CSE299

Junior Design Project

Assist Disabled and Old People Using Hand Gesture Recognition

Submitted By

1721144042 - Ridwanul Haque

1721779042 - Md. Saif Ahammod Khan

1721387042 - Nasim Mahmud Mishu

1721966642 - MD Reyad Hossain

Supervisor

Tanjila Farah (Tnf)

Senior Lecturer & Lab Co-ordinator

ELECTRICAL AND COMPUTER ENGINEERING

NORTH SOUTH UNIVERSITY

Spring 2020

Agreement Form

We take great pleasure in submitting our junior design project report on "Assist Disable and Old People Using Hand Gesture Recognition". This course involves multidisciplinary teams of students who build and test custom-designed systems, components, or engineering processes. We would like to request you to accept this report as a fulfillment of a CSE299, titled "Junior Design Project".

Declared By:

……

Name: Nasim Mahmud Mishu

ID: 1721387042

……

Name: Ridwanul Haque

ID: 1721144042

……

Name: Md. Saif Ahammod Khan

ID: 1721779042

..……

Name: MD Reyad Hossain

ID: 1721966642

Approved By:

……

Supervisor

Tanjila Farah

Sr. Lecturer, Department of Electrical and Computer Engineering

North South University, Dhaka, Bangladesh

……

Dr. Rezaul Bari

Chair, Department of Electrical and Computer Engineering

North South University, Dhaka, Bangladesh

Assist Disabled and Old People Using Hand Gesture Recognition

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Abstract

Since an early age, communication through gestures has not only been used by physically challenged persons but is now used for many other applications. Interacting with the physical world using expressive body movements is much easier and more effective than just speaking. As most of the hand is used to perform gestures, Hand Gesture Recognition has been widely accepted for a wide range of applications, such as human-computer interactions, robotics, sign language recognition, etc. With the help of depth information, depth-based methods have improved performance, but depth cameras are not as widely used and affordable as color cameras. That's where our hand gesture recognition system comes in.

Gesture recognition has proved to be a required field in recent years. But using it effectively and implementing it on the right track is the main challenge here. Enhancing low-resolution images has always been a critical focus in the processing of digital images. Images with a resolution as low as [50×50 pixels] shall also be considered for recognition. The gestures under consideration here are the number of fingers raised by the person (one, two, three, four, or five). Low-resolution gesture images captured from web cameras, mobile phones, or low-cost cameras are systematically processed to generate the number of fingers raised. Hand Gesture recognition techniques are primarily divided into vision-based and sensor-based methods. Initially, the hand region is segmented by applying the skin color model to the YCbCr color space. In the next stage, otsu thresholds are applied to separate foreground and background. The template-based matching technique is finally developed using the Principal Component Analysis (PCA) for recognition. In our project, we are going to use such methods for gesture recognition so that even the disabled people can use it at ease.

Since everyone is trying to recognize hand gestures through HCI(Human-Computer Interaction), we are trying to build a system for disabled people by recognizing simple gestures at various conditions. In other projects, everyone is trying just to recognize gestures for fun facts used or for smartphone features or movement in AR, but we intensely want it to be used for disabled people.

Initially, we used Raspberry Pi as a core module and the Pi-cam to detect the hand gestures. We also developed an android application to get a notification and to use a single system for multiple people. This application also retrieves data from the database to notify users of the condition of the disabled or old individuals. Using this project, a single user can look after multiple disabled or aged individuals, which increases its acceptability and usability.

1.2 Background and Motivation:

Motivation

The primary motivation behind this project was to implement a system for helping the elderly and speech and hearing impaired people using modern tools. Many people all over the world suffer from paralysis and other critical health issues that prevent them from leading a healthy life. Paralysis is most often caused by damage in the nervous system, especially the spinal cord. Other significant causes are stroke, trauma with nerve injury, poliomyelitis, cerebral palsy, peripheral neuropathy, Parkinson's disease, ALS, botulism, spina bifida, multiple sclerosis, and Guillain–Barré syndrome. For example, in Parkinson's disease, patients are affected by a progressive nervous system disorder that affects movement. Symptoms start gradually, sometimes starting with a barely noticeable tremor in just one hand. Tremors are common, but the disorder also commonly causes stiffness or slowing of movement. These types of patients require 24-hour observation, which is very difficult for most of the cases. In times the primary caregiver may need to go to the grocery store, prepare food, do laundry or run an errand. We proposed a system with a phone-based app that could be used to call help from far away by using a hand gesture. There are many projects, applications, and products that may help these people in need, but not all of them. Here we proposed a more straightforward and easy to use the system for almost all types of people and in many ways who can use it to take action by a simple hand gesture. We know that nearly every cell phone has some emergency calling system that can alert others if the owner of the phone is in need, but we are thinking of those people who don't have the privileges, ability or scope to use a cell phone in case of an emergency. We often see that people who travel at night sometimes feel insecure in some areas or face dangerous scenarios from time to time. In that case, they got little or no time at all to react, and obviously, they couldn't call for help most of the time. That's why if

we can use this project in public places after increasing the accuracy and the capacity to detect gestures from larger areas, this could be life-saving for them.

We have tried to make a low cost, affordable, and easy to use hand gesture recognition system to assist the disabled and old people who are suffering from such diseases that they are unable to use voice to communicate or unable to move. To compete in the ever-growing competitive world, we have made our primary goal to make these technologies more affordable and easier to use. Any person who lives in rural areas mostly needs this kind of system so that they can easily bear the cost. We will be trying to make it simpler so that anyone can use it intensely. With more updates, the accuracy level will be accrued in a level to recognize hand gestures in its fullest, more likely in minimal sign. It is vital to use for disabilities as everyone is focusing on just new stuff, but we think it should be used more appropriately. After all, if the technology is not helping someone, it is not required.

Background

There are several products available that use the same technique to detect hand gestures such as Red Panic Button, Myo Gesture Control Armband, Gest hand motion, Microsoft kinetic and Leap Motion, Real-time hand tracking with MediaPipe, EchoFlex: Hand gesture recognition using ultrasound imaging and so on. All these products use technologies to achieve some goals. Like the" Red Panic Button," it is based on smartphone devices. The Red Panic Button application is designed to improve the lives of all citizens by offering them a higher degree of security in our society. As a mobile device, the app provides users safety guidance in unknown environments (using a GPS option) and helps people feel confident and safe when moving or working[14]. But what if an old man who is 95 years old and can barely move his hand. How will he be able to find his smartphone and push the button in case of need? Another one is Myo Gesture Control Armband, which detects electrical activity in users' muscles and motion of their arms. There are five specific hand gestures that Myo can detect: wave left, wave right, fist, double tab, and finger spread, these gestures can be used to control presentation slides, games, piloting a drone, control music, and many more but if we use Myo for the patient or old people, then he/she has to wear a 93-gram device 24 hours and seven days, which is very uncomfortable. Also, Myo has limited

gesture detection, which can not detect any sign language like sign language for help[4]. On the other hand, Gest – a wearable device that allows users to control a computer or mobile device with their hands. Gest is an embedded device built with an inertial measurement unit to track gestures, a low energy Bluetooth sensor for a wireless connection, smart LEDs, rechargeable battery, and micro-USB for easy charging[10]. But it is also a bulky device which has 500mah battery inside and wearing it all the time will be very uncomfortable. The Leap Motion Controller is an optical hand tracking module that captures the movements of users' hands. It is a computer hardware sensor device that supports hand and finger motions as input, analogous to a mouse, but requires no hand contact or touching. Leap Motion is a very promising device for hand gesture detection, but there are no full devices for supporting old and disable people, and the price of a single Leap Motion device is $89.99, which is very high[11]. Moving on to MediaPipe, which is an open-source, cross-platform framework for building pipelines to process perceptual data of different modalities, such as video and audio. This approach provides high-fidelity hand and finger tracking by employing machine learning (ML) to infer 21 3D keypoints of a hand from just a single frame. But we didn't find any suitable product or system based on this platform either[12]. Up next, EchoFlex, which is a wearable device that uses ultrasound imaging to detect hand gestures. Their gesture recognition algorithm combines image processing and neural networks to classify the flexion and extension of 10 discrete hand gestures[13]. But they also have some issues in their project. Their device performance decreases due to cross-session position shifts. All these features are very innovative and for commercial uses, but here we wanted to develop a system that can help the helpless people to lead a much more comfortable life. That's why we made our system based on raspberry pi, and also we synced our policy with an android application that will update every scenario if necessary. In our project, we have used the raspberry pi camera to detect the hand gesture, which is connected to the database that is synced with the android application in real-time. For this, the user of the android application will be notified of every emergency movement at once, and for future medical purposes, these emergency messages will be stored in the database. People suffering from various diseases such as Parkinson's disease or certain paralysis, which makes people unable to move anywhere, will be benefited from this project if they need any kind of support or if they face any emergencies.

