# A Remote Thermostat Control and Temperature Monitoring System of a Single-Family House using openHAB and MQTT

Md. Habibur Rahaman*, M. Tariq Iqbal

***Abstract* — In this research, an open-source IoT platform named openHAB smart home automation is used as a home server, an ESP32 Thing microcontroller board is used to design a remote-control system for a thermal energy storage system. It consists of temperature sensors for real-time temperature monitoring, ESP32 Thing board is used for data receiving, processing and sending it to the MQTT broker, openHAB software installed in the personal computer is used as a home server for creating dashboard panel, MQTT broker is used to establishing the communication in between openHAB home server and ESP32 Thing board, Wi-Fi router is used to create the communication channel, a battery-powered remote-controlled heater with a digital thermostat is used as a testing device where user can set the desired temperature for house heating. The main objectives of this work are to design a low-cost monitoring and control system for thermal energy storage systems, to monitor the real-time temperature data, to design a control system for thermostat settings with the following features such as manual/automatic operations, local/remote control options. The user can access the dashboards locally via any computer and remotely via openHAB Cloud console from anywhere in the world. The proposed system in this work will help residence to manage their heating systems smartly in a cost-effective way, which will be the replacement of the conventional thermostat settings. The utility provider company can also use this system to control the thermostat settings from centrally, wirelessly, and remotely.**

***Index Terms* — open HAB home automation, MQTT, IoT, Sparkfun ESP32 Thing, Remote control, Remote Monitoring.**

## I. Introduction

The space heating system and domestic water heating system is prevalent and used in every house in Canada. Conventionally, the wood, gas, and oil-based heating system have been used, and even nowadays, in some regions, these are being used. In a typical single-family house, the standard electric heating components are a small water tank integrated with the electric heating element, electric heating resistance elements, sensors, a small control system to turn on and off the auxiliary heater. House demand-side management is essential to supply continuous space heating and domestic water heating in various weather scenarios. In cold climate countries like Canada, people are paying high electricity bills monthly. But the electric-based thermal energy storage systems consume a lot of grid electricity; thus, grid energy saving and greenhouse gas (GHG) reduction are the primary concern in Canada [1].

In our previous research [1], [2], we proposed the solar photovoltaic and solar thermal collector based short term and long term (sessional) thermal energy storage systems. There is some one-time investment cost for system modification and solar photovoltaic or thermal collector installation. Still, the rate of return was acceptable (5~7 years), and the system saves grid electricity and GHG.

The system components sizing has been conducted considering the housing demand and weather. The necessary control system has also been proposed and simulated. Without proper control, these losses can not be minimized to apply the effective operation with proper monitoring and control systems.

To address this effective and robust controller, we developed a monitoring system with some sensors to monitor the sensing parameters such as ambient temperature, indoor house temperature, tank water temperature. Similarly, based on the sensor value, a control system has been developed with a low-cost open-source IoT platform named openHAB home automation, which can control the thermostat settings, to turn on/off of auxiliary heater. A demonstration has been presented in this research. The paper organized as follows: In section 3, the details literature review and the latest IoT based automation, control projects have been presented. We presented the problem statement, and how this the proposed system is practically meaningful. In section 3, we presented the latest IoT thermostat control technology and highly popular thermostat which is available in the market. Their market price, operating principle, pros and cons have also mentioned shortly. In section 4, the proposed system architecture, system development approach, the system requirement and specifications have been listed. In section 6, the details description of the proposed system components, their ratings, connection procedure and requirements, short working principle have been presented. The step by step openHAB IoT setup in a personal computer windows, configuration, connection with MQTT broker have been conducted in section 6. The JAVA programming methodologies and details development have been presented there based on the practice experience. In section 7, the proposed system experimental setup, component connection, testing and validation, how to monitor the sensor values and control the thermostat have been presented. In section 8, the proposed system performance, cost and power consumption analysis have been conducted and compared with the existing technology. The strength and highlighted features of the

Published on September 17, 2020.
Md. Habibur Rahaman, Memorial University of Newfoundland, Canada.
(corresponding e-mail: mhrahaman@mun.ca, tariq@mun.ca)
M. Tariq Iqbal, Memorial University of Newfoundland, Canada.

proposed system has been presented. The conclusion and future work has been drawn at the end.

## II. Literature Review

There are several published works on solar thermal energy storage system design and control to ensure an efficient system and reliable control. The system components monitoring and control can reduce system losses and improve system efficiency. Authors in [3] developed an optimal control system of integrated heating and cooling systems that are capable of reducing 5-11% electricity cost depending on the electricity price, weather, and size of the storage tanks. Still, the details system cost calculation is absent. A model predictive control (MPC) has been applied to an on-grid photovoltaic (PV) based heating and cooling energy storage systems in [30], where the authors monitored the operations for 24 hours and achieved 58% energy savings.

Similarly, in [4], the authors proposed another MPC control system incorporated with the artificial neural network and saved 29% operating cost but presented only 1-hour data and did not present the monthly and sessional. An optimal control algorithm has been applied in borehole thermal energy storage systems in various weather scenarios to supply continuous heat and cold [4]. Authors in[5] presented a fuzzy logic-based control system considering weather information and heating demand. Basically, in that research, they did energy management. Similarly, there is more similar simulation-based research available; however, very few research available in hardware applications. The energy storage system can be monitored and controlled by using open source Internet of Things (IoT) such as Thinger.io, ThingsSpeak, openHAB home automation, Home Assistant platforms, and so on. Similar to software simulation, there are several types of research available with the application of single-family residential house apparatus control and monitoring. For instance, authors in [6], [7], designed an IoT based system with Raspberry Pi device with the integration of openHAB platform with integration of hypertext transfer protocol (HTTP) networks, which can monitor the system components remotely, but the system is not customer friendly. Another research[8] presented the design and implementation of the wireless module with openHAB GPIO binding, Raspberry Pi, Arduino, NodeMCU, which is also capable of monitoring the sensor values and controlling the household supporting devices. Other studies in [9] proposed a single-family house demand-side management systems with openHAB platform with various communication protocols such as HTTP and use datagram protocol (UDP) and energy sharing based algorithm. The research conclusion was to monitor and control of house devices, but they did calculate the total device energy consumption and cost. Similar to the above research, some sensors, MQTT protocol, Raspberry Pi, IoT based home assistant, and openHAB platform has been used for home devices automation[10]. Finally, the authors concluded the remote controllability, and which one would be better in between home assistant or openHAB? In a Germany based openHAB conference, the authors in [11] fully automated the most significant public Building using the openHAB platform. Paper[12], [13] also highlighted the security concern of openHAB and open-source IoT based systems.

In 2018, the University of Tartu published a master's thesis [14] about the security concern of openHAB, such as authentication and authorization. Another latest research published in 2019 [15], where the authors proposed a home device control system with ESP32, openHAB, MQTT protocol, that's the almost similar research like me, but the did not mention the system energy consumption, energy management systems and system cost.

There are also other similar works[16], device control using mobile phone or web through Arduino, Bylink, relay, MQTT, and so on.  Another application of IoT is the monitoring of refrigerated temperature is mentioned in [16]. Similarly, authors in [17]-[19]proposed a web-based control system and an intelligent thermostat using IoT, MQTT, Arduino, but they did not work with open source cloud control capabilities.

In this paper, the authors proposed a low-cost open-source IoT based remote monitoring and control system for a thermal energy storage system. Where a thermostat has been controlled from locally and remotely, and the energy storage systems temperature has been monitored by using the openHAB home server dashboard, MQTT establishes the connection in between openHAB home server and Sprukfun ESP32 Thing microcontroller. The authentication and system security has also been ensured and also worked with the open-source cloud development.

## III. Available Technologies in the Market

Several products on the market are working as the same functions, such as Google Nest thermostat, Honeywell thermostat, ecobee thermostat, Sinop thermostat, Lux Kono thermostat, Johnson controls the thermostat, ThermoPro digital hygrometer thermometer, Mysa smart thermostat and so on. Among all of them, Google nest thermostat is highly popular.

### *A. Google Nest thermostat*

It first developed in 2011 form Google bought nest labs, and now the 3rd generation is available in the market, which one brilliant way to control the heating and to cool simply. The compact hardware packages wiring is simple as well, the common point is C, and 24-volt wires can be connected with the other. This smart thermostat can support multi-stage, heat pump, and multi-zone HVAC systems, but it does not like geofencing and expensive, $329 market price [20].

### *B. Honeywell thermostat*

This allows local and remote control options through a computer, tablet, or smartphone. In the digital screen, the humidity, temperature, and thermostat settings options are available. But the worst Thing is that it does not include the wall plate for installation, wiring is too complicated, which can cause expensive equipment damage, it is a non-programmable device as well, and its price is around $149[20].

### *C. Ecobee thermostat*

It is compatible with temperature and humidity sensors, and in the automatic mode, it is estimated that around 23% annual savings are possible in heating and cooling. Similar to another thermostat, the house components can be controlled locally and remotely using a laptop and mobile phone, and its cost is in between the above two thermostats, which is around $229[20]. The main issue with that is it has only one temperature sensor and two occupancy sensors build in at Ecobee4 systems.

These are the most popular available smart and Wi-Fi thermostat, which is used commonly in the single-family house for room heating and cooling application. However, there are other manufacturer's smart thermostats available at different prices.

## IV. System Description

### *A. System Architecture*

Lots of wireless smart thermostats are available in the market that already discussed in section 2 what market price is high, and some thermostat wirings are not so simple.

