**AUTONOMOUS ROBO CAR**

***Dissertation submitted to***

***Shri Ramdeobaba College of Engineering & Management, Nagpur in partial fulfillment of requirement for the award of***

***degree of***

**Bachelor of Engineering**

**In**

**(Computer science & Engineering)**

**Eighth Semester**

*BY*

***Anshul Sarda [BE12CSU029]***

***Jogendra Singh Shekhawat [BE12CSU037]***

***Kaustubh Risodkar* *[BE12CSU039]***

***Kshitij Madan [BE12CSU40]***

***Pratik Upacharya[BE12CSU44]***

***Under the guidance of***

**Dr. R. Hablani**



**Department of Computer Science & Engineering**

**Shri Ramdeobaba College Of Engineering & Management**

**(An Autonomous College of Rashtrasant Tukadoji Maharaj Nagpur University)**

**April 2016**

**SHRI RAMDEOBABA COLLEGE OF ENGINEERING & MANAGEMENT, NAGPUR**

(An Autonomous Institute Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

Department of Computer Science & Engineering

**CERTIFICATE**

This is to certify that the Thesis on

**“Autonomous Robo Car”**

is a bonafide work of

**Anshul Sarda *[BE12CSU029]***

**Jogendra Singh Shekhawat *[BE12CSU037]***

**Kaustubh Risodkar*[BE12CSU039]***

**Kshitij Madan*[BE12CSU040]***

**Pratik Upacharya*[BE12CSU044]***

submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfillment of the award of a Bachelor of Engineering, in Computer Science & Engineering has been carried out at the Department of Electronics Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year 2016.

Date:

Place:

Prof.R. Hablani Dr. M. B. Chandak

Project Guide HOD

Department of CSE

Dr. R. S. Pande

Principal

**DECLARATION**

I, hereby declare that the thesis titled **“AUTONOMOUS ROBO CAR”** submitted herein, has been carried out in the Department of Computer Science & Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other institution / University

Date:

Place:

**GROUP 12**

Anshul Sarda(45)

Jogendra Singh Shekhawat(53)

Kaustubh Risodkar(56)

Kshitij Madan(57)

Pratik Upacharya(61)

**Approval Sheet**

This report entitled **“Autonomous Robo Car”** by Anshul Sarda, Jogendra Singh Shekhawat, Kaustubh Risodkar, Kshitij Madan and Pratik Upacharya is approved for the degree of Computer Science & Engineering.

Name & signature of Supervisor(s) Name & signature of External. Examiner(s)

---------------------------- ----------------------------

---------------------------- ----------------------------

+

Name & signature RRC Members Name & signature of HOD

---------------------------- ----------------------------

---------------------------- ----------------------------

Date:

Place:

**Acknowledgement**

We take this opportunity to express our profound gratitude and deep regards to our project Guide Mr. Ramchand Hablani for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given by him time to time shall carry us a long way in the journey of life on which we’re about to embark.

 We also take this opportunity to express a deep sense of gratitude to our project incharge Prof. A.R. Raipurkar and our Head of department Dr. M.B.Chandak for providing us relevant information and necessary clarifications.

Lastly, we would also like to thank our institution and faculty members for their support and also our family and friends for their co-operation.

GROUP 12

Anshul Sarda, 45

Jogendra Singh Shekhawat, 53

Kaustubh Risodkar, 56

Kshitij Madan,57

Pratik Upacharya,60

**Abstract**

A lot of times we do not want to feel the heat and do not want to drive our car or vehicle. This is because driving is very stressful and tiring at times. And given the traffic out there, it is really had to navigate through it. In this day and age, where everything is AUTONOMOUS, we feel there is a need to design an AUTONOMOUS robot car.

The Autonomous Robot Car we aimed to design, will run on instructions provided by the users that is us and hence reduce the effort and stress related to driving manually. This car rotates through angles and takes turns on the commands given by the users. Also if there is an obstacle which comes in front of the car, the car will stop automatically. After the car comes within a specific distance of an obstacle, the ultrasonic sensors we have in the system detect the object and the car is hence directed to stop.

The given project provided us an opportunity to peep into a different dimension altogether and learn something new. We learned how to use a raspberry-pi microcontroller. Also, we learned new tool called Open- CV. This project helped us to look into how an AUTONOMOUS robot car is built where we used all the different components and assembled them to create a working version of the vehicle

Keywords: Ultrasonic sensors, Autonomous, Raspberry-Pi, Car

# TABLE OF CONTENTS

Page No.

