BRIDGET

HVAC Controller

SYNDICATE MEMBERS

Muhammad Muneeb

Noor ul Huda

Ureed Hassan

M Salim Zaid

SUPERVISOR

Asst. Prof. Dr. Mir Yasir Umair

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology,

in partial fulfillment for the requirements of B.E Degree in Electrical Engineering

July 2020

BRIDGET

HVAC Controller

By

Muhammad Muneeb

Noor ul Huda

Ureed Hassan

M Salim Zaid

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology,

in partial fulfillment for the requirements of B.E Degree in Electrical Engineering

July 2020

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled “BRIDGET (HVAC Controller)*”,* carried out by NC Muhammad Muneeb, NC Muhammad Salim Zaid, NC Ureed Hassan, NC Noor ul Huda under the supervision of Asst. Prof. Dr. Mir Yasir Umair for partial fulfillment of Degree of Bachelor of Electrical (Telecomm.) Engineering in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2019-2020 is correct and approved. Any material or information taken from external sources has been properly referenced to.

Approved by
Asst. Prof Dr. Mir Yasir Umair
Department of EE, MCS

Date: ______________

DECLARATION

It is declared that not any part of this thesis has been written, published or submitted in favor of any other qualification in either this institute or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Muhammad Muneeb
176699

Muhammad Salim Zaid

129297

Ureed Hassan
177604

Noor ul Huda
175468

Signature of Supervisor

We dedicate this effort to our parents and teachers whose tremendous support and cooperation led us to this accomplishment.

ACKNOWLEDGEMENTS

We would like to pay gratitude towards Almighty Allah to guide us throughout this work at every step, opening new doorways of success for us, enlightening us with the knowledge and showering upon us His blessings which enabled us to achieve this success.

We are also extremely thankful to our supervisor Asst. Prof. Dr. Mir Yasir Umair for his help during the project, his constant guidance and encouragement. The project would not have been possible without his expert advice, unfailing patience, and invaluable efforts.
We would also like to mention Mr. Shoaib Ahmed for his incredible provision and assistance. We appreciate his guidance throughout the whole project.

Finally, we would also thank the department of EE and our project coordinators who walked us through all the administration related affairs.

ABSTRACT

Operating individual HVAC (Heating Ventilation and Air Conditioning) systems in an environment where multiple HVAC systems are deployed is a time consuming and tedious task. Each system has to be manually operated via an IR remote controller or over the internet, depending on the make and model of the HVAC system/s in question. Turning on each individual system and then adjusting them to the desirable temperature also requires time and manual labor. Even a person dedicated to performing and adjusting these conventional HVAC systems takes several minutes to do so and that too depends on the number of units the person has to operate.

Clearly a need to cater to this unnecessarily tedious task is required. The required solution however needs to be universal for all makes and models of the many HVAC systems currently in use in the market. In addition to this universal utility, the solution also needs to be cheap, efficient and easily deployable. This is where **'Bridget'** comes in. It provides an efficient, timesaving and relatively cheaper way to control several HVACs at the same time.

Key Words:

HVAC: Heating Ventilation and Air Conditioning

BMS: Building Management System

IoT: Internet of Things

IR: Infra-Red

API: Application Program Interface

IDE: Integrated Development Environment

GUI: Graphical User Interface

REST: Representational State Transfer

API: Application Program Interface

CLI: Command Line Interface

CRUD: Create, Read, Update, Delete

Table of Contents

Index of Figures

Index of Tables

CHAPTER 1

INTRODUCTION

1 INTRODUCTION

Engineering is the not just the science of proposing, generating, and implementing. It, indeed, is the art of projecting the ideas in one’s mind onto the paper, designing life size objects from it and transforming the little spark of neurons created in mind from imagination, to the tremendous innovations that leave the mankind in awe by the gusto of it. Hence the responsibilities commonly attributed to engineers may be interdisciplinary but are always in lieu of increased productivity and higher output. One of the ideas to emerge from engineering minds is automation. The application of machines to perform tasks once performed by human beings or more importantly, increase the productivity of the manually operated tasks is termed as automation. Where mechanization may be considered synonymous to automation, the former refers to the mere replacement of human labor with machines. Automation, however, has a much more diverse outreach where it integrates the machine into a self-monitoring system revolutionizing the areas where it has been implemented. There has barely been any domain of modern life that has been uninfluenced by the marvel that is automation.

1.1 Overview

Heating, ventilation, and air conditioning (HVAC) is a technology of indoor comfort. HVAC systems are employed in both commercial and residential areas with equal frequency in order to maintain the said level of indoor temperature and comfort. That said, the HVAC systems also need to be within a reasonable threshold of maintenance complexity. The maintenance of these HVAC systems includes the running costs, methods by which they operate and their installation procedures.

Ever since the advent of air conditioners themselves, there has been constant development on the domain of HVAC systems, dampening down the maintenance complexity of these systems hence consequently increasing user friendliness. This has been achieved using several different tools and mechanisms provided by the engineering circles. This has rendered the HVAC field of work to be an interdisciplinary and complex domain to work in. The new improvements in the field include the use of advanced sensing components, state of the art control systems and where required, even artificial intelligence. As a result, the longevity of HVAC systems has enabled them to be installed and deployed in developing countries at a rapid rate.

As people’s need for thermal comfort and energy efficiency increases, the development and implementation of effective control techniques for HVAC systems plays a major role in building energy management. In the past many years, people are coming more and more towards automation and IOT, hence the integration on Building Management Systems (BMS) with HVACs is quite common now. Although some traditionally used systems are still effective in local HVAC system units or single small building sections. Although on-off control is the easiest to implement, its ability to control moving processes with nonlinear and time-delay dynamics is limited.

In this world of constant technical evolutions, almost every aspect of our daily lives has disembarked on a journey of automation. User friendliness and effectiveness are given top priority. Bridget also places these parameters in top priority by infusing hardware and software controlling techniques of HVAC systems into an effective hybrid control system that enables automation of a variety of HVAC systems regardless of their unique makes and models.

1.2 Problem Statement

In terms of automation, there is a missing link between HVAC units and their controllers. The multiple units installed in multiple stories of a building cannot be controlled by a single operator at the same time. This missing piece of puzzle is being provided in terms of BRIDGET, which will not only provide the interface to get commands from user but also translate those commands to their respective units.

1.3 Working Principle

Device is mainly based on IOT and is divided into different modules. Each module is linked to the other one forming a whole mesh. Every module takes commands or information processes it and transmit it to the next relevant module. The details of the modules are given below:

- GUI
- Gateway
- Nodes
- BMS

1.3.1 GUI

The user interface is provided by the mobile application. The app is based on flutter and it uses Firebase as its cloud database. User registers itself on the database with its email address. The cloud then sends data/commands to the gateway.

1.3.2 Gateway

The Gateway is the main controller and bridge of the whole project. It gets the commands from users via cloud then it relays those commands to the node for which they were meant for. ESP-32 serves the purpose for the gateway.

1.3.3 Nodes:

All the HVAC units in the mesh network are termed as nodes. The mesh is formed by several nodes. Each node is distinguished from others based on a unique IP address assigned to them. The commands from the gateway are sent to the corresponding node based on that IP address. The mesh is created using ESP8266 which connects the units to the internet.

1.3.4 BMS:

BMS stands for Building Management System. Bridget provides compatibility with the existing BMS via RS 485 protocol through Serial communication.

1.4 Objective

The project serves to several objectives which can be broadly classified into two categories:

1.4.1 Academic Objectives:

Academic objectives envelopes a wide range of outcomes that are expected to be completed through this project and can be explained in a nutshell as:

- Applying the theoretical knowledge into practicality pertaining to the core features of engineering and technology.
- Having a complete grasp on the concepts relating to the problem-solving aspects of outcome-based learning

The end goals of the project can be broadly identified as:

- Design and develop IR module for each node using ESP
- A gateway or controller which will translate the commands to the nodes
- Development of android application
- Compact hardware designing of the device
- Compatibility with the existing BMS

1.4.2 Industrial Objectives:

To present Bridget as a novel automation solution, promising ease of functionality and hassle-free working, enabling the companies and enterprises to have a rather cost-efficient solution as their first choice.

1.5 Scope

The project finds its scope in almost every building equipped with HVAC units. It is an efficient, time saving solution for a tedious manual task of handling all the units individually.

1.6 Deliverables

1.6.1 Stand Alone Integration

Bridget can run independently of the make and model of individual HVAC units. The operation of the units is dependent upon the device configuration. More importantly, Bridget does not require any previous BMS or HVAC controller to be integrated upon however, it can be integrated over existing BMS.

1.6.2 Control Over Mobile application

The control of the device which is then extended to the concerned units is administered over a highly interactive and user-friendly mobile application. Its GUI is based upon a conventional AC remote for ease of use.

1.6.3 Third Party Control

In addition to the above-mentioned features, Bridget can also be controlled via Building Management Systems like Google Home, Alexa etc. giving it the compatibility with a third party.

CHAPTER 2

LITERATURE REVIEW

2 LITERATURE REVIEW

Today, in dynamic world of technology, we have always been trying to add ease, comfort, and modernism to our lifestyle. The solution to this, in terms of technology, is automation. Automation is helping to bring our lives into comfort and cohesion, bringing each and everything to a click of our finger. On the other hand, when we talk about convenience and comfort, then the commodity which has become an integral part of our daily lives, is the HVAC units, or simply air conditioners.

2.1 Industrial and Domestic Background

Pakistan is one of the possible markets for climate control systems in the Asia-Pacific locale; driven by rising speculations, open and private foundation advancement which would drive government spending in a few areas, for example, lodging, framework, business, cordiality, assembling, instruction, and medicinal services. To help the vision, a MoU was marked "China-Pakistan Economic Corridor (CPEC)" among China and Pakistan, in which China would contribute more than $100 billion by 2017.

Pakistan forced air system advertisement size is anticipated to develop at a CAGR of 7.2% during 2017-23. Regardless of political questions and security concerns, Pakistan climate control system showcase is developing at a sound rate inferable from expanding discretionary cash flow and extending white collar class populace. Pay of family units in Pakistan have in the long run expanded attributable to developing working populace bringing about high buying power, which is further adding to the development of the climate control system showcase in the nation.

The nation's economy is foreseen to develop over the coming years inferable from ascend in government spending on framework advancement, Foreign Direct Investment and development in residential speculation. Throughout the following six years, bringing together climate control systems, especially VRF fragments is probably going to show a considerable Pakistan forced air system piece of the pie because of expanding development exercises in the business segment.

Split climate control system portion has overwhelmed the general forced air system showcase because of appeal emerging from the private part. Punjab and Sindh areas are the key income contributing districts are probably going to rule Pakistan forced air system advertise estimated incomes over the coming years.

Remembering this angle, the HVAC business is presently clearing the street towards robotization, incorporating a portion of its very center components with innovations like IoT and expelling bother from the lives of their purchasers.

2.2 Existing Solutions and their drawbacks

There are some companies who are already into this market, including *Sensibo* and *Coolmate*.

2.2.1 Sensibo:

Sensibo provides its users, a smart air conditioner controller, working on IR. Each unit has a node underneath it, which in turn is attached to the main device through a wire. It also provides user control through an app. On the downside of it, they are not providing a complete wireless IoT based solution.

2.2.2 Coolmate:

On the other hand, *Coolmate* is generally meant for bigger air conditioner units. It integrates with the internal mechanism of air conditioner and the control is offered through the Building management system on a wired connection.

2.3 Research Papers:

In addition to getting insights of our competitors in market, we also pulled out the detailed data of the technologies we are using. Currently as the word is transformed into a global village, and the information of whole world can be fitted onto our fingertips, everyone is looking forward to novelty and compactness.

Today, Internet of Things, Automation, machine learning is the talk of the town. Hence to keep in pace with the everchanging modern world, we are using the most advance technologies in our project, be it hardware or the software.

For this purpose, a thorough analysis about the HVAC systems and their automation was obtained through **"Internet-based monitoring and controls for HVAC applications"** by P. I-Hai Lin and H. L. Bromberg. Also, Jignesh Bhatt, H. K. Verma, in their paper **"Design and Development of Wired Building Automation Systems"**, have mentioned the details of networking and automation of devices in a small building.

CHAPTER 3

OVERALL DESCRIPTION

3 OVERALL DESCRIPTION

3.1 Project Hardware

The project consists of following hardware components:

- ESP 12E to design IR module and transmitting commands to nodes
- ESP32 for the main gateway device which will translate commands to respective units
- RS 485 modules for the wired communication from gateway to existing BMS
- 2N2222 transistor to amplify IR transmitter signals
- IR transmitter
- IR Receiver

3.1.1 ESP 12E:

3.1.1.1 Description

The ESP 12E Wi-Fi Module is a enclosed SOC with the integration of TCP/IP protocol, which enables microcontrollers to access the Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

3.1.1.2 Features

- It is easy to develop projects with AT commands
- Integrated TCP/IP protocol Stack
- Built-in antenna
- Wi-Fi @ 2.4GHz support for WPA/WPA2 security

3.1.1.3 Application

This chip is used in our project to create a mesh of all the nodes, connect them to Wi-Fi and enable them to operate the units.

3.1.1.4 Specifications

Sr. No.	Property Name	Property Value
1	Interface Voltage	3.3V
2	Working Current	240mA
3	Baud rate	115200 (default)
4	Wireless Criteria	802.11 b/g/n
5	Standby power consumption	<1.0mW
6	Dimensions	25 x 17 x 4 mm

Table 1: Specifications of ESP-12e

3.1.1.5 Figure

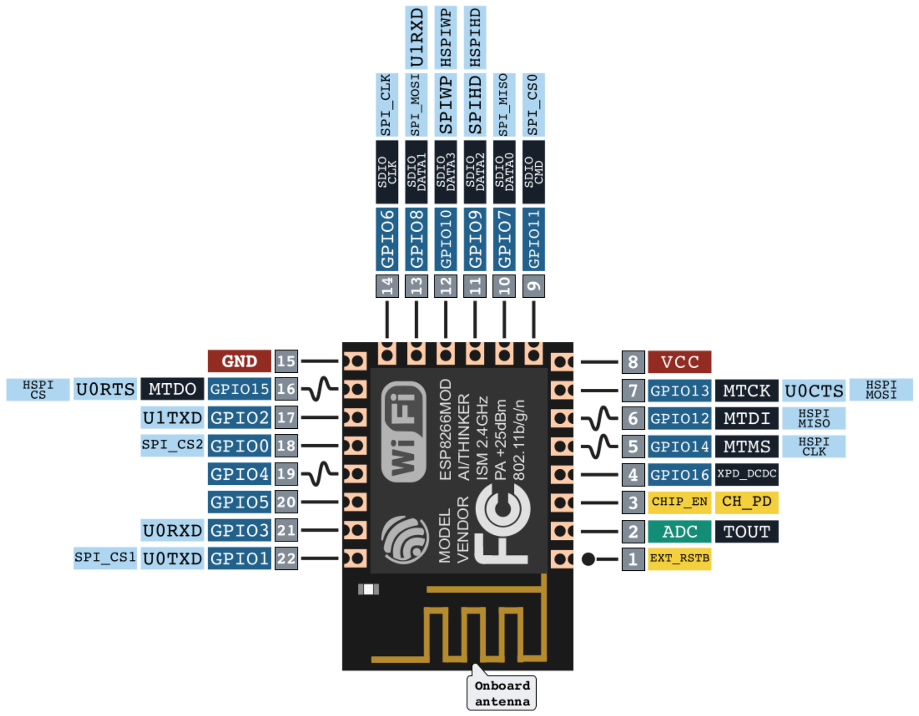


Figure 1: ESP12e Chip

3.1.2 ESP 32:

3.1.2.1 Description

It is a prototyping board that incorporates a microprocessor and IOT. It boasts integrated Wi-Fi and Bluetooth along with an array of built in antenna switches and ultra-low power consumption. It helps in bridging an Arduino with WIFI network.

