**WEARABLE PUBLIC SOCIAL DISTANCE MONITORING DEVICE FOR COVID19**

A PROJECT REPORT

***Submitted by***

**DEJA SHERLIN. A (310617205014)**

**FAMIDHA PARVEEN. J (310617205016)**

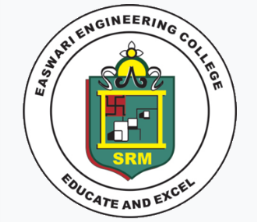
**KEERTHANA. J. P (310617205028)**

***in partial fulfillment for the award of the degree of***

# BACHELOR OF TECHNOLOGY

***in***

INFORMATION TECHNOLOGY



EASWARI ENGINEERING COLLEGE, CHENNAI

(Autonomous Institution)

*affiliated to*

ANNA UNIVERSITY:CHENNAI - 600025

APRIL 2021

EASWARI ENGINEERING COLLEGE, CHENNAI

(AUTONOMOUS INSTITUTION)

AFFILIATED TO ANNA UNIVERSITY, CHENNAI 600025

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“WEARABLE PUBLIC SOCIAL DISTANCE MONITORING DEVICE FOR COVID19”** is the bonafide work of “**DEJA SHERLIN A. (310617205014), FAMIDHA PARVEEN J. (310617205016)** and **KEERTHANA J. P. (310617205028)”** who carried out the project work under my supervision.

|  |  |  |
| --- | --- | --- |
| SIGNATURE |  | SIGNATURE |
| Dr. N. ANANTHI, M.E, Ph.D., |  | Dr. S. Kanagasuba Raja, M.E, Ph.D., |
| HEAD OF THE DEPARTMENT |  | SUPERVISOR |
| Dept. of Information Technology,  Easwari Engineering College,  Bharathisalai, Ramapuram,,  Chennai-600 089 |  | Assistant Professor (SI.Gr),  Dept. of Information Technology,  Easwari Engineering College  Bharathisalai, Ramapuram,  Chennai-600 089 |

# VIVA VOCE EXAMINATION

The viva voce examination of the project work, submitted by **DEJA SHERLIN A., Register number: 310617205014, FAMIDHA PARVEEN J., Register Number: 310617205016 and KEERTHANA J. P., Register Number: 310617205028”** is held on ……………………………....

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our sincere thanks and gratitude to our Founder Chairman **Dr.T.R.PARIVENDHAR** and our beloved Chairman **Dr.R.SHIVAKUMAR**, for their support and inspiration. We would like to convey our due respect and regards to our Principal **Dr.R.S KUMAR** and Head of the Department **Dr.N.ANANTHI**, for their constant encouragement and guidance.

With a deep sense of gratitude we would like to thank **Dr.A.K.MARIAPPAN** for their motivation, timely help and valuable suggestions.

We owe our profound gratitude to our project supervisor **Dr.S.KANAGASUBA RAJA**, Associate Professor, who took keen interest in our project work and guided us all along in bringing out this project in complete shape by providing all the necessary information for developing a good system.

We would like to sincerely thank our project coordinators **DR.R.PRIYADHARSHINI**, Associate Professor for their valuable assistance and support.

Finally we would like to extend our token of appreciation to all the teaching and non-teaching staff of Department of Information Technology for their kind co-operation throughout the project.

**ABSTRACT**

Coronavirus diseases 2019 (COVID-19) is a widespread word around the world. It is a deadly disease that spreads along all over the world. It is a contagious disease that spreads from one person to another person due to droplets of the affected patients that transmits from one to other. So the government announced a several rules. They are 1. Peoples need to wear the masks, 2. Peoples need to maintain social distancing, 3. Peoples need to sanitize their hands at regular intervals. Social distancing is also called as physical distancing. It is one of the important rules that peoples need to follow in day to day life. By maintaining the social distancing, people can avoid each other from the spreading of the virus. Due to COVID-19, the government has issued many protocols during the pandemic, Social distancing is one of the important protocols that needs to be followed by the people. Lack of maintaining the social distancing can lead to a widespread of coronavirus among people. People who can easily gathered in public area which may globally increase the spread of COVID-19 that leads to a people illness. In this paper, the proposed system will develop an electronics device which will provide safe distancing between the peoples. This device is handy and easy to carry. This Wearable device can detect the person and intimate to the user and warn, when the distance between individuals gets reduced than safe distance. It consists of the micro-controller along with camera to detect the presence of human. The person identification will be done using CNN algorithm. It also used an ultrasonic sensor to detect the distance between the person from the user. On mounted buzzer will intimate to the user when distance reduced than safe range.

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGENO.**

ABSTRACT LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

1. **INTRODUCTION** 
   1. ABSTRACT
   2. PROBLEMS
   3. OBJECTIVE
   4. ORGANIZATION OF THE PROJECT REPORT
   5. SUMMARY
2. **LITERATURE SURVEY**

2.1 INTRODUCTION

2.2 RELATED WORKS

2.3 LIMITATIONS OF EXISTING SYSTEM

2.4 SUMMARY

1. **PROPOSED SYSTEM DESIGN**

3.1 INTRODUCTION

3.2 PROPOSED SYSTEM

3.3 SYSTEM ARCHITECTURE

3.4 SUMMARY

1. **SYSTEM IMPLEMENTATION**

4.1 INTRODUCTION

4.2 HARDWARE AND SOFTWARE

SPECIFICATIONS

4.2.1 HARDWARE REQUIREMENTS

4.2.1.1 RASPBERRY PI CAMERA

4.2.1.2 RASPBERRY PI

MICROCONTROLLER

4.2.1.3 ULTRASONIC SENSOR

4.2.1.4 BUZZER

4.2.1.5 PUSH BUTTON

4.2.2 SOFTWARE REQUIREMENTS

4.3 TECHNOLOGIES USED

4.3.1 PYTHON

4.3.2 OPENCV PYTHON

4.3.3 CNN

4.4 MODULE IMPLEMENTATION

4.4.1 MODULES

4.4.1.1 IMAGE CAPTURING

4.4.1.2 FEATURE SELECTION

AND CLASSIFICATION

4.4.1.3 SENSOR INTERFACE

4.5 SNAPSHOTS

4.6 SUMMARY

1. **PERFORMANCE ANALYSIS**
   1. INTRODUCTION
   2. EXPERIMENTAL RESULTS
   3. SUMMARY
2. **CONCLUSION AND FUTURE WORK**

6.1 CONCLUSION

6.2 FUTURE WORK

**APPENDICES**

**REFERENCES**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
|  |  |  |

* 1. PERFORMANCE EVALUATION

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |

* + 1. BLOCK DIAGRAM
    2. FLOW DIAGRAM
    3. SYSTEM ARCHITECTURE
    4. RASPBERRY PI CAMERA
    5. RASPBERRY PI MICROCONTROLLER
    6. ULTRASONIC SENSOR
    7. BUZZER
    8. PUSH BUTTON
    9. INTERMEDIATE RESULT OF MODULE 1
    10. HOW COMPUTER SEES AN IMAGE
    11. APPLYING OF FILTERS
    12. REPRESENTATION OF CURVE IN MATRIX
    13. MULTIPLICATION AND SUMMATION
    14. VIRTUALIZATION OF FILTERS
    15. DIAGRAM OF CNN PROCESS
    16. ACCURACY CHART
    17. SCREENSHOT OF PERSON IDENTIFICATION
    18. ACCURACY GRAPH
    19. INTERIOR PROCESS OF DISTANCE

CALCULATION

* + 1. PIN DIAGRAM
    2. PROTOTYPE OF THE DEVICE

**CHAPTER 1**

**INTRODUCTION**

* 1. **ABSTRACT**

Coronavirus diseases 2019 (COVID-19) is a widespread word around the world. It is a deadly disease that spreads along all over the world. It is a contagious disease that spreads from one person to another person due to droplets of the affected patients that transmits from one to other. So the government announced a several rules. They are 1. Peoples need to wear the masks, 2. Peoples need to maintain social distancing, 3. Peoples need to sanitize their hands at regular intervals. Social distancing is also called as physical distancing. It is one of the important rules that peoples need to follow in day to day life. By maintaining the social distancing, people can avoid each other from the spreading of the virus. Due to COVID-19, the government has issued many protocols during the pandemic, Social distancing is one of the important protocols that needs to be followed by the people. Lack of maintaining the social distancing can lead to a widespread of coronavirus among people. People who can easily gathered in public area which may globally increase the spread of COVID-19 that leads to a people illness. In this paper, the proposed system will develop an electronics device which will provide safe distancing between the peoples. This device is handy and easy to carry. This Wearable device can detect the person and intimate to the user and warn, when the distance between individuals gets reduced than safe distance. It consists of the micro-controller along with camera to detect the presence of human. The person identification will be done using CNN algorithm. It also used an ultrasonic sensor to detect the distance between the person from the user. On mounted buzzer will intimate to the user when distance reduced than safe range.