Some other PLC applications are also available what require particular installation, and the majority of circuit elements, sensors, and relay are connected with other integrating technologies. Those technologies are also useful; however, in this study, we are proposing a smart way, cost-effective structure of house heating and cooling applications, temperature measurement, and monitoring[8]. The openHAB home assistant open-source IoT platform is a highly popular and designer choice IoT platform. It is free and easy to install and easy to integrate with an electronic microcontroller such as Arduino, Sprukfun ESP32 Thing, ESP8266, and so on. Similarly, it is so simple to connect solid state relay and DHT11 temperature and humidity sensors and other analog temperature sensors with those devices. The working system architecture is shown in Fig. 1 below: The proposed system architecture is designed to reduce the complexity and cost of implementation. No expensive and complicated components has been used in this research such personal computer, mobile devices, Wi-Fi router, which is available in every single-family house in Canada.

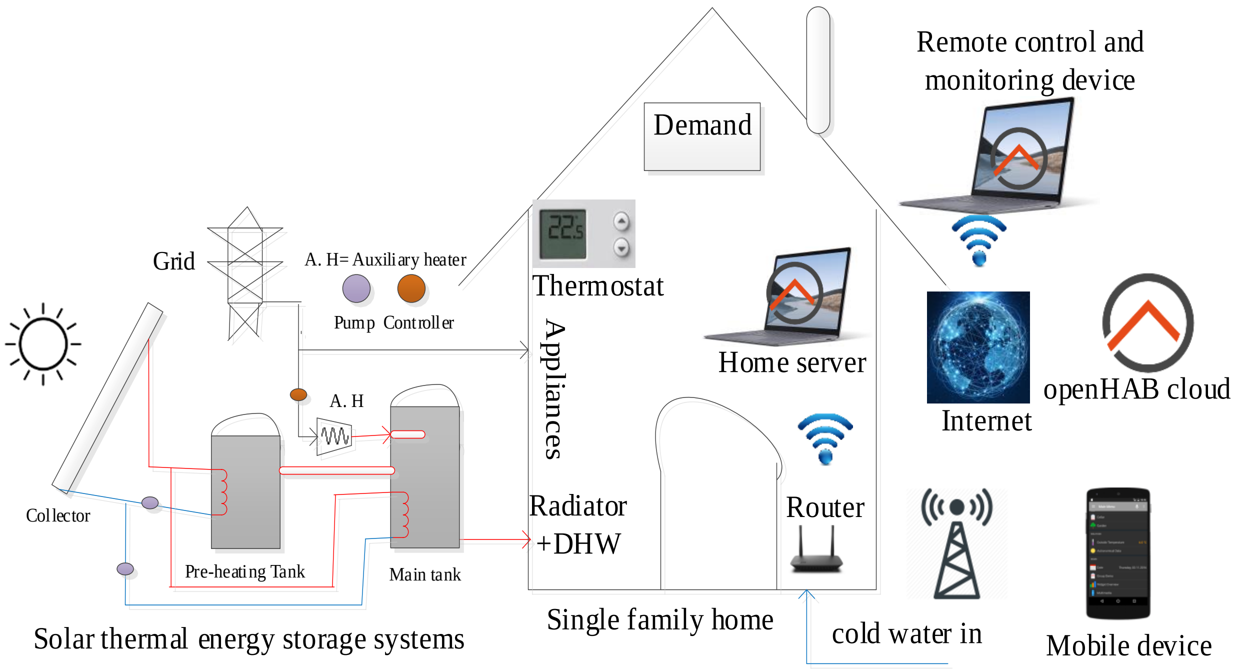


Fig. 1. the system architecture.

The system is designed as user-friendly, low power consumption, low cost, allowing users to use a traditional thermostat, to monitor the room, ambient temperature, and to control the thermostat settings locally and remotely. The users can monitor temperatures and can manage the thermostat settings using their laptops or mobile devices through the local available Wi-Fi network. Through the portable device uses communication services such as GSM, with access to the internet, a user can have access to the home server using openHAB apps or openHAB Cloud and can do the same. The other family member can also have access to the home server using their laptops and monitor and control the thermostat settings. The system architecture has a flexible user interface that can be customized based on user requirements, such as indoor room temperature settings. As the server has no SIM card or separate IP address, so the users do not need to memorize. The system components setup flow is shown in Fig. 2.

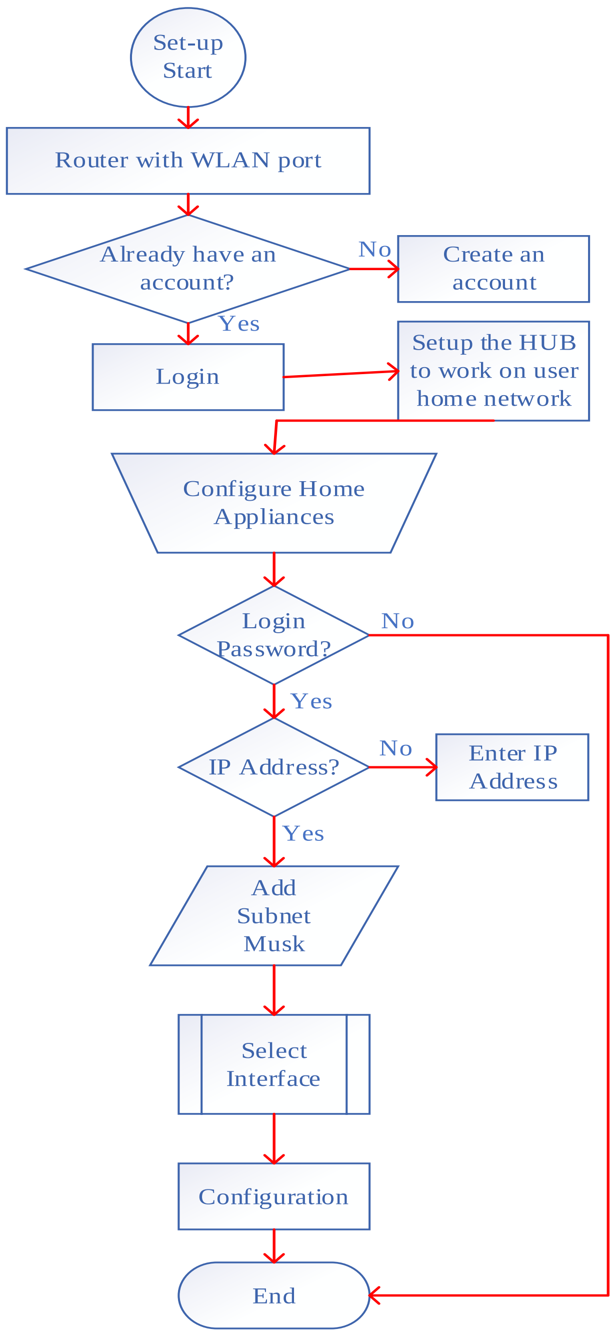


Fig. 2. The Wi-Fi setup and other components setup.

## B. *System Development Approach*

The openHAB home automation is a very user-friendly open-source IoT platform software that is easy to set up at Windows, RaspbberyPi, Linux, Windows, macOS, openHABian, PINE A64, Armbian, Docker and so on. But in this work, we installed openHAB in a laptop with Windows operating systems. It is using the same process of the computer, no need to buy RaspberyPi and pay an extra dollar or RaspberyPi software version is available freely, which is also same as the hardware RaspberyPi. The performance of openHAB with windows installation is excellent as the home server, and there is no need for any extra third party software or hardware. Arduino or ESP23 board is also another microprocessor where the sensors are connected, which is made with the reply, temperature sensors. The temperature sensors measure the environment temperature and sent it to the microcontroller, the ESP32, or the Arduino microcontroller. The microcontroller is connected with the Wi-Fi network [9], so it embedded the data and sent it to the openHAB windows home server. Similarly, the thermostat control signal came from the openHAB windows home server to the Arduino microcontroller through the Wi-Fi network; thus, the solid-state reply controlled, and thermostat settings controlled. The detailed system block diagram is given below in Fig. 3.

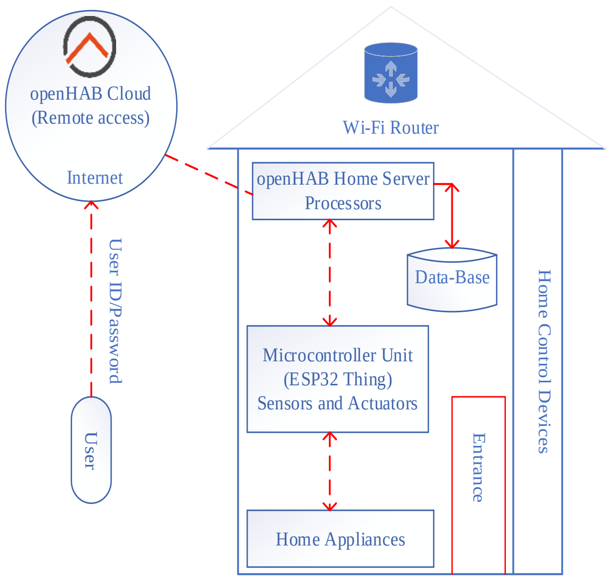

openHAB Cloud (Remote access)
Internet
User ID/Password
User
Wi-Fi Router
openHAB Home Server Processors
Data-Base
Microcontroller Unit (ESP32 Thing) Sensors and Actuators
Home Appliances
Entrance
Home Control Devices

Fig. 3. The overview of system development.