3.1.2.2 Features

- It operates on the frequency of 2.4 GHz with a data rate of 54 Mbps.
- It is a hybrid Wi-Fi and Bluetooth with an on-board antenna.

3.1.2.3 Specifications

Sr. No.	Property Name	Property Value
1	Voltage supply	2.2 V – 3.6 V
2	Current transmitting and receiving	80mA
3	Data rate	54 Mbps
4	Wireless Criteria	802.11 b/g/n
5	Frequency	2.4 GHz
6	Flash	4 MB

Table 2: Specifications for ESP-32

3.1.2.4 Application

The applications for ESP32 are as follows:

- Universal low power IoT sensor hub
- Home automation
- Mesh Network
- Industrial wireless control

3.1.2.5 Figure

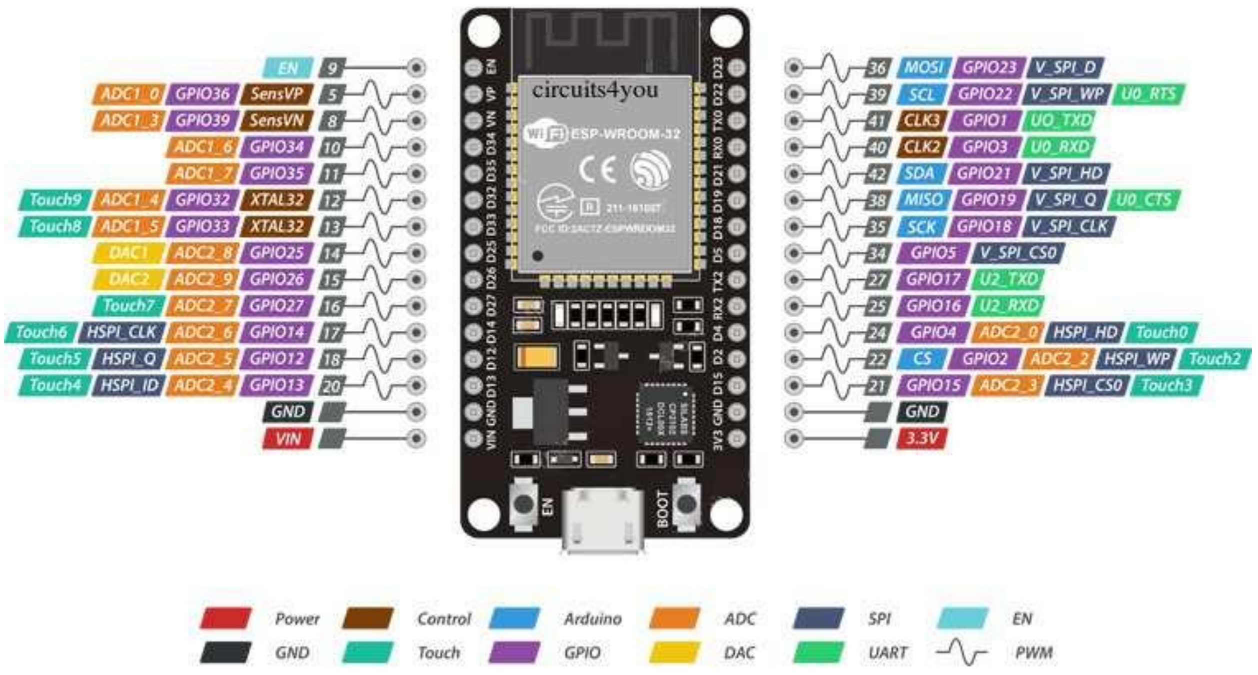

circuits4you
ESP-WROOM-32
EN
BOOT
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
EN 9
ADC1_0 GPIO36 SensVP 5
ADC1_3 GPIO39 SensVN 8
ADC1_6 GPIO34 10
ADC1_7 GPIO35 11
Touch9 ADC1_4 GPIO32 XTAL32 12
Touch8 ADC1_5 GPIO33 XTAL32 13
DAC1 ADC2_8 GPIO25 14
DAC2 ADC2_9 GPIO26 15
Touch7 ADC2_7 GPIO27 16
Touch6 HSPI_CLK ADC2_6 GPIO14 17
Touch5 HSPI_Q ADC2_5 GPIO12 18
Touch4 HSPI_ID ADC2_4 GPIO13 20
GND
VIN
36 MOSI GPIO23 V_SPI_D
39 SCL GPIO22 V_SPI_WP U0_RTS
41 CLK3 GPIO1 U0_TXD
40 CLK2 GPIO3 U0_RXD
42 SDA GPIO21 V_SPI_HD
38 MISO GPIO19 V_SPI_Q U0_CTS
35 SCK GPIO18 V_SPI_CLK
34 GPIO5 V_SPI_CS0
27 GPIO17 U2_TXD
25 GPIO16 U2_RXD
24 GPIO4 ADC2_0 HSPI_HD Touch0
22 CS GPIO2 ADC2_2 HSPI_WP Touch2
21 GPIO15 ADC2_3 HSPI_CS0 Touch3
GND
3.3V
Power
GND
Control
Touch
Arduino
GPIO
ADC
DAC
SPI
UART
EN
PWM

Figure 2 :ESP 32

3.1.3 MAX 485:

3.1.3.1 Description

This low power transceiver module is based on the Maxim MAX485 IC to allow serial communication over very long cable runs (up 4000 feet / 1200 meters). Serial data can be transmitted in both directions (half duplex) at a data rate up to 2.5Mbps. The modules operate from a standard 5V power supply and use 5V logic levels allowing them to be interfaced to the hardware serial ports of microcontrollers such as Arduino/PIC etc.

3.1.3.2 Features

- Direction of communication can be controlled with just one digital pin.
- Each module contains one driver and one receiver.

3.1.3.3 Specifications

Sr. No.	**Property Name**	**Property Value**
1	Max. Distance @ Rate	1200 meter/4000 feet @max. 100 kbps
2	Max. Rate @ Distance	10 Mbps @ 12 meters/50 feet
3	Driver Output Resistance	100 ohms
4	Receiver Input Resistance	4 kilo ohms min.
5	Max. Output Current	150 mA

Table 3: Specifications of RS-485

3.1.3.4 Pin Layout

Sr. No.	**Pin Name**	**Pin Value**
1	VCC	5V
2	A	Non-inverting Receiver Input & Non-inverting Driver Output
3	B	Inverting Receiver Input and Inverting Driver Output
4	GND	0V
5	R0	Receiver Output (to Rx pin of microcontroller)
6	RE	Receiver Output Enable (Low to enable)
7	DE	Driver Output Enable (High to enable)
8	D1	Driver Input (to Tx pin of microcontroller)

Table 4: Pin layout for MAX-485

3.1.3.5 Application

It is possible to connect multiple modules together for communication between 2 or more devices. For even greater distances two modules can be configured as a repeater.

3.1.3.6 Figure

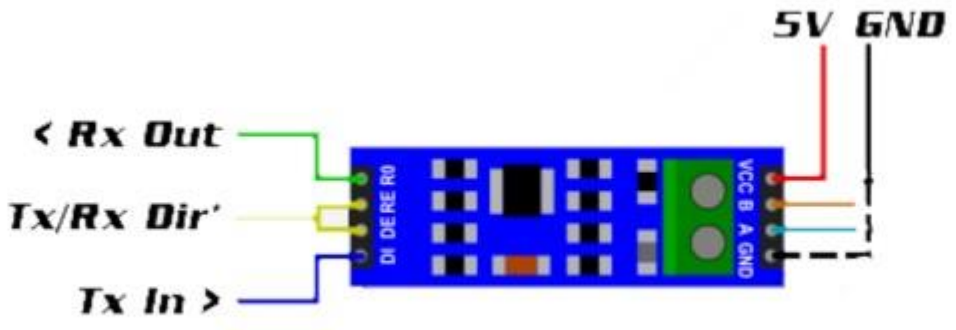


Figure 3: Max 485

3.1.4 2N2222:

3.1.4.1 Description

It is a general-purpose low-power amplifying/switching small-signal transistor. It is bi-polar NPN transistor designed for low to medium current, low power, medium voltage, and can operate at moderately high speeds.

3.1.4.2 Features

- Its large 600mA collector current makes it ideal to use in electronic control relays and high-power LEDs.
- It can be used as a small audio amplifier.

3.1.4.3 Specifications

Sr. No.	**Property Name**	**Property Value**
1	Transistor Polarity	NPN
2	Collector-Emitter Voltage	30V
3	Power Dissipation	500mW
4	DC Collector Current	800mA
5	DC Current Gain hFE	100
6	No. of Pins	3
7	Gain Bandwidth	250 MHz
8	Full Power Rating Temperature	25 degree C

Table 5: Specifications for 2N2222 transistor

3.1.4.4 Applications

- Audio Preamplifiers
- Switching many loads at the same time
- RF Circuits
- Sensor Circuits

3.1.4.5 Figure

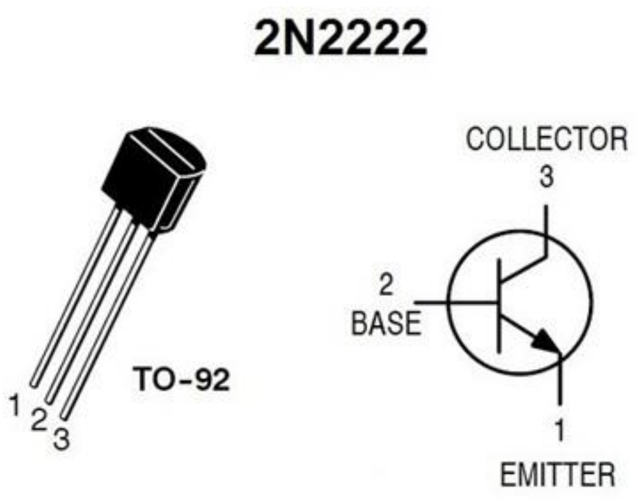


Figure 4: 2N2222 Transistor

3.1.5 IR Tx module:

3.1.5.1 Description

The infrared light emitting diode allows light to emit with the wavelength of up to 940nm, which is the infrared range of the electromagnetic radiation spectrum. The wavelength range varies from 760nm to 1mm. These are mostly used in the remote control of TVs, cameras and different types of electronic instruments. The diagrammatic description of how IR LED works is shown below:

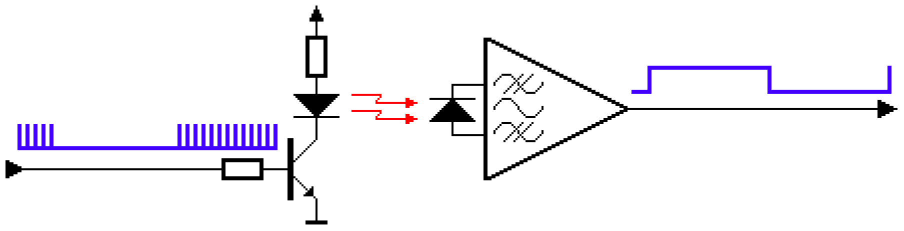


Figure 5: IR Tx LED Description

3.1.5.2 Features

- 1.5 A of surge forward current
- Temperature for storage and operation varies from -40 to 100 degree C
- Soldering temperature should be below 260 degree C
- Power dissipation of 150mW at 25 degree C

3.1.5.3 Specifications

Sr. No.	**Property Name**	**Property Value**
1	Diameter	5mm
2	Wavelength	940nm
3	Spectral Bandwidth	45nm
4	Viewing angle	30 to 40 degree

Table 6: Specifications for IR Tx LED

3.1.5.4 Applications

- Remote control (TV, AC etc.)
- IR Temperature Gun

3.1.5.5 Figure

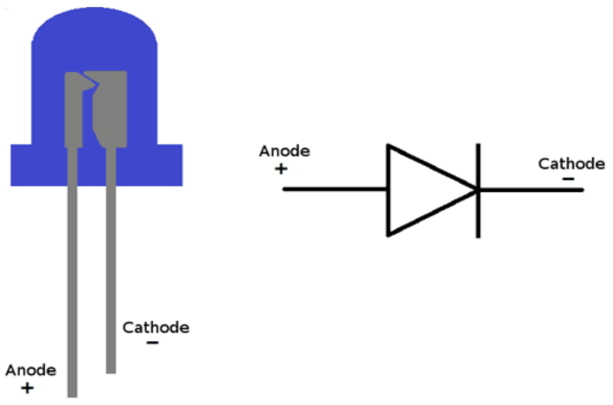


Figure 6: IR Transmitter LED

3.1.6 IR Rx module:

3.1.6.1 Description

Infrared receiver for Arduino adopts 1838 infrared receiving head. Its light resistance, strong electromagnetic interference, and built-in infrared dedicated IC can work under 500 lux light intensity.

3.1.6.2 Features

- Its dimensions are 6.4 x 7.4 x 5.1 mm

3.1.6.3 Specifications

Sr. No.	Property Name	Property Value
1	Receiving Angle	Almost 90 degree
2	Working Voltage	2.7-5.5 V
3	Frequency	37.9KHz
4	Receiving Range	18m

Table 7: Specifications for IR Rx module

3.1.6.4 Applications

- Stereos
- TV
- Set-top Box
- Satellite Receivers
- Air Conditioners

3.1.6.5 Figure

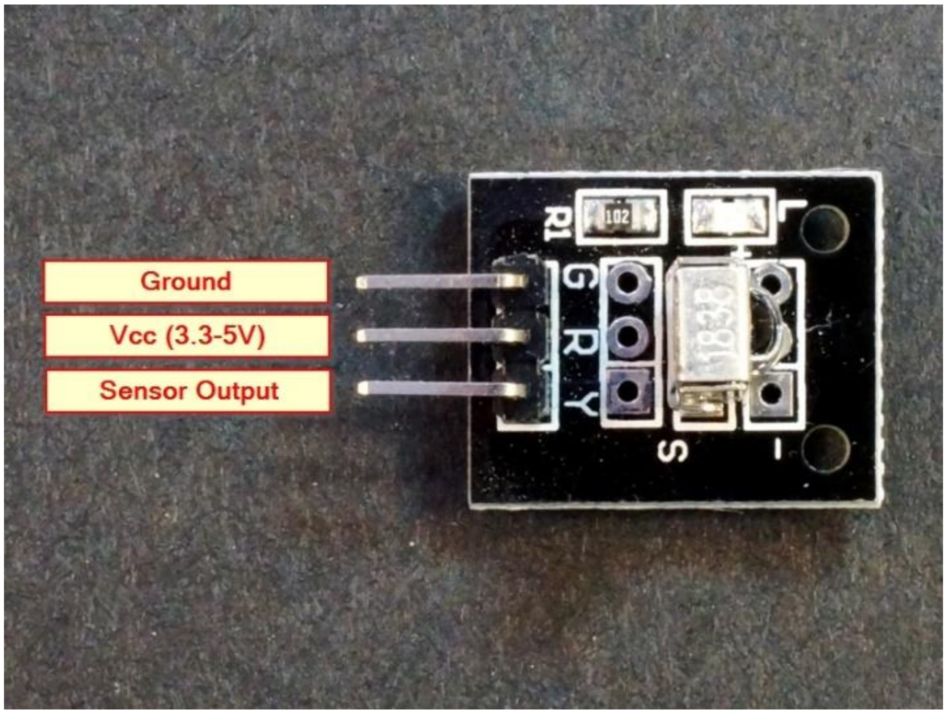


Figure 7: IR Receiver

3.1.7 Power Supply:

3.1.7.1 For Gateway

For gateway power supply, we are using 12V adapter. And a circuit is designed to convert the 12V into 3.3V to supply it to ESP-32.

3.1.7.2 For Node

For a node module, we are using 3.3V lithium-ion rechargeable battery as a power source. It is small enough to be put inside node module case.

3.2 Project Software

The project requires the following software related tasks:

3.2.1 Arduino IDE:

3.2.1.1 Description

The open-source Arduino IDE makes it easy to write and upload codes to the board. Its environment is written in Java and based on Processing and other open-source software.