* 1. **PROBLEMS**

Another way to decrease the spread of contagious diseases, for example, Covid-19, is to avoid socializing. This is not a new idea, as many communities have come to realize the importance of isolating themselves from who are affected by infection for a long time.

The aim is to decrease the spread, delay the pandemic, reduce its size, and spread long-term patients to reduce the pressure on the health care system.

**What Does Social Distancing Mean?**

It is done to avoid contact with others. It has been recommended that maintaining a distance of two meters from one effect is significantly less in the spreading of multiple flu viruses, including COVID-19.

In exercise, this means that avoiding the presence of other people will help reduce the spread of dangerous diseases. Social isolation is one of the non-pharmaceutical ways to control infection that can stop or reduce the spread of infectious disease.

**How does this work?**

 The virus that causes COVID-19 is now very fast spreading from one to another. When a healthy individual comes in contact with airborne droplets from coughing or sneezing from an affected person, they can catch the infection.

The World Health Organization (WHO) states that "COVID-19 is transmitted by droplets and formats during unprotected contact between the infector and the infectee". A fomite is something that can carry infection, such as clothing, etc. Therefore, the spreading of the disease can be reduced by staying far from other people and avoiding contact with infected fomites.

Social segregation aims to reduce the spreading of COVID-19 to humans by reducing communication between potentially affected people and healthy ones, or between groups of people with high levels of spreading and groups of people with or without transmission levels.

**How Effective Is It?**

Research into breakouts of contagious diseases, such as the flu, has shown that social exclusion is a potent method to decrease the number of affected people as long as the procedures are complete and that they continue in a timely manner. A study of the 1918 flu epidemic, comparing different countries in the USA, showed the benefits of using social distance.

A recent, systematic review by Fong et al., (2020) "found further evidence from observational and experimental studies to support the effectiveness of social mobilization measures during the flu season. Of these interventions."

* 1. **OBJECTIVE**

The aim of this project is to increase a safety measures during Pandemic Situation by developing a Raspberry Pi wearable device for person identification to maintain social distancing using CNN during pandemic situation.

* 1. **ORGANIZATION OF THE PROJECT REPORT**

The project work is in various phases which is categorized into chapters. The 2nd chapter tells about the literature survey which we made and the limitations of the existing system.The 3rd chapter tells about the proposed system, block diagram, flow diagram and the system architecture of the proposed system. The 4th chapter tells about the system implementation, hardware and software components, technologies used, modules and its description. The 5th chapter tells about the performance analysis and experimental results. The 6th chapter tells about the conclusion and the possible future works.

* 1. **SUMMARY**

The chapter 1 tells about the problems that is facing by the people due to COVID-19 and how much important is the social distancing during pandemic. We are going to help using our proposed system. Here we also discussed about objective of our device.

**CHAPTER 2**

**LITERATURE SURVEY**

* 1. **INTRODUCTION**

A literature survey is about the searching and the collection of the research works, documents, articles, etc., which will be related to the work which we are going to make for our project work.

Due to COVID-19, the government has issued many protocols during the pandemic, Social distancing is one of the important protocols that needs to be followed by the people. Lack of maintaining the social distancing can lead to a widespread of coronavirus among people. Imagine that by not maintaining the social distance people can easily gathered in public area which may globally increase the spread of COVID-19 that leads to a people illness. It is safe to maintain the physical distance in the public places.

**2.2 RELATED WORKS**

In this paper [1] the person will be detected in the picture using open computer vision. The messages are send to the security operator and server. In this paper [2] the magnetic field is used with the help of sensor for tracking the social distance between people. In this paper [3] the alarming system of S2D warns the human when they cross the threshold minimum safe distance.

In this paper [4], it provide an wearable device for those who are in quarantine which is used for monitoring them, both for checking the status of the health and personnel managing, and provide the facility for getting admission in the hospital; monitoring the patients and detect the diseases by sensing the symptoms of the patients by diagnosing and monitoring continuously with telehealth technologies. In the paper [5], the mobility patterns of the individual and aggregate humans are found for digital contact tracing for social distancing. This paper [6] also used to identify the distance between people by using the Bluetooth technology.

In this paper [7], the image consists of values of pixels with the central point which is used for calculating the distance between the peoples. In the paper [8], it tells about that whether the covid19 is spread through the droplets. The paper [9], tells about that it is used for the pulse checking with the use of the pulse oximeter. In this paper [10], it proposes the Retina Face Mask, which gives high rate of efficient and accuracy with the help of the face mask detector.

* 1. **LIMITATIONS OF EXISTING SYSTEM**
* The existing system was done in the indoor environment.
* It can be used in the only in the places like factories, buildings, shops ,etc.,
* It was not for the individuals.
* This was done with Haar cascade algorithm which has less accuracy.

**2.4 SUMMARY**

This chapter tells about the literature survey which we made and the limitations of the existing system. A framework has been proposed to overcome the limitations of the existing system.

**CHAPTER 3**

**PROPOSED SYSTEM DESIGN**

**3.1 INTRODUCTION**

System design tells about the architecture, components, modules, interfaces, and data for designing the work. It will give a clear pictorial representation for our proposed system. It also helps for the clear understanding the process flow of the proposed system.

**3.2 PROPOSED SYSTEM**

This device consists of Button, Ultrasonic sensor, Buzzer and raspberry pi camera connected with Raspberry Pi. It consists of micro-controller along with camera to detect the presence of human.

Ultrasonic sensor

Raspberry Pi 3

Raspberry pi camera

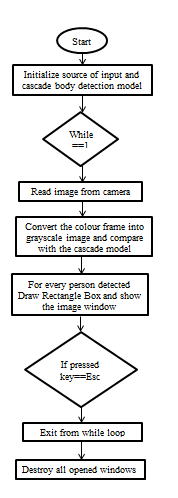
Buzzer

Push button

Power supply Unit

**Figure 1: Block Diagram**

When person identified in the input frame from camera, it detects the person using SSD model and it will alert the user. This model trained by CNN (Convolutional Neural Network) algorithm. The ultrasonic sensor is used for the detecting the distance between the people from the user, the controller triggers the ultrasonic sensor to measure the distance. When the distance will reduce than safe range, then the buzzer will produce sound.



**Figure 2: Flow diagram**

This Wearable device can detect the person and intimate to the user and warn when the distance between individuals gets reduced than safe distance which is between 1 to 1.5 meters.

