POLITECNICO DI MILANO

Department of Electronics, Information and Bioengineering

Master’s Degree in Automation and Control Engineering

DATA ACQUISITION AND ANALYSIS OF ENERGY EFFICIENCY IN A CLASSROOM BUILDING

The thesis presented by:

Arvind Jayakumar, 877297

Supervisor:
Prof. Luca Ferrarini

Correlator:
Ing. Riccardo Babini

2018/2019

PAGE INTENTIONALLY LEFT BLANK

Table of Contents

LIST OF FIGURES

LIST OF TABLES

RINGRAZIAMENTI

First, I like to thank Professor Luca Ferrarini, Ing. Riccardo Babini and the Daisy lab team for the collaboration and constant support and made this thesis possible.

Next, I extend my gratitude and special thanks to my family members, especially to my uncle Dr.Siva Kumar without his support, I could not be able to dream of studying in this esteemed university.

Nextly, I thank all my friends Abhishekh, Abilash, Harrish, Suganth, Abhishekh Jr, Seenu, and my dear friend Giuseppe Incandela for supporting me always.

SOMMARIO

La tesi inizialmente fornisce un confronto dettagliato tra le due piattaforme Internet of Things; sono ThingsBoard e IBM Node-RED e quindi, con l'aiuto della piattaforma IoT Node-RED, viene presentato un modello prototipo per un'aula per l'acquisizione di dati dai valori dei vari dispositivi del sensore dal database di time-series InfluxDB.

In base ai valori di temperatura e umidità relativa, sono stati calcolati gli indici di comfort. Inoltre, la scala del comfort e il valore di ogni indice vengono visualizzati nella dashboard dell'utente. Inoltre, utilizzando questi indici, possiamo comprendere la condizione e la qualità della classe e come robusto il sistema di controllo per il sistema HVAC funziona.

Quindi i dati raccolti vengono analizzati e utilizzati per lo sviluppo di applicazioni come il servizio di bidello, l'occupazione corrente in una classe, vengono sviluppati la media di alcuni parametri fisici come temperatura, umidità e CO2 per un pavimento.

Infine, viene presentata l'estensione della piattaforma mobile e il modo in cui possiamo utilizzarle in modo efficiente.

ABSTRACT

The thesis initially gives a detailed comparison between the two Internet of Things platforms; they are ThingsBoard, and IBM Node-RED and then, with the aid of the Node-RED IoT platform, a prototype model for a classroom for acquiring data from the various sensor device's values form the InfluxDB time-series database is presented.

Based on the temperature and relative humidity values, the comfort indices were calculated. Moreover, the scale of the comfort and the value of each index is displayed in the user's dashboard. Furthermore, utilizing these indices, we can understand the condition and quality of the classroom and how robust the control system for the HVAC system is working.

Then the collected data are analyzed and utilized for developing applications such as janitor service, current occupancy in a classroom, Average of some physical parameters such as temperature, humidity, and CO_2 for a floor are developed.

Finally, the extension to the mobile platform and how we can efficiently utilize them are presented.

LIST OF ABBREVIATIONS

API	Application Program Interfaces
cmd	Command
CoAP	Constrained Application Protocol
CSS	Cascading Style Sheets
DB	Database
e	Vapour pressure
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identification
IoT	Internet of things
IP	Internet Protocol
IT	Information Technology
JS	Java Script
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
msg	Message
NoSQL	Non-Structured Query Language
npm	Node Package Manager
RH	Relative Humidity
SQL	Structured Query Language
T	Temperature

PAGE INTENTIONALLY LEFT BLANK

1 INTRODUCTION

This chapter was dedicated to the reasoning behind this thesis work. In the first section 1.1, a state of art analysis carried out. Then, the central thesis objectives based on the instruction given by the professor are presented in the second section 1.2. Finally, the structure of the thesis document was described in section 1.3 to navigate through the report for the reader quickly.

The IoT is a system of interrelated computing devices, mechanical and digital machines, objects, people, or animals that are provided with a unique identifier and ability to transfer data over a network without requiring human-to-human or human-to-computer interactions. The thing in the IoT in here is the Building, the classrooms and the sensor devices in them, these devices can be assigned to an IP address, and data are transferred through the network through the assigned IP's through the server. The IoT will increase the ubiquity of the internet so we can connect our desktop or mobile devices to monitor the data, in extension, analysis them to control them.

1.1 State of the Art analysis

The first building ever constructed buildings were primitive shelters made from stones, sticks, animal skin, and other natural materials. While they hardly resembled the steel, glass, cement that make up our modern-day buildings such as home, schools, office, hospitals, same as the early day structures, the modern buildings have the same purpose of providing a comfortable space for the people who are inside.

If we look in deep-down, the present-day buildings are complex concatenations of structures, systems, and technology. Over time, each of the components inside a building has been developed and improved continuously, allowing the modern-day building owners to make lighting, security, heating, ventilation, and air conditioning systems to work independently based on the input conditions.

In today's era, energy matters a lot, and there is a necessity of using them in an optimized manner to have zero percent wastage. To have these people are beginning to look outside the four walls, and which allows them to consider the impact of the variation in building power usage on the electrical grid and also the global environment.

For this technological artwork to be possible, we are using the sensors and actuators (as our senses and body parts) along with the Internet thus all together becomes the IoTs, with all these things a building can communicate its state to the operators, occupants of the building, to the electrical grids, to the computer and also to the nearby buildings (if they are sharing the resources) a building system like this are known as the smart buildings. These intelligent buildings use IT during the operation to connect with a variety of subsystems, which typically operate independently, so this system can share information to optimize total building performance. In a nutshell, Smart buildings look beyond the building equipment within the four walls, which allows us to empower with new levels of visibility and actionable information.

1.2 Thesis objectives

Currently, many IoT software platforms provide solutions for the smart building, in which most of them offer both fully developed priced version and some of them gives open-source community software. Notion of this thesis is to find and study the open-source community IoT platform in its all possible dimension in order to discover the potential ability of that particular platform chosen and to compare with others in the market and to provide a comparison summary in the first half and in the second half to pick the best platform and develop data acquisition model through them. So that one who is using this thesis work can have a full overview of the studied platforms in its rudimentary level and how to use them for making a building "Smart."

As said above in the second phase of the thesis work, I have developed a full IoT model for the university building, the building taken contains four floors, and fifteen classrooms and each classroom has one Envisense sensor module which contains thirteen different sensors per module, in total one ninety-five sensor devices our notion is to obtain the data from InfluxDB time-series database, visualize them via "Dashboard" through the chosen IoT platform in computer and then I have also extended the study by developing full dashboard for the android mobile devices.

Finally, the data collected from the devices are analyzed. Based on the analysis we have developed small applications such as average floor temperature, humidity, carbon dioxide level, the current occupant level of a classroom, scheduling the cleaning work for a janitor through email notification and this will ease the work for the person who is monitoring the building premises and also for the peoples who are utilizing the building area.

1.3 Document Structure

Chapter 2 is fully dedicated to discussing overall ability, functionality, and basic requirements for using the platforms. In a nutshell, it provides a comparison with one other IoT platform. In chapter 3, we will be moving with the configuration of each sensor and its implementation in the IoT platform and its dashboard. Chapter 4 provides the configuration and deployment in the mobile device(i.e., android supported devices). Then in Chapter 5 will gather the results from Phase one and Phase two of the thesis and will draw a detailed conclusion.

In chapters 3 and 4 are fully dedicated to IoT development for the building in two different platforms, one is the deployment in the personal computer and the other in a mobile device. One who is reading chapter 3's sensor configuration section, one can find similarities this because they are configured in the same procedure. For security concerns, the report does not have any information about the credentials, server IP address, and device ID, and algorithms. However, the reader can get a general idea of how the works have been carried out.

In a nutshell, these procedures can be used for any building with small changes according to things used.

2 COMPARISON BETWEEN THINGSBOARD AND Node-RED

This chapter describes the entire phase one of the thesis work, which is to choose and study two IoT platforms and to give a detailed genal comparison between the platforms, i.e., ThingsBoard and Node-RED. Both platforms are opensource communities (i.e., for using the platform, no license is required). For the sake of argument, every paragraph will describe ThingsBoard first and then the Node-RED platform.

2.1 Introduction to ThingsBoard

The ThingsBoard is an IoT platform for data collection, processing, visualization, and device management. It enables device connectivity via industry standard IoT protocols such as MQTT, CoAP, and HTTP and supports both cloud and on-premises deployments. It combines scalability, fault-tolerance, and performance, so we will never lose our data. It provides provisions, monitors, and securely controls our IoT entities using rich server-side APIs. We can also define relations between our devices, assets (in our case, it is building, floors, and classroom) and costumers or any other entities. As said, we collect and store telemetry data in a scalable and fault-tolerant way, and we can visualize the data in the flexible dashboard, and this dashboard can also be shared with customers [1]. The collected data can also be processed with the aid of rule chins, which will transform and normalize our data.

The ThingsBoard supports various database options such as SQL, NoSQL, and Hybrid databases in its premium versions. Then allows us to choose where to store main entities and where to store telemetry data.

The ThingsBoard allows three levels of controls. The first level is the system administrator who has the ability create new tenants and manage widgets and servers, the second level is the tenant administrator, who can describe and configure everything in the ThingBoard and finally the customer user, this level can able monitor the dashboard which is created by the tenant administrator, In a nutshell, this level has minimal capacity. Moving further, ThingsBoard allows multitenancy so that with the aid of a single system administrator, we can manage multiple tenants.

2.2 Introduction to Node-RED

Node-RED is a programming tool for wiring together hardware devices, APIs, and online services [2]. It provides us with a browser-based editor that makes it easy to wire together flows using the full range of nodes in the palette that can be deployed in runtime in a single click. JS functions can be created within the editor window using a rich text editor. A built-in library function allows us to save services and flows for re-use. This entire program module was built on Node.js by taking full advantage of its event-driven, non-blocking-model. This makes it ideal for running at the edge of the network on low-cost hardware such as Raspberry pi as well as in the cloud.

The flows created in Node-RED are stored using the JSON program format, which can be easily transported for sharing with others if needed, and even we can publish the flow online so anyone can utilize our work. The MQTT nodes can properly configure TLS connections. The software is an open-source JS Foundation project which licenses under Apache License 2.0.

2.3 Open-source community / Maturity

In the majority of the IoT software available in the market are providing the opensource community with different functionalities and of courses, each one has pros and cons of its own. For using this service, we do not need to pay anything to the service provider. On the other hand, the product provider is under development (i.e., In Continuous testing stage), so it is not as stable as the paid versions. By using the opensource, we are not restricted to follow the manuals; we have many open doors to use utmost.

ThingsBoard opensource is named “Community Edition,” and the version provided is 2.4.1 [1], and there is no particular name for Node-RED opensource, and the software version provided currently is 1.0.0 [2].

2.4 Basic technical requirement

This paragraph describes the software skill set requirements for both platforms. The reader can find similarities in requirements; this is because both platform bases are the same, and then a detail description of the instrument requirements such as sensors, actuators, and so are presented in detail.

2.4.1 Software requirements

To get started with IoT should have some basic knowledge in programmings such as JAVA, JS, JSON, and SQL, and further, we should have some introduction to web page-based programming language scripts such as HTML and CSS. We can use the JS programming structure for both the software modules in stock, and a detailed explanation will be given in the program structure paragraph.

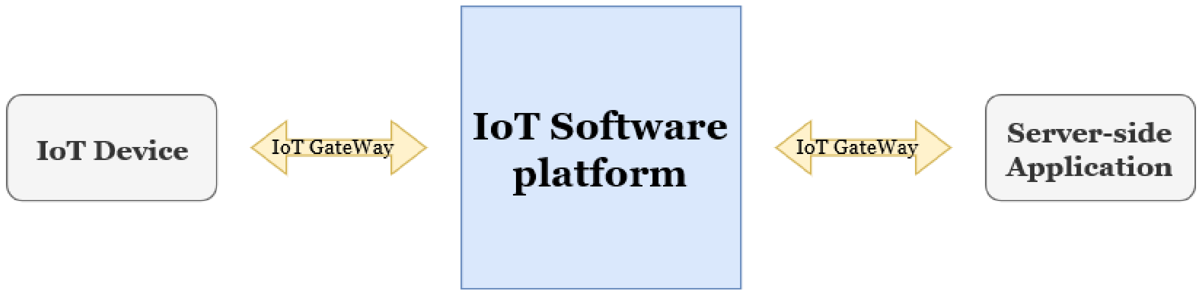


Figure 2.1 General Architecture of IoT

2.4.2 Technical requirements

From Fig 2.1, one can see the general architecture layout of IoT working. It consists of three modules in major. The 1st module includes of the IoT devices, and 2nd module is the IoT software platform chosen, and finally, the 3rd module is the server-side application, if we need to set up an IoT we need these three things apart from other advance things.

In the 1st module, we need to configure two things. Firstly, we need to have a physical device (i.e., Sensor devices), and secondly, it should be connected to the network. The 2nd module is the bridge between the 1st and 3rd module. In this module, we will do our major part of work, such as configuring the devices, describing the relationship between the hierarchy's, configuring the database, and finally designing the dashboard.

2.5 Programming structures

ThingsBoard programming structure is a little different apart from the programming logic. We need to do some additional work. Firstly, we need to have a clear idea about the hierarchy of the system which we are dealing with. In our work, the system is the building which we took. So, the whole building will be on the top in the hierarchy and hallways, classrooms, and devices will be in the lower levels. In doing this way, we are defining which assets contain what and what has a relationship with what. Then we need to configure the devices for collecting the telemetry data, visualize the data via the dashboard, and the platform, we are pushing data to store in the time-series database. The logics are written in the JS format.

On the other hand, the Node-RED provides a browser-based editor which provides a flow-based. The nodes can be deployed from the pallet in just a single click in the runtime. Then we need to configure the nodes; the programming format is in JS. In single flow, we need to set everything such as input nodes where we get the data, and we have the function nodes where we write the logic (i.e., which describes what we want to do with the input telemetry data), and the output of the logics will be taken and configured to database where will store the data, to the dashboard for data visualization and debug node.

2.6 Services Provided

ThingsBoard provides two kinds of services [1]. First, the "Community Edition" and the second is the "Professional Edition." The community edition is 100% open source and provides us with community support, unlimited devices and assets, unlimited software updates, and we can also opt to contribute to the developments. In the Professional edition, we have many types of edition available on the bases of devices and assets, and they are tabulated below.

Edition Name	No of Devices Configured	No of Assets configured
Maker	10	10
Prototype	100	100
Startup	500	500
Business	1000	1000
Enterprise	Unlimited	Unlimited

Table 2.1 ThingsBoard different Software Edition

From both kinds of service, we able to configure everything like device management, assets management, visualizing the data, processing the data, and white labeling service are available only for the professional edition.

In Node-RED provides only opensource [2], and there is no paid version with the aid this platform we can bale to configure mostly everything in the field of IoT. We can also customize the node so that we can also develop the required process through this and deploy it to the flow.

2.7 Documentation Available

In the ThingBoard, we have more documentation [1] available in both written guides and getting started videos. In the documents available in the websites will initially section provides with the following,

- Hello World - In this section, we will learn how to collect IoT device data using MQTT, HTTP or CoAP and visualize it on a simple dashboard
- End users IoT dashboards – This section provides how we can perform the essential operation over Devices, Customers, and Dashboards.
- Device data management – In this section, we will learn how we can perform an essential operation over device attributes to implement practical device management use cases.
- Getting Started with rule engines – In this section, we will learn about the ThingsBoard rule engine and how to enable filtering of incoming telemetry messages.

The second section of the document deals with the device connectivity through various protocol and provides multiple additional things, and they are as follows,

- Connect devices using ThingsBoard HTTP API – In this section, we will learn how to connect devices using HTTP protocol and ThingsBoard built-in payload.
- Connect devices using ThingsBoard MQTT API – In this section, we will learn how to connect devices using MQTT protocol and ThingsBoard built-in payload.
- Connect devices using ThingsBoard CoAP API – In this section, we will learn how to connect devices using CoAP protocol and ThingsBoard built-in payload.

- Connect devices using ThingsBoard IoT Gateway – In this section, we will learn how to deploy IoT Gateway in our local network and route messages from our devices to the cloud.

The third section of the document provides about "Data visualization" in this, we have instruction about how to configure complex ThingsBoard dashboard.

- Visualizing assets data using maps and tables – From this section, we able to get the information about how to create a new dashboard, visualize data from the asset attributes using entities table and map widgets.
- Dashboard states, aliases and widget actions- In this section, we will learn, how to add and configure new dashboard states, create various aliases, visualize the attributes data using the image map widget, create actions in different widgets in order to navigate between states, visualize telemetry data using analog and digital gauges and the time-series widget.
- Remote device control and alarm management – In this, we will learn about, add and use the node widget, create alarm rules, handles alarms using the alarm widget, and then how to make the dashboard public.
- Basic widget settings – In this, we will learn about how to change widget title, background, colors, widget title, background, colors, fonts, shadows, and so.
- Latest values map widget – in this, we will learn how to display our devices with the telemetry data on the time series map widget and modify the widget's properties.
- Time series map widget – in this section, we will learn about how to display our device with the latest telemetry data on the time series map widget and modify the widget properties.

In the fourth section of the guide, we will learn about the "Data Processing and action" with the aid of the Thingsboard rule engine. Then the fifth section of the guide deals with the introduction to the IoT data analytics (these features work only with the professional edition). Finally, the document says about some advanced features in the ThingsBoard.

Comparing to ThingsBoard, Node-RED has very little documentation but is sufficient to get started with. In the documentation section [2] we have installation guides, frequently asked question apart from this we need to focus on the followings,

- User Guide – This section comprehensive of Getting started in which we have details about how to create flow, Node-RED concepts, and using the Node-RED editor. Secondly, we have a description of configuring Node-RED, such as the setting file, setting options, securing Node-RED, and logging. Then we have guides about "Using Node-RED" in this section we have details about what are the core nodes, adding nodes to the palette (since we have large set of libraries), how to use the function nodes in which we will write all our logics, then working with the context and messages, using environmental variables, working with projects, configuring Node-RED, and Finally command-line Admin. At the end of the guide, we have an advanced concept – embedding into an existing application.

