Visvesvaraya Technological University

Belgaum, Karnataka-590 018

A Project Report on

"Voice Recognition and Object Identification Robot"

Project Report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering
In
Electrical & Electronics Engineering

Submitted by

Ashwin Sundar Ram (1CR17EE011)
Delip Antonio (1CR17EE019)
Souptik Mukherjee (1CR17EE068)
Vaisakh Anil (1CR17EE080)

Under the Guidance of

Mr. P Velrajkumar
Associate Professor , Department of Electrical & Electronics Engineering
CMR Institute of Technology

CMR Institute of Technology, Bengaluru-560 037

Department of Electrical & Electronics Engineering

2020-2021

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

AECS Layout, Bengaluru-560 037

Certificate

Certified that the project work entitled **"Voice Recognition and Object Identification Robot"** carried out by Mr. Ashwin Ram Sundar, USN 1CR17EE011; Mr. Delip Antonio , USN 1CR17EE019; Mr. Souptik Mukherjee, USN 1CR17EE068; Mr. Vaisakh Anil, USN 1CR17EE080 are bonafied students of CMR Institute of Technology, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Electrical & Electronics Engineering of the Visvesvaraya Technological University, Belgaum, during the year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of the Guide	*Signature of the HOD*	*Signature of the Principal*
----------	----------	----------
Mr. P Velrajkumar, Associate Professor EEE Department CMRIT, Bengaluru	Dr. K. Chitra Professor & HOD EEE Department CMRIT, Bengaluru	Dr. Sanjay Jain Principal, CMRIT, Bengaluru

External Viva

Name of the Examiners	Signature & Date
1.	
2.	

CMR INSTITUTE OF TECHNOLOY
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
AECS Layout, Bengaluru-560 037

DECLARATION

We, [Mr. Ashwin Ram Sundar **(1CR17EE011),** Mr. Delip Antonio **(1CR17EE019),** Mr. Souptik Mukherjee **(1CR17EE068),** Mr. Vaisakh Anil **(1CR17EE080)],** hereby declare that the report entitled **"Voice Recognition and Object Identification Robot"** has been carried out by us under the guidance of **P Velrajkumar,** Associate Professor, Department of Electrical & Electronics Engineering, CMR Institute of Technology, Bengaluru, in partial fulfillment of the requirement for the degree of **BACHELOR OF ENGINEERING in ELECTRICAL & ELECTRONICS ENGINEERING**, of Visveswaraya Technological University, Belagaum during the academic year 2020-21. The work done in this report is original and it has not been submitted for any other degree in any university.

Place: Bengaluru
Date:

Ashwin Ram (1CR17EE011)

Delip Antonio (1CR17EE019)

Souptik Mukherjee (1CR17EE068)

Vaisakh Anil (1CR17EE080)

Abstract

In this project, we try to implement a voice-controlled object identification robot. A speech recognition system is used to recognize a set of predefined commands such as forward, backward, left, right and stop.

A speech recognition system is used to recognize a set of predefined commands. The robot navigates its way as per the voice-command signal, whilst simultaneously scanning its environment for the required object using object identification.

The objective of this project is to design and build a low cost, fully functional voice-controlled assistant that is capable of locating and identifying the object as instructed by voice command.

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people, who are responsible for the completion of the project and who made it possible, because success is outcome of hard work and perseverance, but stead fast of all is encouraging guidance. So with gratitude we acknowledge all those whose guidance and encouragement served us to motivate towards the success of the project work.

We take great pleasure in expressing our sincere thanks to ***Dr. Sanjay Jain****,* ***Principal, CMR Institute of Technology, Bengaluru*** *for providing an excellent academic environment in the college and for his continuous motivation towards a dynamic career. We would like to profoundly thank* ***Dr. B Narasimha Murthy****, Vice-principal of CMR Institute of Technology and the whole* ***Management*** *for providing such a healthy environment for the successful completion of the project work.*

We would like to convey our sincere gratitude to ***Dr. K Chitra****,* ***Head of Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for her invaluable guidance and encouragement and for providing good facilities to carry out this project work.*

We would like to express our deep sense of gratitude to ***Mr. P Velrajkumar, Assistant Professor, Electrical and Electronics Engineering, CMR Institute of Technology, Bengaluru*** *for his/her exemplary guidance, valuable suggestions, expert advice and encouragement to pursue this project work.*

We are thankful to all the faculties and laboratory staffs of ***Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for helping us in all possible manners during the entire period.*

Finally, we acknowledge the people who mean a lot to us, our parents, for their inspiration, unconditional love, support, and faith for carrying out this work to the finishing line. We want to give special thanks to all our friends who went through hard times together, cheered us on, helped us a lot, and celebrated each accomplishment.

Lastly, to the ***Almighty****, for showering His Blessings and to many more, whom we didn't mention here.*

CONTENTS

Title Page	i
Certificate	ii
Declaration	iii
Abstract	iv
Acknowledgements	v
Contents	vi-vii
List of Figures	viii
Chapter 1 : INTRODUCTION	1
Chapter 2 : LITERATURE REVIEW	2
Chapter 3 : METHODOLOGY	9
Chapter 4 : CODES	13
Chapter 5 : CONCLUSION	39
Chapter 6 : REFERENCES	40

LIST OF FIGURES

Figure 1 *Basic Object Detection Model*

Figure 2 *Working of the Object Detection System*

Figure 3 *MS COCO Dataset Categories*

Figure 4 *Datasets Related to Object Recognition groups*

Figure 5 *Basic Tasks of Robot*

Figure 6 *Working Block Diagram of the Robot*

Figure 7 *Flowchart of Working : Part A*

Figure 8 *Flowchart of Working : Part B*

Figure 9 *Output from the Webcam interfaced with RPi*

Figure 10 *Output from the Webcam interfaced with RPi*

CHAPTER 1

INTRODUCTION

Vision based control of the robotic system is the use of the visual sensors as a feedback information to control the operation of the robot. It is a well known fact that disabled people face different challenges and difficulties regarding their physical movements. Due to limited options available, they are often restricted in their movement and may need to depend on other person's assistance. But this may be quite inconvenient for the person assisting as well as for the person being assisted. In such scenarios, the robots whose operation is controlled by human voice-commands, can provide a potential solution to their problems.

