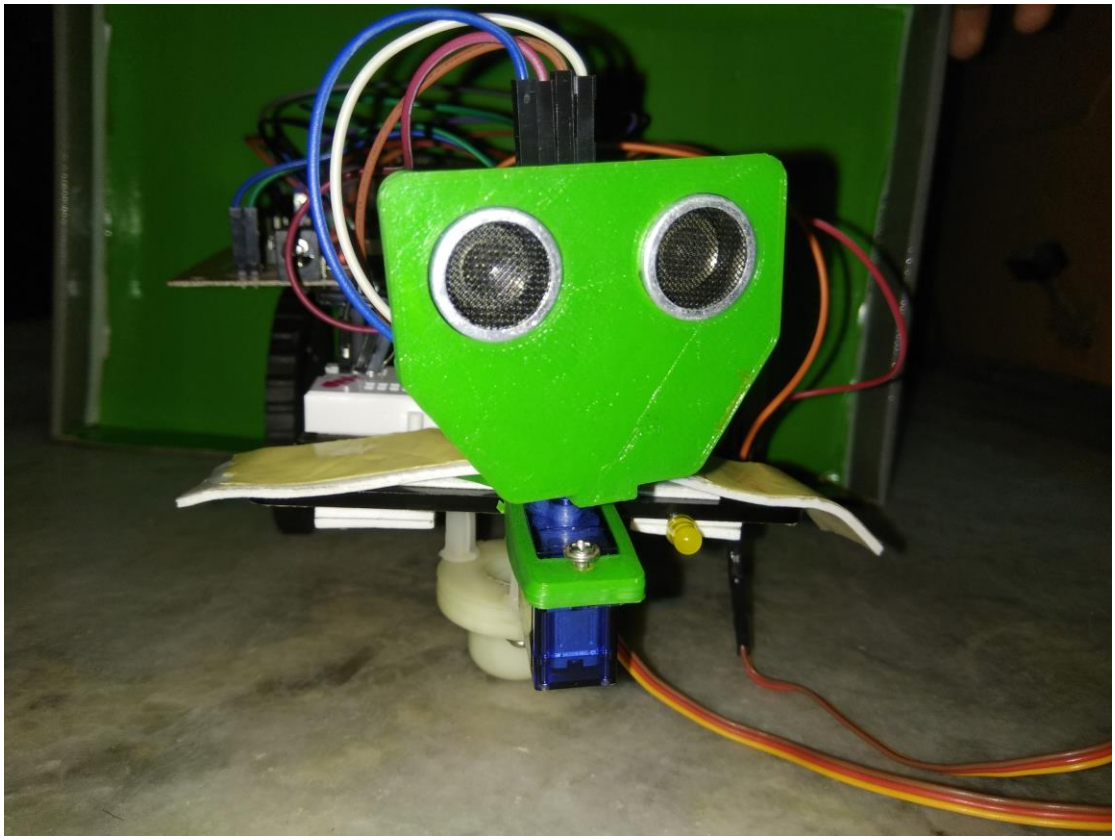
To stop our car to have to stop the movement of both the wheel. This is achieved by applying the same potential at all GPIO that either applying high potential to all four GPIO of the wheel or applying low potential to all the four GPIO of the wheel