3.2.1.2 Features

Arduino runs machine code compiled from either C or C++ or any other language that has a compiler for the Arduino instruction set.

3.2.1.3 Application

Arduino IDE, in this project, is used for:

- Reverse Engineering of HVAC commands
- Gateway module
- Nodes

3.2.1.4 Version

Arduino version 1.8.12 is used in this project.

3.2.2 Android Studio IDE:

3.2.2.1 Description

It is an official integrated development environment (IDE) for Android app development. It is based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance productivity when building Android apps.

3.2.2.2 Features

Its features include:

- Flexible Gradle-based build system
- Fast feature-rich emulator
- Unified environment for all Android devices
- Extensive testing tools and frameworks.
- C++ and NDK support
- Built-in Google Cloud Platform

3.2.2.3 Application

Bridget application was created using Android IDE.

3.2.2.4 Version

The version used for this project on 64-bit windows is 3.6.3.

3.2.3 Proteus:

3.2.3.1 Description

It is a proprietary software tool used primarily for electronic design automation. It is mainly used by electronic design engineers and technicians to create schematics and electronic print for manufacturing printed circuit boards.

3.2.3.2 Features

Its features include:

- Schematic Capture
- Microcontroller Simulation
- PCB Design

3.2.3.3 Application

In this project, it is used to design and simulate the circuit and Schematics for Gateway and node modules.

3.2.3.4 Version

Proteus 8 professional was used for this project.

3.3 Cloud Service

3.3.1 Firebase Cloud:

3.3.1.1 Description

Firebase is a cloud platform provided by Google to build, improve, and grow mobile and web applications. It provides reliable and battery efficient connection between server and devices at no cost. It also provides a real-time database and back-end as a service.

3.3.1.2 Features

Its features include:

- Authentication
- Hosting
- Cloud Firestore
- Cloud Functions

3.3.1.3 Application

It acts as a data-base and back-end for Bridget mobile application and helps us synchronize the user data in real time.

CHAPTER 4

PROJECT DESIGN AND DEVELOPMET

4 PROJECT DESIGN AND DEVELOPMENT

4.1 IR Transmitter Design

4.1.1 Overview:

IR transmitter 940nm LED is available in the market. But the LED alone does not have enough strength to transit signal at long distance with more power. For this purpose, a circuit was designed and integrated with the IR transceiver and node later.

4.1.2 Circuit design with IR LED:

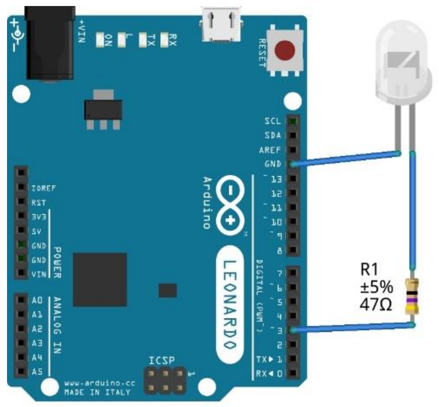
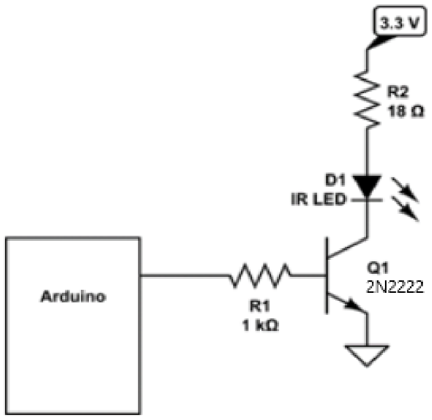


Figure 8: IR transmitter design

4.1.3 LED Power Enhancement:

As the IR LED alone could not deliver desired results, we adopted another approach. We devised a method to strengthen and boost IR LED response using 2N2222 transistor. The circuit design for it is as follows:



4.2 IR Receiver Design

4.2.1 Overview:

At first, we test IR receiver using Arduino UNO. Here, we send IR commands using MP3 player remote. And manually decode the six-digit code from it. But, that's not enough. We will do the IR commands receiving test with ESP-8266 board and KY-022 IR sensor in 4.3.4.

4.2.2 Circuit Design:

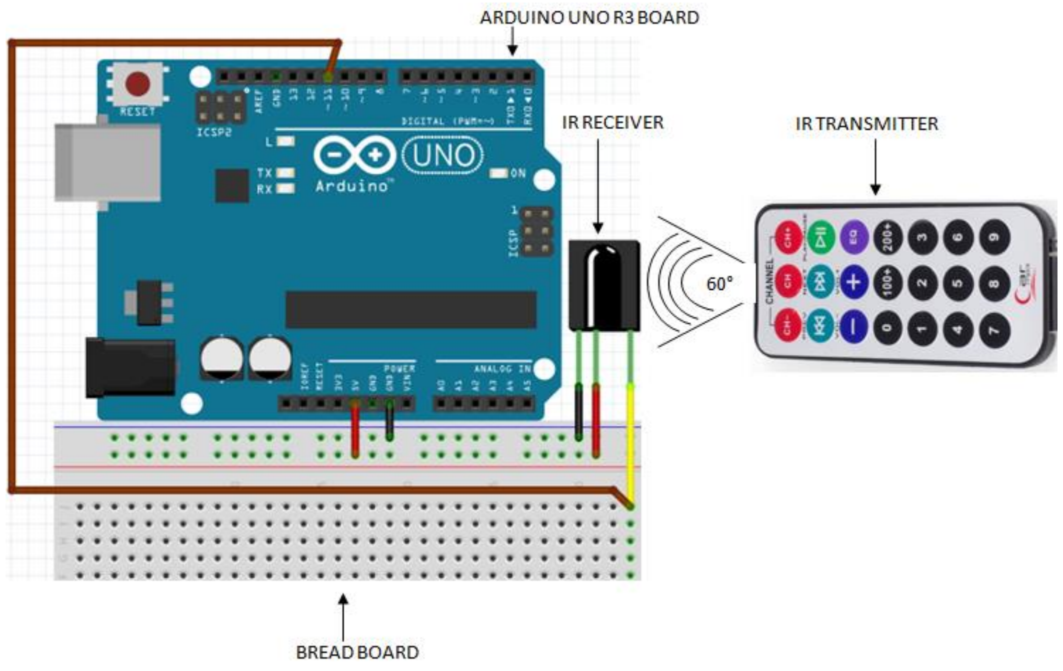


Figure 9: IR receiver design

4.3 Design and development of IR transceiver

4.3.1 Overview:

IR transceiver is designed with esp8266 to capture and send commands form HVAC remote control for reverse engineering of HVAC commands. It is fully functional and can control the AC units when the commands are manually given to it. In this, Commands are received by IR receiver module and then interpreted and sent manually to HVAC unit using micro-controller.

4.3.2 Reverse Engineering of HVAC Commands:

We recorded the bit combination of each mode and operation using "<IRrecvDumpV2.h>" library of Esp8266, analyzed it bit by bit, understood their modulation and configured our device for those commands. Arduino library for those commands are then developed and the already available libraries are modified according to need. These libraries contain the individual function for every command of HVAC remote controller. These commands are then sent using the IR transmitter to test the automation of HVAC units.

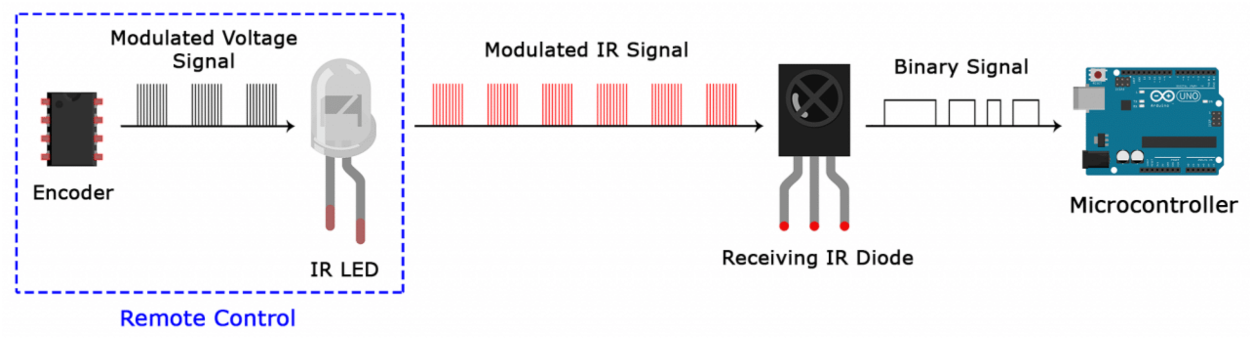


Figure 10: IR signal modulation

4.3.3 Circuit Design:

The circuit design for IR transceiver is shown below:

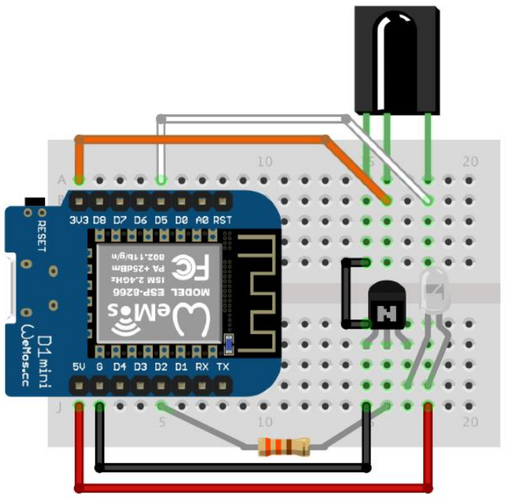
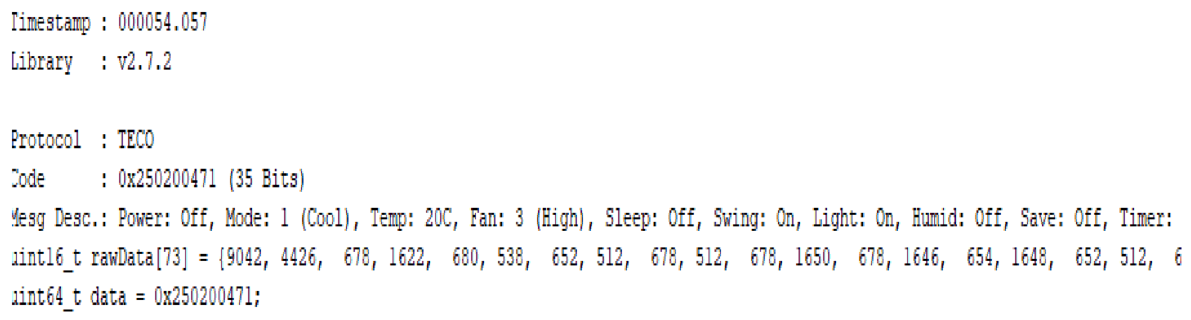


Figure 11: IR transceiver circuit

4.3.4 Testing of IR transceiver:

A pictorial analysis of the testing programs for those libraries is shown below:

- **Receiving commands from HVAC remote**

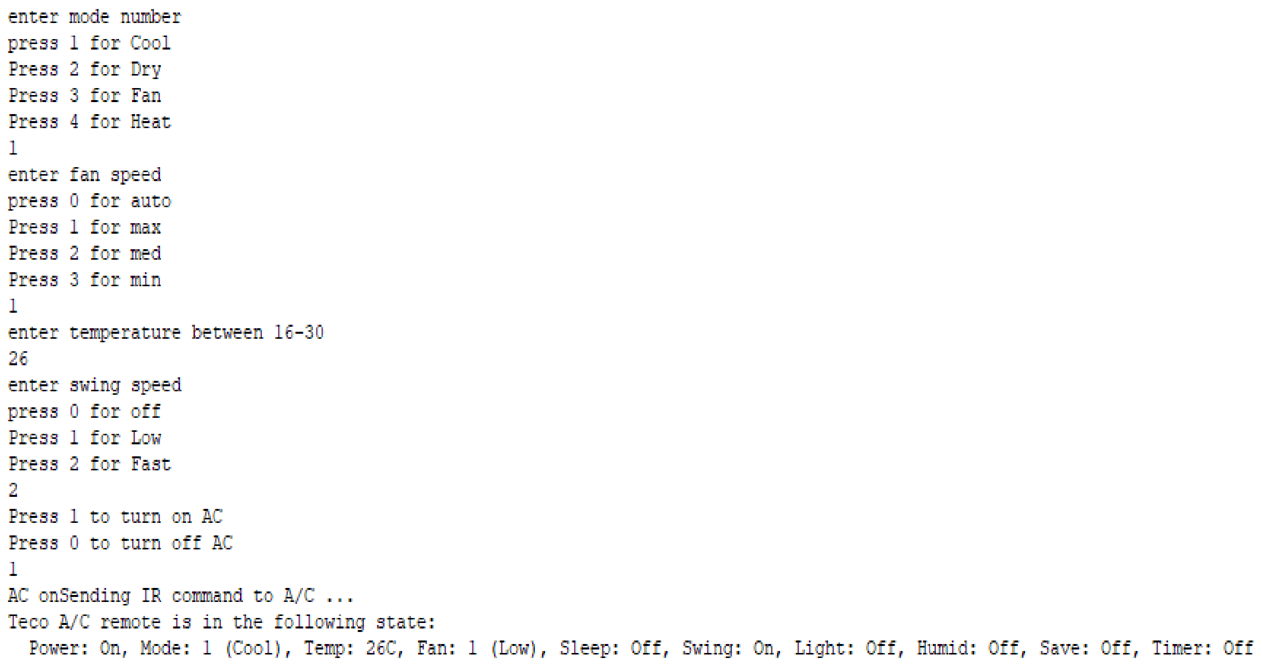


```
Timestamp : 000054.057
Library   : v2.7.2

Protocol  : TECO
Code      : 0x250200471 (35 Bits)
Mesg Desc.: Power: Off, Mode: 1 (Cool), Temp: 20C, Fan: 3 (High), Sleep: Off, Swing: On, Light: On, Humid: Off, Save: Off, Timer:
uint16_t rawData[73] = {9042, 4426,  678, 1622,  680, 538,  652, 512,  678, 512,  678, 1650,  678, 1646,  654, 1648,  652, 512,  6
uint64_t data = 0x250200471;
```

Figure 12: HVAC remote commands

- **Sending manual commands through micro-controller**



```
enter mode number
press 1 for Cool
Press 2 for Dry
Press 3 for Fan
Press 4 for Heat
1
enter fan speed
press 0 for auto
Press 1 for max
Press 2 for med
Press 3 for min
1
enter temperature between 16-30
26
enter swing speed
press 0 for off
Press 1 for Low
Press 2 for Fast
2
Press 1 to turn on AC
Press 0 to turn off AC
1
AC onSending IR command to A/C ...
Teco A/C remote is in the following state:
  Power: On, Mode: 1 (Cool), Temp: 26C, Fan: 1 (Low), Sleep: Off, Swing: On, Light: Off, Humid: Off, Save: Off, Timer: Off
```

Figure 13: Transmission of HVAC commands

4.3.5 Transceiver Module:

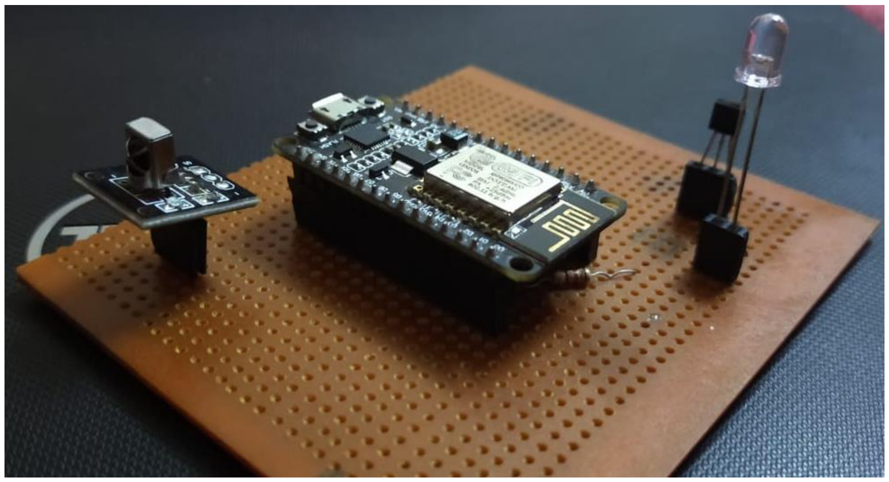


Figure 14: IR transceiver

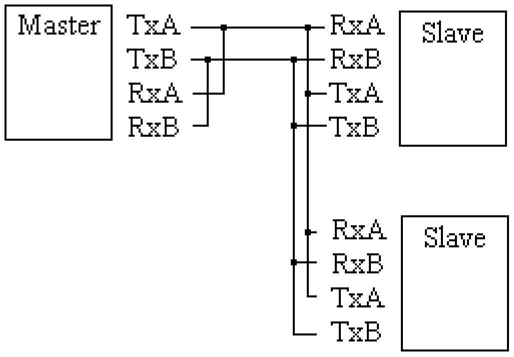
4.4 Development with RS-485

4.4.1 Simplex Communication:

4.4.1.1 Overview

With RS-485, multi-point communication is also possible. But, in our case, Simplex communication or half duplex is tested between Arduino mega and Arduino pro mini using RS-485 module. In this case, Arduino Mega is acting as Master device and Arduino Pro mini is Acting as Slave device.