2.8 Installation of IoT Platforms

This section explains the installation procedures for Thingsboard and Node-RED platforms.

2.8.1 Installation of ThingsBoard

To ThingsBoard community edition is supported in various platforms [1] and they are tabulated below,

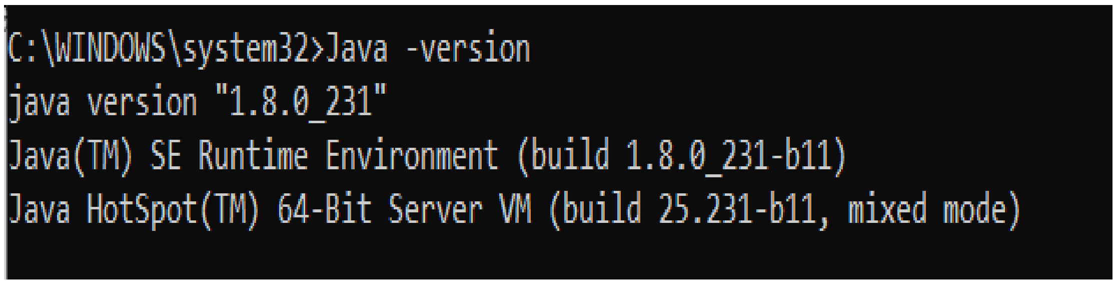
PLATFORMS	LIVE DEMO	On ThingsBoard server
	ON-PREMISE	Ubuntu
		Redhat
		Windows
		Raspberry pi3
		Docker + windows/linux/mac os
		Maveen
		Cluster setup
	CLOUD	Digital ocean

Table 2.2 ThingsBoard Installation on different platforms

From the above, our interest for the sake of the thesis is to install on the Windows platform on a personal computer to do that one has to follow the following steps.

STEP 1

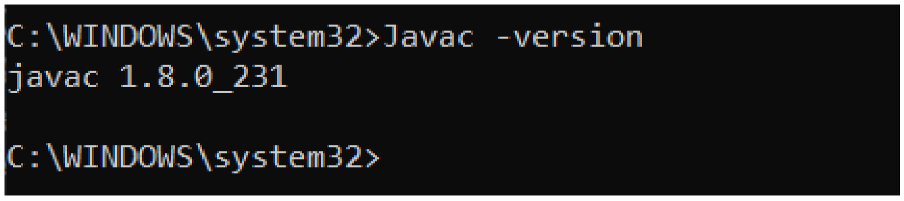
- We need to check the java version in our computer, in order to perform this we need to lunch the command prompt as administrator and need to write the cmd as **"Java -version,"** and we need to get output as shown in figure 2.2 and if this not installed we need to download java from oracle. Make sure that java server virtual machine is installed because the ThingsBoard platform does not support the other formats.



```
C:\WINDOWS\system32>Java -version
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

Figure 2.2 Command Prompt result for executing Java -version cmd

- Nextly, we need to check the Java compiler version in the system by executing the cmd as **"Javac -version"** and the results will be as shown figure 2.3



