CARGOPORTER(OBJECT TRACKING ROBOT)

A Group Project Report

Submitted

in the partial fulfillment of the requirements for

the award of the degree of

BACHLEOR OF TECHNOLOGY

in

Electronics and Communication Engineering [ECE]

by

AVS SAI TEJA [Roll No.17311A04CQ]

SARGADA PUNEETH [Roll No.17311A04CW]

ABHISHEK KATRAGADDA, [Roll No.17311A04DA]

UNDER THE GUIDANCE OF

Mr.T.Venkata Rao

Associate Professor, Dept. of ECE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SREENIDHI INSTITUTE OF SCIENCE & TECHNOLOGY

Yamnampet, Ghatkesar, Hyderabad – 501 301

2019

SREENIDHI INSTITUTE OF SCIENCE & TECHNOLOGY

Yamnampet, Ghatkesar, Hyderabad – 501 301

CERTIFICATE

This is to certify that the Project Work entitled "**CARGOPORTER(OBJECT TRACKING ROBOT)**" being submitted by

AVS SAI TEJA [Roll No.17311A04CQ]

SARGADA PUNEETH [Roll No.17311A04CW]

ABHISHEK KATRAGADDA, [Roll No.17311A04DA]

in fulfillment for the award of **Bachelor of Technology** in **Electronics and Communication Engineering [ECE], Sreenidhi Institute of Science and Technology,** an autonomous institute under Jawaharlal Nehru Technological University, Telangana, is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in the report have not been submitted to any other University or Institution for the award of any degree or diploma.

Mr.T.Venkata Rao	**(Dr.S.P.V. Subba Rao)**
Project co-ordinator	**Professor & Head, ECE**
Associate Professor, ECE	

DECLARATION

We hereby declare that the work described in this report, entitled **"CARGOPORTER(OBJECT TRACKING ROBOT)"** which is being submitted by me in partial fulfilment for the award of **Bachelor of Technology** in **Electronics and Communication Engineering [ECE], Sreenidhi Institute Of Science & Technology** affiliated to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, Hyderabad (Telangana) -500 085 is the result of investigations carried out by us under the Guidance of **Mr.T.Venkata Rao , Associate Professor, ECE Department**, Sreenidhi Institute Of Science And Technology, Hyderabad. The work is original and has not been submitted for any Degree/Diploma of this or any other university.

Place: Hyderabad

Date: Signature

AVS SAI TEJA [Roll No.17311A04CQ]
SARGADA PUNEETH [Roll No.17311A04CW]
ABHISHEK KATRAGADDA, [Roll No.17311A04DA]

ACKNOWLEDGEMENTS

We are very thankful to **Mr. T. Venkat Rao, Associate Professor, ECE Dept.**, Sreenidhi Institute of Science and Technology, Ghatkesar for providing an initiative to this group project and giving valuable timely suggestions over the work.

We convey our sincere thanks to **Dr. S. P.V. Subba Rao, Head of the Department (ECE)**, Sreenidhi Institute of Science and Technology, Ghatkesar for his kind cooperation in the completion of this work.

We even convey our sincere thanks to **Dr. P. Narsimha Reddy, Executive Director** and **Dr. T. Ch. Shiva Reddy, Principal**, Sreenidhi Institute of Science and Technology, Ghatkesar for their kind cooperation in the completion of the group project.

Finally, we extend our sense of gratitude to all our friends, teaching and non-teaching faculty, who directly or indirectly helped us in this endeavor.

AVS SAI TEJA [Roll No.17311A04CQ]
SARGADA PUNEETH [Roll No.17311A04CW]
ABHISHEK KATRAGADDA, [Roll No.17311A04DA]

ABSTRACT

Human error is one of the main reasons for accidents in everyday tasks. Automation of tasks like lifting, carrying, detecting and following will not only reduce the load on humans but also help with errors humans make. Computer Vision especially in real time, is the starting step for such innovation.

The objective was to make a basic prototype for such a bot which can sense color and shape and follow it. The robot tries to find a color which is hard coded, if it finds a ball of that color it follows it.

Raspberry Pi is used as micro-controller for this project as it gives great flexibility to use Raspberry Pi camera module and allows to code in Python which is very user friendly and OpenCV library, for image analysis.

For controlling the motors, an H-Bridge is used to switch from clockwise to counter-clockwise or to stop the motors. This is integrated via code when direction and speed has to be controlled in different obstacle situations.