1.3 Project Goal

No.	Features/Functionalities/Usefulness
01	Users get to sign in using a signup panel to store data under their names. This way, they can be notified separately for different disabled personals.
02	We are going to detect hand gestures through a raspberry pi camera. The pi camera is small in size and easy to use, so no large camera is needed to recognize hand gestures.
03	Displays a message on the screen, depending on the gesture provided by the individuals.
04	An automated voice repeats the message if there's an emergency to warn others around the disabled or old individuals to get their attention. This way, if there's an emergency and the user is away from their cell phone, they also get notified.
05	Retrieves the message by detecting the hand gesture and sends these messages to the database. We have used the "Firebase" database provided by Google to make the database more flexible and easier to use. Firebase is free; that's why it reduces the cost of this project.
06	The database records the message under the user name provided by the user in the first place. This recorded message can help the user to identify the needs if there are multiple disabled or old people in their residents.
07	The database sends these data to an android app used by other individuals who are responsible or willing to help the disabled or old individual to notify them if there's an emergency or if the disabled or old individual needs any attention. This android app notifies other individuals using push notification.
08	The user of the Android can monitor multiple disabled or old individuals at the same time because the messages sent by the system are stored in the database under different user names.

CHAPTER 2

TECHNICAL DESIGN

2.1 Existing Solution

Hand gesture recognition is prevalent and widely used technology in recent years. Various companies and individuals are trying every day to use this technology more efficiently. Some products were made using this technology such as Myo Gesture Control Armband, Gest hand motion, Microsoft kinetic and Leap Motion, Real-time hand tracking with MediaPipe, EchoFlex: Hand gesture recognition using ultrasound imaging and so on. Let's see a brief discussion about these products and services.

Myo Gesture Control Armband:

Figure: 1

The company claims that this is the world's first input for digital devices where hackers and developers are only beginning to unlock the potential of Myo armband. There's a growing community of developers creating and sharing apps where users can explore in the Myo Market Beta. Users can map gestures to keystrokes for customized control. Developers can dig into open API and its free SDK to create custom scripts and applications that put technology at users' fingertips. But the problem here is that it is a wearable device and wearing some device all the time is very uncomfortable. It also has inconvenient warm-up time and calibration. And it costs almost 200 USD, which is a considerable amount to spend behind technology for many people. [3] [4]

Gest hand motion:

Figure: 2

Gest is a wearable device that allows users to control their computer or mobile device with your hands. The coolest part is that Gest is exceptionally versatile. Users can program custom gestures into actions on their devices. Just move with the user's hand, then tie it to any work they want. But the main focus of the developer was, they tried to make using portable devices like laptops, mobile phones totally hand free, which is not even the goal we wanted to achieve with our products. It's also in development mode and not ready to sell among the users[10].

Microsoft kinetic and Leap Motion Controller:

Figure: 3

The Leap Motion Controller is an optical hand tracking module that captures the movements of users' hands with unparalleled accuracy. Low processing power, a wide field of view, and near-zero latency. Whether you're an indie developer or a multinational company, the Leap Motion Controller makes human interaction in digital worlds natural and effortless. But it is a portable device that always has to connect with a computer or similar device to operate, and it has a low field of view, it can only function if the device detects hand gestures from one direction or one point of view. If it disconnects from the central computer, it is almost useless. Also, it costs nearly 89 USD, which is the double of our cost and seems unnecessary for many people to spend[11].

Real-time hand tracking with MediaPipe: MediaPipe is a framework for building multimodal (e.g., video, audio, any time series data), cross-platform (i.e., Android, iOS, web, edge devices) applied ML pipelines. With MediaPipe, a perception pipeline can be built as a graph of modular components, including, for instance, inference models (e.g., TensorFlow, TFLite) and media processing functions. MediaPipe is used by many internal Google products and teams, including Nest, Gmail, Lens, Maps, Android Auto, Photos, Google Home, and YouTube. But it is mostly a framework rather than a product, and though the products made out of it match ours, it's certainly not for the same causes yet, and also we didn't find any related product that competes with ours[12].

EchoFlex, Hand gesture recognition using ultrasound imaging: Ultrasound imaging has remained under-explored in the HCI community despite being non-invasive, harmless, and capable of imaging internal body parts, with applications including smart-watch interaction, prosthesis control, and instrument tuition. The developers compared the performance of different forearm mounting positions for a wearable ultrasonographic device. Location plays a fundamental role in ergonomics and performance since the anatomical features differ among positions. They also investigate the performance decrease due to cross-session position shifts and develop a technique to compensate for this misalignment. Their gesture recognition algorithm combines image processing and neural networks to classify the flexion and extension of 10 discrete hand gestures with an accuracy above 98%. But also as we have said before, it is a wearable device and still in a development stage. And the motive of those developers is to make wearable devices much more efficient to use, which doesn't match with our purposes [13].

2.2 Proposed Solution

In our project, we have tried to make a compact, easy to use, affordable and more likely efficient system that can help the disabled and old people to help with their daily life and to monitor them well enough so that those individuals who are responsible for them can take good care of them. Our system is based on a Raspberry pi and hand detection algorithms that detect hand gestures and sends signals to the designated android application to notify others. The main pros of our project are it's not that expensive, so people from almost every economic condition can afford this. Also, it comes with a dedicated android application that can monitor multiple people at the same time,

so it's not necessary to purchase more products or systems to look after multiple people who are suffering from physical disabilities. We have used a database to store our user information, and our system sends signals depending on the data between the disabled and the user on the other side of the application. Previously explained projects which are mostly similar to ours didn't use any kind of android application to retrieve the information from the patient to the user for better uses. Also, their products cost a lot than ours, and they didn't consider improving the situation of the disabled in any of their details. Our system detects hand gestures with a raspberry pi camera and using hand gesture recognition algorithms. It identifies some hand gestures trained by us. Then it sends the signal depending on the gestures and also sounds the alarm if necessary. Any individual who is responsible for the disabled person can look after the situation through an application and act accordingly. All these information stores in the database for a better understanding of the scenario and to detect the condition of multiple disabled people at the same time. Previously explained, products from different developers don't allow multiple users to use a single device, which is not exactly like ours.

In **figure: 04**, we have trained this symbol to our system that will be used for disabled people in case of emergency. This gesture will be recognized as the emergency symbol for disabled or deaf people as it is a universally accepted sign symbol (mainly American) for help.

Figure: 04

The gesture shown in the **figure: 05** is an emergency symbol for the adult person or any kind of disabled person who is seeking help.

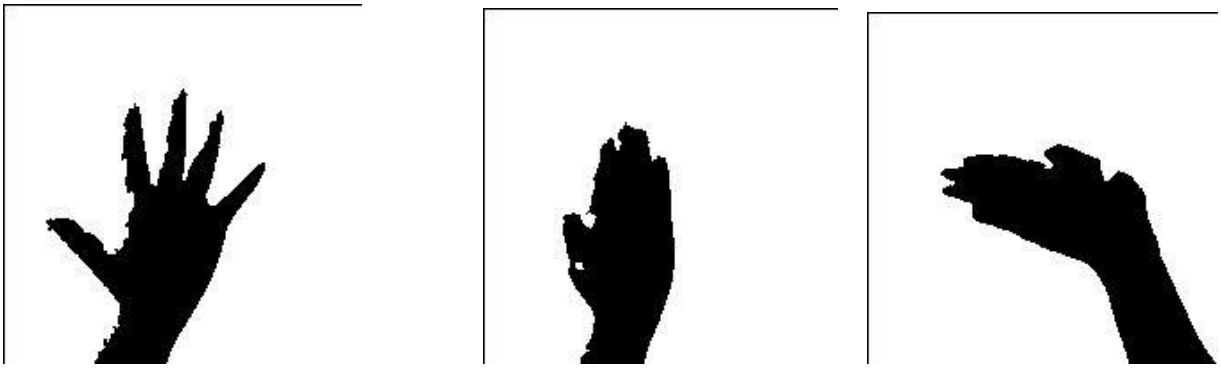


Figure:05

The gesture shown in the **figure: 06** is a help symbol for disabled or old people. When our system detects this symbol, it will call for help but won't ring any alarm.

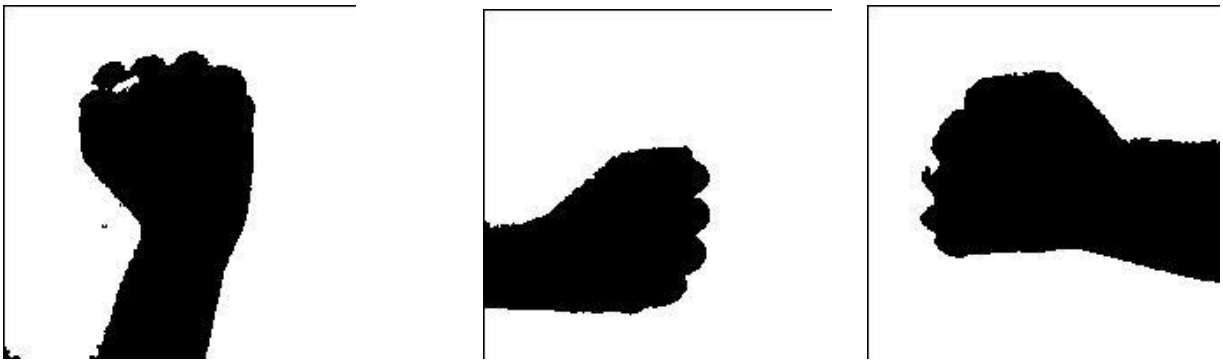


Figure: 06

2.3 Technical Design: Module Level

We can divide our project's module-level into three parts. Those are the android app, Hardware operating, and database. Here database is the bridge between Hardware and android application.

