Dissertation on

"Connected Living Spaces"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Chirag N Vijay	**01FB16ECS099**
D G Sudheer	**01FB16ECS101**
Dhanush Ravi	**01FB16ECS112**

Under the guidance of

Internal Guide
Prasad B Honnavalli
Professor,
PES University

Internal Co-Guide
Charanraj B R
Assistant Professor,
PES University

January – May 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Connected Living Spaces'

is a Bonafede work carried out by

Chirag N Vijay	**01FB16ECS099**
D G Sudheer	**01FB16ECS101**
Dhanush Ravi	**01FB16ECS112**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2020 – May. 2020. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature Prasad B Honnavalli Professor	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the project entitled "Connected Living Spaces" has been carried out by us under the guidance of Prof. Prasad B Honnavalli and Prof. Charanraj BR and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2020. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Chirag N Vijay	**01FB16ECS099**
D G Sudheer	**01FB16ECS101**
Dhanush Ravi	**01FB16ECS112**

Acknowledgment

We would like to thank our guide, Professor Prasad B Honnavalli, for his constant guidance and support he has given us all along and for always being there to help us every time we hit a road block. We would also like to thank Professor Charanraj B R, our co-guide who helping us in procuring all the components and assisting us during the development.

We would also like to express our gratitude to Dr Shylaja SS, Chairperson, Department of Computer Science and Engineering, PES University for all the knowledge and support that we have received from the department and we would also like to thank Dr B K Keshavan, Dean of Faculty, PES University for his help. We would also like to thank Professor Ajoy Kumar, COO, PES University for his help.

We are deeply grateful to Dr M R Doreswamy, Chancellor, PES University, Prof Jawahar Doreswamy, Pro-Chancellor, PES University, Dr J Suryaprasad, Vice Chancellor, PES University for providing us various opportunities every step of the way. Finally, we would like to thank our friends and family for their support and encouragement.

Abstract

With this project, we aimed to establish an ecosystem where various devices and appliances in a house communicate with each other. This enables the existing infrastructure like fans and bulbs to be communication ready with upcoming technologies. The project was aimed to build systems than made homes safe, reliable and self-sustained. New processes were set up that increase the efficiency of the day to day tasks. One key goal along with the rest was to reduce the total carbon footprint of the house on the environment. The key functionalities included were to build secure mechanisms and logic for automated door access control, Air Quality Management & Temperature control, and facial recognition. Along with this, a system comprising of various sensors was built which could monitor water level in tanks, control valves and gas leakage which makes a house safe to live in.

Table of Contents

Table of Figures

Chapter 1 Introduction

The Internet of Things is one of the fastest growing domains of the 21th century. The core principle of IoT is to enable devices communicate with each other, use the data collected by each other and build an ecosystem that is autonomous and self-sustained. With the ever-increasing number of devices which are able to communicate with each other, an architecture had to be built and developed to bridge the gap between the existing infrastructure and the upcoming smart infrastructure.

With the Internet of Things, a network of devices (like thermostats, air conditioners, door locks, televisions etc.) can be built. Each and every device collects data points which can be harvested to make a house a better place to live in. As the energy demand is at a steady increase, a lot of emphasis needs to be made on increasing efficiency and reducing the carbon footprint. With IoT, devices can be monitored remotely and its usage can be moderated as and when required.

The project was initially aimed at building and developing an automated door lock system that can be deployed across the classrooms at PES University. Alone the process of development, a set of new modules were developed along with it like temperature monitoring and control, facial recognition, water level management and user mood detection. The places of implementation was widened from classrooms, to labs, warehouses and lobbies.

The five major modules are,

a. Access Control
b. Temperature Monitoring and Control
c. Lighting and Appliance Management
d. Leakage Detection and Alerts
e. Face and Mood Detection

All of the modules are hosted locally on a server located at the PESU Center of Internet of Things. The system comes with a dashboard which the users can access with the help of an IP Address. On loading the dashboard, the users can control the devices, be it locking/unlocking the doors, turning on/off the fans, turning on/off the lights or setting everything on 'Auto' mode, in which case the user can let the system decide on as to what appliance has to run. Each user has a level of authorisation which has a limit and specific device they can control. A pyramid authorisation model was developed where the Department Chairs and Heads of Centers are at the top and have complete authorisation to control everything. At the bottom, were the students, who have the lowest level of authorisation. Students can access the devices specific only to the classroom they have been assigned to.

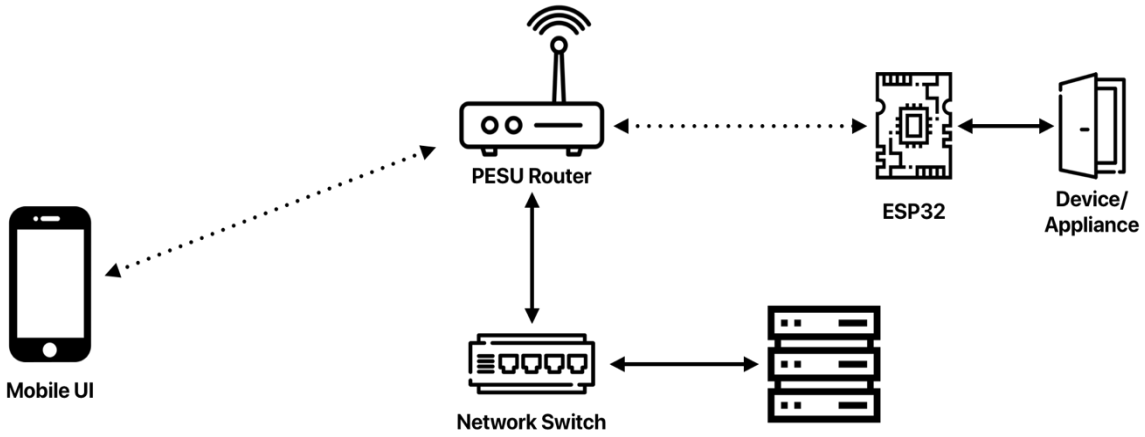


Figure 1: Bluprint of the setup

Node-Red was used as the backend framework, it runs on the central server. Node-Red is a flow based system, where devices can be wired up together for them to communicate with each other locally. Node-Red comes with a GUI where the status of every device can be checked in real time as well as toggle their states with a click of a button.

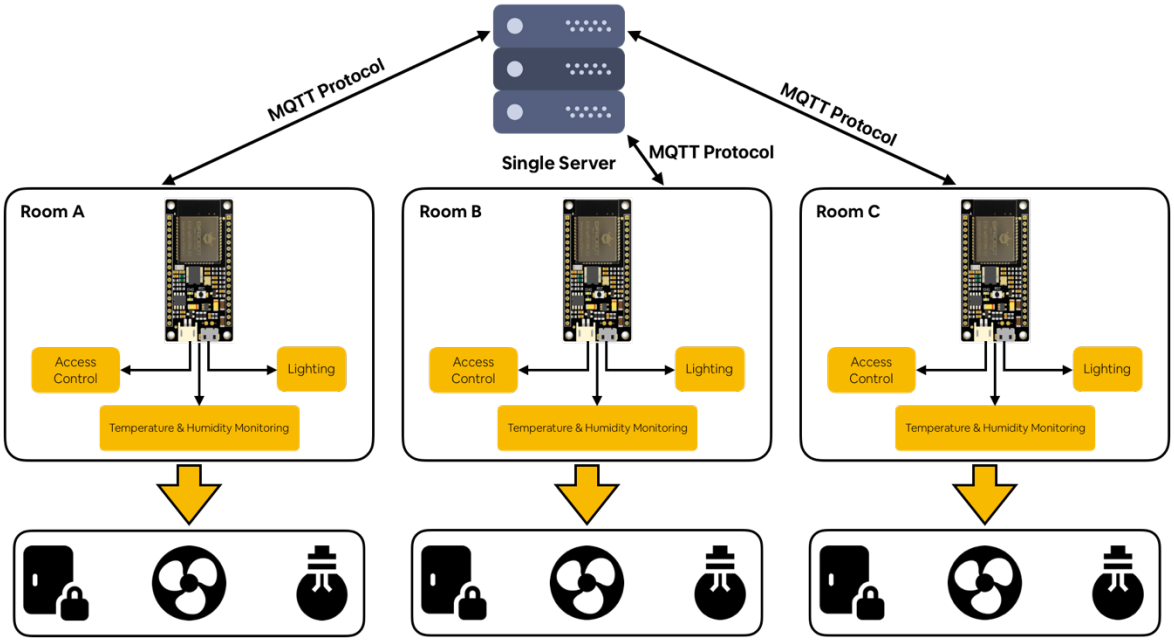


Figure 2: Proposed Architecture

The architecture has been divided into three level,

- The top/first layer is the central server. It has Node-Red running locally and connected to the local network via Gigabit Ethernet. The ethernet connected to the switch which is in turn connected to the building network. One striking advantage is that, this system can be controlled only when connected to the PESU Network.
- The middle layer consists of the ESP-32 microcontroller board which is local to every room. All the devices in a room, be it light, or a fan or a door lock is connected to this. A 5A relay acts like an interface between the ESP-32 and the devices. This is because the operating voltage of the ESP-32 is 3.3V DC and of the devices can be anywhere around 220V. Thus, if directly connected, may lead to the board being burnt. The ESP-32 is a Wi-Fi and Bluetooth enabled board, thus this connects to the PESU Network and communicated with the server.

- At the bottom we have the devices intended to be controlled. Each device has a separate ID unique to the ESP-32 board it's been connected to and each ESP-32 has a unique ID with which it can be identified by the server.

The MQTT protocol is the backbone of the entire project. This underlying protocol is the one the sends and receives messages from one device to another. MQTT stands for message queuing, telemetry and transport. The MQTT protocol runs off the TCP/IP and is lossless and bidirectional.

A voice control interface too was built along with this, to add an extra functionality to the user control. The user can use the 'Google Assistant' application on their phone and control the devices. All they have to say is "Okay Google, turn on my study lamp" and the system does the needful.

Chapter 2 Problem Definition

The project was initially aimed at building and developing an automated door lock system that can be deployed across the classrooms at PES University. Alone the process of development, a set of new modules were developed along with it like temperature monitoring and control, facial recognition, water level management and user mood detection. The places of implementation was widened from classrooms, to labs, warehouses and lobbies.

The five key objectives were,

a. Establish an ecosystem where various devices in a house communicate with each other.
b. Enable existing infrastructure to be communication ready with upcoming technologies.
c. Build safe and reliable homes that are self-sustained.
d. Understand the current underlying infrastructure in Internet of Things.
e. Set up processes that increase the efficiency of the day to day tasks.

Chapter 3 Literature Survey

Of all the papers we referred to for assistance, these are the 7 papers we felt were important and helpful for us. We got to learn and understand on as to how the MQTT protocol works. MQTT protocol is our underlying protocol for on entire project and is its backbone. One more domain we researched into was about how to integrate the sensors and their data so that they can work in unison. The Internet of Things is enabling and extending the advantages of continuous connection of internet to mobile and non-mobile devices. It increases the efficiency of monitoring and gaining data through internet. Home automation and monitoring systems are the interests of many developers. The papers that we have referred to are listed [1], [2], [3], [4], [5]. Most of the systems are limited to only home security and alerting, also paper [2] describes about the new protocol, a variation of existing TCP for home automation and data acquisition.

