The International Journal of Engineering and Science (IJES)
|| Volume || 12 || Issue || 12 || Pages || PP 01-16 || 2023 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

# A Green IoT Solution Coupled with Machine Learning to Tackle Climate Change

Chiang Liang Kok[1], Chan Shimei[2], Yit Yan Koh[1], Chee Kit Ho[3]
[1]*University of Newcastle, Australia*
[2]*Singapore University of Social Science*
[3]*Singapore Institute of Technology*
*Corresponding Author: chiangliang.kok@newcastle.edu.au*

**ABSTRACT**
*The rapid climate change is a huge concern around the world. Singapore is also very concerned as she has also experienced the negative consequences because of climate change. The Singapore government has identified the installation and the uses of air conditioning (AC) as one of the top factors that contributes to climate change. Hence, in this project, the focus is to study and improve on the current heating, ventilation, and air conditioning (HVAC) system. The ultimate objective of this project is to use Internet of Things (IoT) to retrieve and analyze data, and for Machine Learning to train and predict the ON/OFF state of the HVAC's damper. Thereafter, to allow the HVAC system to be automated. This report will highlight the end-to-end processes, starting from data collection using the various IoT devices, to storing and manipulating the interior environment & activity data as well as exterior environment data at MySQL via the Node-RED. Finally, to obtain the target value after training the model. Live data will be used for the system to predict future incoming data.*

***KEYWORDS;-*** *IoT solution, HVAC, air-conditioning system, machine learning, climate change*

Date of Submission: 20-11-2023 Date of acceptance: 05-12-2023

## I. INTRODUCTION

The use of air conditioning (AC) releases many greenhouse gases such as carbon dioxide that are harmful to the environment. Another example of harmful greenhouse gases that are released is carbon monoxide which is produced when electric-driven generators are used to power AC. On top of that, harmful greenhouse gases such as CFCs(chlorofluorocarbons) and HFCs(hydrofluorocarbons) are also found as the main refrigerant used in AC [1]. Just from the year 2018 alone, carbon emissions rose by another 2%. This is the fastest rate the world has seen in 7 years. This shows that the earth is indeed experiencing a climate emergency [2]. Currently, AC usage contributes to approximately 20% of the total electricity used in buildings around the world. The world's consumption of AC devices is increasing and is expected to go up 3-fold by the year 2025. While China, United States and Japan are presently using the most AC, the demand for AC is also on the rise elsewhere, especially in warmer countries such as India, Indonesia, and Middle Eastern countries [3]. Since the usage of AC increases across the globe, more carbon dioxide, carbon monoxide, CFCs and HFCs are also being released into the environment. These harmful greenhouse gases trap heat in the atmosphere, resulting in negative environmental effects such as increase of the Earth's temperature, ozone depletion and climate change.

## II. OBJECTIVE

The motivation for doing this project is to be able to contribute back to save the environment by slowing down the impact of global warming and climate change. The goal is to try and improve on the current heating, ventilation, and air conditioning (HVAC) system by making the system smarter using automation so that we can help to mitigate excessive harmful greenhouse gas emission by controlling and limiting unnecessary usage of AC. The objective of this project is to create a prototype that can determine the state of the damper within the HVAC system by analyzing the surrounding parameters that are both within and outside of the building. In this proposed work, IoT (Internet of Things) sensors and machine learning (ML) algorithms will be deployed to determine if the state of the damper should be ON (1) or OFF (0). One of the factors that determine the state of the damper will also depend on the "user feel-like temperature". Hence, when the user feels that the temperature is warm, the damper will turn on automatically, allowing more cold air to flow in. Likewise, when the temperature is cold, the damper will be turned off to prevent cold air from flowing into the zone. This allows temperature to regulate by itself and allocates the right temperature to the right zones/rooms.

## III. LITERATURE REVIEW

HVAC consists of heating, ventilation, and air-conditioning units. A HVAC system should cool down the temperature and ensure acceptable air quality in the indoor environment. HVAC is a sophisticated system that can detect the temperature inside the building [4][8]. There are three different types of HVAC systems, namely, central system, decentralized and the local system. Typically, for large scale usages such as modern commercial office buildings, shopping malls, the central system is a preferred choice. Within the HVAC, there is a Variable Air volume (VAV). The VAV air-conditioning system is a very complex structure to begin with. The VAV terminal components consist of a controller that is mounted to the actuator which in turn controls and rotates the shaft. The shaft is then connected to the damper. To achieve thermal comfort, the central air handling unit will manipulate the supply air flow rate to each zone. This is done by placing the VAV terminals at different zones with a thermostat in each of the zones to detect the temperature within that zone. Hence, when the thermoset within the system senses that the zone is too hot, it will open the damper, allowing more cold air to flow. Likewise, when it senses that the temperature in the zone is cold, it will reduce the opening of the damper. Thus, reducing the airflow in that channel [8].

## IV. OUR APPROACH

This section covers the architecture, hardware components and software components used as well as data collection. It also includes the method and logic behind detecting activities and number of people within the zone. The use of mobile activity recognition API is also explained. Detailing design data Structure for Activities and Number of Pax, concurrency implementation, machine learning, Node-Red integration, central monitoring system, the implementation and future work will be included as well.

Figure 1 below illustrates the overall architecture of the system. The Interior_RPi1 will be getting and processing the sensor data from AHT10, LDR & Smartphone at every 30 minutes interval. Processed data will then be sent via Message Queuing Telemetry Transport (MQTT) to Node-RED. Similarly, Exterior_RPi2 will also be getting and processing AHT10 data at every 30minutes interval, and then send the data via MQTT to Node-RED. The Node-RED will pass data to the machine learning (ML) model and predict the state of the target before inserting it into the database - MySQL. Users can also access the web portal - central monitoring system to view the state of the damper and its respective sensor information. Data is needed to derive to train the machine learning models and for prediction purposes. To measure the interior and exterior values, 2 microcontrollers will be utilized, whereby 1 is to be placed inside an aircon environment, while the other microcontroller would be placed in a non-aircon environment. Table 1 below will discuss the various hardware components used and a brief explanation on why the components are selected for this project.