CONTENT

LIST OF FIGURES

CHAPTER 1

1.1 INTRODUCTION

Artificial Intelligence(AI) powers the evolution of robotics in the modern era. The core of robotics is AI and computer science is a building block of AI. Highly intelligent robots and robots with AI are fruits of continuous research and years of development. The present project was time and resources bound to build a robot with high level AI; hence, a simple, yet an intelligent robot which could act on real time information was built. Image Processing has been the talk of e-world for some years. Its varied applications have found its uses in almost every field. In Electrical Engineering and Computer Science, image processing is any form of signal processing for which the input is an image, such as photographs or frames of video; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Robotic Vision deals with image processing and Computer Vision. The whole idea behind robotic vision is: viewing an object from the robot's perspective, deciding on object of interest and to act accordingly. It is about giving artificial sight to robots .The implemented robot can be used in various types of industries, and in military applications.

1.2 MOTIVATION

Huge industries involve great deal of transport of goods and supplies from one point to another ,both indoor and outdoor. These might take a lot of time and also can be risky if done manually. However by using an object following bot even greater materials can be transported indoors without any risk. Soldiers in army have to carry heavy cargo and weapons most of the time and the cargo becomes even heavier on various missions. It becomes even more difficult under uneven conditions .But by using cargoporter , the bot can carry most of the cargo on almost any terrain and keep up with the soldiers by only following them simultaneously avoiding obstacles.

1.3 OBJECTIVES

The objective of the project is to create a autonomous robot that can do the following

- Detect a specific object which in this case is a person in real time
- Follow only the required moving object (person in real time)
- Maintain a certain distance from the target object
- Avoid any obstacles that might come in the way

1.4 MAIN CONTRIBUTION

The real time feed was taken in by raspberry pi camera and by using opencv was used to detect the required object. The python code implemented initially divides the video into specific frames per second .then from each frame the object is continuously detected using hsv color space which detects the object based on the objects hue ,saturation and value.Ultrasonic sensors help to detect any obstacles in the path ,and if it does happen based on the code it will retrace its path and move to the point where the object was last seen .

1.5 ORGANIZATION OF THE REPORT

The project report is divided into 6 chapters. This chapter has presented a motivation for and an introduction to artificial intelligence and image processing and also presented introduction about real time computer vision. Chapter 2 gives literature survey and chapter 3 give description about the proposed system used in the project and also its architecture. Chapter 4 gives description about the various software and hardware components used in the making of the project. Chapter 5 deals with design implementation details .Chapter 6 provides necessary data about the simulation and setup of the project and also the design verification. This chapter also deals with the final experimental results.

Summary: This chapter gives introduction to the project ,the main objectives ,major contributions and also the organisation of the report.

CHAPTER 2

LITERATURE SURVEY

To identify and track the real time object is important concept in computer vision. This idea is used for surveillance purpose, monitor the army base, traffic monitoring and human machine interaction. Color-based detections for target following robots were also used by some researchers, since it is one of the feasible decent methods to identify a target. Many microcontrollers are available today but raspberry is the most suited because of its better processing capabilities and a much powerful GPU.

Also raspberry pi 3 supports OpenCV which is useful for real time compuer vision. Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features. Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

