Group 6
CSC 591791 ECE 592792
Personalized Home Automation

Renji Joseph Sabu
Dept. of Computer Science
rsabu@ncsu.edu

Ashish Ajayakumar
Dept. of Computer Engineering
aajayak@ncsu.edu

Jubitta John
Dept. of Computer Science
jjohn6@ncsu.edu

Submitted on: April 25, 2023

Contributions breakdown

Component	**Component weightage**	**Renji's Contribution**	**Jubitta's Contribution**	**Ashish's Contribution**
High level design	0.10	40%	30%	30%
Device Automation Code	0.20	40%	30%	30%
API and Database	0.30	40%	30%	30%
Preferences Console	0.10	50%	30%	20%
Hardware setup	0.20	20%	30%	50%
Report writing	0.10	10%	60%	30%
Per student aggregate contribution		34	33	33

I. Introduction

In today's world, automation has become a reality, with more and more tasks being completed automatically on a daily basis. This includes basic functions such as turning devices on and off, both remotely and in close proximity. As machines take over control of devices, monitoring and reporting become increasingly important. While we are relinquishing power for simple but routine tasks, we still need to maintain as much control as possible over automated processes. Although automation reduces the need for human judgment, it does not eliminate it entirely. Depending on its usage, automation takes on different names such as industrial automation and home automation. In this project, we will focus on home automation, which has evolved from being an industrial application to a low-cost solution for households. Market research suggests that home automation systems will become ubiquitous in the near future.

The aim of this project is to enable personalized control of a room based on the occupant's preferences. Each occupant will be able to define their preferences, which can be changed at any time to control various devices such as the heater, cooler, humidifier, fan, and lighting settings. The occupant's preferences can be activated by scanning their RFIDs when they enter the room. The preferences are restored to their previous values by scanning their RFIDs when leaving the room.

II. Design

A. Overview

The system uses an RFID scanner to permit entry to the room. Upon scanning their RFID ID, the occupant is identified and the room's environmental variables are adjusted accordingly. Following are the environmental variables that can be controlled:

- **Temperature using Heater/Cooler**: DHT11 temperature/humidity sensor can be used to detect the current room temperature. If the preferred temperature setting is below the current room temperature, the cooler is turned ON. Conversely, if the preferred temperature setting is above the current room temperature, the heater is turned ON. In short, either the heater or cooler is turned on at a point to match the preferred temperature.
- **Humidity using the Humidifier**: DHT11 temperature/humidity sensor can be used to detect the current room humidity. If the current humidity of the room is less than the preferred humidity setting, the humidifier is turned ON. Conversely, if the current humidity of the room is greater than the preferred humidity, the humidifier is turned OFF.
- **Fan speed**: The speed of the fan can be changed on a scale of 0 to 5 as per the preferences. A speed of 0 indicates that the fan is OFF.
- **Light brightness and color**: The brightness and color of the light in the room can be adjusted based on the preferences. The light brightness can be varied on a scale of 0- 100. The colors of the light in the room can be set to: white, red, blue, green, yellow, magenta, or, cyan.

The system can encounter the following three different scenarios:

- If only one occupant is present in the room, the environmental variables are set to his/her preferences.
- In the event that more than one occupants are present in the room, the environmental variables are set to the occupants shared preferences which is a set of predefined settings that was agreed upon beforehand.
- If the room is empty, the environmental variables will automatically switch to default settings.

The system employs an **LCD display** as a console device to provide the occupant with real-time updates on the room's status, including details on the current occupants and the environmental settings.

Occupant can change their individual environmental settings or shared environmental settings by using the **webpage** developed. (This serves as the front end of the system). The webpage also displays the current environmental settings for each occupant and their shared preferences.

B. Flow diagram

Given below is the flow diagram that shows in detail the scenarios that can be encountered in the room and scanning. The actions taken in these situations are also addressed in the flow diagram. (The diagram is scaled down to fit the double column format. Kindly zoom in to get a detailed look.)