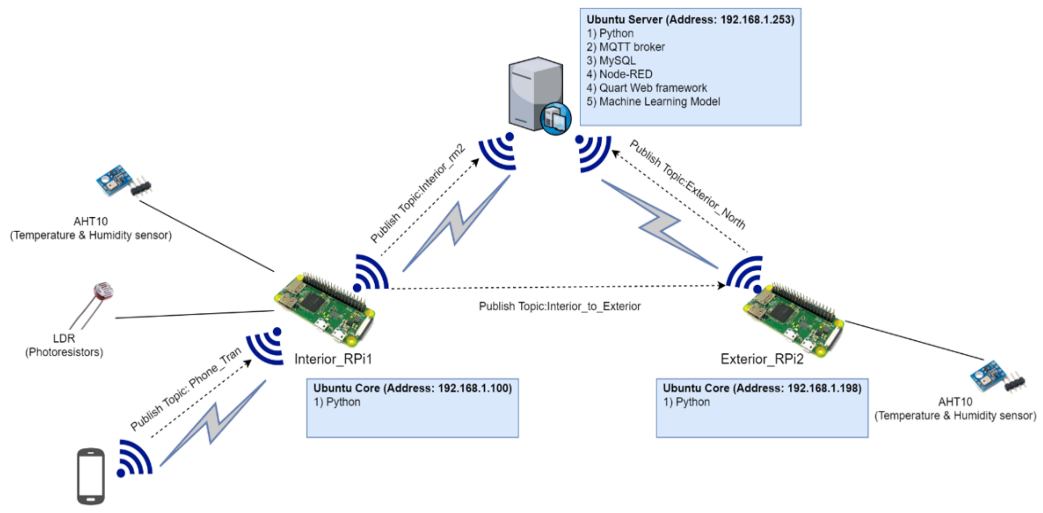

Ubuntu Server (Address: 192.168.1.253)
1) Python
2) MQTT broker
3) MySQL
4) Node-RED
4) Quart Web framework
5) Machine Learning Model
Publish Topic:Interior_rm2
Publish Topic:Exterior_North
AHT10
(Temperature & Humidity sensor)
LDR
(Photoresistors)
Publish Topic:Interior_to_Exterior
Interior_RPi1
Exterior_RPi2
Publish Topic: Phone_Tran
Ubuntu Core (Address: 192.168.1.100)
1) Python
Ubuntu Core (Address: 192.168.1.198)
1) Python
AHT10
(Temperature & Humidity sensor)

Figure 1 Overall Hardware Architecture

Table 1 Hardware Component Requirements

| *Component* | *Purpose* |
|---|---|
| *Raspberry pi* | Process respective components information and transmit to the server for actions needed and reporting via MQTT |
| *AHT10* | Measure the temperature and humidity |
| *LDR* | Measure brightness of the zone |
| *Acceleration sensor (Mobile)* | Measures the acceleration on all three physical axes (x, y, and z) |
| *Gravity sensor (Mobile)* | Measures the force of gravity on all three physical axes (x, y, z) |
| *Gyroscope (Mobile)* | Measures a device's rate of rotation around each of the three physical axes (x, y, and z) |
| *Linear acceleration sensor (Mobile)* | Measures the acceleration force by monitoring acceleration along a single axis |
| *Geomagnetic field sensor (Mobile)* | Measures the ambient geomagnetic field and creating a compass |

Figure 2 below illustrates the connections between the respective sensor components Interior_RPi1.

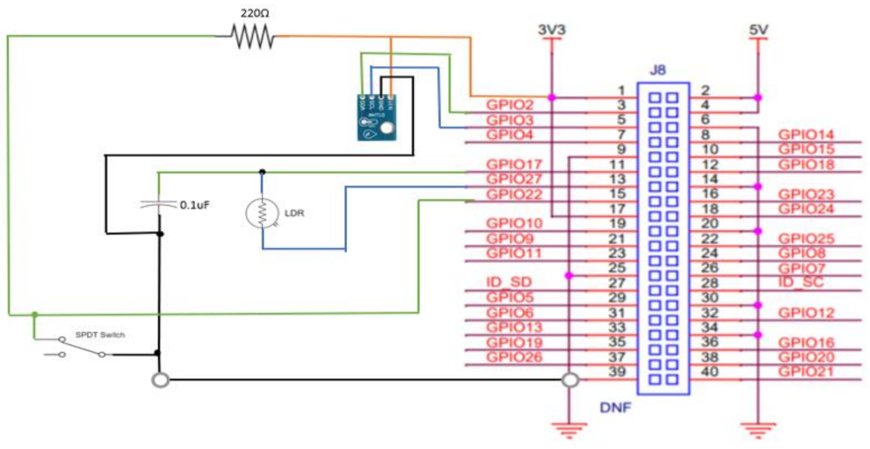


Figure 2 Wiring diagram for Interior_RPi1

Figure 3 illustrates the connections between the respective sensor component to the for Exterior_RPi2.

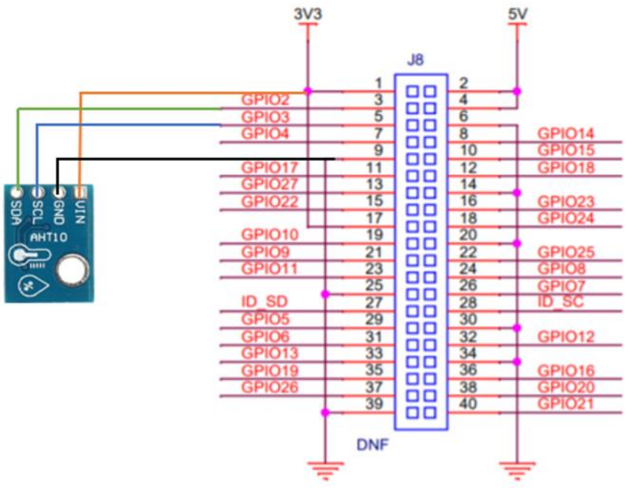


Figure 3 Wiring diagram for Exterior_RPi2

Ubuntu Server provides a package manager that enables ease of installation of python, MQTT broker, MySQL, Node-RED, Quart Web framework, Machine learning Model, Static IP Address. Trending Data is essential for the use of predicting the state of the damper within the HVAC system. In this project, data is collected internally. This is taken into consideration that each company will have its own use workflows and requirements. Hence, data requirements for each company will be different. For this prototyping, data collection is done by collecting at least 300 rows of data from both the interior and exterior microcontrollers. By default, data from both microcontrollers should be sent to the server at intervals of every 30 minutes. Thus, updating the system of the current data readings. A toggle switch will be introduced at the data collection phase to allow users to tell the system when they want to turn it on or off. Hence, apart from every 30 minutes interval where data is transmitted automatically to the server, data could also be transmitted to the server when the switch is toggled. Table 2 below illustrates the behavior of the switch. Only when there is a change in the switch's state, then it will send the data to the server. The latest values will be sent to the server.

Table 2 Behavior of Switch

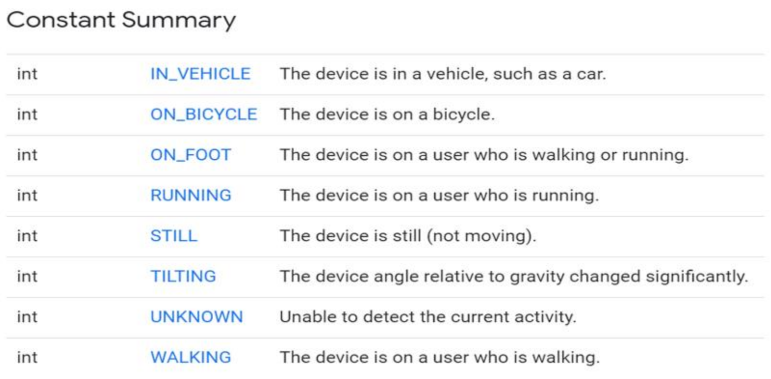
| *Previous state* | *Present state* | Output/ Target |
|---|---|---|
| *0* | 0 | Do nothing |
| *0* | 1 | 1 |
| *1* | 0 | 0 |
| *1* | 1 | Do nothing |

Table 3 describes the data that is sent across different time intervals together with the toggle of the switch at a certain time. It will explain the different scenarios with the target output.