```
C:\WINDOWS\system32>Javac -version
javac 1.8.0_231

C:\WINDOWS\system32>
```

Figure 2.3 Command Prompt result for executing Java -version cmd

STEP 2

This step is very straight forward; in this, we need to download the ThingsBoard Community edition form the Git Repository.

STEP 3

- In this step, we need to set up a database to continue the installation procedure of the ThingsBoard platform. So, PostgreSQL is used as our database.
- This database can be download from the PostgreSQL website and can be configured for local running, and After the installation, we need to create a new database as "ThingsBoard."

STEP 4

- This step is carried out after the deployment of the database; in this, we will install the ThingsBoard via command prompt.
- Open the command prompt as a system administrator and change the working directory to the folder where we had to download the ThingsBoard software.
- Then we need to write the **"install-bat"** as cmd; this command is used to install the batch file.

STEP 5

- After the installation, we can start and stop the ThingsBoard Server from the command prompt by following cmd.
  - **NET START THINGSBOARD** (for starting the ThingsBoard Server)
  - **NET STOP THINGSBOARD** (for Stopping the ThingsBoard Server)
- The ThingsBoard can run from the Ip address *http://localhost:8080/*.

2.8.2 Installation of Node-RED

Node-RED is built on Node-js; this makes it ideal for running at the edge of the network on low-cost hardware such as the Raspberry Pi [2]; the supported platforms are tabulated below.

PLATFORM	Run locally	Windows
		Docker + Linux/mac OS
	On a Device	Raspberry Pi
		Beagle Bone Black
		Interacting with Arduino
		Android
	In the Cloud	IBM Cloud
		Sensetecnic FRED
		Amazon Web Service
		Microsoft Azure

Table 2.3 Node-RED Installation on different platforms

The installation procedure is not as complicated as the ThingsBoard, and our interest in point of the thesis is to Run locally in our window computer and on a device (Android).

STEP 1

To install IBM Node-RED, we need Node.js. If it is not installed on the computer, we can download it from *https://nodejs.org/en/* site.

STEP 2

After installing the Node.js, lunch the Node.js command prompt, this will automatically set our environment for the usage of the Node.js and npm.

In this command prompt, the cmd **"npm install -g –unsafe-perm node-red"** is used to install the Node-RED as a global module along with its dependencies. Once the installation is done, we will get a message about the installation.

STEP 3

After the installation, we start the server via the following cmd, in any web browser on the computer via *http://localhost:1880/*.

- Node-red (For starting the server with flow deployed)

- Node-red –safe (For starting the server without flow used)
- Ctrl + c (To Stop the server)

Then we are moving to an installation procedure for installing Node-RED on a mobile device. Firstly, we need to install the Termux app from the play store; this application makes it easy to run the Node-RED on android devices. After installing it in the prompt type the following cmd lines,

“***apt update***

apt upgrade

apt install coreutils nano node.js

npm i -g –unsafe-perm node-red

node-red”

By executing this line in the application, it will install Node-RED in the mobile device, and then we can start the server by using the same cmd mentioned above, and then we can point a browser to *http://localhost:1880* to access the editor window [2].

2.9 Possible custom extensions

The software considered is opensource and allows us to contribute to the core and mostly to the additional development of the software. Apart from this, we are interested in the nodes, not the core developments, both the software platforms allow us to develop customized nodes for our work. Most of the commonly used nodes are available in the libraries in both the platforms. Since the IoT is a growing platform now, there are no fully developed libraries available platform in the market. As explained above, with the aid of customized nodes, we can build function as per our requirements. Especially in the Node-Red, we can develop nodes which can even communicate with other software platforms apart from IoT platforms. Nodes are produced by using JS and JSON scripts.

2.10 Device connectivity and Management

Since both platforms are open-source, one can easily conclude that there will be a limitation in number device connectivity. Nevertheless, there is no such limitation on both platforms we can connect millions of devices via platforms. However, the thing that matters a lot dealing with the payload, and if we are using a vast number of the device, then we need to have a quite powerful machine to manage the payloads.

While choosing the sensor devices which send value to the platform through server instances and we are caring about the data, then data lose when the server down will be a problem. So, there should be local data storage within the sensor modules; this should be programmed in such a way that the whole server instance is down, then this module should store the data during that time and then needs to push the stored values when the server comes back to live again.

Both the platforms manage the devices very well in their way; in the ThingBoard, there is sperate device management, and its account for in the list and even the other assets relationship with a particular device is specified, so it is excellent to audit the log if any problem arises. Similarly, in the Node-RED, it depends on the programmer because while configuring the devices, he/she needs IoT devices pointed to the log files. As per the IBM Node-RED, it is highly recommended to connect the MQTT or HTTP or CoAP nodes to a debug node; this activity is carried out to check or audit the log while any problem occurs.

2.11 Data Storage

The most crucial thing in any IoT platform is the supportiveness and connectivity of databases, and many of the IoT deals with telemetry data, and most of the data are continuously obtained, so we need to connect a “Time-series database” to be in a safer side. In the ThingsBoard for using the community edition, it is recommended to use the PostgreSQL database apart from this; we use the NoSQL database and the Hybrid database. Similarly, in the Node-RED, it supports a wide variety of database, and there is no restriction or recommendations like ThingsBord.

2.12 Graphical Representation

Both platforms support and have an excellent GUI. Like the way how we are configuring the nodes, devices, and database in the platform in a similar manner, the dashboard (GUIs) also configured. Whichever device need to be displayed in the dashboard panel we need to configure them individually, and we need to set up everything such as what kind of widget we need, at what time we design instance the values should be updated to the gadgets and even the physical appearances such as the size, color, text formats, and their size there are predefined setups these are not predefined because the Dashboards are highly subjected to change based on the projects we are undertaken.

2.13 Scalability

Both platforms support scalability but in their way. Firstly, in ThingsBoard, as said in the previous paragraphs, the platforms support millions of devices, and each device needs to be programmed individually. However, in the Nod-RED, the process is not much complicated as the ThingBoard. In Node-RED, the process is not much complicated; we can export similar flows and edit the nodes according to the devices.

3 IoT CONFIGURATION FOR THE BUILDING

3.1 Building Assets

The IoT is setting up for a university building, the model building taken consisting of four floors (i.e., the ground floor, first, second, and third floor). The other building assets are known from table 3.1.

BUILDING	FLOOR	CLASSROOM	DEVICE
Building 25	Floor 0	D01	Envisense sensor module
		D02	Envisense sensor module
		D03	Envisense sensor module
		D04	Envisense sensor module
	Floor 1	D11	Envisense sensor module
		D12	Envisense sensor module
	Floor 2	D21	Envisense sensor module
		D22	Envisense sensor module
		D23	Envisense sensor module
		D24	Envisense sensor module
		D25	Envisense sensor module
	Floor 3	D31	Envisense sensor module
		D32	Envisense sensor module
		D33	Envisense sensor module

Table 3.1 Building number 25 assets

The model building has 14 classrooms, and each class has an Envisense sensor module. This sensor module consists of the following sensor device,

- Humidity
- Luminosity

- People counting
- People tracking
- PIR attendance detection
- Quality of the area
- Seismic level
- Temperature
- Thermal camera
- TVOC level measurement

The sensors in the sensor module are connected into the server, and data are transmitted through MQTT protocol, and firmware updates can be done via this protocol or through the chosen IoT platform.

3.2 Problem with ThingsBoard platform