# Acknowledgements v

**Abstract vi**

**List of Figures/Tables viii**

1. **INTRODUCTION 1**
   1. Need
   2. Objective
2. **Review Of Literature**  **2**
3. **Hardware Technologies 3**
   1. Raspberry-pi
   2. Johnson Electric Motor
   3. Wi-Fi adapter
   4. Ultrasonic Sensor
4. **Software details 7**
   1. OpenCV
   2. Python
   3. Flask Server
5. **Proposed Approach & system design 10**
   1. Model
   2. Working
6. **System Description 16**
7. **Implementation 17**
8. **Result and discussion**
9. **Conclusion**
10. **Reference**

**List of Figures :**

Page number

Figure 1 : Raspberry Pi 4

Figure 2 : Raspberry Pi block model 4

Figure 3 : Pin diagram of Raspberry Pi 5

Figure 4 : DC motor 5

Figure 5 : Ultra Sonic Sensor 6

Figure 6 : Open CV logo 8

Figure 7 : Python logo 9

Figure 8 : Proposed plan of work 11

Figure 9 : Obstacle avoidance flowchart 12

Figure 10: Main program flowchart 13

Figure 11: Systematic block diagram of hardware 14

Figure 12: Mechanical design process 14

Figure 13: State of charge of the battery used 17

Figure 14: Our model 25

**1.INTRODUCTION**

* 1. **NEED:**

With increasing population and increasing no of vehicles on the road, there is a lot of pressure on people who are driving cars. Driving is really a tedious process and requires a lot of concentration. Therefore, this is a small attempt at formulating a design which can help pave the way for a model in the future where a vehicle can navigate automatically through the traffic without any mechanical control from the user and stopping whenever it sees obstacles by itself.

This model is required because:

* It will reduce driving related stress and pressure.
* It will have wider appeal in the market as not many people are interested in driving manually.
* It is much easier to control with no clutch or shifter or gear to take care of.
* While countering steep climbs or rough roads, an automatic car is always better as it decreases the chances of error.
* There is less wear and tear which has been proven and hence automatic cars have higher life span.
  1. **OBJECTIVE:**

To design an autonomous robot car which:

* Works on its own without any mechanical control from the user.
* Is able to move freely in any direction
* Detects obstacles.
* Uses ultrasonic sensors to calculate the distance and the approximate angle from the obstacle.
* Takes the above information and decides the direction in which it should move.
* Safely navigates through the obstacles.

**2.REVIEW OF LITERATURE**

**Already existing transport system:**

In our day to day life if we need to travel from one place to another we either use or own vehicle or public transportation. Now as easy as it sounds it is as cumbersome. Given the time period you require to reach your destination, imagine the work that can be done during that time if you are not at the driver’s seat driving you vehicle to your destination and not only you, but no one on the driver’s seat and the vehicle driving itself like an advanced robot with human senses.

An autonomous car (driverless car, self-driving car, robotic car) is a vehicle that is capable of sensing its environment and navigating without human input. Autonomous vehicles detect surroundings using radar, lidar, 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 have control systems that are capable of analyzing sensory data to distinguish between different cars on the road, which is very useful in planning a path to the desired destination. The major disadvantage of this is the high cost of production of such an advanced technology. As many expensive sensors and a huge amount of man force is required to put this into implementation, the price of the end product won’t necessarily turn out to be economical for end user.

**Manual Controlled vehicle:**

What we use in today’s world is manually controlled transportation system. There’s nothing wrong with manually driven motor bikes or cars but if they were to be replace by something which drives itself that would account to saving not only a huge amount of time but also an enormous amount of energy that one puts into moving between destinations. Manually driven vehicle amounts to wasting of time and energy in first learning the vehicle and later on driving it daily to workplace or any other destination. Also, it requires the driver to be skilled enough to handle the vehicle. One cannot expect a person riding scooter to drive a train or a car. So skillset is required to drive different types of vehicles. One can also not ignore the human error/accidents that can be caused by people driving vehicles.

**Autonomous Robot Car:**

Autonomous Robot car will be controlled by a low cost, credit card sized computer called Raspberry Pi. This will be in accordance with different sensors attached to the car like ultrasonic sensor, gyro sensor, etc. It will also use a camera to detect obstacle in its pathways and apply different algorithms on the image captured. The major aim is to develop an autonomous car which drives itself, avoiding collision irrespective of the obstacles in its pathway.