Voice-controlled robots can also be useful in applications other than assisting disabled people. For example, in military operations such voice-controlled robots can be moved differentially by tracking a desired object. The key idea behind such applications is assisting the humans in reducing their manual efforts and the risk factors.

The objective of this project is to design and build a low cost, fully functional voice-controlled assistant that is capable of aiding the visually impaired by locating and identifying the required object as instructed by voice command.

CHAPTER 2

LITERATURE REVIEW

Research Paper 1: **A review and an approach for object detection in images January 2017 International Journal of Computational Vision and Robotics 7(1/2):196**

An object detection system finds objects of the real-world present either in a digital image or a video, where the object can belong to any class of objects namely humans, cars, etc. In order to detect an object in an image or a video the system needs to have a few components in order to complete the task of detecting an object, they are a model database, a feature detector, a hypothesizer and a hypothesizer verifier.

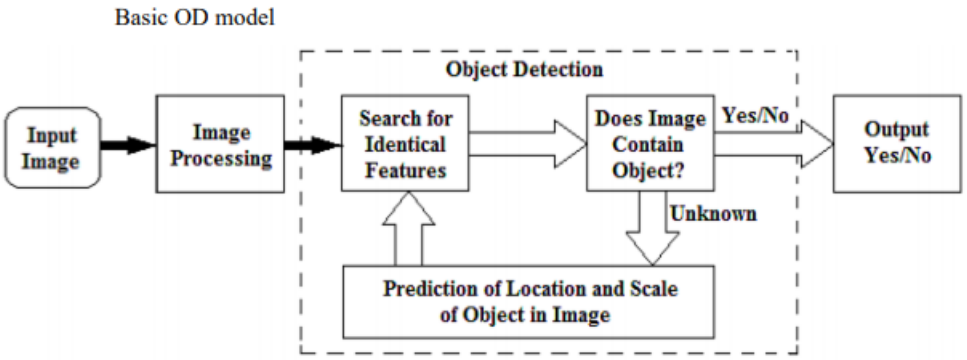


Figure 1

Computer Vision (CV): Computer Vision is using a computer to process a camera stream (2D mathematical values) to get a higher-level understanding of what the camera is capturing.

Machine learning (ML): Use thousands of images of an object --let's assume a cup-- and use this data to calculate a model (picture) for what the average cup looks like based on all the images in the data-set. More pictures = more accurate model.

This particular robot combines Machine Learning with Computer Vision. This means it passes the webcam stream through a machine-learnt model in order to detect objects in the frame. For example, based on the 'average cup model' ML example, the computer can look at the camera's video frame and try to fit the average cup image on to your new image. If it fits within a certain degree of accuracy, the new image will be labeled as a cup. By tracing where exactly this object is detected via the camera frame, we get object detection.

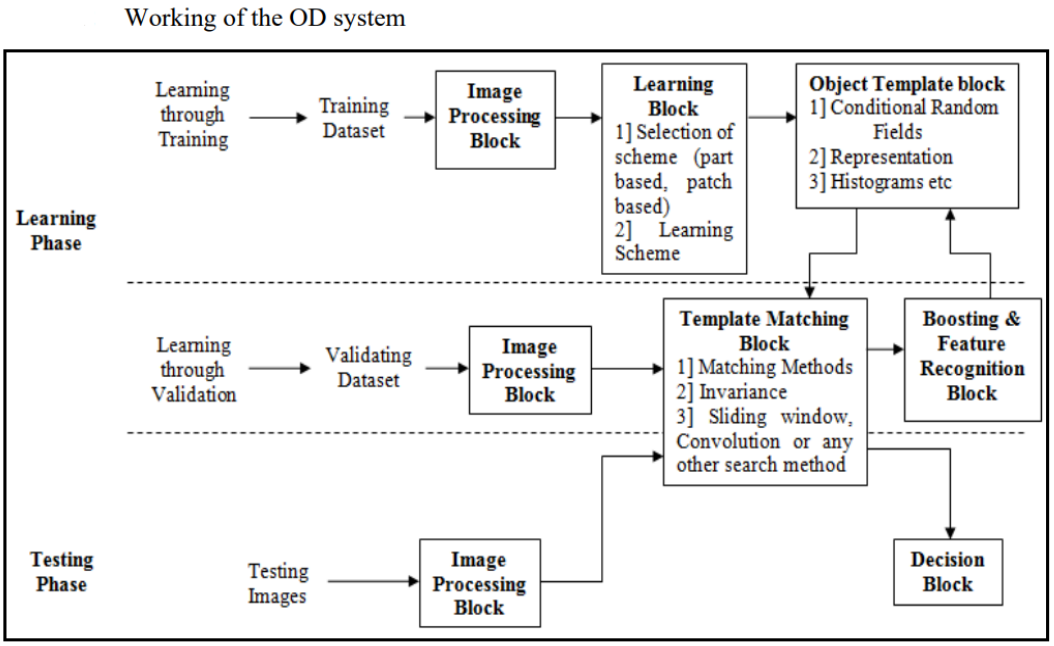
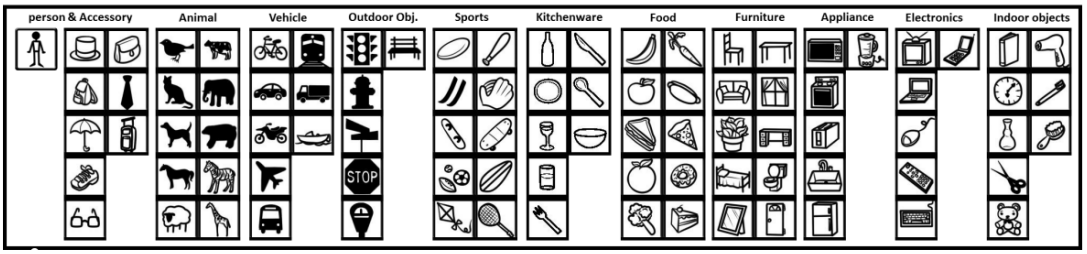


Figure 2

Research Paper 2**: Microsoft COCO: Common Objects in Context: Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár.**

Microsoft presented a new dataset with the goal of advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. This is achieved by gathering images of complex everyday scenes containing common objects in their natural context. Objects are labeled using per-instance segmentations to aid in precise object localization. Our dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images, the creation of our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation. We present a detailed statistical analysis of the dataset in comparison to PASCAL, ImageNet, and SUN.