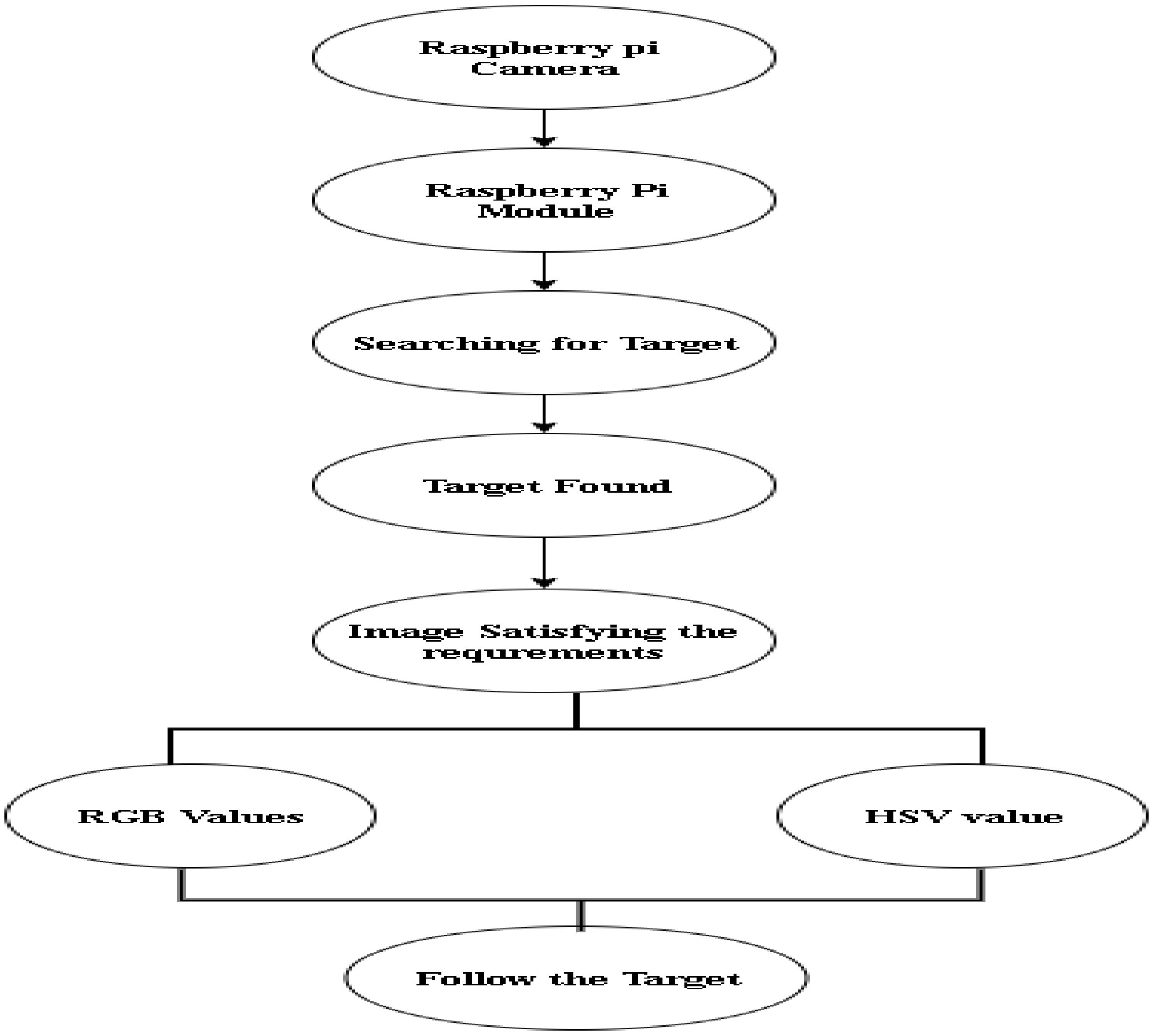
If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”. Digital-Image: An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be. Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

CHAPTER 3

Proposed System Architecture

3.1 The proposed block schematic diagram:

Image Recognizing Processes Of Cargo Porter



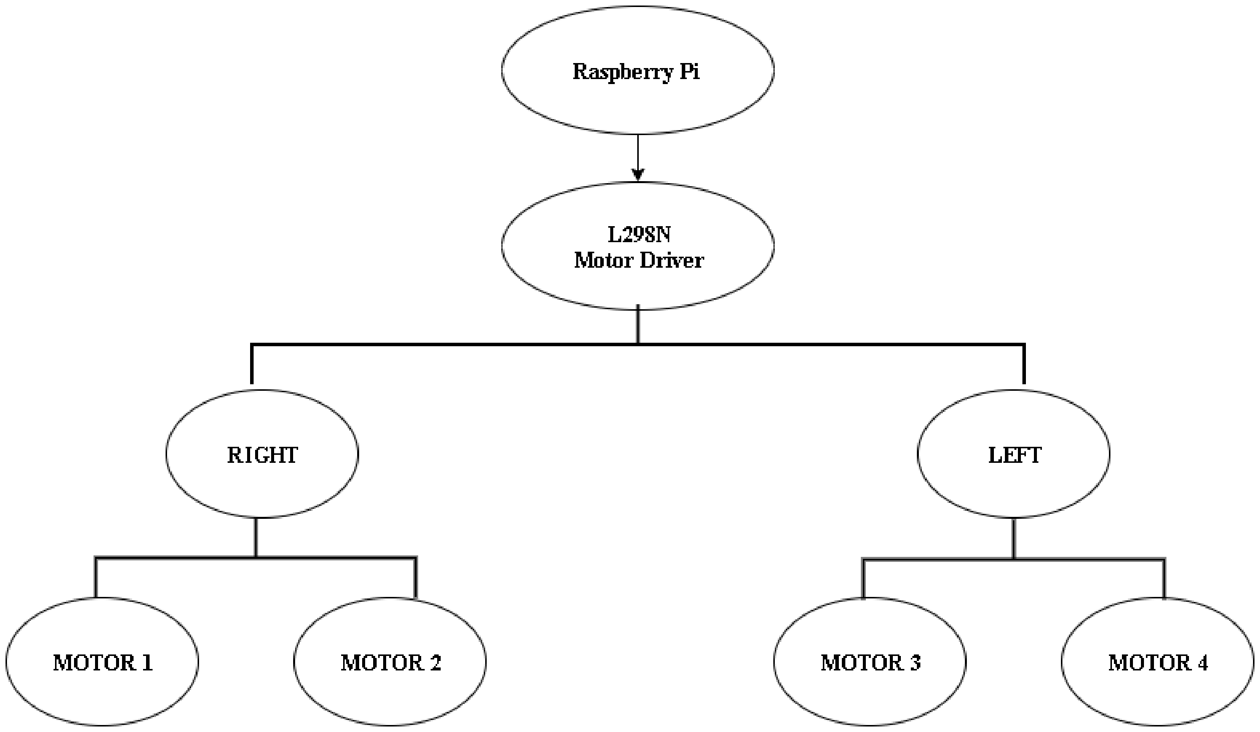
3.1 Block diagram for recognizing an image