Table 3 Target output with different scenarios

| *Time (minutes)* | *Is the switch toggle?* | *Data Send?* | Target | Reason |
|---|---|---|---|---|
| *5* | *0* | No | Assuming target is ‘0’ when it first started | No data will be sent over to the server because the first 30 minutes is not up, and the switch is not toggle |
| *30* | *0* | Yes | 0 | Switch is not toggle. But data will be sent to the server every 30 minutes. Hence the previous state “0" will be sent over to the server. |
| *50* | *1* | Yes | 1 | Switch is toggle; hence target is change from “0” to “1” |
| *60* | *0* | Yes | 1 | Switch is not toggle, but the next 30 minutes is up, hence the previous state “1” will be sent over to the server. |
| *75* | *1* | Yes | 0 | Switch is toggle, hence previous state “1” is change to “0” |

The switch behaved like a remote control. It is only when someone feels hot or cold, then they will toggle the switch to ON/OFF. The reason why the switch is designed in this manner is because by default, no one will keep adjusting the air-con temperature, unless there is a need to change the temperature. Hence in this project, if there is no one toggling the switch, we assume that the current temperature is acceptable and therefore, allow that temperature reading to be sent to the server every 30 minutes. However, if the switch is toggled, then the new reading will be sent to the server. The ultimate outcome is for the system to predict when to turn on and off the damper automatically. Hence, the switch will be deactivated once the data is sufficient to be trained and modeled by machine learning. In this case, 300 sets of data are the goal to achieve. Therefore, once 300 sets of data are received from both microcontrollers, the data will be used for training and modeling. This section covers the approach in determining the activities and number of people by adopting the android activity recognition API as part of the solution integration. For detecting activities type, android Activity Recognition API [5] is utilized. This API is developed and maintained by Android. Hence the source code can be referenced from Android repository [6]. By calling the API, it allows us to return the predicted activity values. The android Activity Recognition API consists of the following activities as shown in Figure 4.



Constant Summary

| | | |
|---|---|---|
| int | IN_VEHICLE | The device is in a vehicle, such as a car. |
| int | ON_BICYCLE | The device is on a bicycle. |
| int | ON_FOOT | The device is on a user who is walking or running. |
| int | RUNNING | The device is on a user who is running. |
| int | STILL | The device is still (not moving). |
| int | TILTING | The device angle relative to gravity changed significantly. |
| int | UNKNOWN | Unable to detect the current activity. |
| int | WALKING | The device is on a user who is walking. |

Figure 4: Summary of activities in Activity_Recognition API

As observed in Figure 4, not all the activities are relevant for the project. This is because only the activities in the room are required. Hence, activities such as IN_VEHICLE, ON_BICYCLE is reductant and can be removed. Hence, the source code will be modified to fit the project requirement. Figure 5 below shows the wireframe on how the smartphone interface will look like after modification. In this illustration, the 4 most common room's activities are taken into consideration. They are “Still”, “Slow Walk”, “Fast Walk” and “Fast Walk".

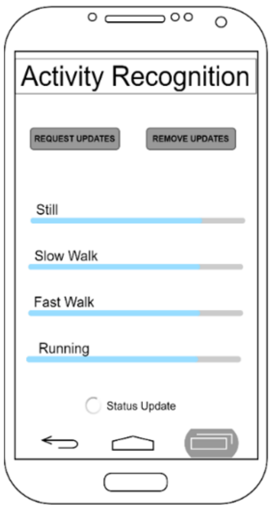


Figure 5: Wireframe (Smartphone)

To determine the number of people or (pax), a counter can be programmed to detect the number of smartphone devices that are connected to the system. Each Uniquely smartphone is identified by utilizing the concatenate of the smartphone's specifications, e.g., mobile brand, display, and board dimension etc. Hence, when a unique ID (UID) is detected, it will increase the count. Likewise, when the UID is deactivated, the count will decrease. One UID defines one person being identified. Therefore, five UID would determine that 5 people are being identified. To ensure that the number of counts holds within the time frame, count will be reset once it is sent to the server. The activity type and the UID are published by the MQTT from the mobile phone. Then the microcontroller's subscriber will interpret the data accordingly. While the above example is good to determine the type of activity for each person, it does not demonstrate what most people are doing in the room. For example, it is unfair to assume that everyone in the room is running just because one person is running. Hence, in the following example, the logic is designed based on 2 most likelihood scenarios to get the aggregate number of people and type of activity. Figure 6 below illustrate scenario 1 which suggests that the activity with the highest people(pax) count shall become the final activity.

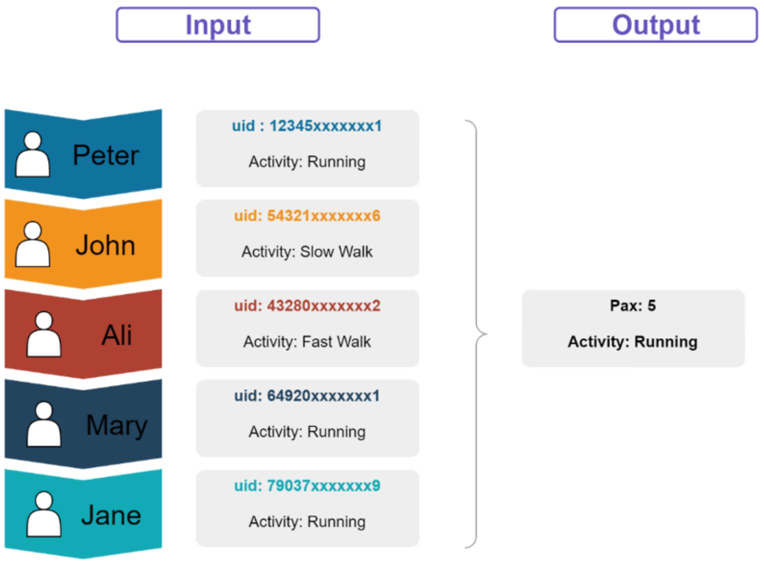


Figure 6: Logic design for pax and activity (scenario 1)

Figure 6 above illustrates that there is a total of 5 people because we have 5 different unique IDs (UID). There are 3 pax running, 1 pax is detected as "slow walk", another 1 pax is detected as "fast walk". Since the running has the highest count of 3, this shows that most of the people in the room are running. Hence, "5 Pax" and "Running" will be published from raspberry pi - Interior_RPi1 to Node-RED via the MQTT. Scenario 2 suggests taking the activity that has the higher intensity should there be any activities that have an equal count. The level of intensity for each activity is illustrate in Figure 7 below:

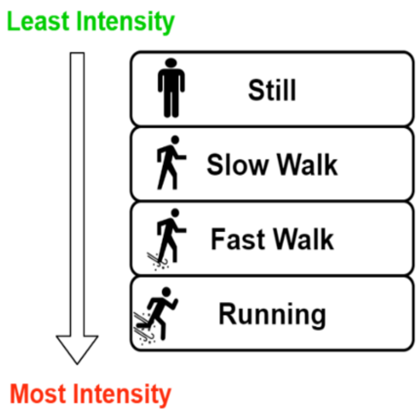


Figure 7: Illustration of intensity level

Figure 8 illustrates that there is a total of 6 people because there are 6 different unique id (UID). The activities detected are: 3 pax "slow walk" and 3 pax "fast walk". Since both activities have an equal count of 3. Therefore, logic will select the activity that has the highest intensity between both activities. In this case, "Fast Walk" has a higher intensity compared to slow walk. Hence, "6 Pax" and "Fast Walk" will be sent to the raspberry pi - Interior_RPi1 via the MQTT broker.

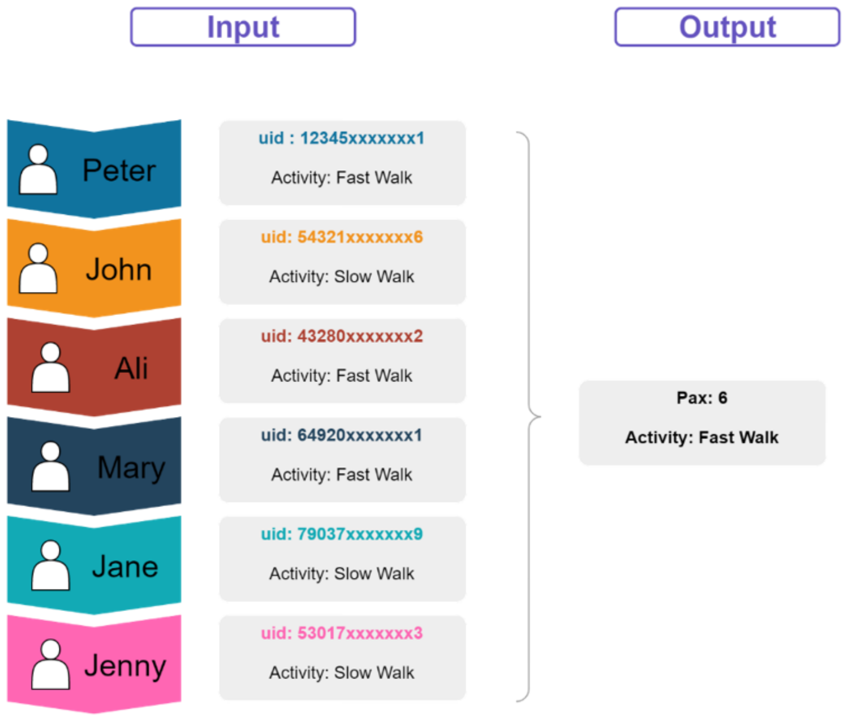


Figure 8: Logic design for pax and activity (scenario 2)

Table 4 Design Structure for Activities and Number of Pax

| **Step** | **Description** | **Design Structure** |
|---|---|---|
| 1 | By default, the dataset should be empty as there is no activity detected yet. Once the *request update* is enabled, it detects the new incoming data and will update the activity accordingly. | dataset = {uid: activity} |
| 2 | Based on the list of dataset, the activity count will increase accordingly.<br><br>By default, the activity count is set to 0. | Activity = {still: 0,<br>slow walk :0,<br>fast walk :0,<br>running: 0} |
| 3 | The UID and activity will be added to the dataset if there are people and activity detected from the smartphone.<br><br>The dataset on the right illustrate how it will look like after UID and activities are added. | dataset = {12345xxxxxxx1 : Fast Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk<br>} |
| 4 | Check if there are any similar UID.<br>If there are similar UID, that means that the same person is still in the room.<br><br>Thereafter, check what is the activity that the person is currently doing. Lastly, update the current activity with the previous activity. | <u>For example:</u><br><br>Previous received data:<br>dataset = {12345xxxxxxx1 : Fast Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk<br>}<br>New incoming data:<br>dataset = {12345xxxxxxx1 : Slow Walk)<br><br>Since UID 12345xxxxxxx1 is already in the dataset, hence for that UID that already exist, update current activity "Slow Walk" with previous "Fast Walk".<br><br>Therefore the final will be:<br>dataset = {**12345xxxxxxx1 : Slow Walk**,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk, |

| | | |
|---|---|---|
| | | 79037xxxxxxx9 : Slow Walk<br>} |
| 5 | Check if there are any new UID. If a new UID is detected, it means there is one more people entering the room. | For example:<br><br>Previous received data:<br>dataset = {12345xxxxxxx1 : Slow Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk<br>}<br><br>New incoming data:<br>dataset = {82549xxxxxxx5 : Running)<br><br>Hence add in new entry of the below into dataset:<br>82549xxxxxxx5 : Running<br><br>Therefor the final will be:<br>dataset = {12345xxxxxxx1 : Slow Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk,<br>**82549xxxxxxx5 : Running**<br>} |
| 6 | Check if anyone has left the room. Remove the person by pop the UID and activity out of the dataset. | dataset.pop(uid, None)<br><br>For example:<br>This person has left the room<br>82549xxxxxxx5 : Running<br><br>Hence the final will be:<br>dataset = {12345xxxxxxx1 : Slow Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk,<br>**82549xxxxxxx5 : Running**<br><br>} |
| 7 | Checking the number of people in the room by counting the rows in the dataset.<br><br>Since each UID uniquely identify each person, thus the number of UID detected will be equal to the number of people detected.<br><br>Therefore, as there are 5 rows of UID, so 5 pax is detected | Final list of the dataset:<br><br>dataset = {12345xxxxxxx1 : Slow Walk,<br>54321xxxxxxx6 : Slow Walk,<br>43280xxxxxxx2 : Fast Walk,<br>64920xxxxxxx1 : Fast Walk,<br>79037xxxxxxx9 : Slow Walk,<br>} |
| 8 | Upon looking at the "Activity" on the right.<br>We can see that 3 people are identified with "Slow Walk" and 2 people are identified with "Fast Walk".<br><br>Since, "Slow Walk" has the highest count of 3, hence "Slow Walk" will be sending to the Interior_RPi1 via the MQTT broker. | Activity = {Still: 0,<br>Slow Walk :3,<br>Fast Walk :2,<br>Running: 0} |
| 9 | After the data is sent to the server, the "Activity" count will reset back to 0 to prepare for another set of new incoming data. | Activity = {Still: 0,<br>Slow Walk :0,<br>fast Walk :0,<br>Running: 0} |

By default, coding is using a top-down approach. As shown in Figure 9 below, based on this logic, assuming the program is in the 30 minutes interval, even if the switch is being toggled, it will still have to wait for the 30 minutes to complete before it can be activated. Hence, it is not very practical.