Icons of 91 categories in the MS COCO dataset grouped by 11 super-categories. We use these icons in our annotation pipeline to help workers quickly reference the indicated object category.

Figure 3

Datasets related to object recognition can be roughly split into three groups: those that primarily address object classification, object detection and semantic scene labeling.

i.) **Image Classification**: The task of object classification requires binary labels indicating whether objects are present in an image. Early datasets of this type comprised images containing a single object with blank backgrounds, such as the MNIST handwritten digits or COIL household objects

ii.) **Object detection**: Detecting an object entails both stating that an object belonging to a specified class is present, and localizing it in the image. The location of an object is typically represented by a bounding box. Early algorithms focused on face detection using various ad hoc datasets. Later, more realistic and challenging face detection datasets were created.

iii.) **Semantic scene labeling**: The task of labeling semantic objects in a scene requires that each pixel of an image be labeled as belonging to a category, such as sky, chair, floor, street, etc. In contrast to the detection task, individual instances of objects do not need to be segmented. This enables the labeling of objects for which individual instances are hard to define, such as grass, streets, or walls. Datasets exist for both indoor and outdoor scenes.

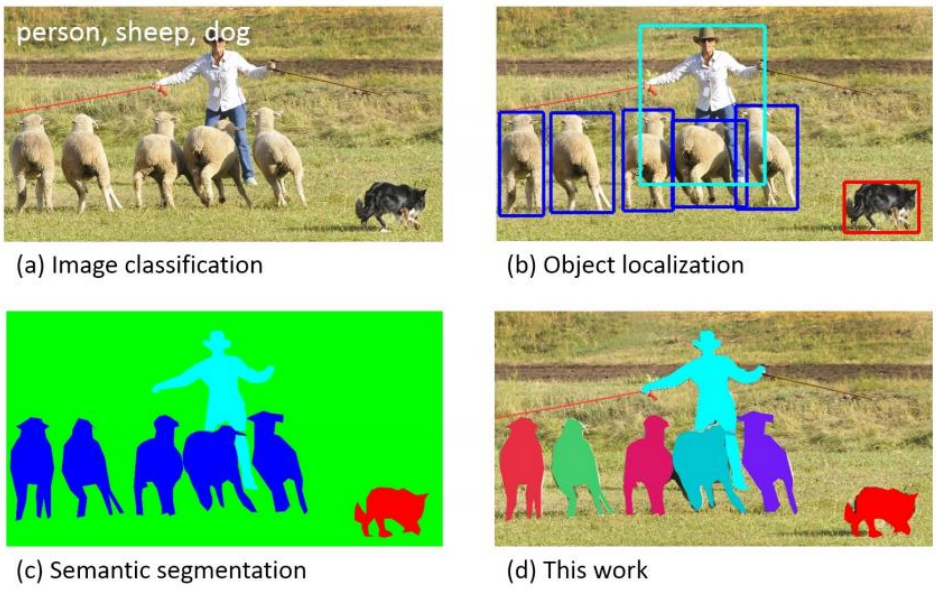


Figure 4

Research Paper 3: **Kannan, K. & Selvakumar, J. (2015). Arduino Based Voice Controlled Robot", International Research Journal of Engineering and Technology (IRJET), Vol. 02, p-ISSN: 2395-0072, e-ISSN: 2395-0056.**

Voice Controlled Robot (VCR) is a mobile robot whose motions can be controlled by the user by giving specific voice commands. The speech is received by a microphone and processed by the voice module. When a command for the robot is recognized, then voice module sends a command message to the robot's microcontroller. The microcontroller analyzes the message and takes appropriate actions.

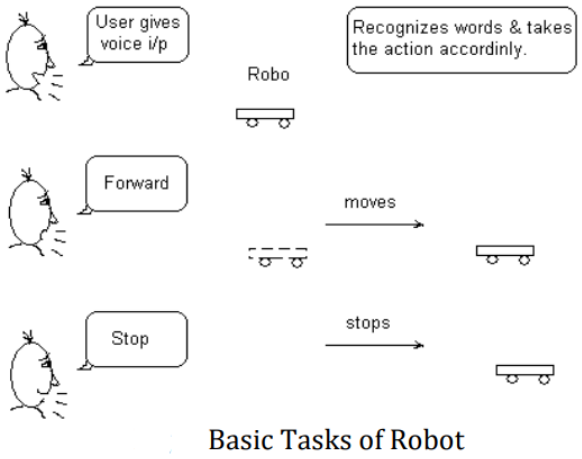


Figure 5

There is camera which is mounted in the head of the robot will give live transmission and recording of the area. The speech recognition circuit functions independently from the robot's main intelligence [central processing unit (CPU)]. This is a good thing because it doesn't take any of the robot's main CPU processing power for word recognition. The CPU must merely poll the speech circuit's recognition lines occasionally to check if a command has been issued to the robot.

How Speech Recognition Works

Speech recognition has its roots in research done at Bell Labs in the early 1950s. Early systems were limited to a single speaker and had limited vocabularies of about a dozen words. Modern speech recognition systems have come a long way since their ancient counterparts. They can recognize speech from multiple speakers and have enormous vocabularies in numerous languages.

