END SEMESTER ASSESSMENT (ESA)
B.TECH. (CSE) IV SEMESTER

UE20CS252 - MICROPROCESSOR AND COMPUTER ARCHITECTURE LABORATORY

PROJECT REPORT
ON

SMART AND SECURE HOME

SUBMITTED BY:

Name	SRN
C V Eswar sai Reddy	PES2UG20CS096
D Tirumala Ravi Teja	PES2UG20CS115
Golla Giridhara Maanas	PES2UG20CS124

JANUARY-MAY 2022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
PES UNIVERSITY, ELECTRONIC CITY CAMPUS,
BENGALURU -560100
KARNATAKA, INDIA

TABLE OF CONTENTS

ABSTRACT OF THE PROJECT

DESCRIPTION

This project is a Model of Smart and Secure home with facilities like automatic door with fingerprint recognition, automatic lights with light sensor placed outside the house, automatic fan with temperature sensor placed inside the house and web interface to control all components of the house (light, fan, door, etc) and view the footage of camera present near the door when the doorbell is pressed. This model can be taken as a base and can be used to convert a home into a smart and secure home by making minor changes

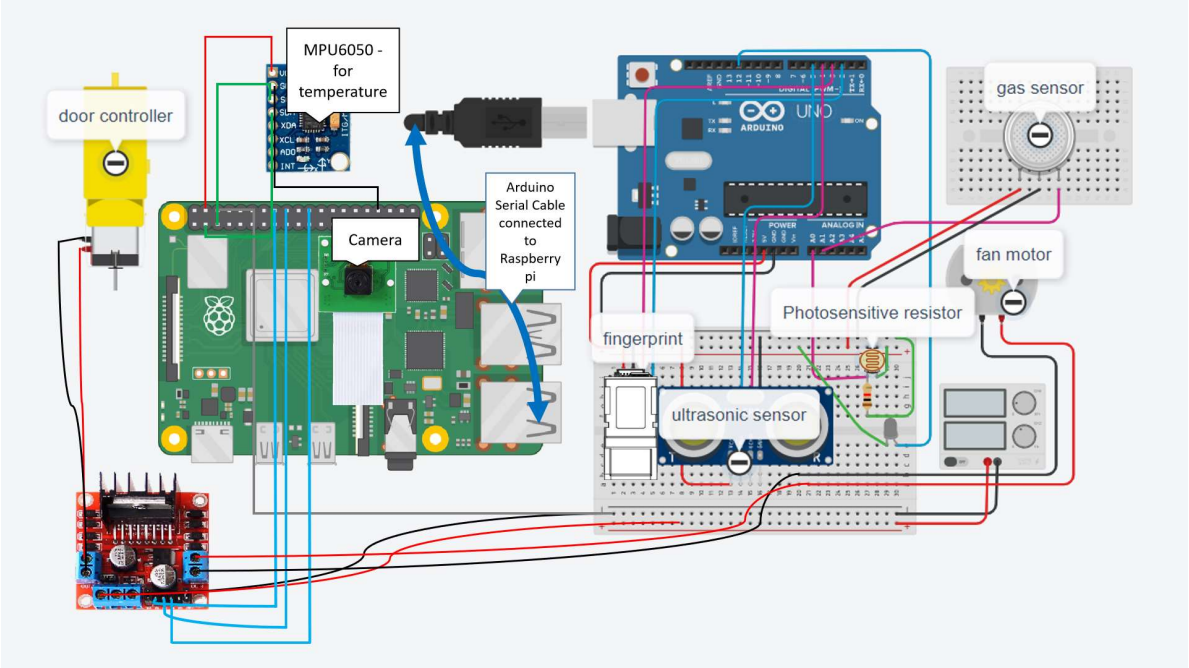
FUNCTIONS OF SENSORS

- The light sensor is used to measure the intensity of light outside the house
- The temperature sensor is used to measure the temperature
- The fingerprint sensor is used to scan fingerprints and provide authentication system
- The camera sensor is used to capture image and display it in the website

GENERAL WORKING

- The Arduino and Raspberry Pi boards communicate with each other using serial interface.
- Arduino controls fingerprint sensor and light sensor
- Raspberry Pi controls temperature sensor and camera
- The light and fan are automated based on light intensity outside the house and temperature. If user sets the mode to **on** or **off** in the website, the model will react to the respective modes.
- The fans and motor which opens the door are controlled by Raspberry pi and the website is hosted by Raspberry Pi
- The light sensor and temperature sensors are placed in the places where the light and temperature are at peak levels to provide the best accuracy in operation of the system.