**3.Hardware Technologies**

The following hardware technologies were used in the project:

* Raspberry Pi
* Johnson Electric motor(12 V)
* Ultra sonic sensor
* Wi-Fi adapter

**3.1 Raspberry Pi**

The Raspberry pi is a single computer board with credit card size, that can be used for many tasks that your computer does, like games, word processing, spreadsheets and also to play HD video. It was established by  the Raspberry pi foundation from the UK. It has been ready for  public consumption  since 2012 with the idea of making a low-cost educational microcomputer for students and children. The main purpose of designing the raspberry pi board is, to encourage learning, experimentation and innovation for school level students. The raspberry pi board is a portable and low cost. Maximum of the raspberry pi computers is used in mobile phones. In the 21st century, the growth of mobile computing technologies is very high, a huge segment of this being driven by the mobile industries. The 98% of the mobile phones were using ARM technology.

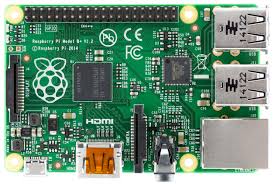


Figure 1: Raspberry Pi

The raspberry pi boards are used in many applications like Media streamer, Arcade machine, Tablet computer, Home automation, Carputer, Internet radio, Controlling robots, Cosmic Computer, Hunting for meteorites, Coffee and also in raspberry pi based projects

3.1.1**Hardware**

The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.

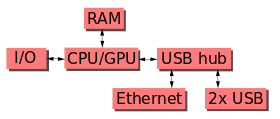
[](https://en.wikipedia.org/wiki/File:Raspberrypi_block_function_v01.svg)

Figure 2: Raspberry Pi block diagram

This block diagram depicts models *A*, *B*, *A+*, and *B+*. Model *A* and *A+* and *Zero* lack the Ethernet and USB hub components. The Ethernet adapter is connected to an additional USB port. In model *A* and *A+* the USB port is connected directly to the SoC. On model *B+* and later models the USB/Ethernet chip contains a five-point USB hub, of which four ports are available, while model *B* only provides two. On the model *Zero*, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

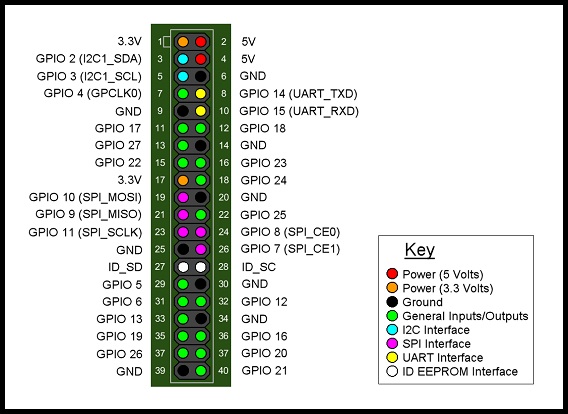


Figure 3: Pin diagram of Raspberry Pi

**3.2Johnson Electric motor(DC)**

A **DC motor** is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.



Figure 4: DC motor

**3.3 Wi-Fi Adapter**

Wi-Fi Adapter is used in the project to establish connection between laptop and raspberry-pi wirelessly. Benefits of using a Wi-Fi adapter are as follows :

* Frees Computer Users from Cables
* Eliminates the Need to Install Internal Hardware
* Eliminates the Need to Run Cables Throughout the Entire House or Office
* Eliminates the Need to Upgrade a Computer
* Can Be Used on Multiple Devices

### 3.4 Ultrasonic Sensor

### In industrial applications, ultrasonic sensors are characterized by their **reliability** and **outstanding versatility**.**** Ultrasonic sensors can be used to solve even the most complex tasks involving **object** **detection** or **levelmeasurment**with **millimeter precision**, because their measuring method works reliably under almost all conditions. No other measuring method can be successfully put to use on such a wide scale and in so many different applications. The devices are **extremely robust**, making them suitable for even the **toughest conditions**. The sensor surface cleans itself through vibration, and that is not the only reason why the sensor is insensitive to dirt. The physical principle—the propagation of sound—works, with a few exceptions, in practically any environment.