## C. *System Specifications and Requirements*

The proposed monitoring and control systems have some requirements which are given below:

- ➢ All sensors, microcontroller devices must be low power consumption, energy-efficient, low cost, smaller sizes.
- ➢ The home server should be easily accessible from locally or remotely. There is no need to memorize the computer or network IP address or SIM number to access the home server.
- ➢ The system must be secured enough to ensure that there will be no third-party intrusion and the safety of the householder.
- ➢ For security and safety operations, just a user login and credential and password will be well enough to access and control the whole systems.
- ➢ The systems should be controllable from anywhere in the world.
- ➢ The system architecture should be simple so that every illiterate user can use these systems.
- ➢ Limit the amount of power an appliance can consume.
- ➢ Update the user on the state of appliances, either running or not.

## V. Components Description

The proposed control and monitoring system are made up of sensors, ESP32 Thing board, relay, digital thermostat, and so on. The temperature sensors have been used for data accusation. Similarly, SparkFun ESP32 Thing micro-controller is a primary device used for data receiving, processing, and transmitting to other hardware. A solid-state relay is used to turn on/off the heater, and a heater with digital thermostat used a plant which is capable of indicating the status of room temperature. To create Wi-FI Router for local Wi-Fi network creation (communication channel), and openHAB home automation server is a local IoT server with a graphical user interface (dashboards) for monitoring the sensor's data and controlling the thermostat locally and remotely. The details description of every component are given below.

## A. OpenHAB Home Automation Local Server IoT Platform

openHAB home automation software is an open-source powerful and user-friendly software for the Internet of Thing (IoT), which can be installed in RaspberyPi, windows, Linux, and so on. It supports the Representational State Transfer (REST) Application Programming Interface (API) which enables controlling and reading of smart devices. REST uses Hypertext Transfer Protocol (HTTP) for communication, and it is an architecture based on the standard of the web. This protocol enables the different machines on the network by considering every component as a resource. The unique features of the HTTP protocol which can connect and manage IoT devices using the automatic discovery of API. openHAB IoT platform is fully supported by GitHub (the highly popular and world-leading open-source development platform). openHAB open-source home automation IoT platform is fully supported with the other software such as MQTT broker and other hardware. openHAB is integrating with Arduino IDE compatible. The following boards which are worked with Arduino IDE, such as Arduino, Arduino+Ethernet Sheild, Arduino+WIfi Sheild, ESP32/8266/13, NodeMCU, TC CC320, and so on. It can operate in Windows and Linux powered devices such as Raspberry Pi (both version), Intel, and any other personal computers working with Windows, Linux running with Ubuntu, or macOS.

In this research, the openHAB software IoT platform has been installed in computer windows, so there is no component associated with it. It can perform well and can communicate well as a home server with all other external microcontrollers at Wi-Fi networks such as Arduino, Sparkfun Esp32 Thing. In this project, openHAB smart home automation software has been installed in C drive of personal computer windows, and openHAB worked as a home server in the Wi-Fi environment, and it will be fully able to communicate with other microcontroller development board. Home server (openHAB) specifications will be the same as the personal computer specifications such as RAM capacity, Processor capacity, HDMI port, Memory capacity, and so on. In this section 8, the authors presented the detailed description of low-cost and low power consumption hardware and software components used in the realization of the proposed openHAB IoT based remote monitoring and control systems design. The ESP32 Thing microcontroller is configured with the MQTT Client to process and publish the sensor data to openHAB and similarly sent the thermostat settings value to the ESP32 microcontroller through MQTT subscribe/publish protocol which is also configured as the MQTT broker as shown in Fig.15. Finally, a Wi-Fi router is used for creating the TCP/IP Wi-Fi connection for the MQTT protocol implementation. The external hardware components associated with this project are described below:

## B. DHT11 digital temperature sensors

The DHT11 is a temperature and humidity sensors, it has a dedicated NTC to measure the environment temperature, and an 8-bit microcontroller inside that can generate the output in the values of temperature as a serial data. The serial pin is connected with one of the digital PIN in ESP32 microcontroller. This is cheap and available anywhere. It can measure temperature from 0 ºC to 50 ºC with an accuracy of ±1°C. It's a small device with a length of 16 mm, and the width is 12.9 mm. DHT11 is available in two different pin configurations, and it may contain 4 pins or 3 pins. The PIN diagram of both DHT11 sensors is shown in Fig. 4.

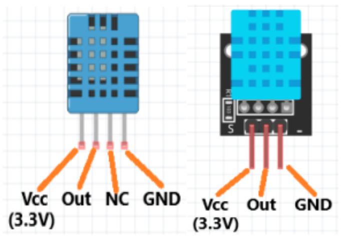


Fig. 4. The pin diagram of DHT11 temperature sensors.

The out (data) pin is connected with an I/O pin of the ESP32 microcontroller digital input pin (pin 34 and 35). Typically, 5 voltage is available in ESP32 think device and power supply board, but DHT11 needs 3.3 V power supply: that's why a 5k pull-up resistor is needed. The out pin (pin 34 and 35) value is a serial data contains temperature and humidity. The library file for DHT11 is available, which has been uploaded to the ESP32 library for successful operation. ESP32 sensors have 3.3 V pin, so the VDD pin of DHT11 is connected to the 3.3 V pin of ESP32. Similarly, the ground of ESP32 and ground of DHT11 are connected. The connection of DHT11 sensors with ESP32 microcontroller is shown in Fig. 5.

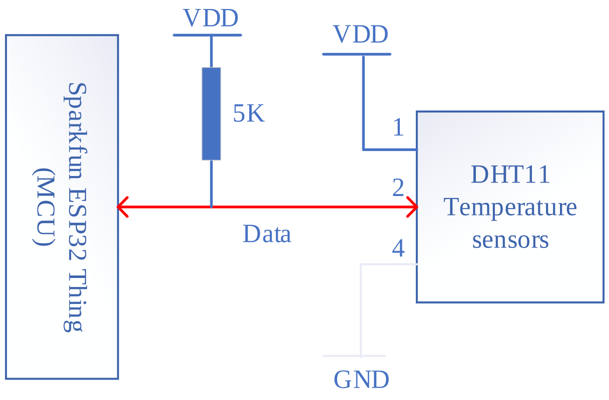


Fig. 5. The connection diagram of ESP32 and DHT11.

The specifications of DHT11 sensors are given below.

- Operating Voltage: 3.5V.
- Operating current: 0.3mA (measuring) 60uA (standby).
- Output: Serial data.
- Temperature tolerable range: 0°C to 50°C.
- Humidity detect range: 20% to 90%.
- Resolution: Temperature and Humidity both are 16-bit.

- Accuracy: ±1°C and ±1%.

### C. *Sparkfun ESP32 Thing Micro-Controller (RTU)*

The Sparkfun ESP32 Thing microcontroller board, manufactured and supplied by the Sparkfun Electronics, is a micro-controller unit that is almost similar to an Arduino development board. It is a Wi-Fi compatible micro-controller with around 30 Input/Output pins, and it supports off Wi-Fi line Bluetooth low-energy such as BLE, BT4.0, Bluetooth Smart. Based on the manufacture report (datasheet), the reason of its name "Thing" is that it is mainly manufactured for the Internet of Things software so that the user can implement and do lots of projects, for example, wireless monitoring and control system development for local or remote control through Wi-Fi network. Compare to all other available IoT supported microcontroller development boards, it has unique characteristics that it has low power consumption (around 0.5W), low cost (about CA$ 20) [21]. There are two ways to supply power to this board with either a 5 V USB power supply cable or with a small lithium-polymer (Lipo) battery. It's operating signal voltage range is 2.2 V to 3.6 V, although the I/O pins of ESP32 Thing board van tolerate 3.3 V. A single Sparkfun ESP32 Thing development board is shown in Fig. 6.

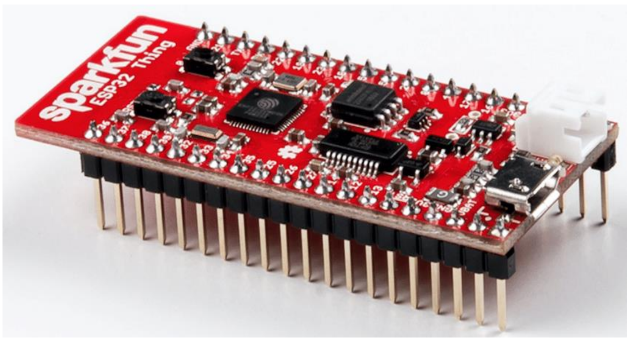


Fig. 6. The picture of Sparkfun ESP32 Thing.

It can be programmed with the Arduino integrated development board (IDE) tools. If the ESP32 board is connected with a personal computer with a USB cable, any program written in Arduino IDE can be uploaded directly to ESP32 micro-controller by selecting the board types and port type. The common name of the Arduino program is Sketches, and typically, it's C/C++ functions. To implement this project, the ESP32 thing microcontroller has been hooked up in a breadboard, and the temperature sensors and solid relay are connected. There two temperature sensors have been used; one is to measure the room temperature, and the other one is to measure the ambient temperature—the C++ program written in Arduino IDE in degree centigrade format and uploaded to the board. The ESP32 thing board has been powered with 5 V USB power cable. The measured and calculated temperatures are displayed in the IDE serial monitor by selecting the specific Baud Rate. Both temperatures will show as a digital form at the user monitoring panel. The Input/Output and analog/digital pin, ground, VCC pins, reset pin, USB connected power supply pin configuration, and details structure of the Sparkfun ESP32 Thing development board are shown in Fig. 7.

