



Bachelor of Technology (B. Tech.)
Computer and Communication Engineering (CCE)
Amrita School of Engineering
Coimbatore Campus (India)

Academic Year – 2022 - 23

Team 17

S. No	Name	Roll Number
1	Mohan V	CB.EN.U4CCE20033
2	Soorya S	CB.EN.U4CCE20059
3	Taushiq B	CB.EN.U4CCE20064
4	Varun S	CB.EN.U4CCE20069

Fifth Semester
19CCE301 Internet of Things (IoT) Project
IoT-Based ATM threat alerting System Using Raspberry Pi

Faculty In-charge – Peeyush K P Sir

Introduction :

The Idea of Designing and Implementation of ATM threat alerting system project is born with the observation in our real life incidents happening around us. This project deals with prevention of ATM theft from robbery. so overcome the drawback found in existing technology in our society. Whenever robbery occurs, Camera is always in processing will find the user emotion and if its fear ,a message with image, date and time is created. The message is sent using vlc to another module which is not accessible by the thief which ensures security and reliability of data. From that another module, the message is send to the nearby police station and corresponding bank through the GSM.

Abstract :

- The Idea of Designing and Implementation of ATM threat alerting project is born with the observation in our real life incidents happening around us.
- This project deals with prevention of ATM theft from robbery.
- So to overcome the drawback in the existing technology. When ever robbery occurs, the situation is analysed and the emotion of the user and the thief are detected using the camera mounted in the ATM.
- The alert message is sent to nearest police station. Pi-Camera used to capture the situation and send the message within time to the nearby police station and corresponding bank through the GSM. ATM is switched off which will prevent the thief to threaten the user to withdraw money.
- This will prevent the robbery and the person involving in robbery can be easily caught.

Hardware components :

Transmitter side:

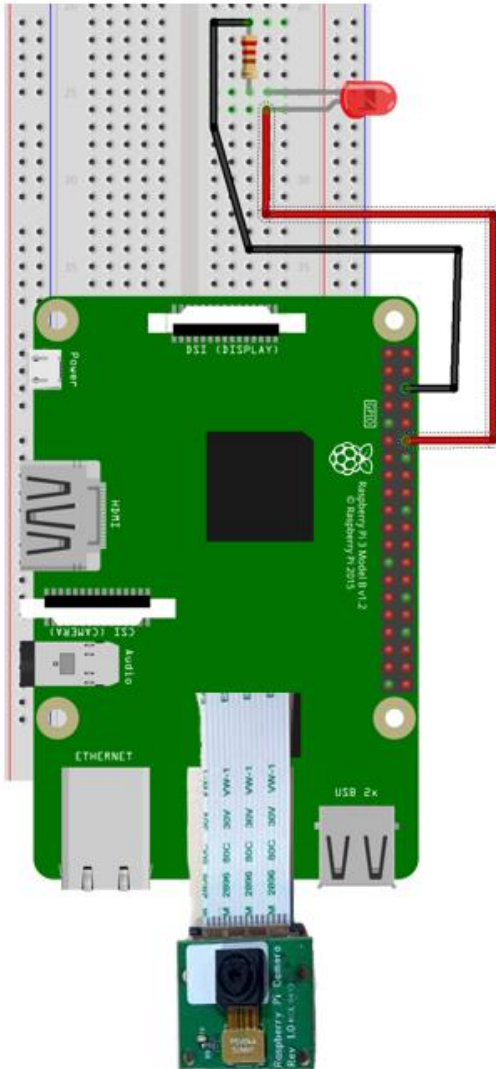
1. LED
2. PiCamera
3. Resistor
4. Raspberry Pi

Receiver side:

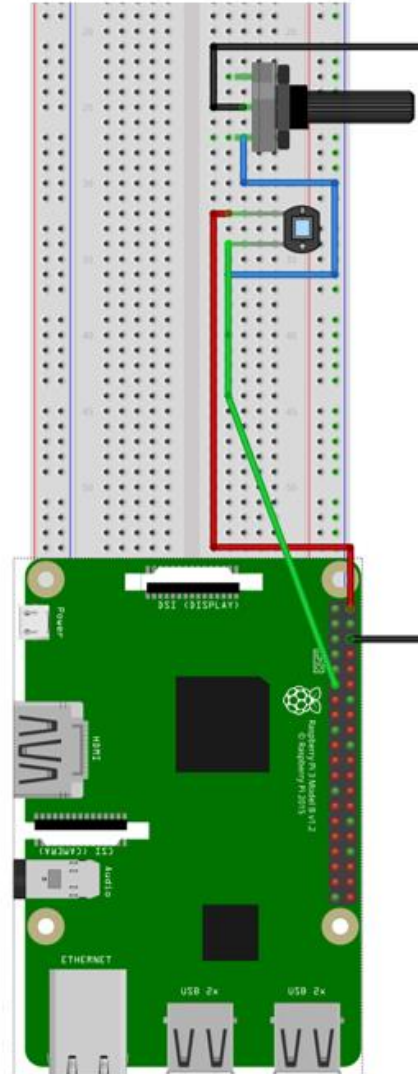
1. Phototransistor
2. Potentiometer
3. Raspberry Pi

Block diagram/ Circuit diagram:

Transmitter side:



Receiver side:



Algorithm /Flow chart :

Transmitter side:

1. Pi-Camera in ATM detects threat
2. Image is captured
3. Alert message and image is encoded as binary code
4. Binary Data sent through LED light

Receiver side:

1. Binary Data received through Photodiode in ceiling

2. Binary code is Decoded
3. Decoded message is sent to police station and bank's server
4. ATM is switched off

Working :

1. OpenCV opens a camera window
2. Using the trained tensor flow-lite model, the emotion "fear" is detected.
3. The image, date and time are encoded into bits and sent to LED light with delay of 0.01 secs
4. The photodiode receives the light from LED
5. The received data is decoded and the message is forwarded to nearest police station and bank's server
6. ATM is switched off which will prevent the thief.

Code :

Final.py:

```
from keras_preprocessing.image import img_to_array
import cv2
from tfliite_runtime.interpreter import Interpreter
import numpy as np
import time

def execfile(filepath, globals=None, locals=None):
    if globals is None:
        globals = {}
    globals.update({
        "__file__": filepath,
        "__name__": "__main__",
    })
    with open(filepath, 'rb') as file:
        exec(compile(file.read(), filepath, 'exec'), globals, locals)

emotion_label_position=(0,0)
face_classifier=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Load the TFLite model and allocate tensors.
emotion_interpreter = Interpreter(model_path="model.tflite")
emotion_interpreter.allocate_tensors()

# Get input and output tensors.
emotion_input_details = emotion_interpreter.get_input_details()
emotion_output_details = emotion_interpreter.get_output_details()
```

```

# Test the model on input data.
emotion_input_shape = emotion_input_details[0]['shape']

class_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']

cap=cv2.VideoCapture(0)
emotion_label=""
while True:
    ret,frame=cap.read()
    labels=[]

    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces=face_classifier.detectMultiScale(gray,1.3,5)
    start = time.time()

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray=gray[y:y+h,x:x+w]
        roi_gray=cv2.resize(roi_gray,(64,64*3),interpolation=cv2.INTER_AREA)

        #Get image ready for prediction
        roi=roi_gray.astype('float')/255.0 #Scale
        roi=img_to_array(roi)
        roi = roi.reshape(1,64, 64,3)

        emotion_interpreter.set_tensor(emotion_input_details[0]['index'], roi)
        emotion_interpreter.invoke()
        emotion_preds = emotion_interpreter.get_tensor(emotion_output_details[0]['index'])

        emotion_label=class_labels[emotion_preds.argmax()] #Find the label
        emotion_label_position=(x,y)
        print(emotion_label)

    cv2.putText(frame,emotion_label,emotion_label_position,cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)

    cv2.imshow('Emotion Detector', frame)
    end=time.time()
    print("Total time=", end-start)

    if cv2.waitKey(1) & 0xFF == ord('q'): #Press q to exit
        break
    if emotion_label=='Fear' or emotion_label=='Sad':
        execfile('vlc.py')
        break

cap.release()
cv2.destroyAllWindows()