### Two ultrasonic sensors are used in the project to find the distance between an obstacle and the car.

### 

### Figure 5: Ultrasonic Sensor

### 4. Software

**4.1 OpenCV**

**OpenCV** (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision, originally developed by Intel's research center in Nizhny Novgorod (Russia), later supported by Willow Garage and now maintained by Itseez.   The library is cross-platform and free for use under the open-source BSD license.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and  MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, and Ruby have been developed to encourage adoption by a wider audience.All of the new developments and algorithms in OpenCV are now developed in the C++ interface. OpenCV runs on a variety of platforms. Desktop: Windows, Linux, OS X, FreeBSD, NetBSD, OpenBSD; Mobile: Android, iOS, Maemo, BlackBerry 10.The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

f the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

OpenCV is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

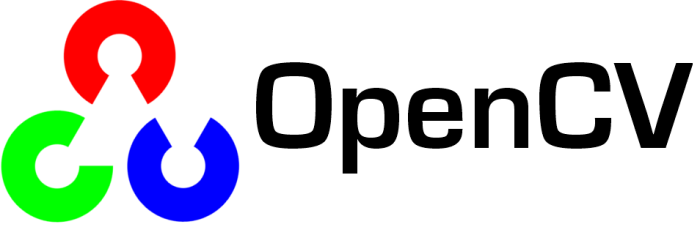


Figure 6: OpenCV logo

**4.2 Python**

**Python** is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Using third-party tools, such as Py2exe or Pyinstaller, Python code can be packaged into stand-alone executable programs for some of the most popular operating systems, allowing the distribution of Python-based software for use on those environments without requiring the installation of a Python interpreter.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by metaprogramming and by magic methods). Many other paradigms are supported using extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. Most Python implementations (including CPython) can function as a command line interpreter, for which the user enters statements sequentially and receives the results immediately (REPL). In short, Python acts as a shell. Other shells add capabilities beyond those in the basic interpreter, including IDLE and IPython. While generally following the visual style of the Python shell, they implement features like auto-completion, retention of session state, and syntax highlighting.

In addition to standard desktop Python IDEs (integrated development environments), there are also browser-based IDEs, Sage (intended for developing science and math-related Python programs), and a browser-based IDE and hosting environment, PythonAnywhere.

****

Figure 7: Python Logo

**4.3 Flask Server**

**Flask** is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed.

As of Feb 2016, the latest stable version of Flask is 0.10.1. Examples of applications that make use of the Flask framework arePinterest, LinkedIn, as well as the community web page for Flask itself.

Flask is called a micro framework because it does not presume or force a developer to use a particular tool or library. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Flask was created by Armin Ronacher of Pocoo which is an international group of Python enthusiasts formed in 2004. According to Armin, "It came out of an April Fool's joke but proved popular enough to make into a serious application in its own right." Flask is based on the Werkzeug [WSGI](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface) toolkit and Jinja2 template engine, both of them Pocoo projects that were created when Ronacher and Georg Brandl were building a bulletin board system written in Python.

Despite the lack of a major release, Flask has become extremely popular among Python enthusiasts.

Features:

* Contains development server and debugger
* Integrated support for unit testing
* RESTful request dispatching
* Uses [Jinja2](https://en.wikipedia.org/wiki/Jinja_(template_engine)) templating
* Support for secure cookies (client side sessions)
* 100% WSGI 1.0 compliant
* [Unicode](https://en.wikipedia.org/wiki/Unicode)-based

**5. Proposed Approach And System Architecture:**

**5.1 Basic Plan:**

In this Project, we plan to build an autonomous robot car using image processing. We required raspberry-pi to control dc motors and ultrasonic sensors for distance and angle approximations and a laptop for image processing.

The laptop communicates with raspberry-pi through serial communication. We used Python for coding in raspberry-pi and C++ for image processing. Our aim is to detect a obstacle and avoid it safely.