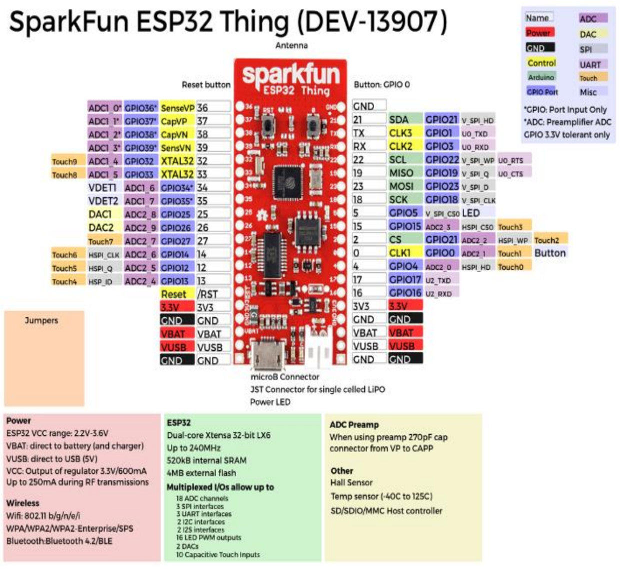

SparkFun ESP32 Thing (DEV-13907)
Antenna
Reset button
Button: GPIO 0
Name
ADC
Power
DAC
GND
SPI
Control
UART
Arduino
Touch
GPIO Port
Misc
*GPIO: Port Input Only
*ADC: Preamplifier ADC
GPIO 3.3V tolerant only
Jumpers
microB Connector
JST Connector for single celled LiPO
Power LED
Power
ESP32 VCC range: 2.2V-3.6V
VBAT: direct to battery (and charger)
VUSB: direct to USB (5V)
VCC: Output of regulator 3.3V/600mA
Up to 250mA during RF transmissions
Wireless
Wifi: 802.11 b/g/n/e/i
WPA/WPA2/WPA2-Enterprise/SPS
Bluetooth:Bluetooth 4.2/BLE
ESP32
Dual-core Xtensa 32-bit LX6
Up to 240MHz
520kB internal SRAM
4MB external flash
Multiplexed I/Os allow up to
18 ADC channels
3 SPI interfaces
3 UART interfaces
2 I2C interfaces
2 I2S interfaces
16 LED PWM outputs
2 DACs
10 Capacitive Touch Inputs
ADC Preamp
When using preamp 270pF cap connector from VP to CAPP
Other
Hall Sensor
Temp sensor (-40C to 125C)
SD/SDIO/MMC Host controller

Fig. 7. The detailed architecture of the ESP32 Thing board.

### D. *Wi-Fi Router (Communication Channel)*

In this monitoring and control system work, the local Wi-Fi serves as the communication channel between the ESP32 Thing (RTU) and the openHAB IoT home server. The high-speed Bell-Aliant Router (Model number Home HUB 3000) has been used to create the local WI-FI network. Its data transfer rate is 1 Gbps, 802.11b/g wireless protocol, and it is IEEE 802.11 standards-compliant. The Router operating frequency is 2.4 GHz, and it has one WAN power and four LAN ports. According to the datasheet of ESP23 Thing (MCU), it can implement TCP/IP full 802.11b/g/e/i WLAN MAC protocol and Wi-Fi direct, so it matched. A local WLAN Ethernet cable has been connected to the LAN port of Router. The Router has been configured to set up the needed local Wi-Fi network. After that, the communication established in between openHAB home IoT server and ESP32 Thing using the server IP address to identify the platform. To ensure the network security, the Wi-Fi Router user credentials (network name SSID and the password has been assigned.

### E. *Digital Thermostat and Heater*

The project is related to temperature monitoring and thermostat control, so these components are the output components where the openHAB thermostat control topology has been applied. It has a digital thermostat, which indicated the running room temperature in degree centigrade in digital format. We selected one digital output port of ESP32 and connected the solid-state relay to turn on and off the heater. The other two digital output pin has been selected in ESP32 Thing are for thermostat control and for setting the thermostat value. In this work, a comfort zone brand heater has been chosen, which has a large easy to read digital display thermostat as well as a remote power by two 1.5 V dry-cell battery. It is a built-in electronic circuit. It is a 23 in black oscillating ceramic tower heater with 1500 watts ratings. It has three temperature settings, lower (800 W), medium (1000 W), and high (1500 W). The connection of heater thermostat and ESP32 Thing is shown in Fig. 8.

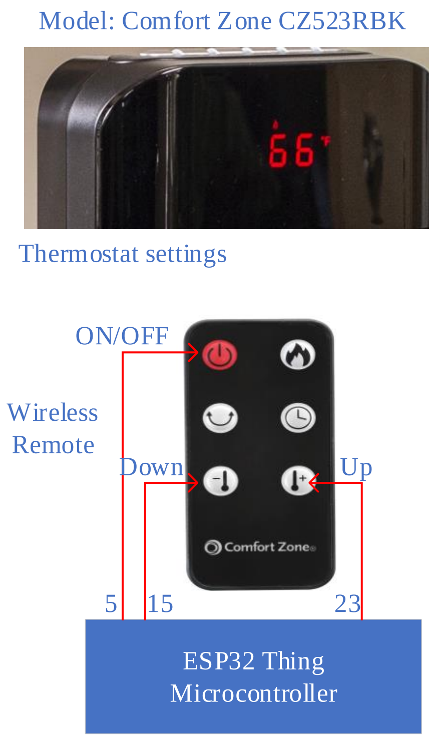


Fig. 8. The connection diagram of remote and ESP32 Thing for thermostat settings.

The remote has two buttons one is used to up the thermostat settings (connected with Pin 15 and 23), and the other one is used to down the thermostat settings. According to the user demand, the user can press these two buttons in remote and wirelessly; the thermostat settings have been changed. These two buttons in remote are connected with the digital output pins of ESP32 Thing to control the thermostat settings from ESP32.

## VI. OPENHAB SETUP AND JAVA PROGRAMMING

Open Home Automation Bus (openHAB) is an open-source IoT platform that is working with the JAVA programming language [22]. There is so many open-source IoT based software available like Thinger.io, ThingSpeak.com, Ubidots, ThingsBoard, Zetta, Node-RED, Flutter, Kaa, and so on. Among all of them, it has some strength such as it is capable of working with Linux, Windows, macOS, Unix, and Solaris operating systems, and it can integrate with other devices and systems, it can provide a uniform user interface and a common approach to automation rules across the entire system, it is more flexible. openHAB first developed in 2010, which is fully customizable and was capable to connects devices and services from various vendors. Now in the 2019 update, it has 300 bindings as OSGI modules, and multiple control options are available such as controlling the lights, relays, and the manual, and automatic controls are also possible in the user interface by triggering the rule developed by JAVA language.

openHAB works on Java virtual machine (JVM), which enables a computer to run Java programs. If the programs are written in other languages that also compiled to Java bytecode, all openHAB Bindings have a separate function. Whatever we need, we need to install that at runtime via OSGi. For storing and querying the sensors data, including relation and time series database, it also supports several persistence backends.

The openHAB IoT platform has a cloud console with an attractive designed front end where a user can monitor and control the connected devices and visualize the connected devices from another computer on the same server or another computer anywhere in the world. To ensure security, a user must log in to the cloud console using his user authentication ID and a protected password to avoid the misuse.

To work with openHAB IoT platform, the first step should be to choose the operating system where to install it, such as Windows, Linux, macOS, Raspberry Pi, Docker, PINE A64.

To start with openHAB platform, the first step is to visit its official webpage and download the openHAB windows software latest version, in this project, the latest release version is Stable 2.5.5 and Snapshot 2.5.6-SNAPSHOT, it is better to download Snapshot latest version.

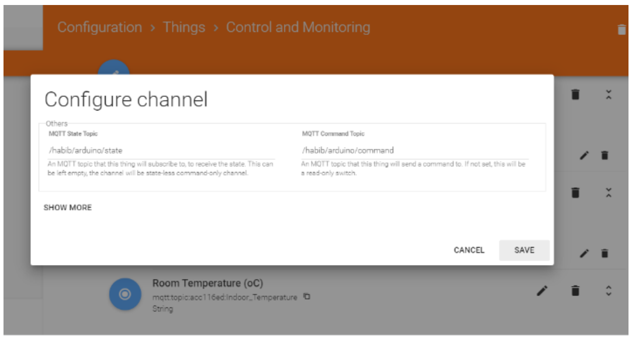
The openHAB is working with JAVA open source development platform, and there are several versions of JAVA platforms available. Among all of them, Zulu 8 is correctly working with openHAB. Then the user should follow the installation steps mentioned in Fig. 9.

### A. *openHAB Configuration*

openHAB is the smartest IoT platform among all of them where the Configuration is also s user friendly, which is capable of connecting all devices available under the Wi-Fi network. Every device is logically and functionally different connected to openHAB. The openHAB-MQTT channel definition is shown in Fig. 10. To represent all of these, openHAB defines the following base components:

TABLE 1: DESCRIPTION OF OPENHAB COMPONENTS

| **openHAB components** | **Short descriptions** |
|---|---|
| Add-ons | To communicate with connected devices |
| Things | Device representation in openHAB |
| Items | Things properties and capabilities |
| Groups | Items collections and categories |
| Sitemaps | User-defined interface to arrange groups, items, and so on. |
| Transformations | Functions to transform user data. |
| Persistence | Store updated data service |
| Rules | It is used to automate the systems. |
| Javascript | Define rule and other runtime objects using Java programming. |



MQTT State Topic
/habib/arduino/state
An MQTT topic that this thing will subscribe to, to receive the state. This can be left empty, the channel will be state-less command-only channel.