The first component of speech recognition is, of course, speech. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text.

Most modern speech recognition systems rely on what is known as a Hidden Markov Model (HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process—that is, a process in which statistical properties do not change over time.

In a typical HMM, the speech signal is divided into 10-millisecond fragments. The power spectrum of each fragment, which is essentially a plot of the signal's power as a function of frequency, is mapped to a vector of real numbers known as cepstral coefficients. The dimension of this vector is usually small—sometimes as low as 10, although more accurate systems may have dimension 32 or more. The final output of the HMM is a sequence of these vectors.

To decode the speech into text, groups of vectors are matched to one or more phonemes—a fundamental unit of speech. This calculation requires training, since the sound of a phoneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.

CHAPTER 3

METHODOLOGY

The object detection model algorithm runs very similarly to the face detection. Instead of using the 'Face Detect' model, we use the COCO model which can detect 90 objects listed. If the object being detected is to the right of the camera frame, the robot moves right to center the object, and if the object is to the left of the camera frame, the robot moves left to center the object.

The movement of the robot is controlled by using voice-commands, which are given using a smart mobile phone to an Android OS based platform.

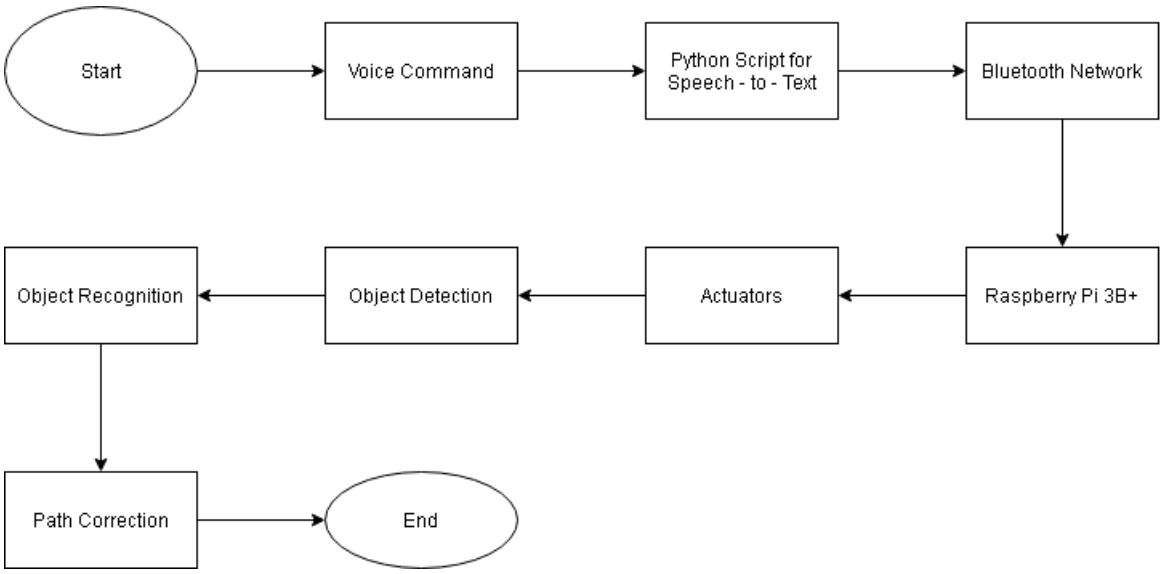


Figure 6

Bluetooth technology exchanges data over a short range but is very proficient way of communicating between two devices such as Rasberry Pi and python script. Data packages are sent and received through Bluetooth channel. It is essential for robots to take commands without any delay so we will use Bluetooth as the main communication method. In daily life such robots can be used for navigation and for control guidance to a certain position.

With the help of the two basic functions which are voice recognition and Bluetooth communication the robot can be used for variable purposes and application commercially and domestically as mentioned above. It is vital to create more technological advances in voice recognition systems to enhance the efficiency of such robots.