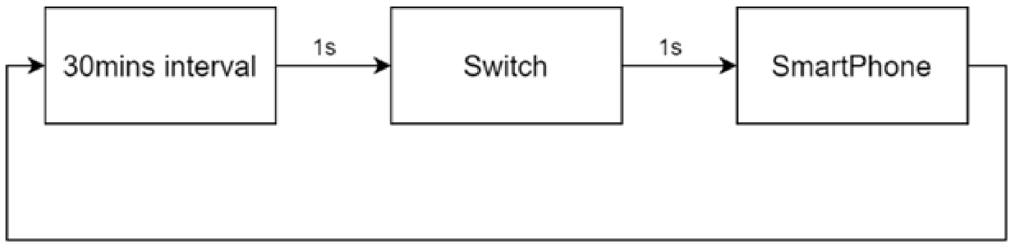


Figure 9 1st design without concurrency

The program is re-designed to allow separate blocks to run independently as described in Figure 10 below. For example, the smartphone activity data will be received and updated at the Interior_RPi1 as and when the data is received. This is all done simultaneously together with all threads. Within 30 minutes, the latest smartphone activity, together with the other sensors data will be sent over to the server. This shows that threading is more efficient as it reduces the processing time by performing concurrency.

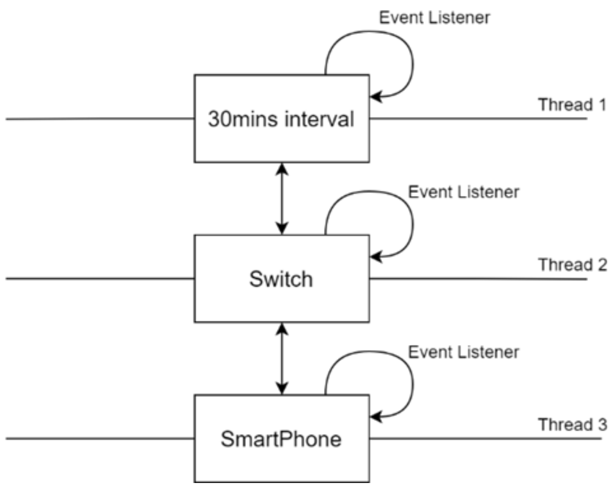


Figure 10 Improved design with concurrency

## V. MACHINE LEARNING

Machine Learning is the study of computer algorithms that improve automatically through experience and using data [9]. It is seen as a part of artificial intelligence. Machine learning identifies patterns and makes decisions with minimal human intervention. There are many machine learning algorithms, and one of them is Random Forest. Random Forest is a very popular machine learning algorithm that is often used either as a classification model or a regression model. For classification models, "Bagging Technique" is used. Random Forest is the machine learning algorithm that is used for this solution implementation as it has been tested and proven to perform well when integrating into the HVAC system [7]. Likewise, during the testing, it is one of the best performing models. Please refer to Table 5 below.

Table 5 Machine Learning Model Results

| *Machine Learning Model* | Result |
|---|---|
| *Vector Classifier* | 58.67% |
| *Logistic Regression* | 96.00% |
| *K Map* | 76.00% |
| *Decision Tree* | 96.00% |
| *Random Forest* | 97.33% |

Figure 11 illustrates the nodes are interconnected together when performing machine learning. It gets its data after subscribing via the MQTT nodes with the topic "Interior_rm2" and "Exterior_North". A join-wait node is used to wait for both data from Interior_rm2 and Exterior_North to arrive. It will then pass it to the machine learning node (python3 ml/processing.py) before inserting them into the database.

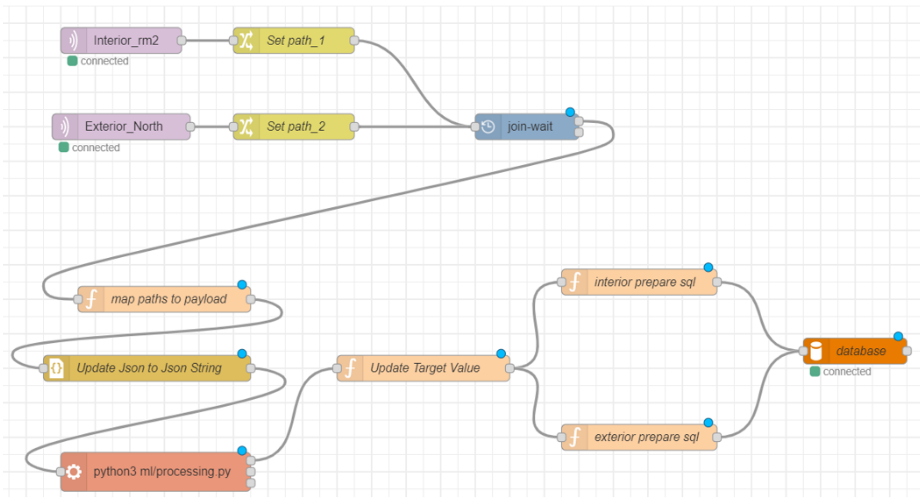
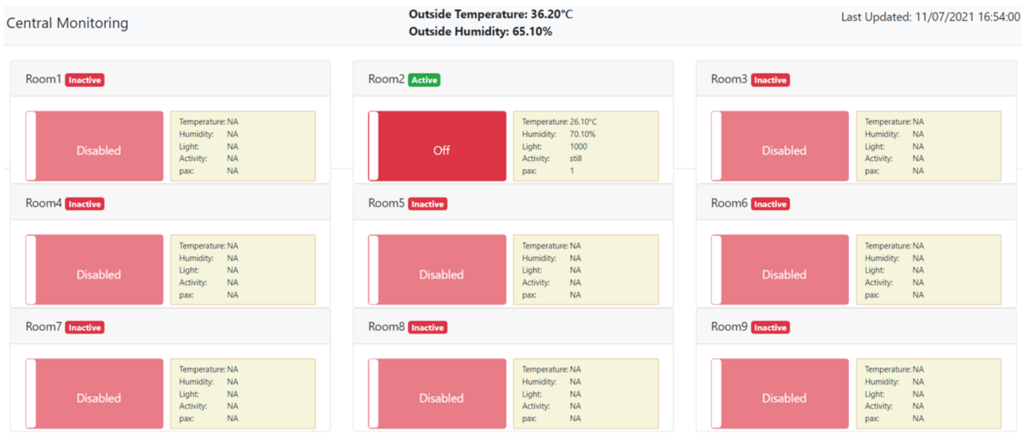


Figure 11 Node-RED connection for Machine Learning

A web portal will be set up to allow the end user to monitor the state of the damper as well as display the current values of the Interior_RPi1 and Exterior_RPi2 sensors. This web portal will be called "central monitoring system". This is because we can monitor all the essential data all in one place. The central monitoring system is illustrated in Figure 12 below.