**3.3 SYSTEM ARCHITECTURE**

CNN

Image Classification

Raspberry pi Camera

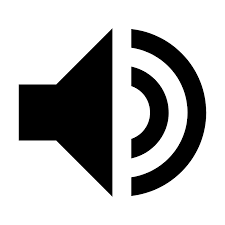
Ultrasonic Sensor

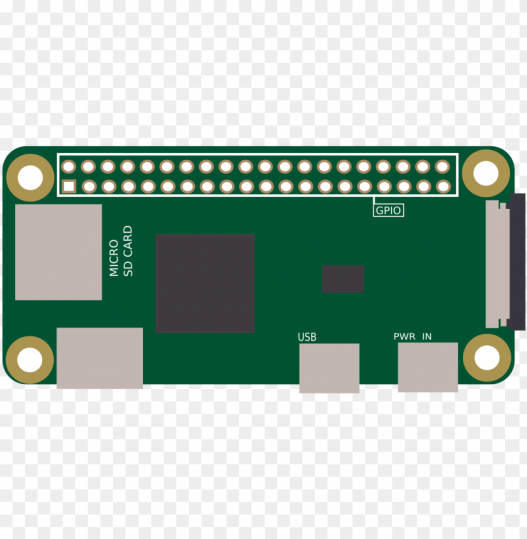
Raspberry Pi Three

Buzzer

Capture images and send number of array







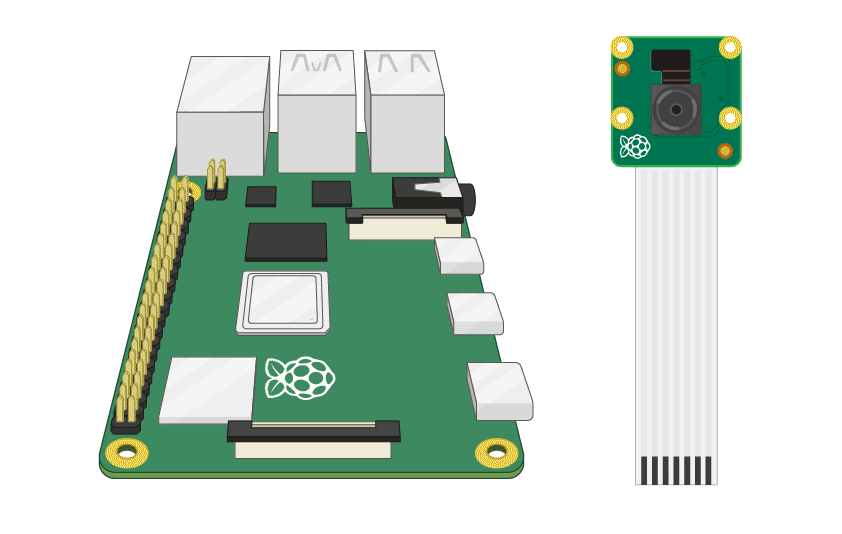


Image Preprocessing

Feature Selection

Send trigger signal and receive echo signal

Send signal to buzzer when person detected

**FIGURE 3: ARCHITECTURE DIAGRAM**

**3.4 SUMMARY**

This chapter tells about the proposed system, block diagram and the flowchart. The system architecture is clearly explained in the diagram itself.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 INTRODUCTION**

The system implementation tells about the detail description about the hardware and software components. The hardware components like raspberry pi camera, raspberry pi microcontroller, ultrasonic sensor, buzzer, push button. The software components like python, CNN. It also tells about the detailed description of the every modules like image capturing, feature selection and classification and the sensor interface. It is the first step in the requirement analysis process. This is an concept of the system implementation.

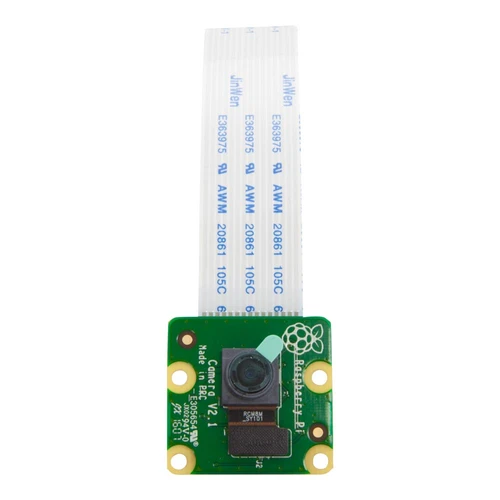
**4.2 HARDWARE AND SOFTWARE SPECIFICATIONS**

**4.2.1 HARDWARE REQUIREMENTS**

* Raspberry Pi Camera
* Raspberry Pi Microcontroller
* Ultrasonic Sensor
* Buzzer
* Push Button

**4.2.1.1 RASPBERRY PI CAMERA**

5 megapixel digicam able to taking pics of 3280 x 2464 pixels seize video at 1080p30, 720p60 and 640x480p90 resolutions. All software program is supported in the modern-day version of Raspbian working system.



**FIGURE 4: RASPBERRY PI CAMERA**

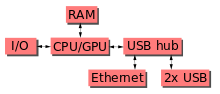
The digital camera v2is the new authentic camera board launched by using the Raspberry Pi foundation.The Raspberry Pi digicam Module v2 is a high great 8megapixel Sony IMX219 imagesensorcustom designed add-onboardfor Raspberry Pi, featuring a set cognizance lens. it's capable of 3280 x 2464 pixel static photographs, and also helps 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi via way of one of the small sockets at the board upper floor and makes use of the committed CSi interface, designed specifically for interfacing to cameras.

The board itself is tiny, at around 25mm x 23mm x 9mm. It additionally weighs simply over 3g, making it best for cell or different programs in which length and weight are crucial. It connects to Raspberry Pi with the aid of manner of a brief ribbon cable. The high first-class Sony IMX219 picture sensor itself has a local decision of 8megapixel, and has a set awareness lens on-board. In terms of still pics, the digicam is capable of 3280 x 2464 pixel static pictures, and additionally helps 1080p30, 720p60 and 640x480p90 video.

**4.2.1.2 RASPBERRY PI MICROCONTROLLER**

Raspberry Pi 3 model B changed into released in February 2016 with a 64 bit quadcore processor, on-board wifi, bluetooth and USB boot abilties. other alternatives are: energy over Ethernet (PoE), USB boot and network boot. This lets in using the Pi in tough-to-attain locations (probable without power). the overall specs for the Raspberry Pi three consist of: CPU: Quad-core 64-bit ARM Cortex A53 clocked at 1.2 GHz. GPU: 400MHz VideoCore IV multimedia. reminiscence: 1GB LPDDR2-900 SDRAM (i.e. 900MHz)





**FIGURE 5: RASPBERRY PI MICROCONTROLLER**

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

**4.2.1.3 ULTRASONIC SENSOR**

Ultrasonic sensor is interfaced with raspberry pi to measure the distance between person and the user. Sensor is triggered only after the person is detected by the system. User will notified by the buzzer sound.

The HC-SR04 Ultrasonic (US) sensor may be a 4-module. Those pins names are Vcc, Trigger, Echo and Ground pin. This sensor could be a very hip sensor employed in many applications. It mainly required for sensing objects and measuring distance. The sensor has two eyes like projects within the front which forms the Ultrasonic transmitter and Receiver. The working of sensor is done with the easy high school formula

Distance = Speed × Time

The Ultrasonic wave is transmitted by ultrasonic transmitter, this wave travels through the air and when it gets opposed to any material it gets reflected back toward the sensor this reflected wave is observed by the ultrasonic receiver module as shown in the image below





**FIGURE 6: ULTRASONIC SENSOR**

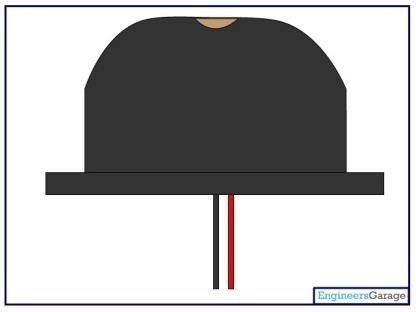
The calculation of the distance will be done with the help of Speed and time. The universal speed of Ultrasonic wave is 330m/s, at room conditions. The trigger pin will get the trigger if the person is identified and the distance will be calculated with the help of the microcontroller or microprocessor which will be return back from the echo pin and the information will be send to the microcontroller or microprocessor. This is the working process of the ultrasonic sensor.

**HC-SR04 Sensor Features**

* Operating voltage: +5V
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15Ma
* Operating Frequency: 40Hz

**4.2.1.4 BUZZER**

The piezo buzzer produces sound primarily based on opposite of the piezoelectric impact. The generation of stress variant or pressure via the utility of electrical capacity throughout a piezoelectric material is the underlying precept. those buzzers may be used alert a user of an event corresponding to a switching action, counter signal or sensor enter. they're extensively utilized in alarm circuits.