CIRCUIT DIAGRAM



ARDUINO CODE

```
// Including required libraries
#include <Adafruit_Fingerprint.h>
// BASIC DECLARATIONS
////////////////////////////////////////////////////////////////////////////////////////////////
#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
SoftwareSerial mySerial(2, 3);
#else
#define mySerial Serial1
#endif
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
// BAISC DECLARATIONS ENDS HERE
////////////////////////////////////////////////////////////////////////////////////////////////
// GLOBAL VARIABLES AND OTHER STUFF
////////////////////////////////////////////////////////////////////////////////////////////////

bool checkfingerprint = 1;
bool checkroomlight = 1;
bool roomlighton = 0;
int roomlight = 13;
int gassensor = A0;
String fingerprintstatus = "checking";


// CODE TO CONTROL THE OPERATIONS RELATED TO FINGERPRINTS
uint8_t getFingerprintID() {
  Serial.println("checking for fingerprint");
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }
  // OK success!
  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
```

```
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }
  // OK converted!
  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK) {
    Serial.println("Found a print match!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
  // found a match!
  Serial.print("reportchange fingerprint detected "); Serial.println(finger.fingerID);
  return finger.fingerID;
}
// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;
  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;
  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;
  // found a match!
  Serial.print("fingerprint found "); Serial.println(finger.fingerID);
  return finger.fingerID;
}
// CODE TO ENROLL FINGERPRINT////////////////////////////////////////////////////////////////////////
uint8_t getFingerprintEnroll(uint8_t id) {
  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  delay(200);
  while (p != FINGERPRINT_OK) {
    delay(200);
    p = finger.getImage();
    delay(200);
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }
  // OK success!
  p = finger.image2Tz(1);
```

```
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
      default:
      return p;
  }
  Serial.println("Remove finger");
  delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  Serial.print("ID "); Serial.println(id);
  p = -1;
  Serial.println("Place same finger again");
  while (p != FINGERPRINT_OK) {
    delay(200);
    p = finger.getImage();
    delay(200);
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    default:
      break;
    }
  }
  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    default:
      return p;
  }
  Serial.print("Creating model for #");  Serial.println(id);
  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    Serial.println("Prints matched!");
  }
  else{
    return p;
  }
  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  }
  else{
    return p;
  }
  return true;
}
// CODE TO ENROLL FINGERPRINT ENDS
HERE//////////////////////////////////////////////////////////////
// CODE TO DELETE FINGERPRINT STARTS
HERE//////////////////////////////////////////////////////////
uint8_t deleteFingerprint(uint8_t id) {
```

```
  uint8_t p = -1;
  p = finger.deleteModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Deleted!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not delete in that location");
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
  }
  return p;
}
// CODE TO DELETE FINGERPRINT ENDS
HERE//////////////////////////////////////////////////////////////////
// COODES RELATED TO FINGERPRINT ENDS HERE
//////////////////////////////////////////////////////////////////
int ledpin = 12;
char DIVIDE_CHAR = ' ';
// MAIN SETUP OF THE
CODE/////////////////////////////////////////////////////////////////////////////////////////////
void setup(){
  pinMode(ledpin, OUTPUT);
  pinMode(roomlight, OUTPUT);
  pinMode(gassensor, INPUT);
  Serial.begin(9600);
  // wait till serial connection is established
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  if (checkfingerprint){
  finger.begin(57600);
  delay(5);
  if (finger.verifyPassword()) {
    Serial.println("Fingerprint Sensor Detected");
  } else {
    Serial.println("Fingerprint Sensor Not Detected");
    // make the code idle and stop all actions to ensure that no issues are caused
    while (1) { delay(1); }
  }
  Serial.println(F("Reading sensor parameters"));
  finger.getParameters();
  Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);
  Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);
  Serial.print(F("Capacity: ")); Serial.println(finger.capacity);