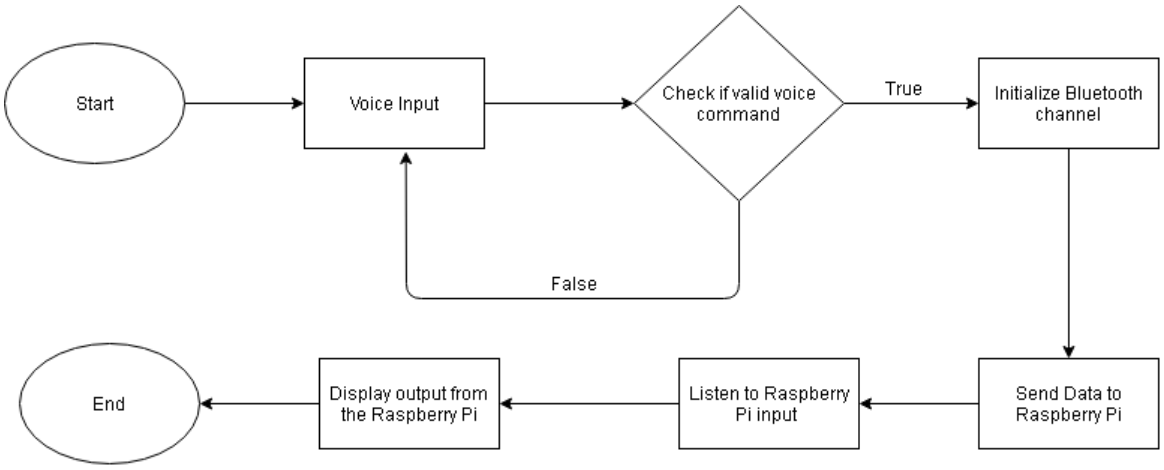


Figure 7

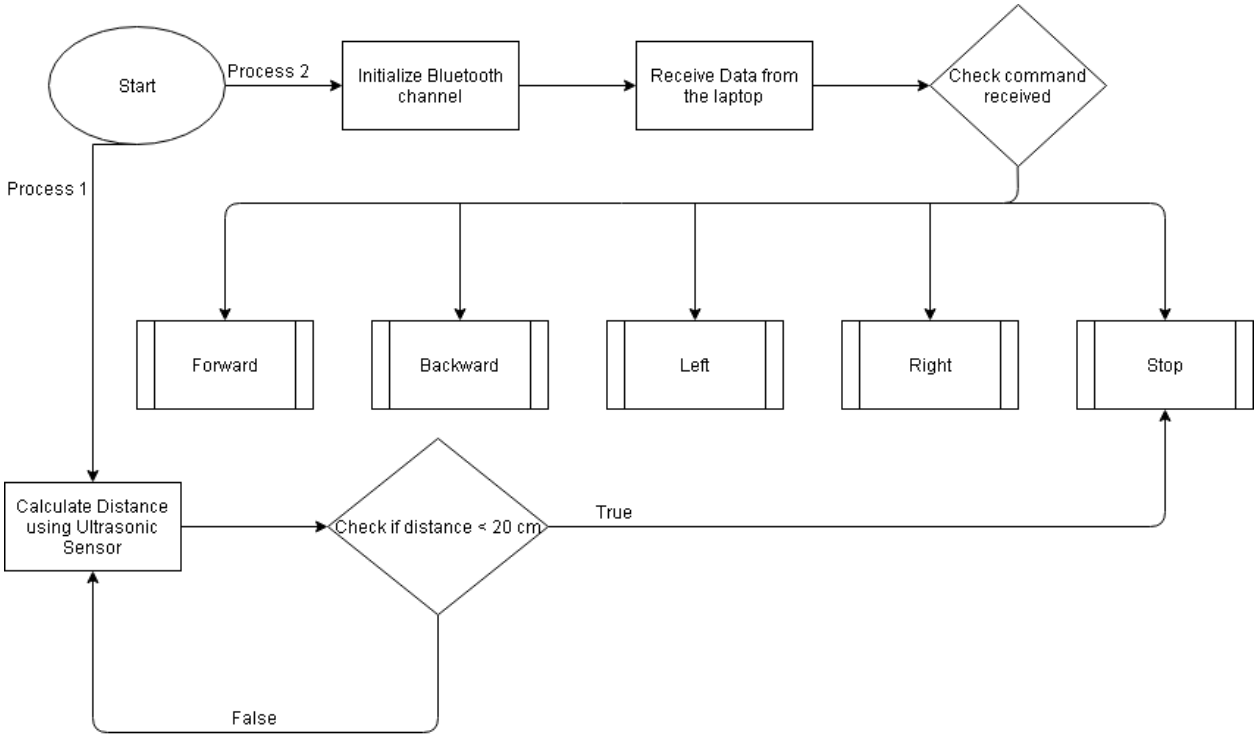

Start
Process 2
Initialize Bluetooth channel
Receive Data from the laptop
Check command received
Process 1
Forward
Backward
Left
Right
Stop
Calculate Distance using Ultrasonic Sensor
Check if distance < 20 cm
True
False

Figure 8

Tools Used

- **Software**

  Python

- **Hardware**

  Rasberry Pi

  Camera Module

  DC Motors

  L298N DC Motor Driver

  Bluetooth Module HC-05

CHAPTER 4

CODES

Codes used in laptop for speech recognition and sending and receiving of data from the laptop and Raspberry Pi 3b+

Code 1:

```
import socket

import voiceRecBlue as vrb

import time


serverMacAddress = 'b8:27:eb:49:6a:de'


print("Starting sending data")

i=1

while(1):

    port = 4

    s = socket.socket(socket.AF_BLUETOOTH, socket.SOCK_STREAM,
socket.BTPROTO_RFCOMM)

    s.connect((serverMacAddress ,port))

    try:

        text = vrb.convertTTS()

        print(text)

        # while text!='':

        #     if text == "quit":
```

```
        #         break
        #     s.send(bytes(text, 'UTF-8'))
        # time.sleep(5)
        if( text == 'forward'):
            s.send(bytes(text, 'UTF-8'))
        if( text == 'backward'):
            s.send(bytes(text, 'UTF-8'))
        if( text == 'left'):
            s.send(bytes(text, 'UTF-8'))
        if( text == 'right'):
            s.send(bytes(text, 'UTF-8'))
        if( text == 'stop'):
            s.send(bytes(text, 'UTF-8'))
        if( text == 'quit'):
            print("Program is closing now")
            s.close()
            exit()
        text=''
        print(f'Execution {i}')
        i+=1
    except KeyboardInterrupt:
        text='term'
        s.send(bytes(text,'UTF-8'))
# s.close()
```

Code 2:

```
# Python program to translate

# speech to text and text to speech



import speech_recognition as sr

import pyttsx3


# Initialize the recognizer


# Function to convert text to

# speech

def SpeakText(command):


    # Initialize the engine

    engine = pyttsx3.init()

    engine.say(command)

    engine.runAndWait()



# Loop infinitely for user to

# speak

def convertTTS():
```

```
# Exception handling to handle
  # exceptions at the runtime
  try:
    r = sr.Recognizer()
    # use the microphone as source for input.
    with sr.Microphone() as source2:

      # wait for a second to let the recognizer
      # adjust the energy threshold based on
      # the surrounding noise level
      r.adjust_for_ambient_noise(source2, duration=0.4)

      #listens for the user's input
      audio2 = r.listen(source2)

      # Using ggogle to recognize audio
      MyText = r.recognize_google(audio2)
      MyText = MyText.lower()
      return MyText

  except sr.RequestError as e:
      return("Could not request results; {0}".format(e))
```

```
except sr.UnknownValueError:
    return("unknown error occured")
```