3.1 Sensor Based Home Automation and Security System

3.1.1 Introduction

In the recent times the security issues are growing very rapidly that the automated home security system is becoming a significant need of the users. Hence it is a prime goal of any home automation system development, to provide the automated security system to protect the users and their properties. There have been many attempts to implement a reliable security system for home but most of those are not really cost-effective and reliable. In this paper the authors have presented a new home security system using field programmable gate array.

3.1.2 Characteristics

Here in this design the developers have provided control and monitoring of security system for home. Through a graphical user interface one can control the security and monitor it. The user can communicate via a web browser. System uses internet to access the service remotely by the user. Also when any security threat arises, the system will send an email

message to the owner quickly. This solution can also be extended to home appliances and devices like fans, air conditioners.

The system which is proposed by the authors of this paper uses a field programmable gate array and the solution was implemented on a NIOS development board cyclone module and interfaces. The NIOS development board also acts as a web server and allows a user to write customized operating system for applications.

3.1.3 Features and Implementation

Remote control of doors and home appliances is the main feature and also it sends messages automatically in case of security breach. Along with the FPGA NIOS board module some sensors like motion sensors will help to achieve this.

The web server is kept at the home and that can be accessed remotely and it is able to handle two-way communication between the user and the sensor. The user interacts with the system using interface through the browser and controls the security and monitors the system over the internet or receives alerts from SMS message.

3.2 MQTT – Messaging Queue Telemetry Transport

3.2.1 Introduction

The Messaging Queue Telemetry Transport protocol was initially developed by IBM and later it was also used by the company Facebook for its messenger application. The need was to have a protocol intended for unreliable network also it must be secured and able to send and receive the data between two entities. All of this is achieved by the MQTT protocol which is a light weight protocol and used for full-duplex communication. Also, MQTT is bandwidth-efficient and since it consumes lesser battery power, energy-efficient as well.

For the modern-day technology, the wireless sensor networks have getting a huge attention because they help in the fields of industrial automation and management, transport business, etc.

3.2.2 Components and Working

This paper proposes the details about publish-subscribe protocol. MQTT protocol is implemented using pub-sub architecture. The architecture consists of one broker (acts as server) and a client. The clients are of two types. They are publishing client (publisher) and subscribe client (subscriber).

Various sensors in the environment acts as the client and connects to server over TCP. The main principle of this protocol is, publisher will connect to a broker and publish the messages and subscriber can subscribe to any number of topics. Here the broker/server plays intermediary role between the publisher and the subscriber. The publisher can issue multiple topics and it passes the message to broker and subscriber selects the topic among those and subscribes to which it needs for the further process.

This protocol is mainly designed for mobile to mobile communications and devices can also be sensors, mobile phones, laptops or embedded systems. It is used for collecting data from many devices and sends those into a data center in a simple manner with security. To ensure the reliability of the data, this protocol supports three levels of quality of service. QoS level zero means, the client sends the data only once and will not check if the data has reached the destination or not. Level one indicates that the data is transported at least once and level two indicates that the message is sent exactly once using handshake method. MQTT uses TCP as underlying protocol hence it is message-oriented. Faster response time, increased throughput, lower usage of power and battery are some other important features of MQTT.

3.3 Microcontroller based Home Security System with Remote Monitoring

3.3.1 Abstract

This paper describes the micro-controller based automated security system for home. The main idea is to protect the house with a password and making a fire alarm system for home. The home security system is a digital system and the user has to enter the password to enter into the house. If any intruder is trying to enter into the house then buzzer is turned on.

3.3.2 Working

The door will have a digital display attached to it and password is entered through this screen. If the user makes three wrong attempts, this will lead to turn on the buzzer indicating the breach of security. Also, this paper describes about temperature controlling module. The system will take the inputs from temperature sensors, fixed inside house and shows the reading on the digital display. ATmega16 AVR microcontroller is used for the purpose of computation and control. In case of very high temperature the microcontroller rises an alarm as indication of fire accident.

3.4 IoT Real time Data Acquisition using MQTT Protocol

Data transport and storage is an important aspect of Internet of Things. Sensors will be sending data in real-time and the data which is coming may not be in the format that is expected. To handle communication of data between the server and the sensor many communication protocols have been developed. Most used protocols for web data exchange are HTTP and MQTT protocol. HTTP is based on request-response architecture and MQTT follows publish-subscribe architecture. In this paper the author has considered MySQL database and MQTT as communication protocol for the study of data acquisition and quality-reliability checking. The paper also compares the performance of HTTP and MQTT protocols by taking data from sensor to server at every 60 seconds. The performance test result shows that the data transferring capacity of MQTT is better than HTTP protocol. Hence for IoT applications, for real-time data acquisition MQTT is well suited protocol.

Chapter 4 Project Requirement Specification

4.1 Class Diagram

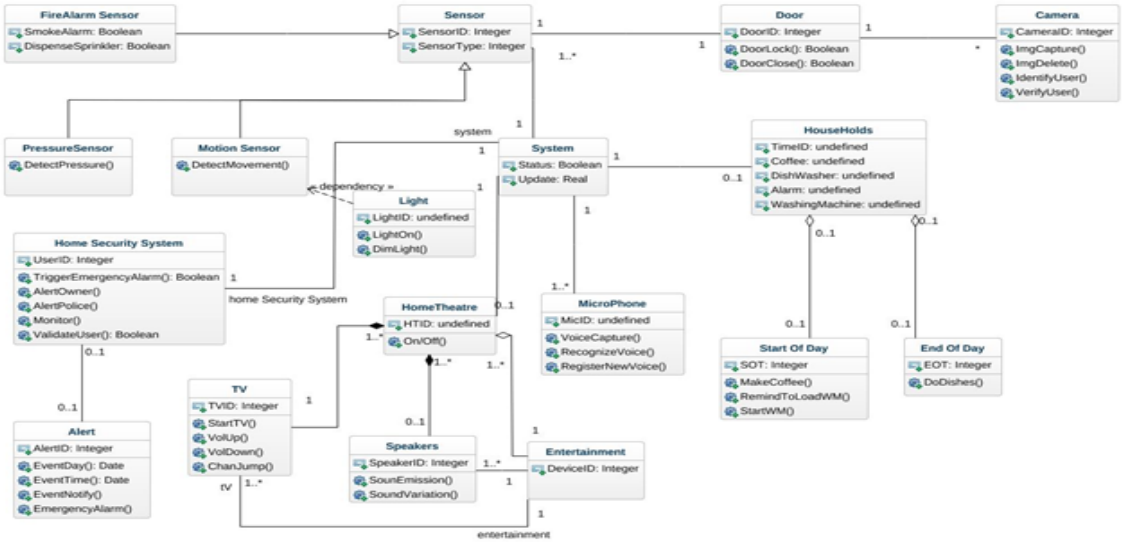


Figure 3: Class Diagram

4.2 Use Case Diagram

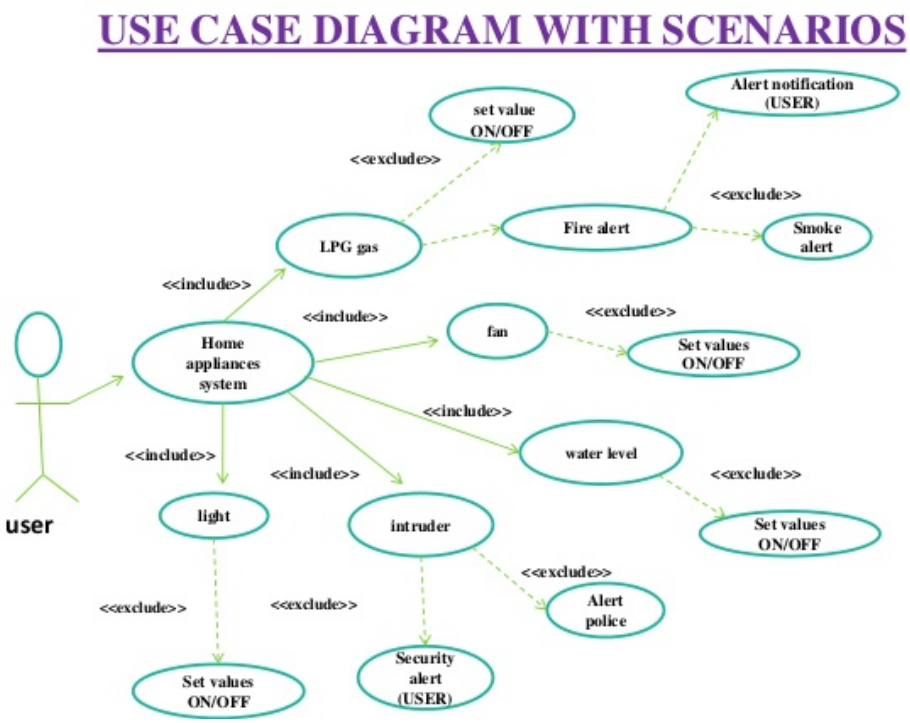


Figure 4: Use Case Diagram

4.3 Modules

4.3.1 Access Control System

This module deals with the automatic access control imparted to the doors/entrances/exits of the facilities. The users are provided with a Web User Interface (compatible with both PC and mobiles) to control their locks. The User interface provides a GUI of all the locks pertaining to their living spaces in the form of switches. Users can also monitor and view the number of locks and unlocks of a particular lock and can also monitor and view the times it was last locked/unlocked. The counters and time are reset every week.

The Web GUI was built using node-red - a JavaScript compatible flow-based software for creating web flows and processes. In this the ESP32 board communicates with the locks and controls the various locks, it communicates with the Node-Red server running on the raspberry pi via an MQTT connection running on the raspberry pi. It then stores important data like the number of locks, unlocks, time the particular lock was locked or unlocked on a local ubuntu server and is sent as a response back to the web graphical user interface. MQTT is a protocol used for communication between the various servers, boards and devices.