**5.1.1 Proposed plan of work**

Research about hardware components

Comparison between micro controllers and micro-processors and other hardware components

Learn working of each hardware and connection component

Connections, testing the components, learning about power supply and designing circuits

Controlling raspberry-pi through computer

Image processing using OPEN-CV on laptop

Writing algorithms for obstacle avoidance

Controlling all components through raspberry-pi

Integrating the project

Figure 8: Proposed Plan Of Work

**5.2 Structural Flow:**

Start

**N**

Obstacle

**Y**

Judge the position

Left

Turn

Turn

**Y Y**

Obstacle

Obstacle

**N N**

Figure 9: Obstacle Avoidance Flow Chart

Start

Initial

Wait

If Laptop sends a command

**N**

Move according to the command

Figure 10: Main Program Flow Chart

**5.3 Designing the Model**

**5.3.1 Control System Design**

Requirements for control system:

(1) We can control the speed and direction of the motors by change the pulse width

(2) The microcontroller turns the signals into PWM wave when receive the laptop

input and control the movement of the robot.

Princip

Motors

Motor Drive Chip

Micro-controller

Laptop

Figure 11: Systematic Block Diagram Of Hardware

Laptop sends motion commands to microcontroller, and then the microcontroller

sends out a PWM pulse corresponding to the motion commands via p1.1 which will

drive the motors by the HB-25,then the robot will realize the corresponding motion.

**5.3.2 Mechanical Design Process**

Function Of Robot

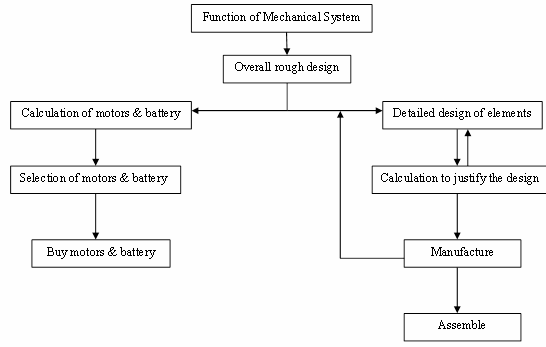
****

Figure 12: Mechanical Design Process

**5.3.3 Motor Calculations**

About the calculation of motors: the parameters we need to know about

The motors are minimum torque, maximum revolving speed, voltage and

Power.

* Minimum torque min *T* : the minimum torque should be able to rotate the

Wheels when the robot is on the ground. The weight of the robot Gcan

Be estimated, and the friction coefficient between the wheels and the

Ground m can be known through table of friction coefficient. The

Maximum speed max *v* and maximum acceleration max *a* can be certified by

The experiment that the images of the cameras won’t blur and laptop has

Enough time to deal with each image. According to Newton’s Second

Law, **2\*(T/r)-(u\*G)= (G/g)\*a,** r is the radius of the wheel

* Maximum revolving speed: The maximum speed of the robot max *v* can be

Certified by the experiment that the images of the cameras won’t blur.

Then the maximum revolving speed of the motors is **n= 30\*v/3.14\*r**

* Voltage: The voltage of motors is usually 12V or 24V. They are both

Acceptable.

* Power: The energy of motors equals the kinetic energy the robot gains

within one second plus friction loss within one second.

* About the calculation of battery: The parameters we need to know about

battery are voltage and current. The voltage and current of battery should be

no bigger than the maximum voltage and current the motors can stand.

Another important parameter of battery is that battery capacity as it

determines how long the battery can work at a certain current.

**5.4 Types of Commands**

* **F00**- Move forward with Full Speed
* **FF0**- Move forward with Half Speed
* **R00**-Move right wheel with Full Speed
* **RR0**- Move right wheel with Half Speed
* **L00**- Move left wheel with Full Speed
* **LL0**- Move left wheel with Half Speed
* **B00**- Move back with Full Speed
* **BB0**- Move back with Half Speed
* **S00**- Stop

**6. SYSTEM DESCRIPTION**

**6.1Requirement Specification**

**6.1.1 Functional Requirements**

* Raspberry-Pi Microcontroller
* Two DC Geared Motors(12V)
* Battery (12 V 13000 Ma)
* USB TTL
* Motor Drivers
* Encoder
* Two Ultrasonic Sensors
* Heat Sink
* Two Free wheels for front and Two wheels for back
* HDMI CableAdapters

### 

### Figure 13: State of charge of the battery used

**7. Implementation:**

We used the following code to implement the model

# Function to measure the Distance of obstacel on left side

def MeasureLeft():

flag=0

print("Left Distance Measurement In Progress")

GPIO.output(TRIG\_L, False)

print("Waiting For Sensor To Settle")

time.sleep(.5)

GPIO.output(TRIG\_L, True)

time.sleep(0.00001)