2.3.1. Hardware Operation

In our project, the hardware part is playing a crucial role. At first, the Hardware will take the user name for signup or login. Then it will start recording video and analyze it. At the same time, it will examine the footage and detect the hand gesture and will take some action based on the gesture type. It will also send the status to the database, and also it will show the state on the screen. We used Python language for hardware operations and used the Anaconda platform.

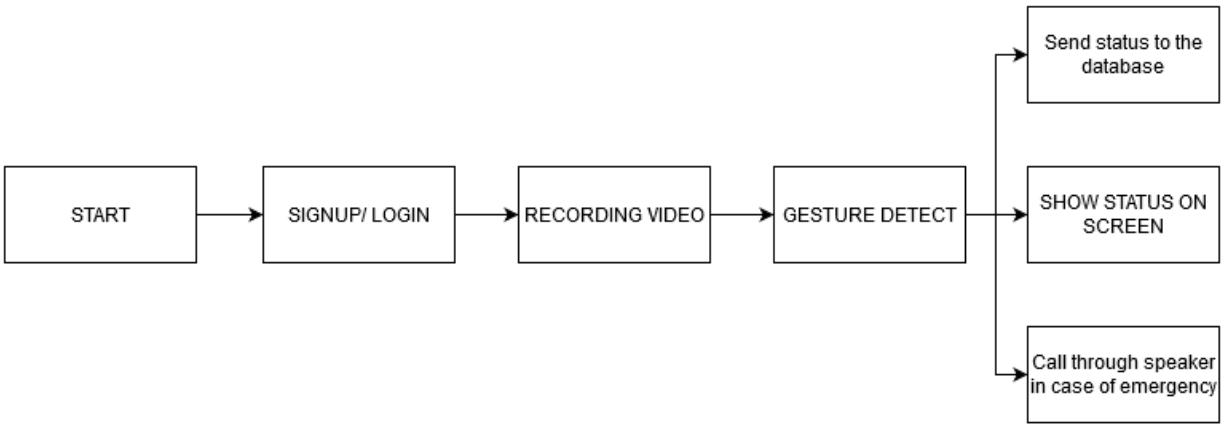


Figure: 07

2.3.1.1. Sign up and log in

Library: gTTs

Codes:

```
def deletems():
  tkWindow.destroy()
#window
tkWindow = Tk()
tkWindow.geometry('400x150')
tkWindow.title('ASSIST HARDWARE SETUP')
#username label and text entry box
usernameLabel = Label(tkWindow, text="User Name").grid(row=0, column=0)
username = StringVar()
usernameEntry = Entry(tkWindow, textvariable=username).grid(row=0, column=1)
#login button
loginButton = Button(tkWindow, text="START", command=deletems).grid(row=1, column=0)
tkWindow.mainloop()
```

2.3.1.2. Recording video

Library: OpenCV
Code:

```
cap = cv2.VideoCapture(0)
# Category dictionary
while True:
        _, frame = cap.read()
        # Simulating mirror image
        frame = cv2.flip(frame, 1)
        # Got this from collect-data.py
        # Coordinates of the ROI
        x1 = int(0.6*frame.shape[1])
        y1 = 80
        x2 = frame.shape[1]-80
        y2 = int(0.4*frame.shape[1])
        # Drawing the ROI
        # The increment/decrement by 1 is to compensate for the bounding box
        cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (0,0,255) ,2)
        # Extracting the ROI
        roi = frame[y1:y2, x1:x2]
        # Resizing the ROI so it can be fed to the model for prediction
        roi = cv2.resize(roi, (64, 64))
        roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
        cv2.imshow("test", test_image)
interrupt = cv2.waitKey(10)
        if interrupt & 0xFF == 27: # esc key
        break
cap.release()
cv2.destroyAllWindows()
```

2.3.1.3. Gesture detect

Library: Keras, Tensorflow
Code:

```
# Loading the model
json_file = open("model-bw.json", "r")
model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(model_json)
```

```
# load weights into a new model
loaded_model.load_weights("model-bw.h5")
# Batch of 1
        result = loaded_model.predict(test_image.reshape(1, 64, 64, 1))
        prediction = {'NORMAL': result[0][0],
                'EMERGENCY': result[0][1],
                'EMERGENCY': result[0][2],
                'HELP': result[0][3]}
        # Sorting based on top prediction
        prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
```

2.3.1.4. Show status on screen

Library: OpenCV
Code:

```
 # Displaying the predictions
        cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 2, (225,0,0),
1)
        cv2.imshow("Frame", frame)
```

2.3.1.5. Send status to the database

Library: Firebase
Code:

```
firebase = firebase.FirebaseApplication('https://assist-5d80b.firebaseio.com/', None)
 try:
        firebase.put(username.get(),'Values',prediction[0][0])
        except:
        print("Setup Failed")
        break
```

2.3.1.6. Call through Speaker

Library:
Code:

```
#Audio Setup
myText = "Emergency"
language = 'en'
output = gTTS(text=myText, lang=language, slow=False)
output.save("output.mp3")
```

3.3.1.7. Full Hardware Code

Data Collect for Train:
Library: OpenCV
Code:

```
import cv2
import numpy as np
import os
#it is like creating a folder
if not os.path.exists("data"):
        os.makedirs("data")
        os.makedirs("data/train")
        os.makedirs("data/train/B")
        os.makedirs("data/train/ED")
        os.makedirs("data/train/EO")
        os.makedirs("data/train/H")
# Train directory loaded
directory = 'data/train/'
cap = cv2.VideoCapture(0) #it is like my webcam is activated
while True:
        _, frame = cap.read()
        # Simulating mirror image
        frame = cv2.flip(frame, 1)
        # Getting a count of existing images
        count = {'BACKGROUND': len(os.listdir(directory+"/B")),
        'EMDES': len(os.listdir(directory+"/ED")),
        'EMOLD': len(os.listdir(directory+"/EO")),
        'HELP': len(os.listdir(directory+"/H"))}
        # Printing the count in each set to the screen
        cv2.putText(frame, "MODE : TRAIN", (10, 50), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,0),
1)
```

```
        cv2.putText(frame, "IMAGE COUNT", (10, 100), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,0), 1)
        cv2.putText(frame, "Background : "+str(count['BACKGROUND']), (10, 120),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,0), 1)
        cv2.putText(frame, "Emergency Diasable : "+str(count['EMDES']), (10, 140),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,0), 1)
        cv2.putText(frame, "Emergency OLD : "+str(count['EMOLD']), (10, 160),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,0), 1)
        cv2.putText(frame, "Help : "+str(count['HELP']), (10, 180), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,0), 1)
        # Coordinates of the ROI
        x1 = int(0.6*frame.shape[1])
        y1 = 80
        x2 = frame.shape[1]-80
        y2 = int(0.4*frame.shape[1])
        # Drawing the ROI
        # The increment/decrement by 1 is to compensate for the bounding box
        cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (0,0,255) ,2)
        # Extracting the ROI
        roi = frame[y1:y2, x1:x2]
        roi = cv2.resize(roi, (200, 200))#SIZE OF ROI THE LITTLE GRAYSCALE WINDOW
        cv2.imshow("Frame", frame)
        roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        _, roi = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
        cv2.imshow("ROI", roi)
        interrupt = cv2.waitKey(10)
        if interrupt & 0xFF == 27: # esc key
        break
        if interrupt & 0xFF == ord('b'):
        cv2.imwrite(directory+'B/'+str(count['BACKGROUND'])+'.jpg', roi)
        if interrupt & 0xFF == ord('d'):
        cv2.imwrite(directory+'ED/'+str(count['EMDES'])+'.jpg', roi)
        if interrupt & 0xFF == ord('o'):
        cv2.imwrite(directory+'EO/'+str(count['EMOLD'])+'.jpg', roi)
        if interrupt & 0xFF == ord('h'):
        cv2.imwrite(directory+'H/'+str(count['HELP'])+'.jpg', roi)
cap.release()
cv2.destroyAllWindows()
```

Data Train:
Library: Keras
Code:

```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=4, activation='softmax')) # softmax for more than 2

# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) #
categorical_crossentropy for more than 2


# Step 2 - Preparing the train/test data and training the model
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('data/train',
                                                 target_size=(64, 64),
                                                 batch_size=5,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('data/test',
                                            target_size=(64, 64),
                                            batch_size=5,
                                            color_mode='grayscale',
                                            class_mode='categorical')
classifier.fit_generator(
        training_set,
        steps_per_epoch=1000, # No of images in training set
        epochs=10,
        validation_data=test_set,
        validation_steps=40)# No of images in the test set


# Saving the model
model_json = classifier.to_json()
with open("model-bw.json", "w") as json_file:
        json_file.write(model_json)
classifier.save_weights('model-bw.h5')
```