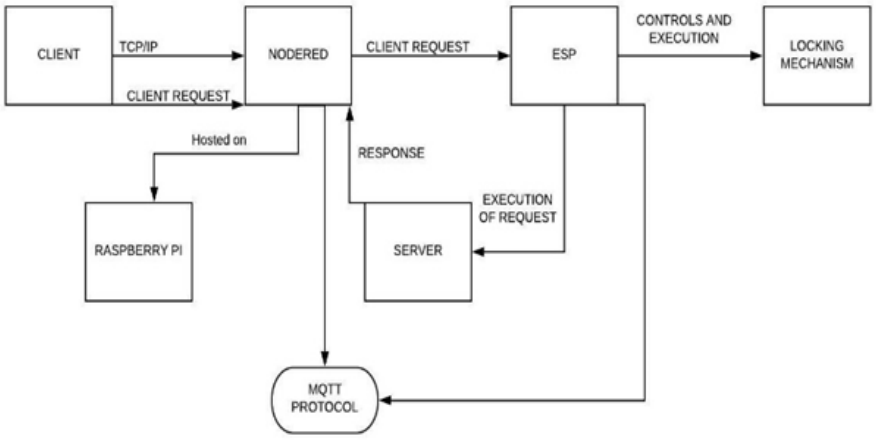


Figure 5: Flow Chart of Access Control Module

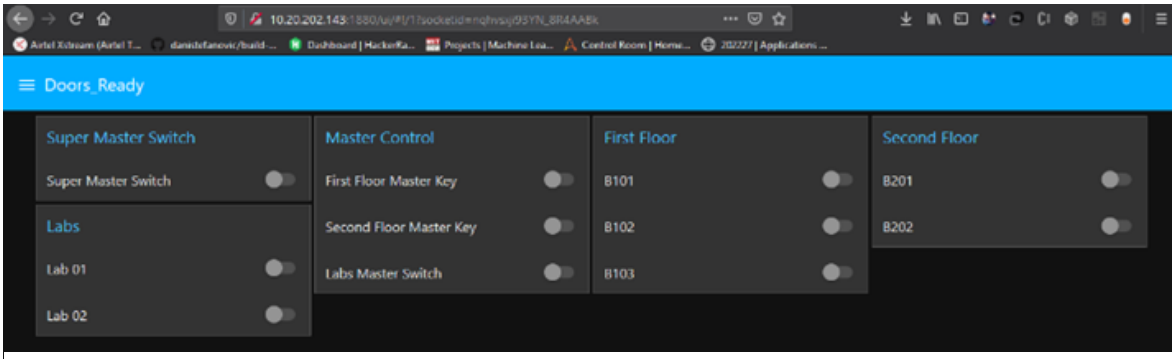


Figure 6: User Interface of Access Control

4.3.2 Temperature Monitoring and Control

This module is about viewing the temperature of each and every zone in the facility and providing means to monitor and control it. Alone the process of development, we developed automatic cooling and heating systems to facilitate the users. Whenever the temperature drops below a certain threshold the heater gets activated and the air conditioner gets activated whenever the current temperature exceeds the upper threshold limit.

Four sensors are located in the four corners of a room and the mean of those gives us the temperature of the room, the temperatures of the four zones and the central temperature is being displayed dynamically on the screen. The data regarding the temperatures, the number of times both the heater and the air conditioner has been used are logged in the database. A failsafe mechanism has been also developed in case any of the sensors fail.

4.3.2.1 Mechanism

The room is shaped to fit that of a square's dimensions and then the room is divided to four zones- Zone A, B, C and D. Then we monitor the temperatures across the zones and calculate the mean temperature and display it on the UI. When the temperature goes above the upper threshold, the cooling system is activated and when it goes below the lower threshold the heating system is activated.

There might arise a scenario where a sensor might go rogue and send in faulty data. Be it because of wear and tear, age or damage, a system is able to identify outlier data and eliminate the use of that sensor.

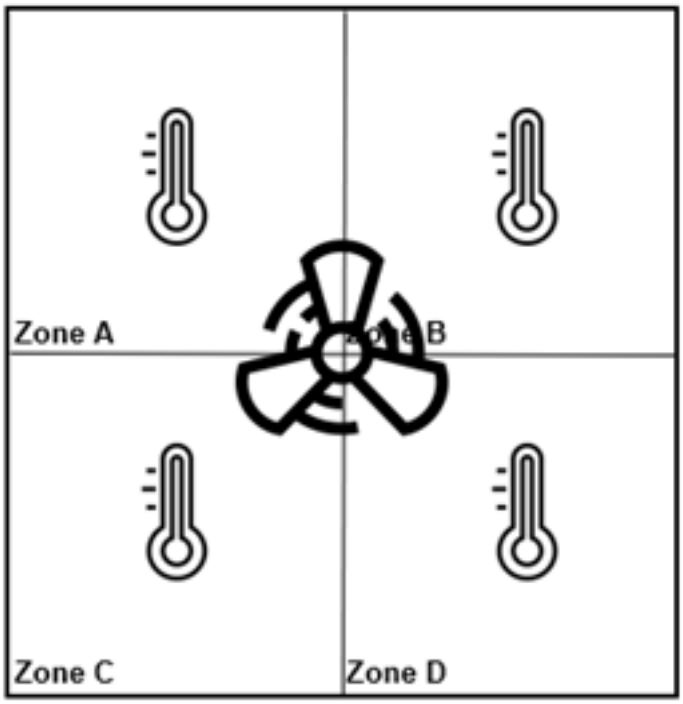


Figure 7: Pictorial Representation of Room Division

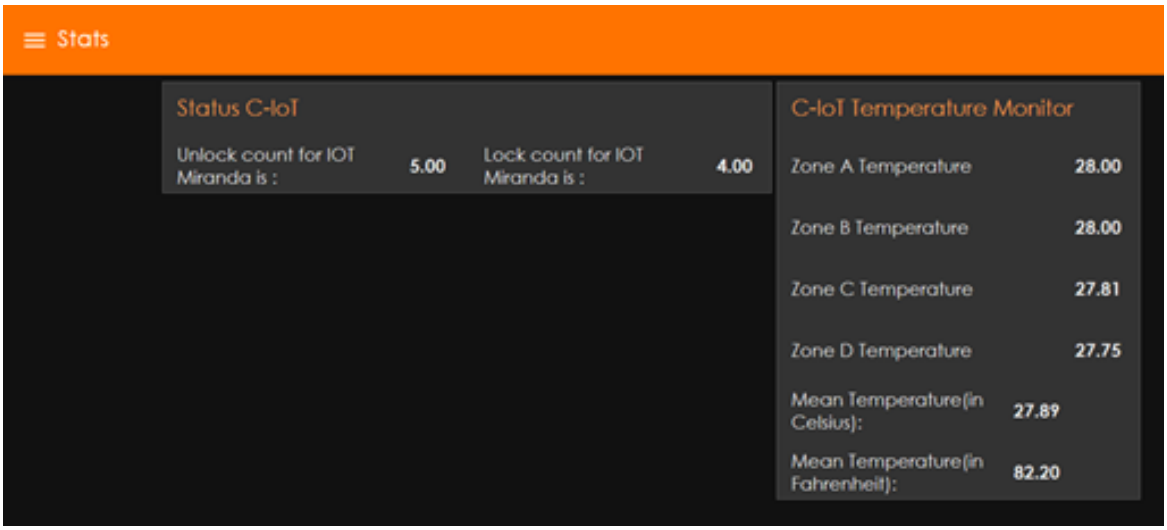


Figure 8: User Interface of Temperature Control

4.3.3 Facial Recognition

Set-up a system where the access control can be controlled using a user's face. This can be used as an alternative to the use of password to unlock doors.

The face recognition module consists of four folders and three python scripts

- Facial_recog
- datasets
- example
- outputs
- Encode_faces.py
- Recognize_face_video.py
- Recognize_face_image.py
- Encodings.pickle

4.3.2.1 Datasets

Suppose there are four authorised users then datasets contain four subfolders with their names as the folder names and each folder has the respective user's pictures. All the images are in standard quality for faster training purposes

4.3.2.2 Encoding

We iterate through every image, we extract the name from the path, convert the OpenCV image to dlib ordering and store the ordering in a variable 'RGB'. Then we detect the x and y coordinates of the defining points of the face in the images and store it in boxes. Then we compute the facial encodings using 'RGB' and 'boxes' and store in encodings .Then we loop through the encodings and add the encoding and name to known names and encoding .Then we create a dictionary containing name and encodings and add the known names and encoding and write it to the encodings.pickle file .

4.3.2.3 Detection Procedure

1.We can either detect it in real time using a video stream or by an image file.

2. Since the implementation is to be in real-time, we stick to the video stream method.

3. We start the video stream first, then convert the OpenCV image to dlib ordering and store the ordering in a variable 'RGB', detect the x and y coordinates of the defining points of the face in the images and store it in 'boxes', we compute the facial encodings using 'RGB' and 'boxes' and store in encodings.

4. We then loop over the encoding, compare it with the encodings from the pickle file.

5. If a match is found, we find the indexes of all the matched faces, calculate the total number of times each face was matched and we select the name with the maximum number of votes

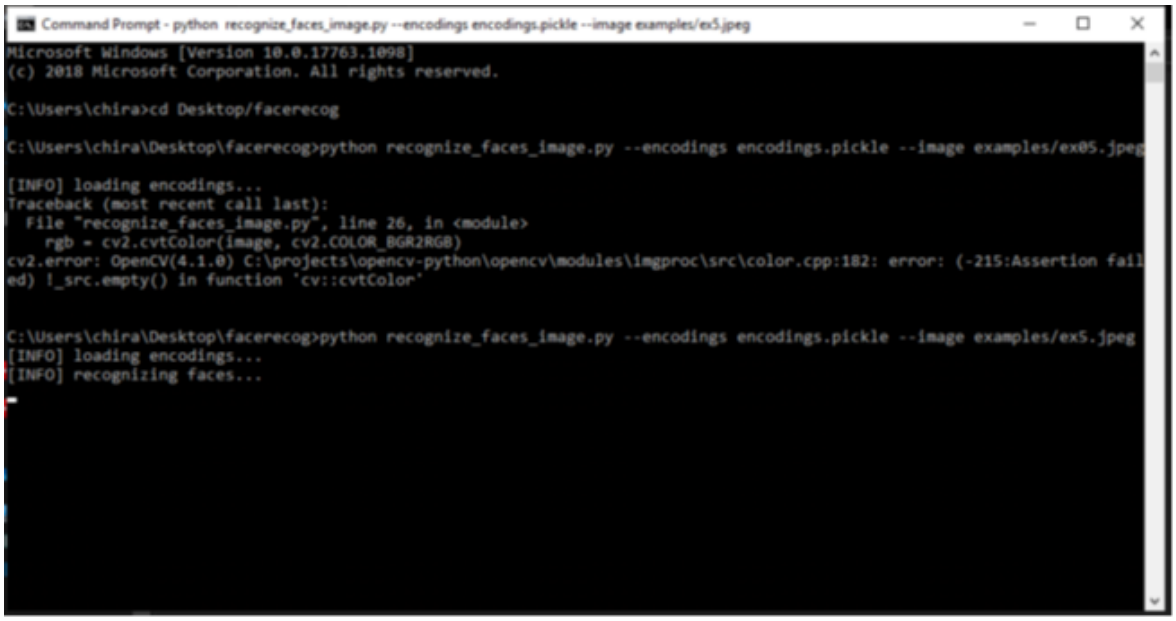


Figure 9: Terminal during Face Recognition

and append it to names.

6. Then we loop over the recognized faces, we rescale the face co-ordinates, construct a rectangle around it and label it with the name of the user from the list 'names'.