Initially, the configuration started with ThingsBoard, but the problem with this IoT platform was that it did not allow us to connect with the InfluxDB because the platform supports only the PostgreSQL. Our primary requirement in the thesis point of view is to acquire the sensorial data from the InfluxDB, and since the platform does not support the required database, there is a necessity to change the database or the platform. To change a different database that is supported by the platform, we will have specific problems like the additional cost to buy the database, then we need to reconfigure the networks, and there will be a problem of losing previously stored data. So, the best option is to change the IoT platform, and since we are focusing the free or community edition and so we do not have the problem of spending the money on the license and the architecture and programming language for most of the IoT platforms are similar. Thus, the decision had taken, and the entire project is furtherly carried out with the IBM Node-RED, which is explained in detail in the upcoming section.

3.3 Node-RED configuration

The programming configuration and other requirements for the Node-RED are described already in detail in chapter 2 under section 2.4 & 2.5, Now in this section one can see the detailed description about how the Node-RED platform is used to acquire the current time-series data from the InfluxDB and the data are displayed in the User's dashboard this will help the building administrator to have an idea about the assets and also to understand about the current behavior of that particular asset viewed if the building has many assets and then furtherly collected data are analyzed to find the trend and also control the HVAC and also are used for developing applications.

3.3.1 Assets and devices in a classroom

With the aid of Figure 3.1, one can understand what are all the sensor device in the sensor module. The blue area in the figure is the sensor module, and the gray area is the classroom, and in general, the figure indicates that the device asset is inside the classroom asset. This kind of module is what installed in the classroom, in general, for any building which is made up of concrete and bricks are always considered as assets, and rooms and classroom are considered as building assets and any electronic device in it will be regarded as device assets.

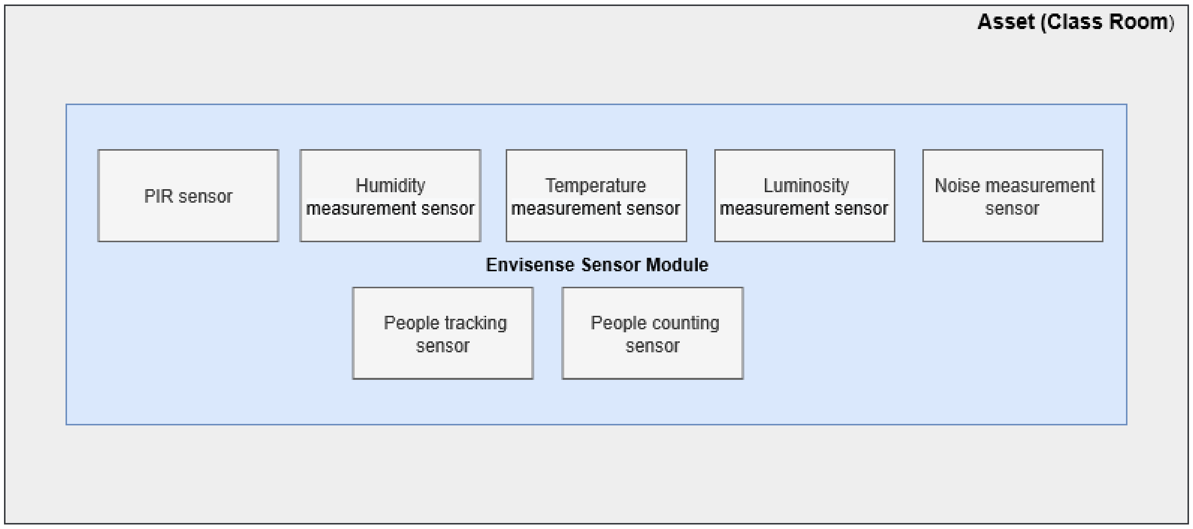


Figure 3.1 Building Asset and device asset

3.3.2 Network Architecture

In this section, from Figure 3.2 we can understand the network architecture scheme, this scheme helps the reader to know how the data is acquired from the device which is installed in the building to the Node-RED platform. The sensors in Envisense module are all IoT gateways, and so the module itself an IoT gateway, The portal is configured in a way so that observed time-series data are stored in the InfluxDB database continuously, and in the Node-RED IoT platform, the InfluxDB node is configured by entering the server ID, Database username and credentials. By doing the mentioned procedures, we can able to connect with the database, and with the aid of the SQL query and JavaScript, we can obtain the data field such as temperature, humidity, and so other fields.

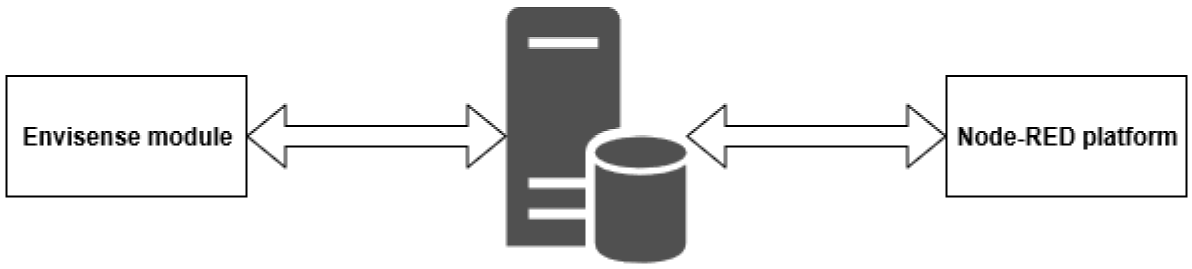


Figure 3.2 Network communication between the device and Node-RED

3.4 Classroom Prototype

This section will explain the detail configuration of each sensor in the module and its purpose of measuring the physical quantity. One can see how the data is acquired from the DB and visualized via the dashboard and how the data is analyzed for furtherly and utilized for the development of the application. The configuration is for per classroom, and the same model can be scalable to other classes in the building with small changes. Each sensor configuration in Node-RED is explained separately in their respective field sections.

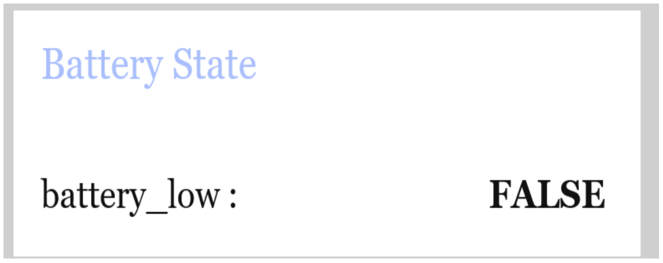
As mentioned earlier, the details about the credentials, IP address, and algorithms are not explained in the section because of the security concerns; only the general procedures are explained. The model of the overall dashboard for a classroom prototype can be seen from Figure 3.29.

3.4.1 Battery State observation

The Envisense sensor module is incorporated with the battery source, which is used as a standby and utilized when the power failure and the sensor module consist of a local data storage unit that is used whenever the server instance is down. This act observing the battery state is a safety feature, and by this, we will ensure that there is no data loss whenever the server instance is down.

So, there is a primary necessity to observe the state of the battery. This observation will check whether the battery state is low or not, and if the battery is low, then the output will be triggered as “TRUE,” or else the output will be “FALSE.”

Figure 3.3 shows the configuration for observing the battery state via the Node-RED platform. From that figure, you can see that an “Injection node” is used to trigger the operation further after the deployment of the flow. This injection node’s output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and it is synchronized with the primary id column in the DB. So, the counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the battery column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding battery state. Since we are interested in the battery state, so we need to split the array object, and for splitting the object, the split node is used; this will split the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.


Battery State

battery_low : FALSE

Figure 3.3 Node-RED dashboard view for observing the battery state

The msg.payload variable is the generic variable of the Node-RED platform we need to convert the variable name and this to set in name indicating the particular module this will be very helpful when we are auditing the log files because we have other sensor values to be observed and all will be stored in the same msg.payload format, and so when we are doing auditing, we cannot be able to understand which data is for what. So, it's strictly recommended to change the default variable name.

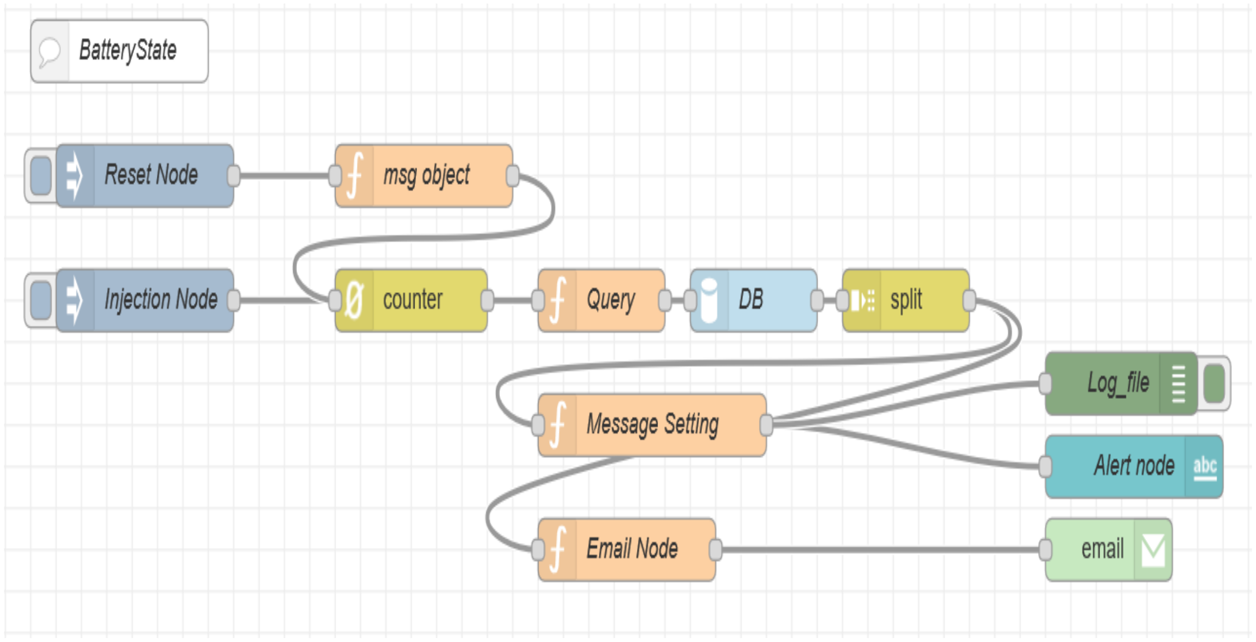


Figure 3.4 Node-RED configuration for observing the battery state

The battery state is displayed in the dashboard, and if the battery state is low, it configured in a way in the Node-RED platform to send an email to maintenance person automatically without the permission from the Building administrator this is done in this way because the battery is the primary requirement in the module to ensure to provide power when the power source is failed and to maintain functionalities of the onboard memory device, the dashboard output stating the battery state can be seen from the Figure 3.4.

3.4.2 CO2 Measurement

According to a study from the Harvard School of Public Health found that carbon dioxide has a direct, negative effect on human decision-making and cognition [3]. These impacts were observed at CO2 levels that most people and their children are frequently exposed to. So, the exposure to CO2 is not only on the outside; nowadays, even inside the classrooms, offices, and homes, we are exposing to CO2. So, there is a strong recommendation to us to measure the CO2 level at the school since a student averagely spends around more than half of the day inside the classroom, and since an increase in CO2 level harms the human health, we need to maintain the CO2 concentration in recommended levels.

So, for all reasons mentioned above we need to maintain the appropriate CO2 level in the room, our Envisense module has a CO2 measurement sensor, and the Configuration of CO2 sensor in the Node-RED platform can be seen from Figure 3.5 shows the configuration for observing the CO2 level via the Node-RED platform. From that figure, you can see that an “Injection node” is used to trigger the operation further after the deployment of the flow. This injection node’s output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and the values form the counter is synchronized with the primary id column in the DB. So, the counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the CO_2 column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding CO2 value. Since we are interested in the CO2 value, so we need to split the array object, and for dividing the object, the split node is used; this will split the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

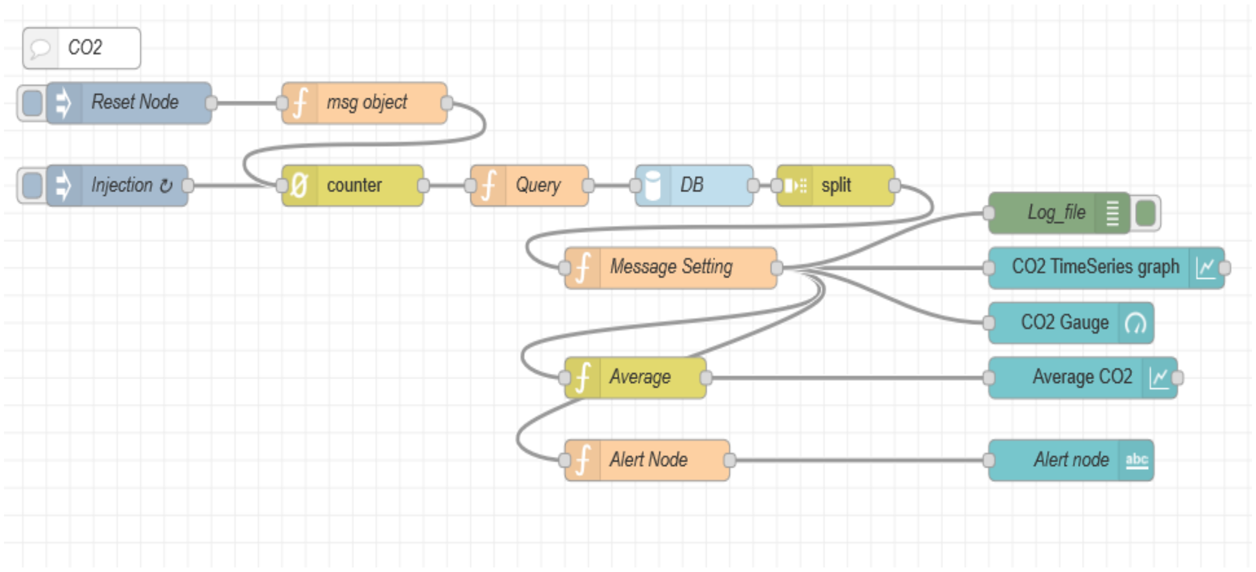


Figure 3.5 Node-RED configuration for observing the CO2

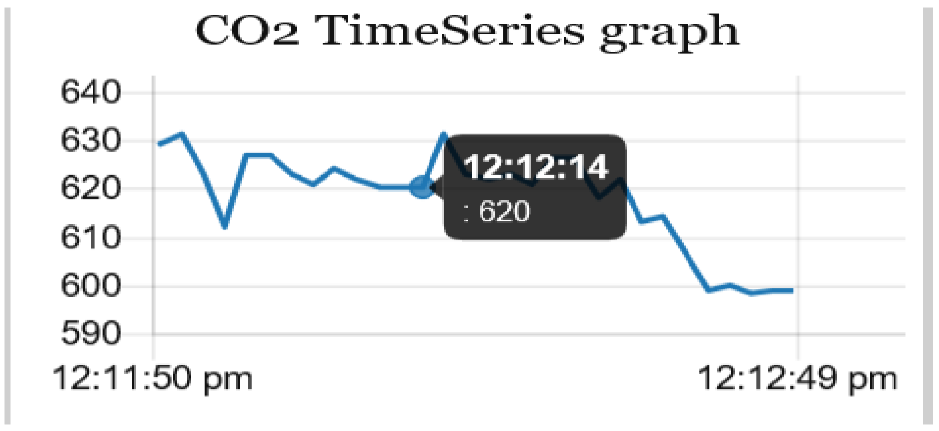


Figure 3.6 Dashboard view of CO2 Timeseries graph

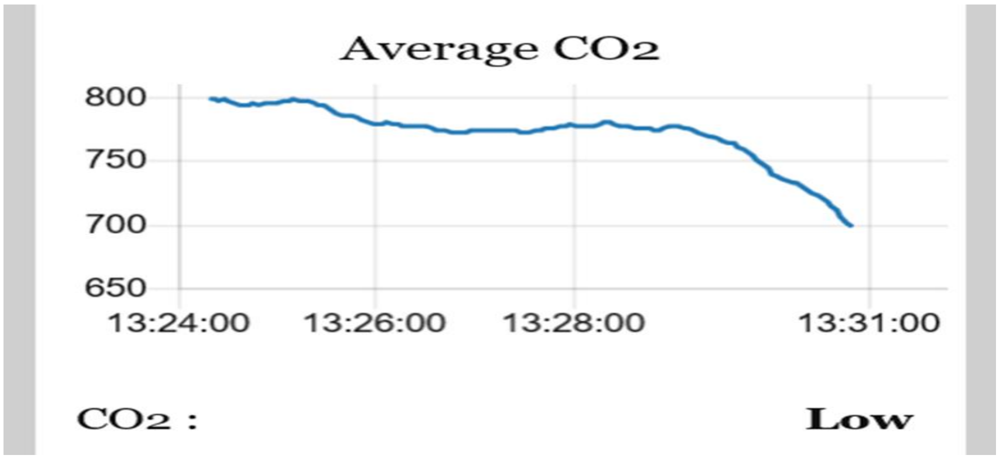


Figure 3.7 Dashboard view of Average CO2 Timeseries graph

The CO2 value from the split node is then fed to a function nodes, which has the function to change the default variable name to the msg.carbondioxide this is done because it will be useful when we checking the log files then it pushes the msg to the dashboard (which is used for the building administrator visualization) and other functions nodes such as average node, alert node.

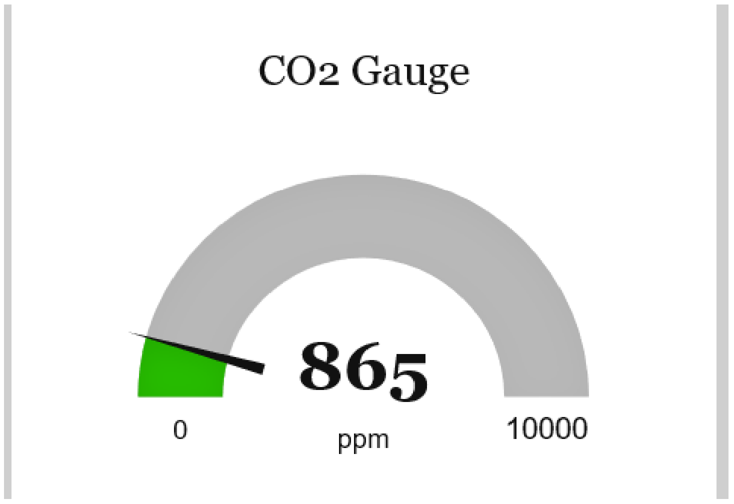


Figure 3.8 Dashboard view of current CO2 via gauge

The average function node which will take the average over the last one hour (i.e., 500 samples) and will push the value to the specified dashboard. Finally, the alert node has a function that will describe the state of the carbon dioxide level in the classroom. The time-series CO2 graph is seen from Figure 3.6, the average CO2 time-series graph and the alert message are seen in figure 3.7, and the current value of the CO2 is seen from the gauge(Figure 3.8).

3.4.3 Humidity Measurement