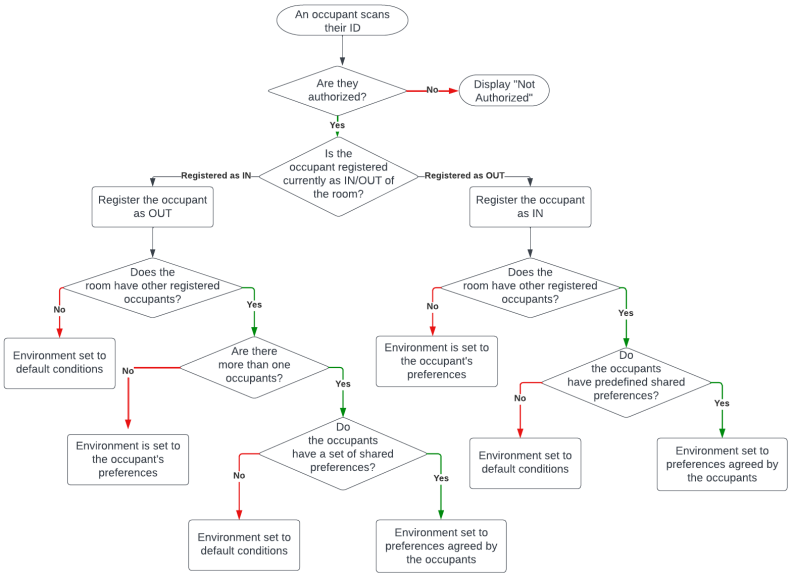

Fig. 1. Flow diagram

C. Block Diagram

The entire system for the personalized home automation developed has 6 major components or blocks. All of these blocks work together for the successful working of the entire system.

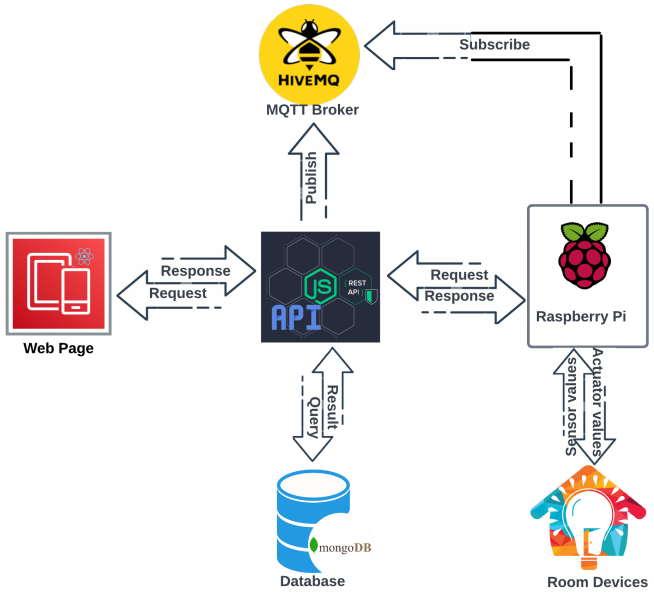

Fig. 2. Block diagram of the entire system

1) Raspberry Pi: The Raspberry Pi serves as the controller of the entire system. Raspberry Pi is the central component that has room devices connected to it. A Python code runs on the Raspberry Pi controlling the devices.

The room devices connected to the Raspberry Pi has sensors and actuators. The sensor values (temperature and humidity) are read every 15 minutes by the Raspberry Pi which uses both the sensor values and the preferred settings of the occupants to compute the output values for the actuators. These output values are then passed to the actuators for appropriate action.

The Raspberry Pi fetches the room settings of the current occupants from the database by making an API call request when it is turned on. The Raspberry Pi also fetches the preferences when a new occupant enters the room by making the API call. The received values are then used for the calculations along with the sensor values to compute the values of the actuators.

The Raspberry Pi is also connected to the MQTT broker. The Raspberry Pi is subscribed to an occupant specific topic which receives an ”update” if they are changed from the web page. The new preferences are then requested from the API. Using MQTT enables the preference changes made through the web page to take effect in real time.