Hardware:

```
import numpy as np
from keras.models import model_from_json
from firebase import firebase
from tkinter import *
from functools import partial
from gtts import gTTS
import operator
import cv2
import sys, os
def deletems():
  tkWindow.destroy()
#window
```

```
tkWindow = Tk()
tkWindow.geometry('400x150')
tkWindow.title('ASSIST HARDWARE SETUP')
#username label and text entry box
usernameLabel = Label(tkWindow, text="User Name").grid(row=0, column=0)
username = StringVar()
usernameEntry = Entry(tkWindow, textvariable=username).grid(row=0, column=1)


#login button
loginButton = Button(tkWindow, text="START", command=deletems).grid(row=1, column=0)
tkWindow.mainloop()
firebase = firebase.FirebaseApplication('https://assist-5d80b.firebaseio.com/', None)
# Loading the model
json_file = open("model-bw.json", "r")
model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(model_json)
# load weights into new model
loaded_model.load_weights("model-bw.h5")
#print("Loaded model from disk")
#Audio Setup
myText = "Emergency"
language = 'en'
output = gTTS(text=myText, lang=language, slow=False)
output.save("output.mp3")
cap = cv2.VideoCapture(0)

# Category dictionary
while True:
        _, frame = cap.read()
        # Simulating mirror image
        frame = cv2.flip(frame, 1)
        # Got this from collect-data.py
        # Coordinates of the ROI
        x1 = int(0.6*frame.shape[1])
        y1 = 80
        x2 = frame.shape[1]-80
        y2 = int(0.4*frame.shape[1])
        # Drawing the ROI
        # The increment/decrement by 1 is to compensate for the bounding box
        cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (0,0,255) ,2)
        # Extracting the ROI
```

```
        roi = frame[y1:y2, x1:x2]
        # Resizing the ROI so it can be fed to the model for prediction
        roi = cv2.resize(roi, (64, 64))
        roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
        cv2.imshow("test", test_image)


        # Batch of 1
        result = loaded_model.predict(test_image.reshape(1, 64, 64, 1))
        prediction = {'NORMAL': result[0][0],
                      'EMERGENCY': result[0][1],
                      'EMERGENCY': result[0][2],
                      'HELP': result[0][3]}
        # Sorting based on top prediction
        prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
        # Displaying the predictions
        cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 2, (225,0,0),
1)
        cv2.imshow("Frame", frame)
#AUDIO PLAY
        if(prediction[0][0]== 'EMERGENCY'): os.system("start output.mp3")
        Try:
#Send Status To Firebase
        firebase.put(username.get(),'Values',prediction[0][0])
        except:
        print("Setup Failed")
        break

        interrupt = cv2.waitKey(10)
        if interrupt & 0xFF == 27: # esc key
        break

cap.release()
cv2.destroyAllWindows()
```

2.3.2. Database

As we mentioned before, the database is the bridge between Hardware and android application. We are sending the status from Hardware to the database, which is doing our project online. Users can easily access our database using our android app from anywhere. Besides, the database will also help us to track history in the future.

Figure: 08

2.3.2.1. Database Security Code

```
{
  "rules": {
          ".read": true,
          ".write": true
  }
}
```

2.3.3. Android App

We have built an android app named "Assist" to make our project more user friendly. The android app was developed based on JAVA, and we used the Android Studio platform to build the app. Using our android app, users can know the status of the patient from any corner of the world where he or she has an internet connection. Users have to open the app and provide a username. Then the app will show the user his patient's status. In case of an emergency, the app will send the user multiple push notifications to grab his or her attention. Another exciting feature is that users can track various patients using the same app.

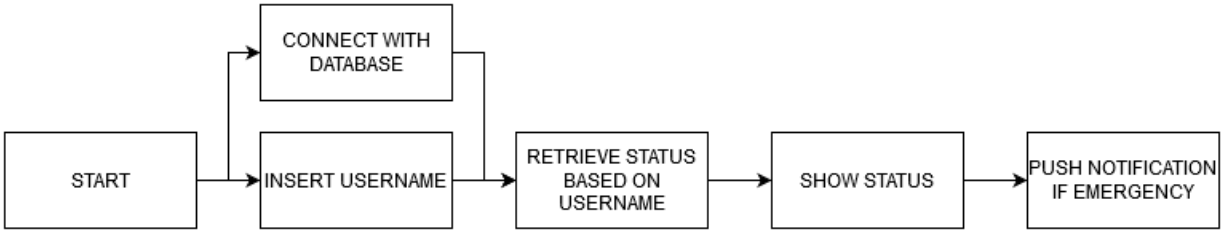


Figure: 09

2.3.3.1. Splash Screen

Class: splashscreen.java

```
package com.cse.assist;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageView;

public class splash screen extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splashscreen);

    //Showing splash screen then going to the login page
    final ImageView imageView = (ImageView) findViewById(R.id.imageView);
    Thread mythread = new Thread(){
      @Override
      public void run() {
        try {
          sleep(4000);
          Intent intent = new Intent(getApplicationContext(),MainActivity.class);
          startActivity(intent);
          finish();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    };
    mythread.start();
  }
}
```

2.3.3.2. Insert Username

Class: MainActivity.java

```
EditText Status, Message,Username;
Username=findViewById(R.id.username);
DatabaseReference Botstatus=database.getReference(String.valueOf(Username.getText()));
```

2.3.3.3. Connect With Database

Class: MainActivity.java

```
FirebaseDatabase database=FirebaseDatabase.getInstance();
DatabaseReference Botstatus=database.getReference(String.valueOf(Username.getText()));
```

2.3.3.4. Retrieve Data From Database

Class: MainActivity.java

```
Botstatus.addValueEventListener(new ValueEventListener() {
  final String temp[]=new String[1];
  @RequiresApi(api = Build.VERSION_CODES.O)
  @Override
  public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
     int i = 0;
     for (DataSnapshot data : dataSnapshot.getChildren()) {
        temp[i] = data.getValue().toString();
        i++;
     }
}
  @Override
  public void onCancelled(@NonNull DatabaseError databaseError) {
  }
});
```

2.3.3.5. Show Status

Class: MainActivity.java

```
Status.setText(temp[0]);
try {
  if(temp[0].equals("NORMAL")){
     Status.setTextColor(Color.parseColor("#00ff00"));
     Status.setText("NORMAL");
     Message.setTextColor(Color.parseColor("#00ff00"));
     Message.setText("Situation is normal. No need to worry about anything. Thank you for being with
The Binary Knight");
  }
  else if (temp[0].equals("EMERGENCY")){
     Status.setTextColor(Color.parseColor("#ff0000"));
     Status.setText("EMERGENCY");
     Message.setTextColor(Color.parseColor("#ff0000"));
     Message.setText("Situation is critical. You must go to your patient right now.");
       }
  else if(temp[0].equals("HELP")){
     Status.setTextColor(Color.parseColor("#ffff00"));
     Status.setText("HELP");
     Message.setTextColor(Color.parseColor("#ffff00"));
     Message.setText("No need to panic. You should go to your patient. Your help is needed");
  }
```

2.3.3.6. Push Notifications

Class: MainActivity.java

```
//PUSH NOTIFICATION
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
// The id of the channel.
String id = "my_channel_01";
// The user-visible name of the channel.
CharSequence name = getString(R.string.channel_name);
// The user-visible description of the channel.
String description = getString(R.string.channel_description);
int importance = NotificationManager.IMPORTANCE_LOW;
NotificationChannel mChannel = new NotificationChannel(id, name,importance);
// Configure the notification channel.
mChannel.setDescription(description);
mChannel.enableLights(true);
// Sets the light notification color for notifications posted to this
// channel, if the device supports this feature.
mChannel.setLightColor(Color.RED);
mChannel.enableVibration(true);
mChannel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300, 200, 400});
mNotificationManager.createNotificationChannel(mChannel);
mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
// Sets an ID for the notification, so it can be updated.
int notifyID = 1;
// The id of the channel.
String CHANNEL_ID = "my_channel_01";
// Create a notification and set the notification channel.
Notification notification = new Notification.Builder(MainActivity.this)
      .setContentTitle("Alert")
      .setContentText("Situation is critical.")
      .setSmallIcon(R.drawable.logo)
      .setChannelId(CHANNEL_ID)
      .build();
// Issue the notification.
mNotificationManager.notify(notifyID, notification);
```

2.3.3.7. Front End

2.3.3.7.1. Preview

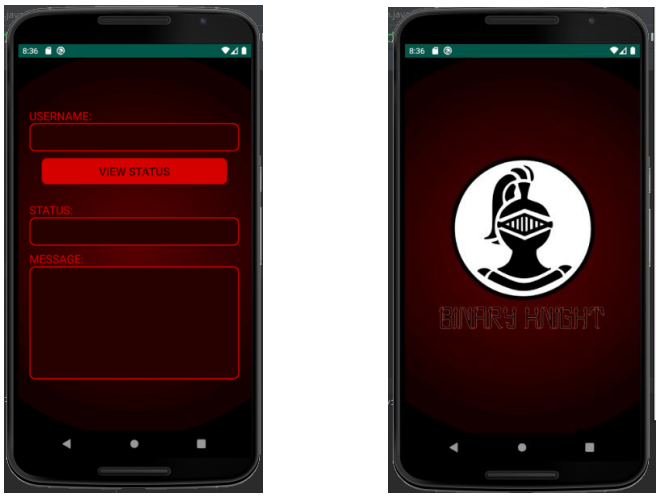


Figure: 10

2.3.3.7.2. Codes

activity_splashscreen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".splashscreen"
  android:orientation="vertical"
  android:gravity="center"
  android:background="@drawable/splashscreen_background"
  >