4.4.1.2 Wiring Diagram



4.4.1.3 Figure

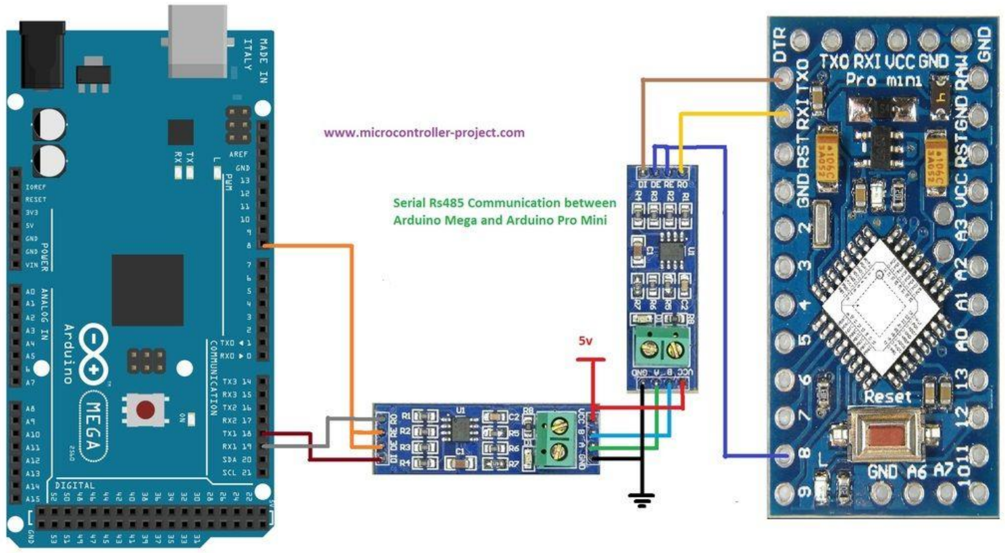

www.microcontroller-project.com
Serial Rs485 Communication between Arduino Mega and Arduino Pro Mini
5v

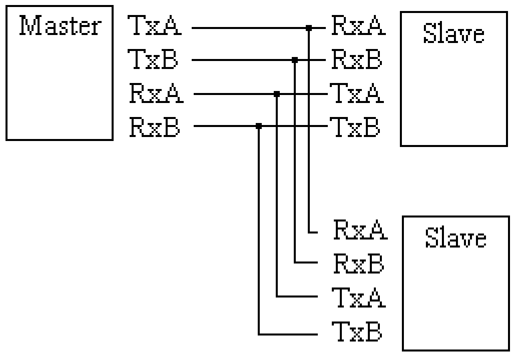
Figure 15: Simplex-Duplex Communication

4.4.2 Full Duplex Communication:

4.4.2.1 Overview

In case of full duplex, the Arduino Mega and Arduino Pro mini are used again. But this time they have the Serial Software ports in them for full duplex mode of communication. This can be useful in cases when we want to send back response to BMS device or any other device which is using Modbus RTU protocol.

4.4.2.2 Wiring Diagram



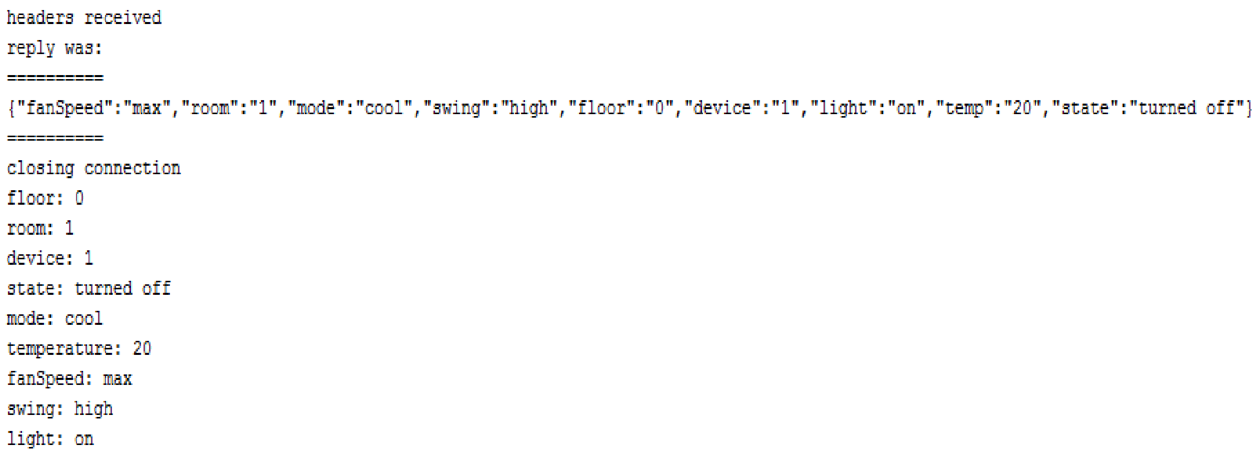
4.5 Design and development of gateway

4.5.1 Overview:

Gateway or Master device is designed using esp32. It has a software serial ports for communication with RS-485 Modbus devices and is also receiving data from the Firebase Cloud using internet. The gateway then processes the data received from the cloud and Modbus devices and send it to respective node over Wi-Fi router.

4.5.2 Processing of Cloud data:

The HTTPS Get request is sent to Firebase. At firebase, a function will be triggered which will send user document stored in Firestore to gateway device. The Json Buffer received in the packet is parsed and the values are stored in string variables.



```
headers received
reply was:
==========
{"fanSpeed":"max","room":"1","mode":"cool","swing":"high","floor":"0","device":"1","light":"on","temp":"20","state":"turned off"}
==========
closing connection
floor: 0
room: 1
device: 1
state: turned off
mode: cool
temperature: 20
fanSpeed: max
swing: high
light: on
```

Figure 16: Cloud Data

4.5.3 Client Configuration:

As, the data will be sent to respective node for execution, we configured client module on the gateway side. This client module send data in the form of array over Wi-Fi router to the respective node.

4.5.4 Circuit Design:

The circuit design for Gateway module is shown below:

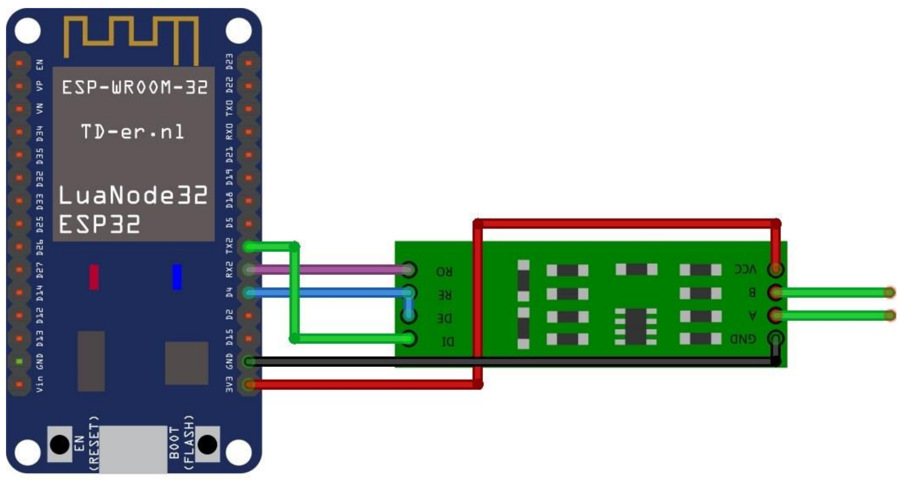


Figure 17: Gateway Circuit Design

4.5.5 Gateway Test Run:

The gateway is then implemented on the bread board for testing.

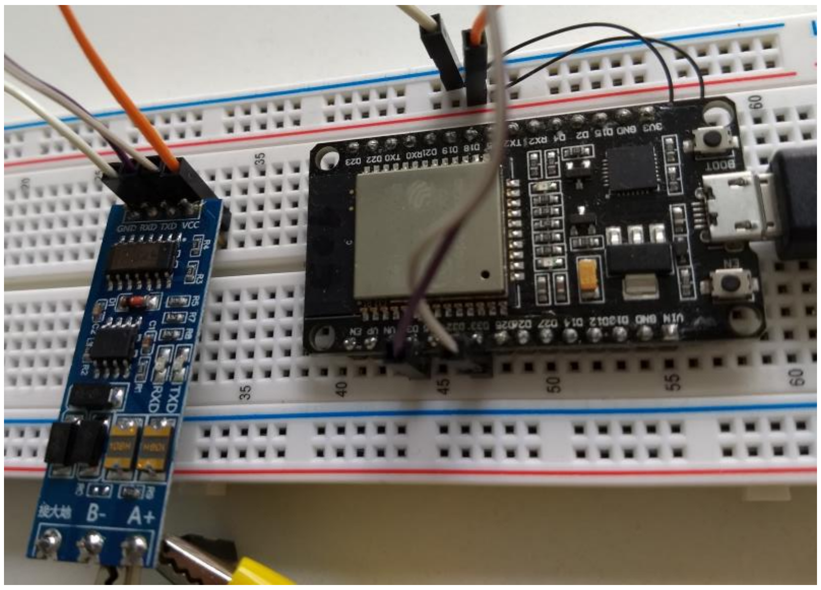


Figure 18: Test Run

4.5.6 Gateway PCB Design:

After testing on bread board, Gateway circuit is designed for the PCB.

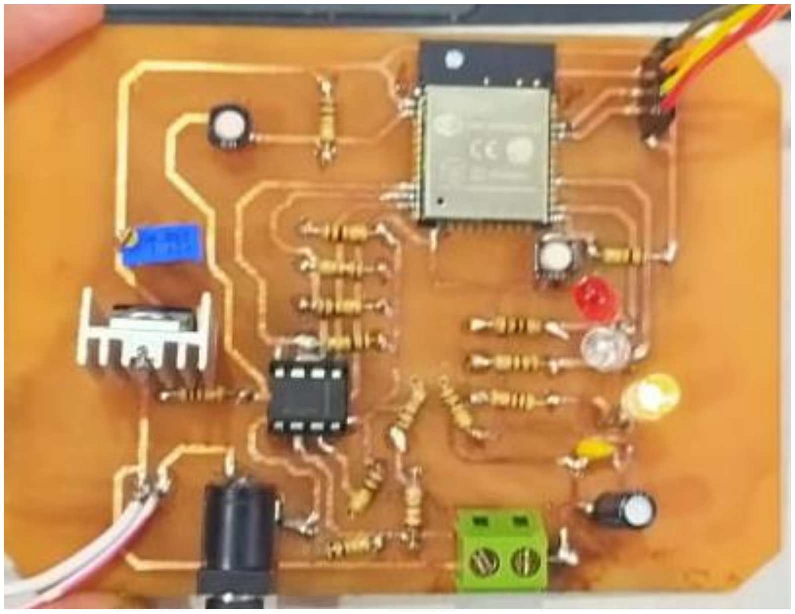


Figure 19: Gateway PCB

4.5.7 Gateway Module:

Finally, the gateway module is set up in a box with a power button on its side.

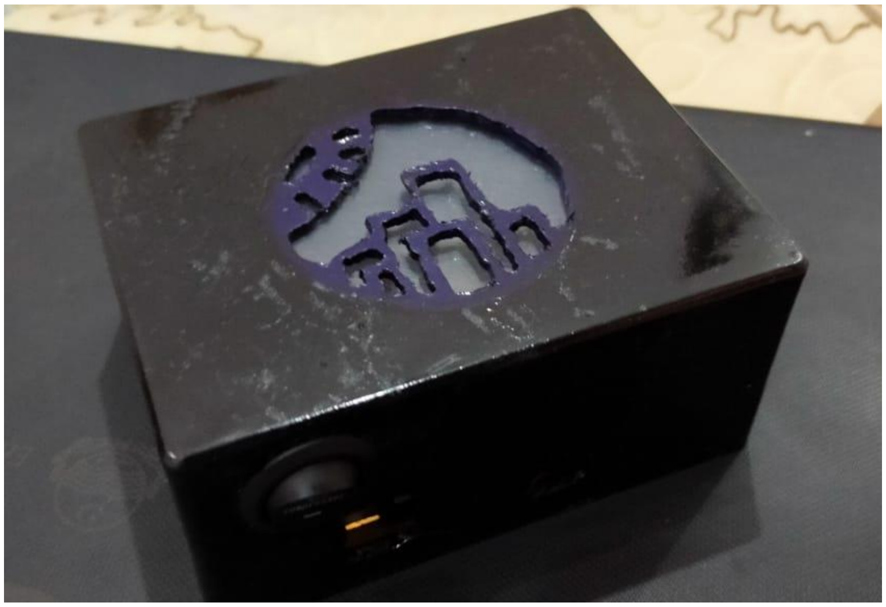


Figure 20: Gateway Module

4.6 Design and development of node

4.6.1 Overview:

To develop a cost-efficient module, we are designing the ESP 12e chip along with IR led to form node module. The node module has a server configured on it. The server receives the commands from gateway and executes the commands on respective HVAC unit.

4.6.2 Server Configuration:

To receive the data, common analogy would suggest the we should configure client on the node. But, to keep the communication smooth, we configured server module on the node side. This server module receives data in the form of array over Wi-Fi router and execute the commands on respective HVAC unit.

4.6.3 Respective library for HVAC:

The respective library and class object are configured and initialized on the node module. This library and object are responsible for execution of HVAC commands.

4.6.4 Circuit Design with ESP8266:

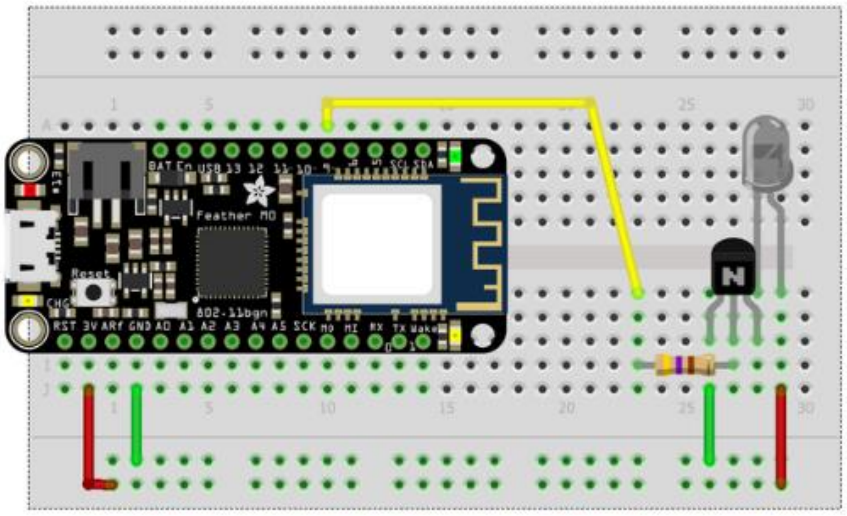


Figure 21: Circuit Design for Node

4.6.5 Programming of ESP12e chip:

The node module was first made and tested with ESP8266. But then converted onto ESP-12e to make it efficient and smooth working module. Circuit design in 4.3.6 shows how the ESP-12e chip is being programmed using ESP8266 board.