7. Then after the user presses q we exit the process.

8. Suppose there is no match, then the name is displayed as 'Unknown'.

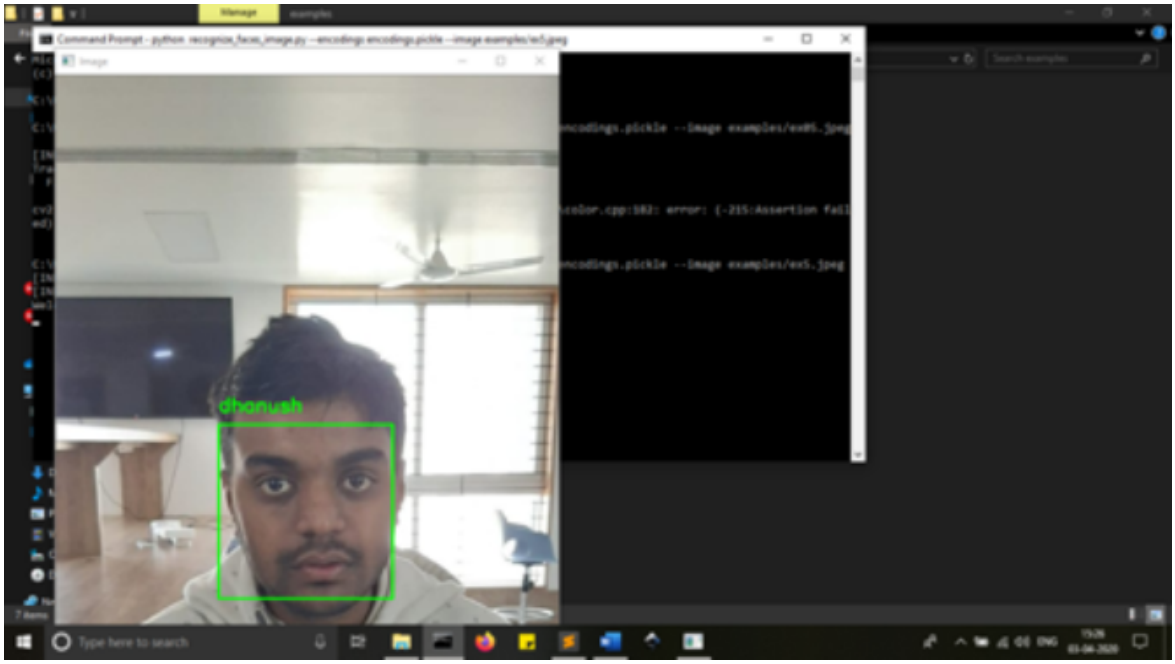


Figure 10: Graphical Output of Face Recognition

```
Command Prompt - python  recognize_faces_image.py --encodings encodings.pickle --image
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\chira>cd Desktop/facerecog

C:\Users\chira\Desktop\facerecog>python recognize_faces_image.py -

[INFO] loading encodings...
Traceback (most recent call last):
  File "recognize_faces_image.py", line 26, in <module>
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2.error: OpenCV(4.1.0) C:\projects\opencv-python\opencv\modules\
ed) !_src.empty() in function 'cv::cvtColor'


C:\Users\chira\Desktop\facerecog>python recognize_faces_image.py -
[INFO] loading encodings...
[INFO] recognizing faces...
Welcome to your living spacedhanush
```

Figure 11: Terminal Output of Face Recognition

4.3.4 Mood Detection

The user listens to their choice of music on “Spotify” Music App. The system built monitors the songs being listened to and analyses the mood of the user. Based on the mood, further recommendations are made.

4.3.4.1 Authentication

In this the user has to provide their client identification number and client secret from his Spotify developer dashboard.
They also have to give his username for the same.
We are creating a token from the username, scope, client_id and client_secret provided. The “authenticate_spotify” function does the authentication and it creates a Spotify instance for future tasks and development

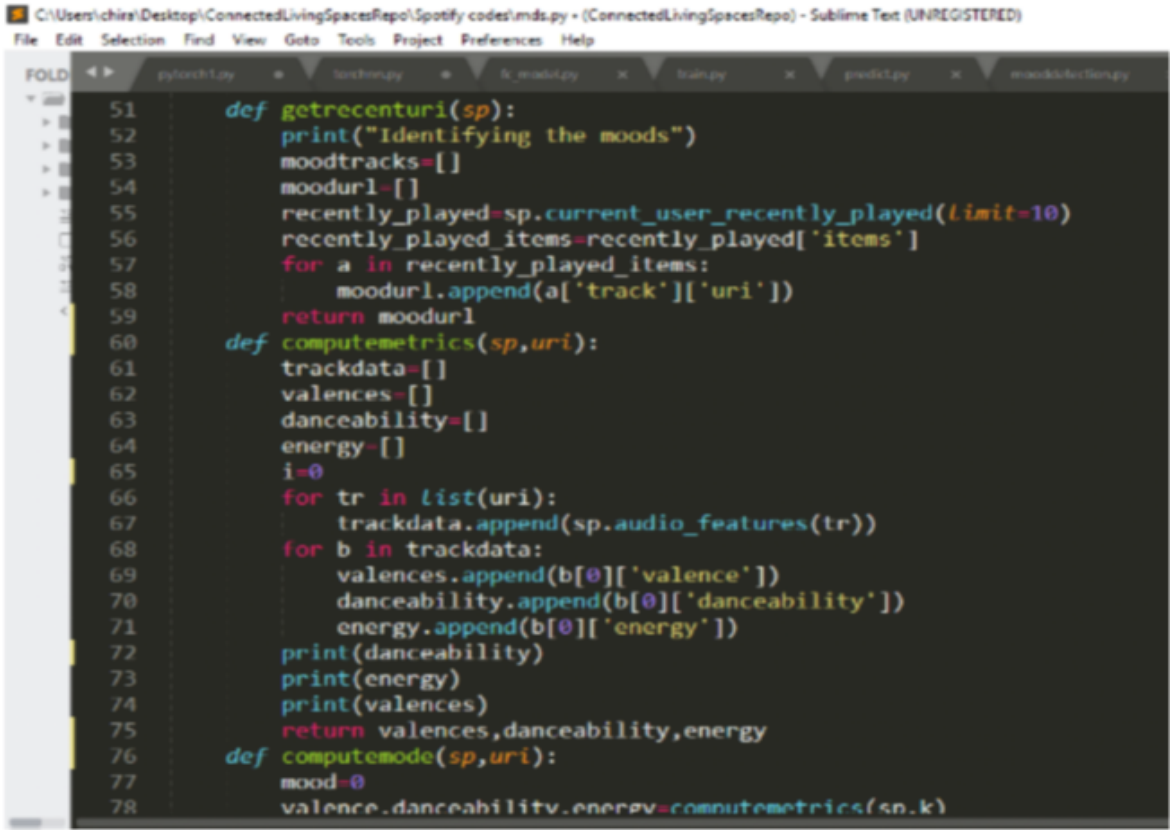
Code:

```
token=util.prompt_for_user_token(username,scope ,client_id=client_id,client_secret=clien
t_secre t,redirect_uri=redirect_uri)
 if token:
      def authenticate_spotify():
                    print("..Connecting to spotify")
                   sp=spotipy.Spotify(auth=token)
                   return sp
```

4.3.4.2 Detection

In this we gather data regarding the last 10 songs the user has listened to. We then gather the audio features of every song and we predict the mood value based on the values of the audio features.
In getrecenturi function we are appending the URI’s of all last 10 listened songs to a list.

In compute metrics function we are calculating valence, danceability and energy values of the songs in the URI list and appending it to 3 different lists.



```
def getrecenturi(sp):
    print("Identifying the moods")
    moodtracks=[]
    moodurl=[]
    recently_played=sp.current_user_recently_played(limit=10)
    recently_played_items=recently_played['items']
    for a in recently_played_items:
        moodurl.append(a['track']['uri'])
    return moodurl
def computemetrics(sp,uri):
    trackdata=[]
    valences=[]
    danceability=[]
    energy=[]
    i=0
    for tr in list(uri):
        trackdata.append(sp.audio_features(tr))
    for b in trackdata:
        valences.append(b[0]['valence'])
        danceability.append(b[0]['danceability'])
        energy.append(b[0]['energy'])
    print(danceability)
    print(energy)
    print(valences)
    return valences,danceability,energy
def computemode(sp,uri):
    mood=0
```

Figure 12: Code for getting URI

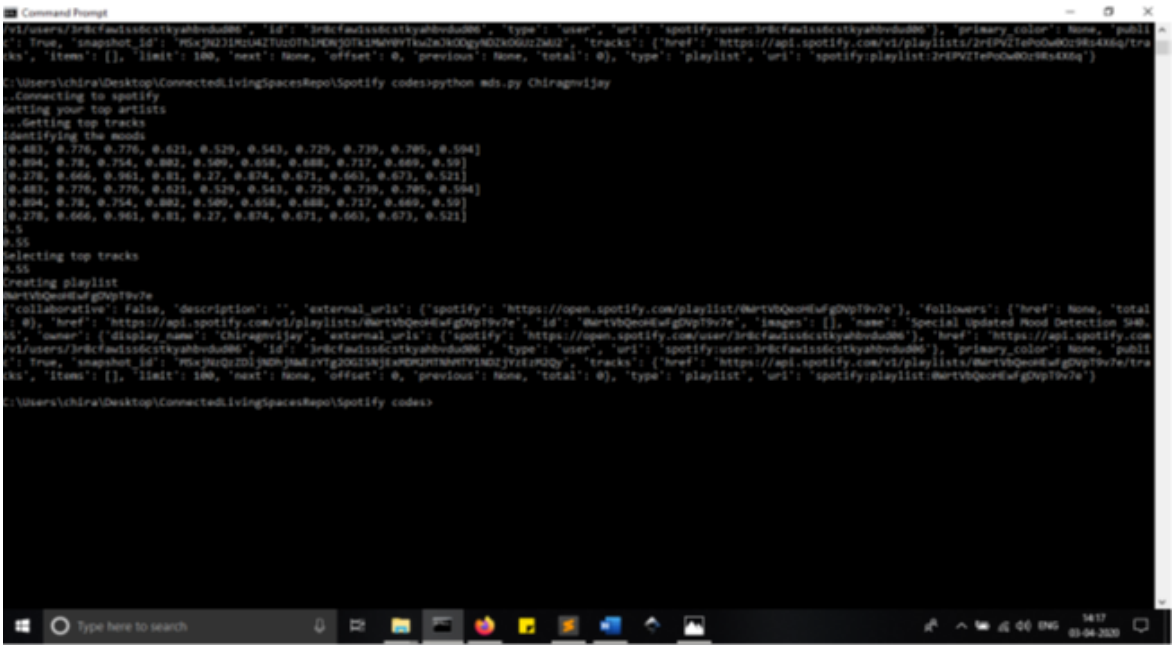


Figure 13: Terminal which listening to music

4.3.4.2 Processing

Aggregate_top_artists function prepares a list of the URI's of the artists the user frequently listens too or follows. aggregate_top_tracks consists of a list of the songs belonging to the artists from aggregate_top_artists.

In this the output from the compute mode function is passed as a parameter to the select tracks function.