```

VLC.py:

```
import time
```

```

import binascii
import RPi.GPIO as GPIO
from time import sleep

#initial setting
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(11, GPIO.IN)

decoded = []
recval = []

def decode(recval):
    no_of_bits = len(recval) / 8
    no_of_bits_int = int(no_of_bits)
    no_of_bits = 8*no_of_bits_int
    recval = recval[0:no_of_bits]
    ul = 8
    ll = 0
    for i in range (0,no_of_bits_int):
        recvaltemp = recval[ll:ul]
        combined = "".join(str(x) for x in recvaltemp)
        combined = int(combined,2)
        converted = str(chr(combined))
        decoded.append(converted)
        ul = ul + 8
        ll = ll + 8
    try:
        decoded.remove("\x00")
    except:
        pass
    return decoded

def encode(text):
    biner = bin(int.from_bytes(text.encode(), 'big'))
    bin2list = biner[2:]
    listbin = []
    for i in bin2list:
        listbin.append(i)
    listbin.insert(0,"0")
    return listbin

text = "hi"#input("Input data to send: ")
print("Data to transmit: ",text)
encoded = encode(text)
print("\n\nTransmitting bits:" ,end=' ')
for i in range(len(encoded)):
    bit = encoded[i]
    print(bit,end=' ')
    if (bit == "0"):
        GPIO.output(12, GPIO.LOW)
    elif (bit == "1"):
        GPIO.output(12, GPIO.HIGH)

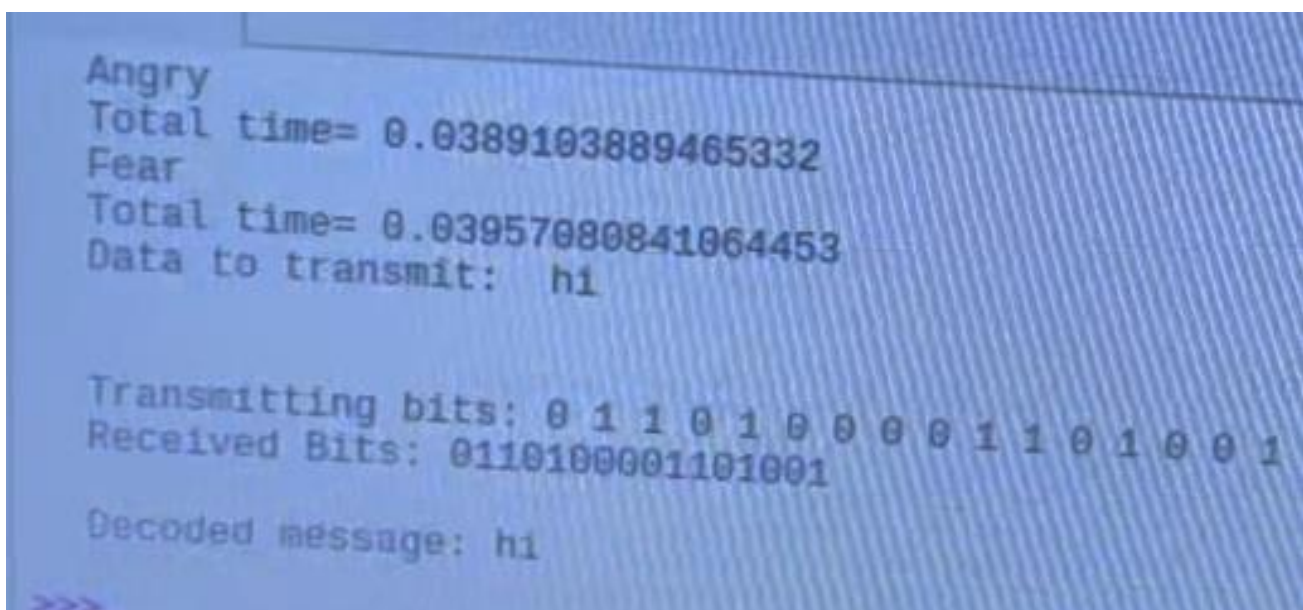
```

```
time.sleep(0.5)
recval.append(int(not GPIO.input( 11 )))
GPIO.output(12, GPIO.LOW) #clearing LED
```

```
recval = "".join(str(x) for x in recval)
print("\nReceived Bits:",recval)
```

```
decoded = decode(recval)
decoded = "".join(str(x) for x in decoded)
print("\nDecoded message:", decoded)
```

Output images:



```
Angry
Total time= 0.0389103889465332
Fear
Total time= 0.03957080841064453
Data to transmit:  hi

Transmitting bits: 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 1
Received Bits: 0110100001101001

Decoded message: hi

>>>
```