GPIO.output(TRIG\_L, False)

while GPIO.input(ECHO\_L)==0:

pulse\_start = time.time()

while GPIO.input(ECHO\_L)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance, 2)

if distance >200:

distance = 0

print("Left Distance:",distance,"cm")

return distance

# Function to measure the Distance of obstacel on Right side

def MeasureRight():

print("Right Distance Measurement In Progress")

GPIO.output(TRIG\_R, False)

print("Waiting For Sensor To Settle")

time.sleep(.5)

GPIO.output(TRIG\_R, True)

time.sleep(0.00001)

GPIO.output(TRIG\_R, False)

while GPIO.input(ECHO\_R)==0:

pulse\_start = time.time()

while GPIO.input(ECHO\_R)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance, 2)

if distance >200:

distance = 0

print("Right Distance:",distance,"cm")

return distance

def Forward():

GPIO.output(21,1)

GPIO.output(22,0)

GPIO.output(23,1)

GPIO.output(24,0)

return 0

def Back():

GPIO.output(21,0)

GPIO.output(22,1)

GPIO.output(23,0)

GPIO.output(24,1)

return 0

def Left():

GPIO.output(21,0)

GPIO.output(22,0)

GPIO.output(23,1)

GPIO.output(24,0)

return 0

def Right():

GPIO.output(21,1)

GPIO.output(22,0)

GPIO.output(23,0)

GPIO.output(24,0)

return 0

def Stop():

GPIO.output(21,0)

GPIO.output(22,0)

GPIO.output(23,0)

GPIO.output(24,0)

return 0

def ForwardSlow():

Forward()

time.sleep(.2)

Stop()

time.sleep(.2)

return 0

def LeftSlow():

Left()

time.sleep(.2)

Stop()

time.sleep(.2)

return 0

def RightSlow():

Right()

time.sleep(.2)

Stop()

time.sleep(.2)

return 0

def Distance(x):

count\_l = 0

dist\_l = 0

count\_r = 0

dist\_r = 0

while 1:

i=GPIO.input(ENCODER\_L)

j=GPIO.input(ENCODER\_R)

Forward()

if i==1 and count\_l == 0:

count\_l = 1

if i==0 and count\_l == 1:

count\_l = 0

if dist\_l == x and dist\_r == x:

break

else:

dist\_l +=1

if j==1 and count\_r == 0:

count\_l = 1

if j==0 and count\_r == 1:

count\_r = 0

if dist\_l == x and dist\_r == x:

break

else:

dist\_r +=1

return 0

def LeftTurn(angle):

count\_r = 0

dist\_r = 0

x = math.radians(angle)

y = 50\*math.sin(x)

while 1:

j=GPIO.input(ENCODER\_R)

Left()

if j==1 and count\_r == 0:

count\_l = 1

if j==0 and count\_r == 1:

count\_r = 0

if dist\_r == x:

break

else:

dist\_r +=1

return 0

def RightTurn(angle):

count\_l = 0

dist\_l = 0

x = math.radians(angle)

y = 50\*math.sin(x)

while 1:

i=GPIO.input(ENCODER\_L)

Right()

if i==1 and count\_l == 0:

count\_l = 1

if i==0 and count\_l == 1:

count\_l = 0

if dist\_l == x:

break

else:

dist\_l +=1

return 0

# main code Starts here

import RPi.GPIO as GPIO # library for accessing GPIO

import time # library for accesing time

import serial # library for accesing serial port

import math

GPIO.setmode(GPIO.BOARD) # calling Raspi pin by pin no. on boards

GPIO.setwarnings(False)

# Setting the Pins for interfacing Hardware

ENCODER\_L = 15

ENCODER\_R = 16

MOTOR\_L1 = 21

MOTOR\_L2 = 22

MOTOR\_R1 = 11

MOTOR\_R2 = 12

ECHO\_L = 29

TRIG\_L = 31

ECHO\_R = 33

TRIG\_R= 35

# defining the hardware type I/P or O/P

GPIO.setup(ENCODER\_L,GPIO.IN)

GPIO.setup(ENCODER\_R,GPIO.IN)

GPIO.setup(TRIG\_L,GPIO.OUT)

GPIO.setup(ECHO\_L,GPIO.IN)

GPIO.setup(TRIG\_R,GPIO.OUT)

GPIO.setup(ECHO\_R,GPIO.IN)

GPIO.setup(MOTOR\_L1,GPIO.OUT)