In this a list of tracks by the artists the users follow is prepared and then the audio features of those songs are compared with the computed mood values and only the ones which match are appended to a list called selected_tracks.

```
def aggregate_top_artists(sp):
    print("Getting your top artists")
    name=[]
    uri=[]
    ranges=['short_term','medium_term','long_term']
    for r in ranges:
        artists_all_data=sp.current_user_top_artists(limit=50,time_range=r)
        artists_data=artists_all_data['items']
        for artist_data in artists_data:
            if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    followed_artists_all_data=sp.current_user_followed_artists(limit=50)
    followed_artists_data=followed_artists_all_data['artists']
    for artist_data in followed_artists_data["items"]:
        if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    return uri
def aggregate_top_tracks(sp,uri):
    print("...Getting top tracks")
    tracks=[]
    for artist in uri:
        top_tracks_all_data=sp.artist_top_tracks(artist)
        top_tracks_data=top_tracks_all_data['tracks']
        for track_data in top_tracks_data:
            tracks.append(track_data['uri'])
    return tracks
```

Figure 14: Code for Artist ID

4.3.4.3 Delivery

We then create a playlist and then add 30 songs from the selected_tracks into the playlist. The user can then access it from the library option from their spotify accounts.

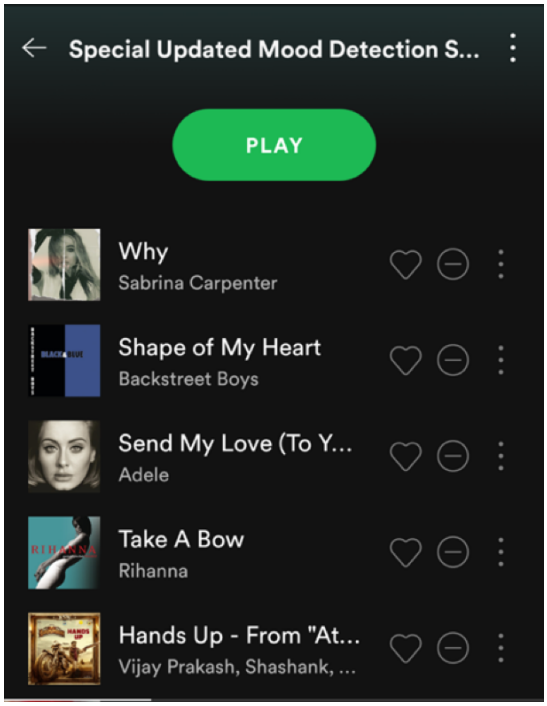


Figure 15: Final Output on Spotify Application

4.3.5 Water Leakage Detection

A water leakage and monitoring system is developed to monitor the water level and to prevent water leaks. We have two water sensor w1 and w2 located at 30% of the height of the tank and at 90% of the height tank respectively. We monitor the readings and when the value of w1 is less than the threshold then the pump is turned on automatically and when the value of w2 is more than the threshold the pump is turned off. The water level and the number of times the pump has been turned on/off has been recorded and shown the dashboard and backed up in a persistent database.

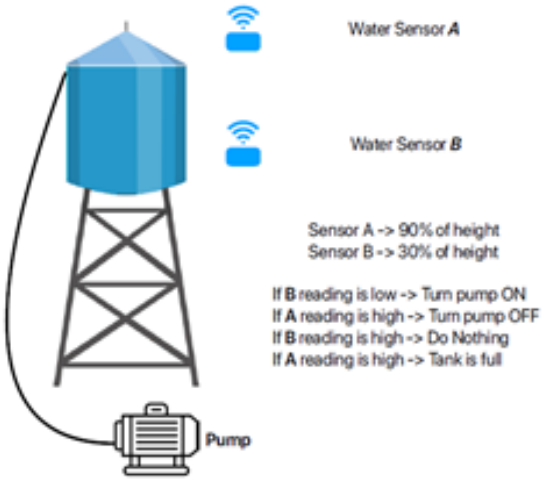


Figure 16: Blueprint of the Water Level Management

4.3.6 Gas Leakage Detection

A gas leakage detection and mitigation system has been developed which uses MQ2 and MQ5 sensors for detecting the gas leaks.MQ2 sensors detect smoke and combustible gas while MQ5 detects LPG and natural gas. There are a number of sensors in potential places e.g.) kitchen. The system identifies the origin of the leak and then closes the valves. It also communicates with the cooling system to turn on the nearby fans and also communicates with the access control system to contain the leak. The number of leaks is displayed on the dashboard and backed up in a persistent database.

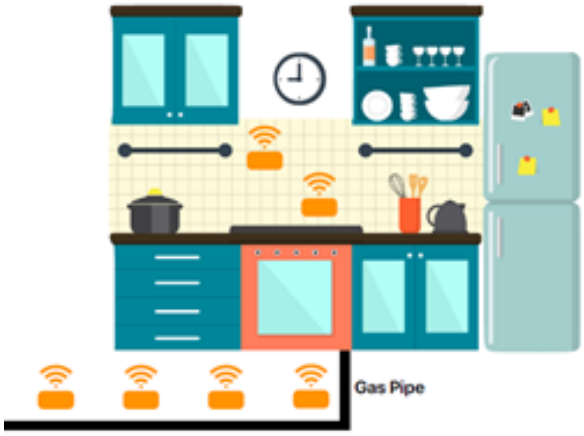


Figure 17: Blueprint of Gas Leakage and Management

4.4 Constraints

1. The project works well with Windows and Linux based Operating systems and is platform independent. Works both with computers and mobile devices
2. The server system requires python 3.5+ preinstalled and various packages such as dlib, face_recognition, Spotify, argparse.
3. The server also needs to have NodeJS and node red installed
4. Uninterrupted power supply is required.

4.5 User Classification

1. The project helps a wide set of users, it is mainly designed for the educational and the corporate sector. The classification of the users is a hierarchical structure.
2. A pyramid authorisation model was developed where the Department Chairs and Heads of Centers are at the top and have complete authorisation to control everything. At the bottom, were the students, who have the lowest level of

authorisation. Students can access the devices specific only to the classroom they have been assigned to.

Figure 18: User Classification Diagram

Chapter 5 System Requirement Specification

5.1 Functional Requirements

5.1.1 Access Control

The Access Control Module has been designed to control the entry/exit of personnel to and from a facility. It involves identifies the individual who is seeking to enter, also with their level of authorization. The time of access too needs to be accounted for, which is in turn stored in the database. In accordance with the classroom scenario, the system can function in either of the two modes, a) open at a designated time, stay open during the course of the day and close at a designated time or b) open to only specific students/faculty who have been authorized that room. The module connects through Wi-Fi to a centralized server.

5.1.2 Temperature Control

This module has been designed to monitor and track the temperature conditions of a facility. At the same time, based on the collected temperature data, the system can decide it turn on/off the air conditioning system. The user has the ability to manually overwrite the same and set the temperature if required. The room will be divided into zones and each zone will have temperature recording apparatus which will write data to the central database. There

might arise situations where the apparatus might malfunction and send in wrong data to the database which might in turn lead to the wrong operation of the air conditioning system. A subsystem has been developed to find the faulty sensors and eliminate their influence on the decision making.

5.1.3 Facial Recognition

The face recognition module was developed to reduce the complexity to the Access Control Module. The face/facial details of the authorized users can be registered on the database/face recognition repository. This can be used instead of the Access Control User-Interface or along with it for added security to the sensitive facilities. The face recognition code base has been hosted on a separate server, as it requires a lot of compute and this server will be networked with the main IoT control server.

5.1.4 Mood Detection

The mood detection module works in accordance with Spotify, the music mobile application. Based on the music the user listens to, a mood number can be obtained which enables the system to understand and set the house conditions which helps the user to ease out. It includes setting the air condition temperature or the light brightness and color.

5.1.5 Lighting Control

The light fixtures in a facility can be monitored and controlled using this module. Again, for this, there are multiple apparatus setup across the facility that monitors the conditions and logs the data to the database. This is useful and vital for situations when a user might forget to turn off a light when not using it. There exists a lighting index which is like a benchmark, below which the lights turn off or on to provide the optimal environment to work at.

5.2 Non-Functional Requirements

- The user interface should work either on desktop web browser and the mobile web browser. The UI should optimize itself to the particular screen its being viewed on without any user intervention.
- Dark Mode and Light Mode themes has been made available for the user preference.

- The user has been given the preference to choose the color scheme for the user interface. A hex color code can be given based on the user preference.
- The user has been given the ability to choose the font face/style and the font size.

5.2.1 Dependencies

- Node-Red
- Server with good ram and processor running Ubuntu
- Access to Intranet

5.2.2 Assumptions

- Uninterrupted power supply is present as the entire module depends on it
- Intranet is up and running continuously

5.3 Hardware Requirements

Each of the micro sub modules runs on an ESP-32 microcontroller board. The ESP-32 has Bluetooth and Wi-fi in built. The Wi-fi ability of the ESP 32 has been harnessed to connect to the routers which in turn means, can communicate with the centralized server. In the case of Access Control, an electromagnet has been fixed on the door which fixates onto a metal strip which is on the door frame when powered on. A 12V/5V SMPS power supply powers the entire module. TH SMPS has been used because of the refined and steady output which leads to longer life of the hardware. A 5A relay has been used to interface the 12V electromagnet and the 3.3V ESP-32. If the relay is not used, the 12V will fry the ESP-32. For the temperature sensing and control module, a DS18B20 module is used. A Dell PC with an i7 processor, 16GB RAM and 4GB graphics card has been used as the server.

Chapter 6 System Design

The chosen system is a hybrid system consisting of computers, microcontrollers and various hardware devices. It consists of a central PC acting as a server which is connected to the Wi-Fi routers. All the microcontroller boards are connected to the system through the local Wi-Fi and the hardware devices such as lights and fans are connected to various ports on the ESP32 microcontroller board. Various sensors and servos and regulators are also connected to the microcontroller boards.

6.1 System Architecture

The various modules and devices communicate through the MQTT protocol over Wi-Fi. MQTT is a messaging protocol designed for communication in the Internet of Things domain.

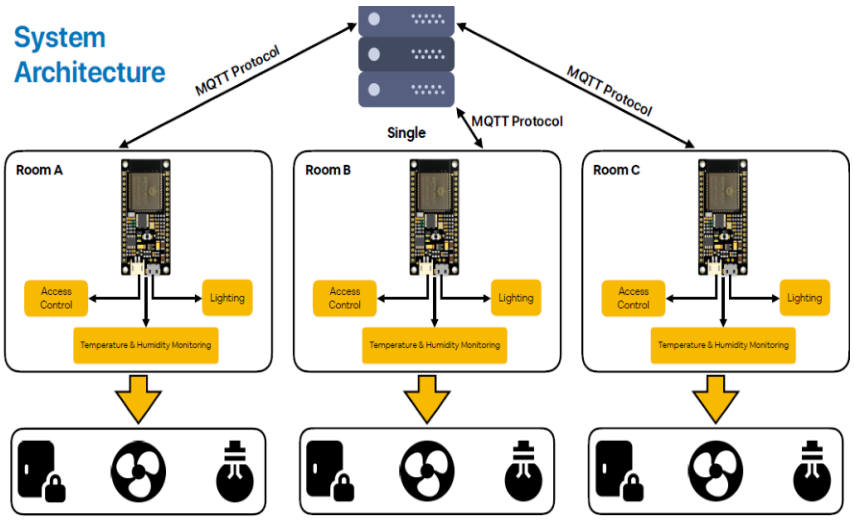


Figure 19: Implemented System Architecture

The primary programming languages being used are C++ and Python. Spotify integration is facilitated and Alexa support is provided.