4.6.5.1 Features:

The ESP-12e chip on the ESP board was disabled by connecting enable pin (EN) on the ESP board to ground (GND). The program is uploading the same way as we do with any normal ESP board. In this case, our ESP-12e module becomes the ESP-12e module mounted on ESP board and help us get our chip programmed.

4.6.6 Circuit Design with ESP-12e:

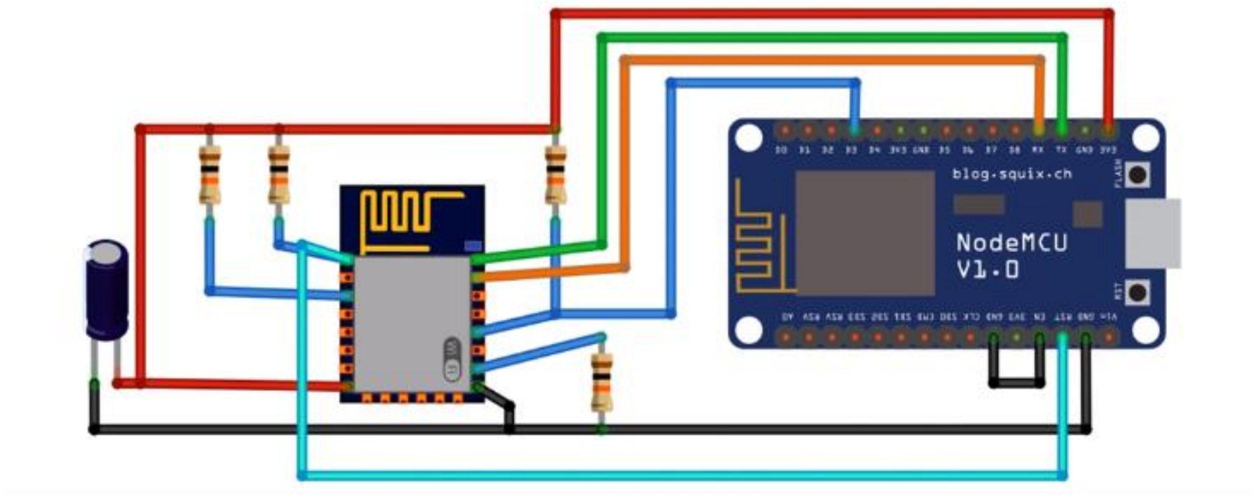


Figure 22: ESP-12e programming circuit

4.6.7 PCB Layout:

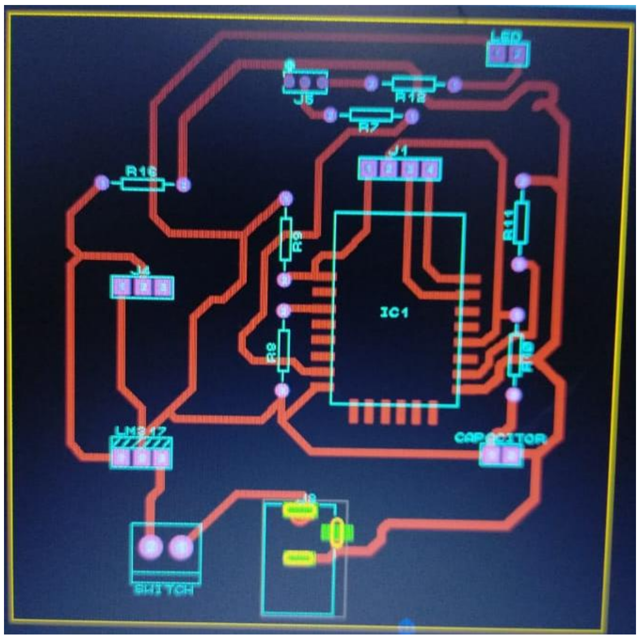


Figure 23: PCB Layout

4.6.8 Node PCB:

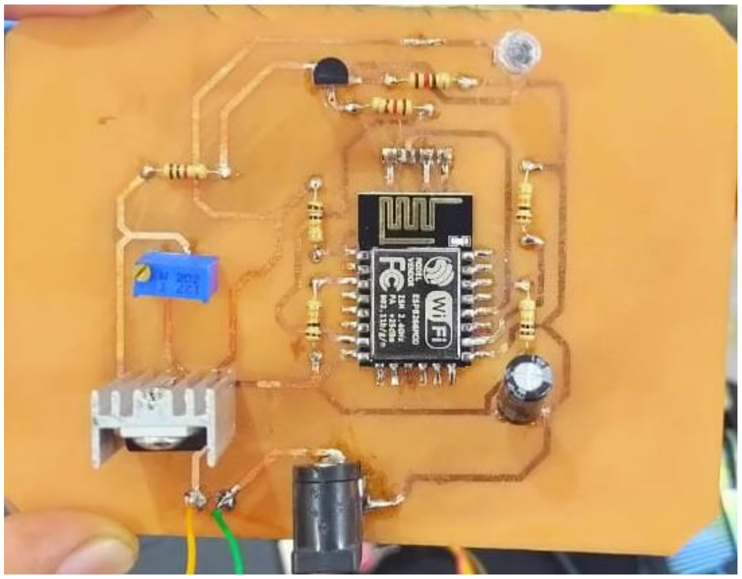


Figure 24: Node PCB

4.6.9 Node Module:

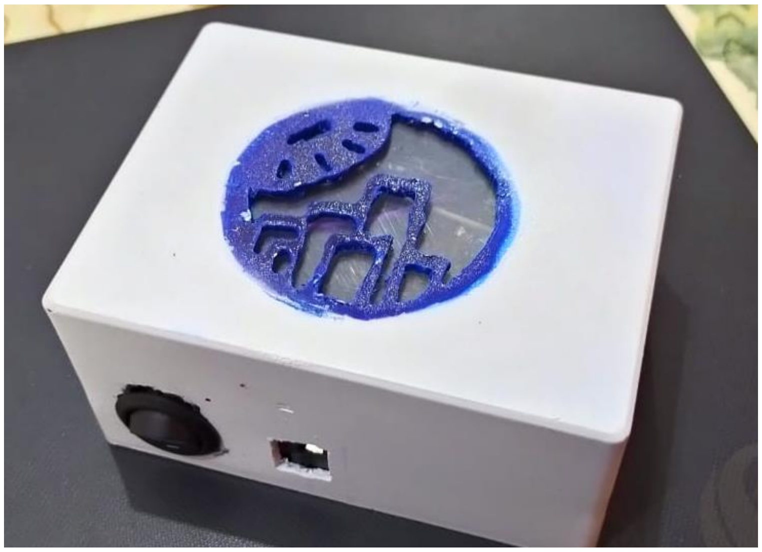


Figure 25: Node Module

4.7 Setting up the Database:

4.7.1 Overview:

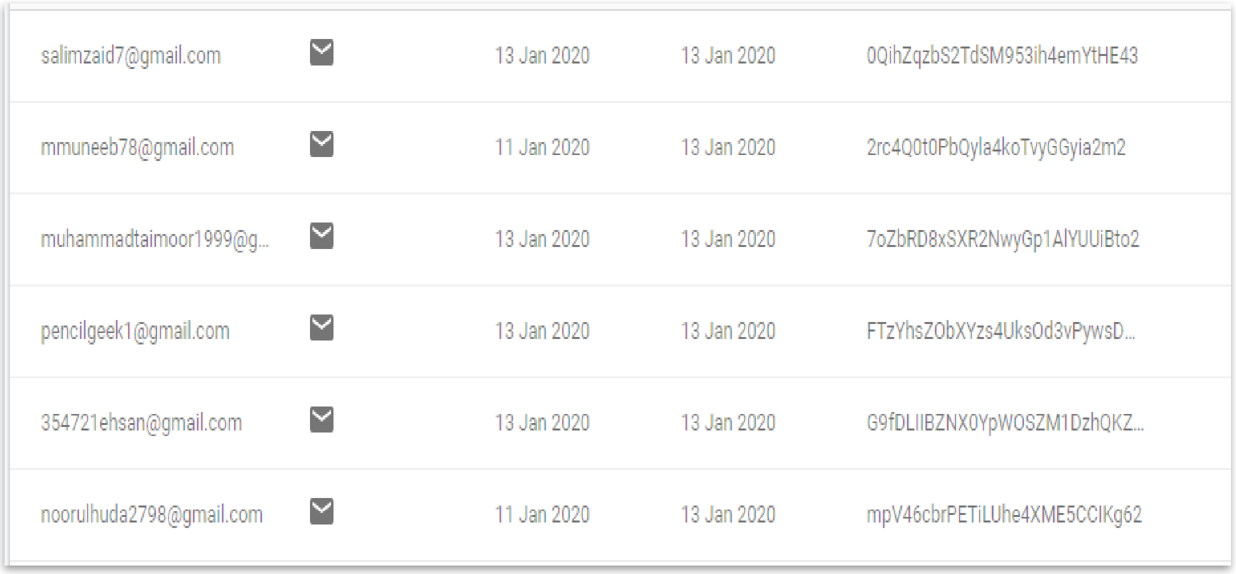
For the data base we are using Google Firebase. It allows the developer to develop a database which can be used as a back end in order to store data as for the APK services. In addition to that it also serves the need of being a real time database and it provides an API, through which user data can be synchronized through all of the devices.

4.7.2 Firebase Project:

A firebase project is created using Bridget's Gmail account. This project creation gives us access to use different features like, user authentication, storage of respective user data in cloud Firestore and real-time synchronization between the user and the Firestore.

4.7.3 Authentication:

The database is accessible through email of the user, specifically Gmail, gets registered in the database using its own unique password and each activity of user is monitored and stored in the database. Every time a user log-in, it gets authenticated first and then the firebase grants it the access to its document

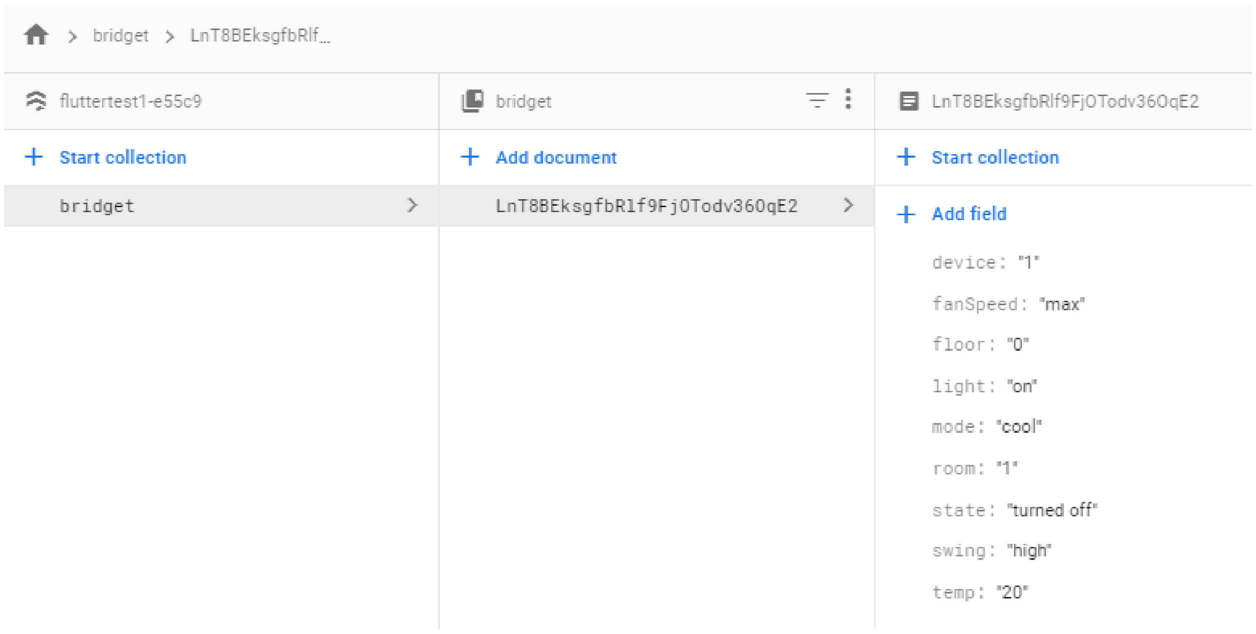


salimzaid7@gmail.com	✉	13 Jan 2020	13 Jan 2020	0QihZqzbS2TdSM953ih4emYtHE43
mmuneeb78@gmail.com	✉	11 Jan 2020	13 Jan 2020	2rc4Q0t0PbQyla4koTvyGGyia2m2
muhammadtaimoor1999@g...	✉	13 Jan 2020	13 Jan 2020	7oZbRD8xSXR2NwyGp1AlYUUiBto2
pencilgeek1@gmail.com	✉	13 Jan 2020	13 Jan 2020	FTzYhsZObXYzs4UksOd3vPywsD...
354721ehsan@gmail.com	✉	13 Jan 2020	13 Jan 2020	G9fDLIIBZNX0YpWOSZM1DzhQKZ...
noorulhuda2798@gmail.com	✉	11 Jan 2020	13 Jan 2020	mpV46cbrPETiLUhe4XME5CCIKg62

Figure 26 Authenticated users

4.7.4 Cloud Fire store:

Cloud Fire store is termed as NoSQL database. It stores and sync data across all users in real-time. Cloud Fire store creates a document against every user. And that document contains all the information required for the user to operate the HVAC unit.



```
device: "1"
fanSpeed: "max"
floor: "0"
light: "on"
mode: "cool"
room: "1"
state: "turned off"
swing: "high"
temp: "20"
```

Figure 27: Fire store Data

4.8 Application Development

A user interface is being provided by a mobile application, which is being developed on **Flutter.** It is an open source platform, enabling the developer to make interactive applications, with flexible, user friendly APKs. It uses **Dart** for coding. Its main advantage over other developing platforms is that its code is compatible with both the Android and IOS and it offers a wide range of built in widgets which can easily be used inside the code. Yet we have designed the registration and authentication page of our application. User can sign up using his/her email address and then can sign in anytime once registered. The authentication is provided by the firebase which lets the user in after checking the password and username.

4.8.1 Register/Sign in:

The application opens-up with an eye-catching **Register** page. It displays two boxes to enter email and password. These credentials are stored in the database and to be used in future for all the sign-in purposes.

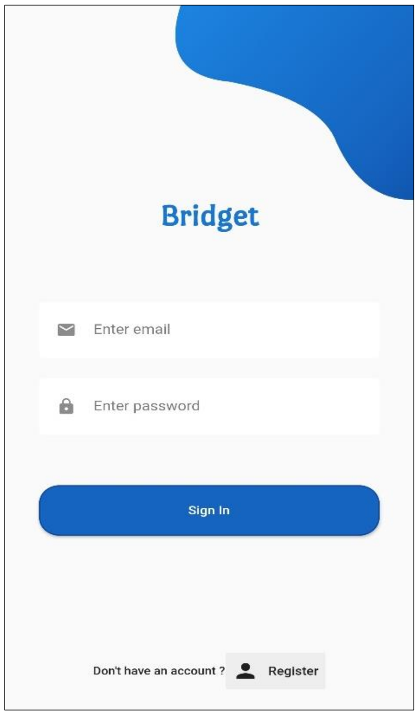


Figure 28: Sign-in

4.8.2 Toggle button:

For the future sign ins, there is a toggle button on the bottom of the register page. It enables you to move between the register and sign in page.