The indoor air quality is vital for the health and academic success of the students. So, to maintain good air quality apart from other factors, humidity is one of the essential elements for keeping the air quality good in the given area (i.e., classroom) [4]. The ideal humidity level is around 30-50%, and any deviations outside these parameters can affect the student's and professor's health. Both low and high level of humidity has an adverse effect, so we need to maintain an average level concerning any internal and external changes.

So, for all reasons, as mentioned above, we need to maintain the appropriate humidity level. Our Envisense module has a humidity measurement sensor, and the Configuration of humidity sensor in the Node-RED platform can be seen from Figure 3.9.

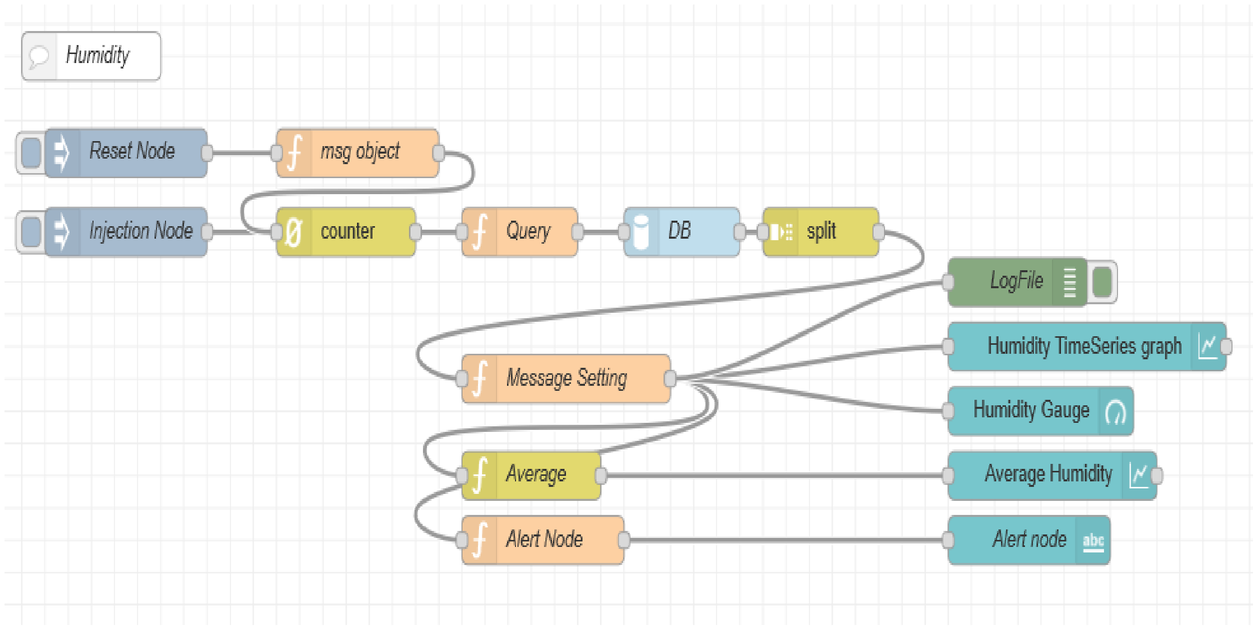


Figure 3.9 Node-RED configuration for observing the Humidity

From that Figure 3.9, you can see that an “Injection node” is used to trigger the operation. Further, after the deployment of the flow, this injection node’s output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and it is synchronized with the primary id column in the DB. The counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the humidity column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding humidity value. Since we are interested in the humidity value, so we need to split the array object, and for dividing the object, the split node is used. This will split the value in an array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

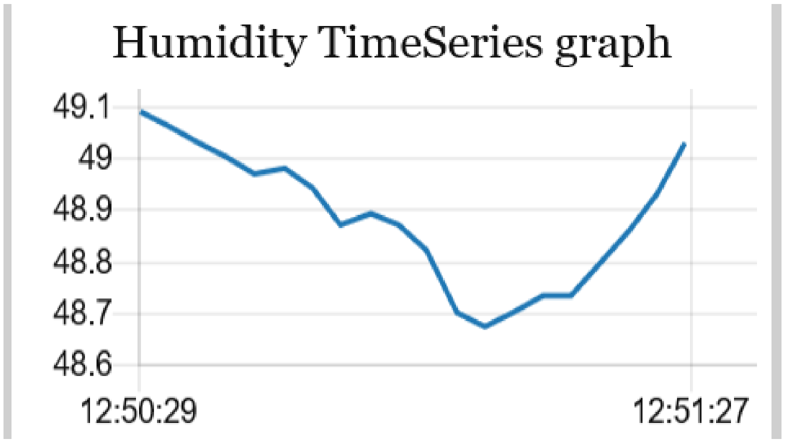


Figure 3.10 Dashboard view of Humidity Timeseries graph

The humidity value from the split node is then fed to three function nodes firstly to the message setting node, which has the function to change the default variable name to the msg. humidity, this is done because it will be useful when we were checking the log files or debug node, then it also has the function to return the message to the dashboard (which is used for the building administrator visualization). Secondly, the average function node which will take the average over the last one hour (i.e., 3600 samples) and will push the value to the specified dashboard. Thirdly to the alert node function and which will describe the level of the humidity level in the classroom. The time-series humidity graph is seen from Figure 3.10, the average humidity time-series graph and the alert message are seen in figure 3.11, and the current value is seen via the gauge this can be seen from Figure 3.12.

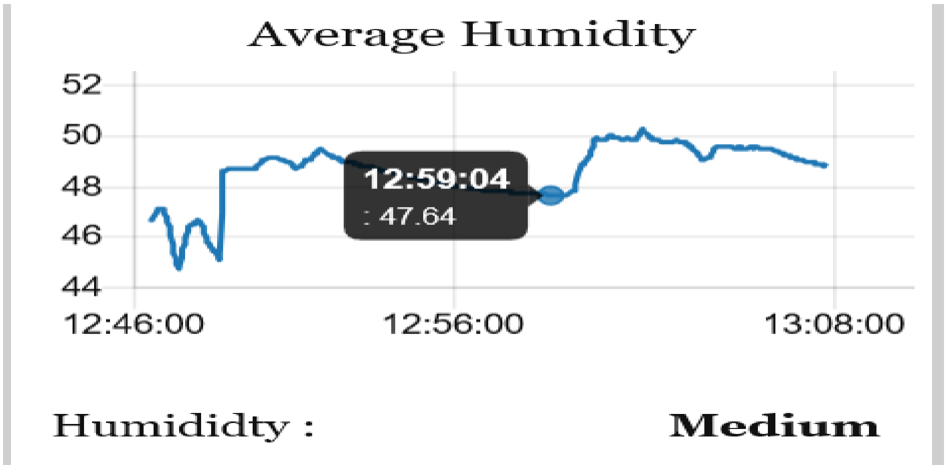


Figure 3.11 Dashboard view of average Humidity Timeseries graph

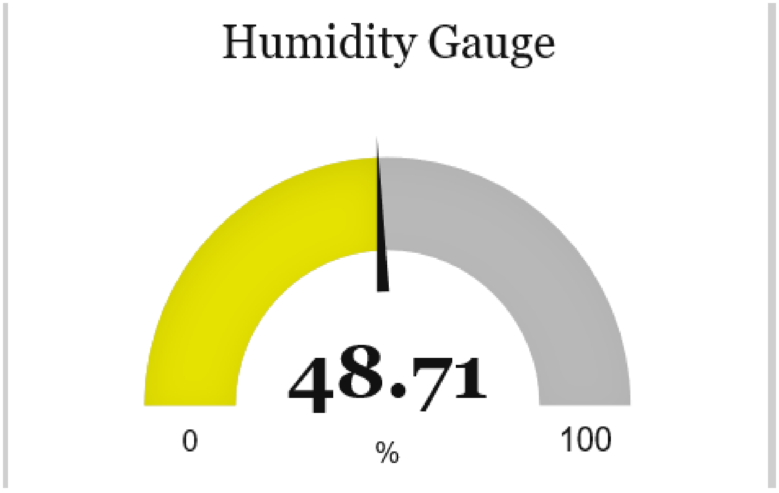


Figure 3.12 Dashboard view of current Humidity via gauge

3.4.4 Luminosity Measurement

Lighting is a dominant factor in the brain's ability to focus. Studies show that learners in brightly lit environments got higher grades than those in dimly lit classrooms [5]. It seems that poor lighting reduces the effectiveness of the brain's power to gather data. And full-spectrum lighting (like natural light) works best to improve behavior, create less anxiety and stress, and improve overall health. To make sure that the proper lighting is available, we need to check the luminosity level in the room.

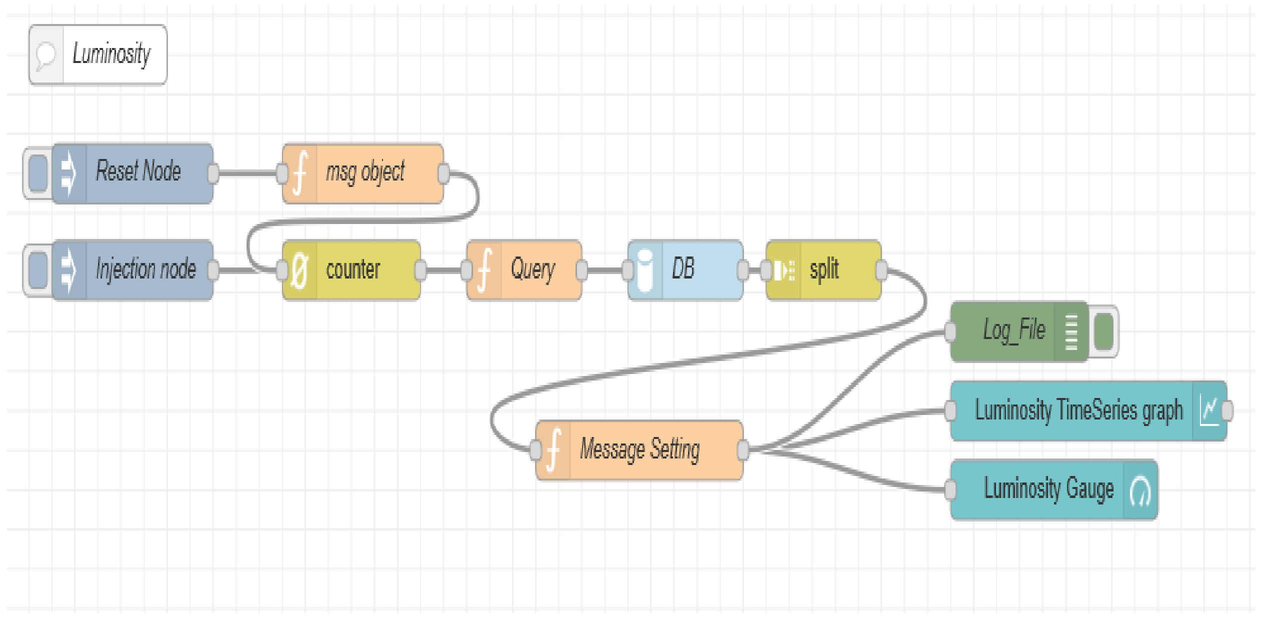


Figure 3.13 Node-RED configuration for observing the Luminosity

So, for all reasons, as mentioned above, we need to maintain the appropriate luminosity level. Our Envisense module has a luminosity measurement sensor, and the configuration of the luminosity sensor in the Node-RED platform can be seen from Figure 3.13. From the figure, one can see that an "Injection node" is used to trigger the operation further after the deployment of the flow. This injection node's output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and the counter is synchronized with the primary id column in the DB. The counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the luminosity column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding luminosity value. Since we are interested in the luminosity value, so we need to split the array object, and for dividing the object, the split node is used; this will split the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

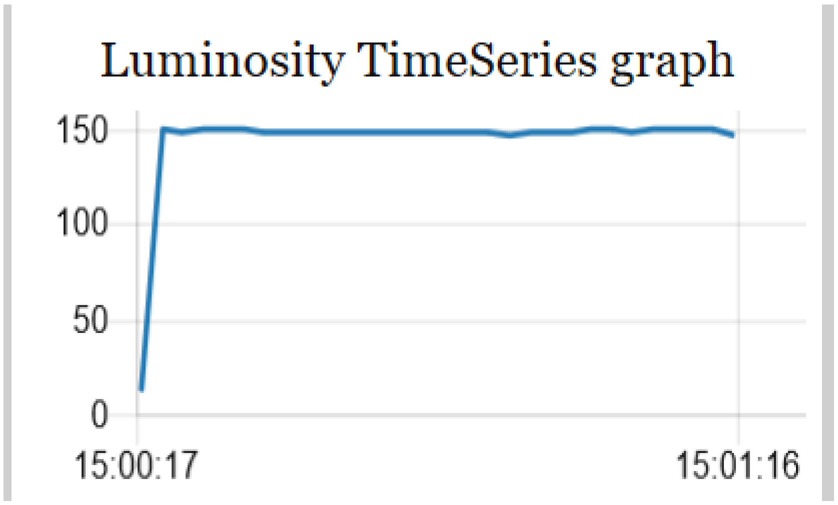


Figure 3.14 Dashboard view of Luminosity Timeseries graph

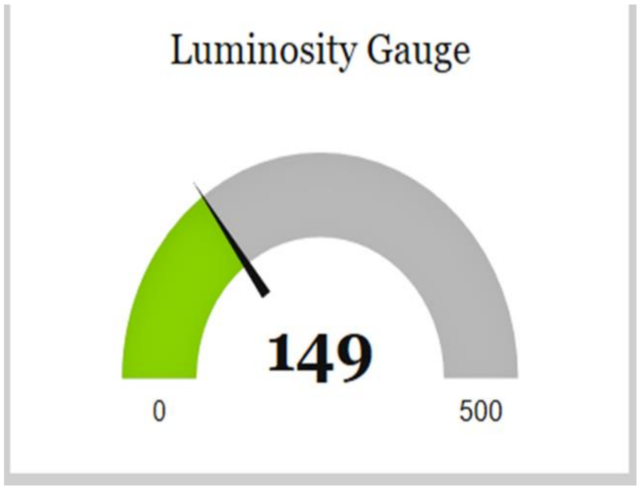


Figure 3.15 Dashboard view of current Luminosity via gauge

The luminosity value from the split node is firstly sent to the message setting node which has the function to change the default variable name to the msg.luminosity this is done because it will be useful when we checking the log files or debug node then it also has the function to return msg to the dashboard (which is used for the building administrator visualization) this can be seen from the Figure 3.14 which show the time-series graph with some past data and current value can is seen via the gauge such as in Figure 3.15.

3.4.5 Noise Measurement

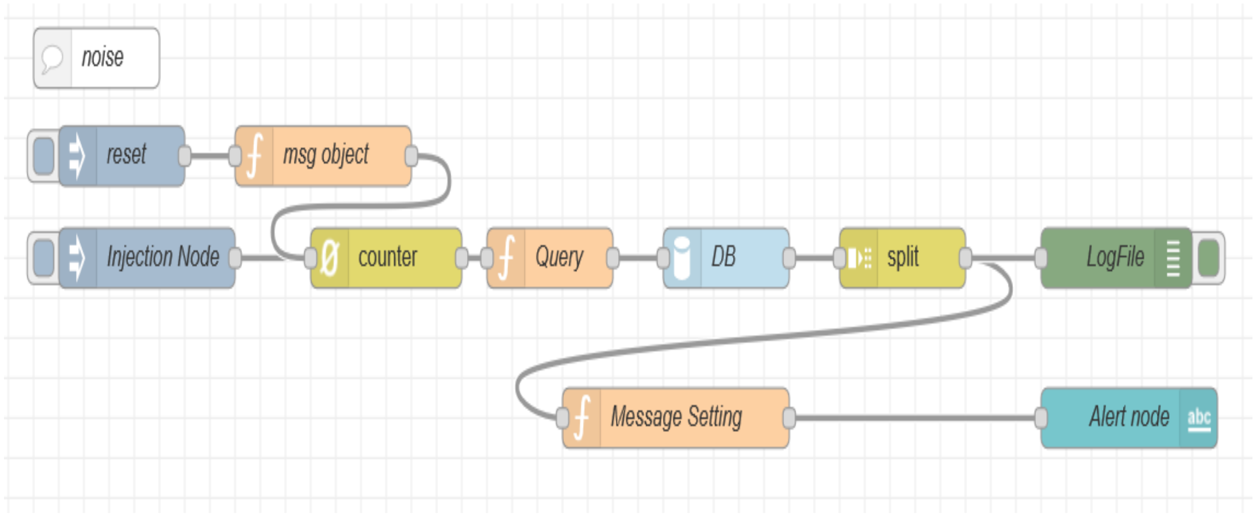


Figure 3.16 Node-RED configuration for observing the Noise level

Our onboard Envisense module has noise measurement to measure the indoor noise. We need to maintain the sound in an appropriate dB because the excessive noise level can create a negative learning environment [6]. The configuration in the Node-RED platform to obtain the noise level is shown in the Figure. Currently, there is no observation observed form this sensor. But their purpose of configuring in IoT platform because to use soon by the building administrators.

3.4.6 People Enumeration

Tracking the movement of people is an essential notion of the smart building because humans use all the things inside the building or classroom or an office (in our case by students or by professors or by maintenance people or some other university officials) and so by tracking them we can automatically control the devices such as light, ventilation and heating appliances and make them work automatically, this can eradicate the human negligence, and this will set a path to use the energy and resource in a most optimized manner. Moreover, in our ground, the installed Envisense module from the classroom has a thermal camera and counters. The thermal camera will detect how many people inside and outside and updates in people in counter, and with the value-from PIR and value from the people inside, we can obtain the classroom occupancy; this is explained in section 3.6.1.

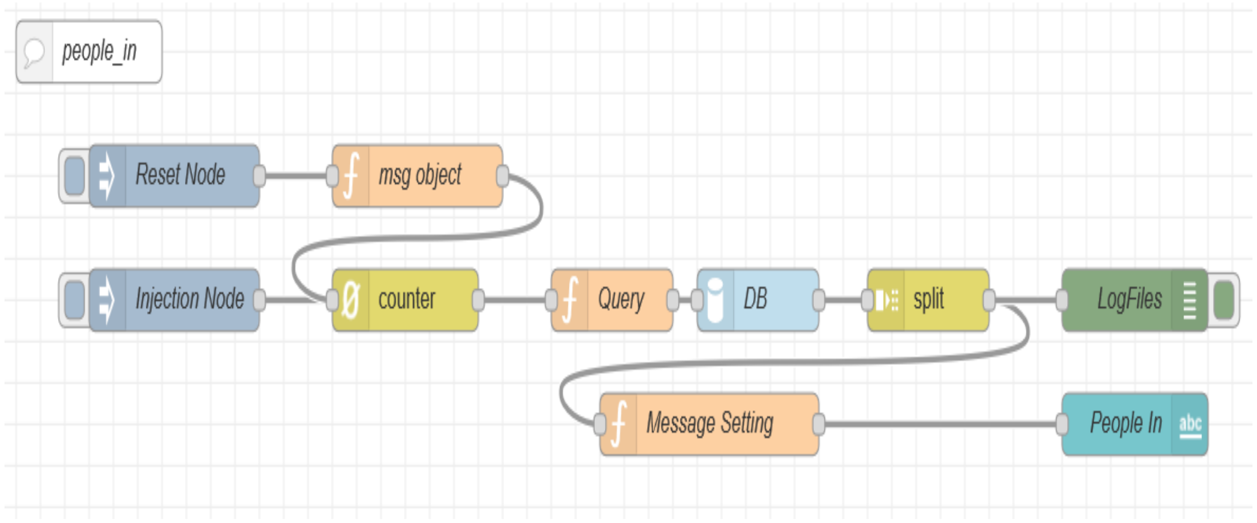


Figure 3.17 Node-RED configuration for observing the peoplein value

3.4.6.1 People In

With the aid of the thermal camera and counter, we count the total number of people inside the classroom, and the Node-RED configuration for obtaining the people in value is seen from Figure 3.17.

3.4.6.2People Out

With the aid of the thermal camera and counter, we count the total number of people inside the classroom, and the Node-RED configuration for obtaining the people out value is seen from Figure 3.18.

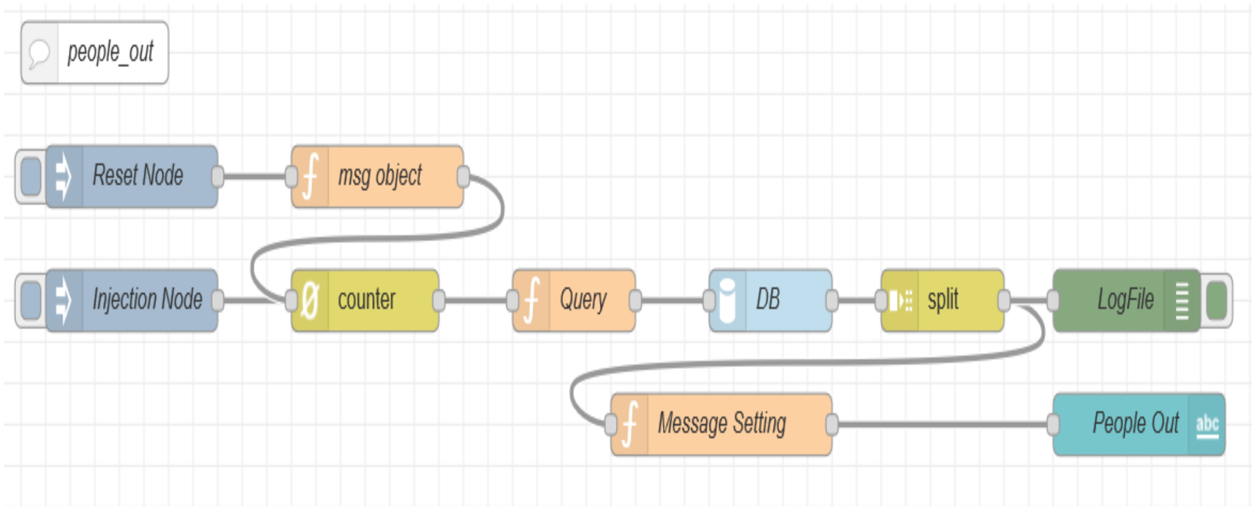


Figure 3.18 Node-RED configuration for observing the peopleout value

3.4.7 PIR Measurement

The Passive Infrared sensor is mostly used as motion detectors; in general, this sensor is correlated with lighting device switching on/ off automatically when it detects.

The Configuration of the PIR sensor in the Node-RED platform can be seen from Figure 3.19. From the figure, we can see that an “Injection node” is used to trigger the operation further after the deployment of the flow. This injection node’s output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and it is synchronized with the primary id column in the DB. The counter value will be continuously updated in the query node in each time the injection node is triggered, and a query is used to fetch the row data from the PIR column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding PIR value. Since we are interested in the PIR value, so we need to split the array object, and for dividing the object, the split node is used; this will split the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

The PIR value from the split node is then fed first to the message setting node which has the function to change the default variable name to the msg.carbondioxide this is done because it will be useful when we are checking the log files then it also has the function to return msg to the dashboard.

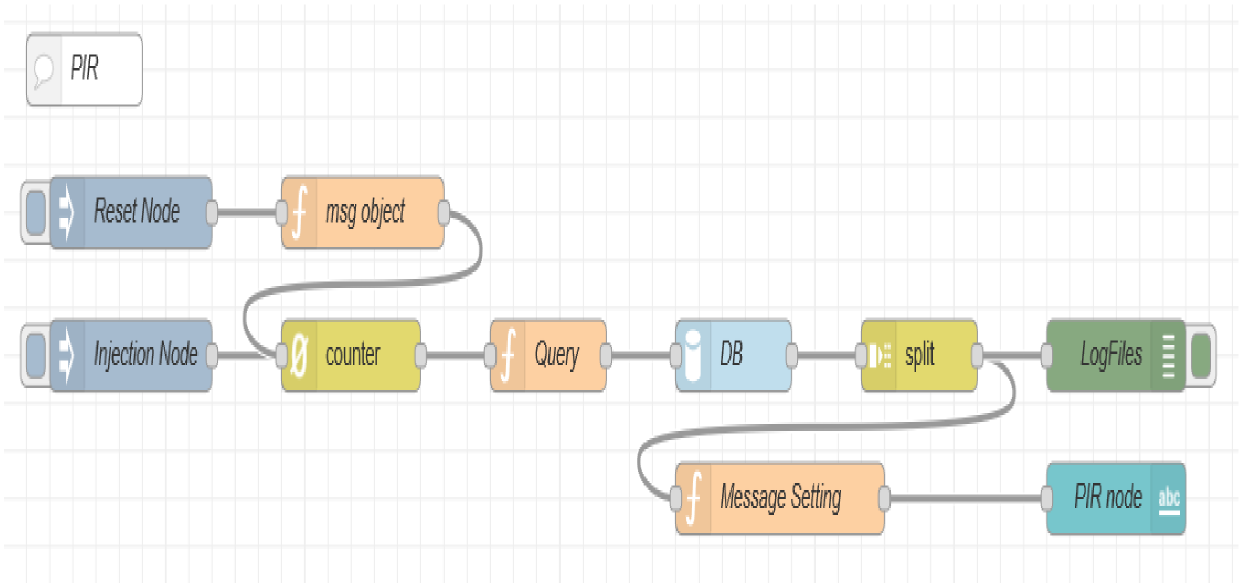


Figure 3.19 Node-RED configuration for observing the PIR value

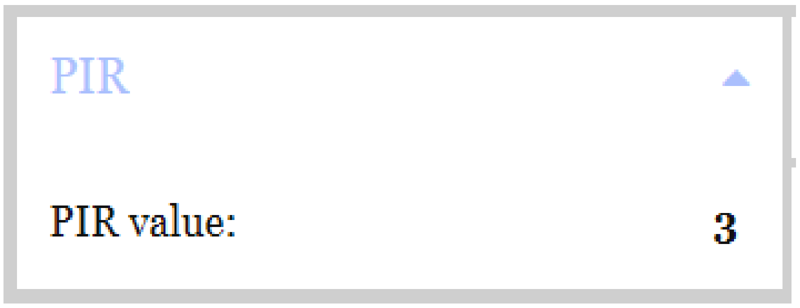


Figure 3.20 PIR Visualization in dashboard

3.4.8 Seismic level

In our onboard Envisense module has a sensor to measure the seismic level, but currently, we are not observing the value from this sensor, but still, the configuration is made in the IoT platform and used soon. the setup for observing the seismic level via the Node-RED platform can be seen from the Figure

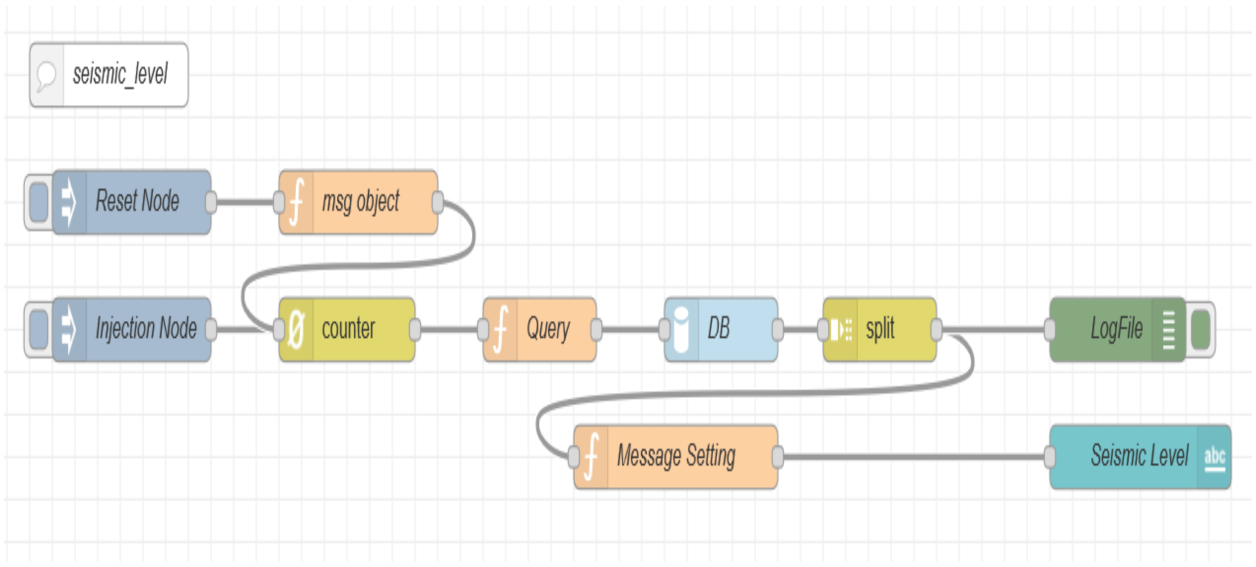


Figure 3.21 Node-RED configuration for observing the seismic level

3.4.9 Temperature Measurement

The temperature is an essential factor in any room or office, or building or home because and inappropriate heat can cause discomfort for people who are inside the building [7]. So, the place like classrooms should be taken utmost care because of discomfortable (too high temperature or too low temperature) can affect the students' performance.