  <ImageView
  android:id="@+id/imageView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:src="@drawable/sspic"
  android:layout_marginLeft="60dp"
```

```
  android:layout_marginRight="60dp"
  />
</LinearLayout>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity"
  android:gravity="center"
  android:orientation="vertical"
  android:background="@drawable/splashscreen_background"
  >
  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginLeft="20dp"
     android:layout_marginRight="20dp"
     android:gravity="left"
     android:text="USERNAME:"
     android:textSize="20dp"
     android:textColor="#D50000"
     />
  <EditText
     android:id="@+id/username"
     android:layout_marginLeft="20dp"
     android:layout_marginRight="20dp"
     android:layout_width="match_parent"
     android:layout_height="50dp"
     android:background="@drawable/edittext_background"
     android:gravity="center"
     android:textColor="#23FF00"
     android:textSize="20dp"
     />
  <Button
     android:layout_marginTop="10dp"
     android:id="@+id/viewstatus"
```

```
    android:layout_marginLeft="40dp"
    android:layout_marginRight="40dp"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="@drawable/button_background"
    android:gravity="center"
    android:textColor="#270000"
    android:textSize="20dp"
    android:text="View Status"

    />
<TextView
    android:layout_marginTop="30dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:gravity="left"
    android:text="STATUS:"
    android:textSize="20dp"
    android:textColor="#D50000"
    />
<EditText
    android:id="@+id/status"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="@drawable/edittext_background"
    android:gravity="center"
    android:textColor="#23FF00"
    android:textSize="20dp"
    />
<TextView
    android:layout_marginTop="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:gravity="left"
    android:text="MESSAGE:"
    android:textSize="20dp"
    android:textColor="#D50000"
```

```
    />
  <EditText
    android:id="@+id/message"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:background="@drawable/edittext_background"
    android:gravity="center"
    android:textSize="20dp"
    />
</LinearLayout>
```

2.3.3.8. Back End

Class: MainActivity.java

```
package com.cse.assist;
import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
public class MainActivity extends AppCompatActivity {
  FirebaseDatabase database=FirebaseDatabase.getInstance();
  EditText Status, Message,Username;
  Button ViewStatus;
```

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     Username=findViewById(R.id.username);
     Status = findViewById(R.id.status);
     Message = findViewById(R.id.message);
     ViewStatus=findViewById(R.id.viewstatus);
     ViewStatus.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
//Retrieving Data From Firebase
           DatabaseReference Botstatus=database.getReference(String.valueOf(Username.getText()));
           Botstatus.addValueEventListener(new ValueEventListener() {
              final String temp[]=new String[1];
              @RequiresApi(api = Build.VERSION_CODES.O)
              @Override
              public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                 int i = 0;


                 for (DataSnapshot data : dataSnapshot.getChildren()) {
                    temp[i] = data.getValue().toString();
                    i++;
                 }
                 Status.setText(temp[0]);

                 try {
                    if(temp[0].equals("NORMAL")){
                       Status.setTextColor(Color.parseColor("#00ff00"));
                       Status.setText("NORMAL");
                       Message.setTextColor(Color.parseColor("#00ff00"));
                       Message.setText("Situation is normal. No need to worry about anything. Thank you
for being with The Binary Knight");
                    }
                    else if (temp[0].equals("EMERGENCY")){
                       Status.setTextColor(Color.parseColor("#ff0000"));
                       Status.setText("EMERGENCY");
                       Message.setTextColor(Color.parseColor("#ff0000"));
                       Message.setText("Situation is critical. You must go to your patient right now.");
                       //PUSH NOTIFICATION
                       NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
```

```
                    // The id of the channel.
                    String id = "my_channel_01";
                    // The user-visible name of the channel.
                    CharSequence name = getString(R.string.channel_name);
                    // The user-visible description of the channel.
                    String description = getString(R.string.channel_description);
                    int importance = NotificationManager.IMPORTANCE_LOW;
                    NotificationChannel mChannel = new NotificationChannel(id, name,importance);
                    // Configure the notification channel.
                    mChannel.setDescription(description);
                    mChannel.enableLights(true);
                    // Sets the notification light color for notifications posted to this
                    // channel, if the device supports this feature.
                    mChannel.setLightColor(Color.RED);
                    mChannel.enableVibration(true);
                    mChannel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300, 200,
400});
                    mNotificationManager.createNotificationChannel(mChannel);
                    mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
                    // Sets an ID for the notification, so it can be updated.
                    int notifyID = 1;
                    // The id of the channel.
                    String CHANNEL_ID = "my_channel_01";
                    // Create a notification and set the notification channel.
                    Notification notification = new Notification.Builder(MainActivity.this)
                            .setContentTitle("Alert")
                            .setContentText("Situation is critical.")
                            .setSmallIcon(R.drawable.logo)
                            .setChannelId(CHANNEL_ID)
                            .build();
                    // Issue the notification.
                    mNotificationManager.notify(notifyID, notification);
                }
                else if(temp[0].equals("HELP")){
                    Status.setTextColor(Color.parseColor("#ffff00"));
                    Status.setText("HELP");
                    Message.setTextColor(Color.parseColor("#ffff00"));
                    Message.setText("No need to panic. You should go to your patient. Your help is
needed");
                }
            }catch (Exception e){
                Status.setTextColor(Color.parseColor("#000000"));
```

```
                    Status.setText("An unknown error occurred");
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });
        //Retrieving Data ends
    }
  });
 }
}
```

2.4 Technical Design: System Level

2.4.1. Block Diagram

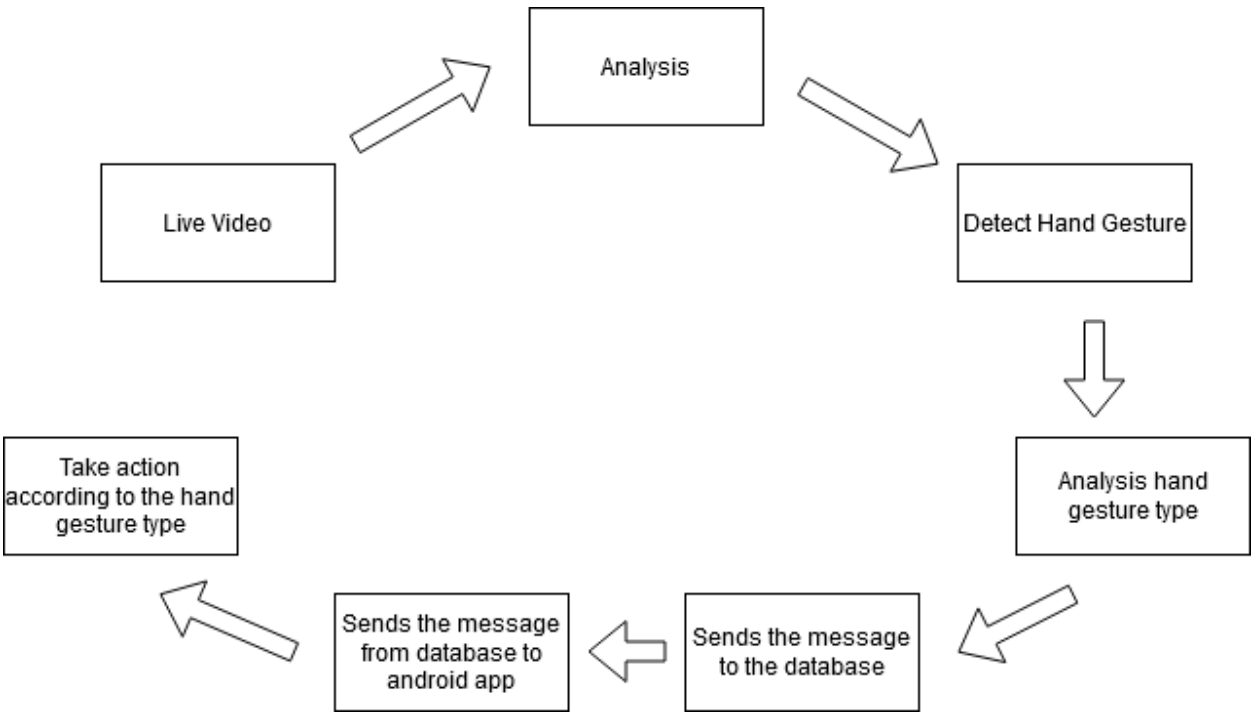


Figure: 11

The basic idea is to retrieve images from the video and detect hand gestures. Depending on the hand gesture, our system will compare it with existing training data and will tell us the type of detection it detects. Then this data will be transmitted to the database where it will notify the user through push notification that if the disabled or old individuals need any kind of assistance.