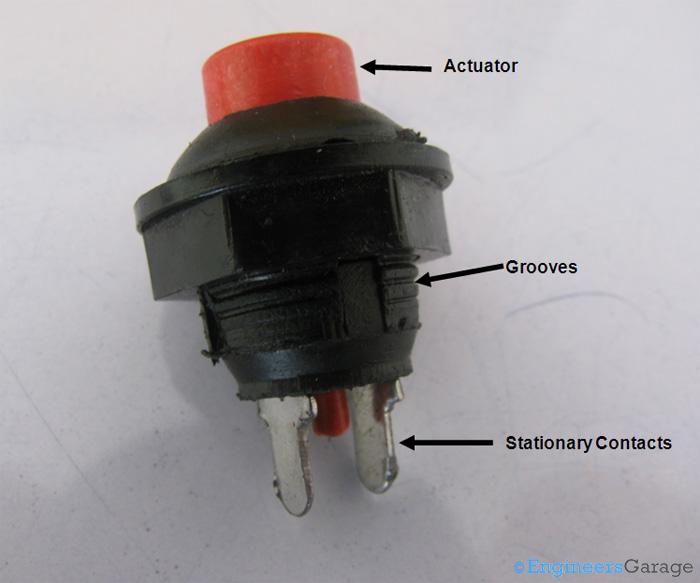


**FIGURE 7: BUZZER**

The buzzer produces a same noisy sound regardless of the voltage variation implemented to it. It consists of piezo crystals among conductors. While a ability is implemented throughout these crystals, they push on one conductor and pull on the alternative. This, push and pull motion, effects in a valid wave. Maximum buzzers produce sound in the range of two to 4 kHz. The crimson lead is attached to the enter and the Black lead is hooked up to ground.

**4.2.1.5 PUSH BUTTON**

 Push button switches are those which can be made to work with the force of a finger or two.



**FIGURE 8: PUSH BUTTON**

The picture above shows the external view of a conventional SPST push button switch. almost all the elements of the switch may be found out by using watching its external shape. The purple coloured bulge is the actuator of the switch. The actuator extends in the direction of the lowest of the transfer and emerges out as a skinny cylinder. Among other prominent extensions are the two stationary metal contacts legs at the lowest. a cool pattern is also supplied for the cause of clean mounting.

Usually, external body of push button switches is crafted from polymer plastics and can have multiple shapes, sizes and output terminals relying upon its use.

**4.2.2 SOFTWARE REQUIREMENTS**

* Language : Embedded ‘C++’,Python
* Compiler : GCC Complier.
* OS : Linux

**4.3 TECHNOLOGIES USED**

**4.3.1 PYTHON**

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* System scripting.

**What can Python do?**

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

**Good to know**

* The maximum recent predominant model of Python is Python 3, which we shall be using on this educational. however, Python 2, although now not being up to date with something apart from safety updates, is still quite famous.
* Python 2.zero became launched in 2000, and the 2.x variations were the general releases till December 2008. At that time, the development team made the selection to launch model 3.zero, which contained some noticeably small however enormous adjustments that were no longer backward compatible with the two.x versions. Python 2 and three are very similar, and a few capabilities of Python three have been backported to Python 2. but in general, they remain not quite compatible.
* Each Python 2 and three have persisted to be maintained and developed, with periodic launch updates for each. As of this writing, the most latest versions to be had are 2.7.15 and three.6.five. but, an legit end Of lifestyles date of January 1, 2020 has been established for Python 2, after which period it'll no longer be maintained.
* Python remains maintained by using a middle improvement crew at the Institute, and Guido is still in fee, having been given the identify of BDFL (Benevolent Dictator For existence) by way of the Python network. The name Python, by means of the way, derives now not from the snake, but from the British comedy troupe Monty Python’s Flying Circus, of which Guido changed into, and presumably still is, a fan. it's miles commonplace to discover references to Monty Python sketches and movies scattered for the duration of the Python documentation.
* It's miles feasible to write down Python in an included improvement environment, along with Thonny, Pycharm, Netbeans or Eclipse which might be especially beneficial while managing large collections of Python documents.

**Python Syntax compared to other programming languages**

* Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

### Python is Interpreted

* Many languages are compiled, meaning the source code you create needs to be translated
* Into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
* This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
* One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
* In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
* For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and [object-oriented programming](https://realpython.com/python3-object-oriented-programming/).
* Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
* Python accomplishes what many programming languages don’t: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

**4.3.2 OPENCV PYTHON**

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. ... OpenCV-Python uses Numpy, which is a fairly optimized library for numerical operations with a MATLAB-fashion syntax. all the OpenCV array systems are converted to and from Numpy arrays.as compared to languages like C/C++, Python is slower. That said, Python may be effortlessly extended with C/C++, which allows us to jot down computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two blessings: first, the code is as rapid because the authentic C/C++ code (due to the fact it is the real C++ code running in heritage) and 2nd, it less complicated to code in Python than C/C++. OpenCV-Python is a Python wrapper for the authentic OpenCV C++ implementation.

**4.3.3 CNN**

Convolutional Neural Network is an technology used for classification. Image annotation is a vital step of which the objective is to label the positions and classes of object spots in the images. A convolution neural networks (CNN) algorithm that provides a frame selection function is developed in Python. The bounding box will be detected and labeled in frame where the person in the image frame. This category of the image is the job of taking pictures as an input image and the magnificence might be outputting (a cat, a dog, a human and so on) or the chance of lessons that could be a better description of the picture. Actually we are going to classify the presence of persons in the images using CNN algorithm. For that we need lot of images based on persons and without persons which is called as dataset. We are going to use these dataset for training and validating using CNN. It is used for classifying the person or dog or cat using feature selection and image classification method.

**4.4 MODULE IMPLEMENTATION**

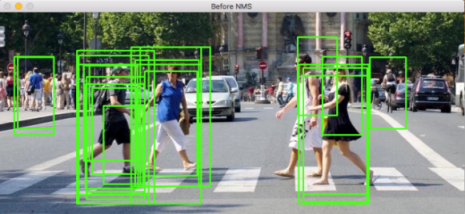
A module implementation consists in a sequence of implementation phrases.

**4.4.1 MODULES**

* Image Capturing
* Feature Selection and Classification
* Sensor interface

**4.4.1.1 IMAGE CAPTURING**

In this module we create a basic infrastructure to establish the proposed feature of image handling in python. The python environment is prepared for accessing the video from the input device with help of pi camera. The images which is read from the camera is get as a colour frame which is converted into grayscale image and compare with the cascade model for the detection of every person. If the person is detected, it will draw rectangle box and show the image window. The images are stored. In this module, these images are collected as a dataset. “Figure 9” represents the intermediate result of this module.



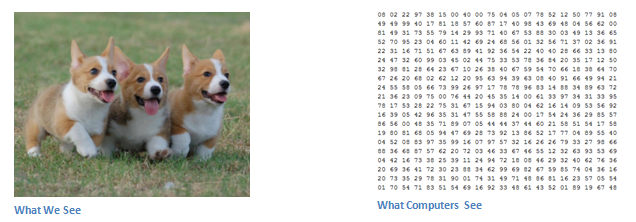
**FIGURE 9:** **INTERMEDIATE RESULT OF MODULE 1**

## 4.4.1.2 FEATURE SELECTION AND CLASSIFICATION

Image annotation is an important step of which the target is to label the classes and position of object spots within the images. For this stage, a convolution neural networks (CNN) algorithm that gives a frame selection function is developed in Python. When the detection is done by the bounding box and it will be labeled in frame where the person in the image frame.

**4.4.1.2.1 Images in array format**

In this module, actually we are going to classify the presence of persons in the images using CNN algorithm. For that we need lot of images based on persons and without persons which is called as dataset. We are going to use these dataset for training and validating using CNN.



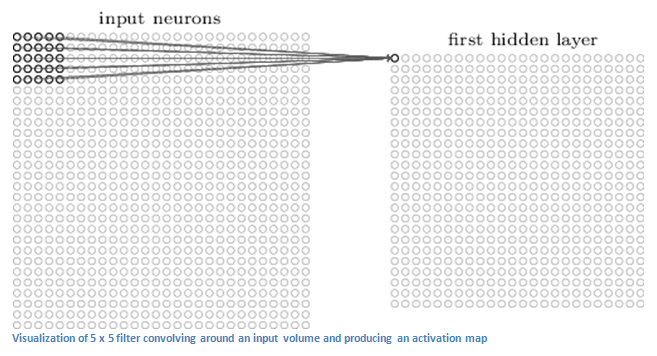
**FIGURE 10: HOW COMPUTER SEES AN IMAGE**

The above image consists of dog which is taken as an example. Whenever PC saw a picture it consists of pixel values in an array format which is represent in the “figure 10 and 12”. Consider as an example, the photo size will appears as 32×32×32 array of numbers. Let’s got a coloration image in JPG shape and its length is 480×480. The representative array may be 480×480×3. Each of those numbers will be in the range from 0 to 255. These numbers are used to perform picture classification. The computer provides a result as the array of numbers and the output will describes as numbers. Actually the image is in the form of RGB color. It will be converted into a gray scale image. And then the resize (it will adjust the pixel size) and rescaling (in rescaling each and every value will be divided by 255) will be done.