Codes used in Raspberry Pi for the movement and object detection

Code 3:

```
import RPi.GPIO as GPIO
import time

in1 = 24
in2 = 23
en = 25

in3 = 22
in4 = 27
enb = 4

temp1=1
temp2=1

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
```

```
GPIO.setup(en,GPIO.OUT)


GPIO.setup(in3,GPIO.OUT)

GPIO.setup(in4,GPIO.OUT)

GPIO.setup(enb,GPIO.OUT)


GPIO.output(in1,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)


p=GPIO.PWM(en,1000)

q=GPIO.PWM(enb,1000)


p.start(100)

q.start(100)


def ultrasonic():

    GPIO.setmode(GPIO.BCM)

    TRIG1 = 5

    ECHO1 = 6


    #print ("Distance Measurement In Process")

    GPIO.setwarnings(False)
```

```
    GPIO.setup(TRIG1, GPIO.OUT)

    GPIO.output(TRIG1, False)


    GPIO.setup(ECHO1, GPIO.IN)

    #print ("Waiting For Sensor1 To Settle")

    time.sleep(.1)

    GPIO.output(TRIG1, True)

    time.sleep(0.00001)

    GPIO.output(TRIG1, False)


    while GPIO.input(ECHO1) == 0:

        pass

        pulse_start1 = time.time()


    while GPIO.input(ECHO1) == 1:

        pass

        pulse_end1 = time.time()


    pulse_duration1 = pulse_end1 - pulse_start1


    distance1 = pulse_duration1 * 17150

    distance1= round(distance1, 1)

    return distance1
```

```
def stop():
    GPIO.output(in1,GPIO.LOW)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,GPIO.LOW)

pin_list = [in1, in2, in3, in4]

def motor_control(motor_status, di):
    if(di > 10):
        j=0
        for i in motor_status:
            if(i):
                GPIO.output(pin_list[j],GPIO.HIGH)
            else:
                GPIO.output(pin_list[j],GPIO.LOW)
            j+=1
    else:
        stop()

def move(data):
    distance=ultrasonic()
    print(distance)
    if(data == b'forward'):
```

```
        ms=[0,1,1,0]
        motor_control(ms,distance)
        print('Done forward')
    elif(data == b'backward'):
        ms =[1,0,0,1]
        motor_control(ms,distance)
        print('Done backward')
    elif(data == b'left'):
        ms =[0,1,0,0]
        motor_control(ms,distance)
        time.sleep(6)
        stop()
        print('Done left')
    elif(data == b'right'):
        ms =[0,0,1,0]
        motor_control(ms,distance)
        time.sleep(6)
        stop()
        print('Done right')
    elif(data == b'stop'):
        stop()
        print('Done stop')
```

Code 4:

```
import bluetooth

import multiprocessing

import integratedv2 as iv2

import time

import socket

import TFLite_detection_webcam as tf



hostMACAddress = 'b8:27:eb:49:6a:de'    # The MAC address of a Bluetooth adapter on the server. The server might have multiple Bluetooth adapters.

port =  420# 1 is an arbitrary choice. However, it must match the port used by the client.

backlog = 1

size = 1024


s = bluetooth.BluetoothSocket( bluetooth.RFCOMM )

s.bind((hostMACAddress,port))

s.listen(backlog)


flag = 0


def voiceRecognition():
```

```
    while(1):
        try:
            client, address = s.accept()
            data = client.recv(size)
            if data:
                if data == b'term':
                    iv2.stop()
                    flag = 1
                else:
                    print(data)
                    client.send(data)
                    iv2.move(data)

        except:
            continue

def distanceCalculator():
    while(1):
        dis = iv2.ultrasonic()
        print(dis)
        if( dis < 20):
            iv2.stop()
            label = tf.objRec()
            print(label)
```

```
        time.sleep(5)
        ms = [1,0,0,1]
        distance = 30
        iv2.motor_control(ms,distance)
        time.sleep(3)
        print('sleep done')
        iv2.stop()


p1 = multiprocessing.Process(target = distanceCalculator)
p2 = multiprocessing.Process(target = voiceRecognition)


def doWork():
    p2.start()
    p1.start()
    while(flag == 0):
        try:
            if(flag == 1):
                p1.terminate()
                p2.terminate()
                s.close()
                exit()

        except KeyboardInterrupt:
            print("Termination of Program")
```

```
        p1.terminate()

        p2.terminate()

        s.close()

        exit()


doWork()
```

Code 5 : Object Detection using USB Webcam interfaced with Raspberry Pi

######## Webcam Object Detection Using Tensorflow-trained Classifier #########

This program uses a TensorFlow Lite model to perform object detection on a live webcam

feed. It draws boxes and scores around the objects of interest in each frame from the

webcam. To improve FPS, the webcam object runs in a separate thread from the main program.

This script will work with either a Picamera or regular USB webcam.

Import packages

import os

import argparse

import cv2

import numpy as np

import sys

import time

from threading import Thread

import importlib.util

def objRec():