Central Monitoring

**Outside Temperature: 36.20°C**
**Outside Humidity: 65.10%**

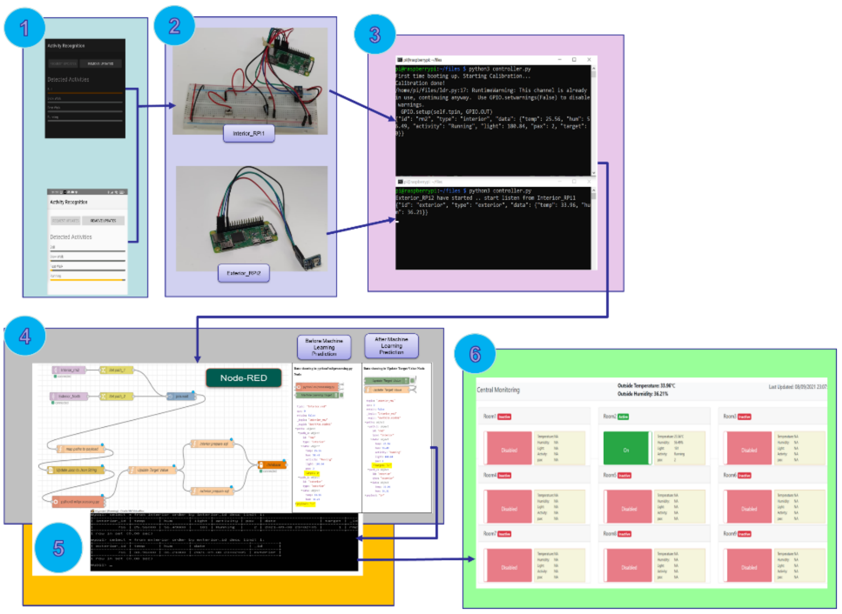
Last Updated: 11/07/2021 16:54:00

| Room | Status | Damper | Temperature | Humidity | Light | Activity | pax |
|---|---|---|---|---|---|---|---|
| Room1 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room2 | Active | Off | 26.10°C | 70.10% | 1000 | still | 1 |
| Room3 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room4 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room5 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room6 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room7 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room8 | Inactive | Disabled | NA | NA | NA | NA | NA |
| Room9 | Inactive | Disabled | NA | NA | NA | NA | NA |

Figure 12 Illustrating interface of the web interface – Central Monitoring System

## VI. IMPLEMENTATION OF PROPOSED WORK

Figure 13 illustrates multiple smartphones are being able to access the network and enter the system concurrently. It also illustrates the wiring for both Interior_RPi1 and Exterior_RPi2. Activity data from smartphones will be published to both Interior_RPi1 and Exterior_RPi2 respectively. AHT10 and LDR will also publish the environment data to Interior_RPi1. Figure 13 also illustrates the data values that are received on both Interior_RPi1 and Exterior_RPi2 and how the nodes are wired up together and that the data are successfully published to the MQTT broker and subscribed to by Node-RED. The data is then processed and predicted by the machine learning node before inserting the incoming data and its predicted damper states into the database. Furthermore, it shows that the data has been successfully inserted into the MySQL database and that the Quart Web Frame has successfully selected the latest sensor's data and target value from the database. Lastly the values are displayed on the Central Monitoring System.



```
pi@raspberrypi:~/files $ python3 controller.py
First time booting up. Starting Calibration...
Calibration done!
/home/pi/files/ldr.py:17: RuntimeWarning: This channel is already
in use, continuing anyway.  Use GPIO.setwarnings(False) to disable
 warnings.
  GPIO.setup(self.tpin, GPIO.OUT)
{"id": "rm2", "type": "interior", "data": {"temp": 25.56, "hum": 5
6.49, "activity": "Running", "light": 100.84, "pax": 2, "target":
0}}
```

```
pi@raspberrypi:~/files $ python3 controller.py
Exterior_RPi2 have started .. start listen from Interior_RPi1
{"id": "exterior", "type": "exterior", "data": {"temp": 33.96, "hu
m": 36.21}}
```

Figure 13 Implementation of Proposed work

Figure 14 illustrates the top-down integration testing, which tests the end-to-end modules using top-down approach.

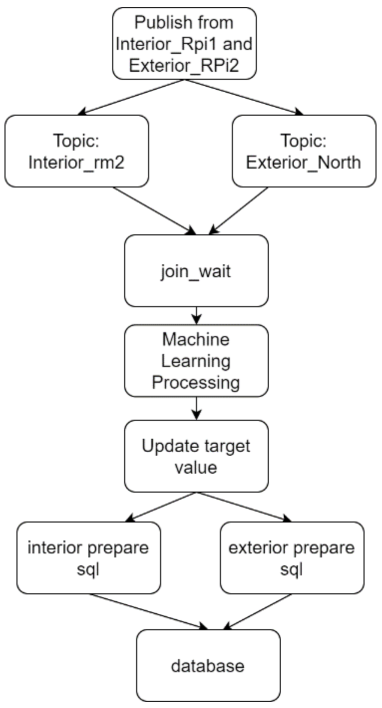


Figure 14 Illustration of top-down integration testing

Figure 15 and 16 illustrate the testing implementation from Interior_RPi1 and Exterior_RPi2 publisher to Node-RED subscriber via the topic “Interior_rm2” and “Exterior_North” respectively.