The pc need to identify the dog or cat which is given in the images by the classification of the features. The special features of cat or dog like four legs or paws. Likewise the features of the persons like eyes, nose, ears, etc., The pc is ready carry out photo class via searching out features which is in low level together with curves and edges, after which building up to extra concepts through convolutional layers which is in sequence. The human brain consists of neurons, which is used for conveying the message. Likewise the same concept is used in CNN with the help of the neural network for image classification. In CNN, the dataset which consist of lots of images is given as an input data. The dataset consist of more than 1500 images. These inputs will be passes through a several steps like image preprocessing, feature selection and image classification.

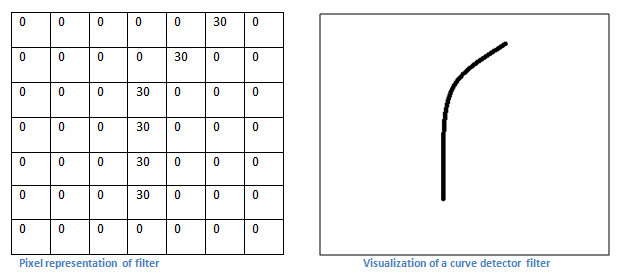
**4.4.1.2.2 Feature Selection**

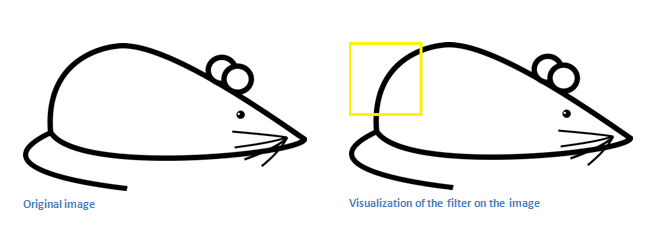
The feature will be extracted in this step. If the input image has taken with an array values of 32 x 32 x 3, now it will be going to implement using CNN. Now the filter (also referred as kernel) is applied upon the array of original images. The filter size will be taken as 5x5x3 array of numbers as an example. Suppose if the filter is applied on the top left corner on the original images, the values in the filter will be multiplied with the values in the original images. The “figure 11” will represent this process. Then it will be summed up to get a value, it will be saved in new array of values.



**FIGURE 11: APPLYING OF FILTERS**

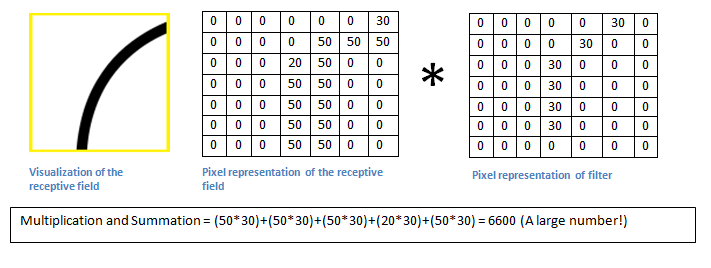
Now the filter will be striding from one place to another in the original image. The same multiplication and summing will be done and it will be saved. Likewise the same procedure will be done throughout the full value of the original image and it will be saved in 28x28 array values.

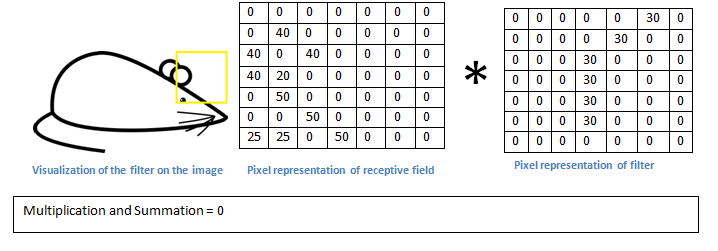




**FIGURE 12: REPRESENTATION OF CURVE IN MATRIX**

Every precise area at the input quantity produces a number. The purpose you get a 28 x 28 array is that there are 784 specific locations that a 5 x 5 clear out can in shape on a 32 x 32 input photo. The 28 x 28 array is mapped with the 784 numbers.

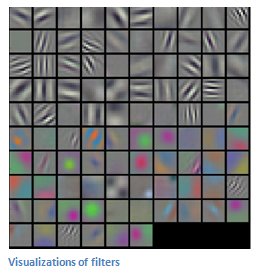




**FIGURE 13: MULTIPLICATION AND SUMMATION**

Suppose we now use two filters instead of one which is 5 x 5 x 3 filters. Then the 28 x 28 x 2 is an output quantity. The spatial dimensions will be higher and capable to hold, when the usage of filters is high. Mathematically, this is often what’s happening in the convolution layer. The 7 x 7 x 3 filter is an 1st filters and it will detect the curve. For the detection of the curve, that the filters out can has a structure of the pixels in that the values of the numerical will be higher along the side that the region, curve will be formed.

The “figure 13” represents that how the multiplication and summation will be done. 6600 is the value of array, the 26 x 26 x 1 is an activation map which is in the top left (26 due to the 7x7 filter in preference to 5x5). The activation maps is zero because there was not in the input extent that triggered and clear out to prompt. It is an only one filter out. This is a filter which is used for locating the curves externally and the curves towards left side which will be produced by us. The maxpooling method will be used which results in a small size array when compared to the original image. The ReLU activation function is used in the CNN algorithm. When there is high numbers of filters then this activation maps depth will increase.

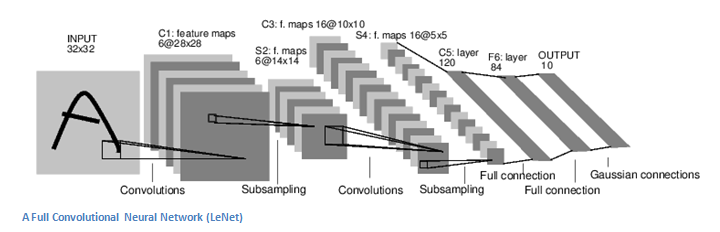


**FIGURE 14: VIRTUALIZATION OF FILTERS**

In the first layer which is convolution we can see the visualization of the filters which is shown in “figure 14”. The extracted feature will be saved as a database file.

**4.4.1.2.3 Image classification**

 If the image consists of person, the prediction is done with the help of the program. The characteristics of persons like eyes or nose, face, etc., The images consists of top values within the activation maps which constitute the characteristics. The “figure 15” represents the diagram of CNN. In the fully connected layers, the neural network is fully used. The neural network is used for classifying whether the image consists of person or not. The neural network consists of three layers. They are input layer, hidden layers and output layers. With the help of the input layers, the input dataset images will be given through it. The neural nodes which consist of inputs are connected with other neurons to give a pattern through the weights and the bias. The input value and the weight will be multiplied and added with the bias to get the certain pattern. The weight and the bias are called as learnable parameters.