    # Define VideoStream class to handle streaming of video from webcam in separate processing thread

```
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

    # Variable to control when the camera is stopped
        self.stopped = False

    def start(self):
    # Start the thread that reads frames from the video stream
        Thread(target=self.update,args=()).start()
        return self

    def update(self):
        # Keep looping indefinitely until the thread is stopped
```

```
            while True:
                # If the camera is stopped, stop the thread
                if self.stopped:
                    # Close camera resources
                    self.stream.release()
                    return

                # Otherwise, grab the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()

        def read(self):
        # Return the most recent frame
            return self.frame

        def stop(self):
        # Indicate that the camera and thread should be stopped
            self.stopped = True

    # Define and parse input arguments
    parser = argparse.ArgumentParser()
    #parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                        #required=True)
    parser.add_argument('--graph', help='Name of the .tflite file, if different than
detect.tflite',
```

```
                    default='detect.tflite')

    parser.add_argument('--labels', help='Name of the labelmap file, if different
than labelmap.txt',

                    default='labelmap.txt')

    parser.add_argument('--threshold', help='Minimum confidence threshold for
displaying detected objects',

                    default=0.5)

    parser.add_argument('--resolution', help='Desired webcam resolution in
WxH. If the webcam does not support the resolution entered, errors may
occur.',

                    default='640x480')

    parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to
speed up detection',

                    action='store_true')


    args = parser.parse_args()

    args.modeldir = 'Sample_TFLite_model'

    MODEL_NAME = args.modeldir

    GRAPH_NAME = args.graph

    LABELMAP_NAME = args.labels

    min_conf_threshold = float(args.threshold)

    resW, resH = args.resolution.split('x')

    imW, imH = int(resW), int(resH)

    use_TPU = args.edgetpu


    # Import TensorFlow libraries
```

```
    # If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular tensorflow

    # If using Coral Edge TPU, import the load_delegate library

    pkg = importlib.util.find_spec('tflite_runtime')

    if pkg:

        from tflite_runtime.interpreter import Interpreter

        if use_TPU:

            from tflite_runtime.interpreter import load_delegate

    else:

        from tensorflow.lite.python.interpreter import Interpreter

        if use_TPU:

            from tensorflow.lite.python.interpreter import load_delegate


    # If using Edge TPU, assign filename for Edge TPU model

    if use_TPU:

        # If user has specified the name of the .tflite file, use that name, otherwise use default 'edgetpu.tflite'

        if (GRAPH_NAME == 'detect.tflite'):

            GRAPH_NAME = 'edgetpu.tflite'


    # Get path to current working directory

    CWD_PATH = os.getcwd()


    # Path to .tflite file, which contains the model that is used for object detection

    PATH_TO_CKPT                                                    =
```

```
os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)


    # Path to label map file
    PATH_TO_LABELS                                                =
os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)


    # Load the label map
    with open(PATH_TO_LABELS, 'r') as f:
        labels = [line.strip() for line in f.readlines()]


    # Have to do a weird fix for label map if using the COCO "starter model"
from
    # https://www.tensorflow.org/lite/models/object_detection/overview
    # First label is '???', which has to be removed.
    if labels[0] == '???':
        del(labels[0])


    # Load the Tensorflow Lite model.
    # If using Edge TPU, use special load_delegate argument
    if use_TPU:
        interpreter = Interpreter(model_path=PATH_TO_CKPT,

experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
        print(PATH_TO_CKPT)
    else:
```

```
    interpreter = Interpreter(model_path=PATH_TO_CKPT)


interpreter.allocate_tensors()


# Get model details

input_details = interpreter.get_input_details()

output_details = interpreter.get_output_details()

height = input_details[0]['shape'][1]

width = input_details[0]['shape'][2]


floating_model = (input_details[0]['dtype'] == np.float32)


input_mean = 127.5

input_std = 127.5


# Initialize frame rate calculation

frame_rate_calc = 1

freq = cv2.getTickFrequency()


# Initialize video stream

videostream = VideoStream(resolution=(imW,imH),framerate=30).start()

time.sleep(1)


#for frame1 in camera.capture_continuous(rawCapture,
```

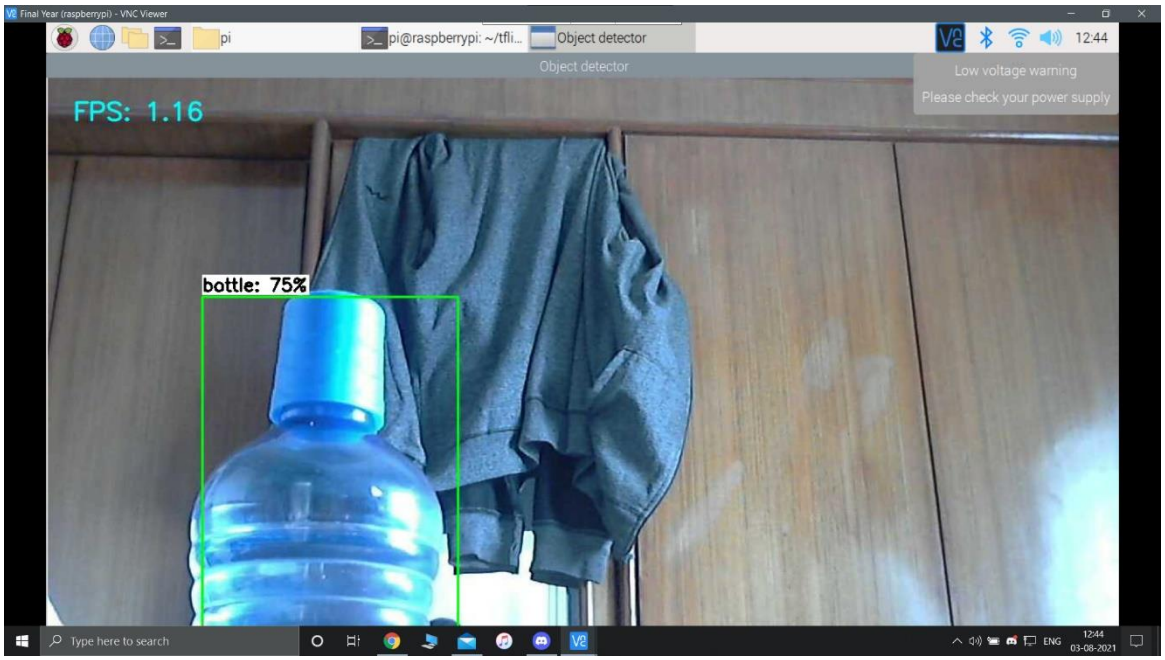
```
format="bgr",use_video_port=True):
   while True:

      # Start timer (for calculating frame rate)
      t1 = cv2.getTickCount()

      # Grab frame from video stream
      frame1 = videostream.read()

      # Acquire frame and resize to expected shape [1xHxWx3]
      frame = frame1.copy()
      frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
      frame_resized = cv2.resize(frame_rgb, (width, height))
      input_data = np.expand_dims(frame_resized, axis=0)

      # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
      if floating_model:
         input_data = (np.float32(input_data) - input_mean) / input_std

      # Perform the actual detection by running the model with the image as input
      interpreter.set_tensor(input_details[0]['index'],input_data)
      interpreter.invoke()
```