The above block diagram represents the processes of recognizing the image/ target which we want to follow. The steps involved in it are explained below.

- Initially the raspberry pi camera searches for the target in its surrounding. The camera specifically searches for the target image which we have provided in the code. The camera sends the captured images to the raspberry pi for further processing.
- The images which the raspberry pi gets from the raspberry pi camera is processed and it checks whether the captured images are meeting the target image requirements like RGB values, HSV etc..The same processes continue until the target is found.

- When the images are reaching the requirements the raspberry pi will activate motors to follow the target where ever it is going.

3.2 Block Diagram Of Raspberry Pi Controlling Motors To Follow The Target:



3.2Raspberry Pi controlling Motors

3.3 Algorithm for Controlling Motors In A Direction:

- **Straight:** Right and Left Motors In clockwise direction.
- **Backward:** Right and Left motors in anticlockwise direction.
- **Right:** Right motors stopped and left side motors running in clockwise direction.
- **Left:** Right motors active running in clockwise direction and left side motors stopped.

CHAPTER 4

Hardware And Software components

4.1 HARDWARE COMPONENTS:

- **Chassis**

A chassis is the load-bearing framework of an artificial object, which structurally supports the object in its construction and function. An example of a chassis is a vehicle frame, the underpart of a motor vehicle, on which the body is mounted; if the running gear such as wheels and transmission, and sometimes even the driver's seat, are included, then the assembly is described as a rolling chassis.

4.1 Chassis

- **Wheels:**

Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. This design is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat, not-so-rugged terrain.

4.2 Wheel

Ultrasonic Sensor:

As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception. An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.

4.3 Ultrasonic sensor

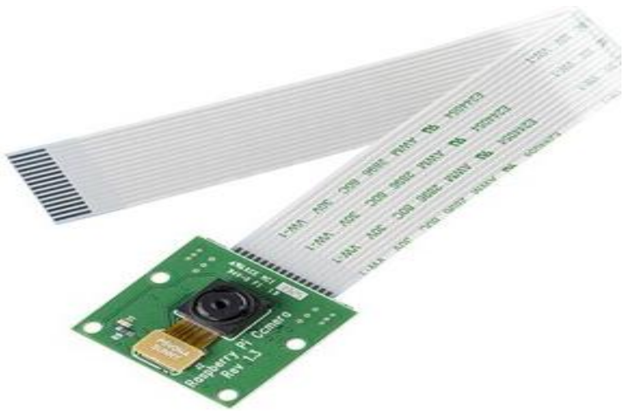
- **Raspberry Pi:**

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote 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.

4.4RaspberryPi

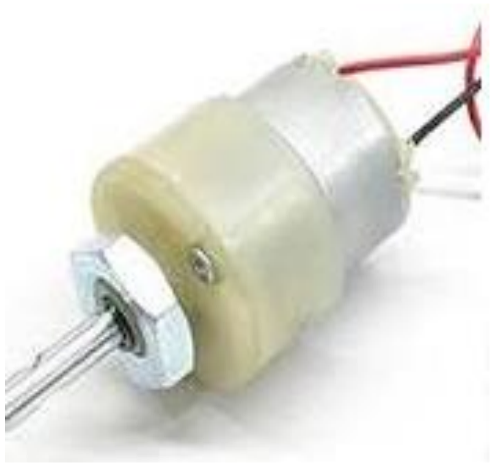
- **Raspberry Pi Camera:**

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It’s easy to use for beginners, but has plenty to offer advanced users if you’re looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. You can also use the libraries we bundle with the camera to create effects.


4.5 Raspberry Pi camera

- **DC Motors:**

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. 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.