2) MQTT Broker: HiveMQ MQTT Broker is used to enable the MQTT communication between the Raspberry Pi and the API which is required to reflect the preference changes to take effect in real time. The Raspberry Pi is a subscriber and API is a publisher. The Broker is run on a Laptop to enable the communication. Whenever an occupant's preferences are added, changed, or when a shared room preferences are approved by all the users, it is published on the occupant's or occupant set's topic. The Raspberry Pi is subscribed to an occupant specific topic which receives the settings preferred by the occupant if they are changed from the web page.

3) Room Devices: The detailed block diagram of the room devices connected to the Raspberry Pi is given below. This includes the sensors and the actuators.

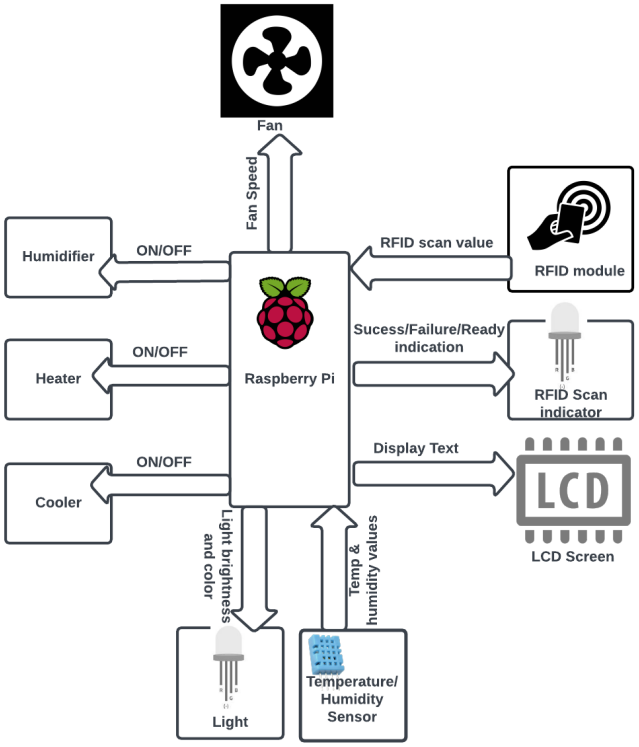

Fig. 3. Block diagram of the room devices and raspberry pi

- **RFID module**: The RFID module consists of two components - the RFID reader/writer and RFID tags assigned to each user attempting to enter the room. Each user is provided with a unique RFID tag containing their name. During the initial setup of the RFID module, the configuration process involved assigning the RFID tags to different users. The code 'rfid.py' written for the initial configuration writes the user's name to the respective RFID tag.
  The occupant enters and leaves the room by scanning the RFID tags. When the tag is scanned, the card owner name is read and this is used to update the rooms' occupants list and change the environment variables accordingly.

- **RFID Scan Indicator**: An RGB 4-pin LED with a common cathode is utilized to signify the outcome of the RFID tag scan (success or failure) as well as the readiness of the RFID reader to scan a RFID tag. The scan indicator shows three colors which indicates the following:
  - Green: Blinking of green light indicates the scan of an authorized occupant of the room.