MQTT Command Topic
/habib/arduino/command
An MQTT topic that this thing will send a command to. If not set, this will be a read-only switch.

Room Temperature (oC)
mqtt:topic:acc116ed:Indoor_Temperature
String

Fig. 10. openHAB and MQTT Configuration.

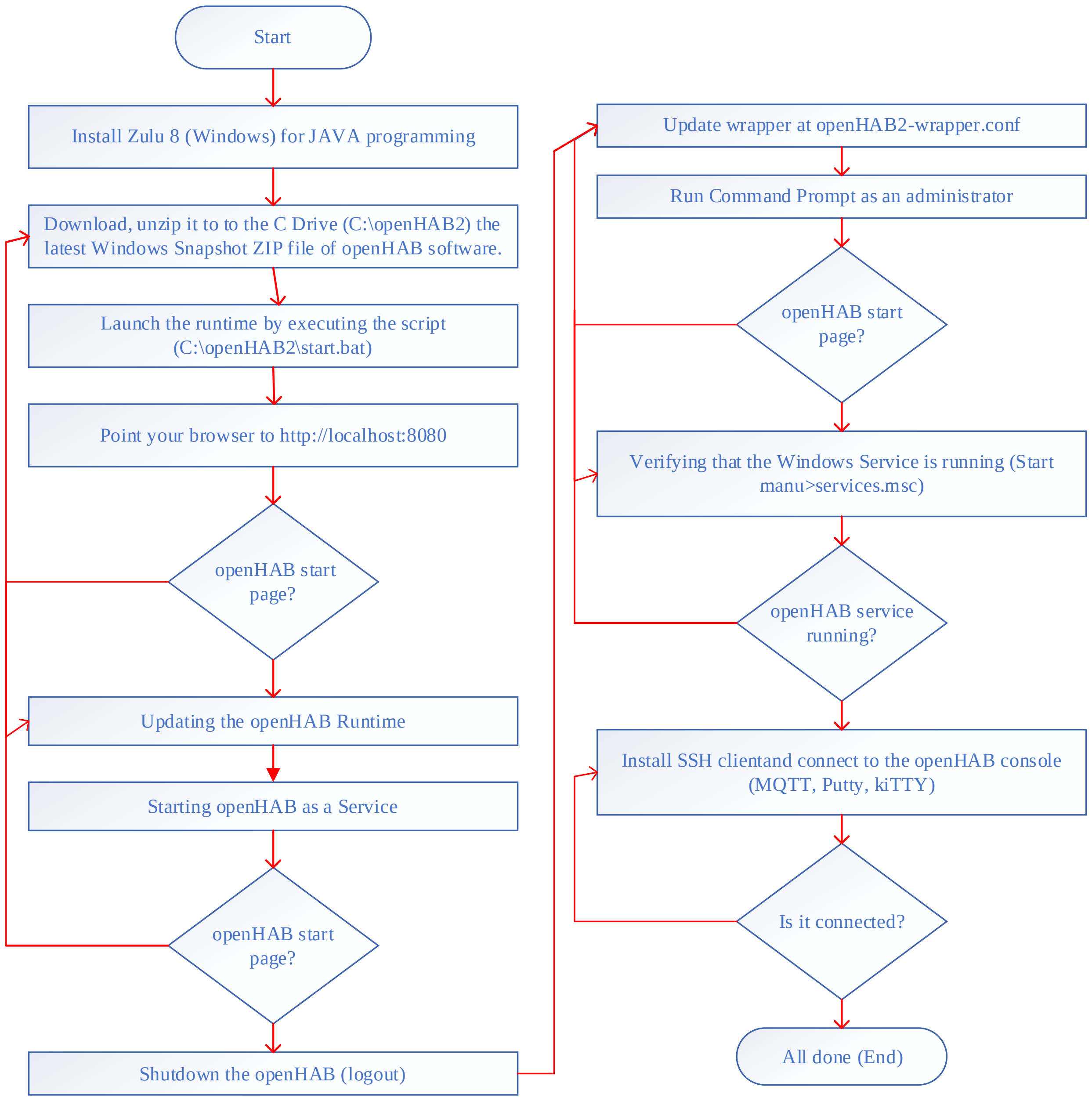

Start
Install Zulu 8 (Windows) for JAVA programming
Download, unzip it to to the C Drive (C:\openHAB2) the latest Windows Snapshot ZIP file of openHAB software.
Launch the runtime by executing the script (C:\openHAB2\start.bat)
Point your browser to http://localhost:8080
openHAB start page?
Updating the openHAB Runtime
Starting openHAB as a Service
openHAB start page?
Shutdown the openHAB (logout)
Update wrapper at openHAB2-wrapper.conf
Run Command Prompt as an administrator
openHAB start page?
Verifying that the Windows Service is running (Start manu>services.msc)
openHAB service running?
Install SSH clientand connect to the openHAB console (MQTT, Putty, kiTTY)
Is it connected?
All done (End)

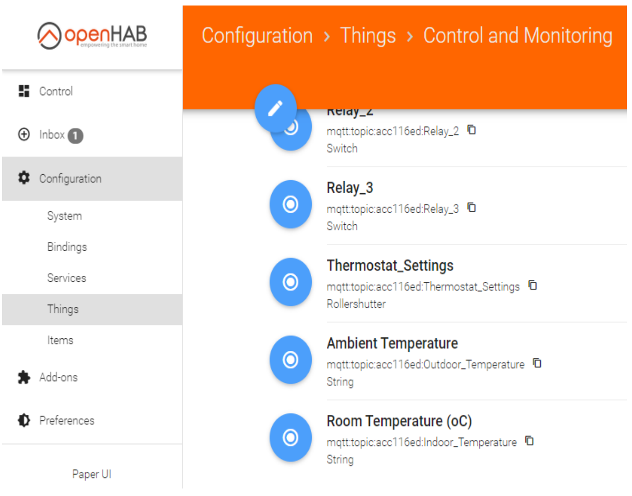
Fig. 9. openHAB installation steps in Windows.

## B. Communication Mechanism

openHAB can communicate electronically with smart and not so smart devices, can perform user-defined actions, and can provide a webpage with user-defined information as well as user-defined tools for interaction with all devices.

It has some specific segments, functions, and operations. The main components of openHAB are described below:

**Channels:** The user can find the channel under the openHAB>PaperUI>Configuration. It is a logical link between a Thing and an Items. The primary function of channels are communication; it originates from Things and communicates with Items or vice versa. During the Thing definition, the user will create channels where items will be lined. Every item has a unique identification number, which we put to the JAVA programming to establish external communication as well with MQTT. Thus the connection has been established between Things and Items. Figure 11 represents the relation between Things and Items and the confirmation of communication establishment in between openHAB and MQTT broker is shown in Fig. 12.



Relay_2
mqtt:topic:acc116ed:Relay_2
Switch

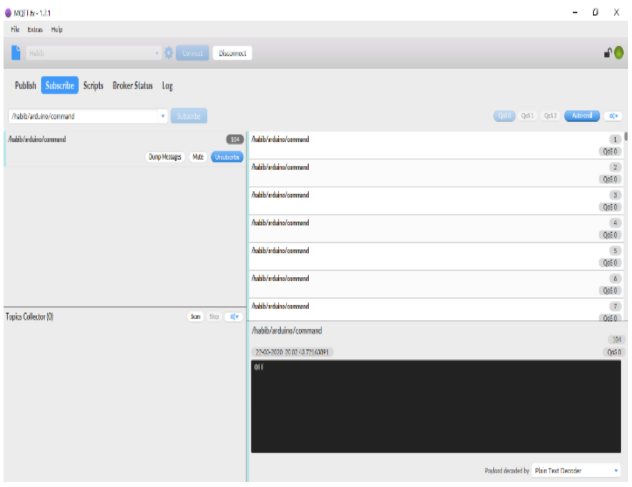
Relay_3
mqtt:topic:acc116ed:Relay_3
Switch

Thermostat_Settings
mqtt:topic:acc116ed:Thermostat_Settings
Rollershutter

Ambient Temperature
mqtt:topic:acc116ed:Outdoor_Temperature
String

Room Temperature (oC)
mqtt:topic:acc116ed:Indoor_Temperature
String

Fig. 11. The relation between Things and Items.



```
/habib/arduino/command    1  QoS 0
/habib/arduino/command    2  QoS 0
/habib/arduino/command    3  QoS 0
/habib/arduino/command    4  QoS 0
/habib/arduino/command    5  QoS 0
/habib/arduino/command    6  QoS 0
/habib/arduino/command    7  QoS 0
/habib/arduino/command    104
22-03-2020 20:02:43.72163091    QoS 0
OFF
```

Fig. 12. openHAB home server and ESP32 Thing communication via MQTT broker.

## VII. Experimental Setup

This research mainly focuses on the thermal energy storage system monitoring and control, as mentioned in Fig. 1. The prototype consists of indoor and outdoor temperature sensor data monitoring and thermostat settings control using Sprukfun ESP32 Thing device and openHAB smart home automation open-source IoT platform. Analog (TMP35) and digital (DHT11) temperature sensors have been used to monitor indoor room and ambient temperature. Similarly, a remote control electric heater with a digital temperature display has been considering as a thermostat settings device. openHAB home server is used for remote monitoring and supervisory control. Here one set of sensors and thermostat control systems is used for testing purposes.