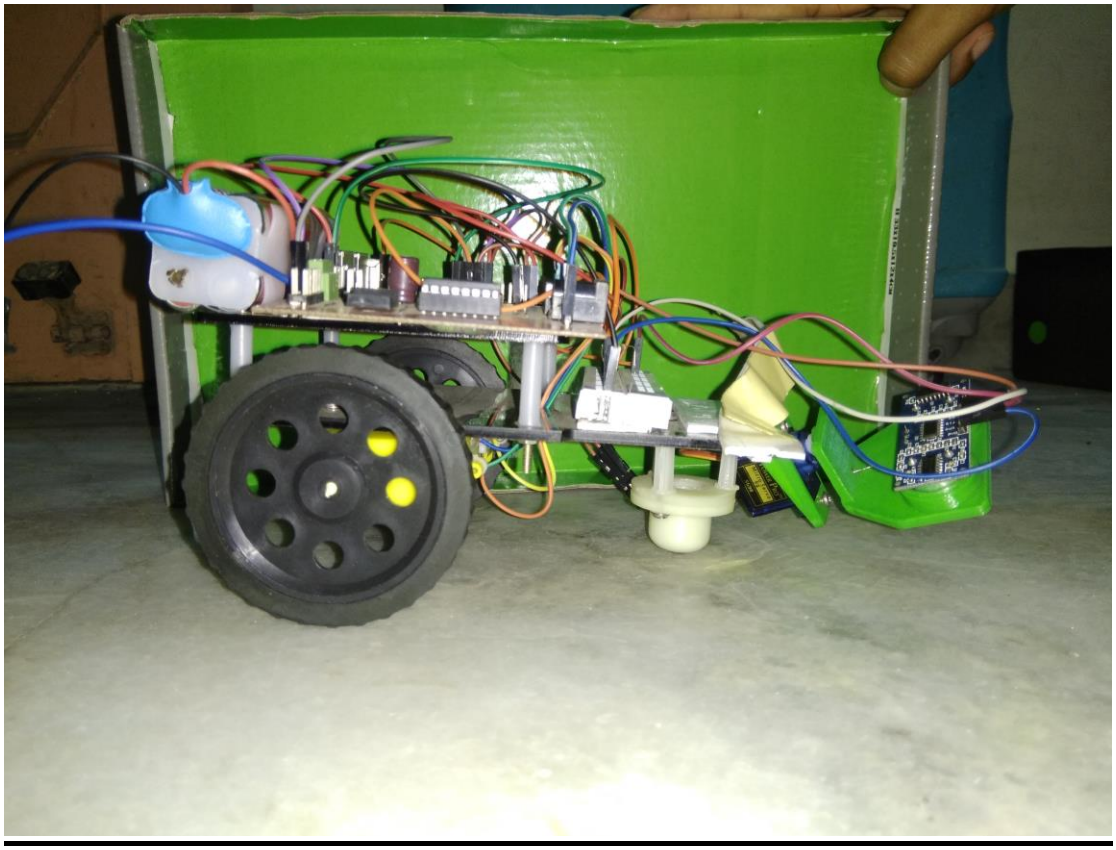
```
def  Stop():
   GPIO.output(27,GPIO.LOW)
   GPIO.output(22,GPIO. LOW)
   GPIO.output(10,GPIO.LOW)
   GPIO.output(9,GPIO.LOW)
```