4.8.3 Home Page:

After the successful sign in, user is led to a home page. Home page shows a colorful assortment of buttons and drop-down menus.

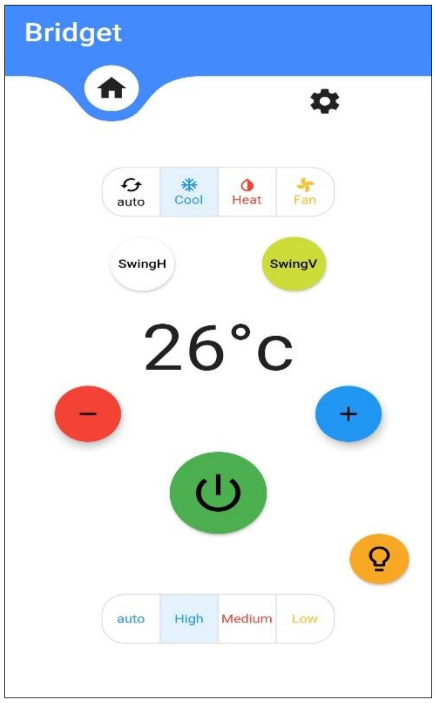


Figure 29: Home Page

4.8.4 Menu Display:

It shows various dropdowns with buttons on side, through these, user can enter settings like:

- Temperature
- Mode
- Fan Speed
- Swing
- Light

4.8.5 Current Values:

Along with changing the options, Home screen also displays the current values on all the user systems/units

4.8.6 Settings Page:

On the top right corner of the Home Page there is a settings icon, which takes user to the settings page. It enables the user to change the settings according to their floor, Air conditioner unit. It has separate buttons for adjusting settings of each AC.

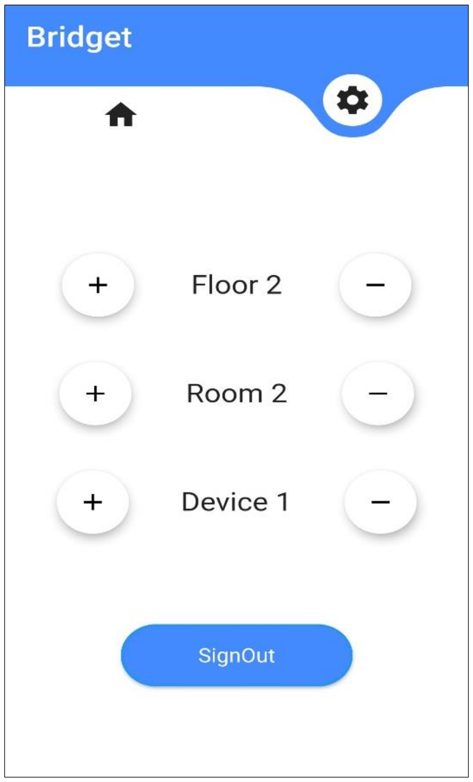


Figure 30: Settings Page

CHAPTER 5

PROJECT INTEGRATION

5 PROJECT INTEGRATION

5.1 Connecting application to database

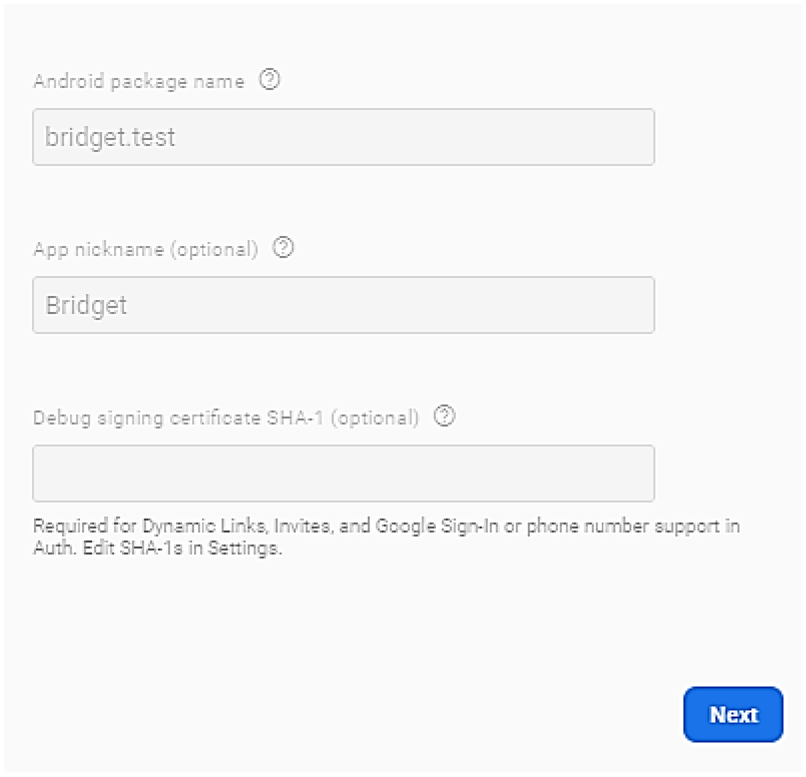
5.1.1 Overview:

Firebase has the ability to let developers integrate their web or mobile application to integrate them with firebase project. And when we create a project on Firebase Database, it provides us with an opportunity to add Bridget app into the Firebase project.

5.1.2 Firebase Connection:

To set-up connection with the firebase:

- **Add android package name into the register the app with firebase**



Android package name: bridget.test

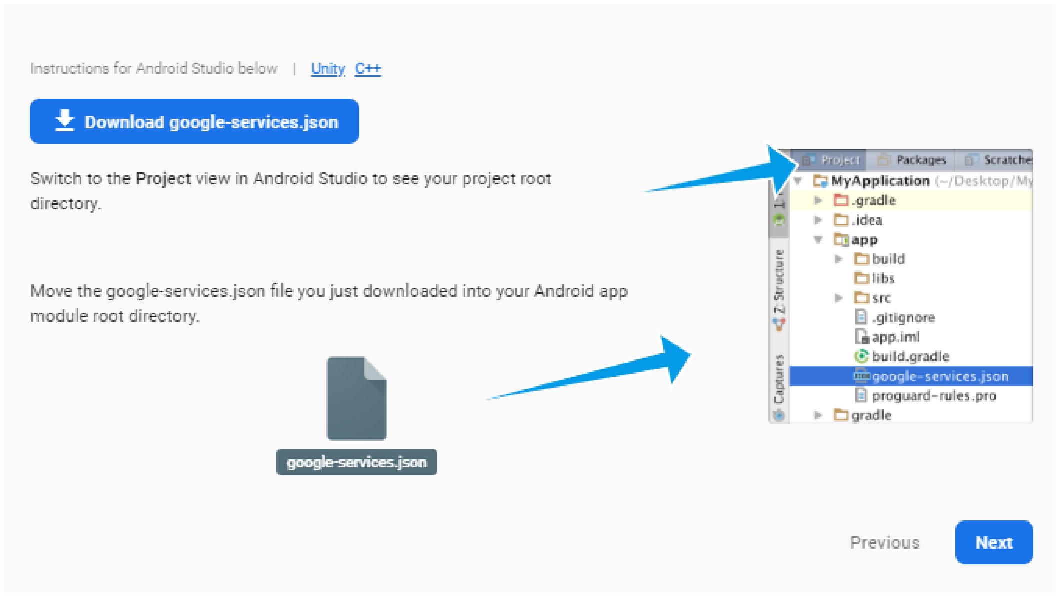
App nickname (optional): Bridget

Debug signing certificate SHA-1 (optional):

Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Figure 31: Firebase Android Package

- **download and placed the Google-service. json package**


Instructions for Android Studio below | Unity C++

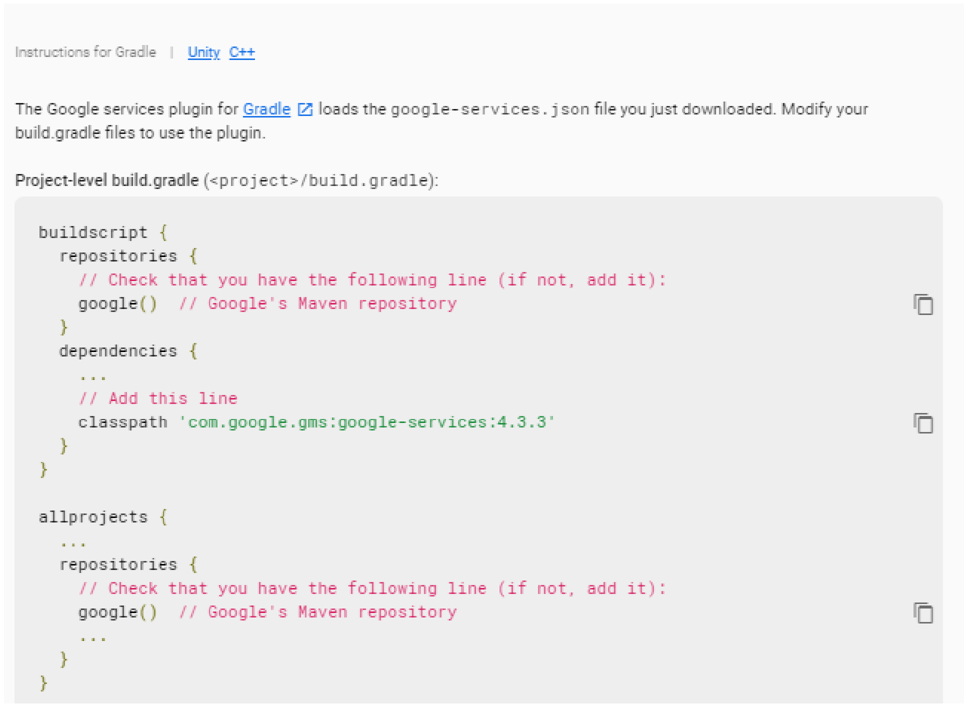
Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Figure 32: Google services dependency

- **Add the firebase SDK in build. gradle files**


Instructions for Gradle | Unity C++

The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

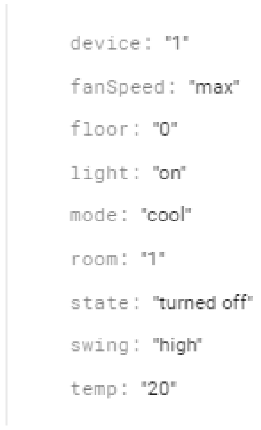
```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.3'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Figure 33: Firebase dependency

5.1.3 HVAC commands from Application:

To control HVAC unit, user will set the desired node to be controlled with in the settings menu. And will then select desired commands from the home menu. The selected commands will be sent to Fire store (Firebase database) in real time. And from there, Gateway unit will get those commands continue the process.



```
device: "1"
fanSpeed: "max"
floor: "0"
light: "on"
mode: "cool"
room: "1"
state: "turned off"
swing: "high"
temp: "20"
```

Figure 34: HVAC commands

5.2 Connecting database to microcontroller

5.2.1 Overview:

Firebase does not quite give access to its Fire store database easily. So, with the help of Express (ES6+) and REST APIs, we establish the link between Gateway device and the Firebase database. Creating Web Sockets, using Snapshots or to Fetch data from Fire store are few other methods. But the method we used involves triggering of Firebase Functions and let us get access to the database and retrieve data from there.

5.2.2 REST APIs:

5.2.2.1 Overview

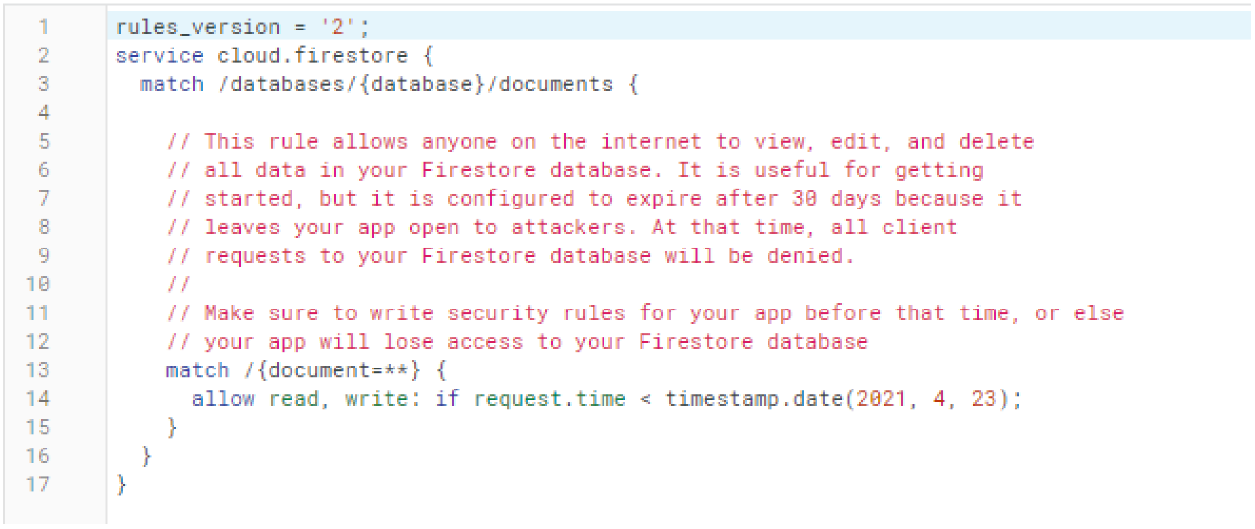
REST stands for "Representational State Transfer". It is a set of rules that developers follow when they create their API. One of these rules states that we get a piece of data when we link to a specific URL.

5.2.2.2 Development

In this project, we are using Firebase Cloud Functions, Firebase Hosting, Cloud Firestore and TypeScript to set up the REST API and get data for the Gateway controller.

- **Firebase Security**

  Before the start, we set the Firebase security rules to allow Read and Write by the user.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone on the internet to view, edit, and delete
    // all data in your Firestore database. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // your app will lose access to your Firestore database
    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2021, 4, 23);
    }
  }
}
```

Figure 35: Firebase Security

- **Firebase Tools**

  Download and Install Node.js then open command prompt. To get all tools needed to build an API, we install "**firebase-tools**" using NPM command.

```
npm install -g firebase-tools
```

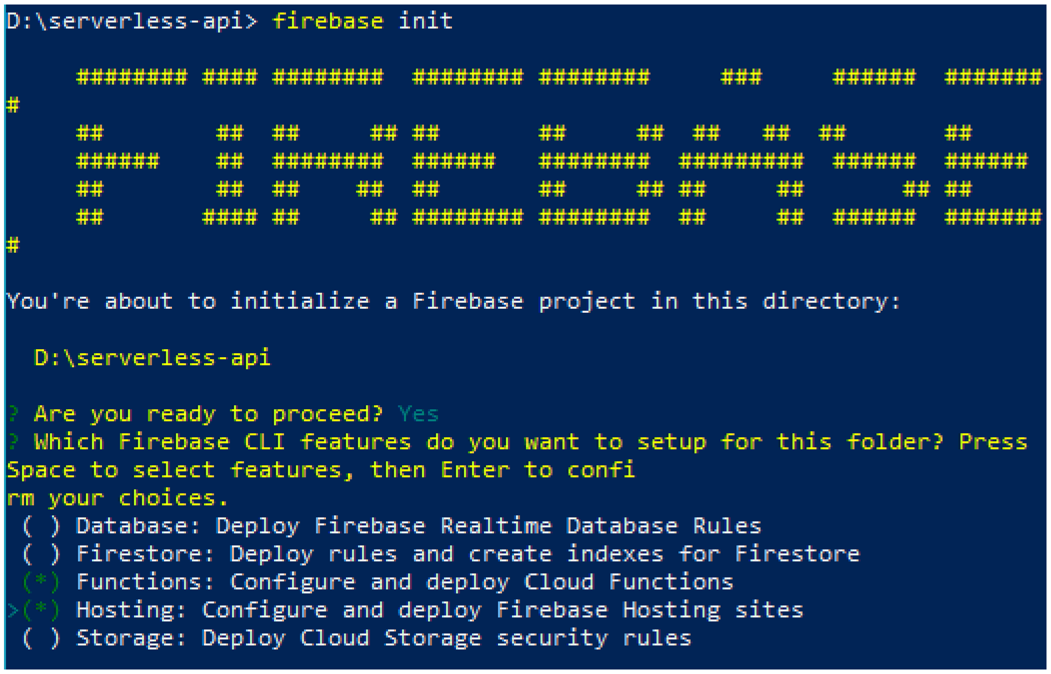
- **Firebase Log in and Initialization**