### A. *Implementation Methodologies*

The primary implementation methodologies are to design a low cost, low power consumption open-source IoT remote monitoring and control system. The analog and digital temperature sensors are connected to the analog and digital input pins of Sparkfun ESP32 Things, respectively. One temperature is placed on the outside, and another one has been placed in the room. These sensors collected temperature data form room and outside environment and sent it to the ESP32 microcontroller through the serial port. The microcontroller is programmed with Arduino IDE to receive these sensor data, displayed them in a serial monitor, and then sent it to the local Wi-Fi network to the MQTT broker. The MQTT broker sent these data to the openHAB home server via the local WI-FI network to display at the openHAB control panel. On the openHAB cloud console, the user can visualize the sensor data by merely putting the user credentials and password, no need to remember the IP/TCP address. The implementation methodology is shown in Table 2 below, and the appearance of the received data on MQTT broker and openHAB home server, which is described earlier, is shown in Fig. 18.

TABLE 2: The implementation methodologies

```
Initialization:
 1.  Read sensor values on digital pin 34 and 35.
 2.  Display the sensor values on Arduino IDE serial monitor
 3.  Connect to the local Wi-Fi network with Wi-Fi Name and
     password.
 4.  Connect to the MQTT broker with local WI-Fi.
 5.  Display the sensor values on the MQTT broker.
 6.  Connect to the openHAB home server by writing localhost:8080
     at the browser URL.
 7.  Go the BasicUI>Sitemaps>Control and monitoring panels to
     display the sensor values.
 8.  Go to the PaperUI>Control to display the sensor values.
 9.  Change the value of relay and thermostat settings on openHAB
     control panel.
 10. Display the user control response in the MQTT broker.
 11. Monitor the output of ESP32 digital output pins to visualize the
     relay and thermostat display change.
 While openHAB home server and MQTT broker Acknowledge data
 receipt do
 12. Display sensor data on openHAB Cloud, and
 13. Display "Ok" on Arduino IDE serial monitor.
 14. If No data receipt acknowledge form openHAB home server then
 15. Display Debug/Error message on Arduino IDE serial monitor and
     MQTT broker.
 16. else
 17. Go to step 1
   end
   end
```

The user will be confirmed that this algorithm is working perfectly when he/she will see the MQTT broker output, as mentioned in Fig. 12, and the sensors post the data as like Fig. 18.

Prototype design: The hardware components and the operational principles for every element discussed above have been used to design and to implement the low-cost monitoring and control systems, as shown in Fig. 15. The sensors, resistor arrangement, and the ESP32 Thing microcontroller are connected following the correct pin configuration on a breadboard. The ESP32 ADC pins require 3.3 V, but the power supply is 5 V because the ESP32 has been powered with a USB power supply. That's why pull-down resistors are needed. The Wi-Fi router has been turned on; thus, ESP32 is connected to the MQTT broker and openHAB home server installed in the personal computer.

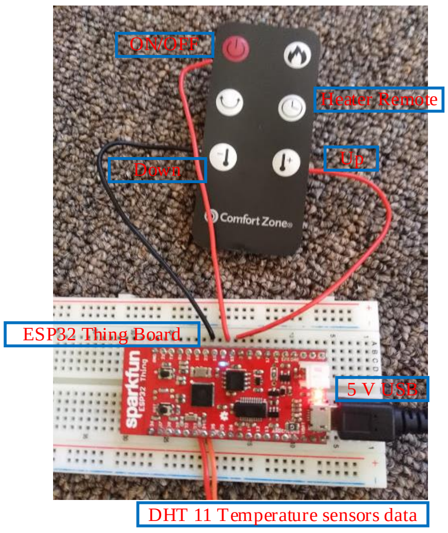


Fig. 14. Hardware connection of the proposed monitoring and control systems.

The proposed monitoring and control system project has been implemented at the author's residential house as shown in Fig. 14 and Fig. 15. Two temperature sensors have been connected, one is in the outside, and another one is in the room inside to measure the ambient and room temperature, respectively. The rest of the components, Wi-Fi- routers, personal computer integrated with openHAB home server, MQTT broker have been set up in the place inside, as shown in Fig. 15. The heater remote and thermostat have been connected with the ESP32 Things. The communication happened bi-directional ways, as shown in Fig. 13. The complete experimental setup is shown in Fig. 15.

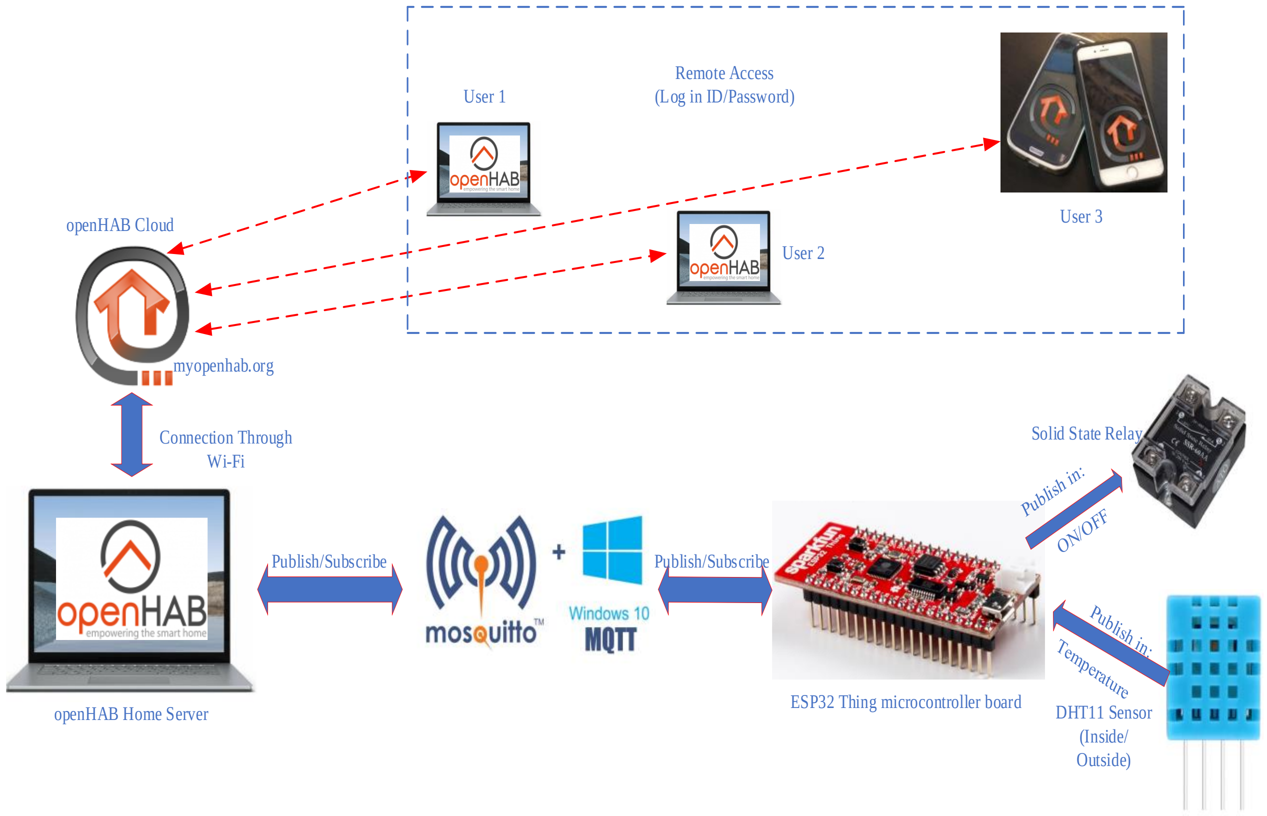


Fig. 13. The communication mechanism.

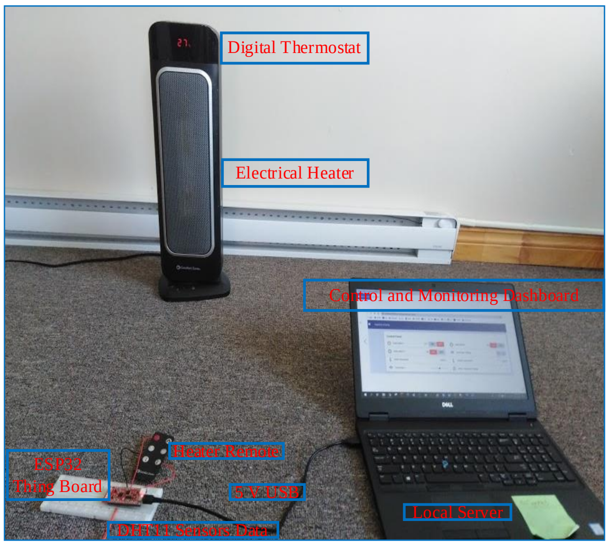
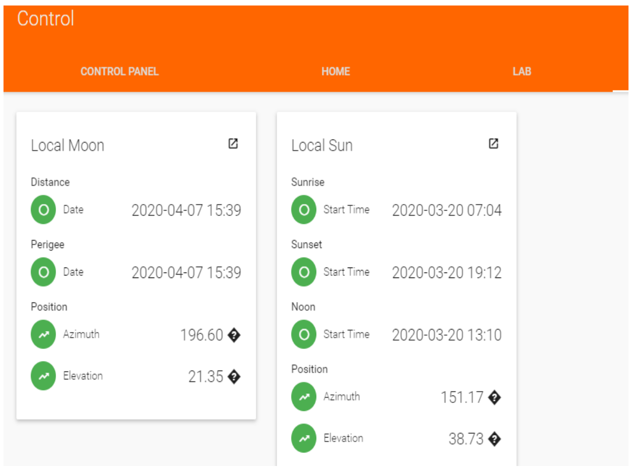


Fig. 15 Experimental Setup of proposed systems.