  Serial.print(F("Security level: ")); Serial.println(finger.security_level);
  Serial.print(F("Device address: ")); Serial.println(finger.device_addr, HEX);
  Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);
  Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);
  finger.getTemplateCount();
  if (finger.templateCount == 0) {
    Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll'
example.");
  }
  else {
    Serial.println("Waiting for valid finger...");
    Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println("
templates");
  }
  }
```

```
}
// MAIN SETUP ENDS
HERE////////////////////////////////////////////////////////////////////////////////////////
// FUNCTION TO FETCH INSTRUCTIONS FROM THE STRING
typedef struct Instructions{
  String main;
  String sub;
  String str1;
  String str2;
  int int1;
  int int2;
} Instructions;
struct Instructions FetchInstructions(String serialstring){
    int index = 0;
    String temp = "";
    Instructions data;
    int len = serialstring.length();
    // getting main command//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
      temp += serialstring[index];
      index += 1;
    }
    data.main = temp;
    temp = "";
    index += 1;
    // getting sub command//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
      temp += serialstring[index];
      index += 1;
    }
    data.sub = temp;
    temp = "";
    index += 1;
    // getting first string//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
      temp += serialstring[index];
      index += 1;
    }
    data.str1 = temp;
    temp = "";
    index += 1;
    // getting second string//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
      temp += serialstring[index];
      index += 1;
    }
    data.str2 = temp;
    temp = "";
    index += 1;
    // getting first int//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
      temp += serialstring[index];
      index += 1;
    }
    data.int1 = temp.toInt();
    temp = "";
    index += 1;
    // getting second int//////////////////////////////////////
    while (serialstring[index] != DIVIDE_CHAR && index < len){
```

```
      temp += serialstring[index];
      index += 1;
    }
    data.int2 = temp.toInt();
    temp = "";
    index += 1;
    return data;
}
// FUNCTION TO EXTRACT USER INSTRUCTIONS ENDS HERE
// FUNCTION TO CHECK FOR USER INSTRUCTIONS STARTS HERE
void CheckForInstructions(){
  if (Serial.available()){
    String serialstring = Serial.readStringUntil('\n');
    Serial.println(serialstring);
     if (serialstring.length() > 0){
      Instructions instructions = FetchInstructions(serialstring);
      ProcessInstruction(instructions);
      //Serial.println(instructions.main);
      //Serial.println(instructions.sub);
      //Serial.println(instructions.str1);
      //Serial.println(instructions.str2);
      //Serial.println(instructions.int1);
      //Serial.println(instructions.int2);
    }
  }
}
void ProcessInstruction(Instructions data){
  if (data.main == "fingerprint") ManageFingerprints(data);
  else if (data.main == "roomlight") ManageRoomlight(data);
}
int ManageFingerprints(Instructions data){
  if (checkfingerprint == 0) return 0;
  // if user requests to add a fingerprint
  if (data.sub == "add"){
    // getting the id of the fingerprint which should be added
    uint8_t fingerprint_id = data.int1;
    // enrolling the fingerprint by asking the user to place their finger
    getFingerprintEnroll(fingerprint_id);

  }
  // if user requests to delete a specific fingerprint
  else if (data.sub == "delete"){
    int fingerprint_id = data.int1;
    deleteFingerprint(fingerprint_id);

  }
  // if user requests to delete all fingerprints
  else if (data.sub == "empty"){
    finger.emptyDatabase();
  }
}
int ManageRoomlight(Instructions data){
  if (data.sub == "on"){
    checkroomlight = 1;
    roomlighton = 1;
  }
  else if (data.sub == "off"){
    roomlighton = 0;
```

```
    CheckLight();
    roomlighton = 0;
    digitalWrite(roomlight, LOW);
    checkroomlight = 0;
  }
  else if (data.sub == "auto"){
    checkroomlight = 1;
    roomlighton = 0;
  }
}
int gassensor_send(){
  int reading = analogRead(gassensor);
  Serial.print("reportchange gassensor "); Serial.println(reading);
}
int CheckLight(){
  int reading = analogRead(A0);
  if ((checkroomlight && reading < 75) || roomlighton){
    digitalWrite(roomlight, HIGH);
    Serial.println("reportchange roomlight on");
  }

  else {
    digitalWrite(roomlight, LOW);
    Serial.println("reportchange roomlight off");
  }
}
// MAIN LOOP CODE STARTS HERE
void loop(){
  delay(500);
  CheckForInstructions();
  if (checkroomlight) CheckLight();
  if (checkfingerprint) getFingerprintID();
  gassensor_send();
}
```