6.2 Why This System

The system was carefully chosen keeping all the constraints in mind. The system is platform independent. The GUI designed is accessible once the user connects to the Wi-Fi. The user can access it from their laptops, computers, mobile phones, tablets, it is truly platform independent. We chose the ESP32 as the primary microcontroller because it has its

own memory, processor and an inbuilt Wi-Fi module. It contains many digital ports through which we can connect the various devices. Its supportability with the MQTT Protocol is another reason the ESP was chosen as the primary microcontroller board. The PC which acts as a server also is a DELL PC with 16GB RAM and 4GB Graphics. We used it as we need to run facial recognition, mood detection and the access control server parallelly. All the data regarding the various modules is stored in a persistent database on the PC. MQTT protocol was chosen for making the devices communicate with each other and thus make existent infrastructure smart.

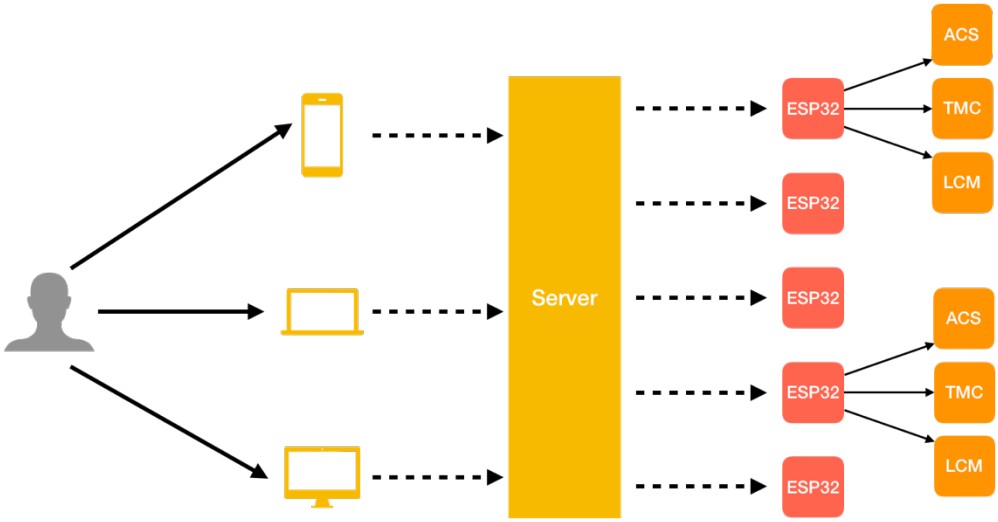


Figure 20: Schmatic of User Interaction

The previous diagram shows the pictorial representation of the architecture. The user can access the UI either from a mobile device, tablet or a desktop. The solid lines represent the wired connection and the dashed lines represent the wireless connection. ACS stands for Access Control System, TMC stands for temperature monitoring and control and LCM stands from the lighting control management.

Chapter 7 Detailed Design

7.1 Access Control Module

The components used in this module are a central server, esp32 board, magnetic lock. The electromagnetic lock is attached to the door and it is connected to the esp32 board which is connected to a power supply. The ESP32 connects to the server which is being hosted on an Ubuntu DELL PC through its inbuilt Wi-Fi module.

Configurations of the PC are,

- Operating System - Ubuntu 18.04
- RAM - 16GB
- Graphics - 4GB
- ROM - 1TB.

The above-mentioned system was chosen as the central server because it has large memory, high processing power and fast processing rate. An Ubuntu PC was chosen as it is open sourced and most of the software that is developed works best with Linux based systems.

ESP32 is a low-cost portable microcontroller with inbuilt Wi-Fi and Bluetooth modules. It is an upgraded version of the widely known ESP8266 boards. It is used as the primary computing device which controls the working of the magnetic locks. ESP32 is chosen as it is compatible with the MQTT Protocol and has its own Wi-Fi module and has a well-furnished software development library. It has 36 digital ports to which the locks, the buzzers and the LEDs are connected.

Electromagnetic locks are used and they are attached laterally to the door and near the hinge. The lock basically is of 2 parts one is the strong electromagnetic part and the other is the iron slab which gets attached to the magnet when turned on hence locking the door. The

terminals from these are then connected to the digital pin of the ESP32 board. These locks were chosen as they are very strong, reliable, can be controlled by electricity and tough to tamper with.

The ESP32 first connects to the server using the Wi-Fi module and then communicates with the server through the MQTT Protocol. MQTT stands for Message Queuing Telemetry Transport Protocol which facilitates the communication between devices. The ESP connects to the broker client installed on the server and communicates with the server and the other connected devices. MQTT was chosen as it is compatible with microcontrollers and has well defined libraries to develop and modify the connections and the communication infrastructure. MQTT also doesn't need internet it just requires an active Wi-Fi connection. Packets are rarely lost and it doesn't require large bandwidths.

The GUI is developed using Node Red. Node Red is an opensource flow-based development tool developed by IBM for visual programming. It uses JavaScript as its backend language and the whole user interface is designed using node red. Node red is used as it has a variety of nodes from text nodes to function nodes and also provides support for MQTT integration and Alexa support.

7.2 Temperature Management Module

This module consists of the central server, temperature sensors, fans and regulators.
At first the DHT22 temperature sensors were used but then they were removed as they are prone to damage and are provide erroneous readings. The DS18B20 sensors were then used as they are widely used in the industry, are of good quality and provide accurate readings. It is in shape of a wire with a steel extension so they are less prone to damage and can be hung or placed at convenient places effortlessly. Their most unique property is any number of sensors can be connected and can be connected to a single wire which can be connected to the ESP32 board. Hence any number of sensors can be connected through a single wire. It is referred to as the one wire protocol. It requires minimum calibration time and are accurate all the time.

Then wires from the switches of the actual AC fans are extended and connected to the L298N motor driver which is a low cost effective driver which acts as a regulator and controls the speed of the fans.L298N is a dual H-Bridge motor driver which controls the speed and directions of the motor at the same time. It is efficient, works on low power and durable. Based on the temperature readings the ESP sets the speed and then turns on the fans in that particular area.

7.3 Facial Recognition System

At first the esp32 cam module was used for facial recognition but then proved to be not so reliable and had a significant error rate. Then the raspberry pi cams were used for the process but the computation of that level wasn't possible on the pi. The pi couldn't support the heavy computation. So, then a simple webcam was used for the process and with help of some functions from face_recognition package which is an opensource python package for facial recognition the system was developed, implemented and tested.

The computation is done on the central server as the server was perfectly capable of handling load and computation of that degree. Firstly, the facial features of the authorized people are encoded and recorded, then the facial features of the test subject are compared to find a match and then decisions are taken accordingly. First the recognition was done using an image file but now for practical implementation there was a transition from image-based detection to video stream based facial detection.

7.4 Mood Detection

In this module there's spotify integration with Python3 as the primary development language. Spotify was chosen as it is one of the most widely known and popular music streaming applications, it can be accessed via mobile phones, laptops and PCs. Spotify also allows for developers to use data for their web applications and applications provided the necessary authentication details.

Python was chosen as spotify has a well known and active python library to access and manipulate user data. Their library is referred to as Spotify and its well documented and widely used in the development of the mood detection system.

7.5 Water Management System

This module consists of water sensors and a pump. The pump is the kind of pump used in houses and institutions for water pumping. It is connected to the esp32 board. Water sensors are kept at 2 points in the tank to monitor the level of water and pump water accordingly. Water sensors were used to keep track of the level of water and monitor it throughout. The ESP is programmed in such a way that the water is pumped when the water level is below 30 percent and it stops pumping once it is 90 percent. C++ is the development language used with the Arduino IDE for writing the code to program the water leakage monitoring ESP. Data is collected regarding the level of water and it can be used by the user to see on how much water he is using.

Chapter 8 Implementation and Pseudo Code

8.1 Access Control Module

```
70  if(topic=="iot/maindoor"){
71      Serial.print("Opening iot main door ");
72      if(messageTemp == "true"){
73        digitalWrite(door1, HIGH);
74        room1_lock_count += 1;
75        formattedDate = timeClient.getFormattedDate();
76        int splitT = formattedDate.indexOf("T");
77        in_timestamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
78        Serial.println("Timestamp");
79        Serial.println(in_timestamp);
80        op_buzz();
81  }
82      else if(messageTemp == "false"){
83        digitalWrite(door1, LOW);
84        Serial.print("Closing iot main door");
85        room1_unlock_count += 1;
86        formattedDate = timeClient.getFormattedDate();
87        int splitT = formattedDate.indexOf("T");
88        out_timestamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
89        Serial.println("Timestamp");
90        Serial.println(out_timestamp);
91        cl_buzz();
92       }
```

Figure 21: Access Control Code

The above piece of code helps us control the state of the doors. First it checks for the topic name and then checks whether the request was for opening or closing the door. Line number 73 and 83 are for opening and closing the door respectively and the lines74-77 and 85-88 fetch the data such as number of locks and unlocks and the time it was locked and unlocked. We trigger the buzzer 1 time indicated by line 80 1 time when the door opens and 2 times indicated by line 91 when we close the door.

8.2 Temperature Monitoring and Control Module

```
145   if (meantempC >= 28) {
146     Serial.println("Turning fan on");
147     ctrl.forward();
148   }
149   else {
150     Serial.println("Turning fan off");
151
152   }
153   if(meantempC<=18){
154     Serial.println("Turning heater on");
155     digitalWrite(heater,HIGH);
156   }
157   else{
158     Serial.println("Turning heater off");
159     digitalWrite(heater,LOW);
160   }
161   if(meantempC>=28 && meantempC <31){
162     ctrl.setSpeed(255);
163     ctrl.forward();
164     Serial.println("Speed 2");
165   }
166   else if(meantempC>=31 && meantempC <34){
167     ctrl.setSpeed(150);
168     ctrl.forward();
169     Serial.println("Speed 3");
170   }
171   else if(meantempC>=34 && meantempC <37){
172     ctrl.setSpeed(180);
173     ctrl.forward();
174     Serial.println("Speed 4");
```

Figure 22: Temperature Monitoring and Control Code

Once we calculate the mean temperature we check if it is greater than 28 degrees and then we have intervals of width 3 degrees from 28 to 43 and we increase the speed of the fan as the temperature rises. E.g.) In line 162 we are setting the speed and then in line 163 we are switching the regulator which controls the fan on with the speed set above. This is done for every temperature interval.