## B. System Testing

The components have been set up at one of the author's residential homes to implement the proposed monitoring and control system. The overview and flow chart of the data acquisition, processing, visualization, and supervisory control process from the sensors to the openHAB home server IoT platform is shown in Fig. 16.



Control

CONTROL PANEL HOME LAB

Local Moon

Distance
Date 2020-04-07 15:39
Perigee
Date 2020-04-07 15:39
Position
Azimuth 196.60 �
Elevation 21.35 �

Local Sun

Sunrise
Start Time 2020-03-20 07:04
Sunset
Start Time 2020-03-20 19:12
Noon
Start Time 2020-03-20 13:10
Position
Azimuth 151.17 �
Elevation 38.73 �

Fig. 16. The sensor data is in the openHAB server.

The experimental trouble shooting and data verification has been conducted based on the flowchart in Fig. 17.

## C. Experimental Results

The openHAB local server IoT platform was configured in the personal computer following the steps shown in Fig. 9, configured the MQTT broker to the personal computer as well, and the necessary library files (MQTT, DHT11 sensor) uploaded to the Arduino IDE to configure and establish the connection among the openHAB home server, MQTT broker and ESP32 Thing.

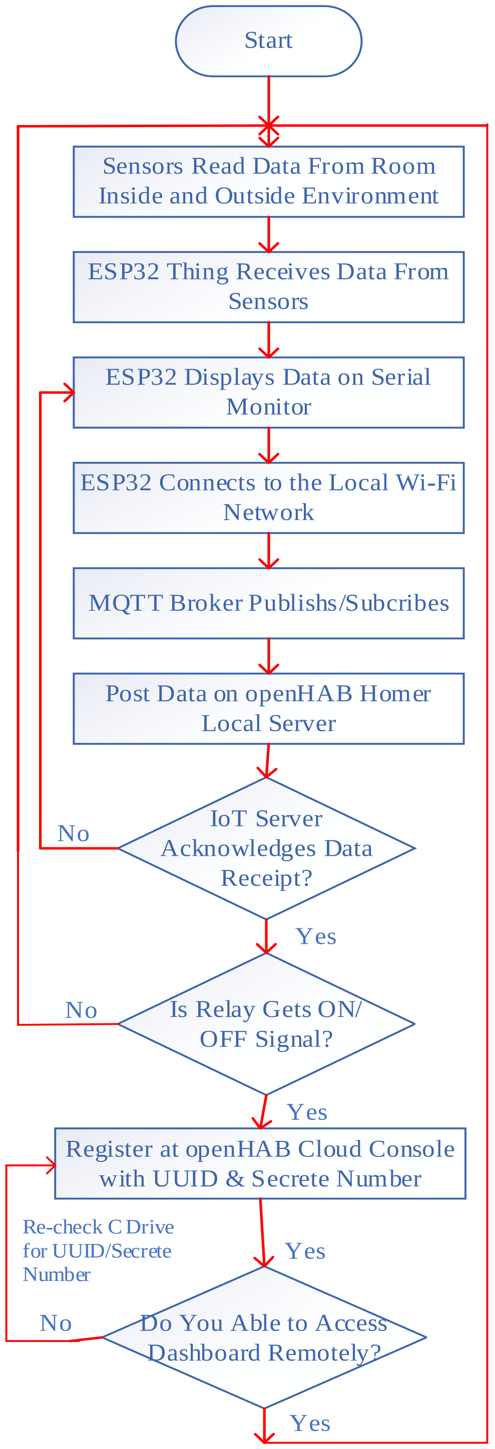
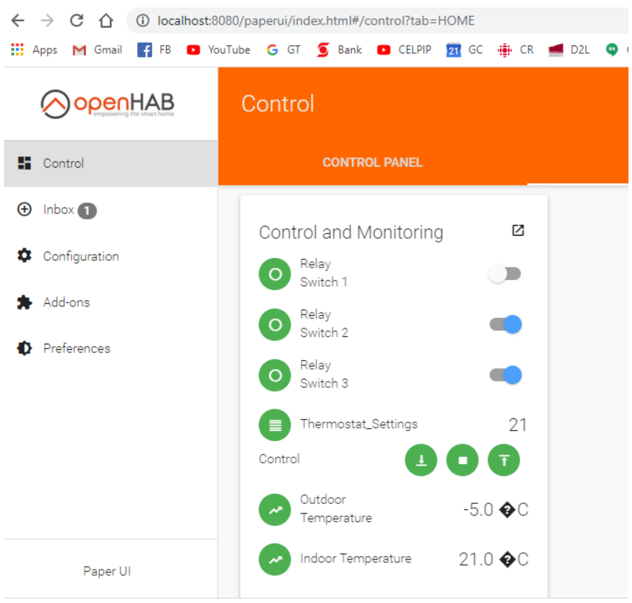

Start
Sensors Read Data From Room Inside and Outside Environment
ESP32 Thing Receives Data From Sensors
ESP32 Displays Data on Serial Monitor
ESP32 Connects to the Local Wi-Fi Network
MQTT Broker Publishes/Subcribes
Post Data on openHAB Homer Local Server
IoT Server Acknowledges Data Receipt?
No
Yes
Is Relay Gets ON/ OFF Signal?
No
Yes
Register at openHAB Cloud Console with UUID & Secrete Number
Re-check C Drive for UUID/Secrete Number
Yes
Do You Able to Access Dashboard Remotely?
No
Yes

Fig. 17. Flow chart of the proposed system.

The sensor's data were posted on the openHAB local monitoring and control system dashboard and Cloud console for remote monitoring and control using a mobile phone and any other computer devices. The openHAB server control panel and main user control panel from Sitemap are shown in Fig. 18 and Fig. 19 respectively.



Control and Monitoring

Relay Switch 1

Relay Switch 2

Relay Switch 3

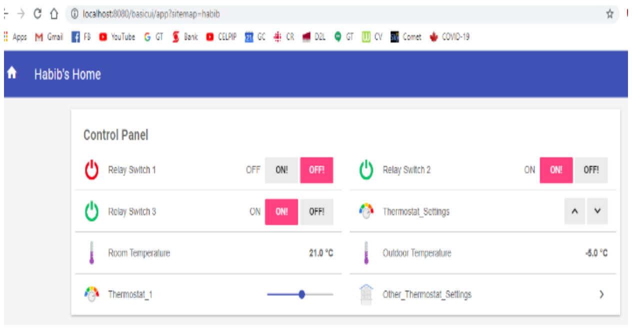
Thermostat_Settings 21

Control

Outdoor Temperature -5.0 �C

Indoor Temperature 21.0 �C

Fig. 18 The openHAB home server control panel.



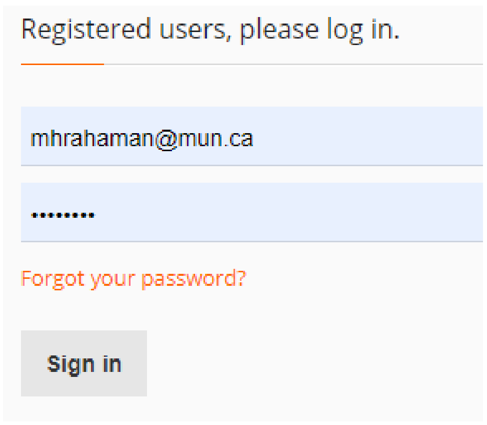
Control Panel

| | | | |
|---|---|---|---|
| Relay Switch 1 | OFF ON! OFF! | Relay Switch 2 | ON ON! OFF! |
| Relay Switch 3 | ON ON! OFF! | Thermostat_Settings | ^ ˅ |
| Room Temperature | 21.0 °C | Outdoor Temperature | -5.0 °C |
| Thermostat_1 | | Other_Thermostat_Settings | › |

Fig. 19. Main control and monitoring dashboards.

## *D. Remote Control and monitoring*

openHAB Cloud is a smart service that allows users to quickly put the user credentials and explore the dashboard using any computer, laptop, tablets or mobile devices from anywhere in the world. During the setup and Configuration, remote access option was allowed. Similarly, in Fig. 9, it is mentioned that how to registrar at openHAB Cloud and where to get the UUID and secrete code for registration. Now, for remote access, the user need to browse the openHAB Cloud console by visiting https://myopenhab.org/login and simple put the user credentials as shown in Fig. 20.



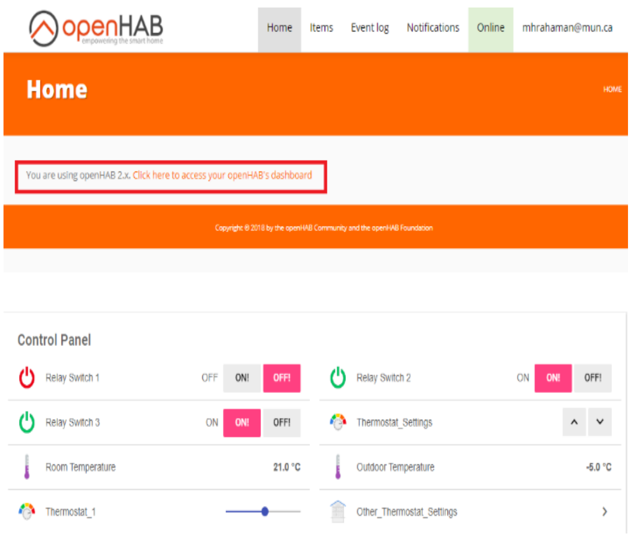
Registered users, please log in.

mhrahaman@mun.ca

••••••••

Fig. 20. User credential for remote access.