2.4.2. Component Diagram

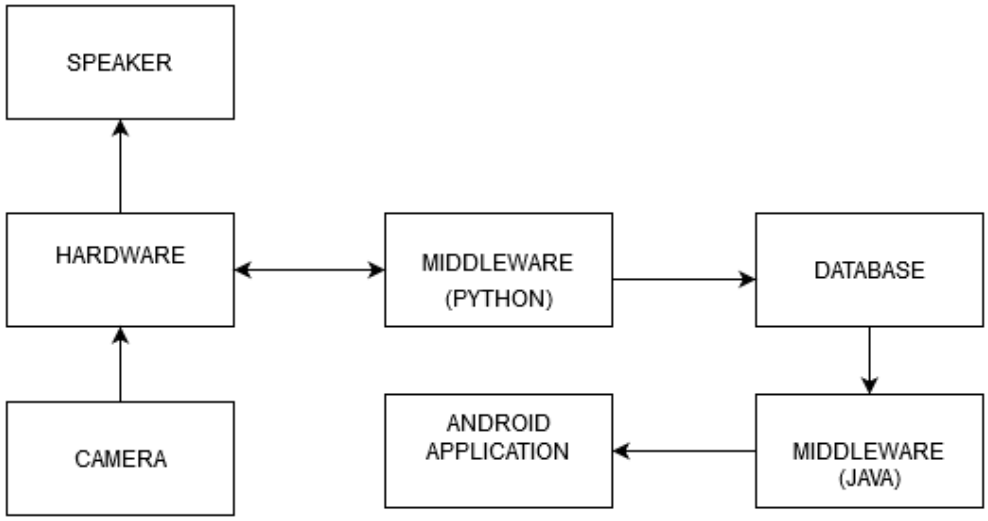


Figure: 12

Our system based on machine learning, which is based on Python, and android studio, mainly based on Java. The database connects these two environments to fulfill our target. A camera detects the images, and the algorithm filters the image through greyscale. All this process is completed by Raspberry pi, which we use as our base operating component.

2.4.3 Flow Chart

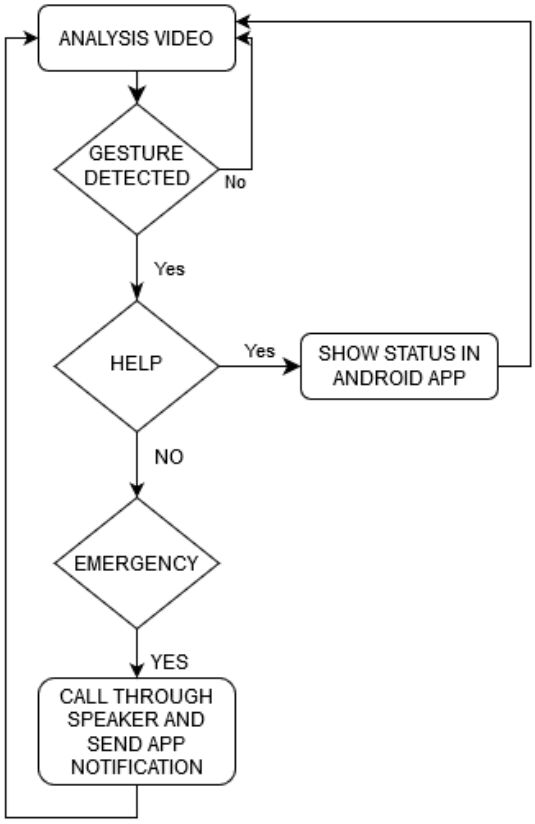


Figure: 13

The functionality of our system is quite simple. Our policy will first try to detect hand gestures through Pi-cam. If it detects any gesture, then it will check if the gesture shows any **Normal Call** by comparing it with previously inserted training data. If it is a Normal Call, then it will ring the alarm, but if it is not the **Normal Call**, then it must be an **Emergency Call,** and it will immediately notify the user who is using the android app and can call for medical assistance.

2.4.4 Hardware Connection:

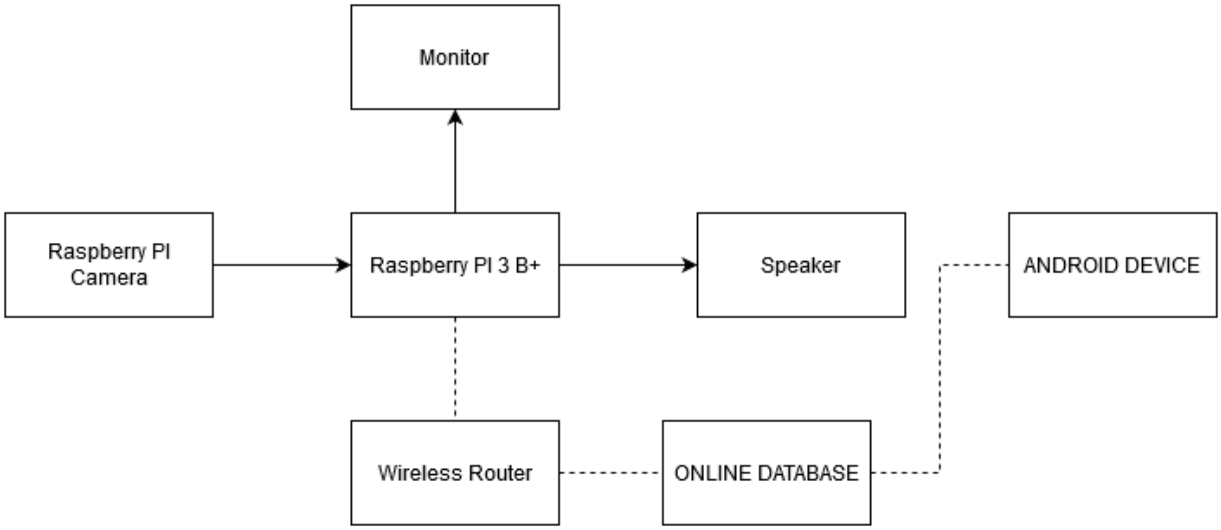


Figure: 14

We are using Raspberry PI 3 B+ as our primary computing device. Where we are running our python code to operate all the instruments and send data to the database, for recording video, we connected a raspberry pi camera with Raspberry PI 3 B+. or sending the data to the database Raspberry PI must be connected with a router for internet access. We also need a display (Monitor) for watching the output of the camera, and we also have to connect a speaker for emergency warning. Raspberry PI will capture live video using a Raspberry Pi camera. Then it will process the video and detect the hand gesture, then it will show the predicted value on screen and will send the value to the database through the internet. In case of emergency status, the Raspberry PI will also play voice alarm using the speaker. For viewing the state from the database, anyone with the android application named 'ASSIST' will be able to see the patient status from any corner of the world. Only he or she will need an internet connection and login information.

2.5 Required Skills

There are some components we have used to complete this project. They are given below.

- Hardware
    - Raspberry Pi
    - Raspberry Pi Camera
    - Speaker
- Software
    - Anaconda
    - Android Studio
- Programming language
    - Python
    - Java

- Database
    - Firebase

As a base component, we have used Raspberry pi to operate the whole system, and for storing all the information, we have used "Firebase," a free database storage by Google, to make the android application, we have used "Android Studio," and there, we have used Java programming language. For the primary coding and compiling, we have used "Anaconda," which is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Also, we have used a Raspberry Pi camera to detect hand gestures.

Chapter 3: Essential parts and Devices

3.1. Description of Components

We have used some hardware and software to complete this project. Those are given below.

- Hardware
    - Raspberry Pi
    - Raspberry Pi Camera
    - Speaker
- Software
    - Anaconda
    - Android Studio
- Database
    - Firebase

Raspberry Pi:

The Raspberry Pi(figure: 13) is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a little capable device that enables people of all ages to explore computing and to learn how to program in languages like Scratch and Python. It's capable of doing everything anyone expects a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

Figure: 15

Raspberry Pi camera:

The **Pi camera (figure: 13)** module is a portable, lightweight **camera** that supports Raspberry **Pi**. It communicates with **Pi** using the MIPI **camera** serial interface protocol. It is usually used in image processing, machine learning, or surveillance projects.

Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

Android Studio:

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

Firebase:

Firebase is Google's mobile application development platform that helps anyone to build, improve, and grow their app. Firebase Storage has its system of security rules to protect users' GCloud bucket from the masses while granting full write privileges to their authenticated clients.

Chapter 4: Working Sheets

4.1 Work Breakdown Structure

WEEK	DATE	Expected Progress
Week 1	26-JAN-2020	Project Deciding
Week 2	2-FEB-2020	Project Deciding
Week 3	9-FEB-2020	Project Deciding
Week 4	16-FEB-2020	Project Proposal Submission and Environment Setup
Week 5	23-FEB-2020	Start Work on ML and How to Detect Hand Gesture
Week 6	1-MAR-2020	Hand Detect and Start Work on Mobile Application
Week 7	8-MAR-2020	Mobile Application Stage 1 Ready
Week 8	15-MAR-2020	Hand Gesture Detect

Week 9	22-MAR-2020	Hand Gesture Detect and Show Conclusion
Week 10	29-MAR-2020	Send Hand Gesture Conclusion to the Database and Connect it With the app
Week 11	5-APR-2020	Project Ready and Start Debug
Week 12	12-APR-2020	Final Project and Report Submission

4.2 Financial Plan and Costs

The estimated cost of the equipment we are going to use in our project is given below.