So, for all the reasons mentioned above, we need to maintain the appropriate temperature level. Our Envisense module has a temperature measurement sensor, and the Configuration of the temperature sensor in the Node-RED platform can be seen from Figure 3.6 shows the configuration for observing the temperature level via the Node-RED platform. From that figure, you can see that an "Injection node" is used to trigger the operation further after the deployment of the flow. This injection node's output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and the counter is synchronized with the primary id column in the DB. The counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the temperature column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding temperature value. Since we are interested in the temperature value, so we need to split the array object, and for breaking the object, the split node is used; this will cut the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

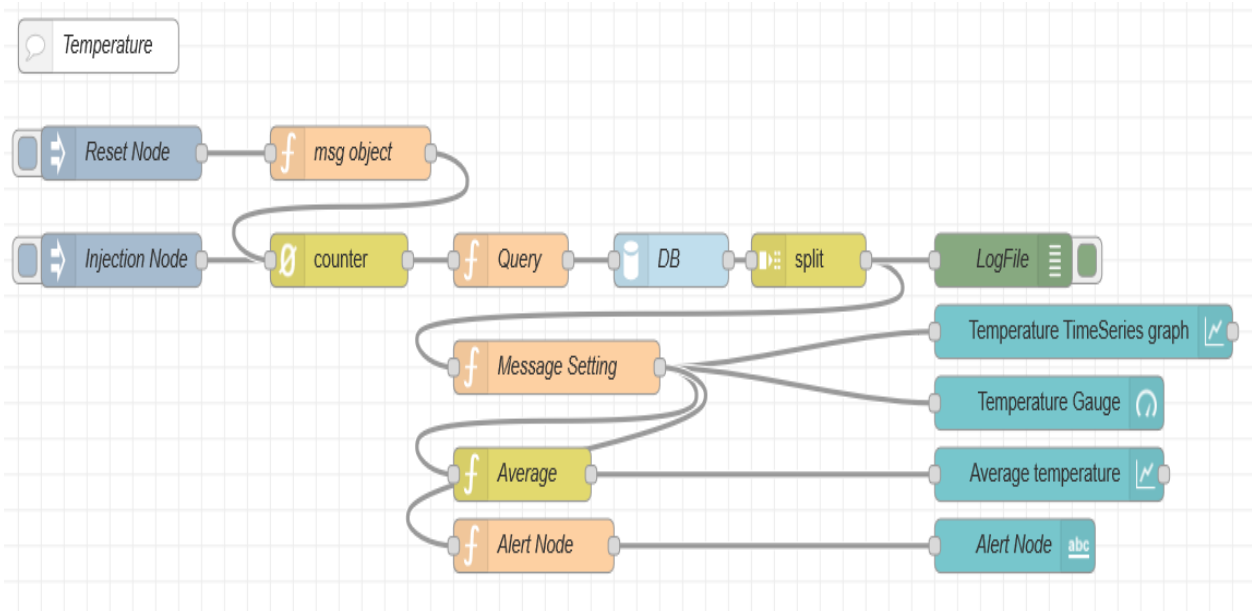


Figure 3.22 Node-RED configuration for observing the temperature level

The temperature value from the split node is then fed to three function nodes firstly to the message setting node which has the function to change the default variable name to the msg.temperature this is done because it will be useful when we checking the log files or debug node then it also has the function to return msg to the dashboard (which is used for the building administrator visualization). From the message setting node to the average function and alert nodes, the average function node will take the average over the last one hour (i.e., 3600 samples) and will push the value to the specified dashboard. Thirdly to the alert node function and which will describe the level of the temperature level in the classroom. The entire temperature sensors dashboard visualization can be seen from the figure.

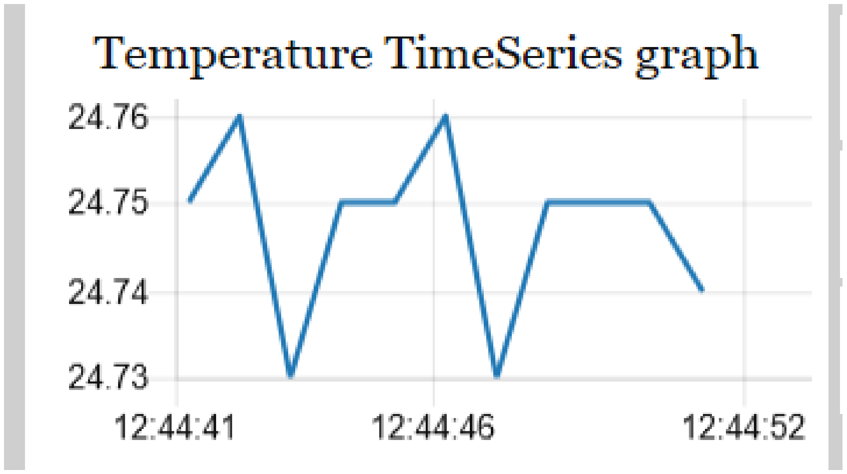


Figure 3.23 Dashboard view of Temperature Timeseries graph

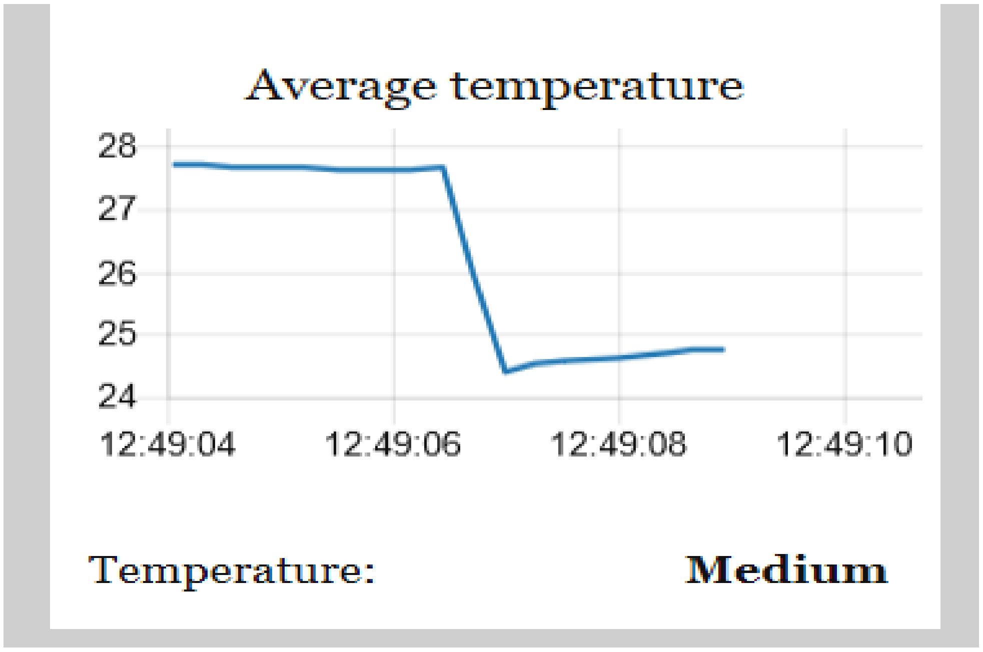


Figure 3.24 Dashboard view of average Temperature Timeseries graph

3.4.10 TVOC Measurement

For maintaining the indoor air quality apart from the humidity and temperature, we need to check the level of volatile organic compounds inside the room because the VOC has potential health effects such as Sensory Irritation, Cognitive abilities, and sick building syndrome [3].

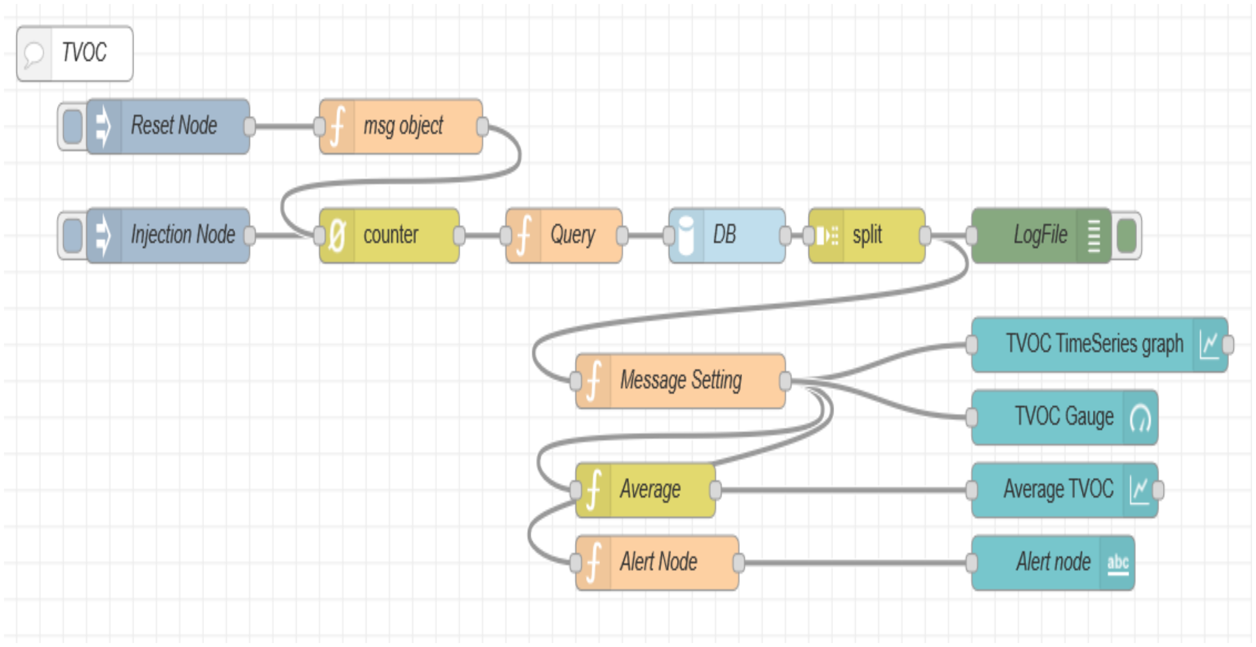


Figure 3.25 Node-RED configuration for observing the TVOC level

So, for all the reasons, as mentioned above, we need to maintain the appropriate TVOC level. Our Envisense module has a TVOC measurement sensor, and the Configuration of TVOC sensor in the Node-RED platform can be seen from Figure 3.25.

From the figure, we can see that an "Injection node" is used to trigger the operation further after the deployment of the flow. This injection node's output is the input to a counter, and when each time counter gets input from the injection node, then it will add one with the previous integer value, and initially, the counter is set at zero. This counter variable is the replication of the primary id column, and it is synchronized with the primary id column in the DB. The counter value will be continuously updated in the query node in each time the injection node is triggered, and this query is used to fetch the row data from the TVOC column entry in the InfluxDB database. The value fetched from the database will be in an array object with two entries on being the primary id column value, and the other is the corresponding TVOC value. Since we are interested in the TVOC value, so we need to split the array object, and for breaking the object, the split node is used; this will split the value in array object and returns the mentioned scalar value. The output from the split node is transported in the msg.payload variable to the other nodes.

The TVOC value from the split node is then fed to three function nodes firstly to the message setting node, which has the function to change the default variable name to the msg.TVOC is done because it will be useful when we are checking the log files or debug node, then it also has the function to return msg to the dashboard (which is used for the building administrator visualization). Secondly, the average function node which will take the average over the last one hour (i.e., 3600 samples) and will push the value to the specified dashboard. Thirdly to the alert node function and which will describe the level of the TVOC level in the classroom. The entire TVOC sensors dashboard visualization can be seen from the figure.

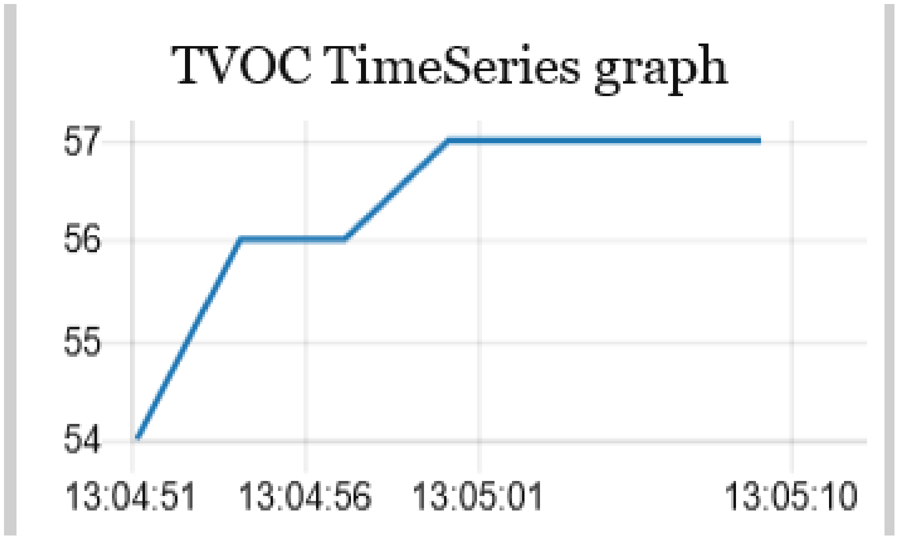


Figure 3.26 Dashboard view of TOVC Timeseries graph

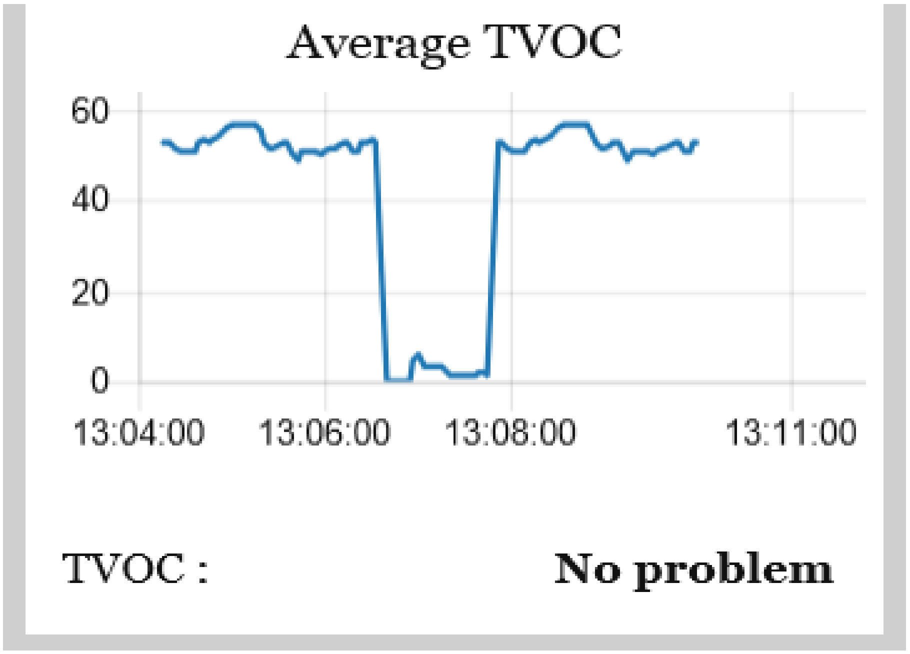


Figure 3.27 Dashboard view of average TVOC Timeseries graph

3.4.11 White Level Measurement

The white level measurement is related to the white noise measurements; for now, we are not using this sensor; it's set up for the future purpose. But the configuration is done in the platform, and at the time of using the node will be deployed, and the configuration in the Node-RED platform is seen from Figure 3.28.

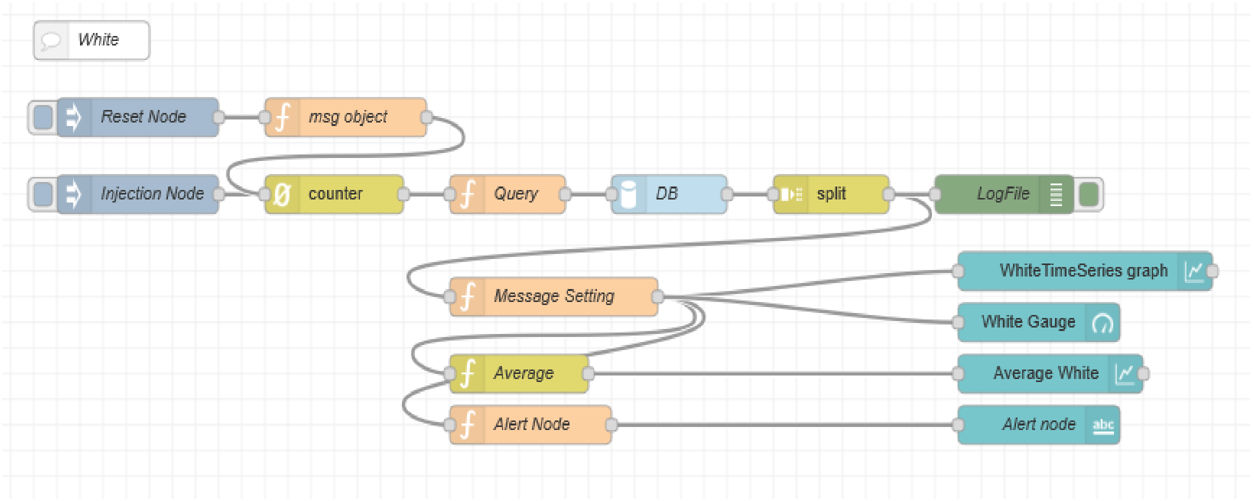


Figure 3.28 Node-RED configuration for observing the white level

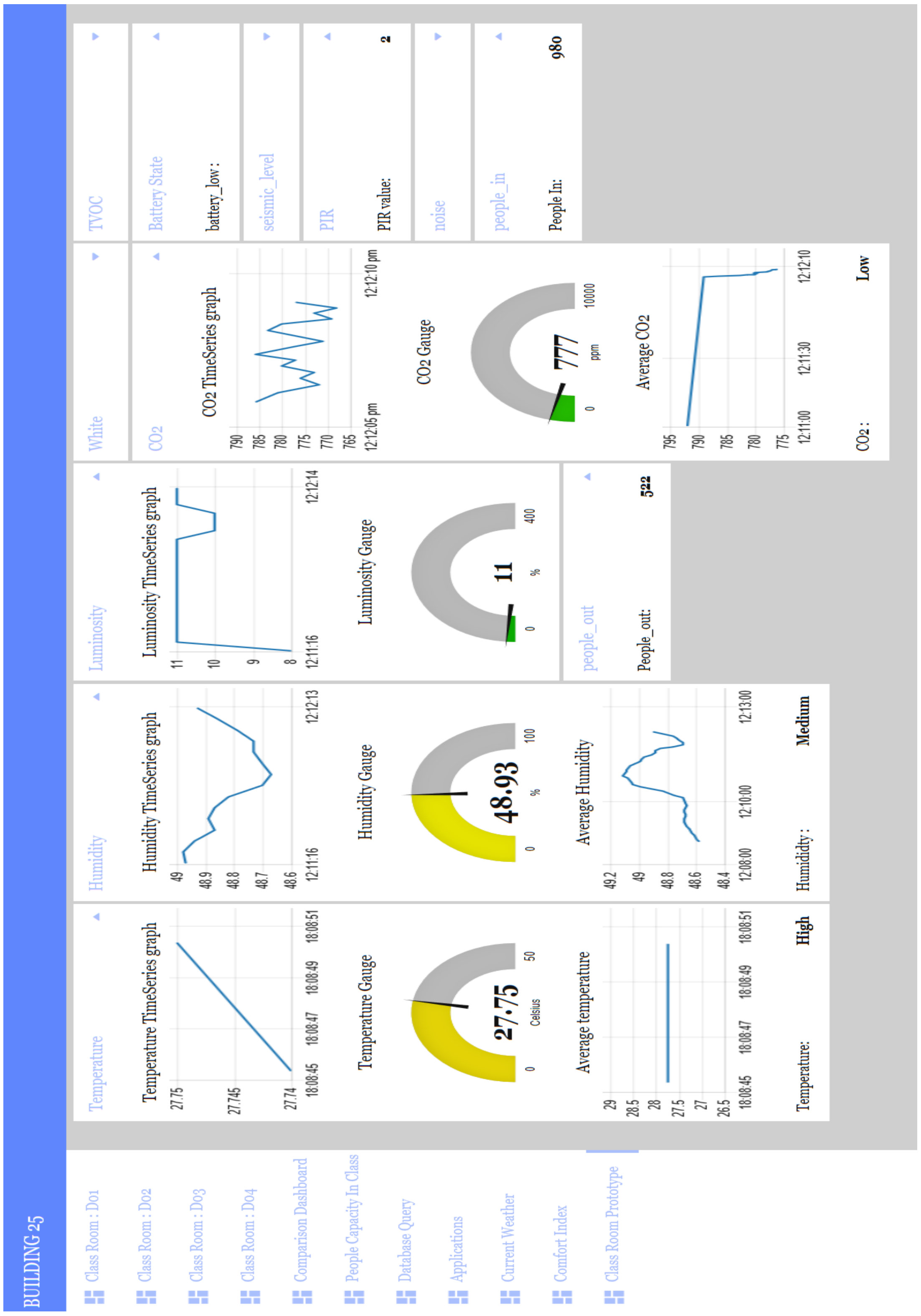


Figure 3.29 Full dashboard view for a classroom

3.5 Comfort Indices

With the measurement of temperature and relative humidity from the classroom, we can able to calculate the various scales of comfort indices, and they are as follows.

3.5.1 Dewpoint Temperature

By calculating the dewpoint value, we can necessarily measure the atmospheric moisture in the classroom [8]. The T_D value can be calculated based on equation 3.1, and T_D value is displayed in the user's dashboard (Figure 3.15), and the configuration in Node-RED is merely effortless and can be seen from Figure 3.30. We use a function node (Dewpoint) where it gets the value of temperature and relative humidity from the global variable of RH and T from the sensor measurements using the get function, the equation 3.1 is written in the function node, and then a new value is calculated every time the injection node is triggered.

$$T_D = \left(\frac{H}{100}\right)^{1/8} \times (112 + 0.9 \times T) + 0.1 \times T - 112 \qquad (3.1)$$

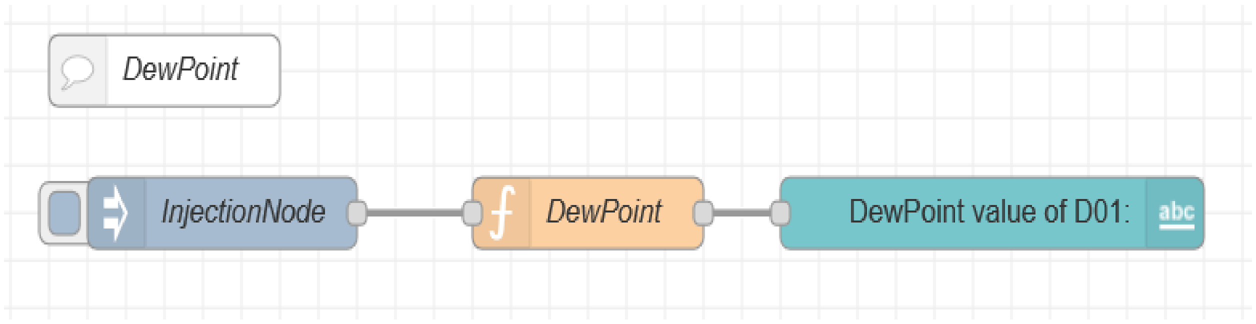


Figure 3.30 Node-RED configuration for calculating Dewpoint Temperature

3.5.2 Discomfort Index

By calculating the discomfort index, we can able to figure out how the occupants in the classroom feel [9]. The DI value is calculated by using equation 3.2, and the Scale of discomfort index is read from table 3.5, and the configuration in Node-RED is merely effortless, and it is seen from Figure 3.31. We use a function node (Discomfort Index) where it gets the value of temperature and relative humidity from the global variable of RH and T from the sensor measurement using the get function. The equation 3.2 is written in the same node function node. And then, a new value is calculated and updated in the dashboard every time the injection node is triggered.

$$\boldsymbol{DI = T - (0.55 \times (1 - 0.01H) \times (T - 14.5))} \qquad (3.2)$$

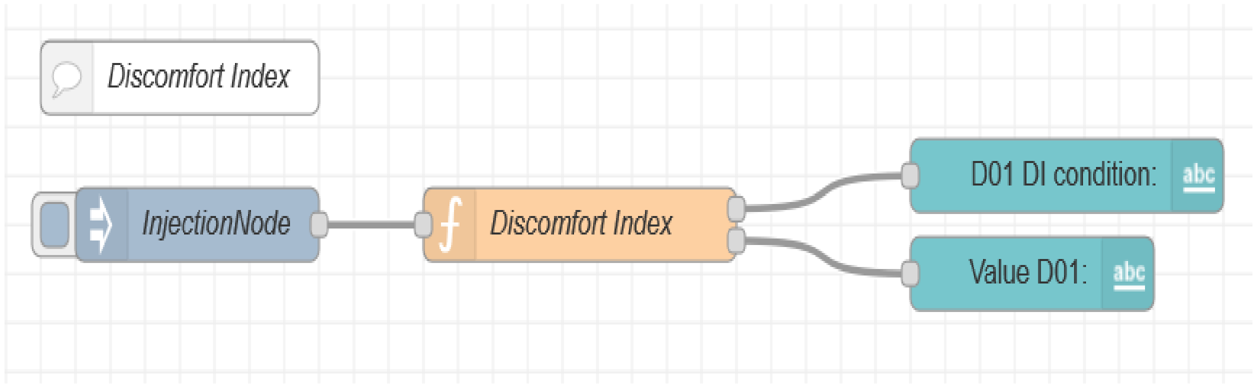


Figure 3.31 Node-RED configuration for calculating Discomfort Index

Range of Discomfort Index	Scale of Comfort
Below 21°C	No discomfort
21°C to 24°C	Under 50% population feels discomfort
24°C to 27°C	Most 50% population feels discomfort
27°C to 29°C	Most of the population feels discomfort
29°C to 32°C	Everyone feels severe stress
Above 32°C	State of a medical emergency

Table 3.2 Range and Scale of Discomfort Index

3.5.3 Heat Index

The heat index (HI) or humiture is an index that combines air temperature and relative humidity, and HI is sometimes referred to as the apparent temperature, which is a measure of how hot it feels [10]. The range of heat index and scale of comfort can be read from Table 3.4, the value of HI is calculated by equation 3.3, and the configuration in Node-RED is merely comfortable. We use a function node (Heat Index) where it gets the value of temperature, and relative humidity from the global variable of RH and T from the sensor measurements using a get function, and the equation 3.3 is written in the node, and the new value is calculated every time the injection node is triggered, and the scale is selected based on the value obtained through the calculating the equation 3.3 and then HI value and the corresponding scale of comfort is then pushed to dashboard the configuration in the Node-RED can be seen from the Figure 3.32.

$$\boldsymbol{HI} = \mathbf{0.5} \times (\boldsymbol{T} + \mathbf{61} + [(\boldsymbol{T} - \mathbf{68}) \times \mathbf{1.2}] + [\boldsymbol{RH} \times \mathbf{0.094}]) \tag{3.3}$$

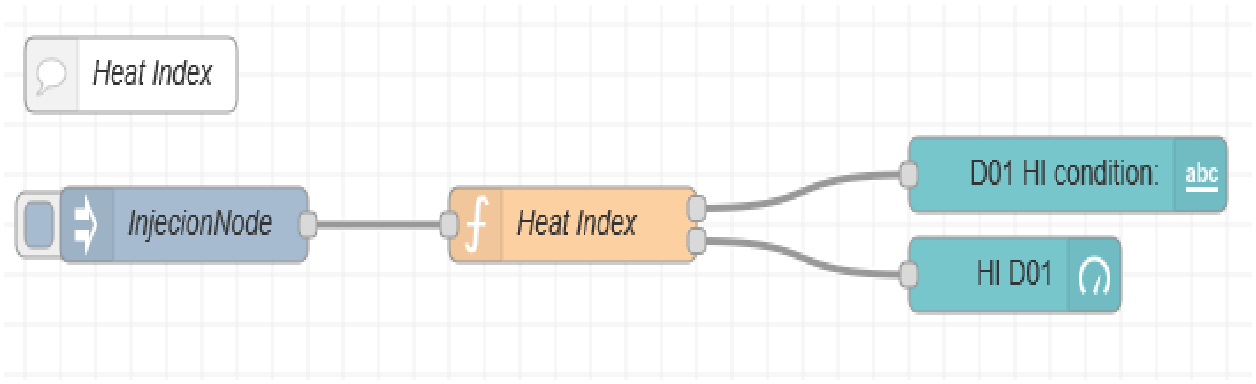


Figure 3.32 Node-RED configuration for calculating Heat Index

Range of Heat Index	Scale of Comfort
26ºC to 32ºC	Caution: fatigue is possible with prolonged exposure and activity. Continuing activity could result in heat cramps.
32ºC to 41ºC	Extreme caution: heat cramps and heat exhaustion are possible. Continuing activity could result in heatstroke.
41ºC to 54ºC	Danger: heat cramps and heat exhaustion are likely; heatstroke is probable with continued activity.