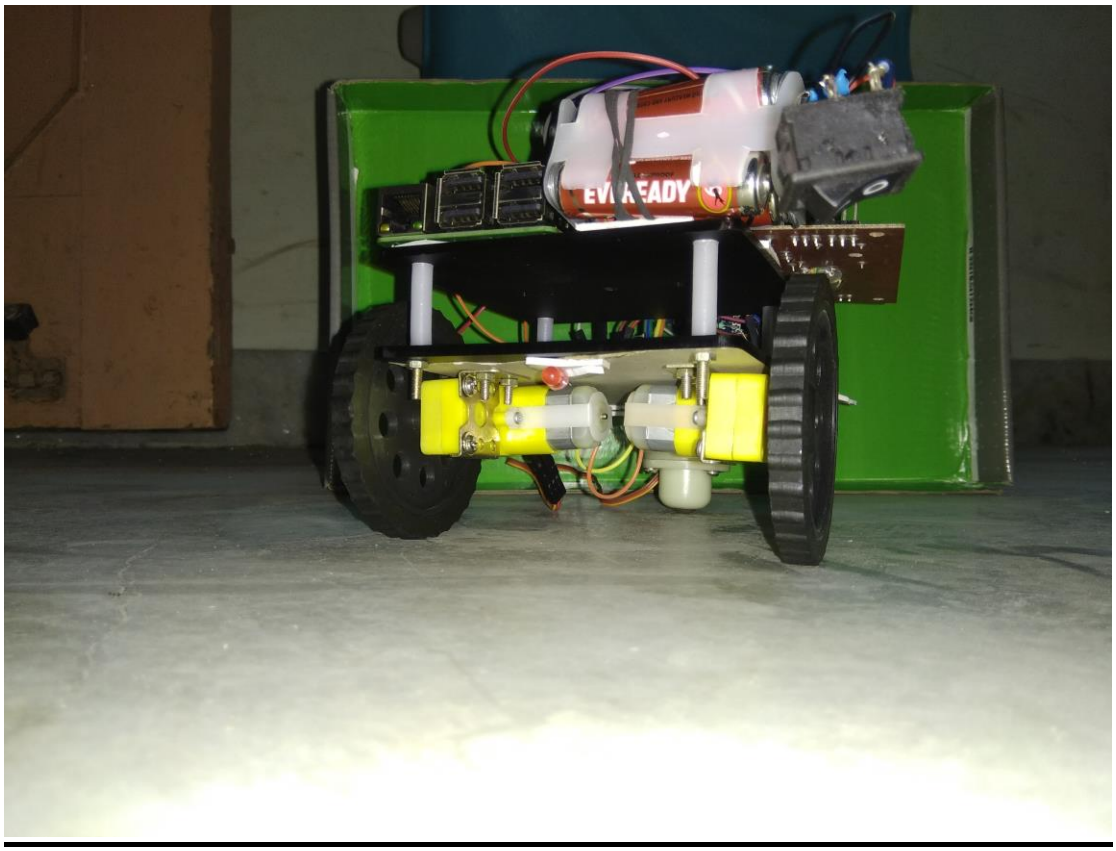
This is the function we are using to stop the car

Our car chassis

Font look of the car






EVEREADY

CHAPTER 5

Project Objectives

The main objective of the project is to reduce the number of accident on the road causes due to the mistake of the driver and make road a safe place. As the car is control by an intelligent computer, so it will give you the optimal path by which you can cover the distance in shorter amount of time. One of the main advantages of this system is that it will reduce congestion problem (traffic jam).

This project has many scopes like :-

- Ambulance – This is the most extreme cases of shortage of time.
- Public transports
- Delivery vehicle
- And also our personal vehicle

CHAPTER 5

MOTIVATION

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting area of research. The opportunity to learn about a new area of computing not covered in lectures was appealing. This area is possibly an area that I might study at postgraduate level.

CHAPTER 5

FEASIBILITY AND ADVANTAGES

- Easier to use
- Less manually restrictive
- Greatly reduced risk of stalling
- Easier to use in heavy traffic
- Automatic transmission are easier to use and more comfortable for the driver

CHAPTER 6

WORK LOAD MATRIX

1.Configuration study of Raspberry pi through the study of IOT,requirement elicitation and study about prerequisites.

2.Developing a circuit for the Raspberry pi functionality and controlling of car.

3.Establishing connection between devices and working on the display part.

4.Backend coding with the help of python programming language and enabling the project functionality.

5.Designing user interface i.e. webpage,establishing database connectivity,working on wifi module.

6.Testing and analysis of the project.

CHAPTER 7

CONCLUSION

Despite the inherent benefits, autonomous vehicle technology must overcome many social barriers .Much like the issues faced by the first automobiles, influence of mental model can impede the advancement of technology. However ,new legislation is creating opportunities for these cars to prove their viability. As more state legalize driverless cars, the social obstruction give way, allowing for the largest revolution in personal transportation since the introduction of automobile.

Autonomous vehicle selected as the most promising form of intelligent transportation ,anticipating that they will account upto75% of cars on the road by the year 2040

CHAPTER 8

REFERENCES

1. https://ieeexplore.ieee.org/document/8125889/ (IEEE xplore digital library).

2.https://en.wikipedia.org/wiki/Internet_of_things (IOT concept)

3.https://www.tutorialspoint.com. (GPIO Tutorial)

4.https://www.elprocus.com/architecture-of-wireless-sensor-network-and-applications/ (Sensor network)

5.www.instructables.com/id/Raspberry-Pi/ ()

6.https://www.google.com/Autonomous-car