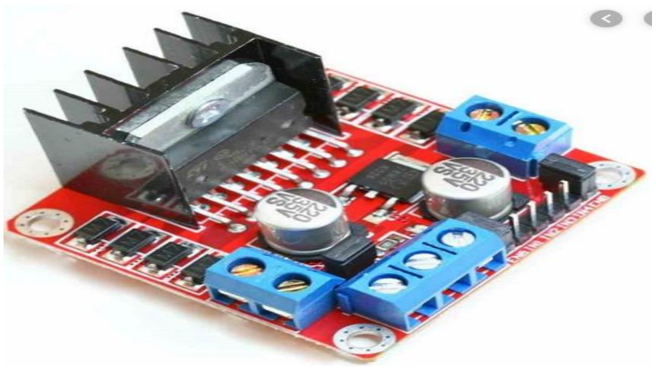


4.6 DC Motor

- **L298N Motor Driver:**

The L298N is an integrated monolithic circuit in a 15- lead Multiwatt and PowerSO20 packages. It is a high voltage , high current dual full-bridge driver de-signed to accept standard TTL logic level sand drive inductive loads such as relays, solenoids, DC and stepping motors.

Two enable inputs are provided to enable or disable the device independently of the in-put signals .The emitters of the lower transistors of each bridge are connected together rand the corresponding external terminal can be used for the connection of an external sensing resistor. An additional Supply input is provided so that the logic works at a lower voltage.



4.7 L298N

4.2 Software:

- **OpenCV:**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

CHAPTER 5

Design Implementation Details

5.1. Chassis Design Details

The chassis selected is made of wood and is in the shape of an octagon. Wood was selected for the chassis as it is light, less expensive, and easy to make changes on compared to other options.
Due to the use of a light chassis, the bot only weighed a little over 2 kilograms. This also proved to be beneficial for the speed of the bot.

5.2. Circuit Design Details

Three ultrasonic sensors were employed to detect any incoming obstacles, they were placed on the front side of the bot. The L298N motor driver was connected to the motors to control their output depending on the input received by the camera and ultrasonic sensors. The camera is connected to a Raspberry Pi 3, and it is used for the processing work.

5.3. Program Design Details

5.3.1 Program

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import RPi.GPIO as GPIO
import time
import cv2
import cv2.cv as cv
import numpy as np


GPIO.setmode(GPIO.BOARD)

TRIGGERL = 29
ECHOL = 31

TRIGGERF = 36
ECHOF = 37

TRIGGERR = 33
ECHOR = 35
```

```
MOTORL1=18  #Left Motor
MOTORL2=22

MOTORR1=19  #Right Motor
MOTORR2=21

# Set pins as output and input
GPIO.setup(TRIGGERL,GPIO.OUT)
GPIO.setup(ECHOL,GPIO.IN)
GPIO.setup(TRIGGERF,GPIO.OUT)
GPIO.setup(ECHOF,GPIO.IN)
GPIO.setup(TRIGGERR,GPIO.OUT)
GPIO.setup(ECHOR,GPIO.IN)

GPIO.output(TRIGGERL, False)
GPIO.output(TRIGGERF, False)
GPIO.output(TRIGGERR, False)

def sonar(TRIGGER,ECHO):
    start=0
    stop=0
    GPIO.setup(TRIGGER,GPIO.OUT)
    GPIO.setup(ECHO,GPIO.IN)


    GPIO.output(TRIGGER, False)


    time.sleep(0.01)

    while distance > 5:

    GPIO.output(TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(TRIGGER, False)
    begin = time.time()
    while GPIO.input(ECHO)==0 and time.time()<begin+0.05:
        start = time.time()

    while GPIO.input(ECHO)==1 and time.time()<begin+0.1:
        stop = time.time()
    elapsed = stop-start
    distance = elapsed * 34000
    distance = distance / 2
    print "Distance : %.1f" % distance
    return distance

GPIO.setup(MOTORL1, GPIO.OUT)
GPIO.setup(MOTORL2, GPIO.OUT)
GPIO.setup(MOTORR1, GPIO.OUT)
GPIO.setup(MOTORR2, GPIO.OUT)

def forward():
    GPIO.output(MOTORL1, GPIO.HIGH)
    GPIO.output(MOTORL2, GPIO.LOW)
    GPIO.output(MOTORR1, GPIO.HIGH)
    GPIO.output(MOTORR2, GPIO.LOW)

def reverse():
    GPIO.output(MOTORL1, GPIO.LOW)
    GPIO.output(MOTORL2, GPIO.HIGH)
    GPIO.output(MOTORR1, GPIO.LOW)
    GPIO.output(MOTORR2, GPIO.HIGH)

def rightturn():
    GPIO.output(MOTORL1,GPIO.LOW)
    GPIO.output(MOTORL2,GPIO.HIGH)
    GPIO.output(MOTORR1,GPIO.HIGH)
    GPIO.output(MOTORR2,GPIO.LOW)

def leftturn():
    GPIO.output(MOTORL1,GPIO.HIGH)
    GPIO.output(MOTORL2,GPIO.LOW)
    GPIO.output(MOTORR1,GPIO.LOW)
    GPIO.output(MOTORR2,GPIO.HIGH)

def stop():
    GPIO.output(MOTORL1,GPIO.LOW)
    GPIO.output(MOTORL2,GPIO.LOW)
    GPIO.output(MOTORR1,GPIO.LOW)
    GPIO.output(MOTORR2,GPIO.LOW)