  - Red: Blinking of red light indicates the scan of an unauthorized occupant of the room.
  - Blue: Blue light stays on when the RFID scanner is ready to scan a RFID tag.
- **LCD Screen**: A 20x4 LCD module is used as a console to display the following informations:
  - Welcome message when an occupant enters the room.
  - Good bye message when an occupant leaves the room.
  - Current occupants of the room.
  - Current environment variables of the room.
- **Temperature/Humidity sensor**: The DHT11 temperature humidity module is utilized to sense the current temperature and humidity of the room. Based on the readings, it controls the operation of the cooler/heater and humidifier to maintain the desired temperature and humidity levels. The values from the sensor are passed to the Raspberry Pi every 15 minutes.
- **Light**: A RGB 4 pin common cathode LED is used to control the light brightness and color of the light based on the preferences. The light brightness can be varied on a scale of 0- 100. The colors of the light in the room can be set to: white, red, blue, green, yellow, magenta, or, cyan.
- **Cooler**: A 2 pin LED is used to specify the ON/OFF status of the cooler. The LED lights up when the cooler is ON and vice versa when it is OFF.
- **Heater**: A 2 pin LED is used to specify the ON/OFF status of the heater. The LED lights up when the heater is ON and vice versa when it is OFF.
- **Humidifier**: A 2 pin LED is used to specify the ON/OFF status of the humidifier. The LED lights up when the humidifier is ON and vice versa when it is OFF.
- **Fan**: A DC motor with mini fan is connected to the Raspberry Pi using I2C bus using the motor driver. The motor is powered using AAA batteries. The speed of the fan can be changed on a scale of 0 to 5 as per the preferences.

4) API: A REST API is implemented in Node.js and Express to enable the communication between the web page(frontend), database and the Raspberry Pi(controller).

Raspberry Pi calls the API to fetch and update data in the database for the following two cases:

- GET /rooms/:room/settings endpoint is called when the Raspberry Pi controller is turned on. This is used to get the current room settings and the occupants.
- PUT /rooms/:room/users/:user/scan endpoint is called when an RFID tag is scanned to mark an occupant in or out of a room and return the current occupants and settings of a room. This API also prints the error when an unauthorized user scans.

Web page calls API in the following cases:

- When an authorized occupant adds or updates the environment variable preferences.
- When a new request is made by an occupant for a shared environment variable preference settings.
- When an update request is made by an occupant for a shared environment variable preference settings.
- When a pending request is approved by an occupant or by all the occupants for a shared environment variable preference settings.

API also publishes to MQTT broker whenever an occupant's preferences are added, changed, or when a shared room preferences request is approved by all the users to enable the changes to take effect in real time.

5) Database: The database is implemented in MongoDB which stores the data handled by the API.The connection to the database is established by specifying the URL of the Database in the .env file of API setup.

The database used for this project has the following collections in it:

- *preferences*: Stores individual and shared environment variable preferences.
- *requests*: Stores requests from occupants to update the shared preferences when it isn't approved by all the occupants.
- *rooms*: Stores the list of current occupants of the room.
- *users*: Stores the list of authorized occupants of the room.

6) Web page: The web page built in React serves as the frontend of the system. The web page can be used to establish the following tasks using the API calls:

- To view and update an individual occupant's preferences.
- To add and update the shared preferences request by an occupant and to view them.
- To approve a pending preferences from another occupant for the shared preferences setting.

D. User controllable parameters

There are two sets of user controllable parameters that can be changed while running the entire system.

- RFID scanning: The user can scan the RFID tags when entering or exiting a room. This change controls the personalized automation of the room.
- Web Page changes(changes in environmental variables preferences): The web page enables the addition and changes in the environment variable preferences as explained in the previous section. This change controls the choice of the personalized automation of the room. The sample json file that contains all the environmental variable preferences are given below:


```
{
  "preferences": {
    "_id": "Ashish",
    "temperatureInF": 65,
    "fanSpeed": 3,
    "lightBrightness": 68,
```

```
    "lightColor": "Red",
    "humidity": 61
  }
}
```


E. Circuit diagrams

The circuit diagram connection of the room devices to the Raspberry Pi is given below. There are two diagrams. The first one is the fritzing diagram and the second one is the pin diagram.