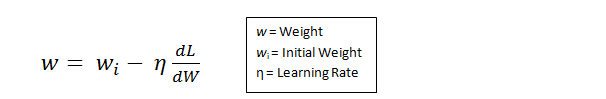


**FIGURE 15: DIAGRAM OF CNN PROCESS**

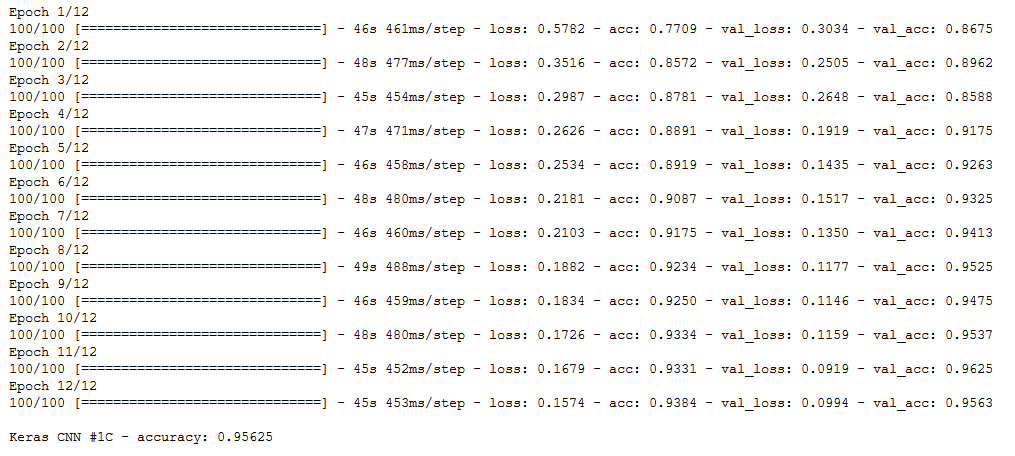
Now this pattern will pass through the hidden layers. It consists of number of neurons layers. The input patterns will pass through the activation function also called as ReLU, it will give the variety of data in a certain range. With the help of this activation function the error value can be identified. The curve will be gets through this. It is used to identify the non-linearity pattern which will be found through the curve. With the help of this certain pattern which we get, it will be check with the trained dataset. The trained dataset is a data which we get from the feature selection database file. It will be compared with the pattern. If the features will match, then the output layer conveys it as the person is present in the image. If there is a loss then it will be rectified using back propagation.

*Etotal=*  (1)

A loss perform is also delineated in many specific strategies but a standard one is MSE (imply square blunders), that's ½ instances (real - predicted) square.

 (2)

In the back propagation method, all the weights and bias will be updated based on the variables and then it will pass till the input layers in backward. The above equation consists of a **dL/dW**, W are the weights. The loss will be rectified by adjusting the values. After that, the new input data images will be passed through the layers with the new weights and filters for better classification.

 **FIGURE 16: ACCURACY CHART**

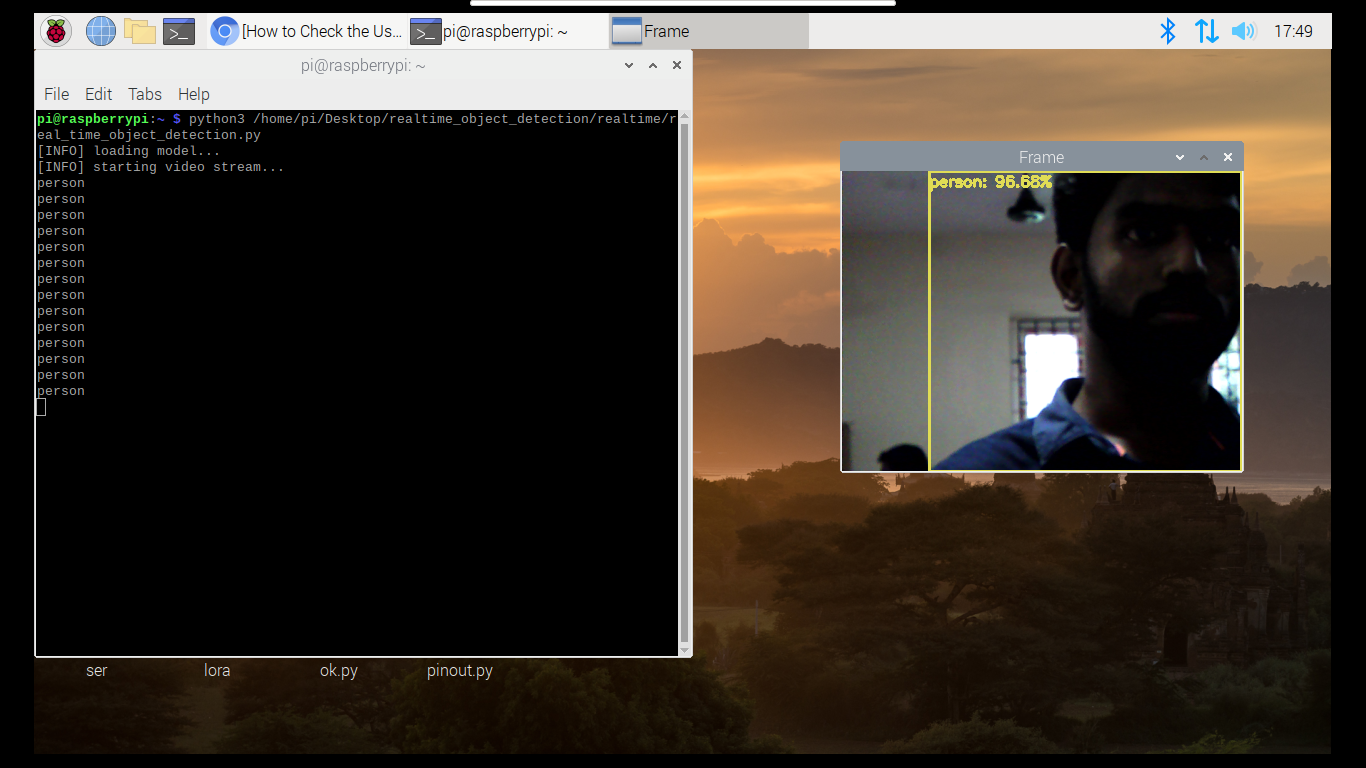
The “figure 16” represents the graph of accuracy and epoch. The epoch is used for attaining the accuracy by continuously passing the error values for better accuracy. From the above chart, we can say that we are attaining the accuracy of 95%.

**4.4.1.2.4 Testing**

Subsequently, to visualize whether or not the CNN works, we've a particular set of images and labels and pass the pictures through the CNN. We tend to judge the outputs to the ground truth and spot if our network.

## 4.4.1.3 SENSOR INTERFACE

In the sensor interface module, the raspberry pi, raspberry pi camera, buzzer and ultrasonic sensor is connected. When the raspberry pi camera is on, it will live capture the images. The camera captures the images per second. The camera is connected with the raspberry pi. The raspberry pi consists of an operating systems with the help of memory card where the files will be saved. The CNN algorithm which is saved in it will run as a loop for infinity times. The camera captures the images per second and the image classification will be done. The following “figure 17 and 19” shows the interior process person identification and image classification.



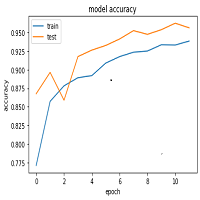
**FIGURE 17: SCREENSHOT OF PERSON IDENTIFICATION**

When the person is detected then it will trigger the ultrasonic sensor. Ultrasonic sensor is interfaced with raspberry pi to measure the distance between person and the user. The ultrasonic wave will be transmitted by an ultrasonic transmitter, the wave will travel through the air and when it gets opposed by any material it gets reflected back towards the sensor, this reflected wave is observed by the ultrasonic receiver module.

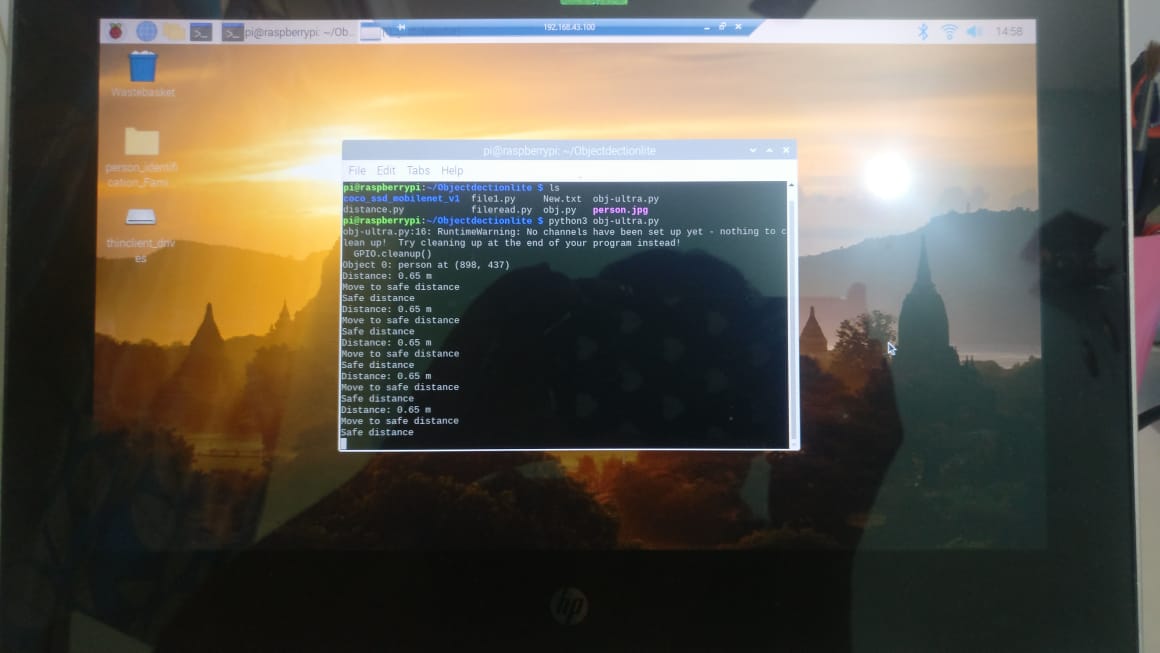
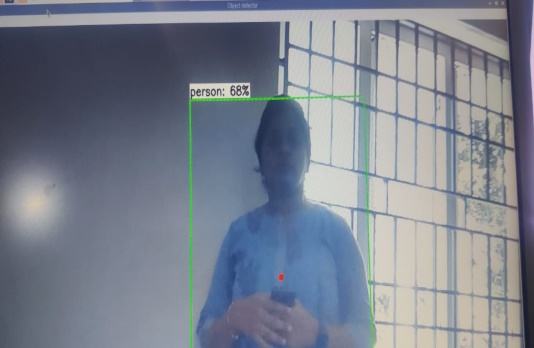
Distance = Speed × Time (3)