8.3 Water Level Management Module

In this we are checking whether the water level is less than 4095 if yes, we are turning on the pump, once it exceeds 4095 we are turning the pump off. Here the sensor gives a 4 digit numeric value as the output as the water level.

```
81   if (now - lastMsg > 5000) {
82     lastMsg = now;
83  Serial.print("Water level: ");
84   watValue = analogRead(watPin);
85   Serial.println(watValue);
86   delay(500);
87   if(watValue >=0 && watValue<2800){
88     Serial.println("Turning pump on");
89     digitalWrite(pump,LOW);
90     client.publish("Stat",a);
91     leakage += 0;
92   }
93   if(watValue >=2800 && watValue <=4095){
94     Serial.println("Waterleaking");
95     leakage += 1;
96     digitalWrite(pump,HIGH);
97     client.publish("Stat",s);
98   }
```

Figure 23: Water Level Management Code

Chapter 9 Testing

The testing was conducted module wise and across various phases of development. As the project involved working with both hardware and software, different methods of testing were employed.

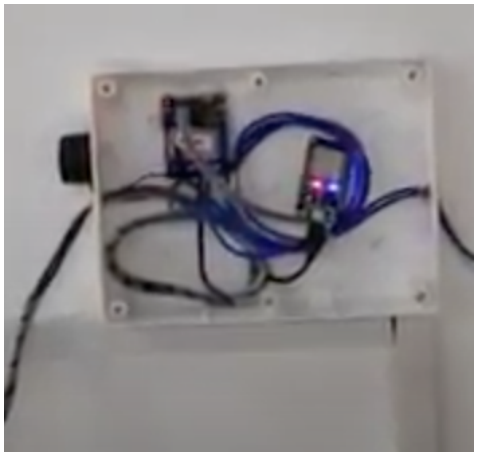


Figure 24: Hardware Test Bench

A hardware test bench was created to check if all the used hardware was compliant. The testbench had the required voltage input and had a multimeter connected at the output for checking the PWM signal coming out.

A 24hour endurance test was conducted where the access control system was deployed to the door of the IoT Lab. A few minor errors like Wi-Fi failure and network failure was experienced and sorted out along the way.

For testing the face recognition module, a group of 5 people was consisted where each member's face was registered on the system. Again a 5-hour endurance test was performed to check for the accuracy of the recognition under various lighting conditions.

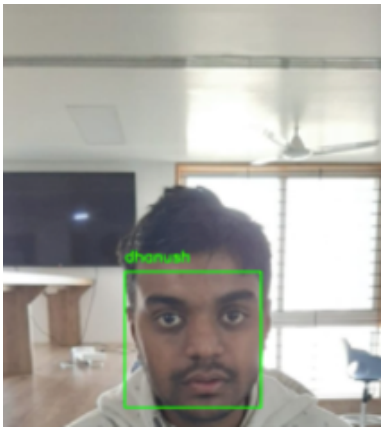


Figure 25: Face Recognition Testing

Chapter 10 Results and Discussion

The following were observed at the end of the project development,

- The Access Control Module was installed at G04 Classroom of Computer Science Department. The department office was given control of the system and was used for a duration of 2 weeks under the testing phase.
- The temperature control module has been set up at the PESU C-IoT lab where a mock-up was built with 4 temperature sensors and a fan. The user can choose between automatic or manual modes to control the fan. Under the automatic mode, the fan turns on at temperatures above 28 degrees centigrade.
- The gas detection module can find the presence of carbon di oxide in the atmosphere along with carbon monoxide.
- The water level management system detects the presence of water in the tank and also the level of water present in the tank, a mock-up of the same has been built and tested.
- The mood detection module keeps a track of the songs listened to by the user, comes up with a mood factor number and understands a user's current mood. Based on the factor, it recommends new songs and builds a playlist to help make the user feel better.

Chapter 11 Snapshots

11.1 Access Control System

11.1.1Buzzer Control

```
void op_buzz(){
  digitalWrite(door2,LOW);
  delay(700);
  digitalWrite(door2,HIGH);
  }
void cl_buzz(){
   digitalWrite(door2,LOW);
  delay(700);
  digitalWrite(door2,HIGH);
  delay(700);
  digitalWrite(door2,LOW);
  delay(700);
  digitalWrite(door2,HIGH);
 }
```

Figure 26: Buzzer Control Code

11.1.2 Wi-Fi Setup

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    p += 1;
    if(p>10){
      ESP.restart();
    }
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
  timeClient.begin();
  timeClient.setTimeOffset(offset);
 }
```

Figure 27: Wi-Fi setup Code

11.1.3 Topic Processing and Access Control

```
70   if(topic=="iot/maindoor"){
71       Serial.print("Opening iot main door ");
72       if(messageTemp == "true"){
73         digitalWrite(door1, HIGH);
74         room1_lock_count += 1;
75         formattedDate = timeClient.getFormattedDate();
76         int splitT = formattedDate.indexOf("T");
77         in_timestamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
78         Serial.println("Timestamp");
79         Serial.println(in_timestamp);
80         op_buzz();
81 }
82       else if(messageTemp == "false"){
83         digitalWrite(door1, LOW);
84         Serial.print("Closing iot main door");
85         room1_unlock_count += 1;
86         formattedDate = timeClient.getFormattedDate();
87         int splitT = formattedDate.indexOf("T");
88         out_timestamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
89         Serial.println("Timestamp");
90         Serial.println(out_timestamp);
91         cl_buzz();
92        }
```

Figure 28: Time Processing fro Access Control

11.1.4 Publishing to Topics

```
122 char lock1string[8],unlock1string[8],intime1[16],outtime1[16],lock2string[8],unlock2string[8],intime2[16],outtime2[16];
123 dtostrf(room1_lock_count, 1, 2,lock1string );
124 dtostrf(room1_unlock_count, 1, 2, unlock1string);
125 dtostrf(room2_lock_count, 1, 2,lock2string );
126 dtostrf(room2_unlock_count, 1, 2, unlock2string);*/
127 in_timestamp.toCharArray(intime1,16);
128 out_timestamp.toCharArray(outtime1,16);
129 in_timestamp2.toCharArray(intime2,16);
130 out_timestamp2.toCharArray(outtime2,16);
131 client.publish("Lock/iotmain",lock1string);
132 client.publish("Unlock/iotmain",unlock1string);
133 client.publish("Lock/cabin",lock2string);
134 client.publish("Unlock/cabin",unlock2string);
135 client.publish("iotmain/in",intime1);
136 client.publish("iotmain/out",outtime1);
137 client.publish("cabin/in",intime2);
138 client.publish("cabin/out",outtime2);
139 }
```

Figure 29: Publishing Topics Code

11.1.5 Reconnection function

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    digitalWrite(door3,HIGH);
    delay(2000);
    digitalWrite(door3,LOW);
    if (client.connect("iotmain")) {
      Serial.println("connected");
      digitalWrite(door3,HIGH);
      client.subscribe("iot/maindoor");
      client.subscribe("iot/cabin");
      }
    else {
      Serial.print("failed, rc=");
      ESP.restart();
      Serial.print(client.state());
      Serial.println(" try again in 4 seconds");
      delay(4000);
      }
  }
}
```

Figure 30: Wi-Fi Reconnection Code

11.1.6 Setup and Loop Function

```
void setup() {
  pinMode(door1, OUTPUT);
  pinMode(door2, OUTPUT);
  pinMode(door3, OUTPUT);
  digitalWrite(door1,HIGH);
  Serial.println("Lock on");
  digitalWrite(door2,HIGH);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
void loop() {
if (!client.connected()) {
    reconnect();
}
  if(!client.loop())
    client.connect("ESP32Client");
}
```

Figure 31: ESP Setup Code

11.2 Temperature Monitoring and Control

11.2.1 Device Address Function

```
202 unsigned char printAddress(DeviceAddress deviceAddress) {
203   for (uint8_t i = 0; i < 8; i++) {
204     if (deviceAddress[i] < 16) Serial.print("0");
205     Serial.print(deviceAddress[i], HEX);
206     return (deviceAddress[i]);
207   }
208 }
```

Figure 32: Issuing Device ID Code

11.2.2 Setup Sensors

```
86  sensors.begin();
87  numberOfDevices = sensors.getDeviceCount();
88  Serial.print("Locating devices...");
89  Serial.print("Found ");
90  Serial.print(numberOfDevices, DEC);
91  Serial.println(" devices.");
92  for (int i = 0; i < numberOfDevices; i++) {
93    if (sensors.getAddress(tempDeviceAddress, i)) {
94      Serial.print("Found device ");
95      Serial.print(i, DEC);
96      Serial.print(" with address: ");
97      devices[i] = printAddress(tempDeviceAddress);
98      Serial.println();
99    } else {
100     Serial.print("Found ghost device at ");
101     Serial.print(i, DEC);
102     Serial.print(" but could not detect address. Check power and cabling");
103   }
104 }
```

Figure 33: Sensor Setup Code

11.3 Facial Recognition

11.3.1 Encode Faces

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
    help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
    help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
    help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))
knownEncodings = []
knownNames = []
for (i, imagePath) in enumerate(imagePaths):
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    boxes = face_recognition.face_locations(rgb,
        model=args["detection_method"])
    encodings = face_recognition.face_encodings(rgb, boxes)
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
```

Figure 34: Encoding Faces

11.3.2 Face Detection Video Stream

```
 8  ap = argparse.ArgumentParser()
 9  ap.add_argument("-e", "--encodings", required=True,
10      help="path to serialized db of facial encodings")
11  ap.add_argument("-o", "--output", type=str,
12      help="path to output video")
13  ap.add_argument("-y", "--display", type=int, default=1,
14      help="whether or not to display output frame to screen")
15  ap.add_argument("-d", "--detection-method", type=str, default="cnn",
16      help="face detection model to use: either `hog` or `cnn`")
17  args = vars(ap.parse_args())
18  print("[INFO] loading encodings...")
19  data = pickle.loads(open(args["encodings"], "rb").read())
20  print("[INFO] starting video stream...")
21  vs = VideoStream(src=0).start()
22  writer = None
23  time.sleep(2.0)
24  while True:
25      frame = vs.read()
26      rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
27      rgb = imutils.resize(frame, width=750)
28      r = frame.shape[1] / float(rgb.shape[1])
29      boxes = face_recognition.face_locations(rgb,
30          model=args["detection_method"])
31      encodings = face_recognition.face_encodings(rgb, boxes)
32      names = []
33      for encoding in encodings:
34          matches = face_recognition.compare_faces(data["encodings"],
35              encoding)
```

Figure 35: Face Detection Video Stream