Raspberry Pi Code

```
import json
import serial
from time import time
from threading import Thread
from multiprocessing import Process
from sys import platform
from gevent.pywsgi import WSGIServer
from flask import Flask, Response, render_template, stream_with_context
from gevent import monkey
import os
import RPi.GPIO as GPIO
from mpu6050 import mpu6050
sensor = mpu6050(0x68)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
from time import sleep
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
OPEN = 16
CLOSE = 18
GPIO.setmode(GPIO.BOARD)
GPIO.cleanup()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(37, GPIO.OUT)
TEMPLATE_DIR = os.path.abspath('../templates')
STATIC_DIR = os.path.abspath('../static')
# creating flask object
app = Flask(__name__)
counter = 0
SEND_DATA = True
def DoorControl(PORT, duration):
    GPIO.setmode(GPIO.BOARD)
    GPIO.output(PORT, GPIO.HIGH)
    sleep(duration)
    GPIO.setmode(GPIO.BOARD)
    GPIO.output(PORT, GPIO.LOW)
# Main Data Structure
Data = {
    "light": {
        "status": "on", "mode": "auto"
    },
    "fan": {
        "status": "on", "mode": "auto"
    },
    "fingerprint": {
        "last_event": None, "mode": "Checkikng for fingerprint"
    },
    "gasreading": 0,
    "temperature": {},
}
def gen():
    if platform != "win32":
        #start = time()
        #os.system("raspistill -o /home/pi/mainfiles/pic.jpg")
```

```
        return (b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + open('/home/pi/mainfiles/pic.jpg', 'rb').read()
+ b'\r\n')
    else:
        ui.screenshot().save("pic.jpg")
        return (b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + open('pic.jpg', 'rb').read() + b'\r\n')
@app.route('/video_feed')
def video_feed():
    print("Video feed route called")
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(), mimetype='multipart/x-mixed-replace; boundary=frame')
# sending main page to user
@app.route("/")
def render_index():
    global SEND_DATA
    SEND_DATA = True
    return render_template("index.html")
# Sending data to user on event trigger
# changing the value of SEND_DATA to True will make the code to send message to
user
# SEND_DATA will be set to False after sending data to user
# the data is used to change the content of page
# Front-end changes wont occour unless there is a command from backend
# This will prevent conflits between the current stage of page and system
@app.route("/listen")
def listen():
    return Response(json.dumps(Data), mimetype='text/event-stream')
@app.route("/setmode/<main>/<sub>/<number>")
def showinfo(main, sub, number):
    print(f"{main} --> {sub} --> {number}")
    ProcessMode(main, sub, number)
    return Response(json.dumps(Data), mimetype='text/event-stream')
def DoorUnlock():
    DoorControl(OPEN, 4.5)
    sleep(5)
    DoorControl(CLOSE, 4.5)
def ProcessMode(main, sub, number = -1):
    global Data
    if main in ["light", "fan"]:
        Data[main]["mode"] = sub
        if sub in ["on", "off"]:
            Data[main]["status"] = sub
        if main == "light" and platform != "win32":
            SerialWrite(f"roomlight {sub}")
    if main == "fingerprint":
        print("Processing fingerprint command")
        if int(number) in range(0, 101):
            print("Sending input to arduino")
            Data["fingerprint"]["last"] = f"{sub} id {number}"
            if platform != "win32":
                print((f"{main} {sub} _ _ {number}"))
                SerialWrite(f"{main} {sub} _ _ {number}")
    SEND_DATA = True
ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
ser.reset_input_buffer()
def UpdateTemp():
    while True:
        try:
```

```
            GPIO.output(37, GPIO.LOW)
            readings = sensor.get_all_data()
            Data["temperature"] = int(readings[2])
        except:
            GPIO.output(37, GPIO.LOW)
            print("Failed to get data from sensor")
            Data["temperature"] = 30
        if ((Data["temperature"] > 25 and Data["fan"]["mode"] == "auto") or
Data["fan"]["mode"] == "on") and Data["fan"]["mode"] != "off":
            if Data["fan"]["mode"] != "off" and Data["fan"]["status"] != "off":
                GPIO.output(37, GPIO.LOW)
                GPIO.output(37, GPIO.HIGH)
                print("\nFan On\n")
                Data["fan"]["status"] = "on"
        else:
            GPIO.output(37, GPIO.LOW)
            print("\nFan On\n")
            Data["fan"]["status"] = "off"
        if Data["fan"]["mode"] == "off":
            print("\nFan Off\n")
            GPIO.output(37, GPIO.LOW)
            Data["fan"]["status"] = "off"
        print(f"Temperature: {Data['temperature']}")
        sleep(5)
Thread(target = UpdateTemp).start()
def DoorControl(PORT, duration):
    GPIO.output(PORT, GPIO.HIGH)
    sleep(duration)
    GPIO.output(PORT, GPIO.LOW)
def SerialRead():
    while True:
        #try:
            line = ser.readline().decode('utf-8').rstrip()
            if line != "" and line != "\n":
                if "reportchange" in line:
                    ProcessChange(line);
                else:
                    print(f"| {line}")
unlocking = False
def ProcessChange(line):
    global unlocking
    if "reportchange fingerprint detected" in line and unlocking == False:
        unlocking = True
        GPIO.setmode(GPIO.BOARD)
        DoorUnlock()
        unlocking = False
    else:
        line = line.split()
        if line[1] == "gassensor":
            print(f"Gassensor {line[2]}")
            gasreading = int(line[2])
            Data["gasreading"] = gasreading;
Process(target = SerialRead).start()
def CaptureFeed():
    while True:
        os.system("raspistill -o /home/pi/mainfiles/pic.jpg")
        print("Stored image")
        sleep(2)
def SerialWrite(string):
```

```
    ser.write((string + "\n").encode("utf-8"))
def SerialWriteUserInput():
    while True:
        ser.write((input("--> ") + "\n").encode("utf-8"))
if __name__ == "__main__":
    app.run(debug = True, port=8000, host='0.0.0.0', threaded = True)
```