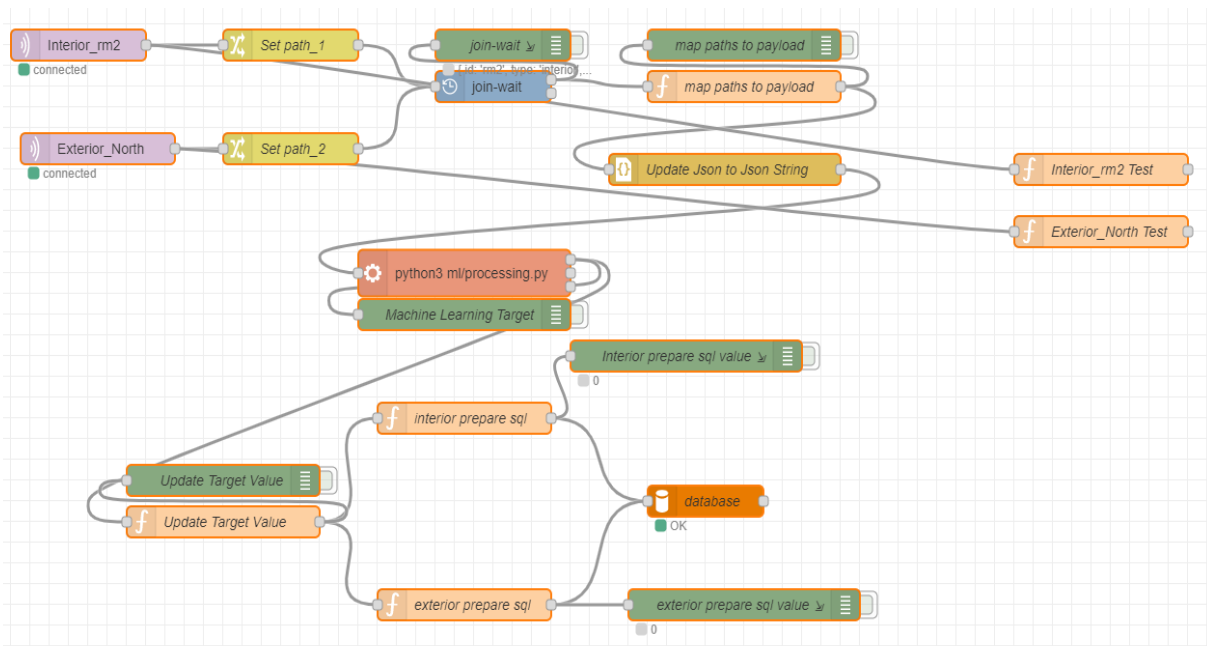
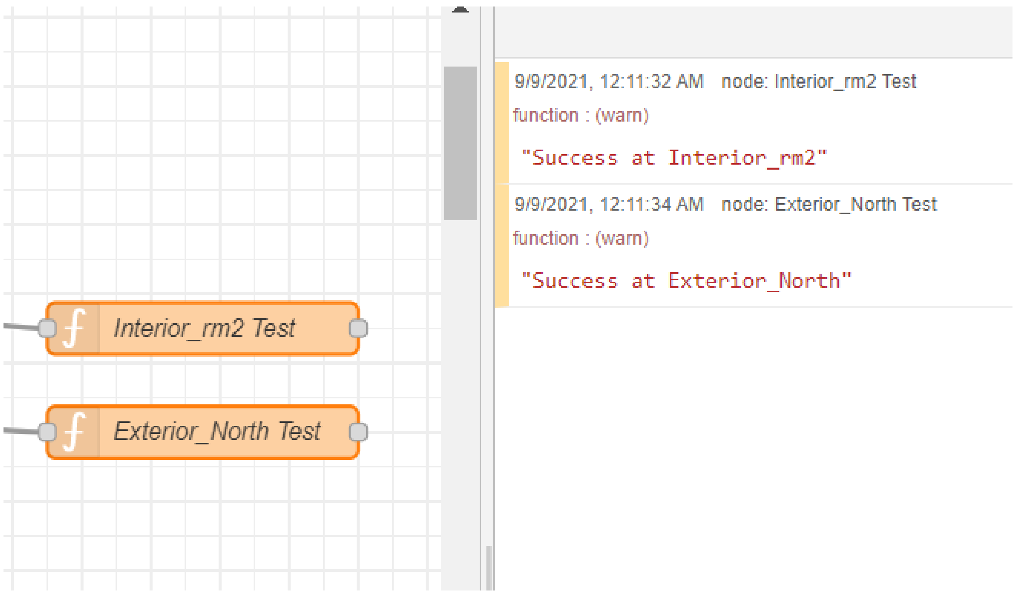


Figure 15 Integration Testing (Part 1)



```
9/9/2021, 12:11:32 AM   node: Interior_rm2 Test
function : (warn)
"Success at Interior_rm2"
9/9/2021, 12:11:34 AM   node: Exterior_North Test
function : (warn)
"Success at Exterior_North"
```

Figure 16 Integration Testing Results (Part 1)

Figure 17 and 18 illustrate the testing Implementation from “Interior_rm2” and “Exterior_North” node to ensure that “join-wait” node receive the data. Thereafter, “join-wait” and “machine learning (ML) processing” node is tested to ensure that the target value is return. The next end to end point between “machine learning (ML) processing” and “update target value” node ensures that the targeted value is updated accordingly. Then, “update target value” and “interior prepare sql” are tested to ensure that the updated data are formatted into the interior SQL statement. Likewise, for the “exterior prepare sql” node. Finally, “database” is tested to ensure the SQL statement is inserted successfully into the database.

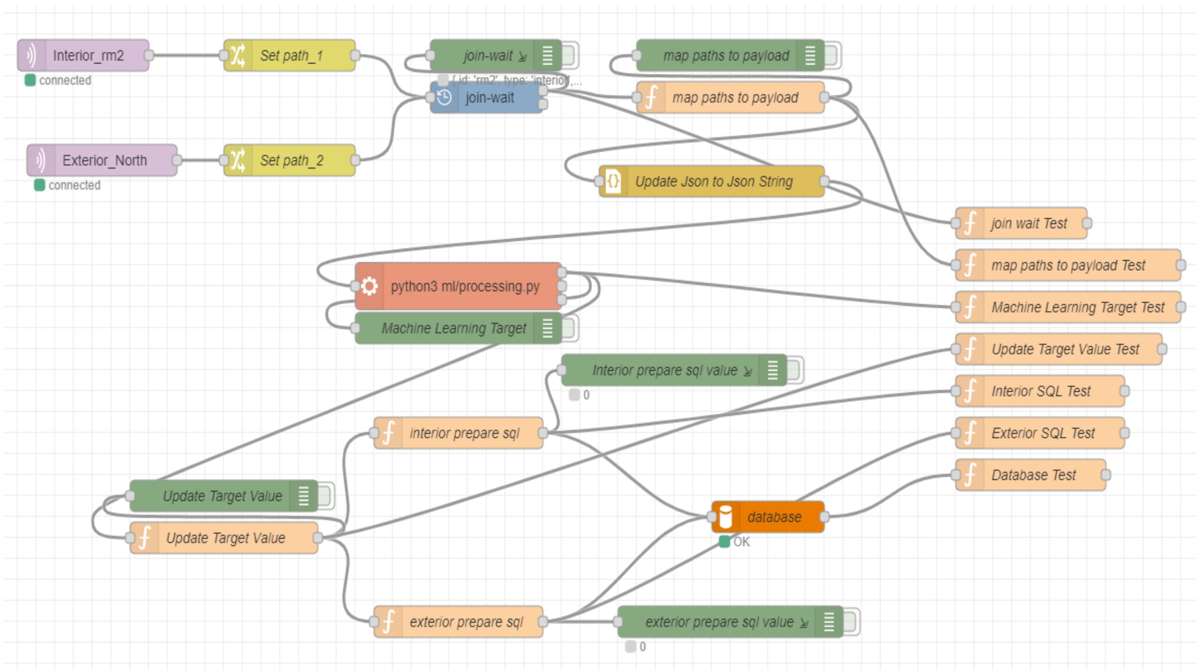
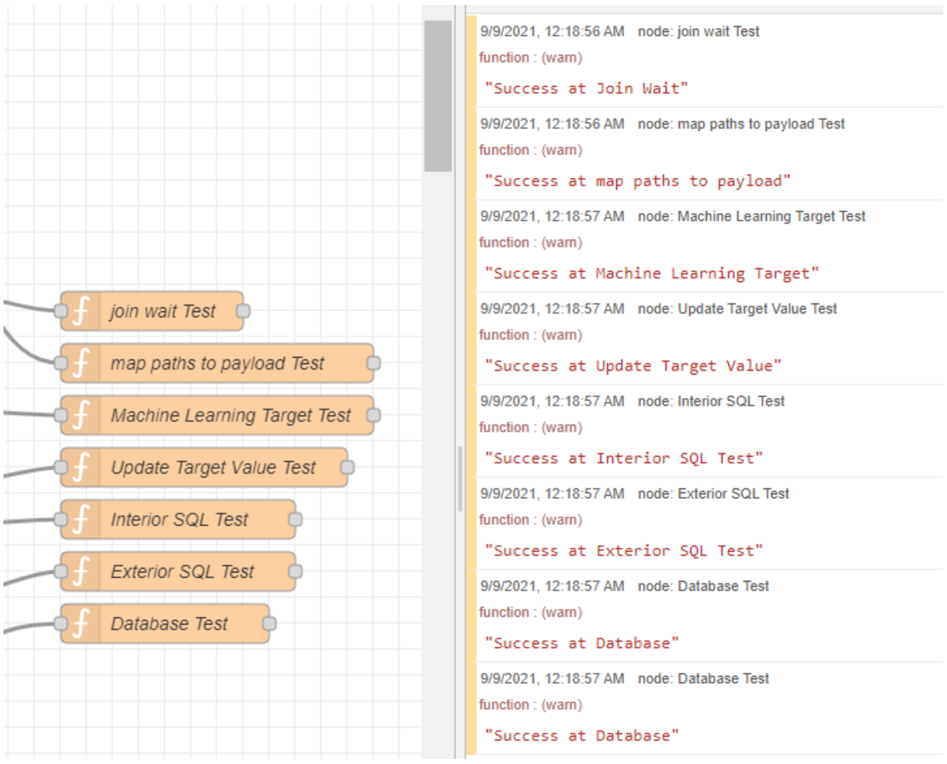