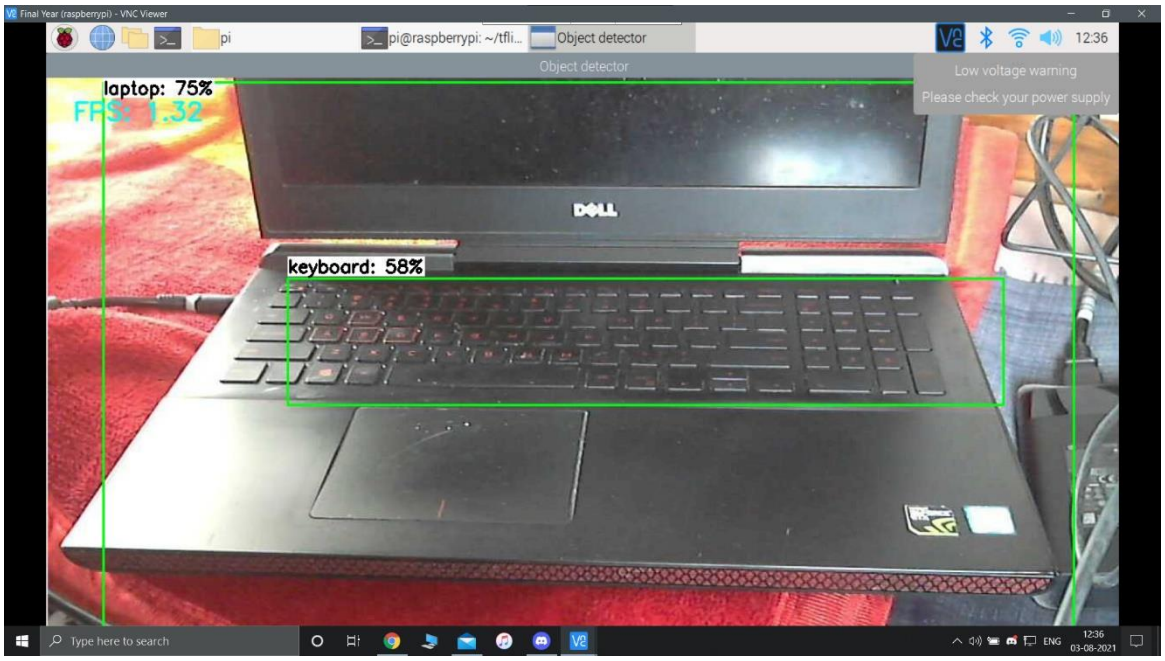
```
        # Retrieve detection results

        boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects

        classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects

        scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects

        #num = interpreter.get_tensor(output_details[3]['index'])[0] # Total number of detected objects (inaccurate and not needed)


        # Loop over all detections and draw detection box if confidence is above minimum threshold

        for i in range(len(scores)):

            if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):


                # Get bounding box coordinates and draw box

                # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()

                ymin = int(max(1,(boxes[i][0] * imH)))

                xmin = int(max(1,(boxes[i][1] * imW)))

                ymax = int(min(imH,(boxes[i][2] * imH)))

                xmax = int(min(imW,(boxes[i][3] * imW)))


                cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)


                # Draw label
```

```
            object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index

            label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'

            labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size

            label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window

            cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in

            cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

            #return label

            print(label)

    # Draw framerate in corner of frame

    cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,0),2,cv2.LINE_AA)


    # All the results have been drawn on the frame, so it's time to display it.

    #cv2.imshow('Object detector', frame)


    # Calculate framerate

    t2 = cv2.getTickCount()

    time1 = (t2-t1)/freq

    frame_rate_calc= 1/time1
```

```
        # Press 'q' to quit
        if cv2.waitKey(1) == ord('q'):
            break


    # Clean up
    cv2.destroyAllWindows()
    videostream.stop()
```

OUTPUT FROM THE WEBCAM:





CHAPTER 5

CONCLUSION

The report has discussed the development of a “Voice Recognition and Object Detection Rover”. The objective of this project was to develop the necessary hardware and software to perform Voice Recognition and Object Detection successfully.

This project can be upgraded to assist the visually impaired by providing identification of objects in a 360-degree environment. To reduce the cost of a power source, solar panels of good efficiency can be provided for the rover. Also, a mobile application can be created to integrate the scripts, thus making the process much more seamless.

This object detection model (COCO Dataset) has a moderate confidence rate with medium level accuracy. Also the field of view of the camera detection is linear and parallel to the path of the rover movement.

CHAPTER 6

REFERENCES

[1] Rahmadi Kurnia, Md. Altab Hossain, Akio Nakamura, and Yoshinori Kuno, "Object Recognition through Human-Robot Interaction by Speech", International Workshop on Robot and Human Interactive Communication, pp. 619-624, 2004.

[2] Sajkowski, M., "Voice control of dual-drive mobile robots-survey of algorithms", Robot Motion and Control, 2002. RoMoCo ’02., pp. 387-392, 2002.

[3] Peter X. Liu, A. D. C. Chan, R. Chen, K. Wang, Y. Zhu, "Voice Based Robot Control", International Conference on Information Acquisition, pp.543-547, 2005.

[4] Bojan Kulji, Simon Janos and Szakall Tibor, "Mobile robot controlled by voice", International Symposium on Intelligent Systems and Informatics, pp. 189-192, 2007.

[5] A review and an approach for object detection in images January 2017 International Journal of Computational Vision and Robotics 7(1/2):196

[6] Microsoft COCO: Common Objects in Context: Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár.

[7] Kannan, K. & Selvakumar, J. (2015). Arduino Based Voice Controlled Robot”, International Research Journal of Engineering and Technology (IRJET), Vol. 02, p-ISSN: 2395-0072, e-ISSN: 2395-0056.