Above 54ºC	Extreme danger: heat stroke is imminent.

Table 3.3 Range and Scale of Heat Index

3.5.4 Humidex

The humidex (short for humidity index) is an index number used by Canadian meteorologists to describe how the weather feels too average person by combining the effect of heat and humidity [4]. The range and scale of Humidex can be read from Table 3.4, and the value of humidex is calculated by equation 3.4, and vapor pressure is calculated by using equation 3.5, the value is used in humidex equation, and the configuration in Node-RED merely is easy we use a function node (humidex) where it gets the value of temperature and relative humidity from the global variable of RH and T in the sensor measurements using a get function and the equation 3.4 and equation 3.5 are written in the function node, and whenever the injection node is triggered new value is calculated and pushed to the dashboard node configuration for humidex is seen from Figure 3.33.

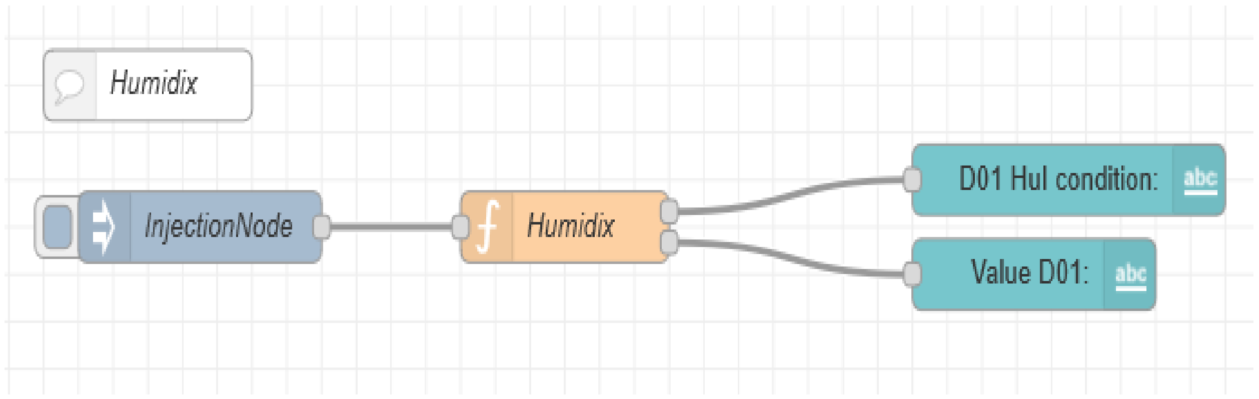


Figure 3.33 Node-RED configuration for calculating Humidex

$$\boldsymbol{Humidex = T + \left(\frac{5}{9} \times e^{-10}\right)} \tag{3.4}$$

$$\mathbf{e} = \left(\mathbf{6.112 \times 10^{(7.5 \times T/(237.7+T)) \times (H/100)}}\right) \tag{3.5}$$

Range of Humidex	Scale of Comfort
20ºC to 29ºC	Little to no discomfort
30ºC to 39ºC	Some discomfort
40ºC to 45ºC	Significant Discomfort; avoid exertion
Above 45ºC	Dangerous; Heatstroke quite possible

Table 3.4 Range and Scale of Humidex

3.5.5 Temperature Humidity Index

With the measure THI, we can able to find the reaction of the human body to a combination of heat and humidity [10] [7]. From the equation 3.6, we can calculate the THI value, and from the Table 3.6 we can read the range and scale of comfort of the THI and the configuration in Node-RED merely is simple, and we use a function node (Temperature Humidity) where it gets the value of temperature, and relative humidity form the global variable of T and RH using a get function, and the equation 3.6 is written in the function node, and the new value is calculated every time the injection node is triggered, the configuration in the Node-RED can be seen from the Figure 3.34.

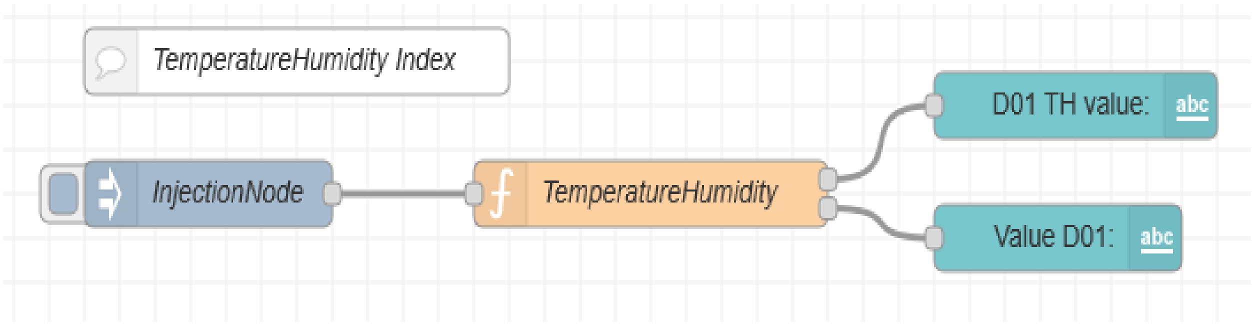


Figure 3.34 Node-RED configuration for calculating Temperature Humidity Index

$$THI = \left[\left[1.8 + T_a - \left(1 - \frac{H_a}{100}\right) \times (T_a - 14.3)\right] + 32\right] \tag{3.6}$$

Range of Temperature Humidity Index	Scale of Comfort
Below 68	No discomfort
68<THI<72	Slightly discomfort
72<THI<75	Discomfort
75<THI<79	Alert
Above 79	Emergency

Table 3.5 Range and Scale of Temperature Humidity Index

3.5.6 Wind Chill Index

The wind chill index measures how cold people feel when outside. The wind chill is based on the rate of heat loss from exposed skin caused by wind and cold [11]. As the wind increase, it draws heat from the body, driving down skin temperature and eventually the internal body temperature the windchill index value is calculated by the equation 3.7 and the range and scale can be read from the Table 3.7 and the configuration in Node-RED merely is easy we use a function node (Windchill Index) where it gets the value of outside temperature and wind speed (these values are obtained through third-party weather prediction platforms, and details of this will be explained in the upcoming section 3.6.4) from the global variable using a get function, and the equation 3.7 is written in the node, and the new value is calculated every time the injection node is triggered, the configuration in the Node-RED can be seen from the Figure 3.35.

$$\boldsymbol{WCI = 13.12 + 0.6215 \times T_a - 11.37 \times V^{0.16} + 0.3965 \times T_a \times V^{0.16}} \qquad (3.7)$$

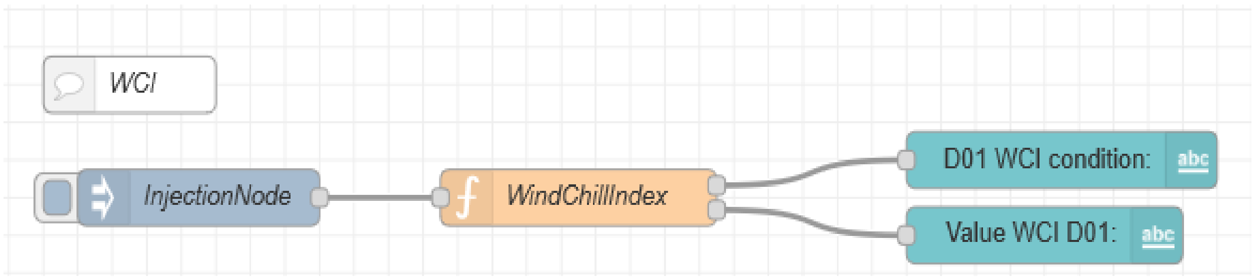
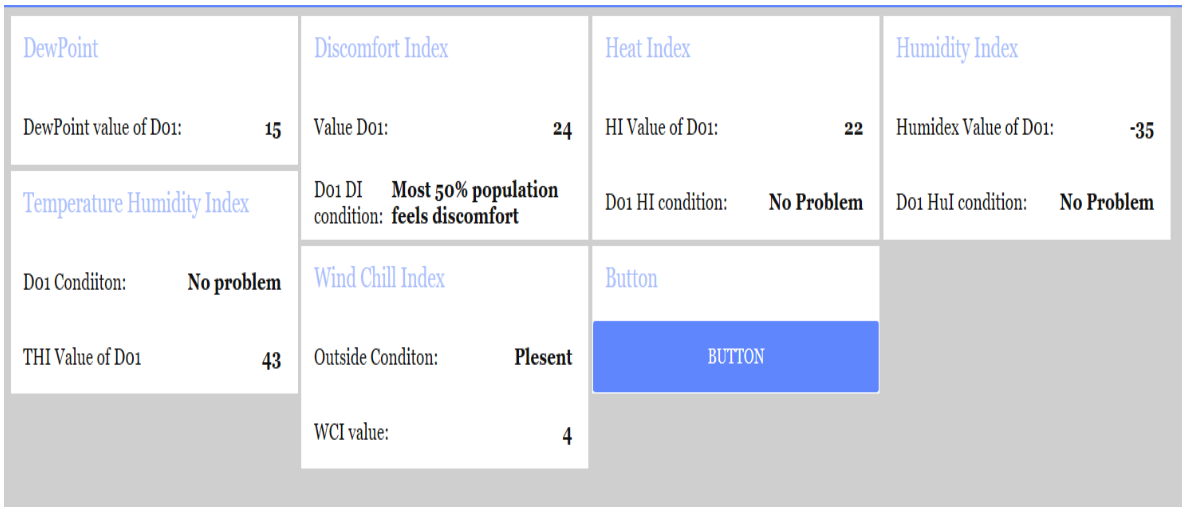


Figure 3.35 Node-RED configuration for calculating WCI

Range of WCI Index	Scale of Comfort
-1.11 to -9.4	Low Risk
-9.4 to -26.11	Moderate Risk
-26.11 to -45.56	Cold
-45.56 to -59.44	Extreme
-59.44 below	Very Extreme

Table 3.6 Range and Scale of WCI Index



DewPoint value of D01: 15

Value D01: 24

D01 DI condition: Most 50% population feels discomfort

HI Value of D01: 22

D01 HI condition: No Problem

Humidex Value of D01: -35

D01 HuI condition: No Problem

D01 Condiiton: No problem

THI Value of D01 43

Outside Conditon: Plesent

WCI value: 4

Figure 3.36 Dashboard View of Comfort Index for a classroom

3.5.7 Summary of Comfort Index results from the dashboard point of view

The above all section explains how to calculate the comfort indices in Node-RED and all the comfort indices values are displayed in the dashboard we can see this form the Figure 3.14. By going through Figure 3.36, we can notice that from the "discomfort index," it shows that "most 50% population feels discomfort" because of the controller where not well tuned at the moment of reading.

3.6 Application

The primary purpose of collecting the data is to utilize it and find the trend, and based on the data, we can decide and to calculate other correlated values, and we are developing some application collected data, etc.

3.6.1 People occupancy

From section 3.4.6. So to find out the people occupancy we need to utilize the values from the people in and people out values and idea behind to calculate the people occupancy is just simple math we need to make the difference between the people in and people out. For doing this, we do not want to connect the database again. As explained in section 3.4.5.2 and 3.4.5.3, these variables are already set as a global variable, so we can call the global variables via getting function and find the difference between them to see the occupancy, and these values are displayed in the dashboard. The program configuration is shown in Figure 3.37.

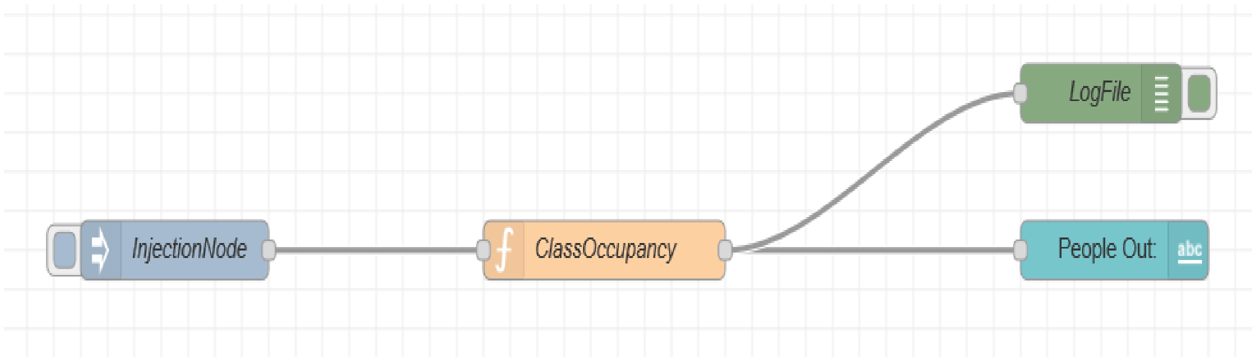


Figure 3.37 Node-RED configuration for calculating People occupancy

3.6.2 Janitor service

Again, with the help of the section 3.4.5, we can develop janitor service, the idea behind this is we can automate the process of scheduling the cleaning service. To implement this, we need to count the people with the help of the value from the people detection sensor, and we will set the threshold to indicate the cleanliness of the area as clean, somewhat litter, an area completely litter. Moreover, this implantation in Node-RED is shown in Figure 3.38

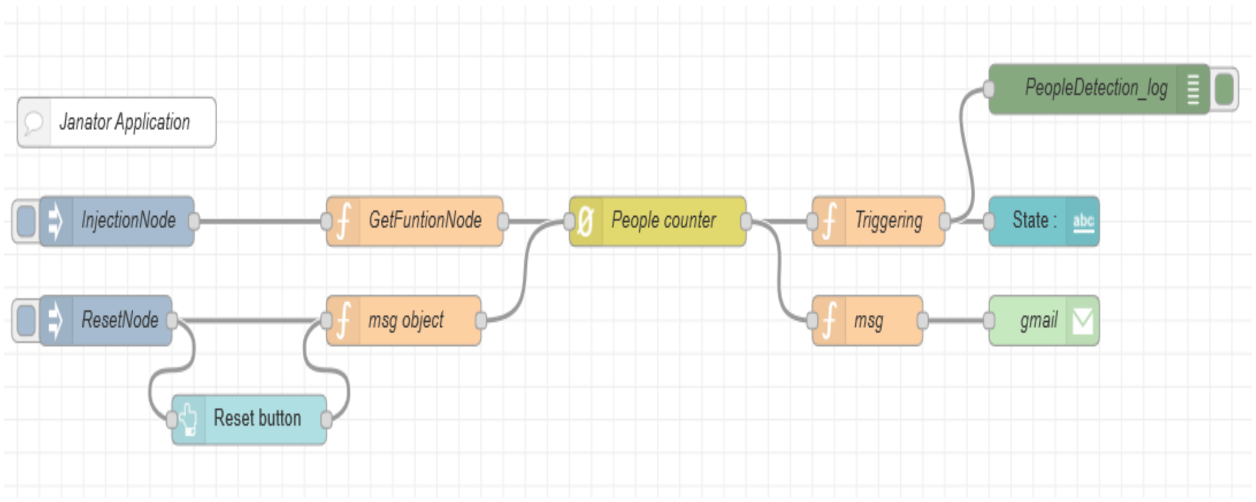


Figure 3.38 Node-RED configuration for janitor service

3.6.3 Average Floor values

3.6.3.1 Average floor temperature

The average floor temperature can be done by getting the values of the classroom on each floor. For the sake of the documentation, The configuration is just for the ground floor (floor 0), and the same type of setup is done for the rest of the stories.

The average floor Node-RED configuration is shown in Figure 3.39,

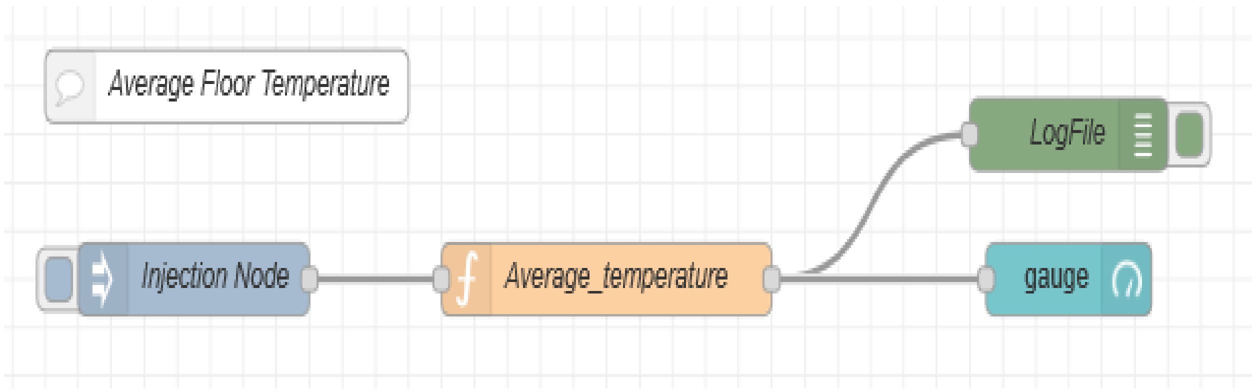


Figure 3.39 Node-RED configuration for average floor temperature

3.6.3.2 Average floor Humidity

The average floor humidity can be done by getting the values of the classroom on each floor. For the sake of the documentation, The configuration is just for the ground floor (floor 0), and the same type of setup is done for the rest of the stories.

The average floor Node-RED configuration is shown in Figure 3.40,

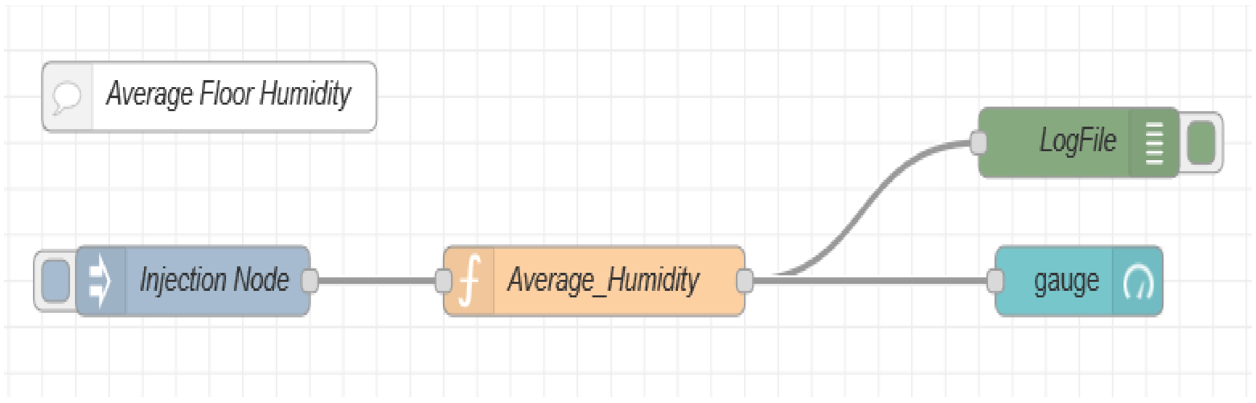


Figure 3.40 Node-RED configuration for average floor humidity

3.6.3.3 Average floor Carbon dioxide level

The average floor carbon dioxide can be done by getting the values of the classroom on each floor. For the sake of the documentation, The configuration is just for the ground floor (floor 0), and the same type of configuration is done for the rest of the stories.

The average floor Node-RED configuration is shown in Figure 3.41,

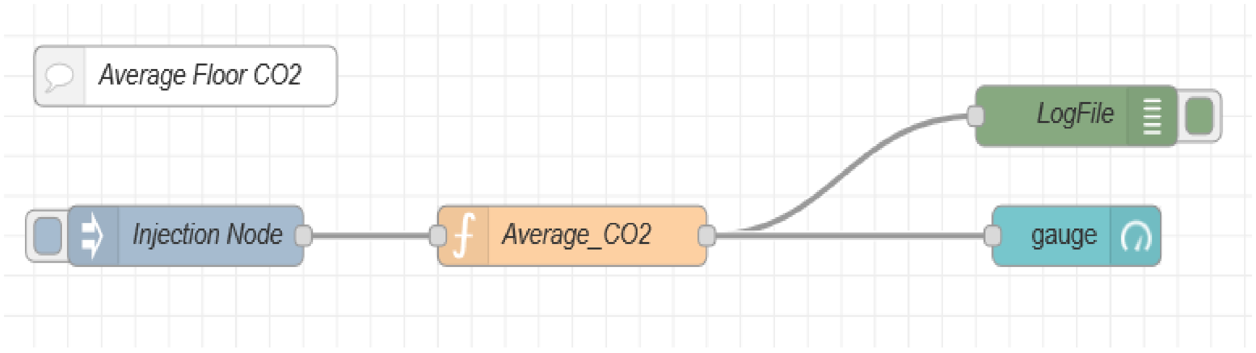
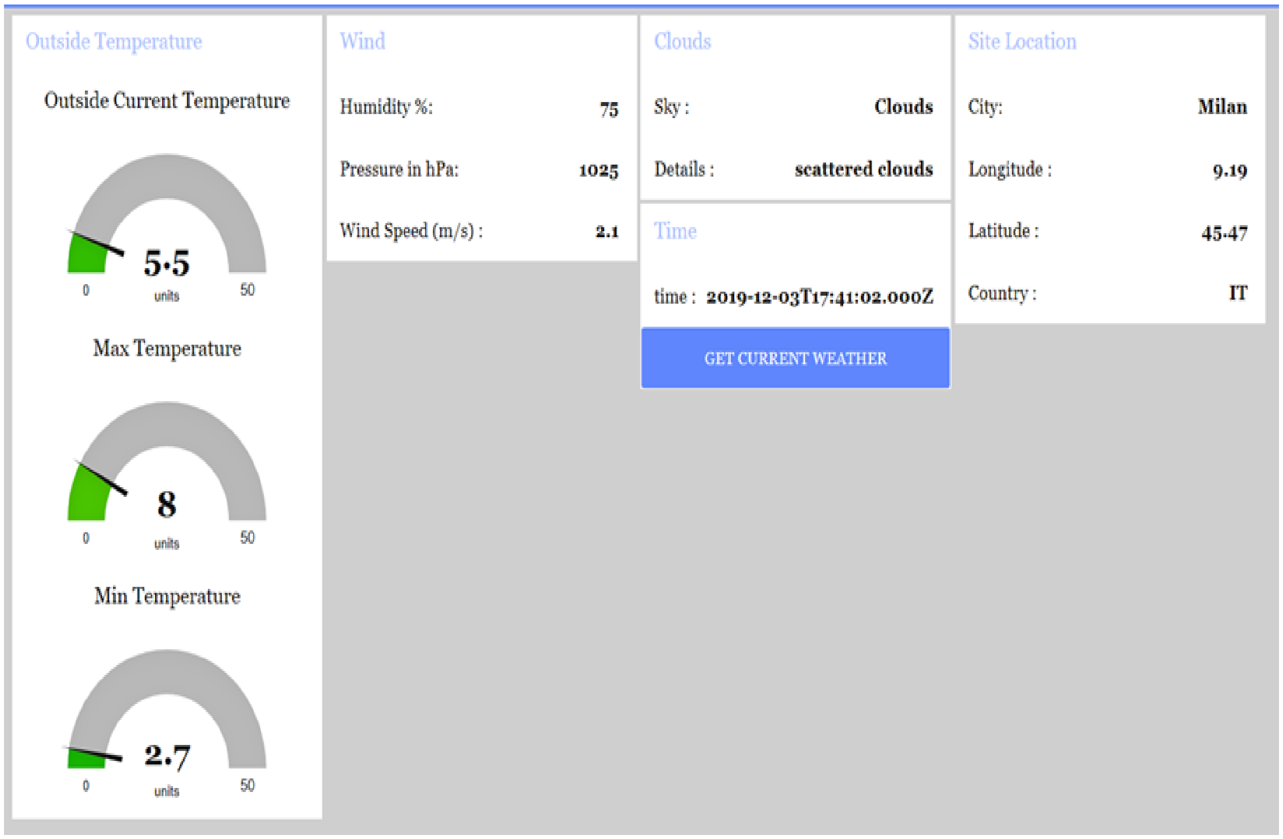


Figure 3.41 Node-RED configuration for average floor carbon dioxide

3.6.4 Obtaining weather from Online

The “Openweather.Ltd” provides an online weather forecast and to get this forecast inside we are using an Openweathermap node and configuring them with an API from the open weather website the obtained data will be in an object and so to split the data in here, we are using the function node to split the values and pushes the split value to the dashboard. This configuration can be seen from Figure 3.42, and the user dashboard can be seen from Figure 3.43.



Humidity %: 75
Pressure in hPa: 1025
Wind Speed (m/s) : 2.1
Sky : Clouds
Details : scattered clouds
time : 2019-12-03T17:41:02.000Z
City: Milan
Longitude : 9.19
Latitude : 45.47
Country : IT

Figure 3.42 Dashboard view of open weather forecast

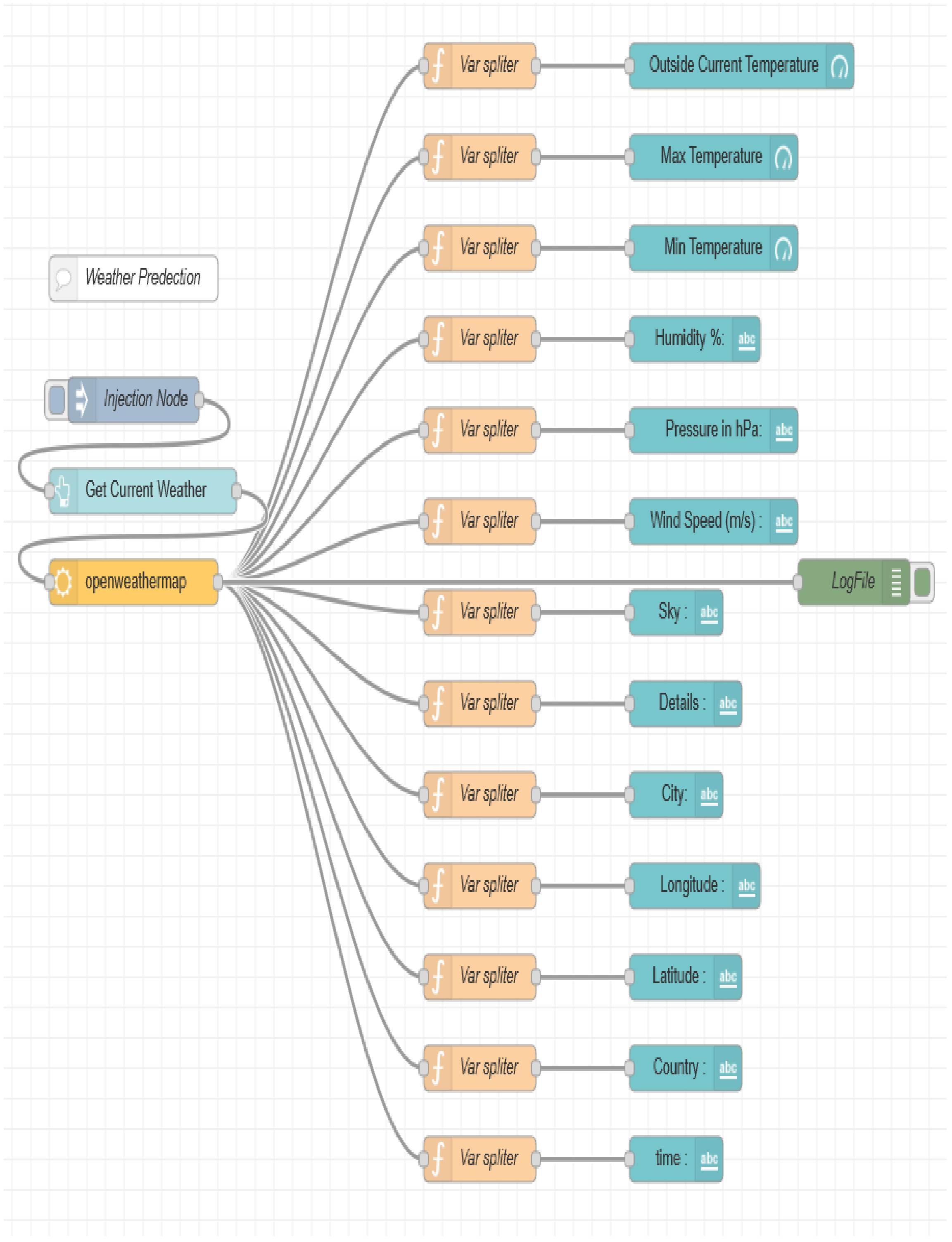


Figure 3.43 Node-RED configuration for obtaining the weathercast from open weather map

4 MOBILE DEVICE DEPLOYMENT

This section will explain how to install the Node-RED IoT platform in the android platform and how it can be used for our application. The section says about the endless possibilities in the mobile platform, and the work is not carried in-depth as it carried in chapter 3.

4.1 Installation of Node-RED in android platform

To use the Node-RED in an android platform, we need to install the "Tremux app" from the android app store and use the flowing command lines at the prompt window.