No.	Name of the Equipment	Estimated Cost
01	Raspberry Pi 3 B+	3300 BDT
02	Raspberry Pi Camera	700 BDT
03	Speaker	200 BDT
04	Adapter	120 BDT
05	Database	0 BDT
06	Android application	0 BDT
	Total estimated cost	**4320 BDT**

Chapter 5: Project Summary

5.1 Result and Discussion

So our system starts with training the data we previously inserted in the input, then when we run the train data, it will prepare itself for the output of similar images. After that, when we run the "Detect" file, it will show a popup login form(**figure: 16**) where the user will write his/her name, and this will be stored in the database. Under the user name, all the messages of gesture signals will be saved. As you can see in **figure: 17**, there is a background with no gesture which stores in the database under the user name, and the data then sends the signal to the android application where the other will be notified.

Figure: 16

In **figure: 17**, the background shows as a "Normal" message, and it sends this message to the database, which forwards this message to the application.

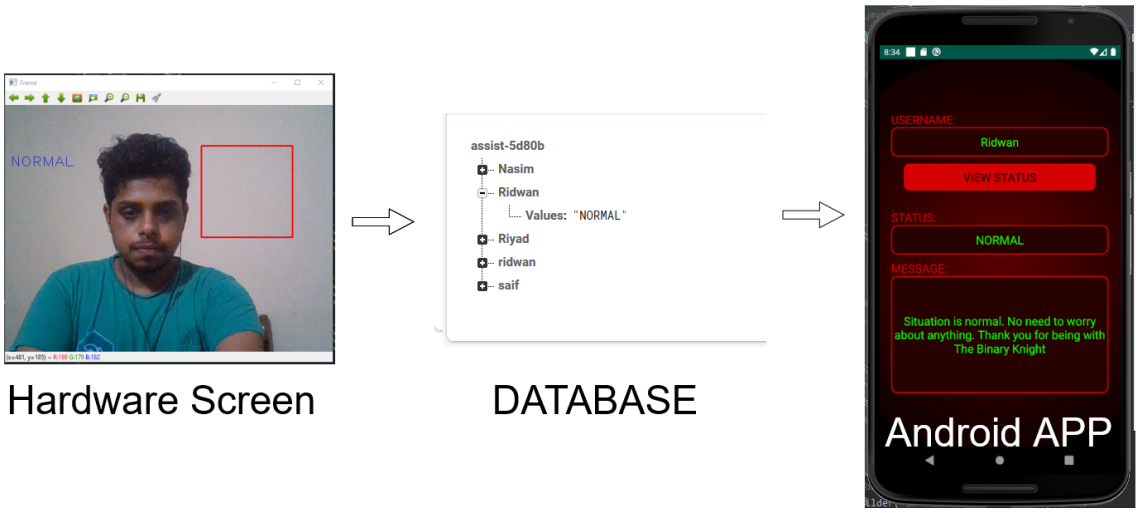


Figure: 17

If he shows his fist(**figure: 18**) in the designated section in the frame, then based on the trained data, it will detect the "HELP" sign, and like before, it will send this message to the database. Then the database will send this message to the android application and show a message to the user.

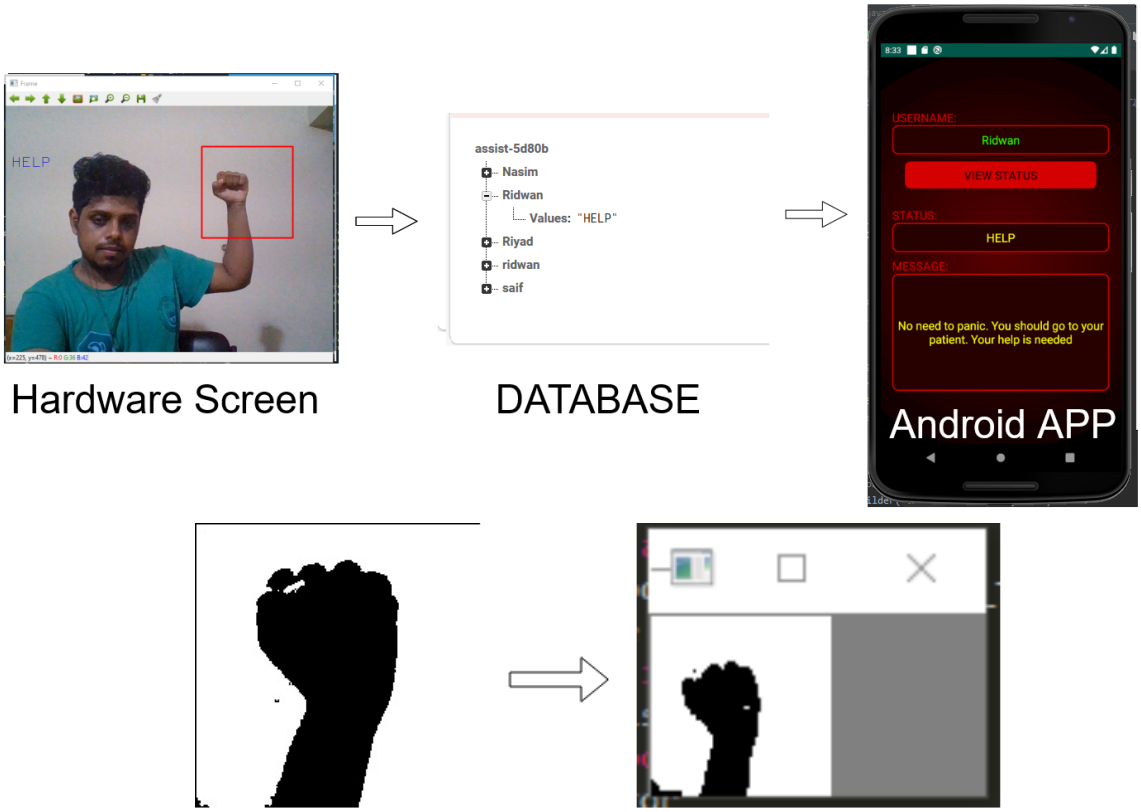
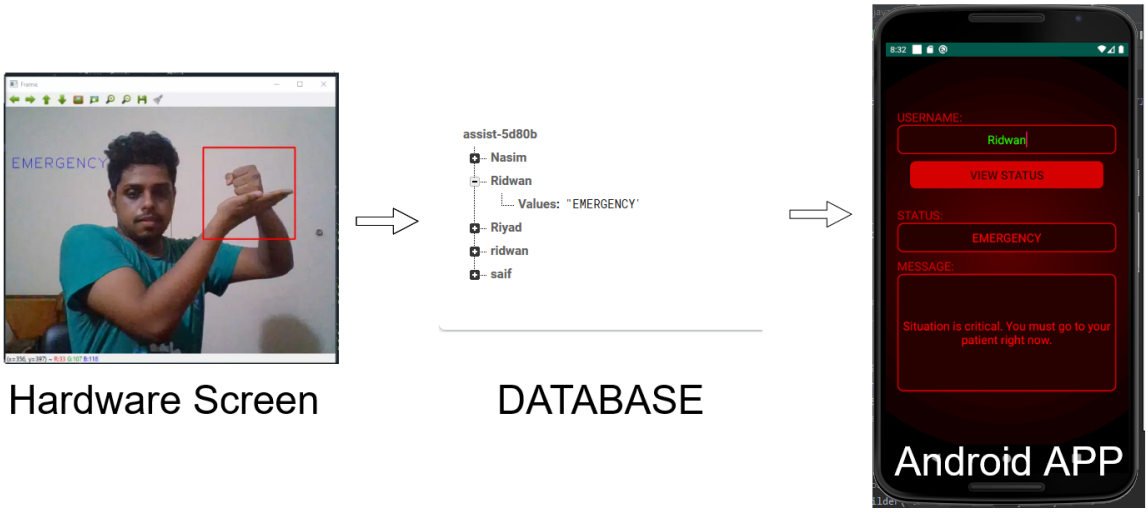


Figure: 18

Like this, if the user shows this gesture shown in **figure: 19,** then it will detect the gesture as an "Emergency" sign and will send the message to the database and so on.