GPIO.setup(MOTOR\_L2,GPIO.OUT)

GPIO.setup(MOTOR\_R1,GPIO.OUT)

GPIO.setup(MOTOR\_R2,GPIO.OUT)

# main loop starts here

while True:

print("---------------------------")

ser = serial.Serial ("/dev/ttyAMA0") # opening serial port

ser.baudrate=9600 # setting baud rate 9600 bits/sec

Distance\_Left = MeasureLeft();

Distance\_Right = MeasureRight();

# ser.write("L",str(int(Distance\_Left)),"R",str(int(Distance\_Right)))

ser.write("L")

ser.write(str(int(Distance\_Left)))

ser.write("R")

ser.write(str(int(Distance\_Right)))

data = ser.read(3)

if data == "F00":

Forward()

print("Forward")

elif data == "L00":

Left()

print("Left")

elif data == "R00":

Right()

print("Right")

elif data == "S00":

Stop()

print("Stop")

elif data == "FF0":

ForwardSlow()

print("Forward Slow")

elif data == "LL0":

LeftSlow()

print("Left Slow")

elif data == "RR0":

RightSlow()

print("Right Slow")

else:

if data[0] == 'F':

y= 10\*int(data[1]) + int(data[2])

print("Forward \_encoder:",y)

Distance(y)

elif data[0] == 'L':

y= 10\*int(data[1]) + int(data[2])

print("Left \_encoder:",y)

LeftTurn(y)

elif data[0] == 'R':

y= 10\*int(data[1]) + int(data[2])

print("Right \_encoder;",y)

RightTurn(y)

ser.close()

GPIO.cleanup()



Figure 14: Our model

**8.Result and Discussions**

Our project has resulted into working model of an automated robot car which can detect obstacle in its pathway and avoid it by turning to a certain angle which is calculated in respect to the obstacle’s width. The robot car avoids obstacles when encountered by deviating from its original pathway. The scope of this project was to create a car that can operate without human support.

**8.1 Evaluation of results**

All the features that were mentioned in the synopsis were implemented in out robot. Currently, the robot satisfies all the hardware and software requirements and functionalities that were mentioned in the scope of the project for this semester. It also has the room for including other functionalities in future.

**8.2 Evaluation of methods**

The methods which were used in this project were very useful. The major issues were we required certain methodologies to proceed further included the mechanical construction on the car, the obstacle avoidance part using image processing and the integration of hardware with software. The development methods which were used for developing object avoidance algorithm were iterative in nature. Testing methodology was based on considering different scenarios of obstacle occurrences.

**8.3 Evaluation of process**

The process of this project can be defined as rapid development. The project was done within a window of three months. The completion of entire work was due to the resilience and hard work of all the group members and our guide. This process was divided into phases and weekly reporting was done and thus the project achieved its completion under the supervision of our guide.

**8.4 Evaluation of learning**

During the project we learned many technologies which were required for its development. Python was learned for coding in Raspberry Pi. OpenCV library was studied for image processing using OpenCV and C++. A bit of Matlab was studied for the same. We learned about Raspberry Pi and how DC motors are used and integrated with Raspberry Pi to build a robot. Software and hardware integration was also studied.

**9. Conclusion**

Autonomous Robo car was implemented to be used as an artificial intelligence. It is an attempt to provide more lavishing and tech-friendly future by building a car that can run without a driver just based on simple algorithms. Obstacle avoidance based on the image that camera captures in real time was the basic idea behind the project. The project substantiates the claim for development of a fully automated car which would require no driver and run on its own without colliding and also control speed based upon its environment.

Knowledge for the project was gathered by surfing various websites. The basics of OpenCV and its integration with C++ was studied using OpenCV documentation available on Opencv.com.

Future Work:

• Implement the entire image processing part in Raspberry Pi rather than laptop.

• Implement the same project on android platform using mobile instead of laptop.

• Include some new functionalities.

**10.References**

[1] Roger Pressman, SOFTWARE ENGINEERING A Practitioner's Approach, New York: McGraw Hill, 2005.

[2] Dharani.S.J1, Anitha.V2. Traffic Density Count by Optical Flow Algorithm using Image Processing, IJIRSET

[3] Image Processing Toolkit (Documentation, Matlab)

[4] <http://in.mathworks.com/help/vision/examples/>detecting-cars-using-gaussian-mixture-models.html