  Use NPM commands in command prompt to log-in into Firebase project we created earlier. And select the project you want to work on or create new Firebase project.

```
firebase login
firebase init
```

- **Firebase CLI features**

  Once we logged into Firebase, Type "**Yes**" to proceed. And select Hosting and Functions. The functions will be triggered when gateway controller request for the data and Hosting is to give us custom URL for our API project. Select TypeScript as the language for writing functions.



```
D:\serverless-api> firebase init

     ######## #### ########  ######## ########     ###     ######  #######
#
     ##        ##  ##     ## ##       ##     ##  ##   ##  ##       ##
     ######    ##  ########  ######   ########  #########  ######  ######
     ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
     ##       #### ##     ## ######## ########  ##     ##  ######  #######
#

You're about to initialize a Firebase project in this directory:

  D:\serverless-api

? Are you ready to proceed? Yes
? Which Firebase CLI features do you want to setup for this folder? Press
Space to select features, then Enter to confi
rm your choices.
 ( ) Database: Deploy Firebase Realtime Database Rules
 ( ) Firestore: Deploy rules and create indexes for Firestore
 (*) Functions: Configure and deploy Cloud Functions
>(*) Hosting: Configure and deploy Firebase Hosting sites
 ( ) Storage: Deploy Cloud Storage security rules
```

Figure 36: Firebase CLI

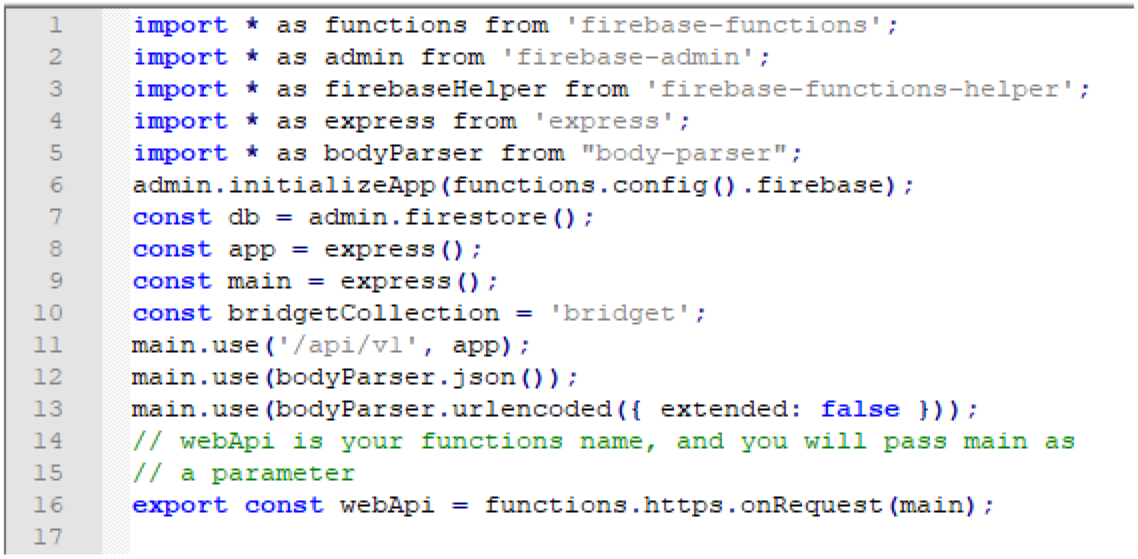
- **Package Dependencies**

  Get the "**Firebase-function-helper**" package inside the functions folder.

```
cd functions
npm install --save express body-parser firebase-functions-
helper
```

- **Functions Writing**

  Import the necessary package and write the functions logic in functions/src/index.ts file.



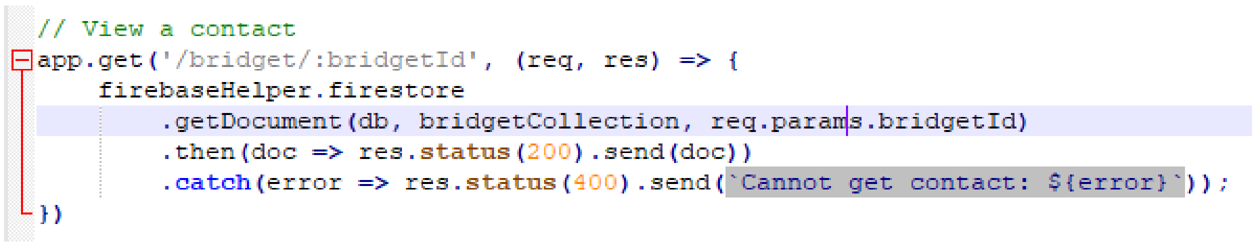
```
import * as functions from 'firebase-functions';
import * as admin from 'firebase-admin';
import * as firebaseHelper from 'firebase-functions-helper';
import * as express from 'express';
import * as bodyParser from "body-parser";
admin.initializeApp(functions.config().firebase);
const db = admin.firestore();
const app = express();
const main = express();
const bridgetCollection = 'bridget';
main.use('/api/v1', app);
main.use(bodyParser.json());
main.use(bodyParser.urlencoded({ extended: false }));
// webApi is your functions name, and you will pass main as
// a parameter
export const webApi = functions.https.onRequest(main);

```

Figure 37: Functions Definition

- **CRUD Routing**

  The functions/src/index.ts file also holds the CRUD routing, meaning it consists of information whether we want to post data to Cloud Fire store database or to get data from there and in what form.



```
// View a contact
app.get('/bridget/:bridgetId', (req, res) => {
    firebaseHelper.firestore
        .getDocument(db, bridgetCollection, req.params.bridgetId)
        .then(doc => res.status(200).send(doc))
        .catch(error => res.status(400).send(`Cannot get contact: ${error}`));
})
```

- **Deploy**

  Before deploying our API to firebase, there are some changes to be updated in json file

```
firebase deploy
```

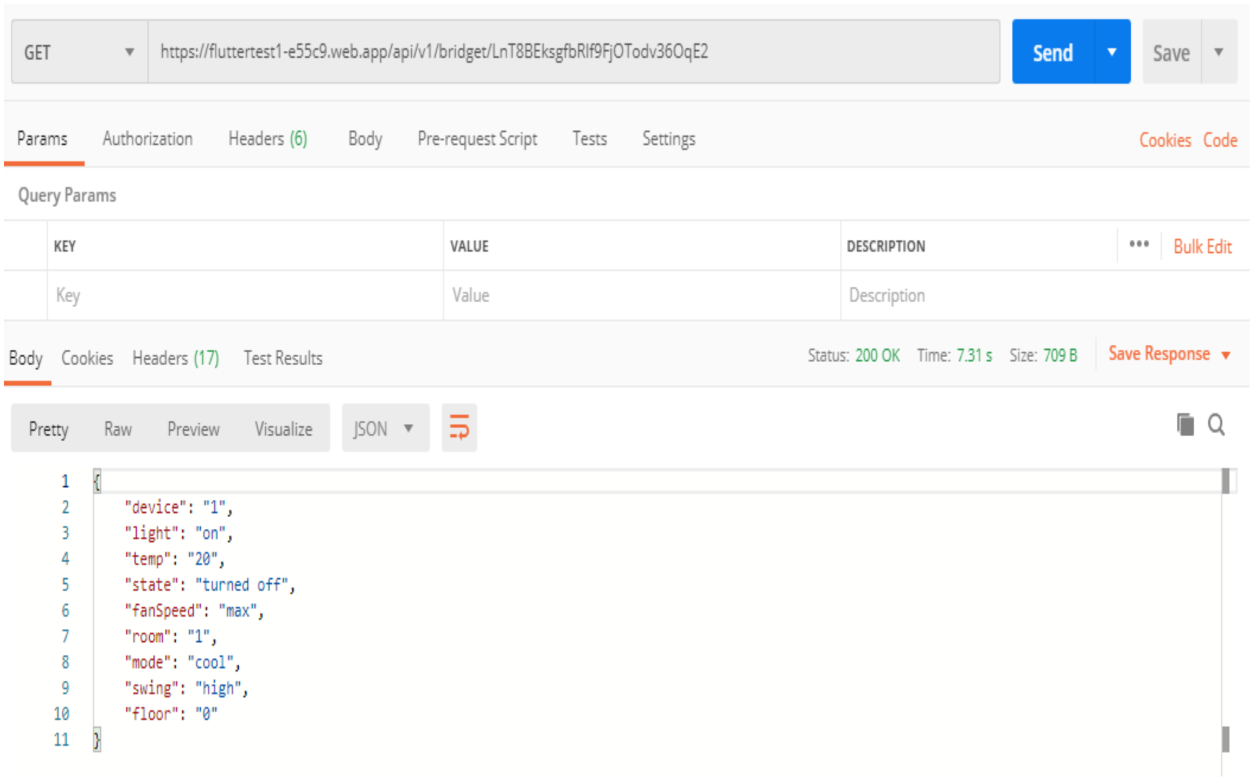
- **URL Mapping**

  The Firebase Hosting function will map the Firestore database URL, so that web-API will trigger when we call the URL.

```
https://serverless-api-dn.firebaseapp.com/api/v1/**
```

5.2.2.3 Testing

To test our API, we use Postman software. This software is used to test the web-APIs and the Post or Get data using API-provided URL. The photo of the test is shown below:



```
GET https://fluttertest1-e55c9.web.app/api/v1/bridget/LnT8BEksgfbRIf9FjOTodv36OqE2
```

```
Status: 200 OK   Time: 7.31 s   Size: 709 B
```

```
{
    "device": "1",
    "light": "on",
    "temp": "20",
    "state": "turned off",
    "fanSpeed": "max",
    "room": "1",
    "mode": "cool",
    "swing": "high",
    "floor": "0"
}
```

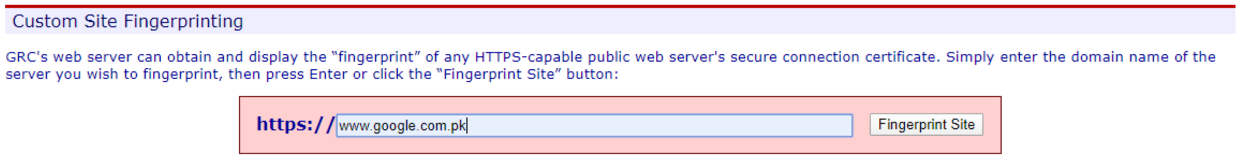
5.2.3 Fingerprint:

5.2.3.1 Overview

Fingerprints can be used to detect users or devices even fully or partially when cookies are turned off. And with the help of that, we use fingerprint to initiate handshake with Firestore database server.

5.2.3.2 GRC (Gibson research corporation)

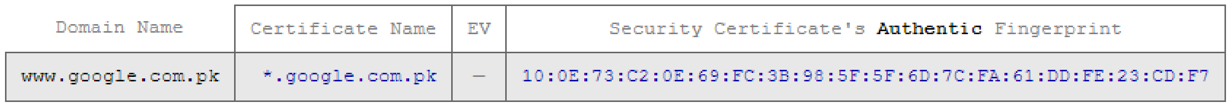
It is a custom fingerprint identification site. GRCs server help us identify the HTTPS-capable Firestore Database server. From which we retrieve data using HTTPS requests.


Custom Site Fingerprinting

GRC's web server can obtain and display the "fingerprint" of any HTTPS-capable public web server's secure connection certificate. Simply enter the domain name of the server you wish to fingerprint, then press Enter or click the "Fingerprint Site" button:

5.2.3.3 Firebase fingerprint

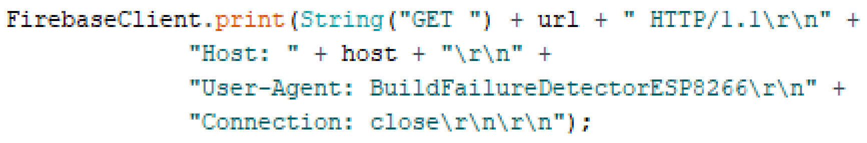
The GRC server gives us the fingerprint of Firestore server security certificate. But it does not help us establish a constant connection with the server. Google server certificate example has been used here.



Domain Name	Certificate Name	EV	Security Certificate's **Authentic** Fingerprint
www.google.com.pk	*.google.com.pk	—	10:0E:73:C2:0E:69:FC:3B:98:5F:5F:6D:7C:FA:61:DD:FE:23:CD:F7

5.2.4 HTTPS Requests:

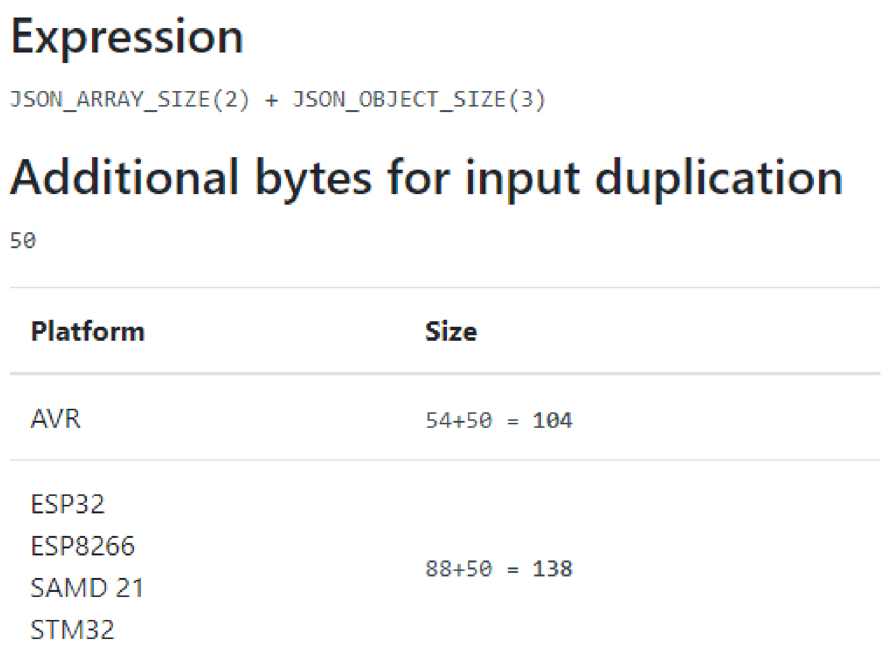
As the firebase is HTTPS based site. So, we use the fingerprint identified by GRC server to approach with the HTTPS GET request every time we need to send commands to HVAC units. It sends the packet to firebase every time including Host name and Port number in its header.



```
FirebaseClient.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "User-Agent: BuildFailureDetectorESP8266\r\n" +
             "Connection: close\r\n\r\n");
```

5.2.5 Json Array:

The Json Array captured by the https packet will be dynamically parsed and the property values will be distinguished according to the properties and thus stored in string arrays. This array can be parsed statically but it would the buffer size will have to be large enough to handle data variations which, in most cases, is difficult to determine. And while dynamically parsing the array, dynamic object will automatically calculate the data size and allocate required buffer space to it.