```
apt update
apt upgrade
apt install coreutils nano nodejs
npm i -g –unsafe-perm node-red
node-red
```

Then after installing the Node-RED, we can point a browser that is on our phone to http://localhost:1880.

4.2 Flow editor and dashboard in android device

The configuration is done in the PC, and the same program can be imported to the mobile platform. However, we want to limit our self while using less function than using in pc. From the general idea, the mobile device needs to be configured for application purposes like for janitor where he/she can see the state of floor cleanliness. The visualization of the Node-RED flow editor in the mobile platform can be seen from Figure 4.1, and the dashboard is seen in Figure 4.2. In a nutshell, this chapter is to show that there are possibilities to work on an IoT platform on a mobile device.

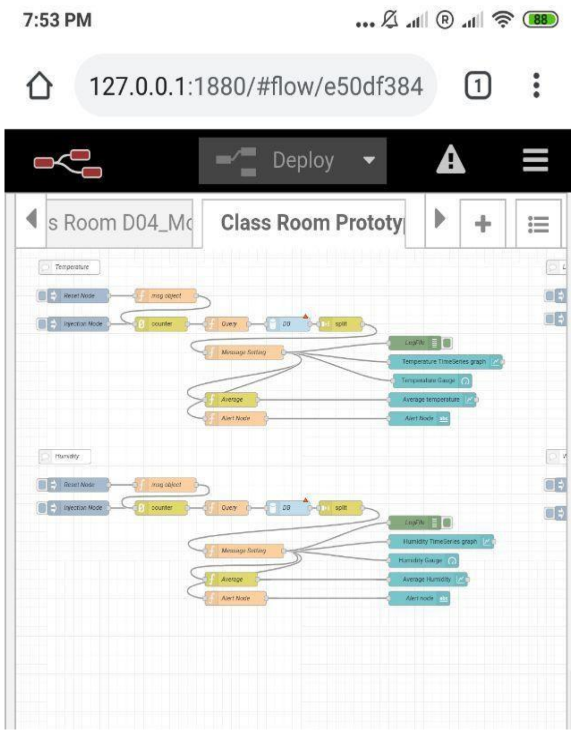


Figure 4.1 Mobile view of Node-RED flow editor

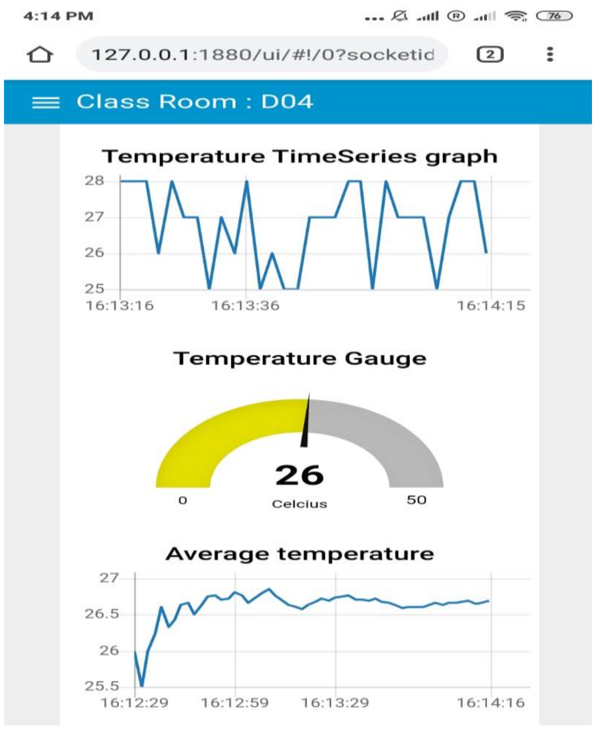


Figure 4.2 Mobile view of Node-RED dashboard

5 CONCLUSION

This chapter summarizes the results of the thesis work and explains the possible future works.

Making the traditional building as “Smart Building” we are efficiently utilizing the resources without wastage. The point of this thesis observing the physical parameters such as temperature, humidity, etc., make them visualize in GUI with the aid of the IoT platform. By monitoring the data, we can have a general idea about the classroom condition, and we can improve the classroom environment if needed, and also, we can ha a prediction about the power usage in a day.

5.1 Possible future works

In the future, if the university decides to make every building on the campus to be smart, then the Amazon web service can be used as a possible choice of the platform; the same work can be shifted and can be scaled for the use.

Utilizing AWS, we can use machine learning or artificial neural networks to analyze the data. And then, this analyzed data can be used by the controllers if they need it.

6 BIBLIOGRAPHY

[1] The ThingsBoard Authors, "ThingsBoard Documentation," ThingsBoard , [Online]. Available: https://thingsboard.io/docs/.

[2] Node-RED, "Documentation," IBM, 2016. [Online]. Available: https://nodered.org/docs/.

[3] J. G. Allen, P. MacNaughton, U. Satish, S. Santanam, J. Vallarino and J. D. Spengler, "Associations of Cognitive Function Scores with Carbon Dioxide, Ventilation, and Volatile Organic Compound Exposures in Office Workers: A Controlled Exposure Study of Green and Conventional Office Environments," *Environmental Health Perspective,* vol. 124, no. 6, pp. 805-812, 2016.

[4] D. N, B. HL and B. CM, "The Importance of Humidityin the Relationship between Heat and Population Mental Health: Evidence from Australia," *PLOS ONE,* pp. 1-15, 2016.

[5] S. A. Samani and S. A. Samani, "The Impact of Indoor Lighting on Students' Learning Performance in Learning Environments: A knowledge internalization perspective," *International Journal of Business and Social Science,* vol. 3, no. 24, pp. 127-136, 2012.

[6] B. Goswami, D. Y. Hassan and D. A. J. Sarma, "The Effects of Noise on Students at School: A Review," *International Journal of Latest Engineering and Management Research ,* vol. 3, no. 1, pp. 43-45, 2018.

[7] M., Puteh, M. h. Ibrahim and M. Adnan, "Thermal Comfort in Classroom: Constraints and Issues," *ELSEVIER,* vol. 46, pp. 1834-1838, 2012.

[8] L. A. Wood, "The Use of Dew-Point Temperature in Humidity Calculations," *JOURNAL OF RESEARCH of the Notional Bureau of Standards ,* pp. 117-122, 1970.

[9] J. Mazon, "The influence of thermal discomfort on the attention index of teenagers: an experimental evaluation," *International Journal of Biometeorology,* vol. 58, no. 5, pp. 717-724, 2014.

[10] A. GB, B. ML, and P. RD, "Methods to Calculate the Heat Index as an Exposure Metric in Environmental Health Research," *Environmental Health Perspectives,* vol. 121, pp. 1111-1119, 2013.

[11] R. Osczevski and M. Bluestein, "The new Wind Chill Equivalent temperature chart," *American metrological Society,* pp. 1453-1458, 2005.

[12] D. Goodman and M. Morrison, JavaScript Bible, Sixth Edition, Indianapolis: Wiely Publishing, Inc., 2007.

[13] K. Henderson, The Guru's Guide to Transact-SQL, Boston: Addison-Wesley Longman Publishing Co., Inc., 2000.

[14] Z. Chaczko and R. Braun, "Learning Data Engineering: Creating IoT Apps using the Node-RED and the RPI Technologies," in *16th International Conference on Information Technology Based Higher Education and Training (ITHET)* , Ohrid, Macedonia, 201`7.

[15] "IoT sensor integration to Node-RED platform," *17th International Symposium INFOTEH-JAHORINA (INFOTEH),* pp. 185-189, 2018.

[16] "IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems," *IEEE Internet of Things Journal,* vol. 4, pp. 269-283, 2017.

[17] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of Things (IoT) communication protocols," in *8th International Conference on Information Technology (ICIT)* , Amman, 2017.

[18] X. F, Y. LT, W. L, and V. A, "Internet of things," *International Journal of Communication System,* 2012.