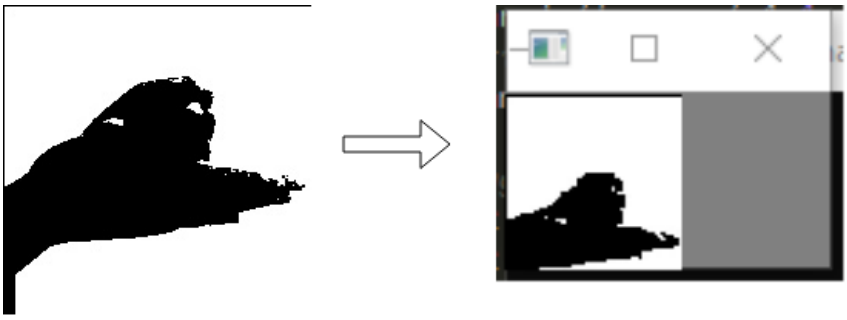
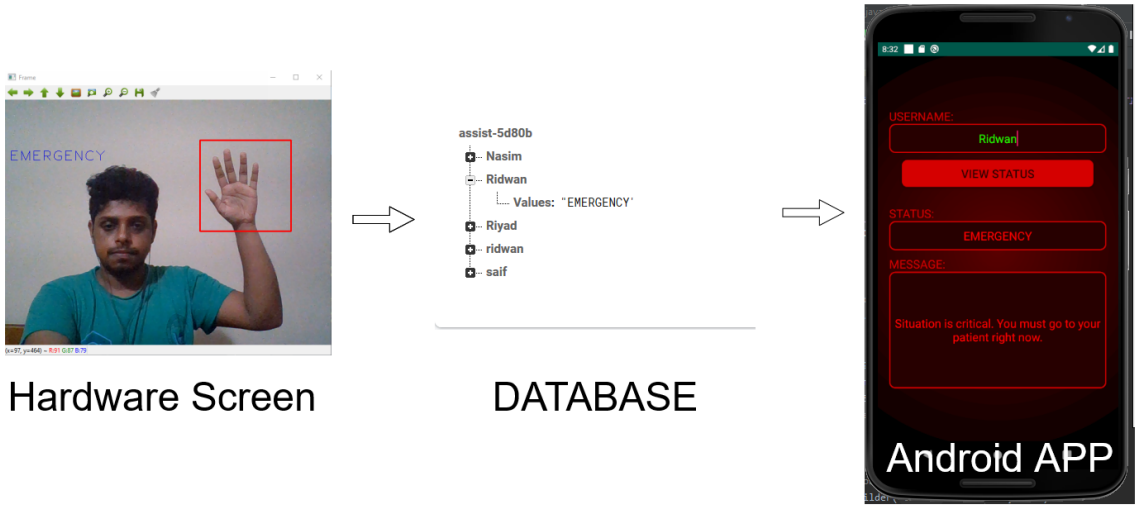


Figure: 19

And lastly, if the user shows his palm(**figure: 20**), it will also detect an "Emergency" sign and will send a message to the database under that user name and will show the result in the android application.



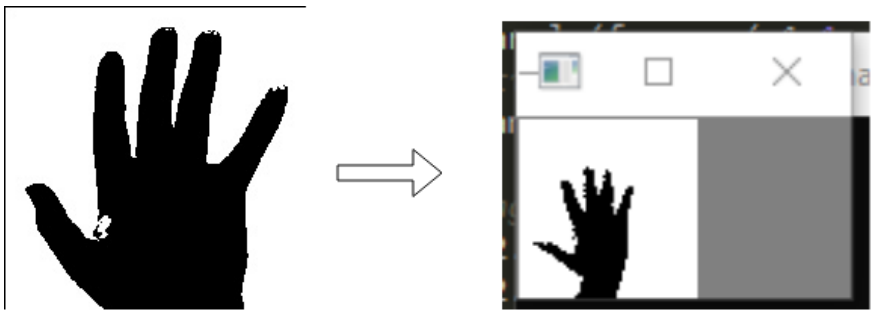


Figure: 20

And this is how our project detects hand gestures and reacts, as shown above. The greyscale images are the train data and the output data from detection. On the left side in every figure, we can see the train data, and on the right side, we can see the output data after detecting the hand gesture. These grayscale images are compared by the machine learning algorithm and show the result according to the input.

5.2 Feasibility Study

The result of this project demonstrates the feasibility of hand gesture recognition based on raspberry pi. The gesture recognition studies show that it has a wide range of possibility nowadays around the world. From driving a car to home automation, from running a computer to operating critical surgery, hand gestures show its importance in every aspect of modern technology. In our project, we tried to build a system that helps the old and disabled to lead their daily life more comfortable, and it shows a great opportunity in recent days. In case of financial feasibility, we can say that, among many types of equipment that uses the same technique like us, our project is a very low-cost project, and as we have tried to build the system for maximum people to use, it will be beneficial for them if it costs as less as possible. Currently, many varieties of cutting edge technology are flooding our market, but most of them are quite expensive and require some suitable knowledge to use. In that case, our project is also ahead of those terms and very easy to use and requires very little prior knowledge of this technological equipment. In the case of operational feasibility, we are not sure that our project will show 100% accuracy in practical uses, but it will perform above average, which is not bad in the beginning stages. If we want to reduce the cost of this project even more, then we need to mass-produce this product as it will decrease the price of every single unit of the product.

5.3 Problem Faced and Solutions

Detecting hand gestures is not an easy task to complete if you don't have any prior knowledge of specific machine learning techniques and algorithms. We also faced this problem as we didn't have any previous experience of machine learning, but from time to time, we adapted ourselves in this situation and were able to complete the base of the detection of hand gestures. After building up the codes, it was time to increase the accuracy of the exposure as it was deficient in our first phase. After that, we had to connect the database with our python hand detection codes using Spyder(a compiler in Anaconda distribution), which made our project hard to complete in time, but eventually, we sorted it out. And the last problem we faced was during connecting our android application with our project because it required a specific library that we didn't know about. After searching for hours, we solved this problem too.

5.4 Future Development

There is a vast area to develop in these hand detection criteria. Hand detection nowadays becomes a fascinating subject for many developing countries and also to hundreds of developers because they can create many opportunities using this method. Like this, we also have some development ideas that require some time and more funds to take this project forward. Such as, there are only three gestures that our system can detect, but if we have some time to improve it, we can surely add some more gestures to express emotions broadly. Also, our system only operates in one language (English), but we have thought of adding some more languages so that people from different countries or areas can select their language and use it their way. If we can develop our system a little bit and if we could make it more stable, we can use it in large sectors like in hospitals or public places for civilian protection purposes. If we use this in hospitals, we can monitor multiple people at the same time, and as we have a shortage of medical personnel, it would be time-consuming and life-saving. If we use it in public places, civilians will feel much more secure as it will help them in case of emergency in late hours or at any time if they need any kind of attention. So, as you see, there are a lot of areas we can still develop in our project.

5.5 Conclusion

Technology is advancing day by day, and it requires much more every day from us. With the help of many technological methods, developers and also students are inventing and adding new approaches to the existing one, which creates a significant opportunity for many people. These days, we can see many cutting-edge technologies around the world, which is made possible by this ever-growing knowledge. Our project was also possible for this kind of method, and we are trying to make it more applicable and efficient for disabled and old people who nowadays lead a hard life. Technology teaches us to learn something new and to make the best out of it so that we can make this world a better place. So, we tried to create something new out of it and also tried to make a system that will help the people who need it the most.

REFERENCES

[1] A. Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, "Hand Gesture Recognition for Human Computer Interaction," *Procedia Computer Science*, vol. 115, pp. 367–374, 2017.

[2] R. Parashar and R. Pareek, "Event Triggering Using Hand Gesture Using Open CV," *International Journal Of Engineering And Computer Science*, vol. 5, no. 2, pp. 2278–1024, 2016.

[3] P.-J. Gonzalo and A. H.-T. Juan, "Control of home devices based on hand gestures," *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, 2015.

[4] M. E. Benalcazar, C. Motoche, J. A. Zea, A. G. Jaramillo, C. E. Anchundia, P. Zambrano, M. Segura, F. B. Palacios, and M. Perez, "Real-time hand gesture recognition using the Myo armband and muscle activity detection," *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017.

[5] F. F. M. Maasum, S. Sulaiman, and A. Saparon, "An Overview of Hand Gestures Recognition System Techniques," *IOP Conference Series: Materials Science and Engineering*, vol. 99, p. 012012, 2015.

[6] S. P. More and A. Sattar, "Hand gesture recognition system using image processing," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016

[7] J. Farooq and M. B. Ali, “Real time hand gesture recognition for computer interaction,” *2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE)*, 2014.

[8] R. Shrivastava, “A hidden Markov model based dynamic hand gesture recognition system using OpenCV,” *2013 3rd IEEE International Advance Computing Conference (IACC)*, 2013..

[9] T.-H. Tsai, C.-C. Huang, and K.-L. Zhang, “Embedded virtual mouse system by using hand gesture recognition,” *2015 IEEE International Conference on Consumer Electronics - Taiwan*, 2015.

[10] *Gest*. [Online]. Available: https://gest.co/. [Accessed: 29-Jun-2020].

[11] Ultraleap, “Tracking: Leap Motion Controller,” *Ultraleap*. [Online]. Available: https://www.ultraleap.com/product/leap-motion-controller/. [Accessed: 29-Jun-2020].

[12] “On-Device, Real-Time Hand Tracking with MediaPipe,” *Google AI Blog*, 19-Aug-2019. [Online]. Available: https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html. [Accessed: 29-Jun-2020].

[13] J. Mcintosh, A. Marzo, M. Fraser, and C. Phillips, “EchoFlex,” *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017.

[14] U. C. S. Ltd, “Red Panic Button app – safety app for emergency – SOS call app for iOS and Android,” *Red Panic Button*. [Online]. Available: http://redpanicbutton.com/. [Accessed: 29-Jun-2020].