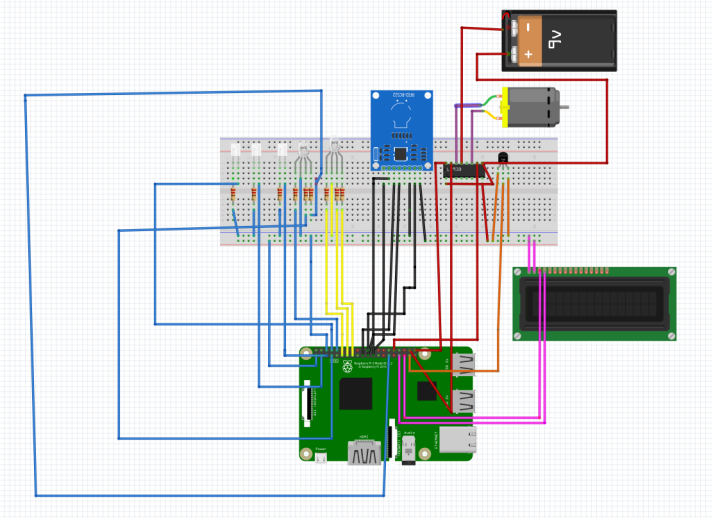

Fig. 4. Fritzing diagram of the connection of room devices

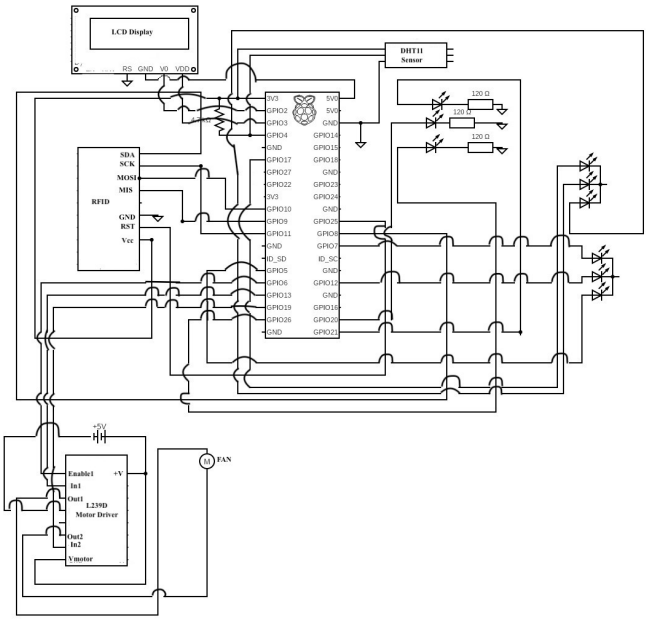

Fig. 5. Pin diagram of the connection of room devices

III. Implementation

A. Demo Steps

- **Step1**
  Prerequisite: The room should be empty.
  Description: Occupant 1 scans their RFID to enter the room.
  Expected Behavior: Room settings changed to occupant 1 preferred settings.
- **Step2**
  Prerequisite: The room should be empty.
  Description: Occupant 1 scans their RFID on the way out of the room, and the room is now empty.
  Expected Behavior: Room settings change to default settings.
- **Step3**
  Prerequisite: The room should be empty.
  Description: Occupant 1 enters the room followed by Occupant 2 and scan their IDs in that order.
  Expected Behavior: Room settings change to settings preferred when both occupants are present.
- **Step4**
  Prerequisite: The room should be empty.
  Description: Occupant 1 walks out of the room, and Resident 2 remains.
  Expected Behavior: Room settings changed to occupant 2 preferred settings.
- **Step5**
  Prerequisite: Occupant 1 is alone in the room.
  Description: Occupant 1 changes their preferences.
  Expected Behavior: The room settings change to the updated preferences.
- **Step6**
  Prerequisite: The room has both Occupant 1 and Occupant 2 and them only.
  Description: Occupant 1 sends a request to Occupant 2 with an updated set of suggested preferences on the webpage. Occupant 2 accepts the request.
  Expected Behavior: The room settings change to the updated set of preferences.