With the help of the above formula the distance can be calculated using a microcontroller or microprocessor. If the value which we get through is below the threshold then it will trigger the buzzer sound for alerting the user who is not in a social distancing. The “figure 19” shows that the interior process of the distance calculation for safe distance. The “figure 20” tells about the pin diagram of the device. User will notified by the sound of the buzzer.

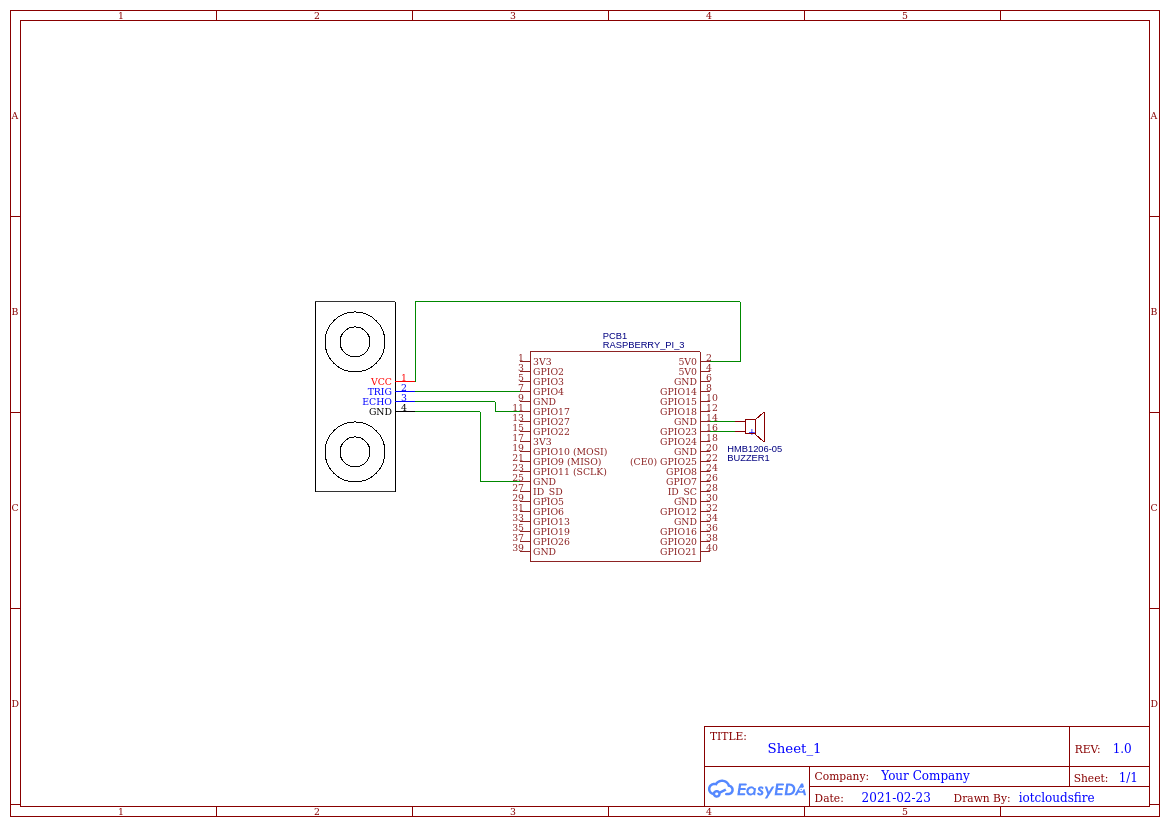
**4.5 SNAPSHOTS**



**FIGURE 18: ACCURACY GRAPH**

**FIGURE 19: INTERIOR PROCESS OF DISTANCE CALCULATION**



**FIGURE 20: PIN DIAGRAM**

**4.6 SUMMARY**

This chapter tells about the detail implementation of the proposed system. It tells about the hardware and software components, technologies used, detailed description of the modules and the snapshots.

**CHAPTER 5**

**PERFORMANCE ANALYSIS**

* 1. **INTRODUCTION**

Performance analysis tells about the efficiency of the working of the software program. It also tells about the comparison part of the algorithms and its efficiency which can also be specified with the help of the tabulation. It also tells about the evaluation and the functioning of the software. The objective of performance analysis is to ensure the software program is working at optimum efficiency. It also used for correction of an issues or errors with is occurring during the testing part.

* 1. **EXPERIMENTAL RESULTS**

The experimental result will tells about how the person will be detected by the use of this device. The device is used to inform the user if he/she is not in an social distancing which will be intimated by using the buzzer.



**FIGURE 21: PROTOTYPE OF THE DEVICE**

The prototype of the device is shown in the “figure 21”. The ultrasonic sensor and the buzzer are fitted outside and the raspberry pi is fitted inside. The below table represent the performance evaluation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Training Accuracy** | **Training Error Rate** | **Validation Accuracy** | **Validation Error Rate** |
| CNN | 97% | 0.04 | 95% | 0.06 |
| Inception V3 | 90% | 0.06 | 92% | 0.09 |
| Haar-cascade | 85% | 0.12 | 83% | 0.17 |

**TABLE I: PERFORMANCE EVALUATION**

The above tabulation tells the comparison between these algorithms. The CNN algorithm is implemented in the proposed system device. The device will inform the person who are between in the distance of 1 to 1.5 meters. It will accurately detect the persons using CNN algorithm with 95%.

* 1. **SUMMARY**

This chapter tells about the performance analysis and the experimental results. The performance is evaluated in the table. Using performance evaluation we can see that the accuracy is increased compared with the others.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 CONCLUSION**

The benefit of the proposed device is used for maintaining the social distancing. It also informs the user if the user is not in a social distancing. During any pandemic, it will be very useful and cost efficient.

**6.2 FUTURE WORK**

In future it may be possible with the proposed system to implement in the electronic items or any device with the face mask detection then it will be very useful. Lockdown is an effective social distancing measure to break the chain of COVID-19 transmission. Social distancing measures are expected to be in place for a sustained period of time in order to limit the spread of COVID-19. Maintaining social distancing is likely to become habit forming, with its effects going well beyond limiting the spread of the disease. The benefit of the proposed device is used for maintaining the social distancing. It also informs the user if the user is not in a social distancing.