User no need to memorize the IP/TCP address or any other credentials. Rather than, users will be simply able to access the monitoring and control dashboard from anywhere in the world, as shown in Fig. 21.



You are using openHAB 2.x. Click here to access your openHAB's dashboard

Copyright © 2018 by the openHAB Community and the openHAB Foundation

Control Panel

Relay Switch 1 OFF ON! OFF!
Relay Switch 2 ON ON! OFF!
Relay Switch 3 ON ON! OFF!
Thermostat_Settings
Room Temperature 21.0 °C
Outdoor Temperature -5.0 °C
Thermostat_1
Other_Thermostat_Settings

Fig. 21. The access of the dashboard from a remote place.

## VIII. Discussion

Some of the highlighted features of the proposed low-cost, open-source smart monitoring and control systems are discussed below:

IoT based system: The proposed system is based on the Internet of Thing (IoT) which have four basic parts such as the heater with a digital thermostat is used as the facilities (plant) to be managed; sensors are the field instrumentation devices used as a data collection and acquisition, the ESP32 Thing microcontroller device is used as a Master Terminal Unit (MTU) as it has data handling, processing, and human-machine interactions capabilities. The wireless Wi-Fi router is created a communication channel between the openHAB home server and MTU.

Low-cost and open-source: All the components discussed above are pretty cheap and readily available everywhere around the world (nearby local store and online). It is the customer's choice to buy the components from their own, which is the key feature of open-source systems. It is assumed that the thermostat and Wi-Fi router are available in every house, so those components cost have been excluded in the cost calculation. The cost of induvial components and over the system is summarized in Table 1. As seen, the overall system costs are just around $50 CAD (Table 3). So, it is proved that the proposed systems are a low-cost system compared to the other available technology in the market.

TABLE 3: Project Cost Calculation

| S/N | Name of the components | QTY | Price (CA$) |
|---|---|---|---|
| 1 | ESP32 Thing | 1 | 31.90 |
| 2 | TMP35 temperature sensors | 2 | 2 |
| 3 | DHT11 Temperature sensor | 2 | 2 |
| 4 | Miscellaneous (Breadboard, Resistors, Wires, Boxes, etc.) | 1 | 10 |
| | **Grad total:** | | **45.9** |

Low power: The components described above in Table 1 consume very negligible power around (W) in total, which is very low. The power consumption of all components have been measured during the operation and summarized in Table 4.

TABLE 4: Project Components Power Calculation

| S/N | Hardware | Power (W) |
|---|---|---|
| 1 | ESP32 Thing | 0.5 |
| 2 | Breadboard (with Sensors, ESP32, Resistors, etc. connected) | 3.3 |
| | **Total:** | **3.8 W** |

Monitoring: The openHAB home server system has a dashboard for data monitoring and thermostat control. These dashboards are accessible to the user in so many ways, such as the server personal computer, other personal computers and remote computer or mobile devices through openHAB mobile App.

Supervisory control: The system allows the user to access the home server via a personal computer to monitor and control. Similarly, the user can log in to the openHAB cloud console by merely putting their user credential and password and can monitor and control the devices anywhere in the world.

## IX. Conclusion

In the most residential house in Canada uses a large amount of electricity for space heating and water heating. To do that, there are huge monthly electricity bills every residence needs to pay, and the grid became overload sometimes. Due to the mismanagement of the electrical heater, radiator, the consumption would be more as well. To ensure the optimum management of heating elements, to monitor the status of room temperature and ambient temperature from locally and remotely, to reduce the system supervisory system cost and power consumption, the proposed system would be highly useful for real application. This system ensured the reliable, flexible, cost-effective, lower power requirement, and sophisticated, coordinated monitoring and control of measurement of temperature and heater/radiator thermostat settings. Although there is some similar system existing in the market which is described in the introduction section, these systems have several drawbacks, such as there is no way to develop the system because the supply industry controls it. There are some limitations in several device integrations, and those systems are costly. Rather than the proposed method is cost-effective, simple, the user can add the Bindings, channels and can add as much as a device he needs to connect for the whole operations. The utility company, such as Newfoundland power, can also use these systems for controlling the thermostat settings in a large area centrally. For example, the utility company will monitor the power generation and demand, as they know that around 60% consumptions are for space heating. Approximately 20% consumptions are for water heating, if they set the thermostat settings as a specific temperature for all house by using the openHAB Cloud console from their office (remotely), then the residence cannot increase the heating temperature beyond this settings, thus grid overloading will be resolved, and mismanagement issue will be resolved.

## Acknowledgment

The authors are grateful to the NSERC, Canada, and Memorial University of Newfoundland for providing sufficient funds and support for this project.

## Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## References

[1] H. Rahaman, R. Rasha, and M. T. Iqbal, “Design and analysis of a solar water heating system for a detached house in newfoundland,” in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, May 2019, pp. 1–4, doi: 10.1109/CCECE43985.2019.8995175.

[2] M. H. Rahaman and T. Iqbal, “A Comparison of Solar Photovoltaic and Solar Thermal Collector for Residential Water Heating and Space Heating System,” *European Journal of Engineering Research and Science*, vol. 4, no. 12, pp. 41–47, Dec. 2019, doi: 10.24018/ejers.2019.4.12.1640.

[3] D. Rohde, B. R. Knudsen, T. Andresen, and N. Nord, “Dynamic optimization of control setpoints for an integrated heating and cooling system with thermal energy storages,” *Energy*, vol. 193, p. 116771, Feb. 2020, doi: 10.1016/j.energy.2019.116771.

[4] F. De Ridder, M. Diehl, G. Mulder, J. Desmedt, and J. Van Bael, “An optimal control algorithm for borehole thermal energy storage systems,” *Energy and Buildings*, vol. 43, no. 10, pp. 2918–2925, Oct. 2011, doi: 10.1016/j.enbuild.2011.07.015.

[5] M. LeBreux, M. Lacroix, and G. Lachiver, “Control of a hybrid solar/electric thermal energy storage system,” *International Journal of Thermal Sciences*, vol. 48, no. 3, pp. 645–654, Mar. 2009, doi: 10.1016/j.ijthermalsci.2008.05.006.

[6] “Keeping eyes on your home: Open-source network monitoring center for mobile devices - IEEE Conference Publication.” https://ieeexplore.ieee.org/document/7296336 (accessed May 31, 2020).

[7] S. Chivarov, P. Kopacek, and N. Chivarov, “Cost Oriented Humanoid Robot communication with IoT devices via MQTT and interaction with a Smart Home HUB connected devices,” *IFAC-PapersOnLine*, vol. 52, no. 25, pp. 104–109, Jan. 2019, doi: 10.1016/j.ifacol.2019.12.455.

[8] T. Sysala, D. Fogl, and P. Neumann, “The family house control system based on Raspberry Pi,” *MATEC Web Conf.*, vol. 125, p. 02034, 2017, doi: 10.1051/matecconf/201712502034.

[9] “Demand Side Management of Smart Homes Using OpenHAB Framework for Interoperability of Devices - IEEE Conference Publication.” https://ieeexplore.ieee.org/document/8506917 (accessed May 31, 2020).

[10] S. Saxena, S. Jain, D. Arora, and P. Sharma, “Implications of MQTT Connectivity Protocol for IoT based Device Automation using Home Assistant and OpenHAB,” in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019, pp. 475–480.

[11] D. Uckelmann, B. Wohlfarth, and M. Guedey, “Smart Public Building,” Dec. 2018.

[12] M. Ramljak, “Security analysis of Open Home Automation Bus system,” in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2017, pp. 1245–1250, doi: 10.23919/MIPRO.2017.7973614.

[13] C. Coté, M. Heidarinejad, and B. Stephens, “Elemental: An Open-Source Wireless Hardware and Software Platform for Building Energy and Indoor Environmental Monitoring and Control,” *Sensors*, vol. 19, p. 4017, Sep. 2019, doi: 10.3390/s19184017.

[14] J. A. S. Velázquez, “Securing openHAB Smart Home through User Authentication and Authorization,” 2018.

[15] “Implementation of Home Automation System Using OpenHAB Framework for Heterogeneous IoT Devices - IEEE Conference Publication.” https://ieeexplore.ieee.org/document/8980370 (accessed May 31, 2020).

[16] H. S. Doshi, M. S. Shah, and U. S. A. Shaikh, “INTERNET of THINGS (IoT): INTEGRATION,” vol. 1, no. 4, p. 9, 2017.

[17] A. Loumpas, G. Panaras, and M. Dasygenis, “Design and implementation of an open-source infrastructure and an intelligent thermostat,” May 2018, pp. 1–4, doi: 10.1109/MOCAST.2018.8376651.

[18] M. H. Yaghmaee and H. Hejazi, “Design and Implementation of an Internet of Things Based Smart Energy Metering,” in *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, Aug. 2018, pp. 191–194, doi: 10.1109/SEGE.2018.8499458.

[19] I. Froiz-Míguez, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, “Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes,” *Sensors (Basel)*, vol. 18, no. 8, Aug. 2018, doi: 10.3390/s18082660.

[20] M. Divyashree and H. G. Rangaraju, “Internet of Things (IoT): A Survey,” in *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, Dec. 2018, pp. 1–6, doi: 10.1109/ICNEWS.2018.8903919.

[21] L. O. Aghenta and M. T. Iqbal, “Low-Cost, Open Source IoT-Based SCADA System Design Using Thinger.IO and ESP32 Thing,” *Electronics*, vol. 8, no. 8, p. 822, Aug. 2019, doi: 10.3390/electronics8080822.

[22] “openHAB.” https://www.openhab.org/ (accessed May 31, 2020).