#Image analysis
def segment_colour(frame):
   hsv_roi =  cv2.cvtColor(frame, cv2.cv.CV_BGR2HSV)
   mask_1 = cv2.inRange(hsv_roi, np.array([160, 160,10]),
np.array([190,255,255]))
   ycr_roi=cv2.cvtColor(frame,cv2.cv.CV_BGR2YCrCb)
   mask_2=cv2.inRange(ycr_roi, np.array((0.,165.,0.)),
np.array((255.,255.,255.)))
```

```
    mask = mask_1 | mask_2
    kern_dilate = np.ones((8,8),np.uint8)
    kern_erode  = np.ones((3,3),np.uint8)
    mask= cv2.erode(mask,kern_erode)
    mask=cv2.dilate(mask,kern_dilate)
    return mask

def find_blob(blob): #returns the red colored circle
    largest_contour=0
    cont_index=0
    contours, hierarchy = cv2.findContours(blob, cv2.RETR_CCOMP,
cv2.CHAIN_APPROX_SIMPLE)
    for idx, contour in enumerate(contours):
        area=cv2.contourArea(contour)
        if (area >largest_contour) :
            largest_contour=area

            cont_index=idx



    r=(0,0,2,2)
    if len(contours) > 0:
        r = cv2.boundingRect(contours[cont_index])

    return r,largest_contour

def target_hist(frame):
    hsv_img=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    hist=cv2.calcHist([hsv_img],[0],None,[50],[0,255])
    return hist

#CAMERA CAPTURE
camera = PiCamera()
camera.resolution = (160, 120)
camera.framerate = 16
rawCapture = PiRGBArray(camera, size=(160, 120))

time.sleep(0.001)



for image in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
      frame = image.array
      frame=cv2.flip(frame,1)
```

```
global centre_x
global centre_y
centre_x=0.
centre_y=0.
hsv1 = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask_red=segment_colour(frame)
loct,area=find_blob(mask_red)
x,y,w,h=loct

#distance coming from front
distanceC = sonar(TRIGGERF,ECHOF)
#distance coming from right
distanceR = sonar(TRIGGERR,ECHOR)
#distance coming from left
distanceL = sonar(TRIGGERL,ECHOL)


if (w*h) < 10:
    found=0
else:
    found=1
    simg2 = cv2.rectangle(frame, (x,y), (x+w,y+h), 255,2)
    centre_x=x+((w)/2)
    centre_y=y+((h)/2)
    cv2.circle(frame,(int(centre_x),int(centre_y)),3,(0,110,255),-1)
    centre_x-=80
    centre_y=6--centre_y
    print centre_x,centre_y
initial=400
flag=0

if(found==0):
    #if object not found, bot will rotate in previous seen direction
    if flag==0:
        rightturn()
        time.sleep(0.05)
    else:
        leftturn()
        time.sleep(0.05)
    stop()
    time.sleep(0.0125)

elif(found==1):
    if(area<initial):
        if(distanceC<10):
                #if object is there, but obstacle in front of object, it avoids
obstacle to reach object
                if distanceR>=8:
                    rightturn()
                    time.sleep(0.00625)
                    stop()
                    time.sleep(0.0125)
                    forward()
                    time.sleep(0.00625)
                    stop()
                    time.sleep(0.0125)
                    while found==0:
                    leftturn()
                    time.sleep(0.00625)
                elif distanceL>=8:
                    leftturn()
                    time.sleep(0.00625)
                    stop()
                    time.sleep(0.0125)
                    forward()
                    time.sleep(0.00625)
                    stop()
                    time.sleep(0.0125)
                    rightturn()
                    time.sleep(0.00625)
                    stop()
                    time.sleep(0.0125)
                else:
                    stop()
                    time.sleep(0.01)
            else:

                forward()
                time.sleep(0.00625)
        elif(area>=initial):
            initial2=6700
            if(area<initial2):
                if(distanceC>10):
                    #move coordinates of object to center of camera's
imaginary axis
                    if(centre_x<=-20 or centre_x>=20):
                        if(centre_x<0):
                            flag=0
                            rightturn()
                            time.sleep(0.025)
                        elif(centre_x>0):
                            flag=1
                            leftturn()
                            time.sleep(0.025)
                    forward()
                    time.sleep(0.00003125)
                    stop()
                    time.sleep(0.00625)
                else:
                    stop()
                    time.sleep(0.01)

            else:
                #if it found the object and it is too close it stops
                stop()
                time.sleep(0.1)

    rawCapture.truncate(0)

    if(cv2.waitKey(1) & 0xff == ord('q')):
        break

GPIO.cleanup()
```

5.3.2. Programs Explanation

In the first lines, all libraries required are imported, and connections are assigned to the sensors and the motor driver.

A function called sonar is defined, here the distance using ultrasonic sensors is determined. First, the trigger pin is set to high for 10 microseconds, then the echo pin is high which receives the pulse. It calculates the time taken to send and receive the pulse and distance is calculated. This distance is halved as it includes distance to reach object and back to sensor.

Functions for directions of the bot are given by influencing the motor control. It is as follows:

Forward: All motors set to move forward.

Left: Right motors set to move forward and left motors move backward.

Right: Left motors set to move forward and right motors move backward.

Backward: All motors set to move backward.

Stop: All motors set to low.