Figure 17 Integration Testing (Part 2)



```
9/9/2021, 12:18:56 AM   node: join wait Test
function : (warn)
"Success at Join Wait"
9/9/2021, 12:18:56 AM   node: map paths to payload Test
function : (warn)
"Success at map paths to payload"
9/9/2021, 12:18:57 AM   node: Machine Learning Target Test
function : (warn)
"Success at Machine Learning Target"
9/9/2021, 12:18:57 AM   node: Update Target Value Test
function : (warn)
"Success at Update Target Value"
9/9/2021, 12:18:57 AM   node: Interior SQL Test
function : (warn)
"Success at Interior SQL Test"
9/9/2021, 12:18:57 AM   node: Exterior SQL Test
function : (warn)
"Success at Exterior SQL Test"
9/9/2021, 12:18:57 AM   node: Database Test
function : (warn)
"Success at Database"
9/9/2021, 12:18:57 AM   node: Database Test
function : (warn)
"Success at Database"
```

Figure 18 Integration Testing Results (Part 2)

## VII. CONCLUSION

Table 5 illustrates the tasks that I have completed. As can be seen below, I have completed all of the functional requirements under the "Must Have" and "Should Have" category. Hence, the overall project objective that is set from the start has been met. On top of that, two of the tasks from "Should Have" are also done as an add-on improvement to the overall project.

Table 5 Conclusion recap of functional requirements and completed tasks.

| Functional Requirements | | |
|---|---|---|
| **S/N** | **"Must Have"** | **Completed?** |
| 1 | I want to capture the activity so that it can help to predict the state of the damper. | ☑ |
| 2 | I want to count the number of people in the room so that it can help to predict the state of the damper. | ☑ |
| 5 | I want to have a MQTT so that my devices can communicate among each other. | ☑ |
| 6 | I want to have the temperature and humidity sensor so I can have the environment data. | ☑ |
| 7 | I want to have a LDR so that it can help to predict the state of the damper. | ☑ |
| 11 | I want to design concurrency so that I can run the processes in parallel. Hence, will shorten the processing time. | ☑ |
| 12 | I want to do machine learning (ML) so that I can let the machine decide the state of the damper. | ☑ |
| | **"Should Have"** | |
| 4 | I want to have my own external RPi to collect the external temperatures so that the data collected will be more accurate. | ☑ |
| 8 | I want to setup a database so that I can Create, Read Update, Delete (CRUD) data. | ☑ |
| 9 | I want to have Node-RED so that I can easily individual devices within the IoT architecture. | ☑ |
| 13 | I want to perform data collection so that I can perform machine learning (ML). | ☑ |
| 14 | I want to add a switch so that users are able tell us what they are feeling by toggling the switch. | ☑ |

## REFERENCE

[1]. Calm, James. (2002). Emissions and environmental impacts from air-conditioning and refrigeration systems. International Journal of Refrigeration. 25. 293-305. 10.1016/S0140-7007(01)00067-6.

[2]. THE ECONOMIC TIMES. 2019. Air conditioning is the world's next big threat. [ONLINE] Available at: https://economictimes.indiatimes.com/news/science/air-conditioning-is-the-worlds-next-big-threat/articleshow/69999842.cms?from=mdr.

[3]. A Climate Institute. 2018. The Trouble with Air Conditioning. [ONLINE] Available at: https://climate.org/cooling-your-home-but-warming-the-planet-how-we-can-stop-air-conditioning-from-worsening-climate-change/#:~:text=The%20Trouble%20with%20Air%20Conditioning&text=As%20our%20planet%20warms%2C%20the,conditioning%20and%20refrigeration)%20will%20rise.&text=HFCs%20are%20a%20much%20more,the%20disposal%20of%20old%20units.

[4]. Jiang, Hongmei & Li, Zhanming & Tang, Weiqiang. (2019). Research on Optimized Energy Saving Control System of VAV Air Conditioning System. IOP Conference Series: Earth and Environmental Science. 237. 042015. 10.1088/1755-1315/237/4/042015.

[5]. developer android. 2019. Detect when users start or end an activity.[ONLINE] Available at: https://developer.android.com/guide/topics/location/transitions.

[6]. github. ActivityRecognition. [ONLINE] Available at: https://github.com/android/location-samples/tree/main/ActivityRecognition.

[7]. SpringerLink. 2020. A comparison of machine learning algorithms for forecasting indoor temperature in smart buildings. [ONLINE] Available at: https://link.springer.com/article/10.1007/s12667-020-00376-x.

[8]. Seyam, Shaimaa. (2018). Types of HVAC Systems. 10.5772/intechopen.78942.

[9]. Machine Learning Definition | Tom M. Mitchell| McGraw-Hill Science/Engineering/Math; (March 1, 1997), Page 1 | http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html

Chiang Liang Kok, et. al. "A Green IoT Solution Coupled with Machine Learning to Tackle Climate Change." *The International Journal of Engineering and Science (IJES)*, 12(12), (2023): pp. 01-16.