**APPENDIX**

**CODE**

**4.4.1.2 FEATURE SELECTION AND CLASSIFICATION CODE:**

import numpy as np

import seaborn as sns

from keras.preprocessing.image import load\_img, img\_to\_array

import matplotlib.pyplot as plt

import os

# size of the image: 48\*48 pixels

pic\_size = 48

# input path for the images

base\_path = "E:/ped/images/"

plt.figure(0, figsize=(12,20))

cpt = 0

for expression in os.listdir(base\_path + "train/"):

for i in range(1,5):

cpt = cpt + 1

plt.subplot(7,4,cpt)

img = load\_img(base\_path + "train/" + expression + "/" +os.listdir(base\_path + "train/" + expression)[i], target\_size=(pic\_size, pic\_size))

plt.imshow(img, cmap="gray")

plt.tight\_layout()

plt.show()

for expression in os.listdir(base\_path + "train"):

print(str(len(os.listdir(base\_path + "train/" + expression))) + " " + expression + " images")

from keras.preprocessing.image import ImageDataGenerator

# number of images to feed into the NN for every batch

batch\_size = 128

datagen\_train = ImageDataGenerator()

datagen\_validation = ImageDataGenerator()

train\_generator = datagen\_train.flow\_from\_directory(base\_path + "train",

target\_size=(pic\_size,pic\_size),

color\_mode="grayscale",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=True)

validation\_generator = datagen\_validation.flow\_from\_directory(base\_path + "validation",

target\_size=(pic\_size,pic\_size),

color\_mode="grayscale",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False)

train\_generator.class\_indices

from keras.layers import Dense, Input, Dropout, GlobalAveragePooling2D, Flatten, Conv2D, BatchNormalization, Activation, MaxPooling2D

from keras.models import Model, Sequential

from keras.optimizers import Adam

# number of possible label values

nb\_classes = 2

# Initialising the CNN

model = Sequential()

# 1 - Convolution

model.add(Conv2D(64,(3,3), padding='same', input\_shape=(48, 48,1)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

# 2nd Convolution layer

model.add(Conv2D(128,(5,5), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

# 3rd Convolution layer

model.add(Conv2D(512,(3,3), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

# 4th Convolution layer

model.add(Conv2D(512,(3,3), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

# Flattening

model.add(Flatten())

# Fully connected layer 1st layer

model.add(Dense(256))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.25))

# Fully connected layer 2nd layer

model.add(Dense(512))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(nb\_classes, activation='softmax'))

opt = Adam(lr=0.0001)

model.compile(optimizer=opt, loss='categorical\_crossentropy', metrics=['accuracy'])

get\_ipython().run\_cell\_magic('time', '', '\n# number of epochs to train the NN\nepochs = 10\n\nfrom keras.callbacks import ModelCheckpoint\n\ncheckpoint = ModelCheckpoint("E:/ped/model\_weights.h5", monitor=\'val\_acc\', verbose=1, save\_best\_only=True, mode=\'max\')\ncallbacks\_list = [checkpoint]\n\nhistory = model.fit\_generator(generator=train\_generator,\n steps\_per\_epoch=train\_generator.n//train\_generator.batch\_size,\n epochs=epochs,\n validation\_data = validation\_generator,\n validation\_steps = validation\_generator.n//validation\_generator.batch\_size,\n callbacks=callbacks\_list\n )')

# serialize model structure to JSON

model\_json = model.to\_json()

with open("E:/ped/model.json", "w") as json\_file:

json\_file.write(model\_json)

# # Analyze the results

# plot the evolution of Loss and Acuracy on the train and validation sets

import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : Adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.legend(loc='upper right')

plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(history.history['acc'], label='Training Accuracy')

plt.plot(history.history['val\_acc'], label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()

**4.4.1.3 SENSOR INTERFACE CODE:**

import os

import argparse

import cv2

import numpy as np

import sys

import time

from threading import Thread

import importlib.util

import RPi.GPIO as GPIO

Buzzer=37

#Ultrasonic sensor setting

GPIO.cleanup()

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

PIN\_TRIGGER = 7

PIN\_ECHO = 11

GPIO.setup(PIN\_TRIGGER, GPIO.OUT)

GPIO.setup(Buzzer, GPIO.OUT)

GPIO.setup(PIN\_ECHO, GPIO.IN)

GPIO.output(Buzzer, GPIO.LOW)

def dist():

for x in range (5):

GPIO.output(PIN\_TRIGGER, GPIO.LOW)

#print ("Waiting for sensor to settle")

time.sleep(2)

#print ("Calculating distance")

GPIO.output(PIN\_TRIGGER, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(PIN\_TRIGGER, GPIO.LOW)

while GPIO.input(PIN\_ECHO)==0:

pulse\_start\_time = time.time()

#print(pulse\_start\_time)

while GPIO.input(PIN\_ECHO)==1:

pulse\_end\_time = time.time()

#print(pulse\_end\_time)

pulse\_duration = pulse\_end\_time - pulse\_start\_time

distance = round(pulse\_duration \* 17150, 0)

distance=int(distance)

distance=distance/100

print ("Distance:",distance,"m")

if(distance<1 and distance>0.5):

print("Move to safe distance")

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(2)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(2)

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(2)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(2)

if(distance<0.5 and distance>0.3):

print("Move to safe distance")

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(1)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(1)

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(1)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(1)

elif(distance<0.3 and distance>0.01):

print("Move to safe distance")

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(0.5)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(0.5)

GPIO.output(Buzzer, GPIO.HIGH)

time.sleep(0.5)

GPIO.output(Buzzer, GPIO.LOW)

time.sleep(0.5)

else:

print("Safe distance")

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

# Source - Adrian Rosebrock, PyImageSearch: https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/

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',

default='/home/pi/Objectdectionlite/coco\_ssd\_mobilenet\_v1')

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='1280x720')

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

default='500x500')'''

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

action='store\_true')

args = parser.parse\_args()

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

'''MODEL\_NAME = 'coco\_ssd\_mobilenet\_v1'

GRAPH\_NAME = 'detect.tflite'

LABELMAP\_NAME = 'labelmap.txt'

min\_conf\_threshold = float(0.5)

imW, imH = int(1280), int(720)

use\_TPU = 'store\_true'''

# 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)

# Create window

cv2.namedWindow('Object detector', cv2.WINDOW\_NORMAL)

#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] >= 0.6)):

# 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

if(object\_name=='person'):

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

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

# Draw circle in center

xcenter = xmin + (int(round((xmax - xmin) / 2)))

ycenter = ymin + (int(round((ymax - ymin) / 2)))

cv2.circle(frame, (xcenter, ycenter), 5, (0,0,255), thickness=-1)

# Print info

print('Object ' + str(i) + ': ' + object\_name + ' at (' + str(xcenter) + ', ' + str(ycenter) + ')')

time.sleep(0.5)

cv2.imwrite('person.jpg',frame)

dist()

# 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()

**REFERENCES**

1. Nenad Petrovic, Dorde Kocic, “IOT based system for COVID-19 Indoor Safety Monitoring”, September 2020.
2. Sizhen Bian, Bo Zhou, Hymalai Bello, Paul Lukowicz, “A Wearable Magnetic Field Based Proximity Sensing System for Monitoring COVID-19 Social Distancing”, September 2020.
3. “Social Distancing Alarming Through Proximity Sensors for COVID-19”, June 2020.
4. Xiaorong Ding, David Clifton, Nan Ji, Nigel H. Lovell, Paolo Bonato, “Wearable Sensing and Telehealth Technology with Potential Applications in the Coronavirus Pandemic”, June 2020.
5. Ajay K. Sethi, Yuhao Kang, Jonathan Patz, “Mobile phone location data reveal the effect and geographic variation of social distancing on the spread of the COVID-19 epidemic”, April 2020.
6. Md. Shirajum Munir, Do Hyeon Kim, Sarder Fakhrul Abedin,“A Risk-sensitive Social Distance Recommenda-tion System via Bluetooth Towards the COVID-19 Private Safety”, March 2020.
7. Afiq Harith Ahamad, Norliza Zaini, Mohd Fuad Abdul Latip, “Person Detection for Social Distancing and Safety Violation Alert based on Segmented ROI”, August 2020.
8. T. Galbadage, B. Peterson, R. Gunasekera, “Does COVID-19 Spread Through Droplets Alone?”, Frontiers in Public Health, vol. 8, April 2020, pp. 1-4, 2020.
9. Y. S. Yan and Y. T. Zhang, "An efficient motion-resistant method for wearable pulse oximeter," IEEE Transactions on Information Technology in Biomedicine, vol. 12, no. 3, pp. 399-405, 2008.
10. Mingjie Jiang\_, Xinqi Fan\*, “RetinaFaceMask: A Face Mask Detector”, 2020.