For image analysis, each frame is taken and is masked with the colour red. Then, the noise is eroded and all the major blobs are dilated, this helps with noise reduction. All contours of the object are then found and the largest among them is bound in a rectangle. Then, the rectangle is shown on the main image. The coordinates of the center of the rectangle are found.

Based on the coordinates of the center of the rectangle, the bot will move. If the object is moving towards the left side, the bot will try to bring the object back into focus by moving left; the same thing happens if the object moves right. The bot will stop or slow as the object gets closer.

CHAPTER 6

Simulation and Design Verification

6.1. Experimental Setup

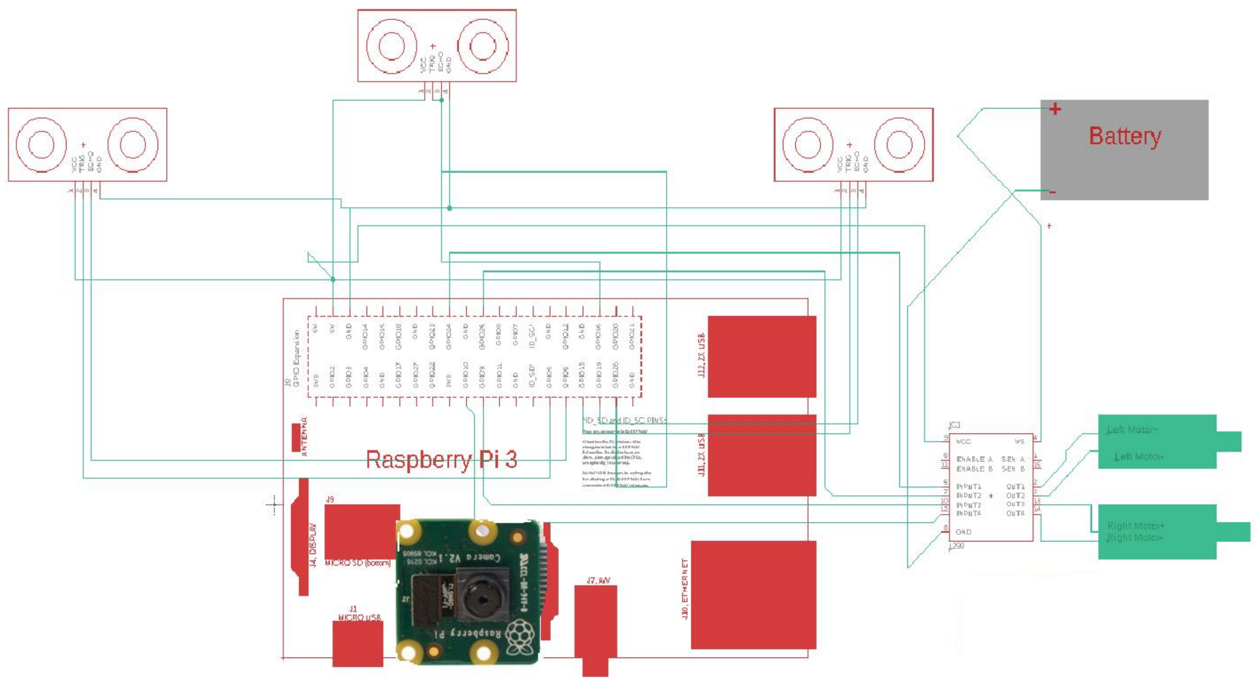


Fig 6.1 Experimental Circuit Setup

6.2. Experimental Results

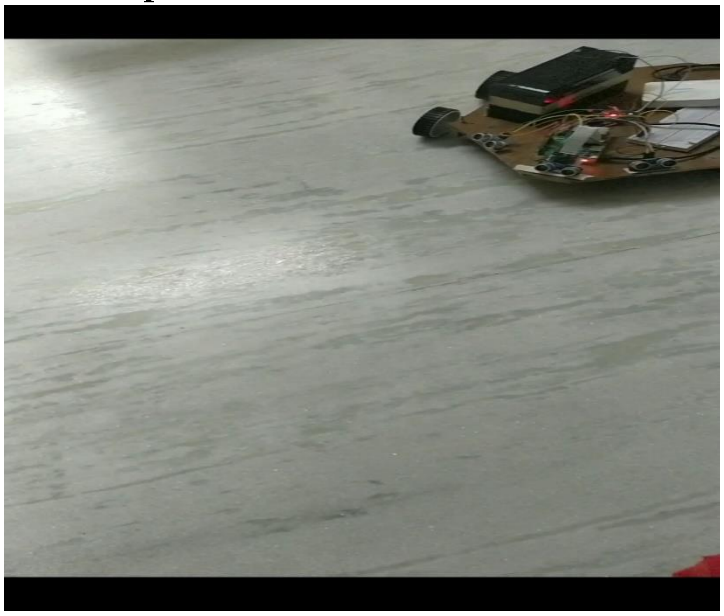


Fig 6.2 Experimental Result 1
Object is at right side.

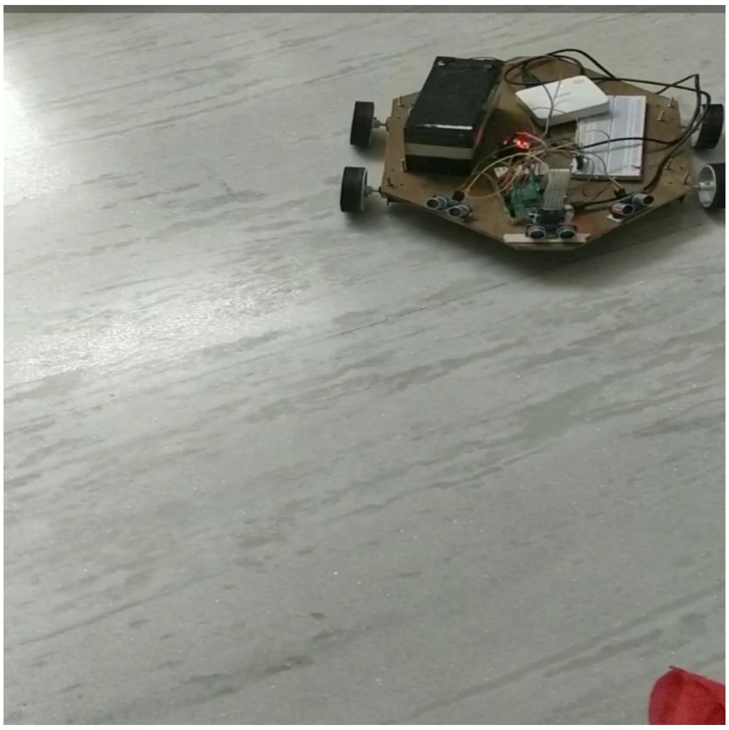


Fig 6.3 Experimental Result 2
The bot moves towards right and starts following.

6.3. Summary

The robot is able to detect the object in real time and follow it. Any obstacles in front of the robot are being avoided. Other than lighting issues there have been no other problems whatsoever. The robot is working as intended.

Conclusion

Automation of tasks like following a specific object in real time will prove to be very useful in various fields, especially in those where there is a higher chance of accidents due to human error. If this were used in warehouses or in transporting heavy cargo through hilly areas, it will not only reduce workload on humans, but also decrease scope for human error.

REFERENCES

- https://machinelearningmastery.com/what-is-computer-vision/
- https://linuxize.com/post/how-to-install-opencv-on-raspberry-pi/
- Puri, Raghav & Gupta, Archit. (2018). Contour, Shape & Color Detection using OpenCV-Python.
- Hani Hunud A Kadouf and Yasir Mohd Mustafah (2013). Colour-based Object Detection and Tracking for Autonomous Quadrotor UAV.
- X.L. Wang, M.Q. Zhou, and G.H. Geng, “An Approach of Vehicle Plate Extract Based on HSV Color Space”, Computer Engineering, 30(17), 2004.9