B. Hardware used

- Basics
  - Raspberry Pi 4
  - MicroSD card
  - Breadboard
  - Power Supply for Raspberry Pi
  - Wires
  - Resistors
- RFID read or write
  - RC522 RF IC Card Sensor Module
  - S50 Blank Card * 2, one for each resident
- DHT11 Temperature/Humidity Sensor
- Display - 20x4 LCD Display
- Fan Setting
  - 5V DC Motor with mini fan

  - L293D Motor Driver
  - AAA batteries to power the DC motor
- Lighting - 4 pin RGB LED
- Humidifier and air conditioning
  - LEDs
  - Resistors

C. Softwares and Tools used

Find the details of the softwares and tools used for each block below:

- **Raspberry Pi**: Python, RPi.GPIO, mfrc522, Adafruit-DHT, RPLCD, paho-mqtt, python-decouple.
- **Database**: MongoDB.
- **MQTT Broker**: HiveMQ.
- **API**: Node.js, Express.js, Node.js MQTT library.
- **Web Page**: JavaScript with React and Next.js.

D. Source code info

The source code can be divided into three major sets:

- Raspberry Pi controller source code
- Web page(front end) source code
- RESTful API source code

1) Raspberry Pi controller source code: Personalized Home Automation Raspberry Pi controller code works along with the Personalized Home Automation API which serves the endpoints, Personalized Home Automation Preferences Console to view and update preferences to set the preferences of a room according to the occupants set of preferences.

This set of changes has the following files in them:

- .env: File containing info on the environment variables.
- rfid/main.py: Used to configure the RFID tags.
- requirements.txt: The file containing the required installations.
- main.py: The main program that controls the entire flow of the Raspberry Pi module connected to the room devices.
- src: This folder contains the program files that are used to implement the sensor and actuators.
  - dht11.py: Reads the temperature and humidity sensor values and turns on or off the cooler, heater and humidifier based on it.
  - display.py: Used to display the text in LCD screen.
  - environment.py: Stores the global variables.
  - fan.py: Used to control the fan speed.
  - handler.py: Used to handle the case where an unauthorized user scans.
  - lighting.py: Used to change the brightness and color of light.
  - scanIndicator.py: Used to control the LED that is the RFID scan indicator.
  - subscriber.py: Used to implement the MQTT subscriber changes.

2) Web page source code: This contains changes for the webpage built on React and Next.js and is used by an occupant to see and update their individual preferences, any shared preferences and any requests for shared preferences. This is a Next.js project bootstrapped with create-next-app. The changes are too many to be explained in detail here.

3) RESTful API source code: This is a set of RESTful API serves the endpoints used by the Personalized Home Automation project. The endpoints are called by the Raspberry Pi controller, which controls the home devices and also by the Preferences Console, which is used by the user to view and change their preferences.

The following are the code changes present in this module:

- .env: File containing info on the environment variables.
- index.js: The main entry point of the application containing the server configuration.
- package.json: File that contains metadata about the project, such as the project name, version number, and a list of dependencies.
- package-lock.json: File that is automatically generated by npm when installing packages.
- src: This folder contains the API changes. The files inside are:
  - mongo.js: Used for implementing the functionality to connect to a MongoDB database.
  - mqtt.js: Used for establishing MQTT connection.
  - models: Performs queries to the database.
  - controllers: Components that manage the business logic and mediate the interaction between models and handlers.
  - handlers: Functions or components that process incoming requests and send responses.

  models, controllers and handlers folder inturn has 4 files inside them for each of the 4 modules of the API.

IV. Results and Discussion

A. Outputs observed by user

The output can be observed by the user on three different modules:

- The room devices connected to the Raspberry Pi: This shows the entire set of room devices explained in the block diagram.
- The web page viewed from the phone.
- The LCD screen display.

Find the images for these three modules in the next page:

B. Observations

The system which we have designed and built uses a lot of wired connections. To work even a small component like a 9V DC motor, we had to use a motor driver to protect the Raspberry Pi pins and made the system bulkier. If this system is made practical, we will need to use a more complex wiring which includes relays for the heater, cooler and humidifier, and this could even require work on the original wiring inside the product. Working on this project made us think about how this