Expression

```
JSON_ARRAY_SIZE(2) + JSON_OBJECT_SIZE(3)
```

Additional bytes for input duplication

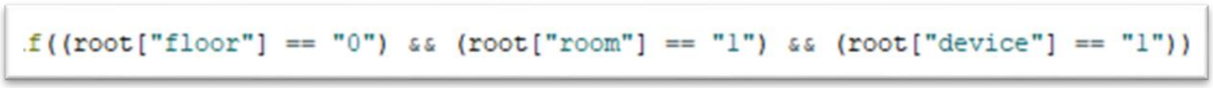
```
50
```

Platform	Size
AVR	54+50 = 104
ESP32 ESP8266 SAMD 21 STM32	88+50 = 138

5.3 Connection with the nodes

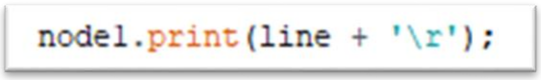
5.3.1 Gateway Configuration:

An “if” condition is applied at gateway to identify which node will receive the HVAC commands received by Json buffer.



```
f((root["floor"] == "0") && (root["room"] == "1") && (root["device"] == "1"))
```

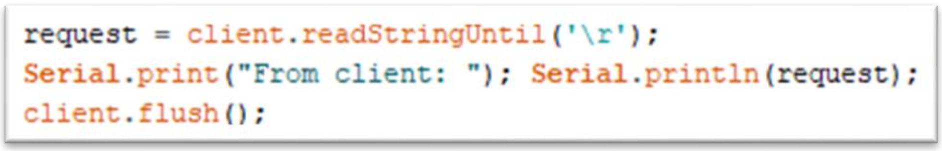
Technically, gateway will act as a client and send the executable commands to respective node.



```
node1.print(line + '\r');
```

5.3.2 Node Configuration:

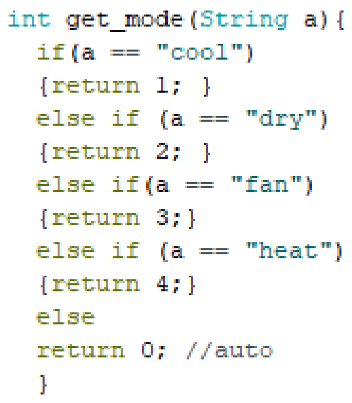
At node, a server is established, and it is in listening mode all the time. It listens for the data packet from gateway. When it gets it, it stores and display the data packet through an array.



```
request = client.readStringUntil('\r');
Serial.print("From client: "); Serial.println(request);
client.flush();
```

5.3.2.1 Translation of Commands

HVAC commands received from application are in the form of string. But the functions in HVAC libraries which are designed and modified takes only Integer variables as input. And Every HVAC unit has its own modulation and set of commands. Although some has same set of variables required but sometimes their arrangement differs. So, to cater that, we develop a unique translation scheme for every HVAC unit. For example:

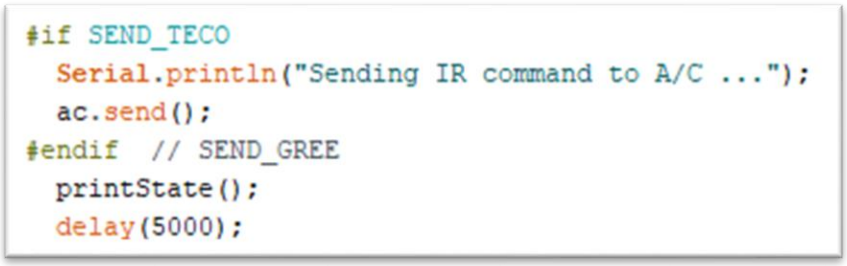


```
int get_mode(String a){
  if(a == "cool")
  {return 1; }
  else if (a == "dry")
  {return 2; }
  else if(a == "fan")
  {return 3;}
  else if (a == "heat")
  {return 4;}
  else
  return 0; //auto
  }
```

Figure 38: Translation of HVAC commands

5.3.2.2 Command Execution

For commands execution, every HVAC unit has its own Class and Objects define and declared in the node. And they execute the translated commands via IR transmitter circuit designed for this very purpose.



```
#if SEND_TECO
  Serial.println("Sending IR command to A/C ...");
  ac.send();
#endif  // SEND_GREE
  printState();
  delay(5000);
```

CHAPTER 6

ANALYSIS AND EVALUATION

6 ANALYSIS AND EVALUATION

6.1 Project block Diagram

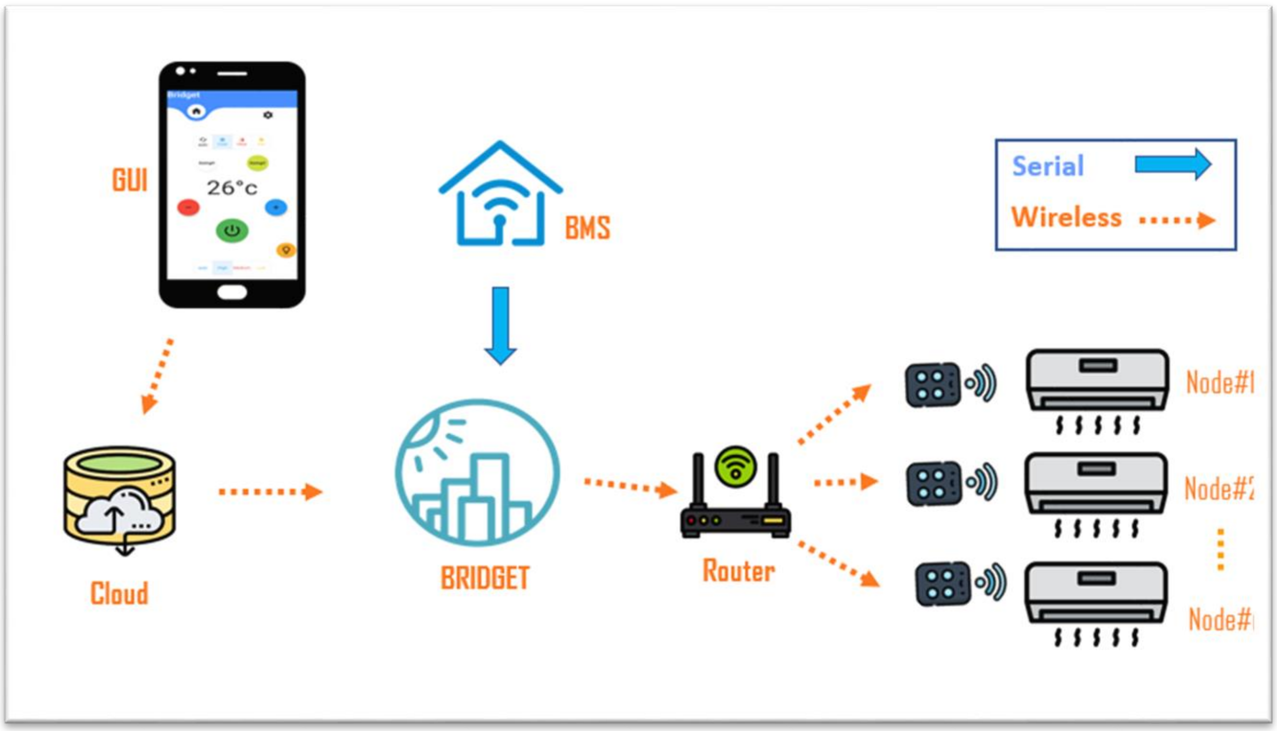


Figure 39: Block Diagram

6.2 Controlling of HVAC system with Bridget App

To control the device with the help of application a user manual is designed, which is attached in Annex A.

6.3 Objectives Achieved

At the completion of project, the objectives achieved are:

- Successful identification of different nodes through the centralized database
- Transmission and Receiving of commands from the user end to the nodes
- Controlling of different make and models of different AC units via commands through app
- Successful update of database server upon entry/ changing of any parameter

CHAPTER 7

FUTURE WORK

7 FUTURE WORK

7.1 EXTENDED SCOPE

Where the scope of our project fulfills its set objectives, it does entail room for improvement as does any innovative solution.

7.1.1 HVAC Monitoring:

Bridget extensions may include individual units' real time monitoring. This would allow the user to monitor individually selected units.

7.1.2 Voice Commands:

The mobile application can be extended to include voice commands and recognition. The overall response time can also be reduced by using dedicated servers as well as app upgrades over the time.

CHAPTER 8

CONCLUSION

8 CONCLUSION

The objective behind automating HVAC commands is basically attributed to increased productivity and efficiency. Bridget, in addition to providing higher output rates, cuts down on labor hours, hence saving time. By providing control to the user of several individual HVAC units over an easy to use mobile app, Bridget allows the tedious task of individually controlling each and every unit to become just a tap of a button. The HVAC units are controlled via the mobile app which in turn is user specific ensuring security. The user specific approach also aids in database updates, keeping log of the individual units and their responses. Bridget ensures a more efficient use of a valuable resource, time and helps cut down on unnecessary labor exertion.

9 BIBLIOGRAPHY

1. P. I-Hai Lin and H. L. Bromberg, **"Internet-based monitoring and controls for HVAC applications",** in IEEE Industry Applications Magazine, vol. 8, no. 1, pp. 49-54, Jan.-Feb. 2002.
2. Jignesh Bhatt, H. K. Verma, **"Design and Development of Wired Building Automation Systems"**, in Energy and Buildings, Vol. 103, pp. 396-413, 15 September 2015.
3. Jooma Matikainen**, "Controlling RS232 equipped devices using Raspberry Pi and C++",** Metropolia University of Applied Sciences Bachelor of Engineering Degree Program in Information Technology, Thesis, 29 November 2018.
4. https://itnext.io/building-a-serverless-restful-api-with-cloud-functions-firestore-and-express-f917a305d4e6
5. https://www.grc.com/fingerprints.htm
6. https://arduinojson.org/v5/assistant/
7. https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/
8. http://ckp.made-it.com/rs485.html
9. https://hobbycomponents.com/wired-wireless/663-max485-rs485-transceiver-module
10. https://hallroad.org/ch340-lolin-nodemcu-v3-lolin-based-esp8266-wifi-development-board-iot-development-board-in-pakistan.html
11. https://hallroad.org/esp32.html
12. https://hallroad.org/esp-12-esp8266-wifi-module-wireless-iot-board-module.html
13. https://github.com/esp8266/Arduino
14. https://github.com/esp8266/Arduino/tree/master/libraries
15. https://github.com/crankyoldgit/IRremoteESP8266
16. https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi
17. https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFiClientSecure
18. https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiClientSecure.h
19. https://github.com/bblanchon/ArduinoJson
20. https://www.arduino.cc/en/reference/SPI
21. https://flutter.dev/

ANNEX A

USER MANUAL

Contents

List of Figures

1 GENERAL INFORMATION

This section explains the general information in simple and clear words, which will make even layman to understand the working of Bridget

1.1 System Overview

Bridget is an automation device, transforming your hectic routines into hassle-free, modern lifestyle. It will give you the magnificent experience of controlling your gigantic devices with a tap of your finger.

1.2 Organization of Manual

The manual is divided into four main sections:

- General Information
- Product Overview
- Getting Started
- Trouble Shooting

2 PRODUCT’S OVERVIEW

Bridget comes with two main components:

2.1 Gateway/ Master Device

It is the main device responsible for all the major decisions. It can be placed anywhere inside the building/house, preferably near your BMS and the internet router.

2.2 Nodes

These are the small devices, which will be mounted on a wall. You can buy nodes according to the number of ACs you have. Each unit will have one node, mounted underneath it.

2.3 Mobile Application

It can be downloaded and installed in your phone. It requires two easy steps to register and then sign in, and you will be ready to go.

3 GETTING STARTED

First, the hardware needs to be setup. After that you need to install the app and register on our data base.

3.1 Register/ Sign up

For the new customers, app provides an interface for registration. You need to get registered using your email address. These credentials will come in handy, next time you will login.

3.2 Sign In

Existing users will sign in using the email and password provided at the time of registration. Once you have signed in successfully, you will see our home page.

3.3 Home

In the home screen, you will see a number of options. Here, the settings button on the top right corner will take you to the settings screen.

In the figure below, you can see several options. You can turn the device ON/OFF, change the temperature, change the mode, and fan settings of your AC.

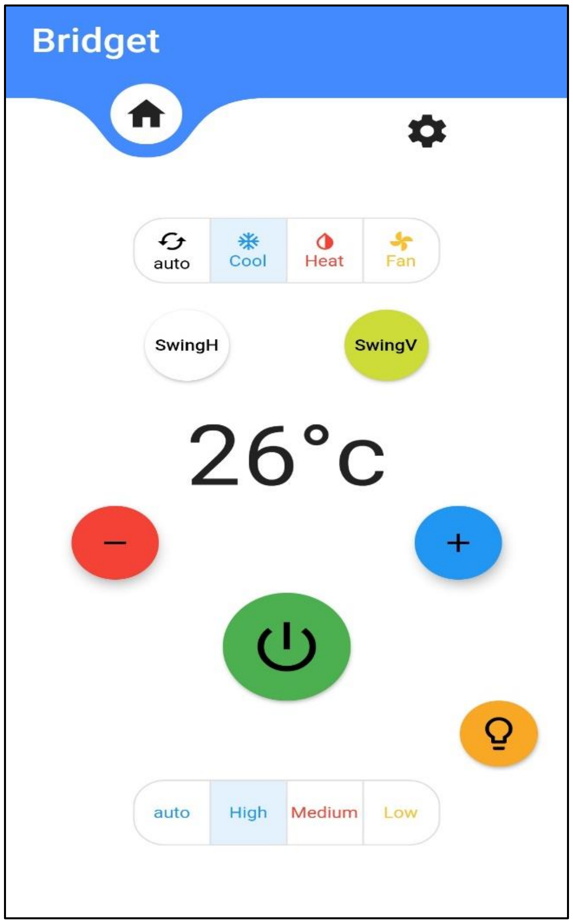


Figure 1: Home Screen

After successfully setting up the parameters, you can repeat the same for other units too.

3.4 Setting the Parameters

Through the settings screen, you can choose the particular device which you want to operate. It offers you three buttons to change device, room and floor.

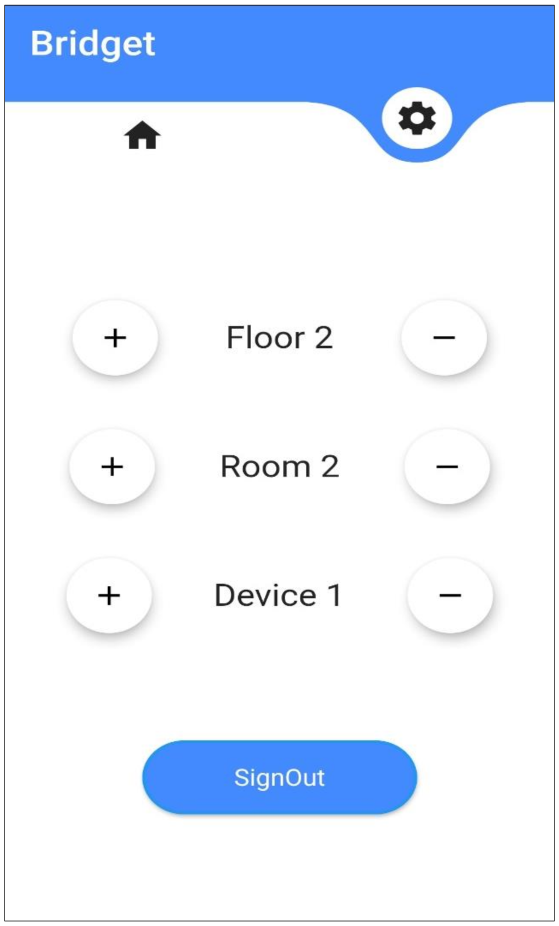


Figure 2: Selecting Node

Here you need to enter the floor, room, and device number of that particular unit, you wish to control as shown in the figure 2.

4 TROUBLESHOOTING

During the usage, if you experience any kind of glitches then you can trouble shoot the device by:

- Checking the availability of internet
- Making sure that gateway device is placed near the router
- Checking that the IR sensor of node is in perfect line of sight with the IR receiver of AC