```
                counts[name] = counts.get(name, 0) + 1
            name = max(counts, key=counts.get)
        names.append(name)
    for ((top, right, bottom, left), name) in zip(boxes, names):
        top = int(top * r)
        right = int(right * r)
        bottom = int(bottom * r)
        left = int(left * r)
        cv2.rectangle(frame, (left, top), (right, bottom),
            (0, 255, 0), 2)
        y = top - 15 if top - 15 > 15 else top + 15
        cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
            0.75, (0, 255, 0), 2)
    if writer is None and args["output"] is not None:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 20,
            (frame.shape[1], frame.shape[0]), True)
    if writer is not None:
        writer.write(frame)
    if args["display"] > 0:
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break
cv2.destroyAllWindows()
vs.stop()
if writer is not None:
    writer.release()
```

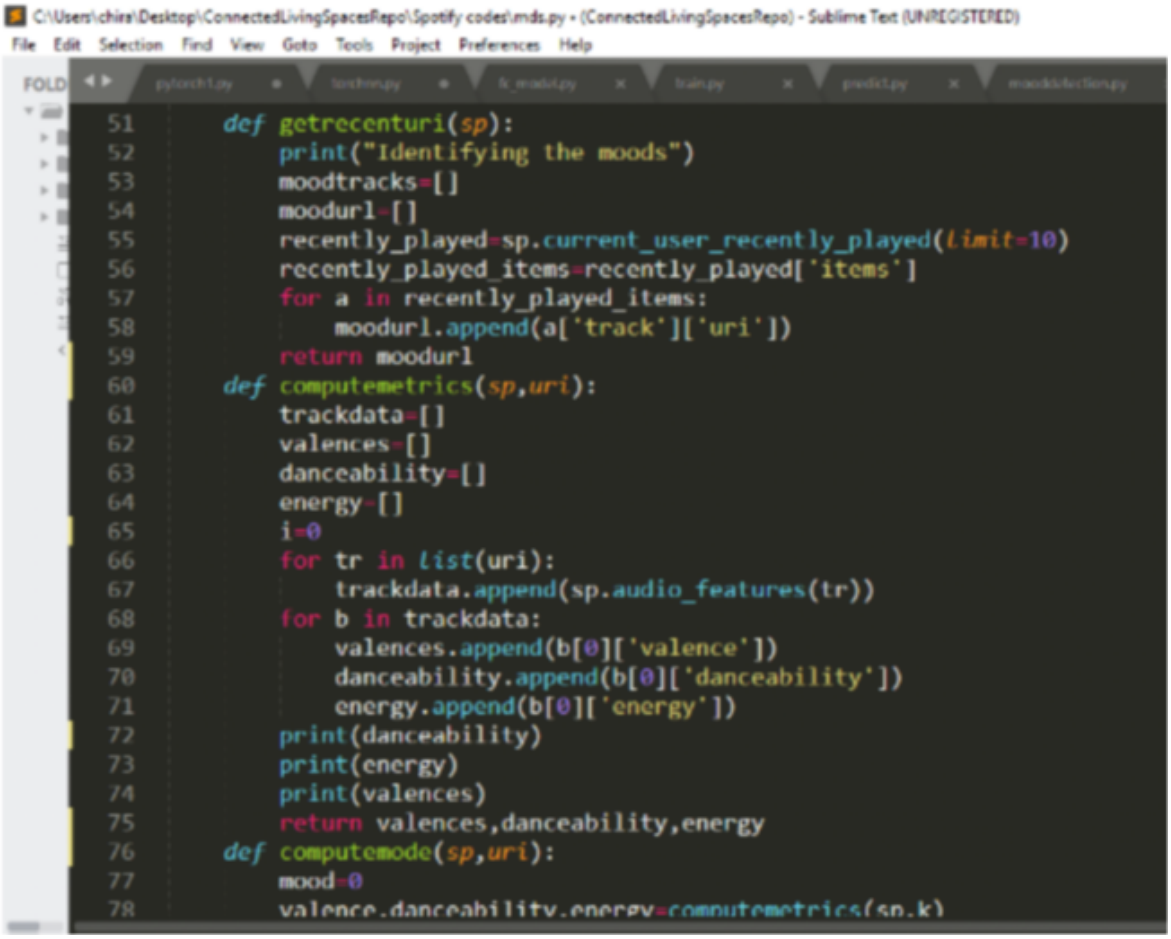
Figure 36: Face Detecion Output

11.4 Mood Detection System

11.4.1 Authentication

token=util.prompt_for_user_token(username,scope ,client_id=client_id,client_secret=clien
t_secre t,redirect_uri=redirect_uri)
if token:
def authenticate_spotify():
print("..Connecting to spotify")
sp=spotipy.Spotify(auth=token)
return sp

11.4.2 Detection



```
def getrecenturi(sp):
    print("Identifying the moods")
    moodtracks=[]
    moodurl=[]
    recently_played=sp.current_user_recently_played(limit=10)
    recently_played_items=recently_played['items']
    for a in recently_played_items:
        moodurl.append(a['track']['uri'])
    return moodurl
def computemetrics(sp,uri):
    trackdata=[]
    valences=[]
    danceability=[]
    energy=[]
    i=0
    for tr in list(uri):
        trackdata.append(sp.audio_features(tr))
    for b in trackdata:
        valences.append(b[0]['valence'])
        danceability.append(b[0]['danceability'])
        energy.append(b[0]['energy'])
    print(danceability)
    print(energy)
    print(valences)
    return valences,danceability,energy
def computemode(sp,uri):
    mood=0
    valence,danceability,energy=computemetrics(sp,k)
```

Figure 37: Song Detection for Mood

11.4.3 Processing and Delivery

```
def aggregate_top_artists(sp):
    print("Getting your top artists")
    name=[]
    uri=[]
    ranges=['short_term','medium_term','long_term']
    for r in ranges:
        artists_all_data=sp.current_user_top_artists(limit=50,time_range=r)
        artists_data=artists_all_data['items']
        for artist_data in artists_data:
            if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    followed_artists_all_data=sp.current_user_followed_artists(limit=50)
    followed_artists_data=followed_artists_all_data['artists']
    for artist_data in followed_artists_data["items"]:
        if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    return uri
def aggregate_top_tracks(sp,uri):
    print("...Getting top tracks")
    tracks=[]
    for artist in uri:
        top_tracks_all_data=sp.artist_top_tracks(artist)
        top_tracks_data=top_tracks_all_data['tracks']
        for track_data in top_tracks_data:
            tracks.append(track_data['uri'])
    return tracks
```

Figure 38: Processing Mood from Songs

```
def select_tracks(sp,tracks,mood):
    print("Selecting top tracks")
    print(mood)
    selected_tracks=[]
    random.shuffle(tracks)
    for t in list(tracks):
        tracks_all_data=sp.audio_features(t)
        for track_data in tracks_all_data:
            try:
                if mood<0.10:
                    if(0<=track_data["valence"]<=(mood +0.15) and track_data["danceability"]<= (mood*8) and track_data["energy"] <= (mood*10)):
                        selected_tracks.append(track_data["uri"])
                elif 0.10 <= mood <= 0.25:
                    if((mood-0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]<= (mood*4) and track_data["energy"] <= (mood*5)):
                        selected_tracks.append(track_data["uri"])
                elif 0.25 <= mood <= 0.50:
                    if((mood - 0.05)<=track_data["valence"]<=(mood +0.05) and track_data["danceability"]<= (mood*1.75) and track_data["energy"] <= (mood*1.75)):
                        selected_tracks.append(track_data["uri"])
                elif 0.50 <= mood <= 0.75:
                    if((mood - 0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]>= (mood/2.5)and track_data["energy"] >= (mood/2)):
                        selected_tracks.append(track_data["uri"])
                elif 0.75 <= mood <= 0.90:
                    if((mood - 0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]>= (mood/2)and track_data["energy"] >= (mood/1.75)):
                        selected_tracks.append(track_data["uri"])
                elif mood > 0.90:
                    if((mood - 0.15)<=track_data["valence"]<= 1 and track_data["danceability"]>= (mood/1.75)and track_data["energy"] >= (mood/1.5)):
                        selected_tracks.append(track_data["uri"])
            except TypeError as te:
                continue
    return selected_tracks
```

Figure 39: Processing Mood Value

11.5 Water Leakage System

11.5.1 Reading Water Level and Controlling Pump

```
81   if (now - lastMsg > 5000) {
82     lastMsg = now;
83  Serial.print("Water level: ");
84   watValue = analogRead(watPin);
85   Serial.println(watValue);
86   delay(500);
87   if(watValue >=0 && watValue<2800){
88     Serial.println("Turning pump on");
89     digitalWrite(pump,LOW);
90     client.publish("Stat",a);
91     leakage += 0;
92   }
93   if(watValue >=2800 && watValue <=4095){
94     Serial.println("Waterleaking");
95     leakage += 1;
96     digitalWrite(pump,HIGH);
97     client.publish("Stat",s);
98   }
```

Figure 40: Reading Water Level Values

11.5.2 Setup Wi-Fi Function

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    p += 1;
    if(p>10){
      ESP.restart();
    }
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
  timeClient.begin();
  timeClient.setTimeOffset(offset);
}
```

Figure 41: Connecting sensors to Wi-Fi

11.5.3 Call Back Function

```
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;
  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();
}
```

Figure 42: Call Back Function

11.5.4 Reconnect Function

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client")) {
      Serial.println("connected");
      digitalWrite(door2, HIGH);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      ESP.restart();
      delay(5000);
    }
  }
}
```

Figure 43: Wi-Fi Reconnection intime of Failure

Chapter 12 Conclusions

- The users are able to control the access to specific rooms, doors using the access control system, facial recognition system and the voice recognition system with minimum delay.

- They can also access the data regarding the number of locks and the unlocks and arrival of intruders and the time a specific door was locked and unlocked.

- The users are also able to benefit from the curated playlists created by the mood detection system.

- The users reap benefits from the water leakage system and the gas leakage system and makes life simpler and easier.

- The automatic cooling system keeps the temperature normal every time and monitors the temperature continuously.

Chapter 13 Further Enhancements

- A RFID interface can be incorporated to register individual students.
- A Biometric system can be setup at sensitive facilities which need an added level of security.
- The Access Control Module can be used to mark the attendance of the students present inside the class by using the data logged from the biometric/RFID system.
- User Registration can be made simpler by linking to “Sign up using Google” or similar services.

Bibliography

The following are the resources which were used to understand and learn during the development phase:

- https://spotipy.readthedocs.io/en/2.11.1/
- https://face-recognition.readthedocs.io/en/latest/
- https://randomnerdtutorials.com/guide-for-mq-2-gas-smoke-sensor-with-arduino/
- https://www.tutorialspoint.com/arduino/arduino_water_detector_sensor.htm
- https://github.com/espressif/arduino-esp32
- https://randomnerdtutorials.com/esp32-multiple-ds18b20-temperature-sensors/
- https://www.pubnub.com/blog/pubsub-nodemcu-32s-esp32-mqtt-pubnub-arduino-sdk/
- https://www.instructables.com/id/IoT-Androino-Doorlock/
- https://nodered.org/docs/tutorials/
- https://iotdesignpro.com/projects/arduino-based-amazon-alexa-controlled-home-automtaion
- https://www.instructables.com/id/Remote-Temperature-Monitoring-Using-MQTT-and-ESP82/
- https://www.youtube.com/watch?v=XDrwgMSQrEY

Appendix A

- MQTT - Lightweight, publish-subscribe network protocol that transports messages between devices
- ONEWIRE - Device Communication bus system that provides data, signalling and power over a single conductor
- CALLBACK- Function passed to another function as an argument to complete some kind of routine or action
- NODERED - A flow based development tool created by IBM for visual computation
- OPENCV - A library of functions aimed at real time computer vision
- DLIB - General purpose cross-platform software library written in C++ which supports machine learning algorithms and concepts
- URI-String of characters used to identify a resource on a computer network.
- THRESHOLD - An amount or a limit on a scale based on which decisions are made or action is taken.
- RGB - Red, green, blue it's a way the colour of an object is represented
- PICKLE - A file format for string data