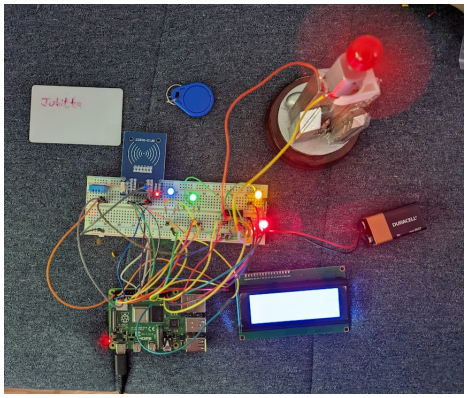

Fig. 6. Room devices connected to the Raspberry Pi

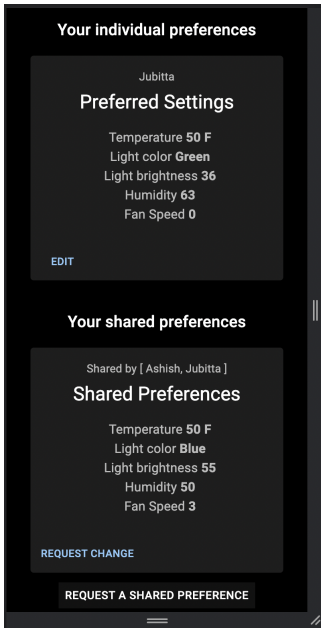

Fig. 7. Web page

product can be improved by using Smart Plugs, which works wireless, pausing a much lesser danger to the Raspberry Pi controller and the devices, while also highly improving the compactness, feasibility, and durability of the system while also making it user friendly.

C. Challenges encountered

- The free json database Pantry Cloud was the initial choice for the database. And we nearly wrote half the code to access data from the database. But during the demo and testing, the Pantry Cloud was found to be extremely slow and unreliable. To resolve this issue, a locally hosted REST API is developed to serve the data, improving reliability.
- Preferences changed through the web page were not designed to reflect in real time. This was because the relation between the Raspberry Pi controller and the API was purely client-server based, and so the Raspberry Pi could only get updates when some event triggers an API call. We added real time updating of room settings to the system by using MQTT to push an update from the API to the subscribing Raspberry Pi controller.
- We also encountered issues with some components not working due to loose connections. It was tough to troubleshoot the problem because of the large number of connections involved. We initially planned to solder all the components to PCB to address this issue. But because of time constraints we had to discard this plan and not go with soldering the components that caused the most issue such as battery and motor fan.
- We also faced issues with some of the hardware components not working in between the development phase and had to replace the items.

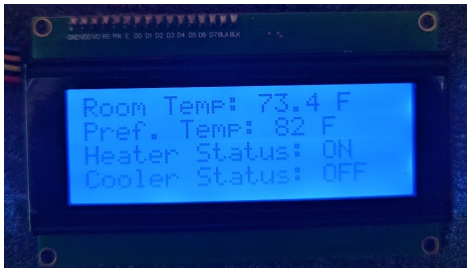

Fig. 8. LCD screen display

D. Future Scope

- Connect to smart plugs to automate wirelessly.
- Room specific preferences for each user.
- Enable voice commands

REFERENCES

[1] PiMyLife, “How to setup a Raspberry Pi RFID RC522 Chip” . https://www.pimylifeup.com/raspberry-pi-rfid-rc522/
[2] ElectronicsHub, ” Controlling a DC Motor with Raspberry Pi ”, https://www.electronicshub.org/controlling-a-dc-motor-with-raspberry-pi/
[3] Autodesk Instructubles, ” Raspberry Pi Tutorial: How to Use a RGB LED ” https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-a-RGB-LED/
[4] Fredric, Freva, ” How to connect an LCD display to a Raspberry Pi ” https://www.freva.com/how-to-connect-an-lcd-display-to-a-raspberry-pi/
[5] Scott Campbell, Circut Basics ” How to set up the DHT11 Humidity sensor on the Raspberry Pi” https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/
[6] HiveMQ: https://www.hivemq.com/downloads/hivemq/
[7] MongoDB: https://www.mongodb.com
[8] npm mqtt: https://www.npmjs.com/package/mqtt