Model Structure

Main View

Top View

Front View

Back View

Left View

Right View

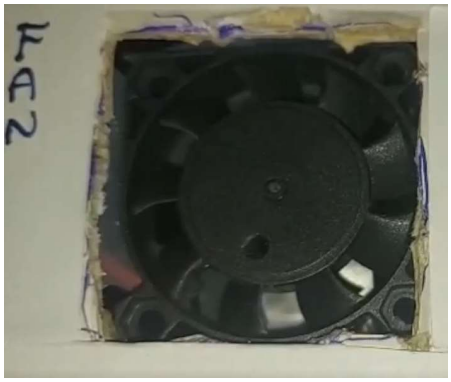
OUTPUTS

LIGHT ON

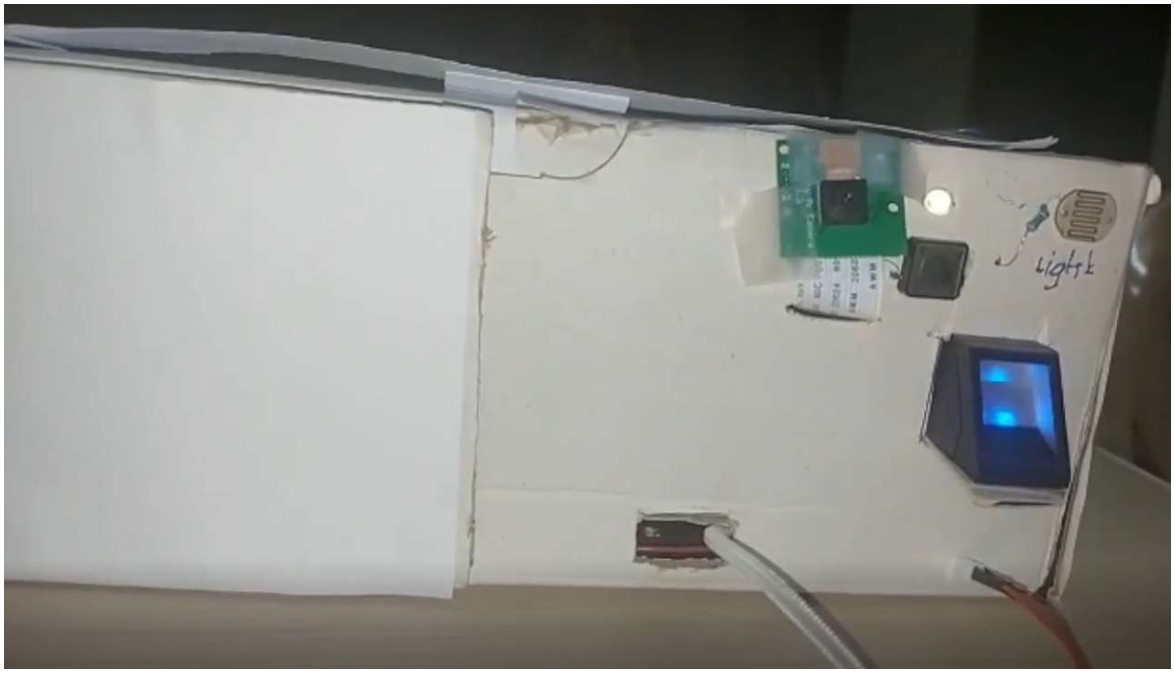
LIGHT OFF

FAN ON

FAN OFF



FINGERPRINT INVALID



FINGERPRINT VALID

References

https://docs.arduino.cc/
https://www.raspberrypi.com/documentation/
https://flask.palletsprojects.com/en/2.1.x/
https://stackoverflow.com/
https://www.w3schools.com/
https://robu.in/wp-content/uploads/2018/05/r307-fingerprint-module-user-manual.pdf
https://projects.raspberrypi.org/en/projects/getting-started-with-picamera

GitHub Link

https://github.com/eswar-cv/python-flask-home-automation