**PROJECT REPORT**

***on***

# SMART HUB for domestic appliance switching

*submitted by*


| NAME | SAP ID |
|---|---|
| **Jugal Manek** | **60002198007** |
| **Jash Shah** | **60002180044** |
| **Aman Sharma** | **60002180009** |


*under the guidance of*

**Prof. Sanjay Deshmukh**

(Assistant Professor)


**DEPARTMENT OF**

**ELECTRONICS AND TELECOMMUNICATION ENGINEERING**


**Academic Year: 2021-2022**

**Shri Vile Parle Kelavani Mandal's**

# Dwarkadas J. Sanghvi College of Engineering

Plot no. U-15, JVPD Scheme, Bhaktivedanta Swami Marg,
Vile Parle (W), Mumbai – 400 056

# DECLARATION

*We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.*

*Name* *Signature*

**Jugal Manek - 60002198007**

**Jash Shah - 60002180044**

**Aman Sharma - 60002180009**

*Date: 01/04/2022*

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

## Department of Electronics and Telecommunication Engineering

This is to certify that the Project Stage – II

# " SMART HUB for domestic appliance switching "

Submitted by:


1. **Jugal Manek - 60002198007**
2. **Jash Shah - 60002180044**
3. **Aman Sharma - 60002180009**


Students of **Electronics and Telecommunication Engineering** have successfully completed their **Project Stage – II** required for the fulfillment of **SEM VIII** as per the norms prescribed by the **University of Mumbai** during the First half of the year 2022. The project report has been assessed and found to be satisfactory.

| | |
|---|---|
| **Internal Guide** | **External Guide** |
| **Head of Department** | **Principal** |
| **Internal Examiner** | **External Examiner** |

# ACKNOWLEDGEMENTS

The satisfaction and euphoria that accompany the successful completion of our project would be incomplete without mentioning the people whose constant guidance and encouragement made it possible. We take pleasure in presenting you, our project, which is the result of a studied blend of both research and knowledge.

We owe deep gratitude to our project guide, Prof. Sanjay Deshmukh, Department of Electronics and Telecommunication, who took a keen interest in our project work and guided us all along, till the completion of our project by providing all the necessary information and support till the final stage of our project.

We would also like to thank Dr. Amit Deshmukh, the Head of Department, for giving us the opportunity to showcase our knowledge in a more practical way.

We would also like to express our deep gratitude to Dr. Hari Vasudevan for giving us this opportunity to work on a project.

Finally, we express our gratitude to all the members who are involved either directly or indirectly in the completion of this project.

# TABLE OF CONTENTS

**Page**

# ABSTRACT

With the advent of embedded systems Technology,we are at the doorstep of domesticating Advanced Technological concepts incorporated into products that include (but are not limited to )embedded systems, the internet of things, connected devices, computer networks ,mobile communication etc. This statement is justified by the rising trend in home automation and connected devices in the Western world ,the Eastern part of the globe seems to have been left behind in the race owing to lack of options in the market, high costs and maintenance charges,need to change in initial wiring bulky and monotonous systems sold in the name of Home Automation.

The aim is to build a prototype which would lead to a product solving a common need in home automation in households with pre-existing wirings and installed switchboards.
The Unique Selling Points of this project are:

- Affordability
- Versatility
- Retrofit
- State of art

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| Abbreviation | Full Form |
|---|---|
| USP | Unique Selling Point |
| AC | Alternating Current |
| iOS | iPhone OS |
| IoT | Internet of Things |
| CFL | Compact fluorescent lamp |
| TL | Tube luminescent |
| LED | Light Emitting Diode |
| Wi-Fi | Wireless Fidelity |
| PCB | Printed circuit boards |
| SoC | System on Chip |
| RF | Radio Frequency |
| SRAM | Static Random Access Memory |
| ROM | Read-Only Memory |
| RTC | Remote Time Clock |
| BLE | Bleutooth Low Enery |
| GPIO | General Purpose Input/Output |
| SAR | Successive Approximation Register |
| ADC | Analog-Digital Converter |
| DAC | Digital-Analog Converter |
| SPI | Serial Peripheral Interface |
| I2C | Inter-Integrated Circuit |
| I2S | Integrated Inter-IC Sound Bus |
| UART | Universal Asynchronous Receiver-Transmitter |
| MAC | Media Access Control |
| LAN | Local Area Network |
| SD | Secure Digital |
| SDIO | Secure Digital Input/Output |

| | |
|---|---|
| PWM | Pulse Width Modulation |
| IDE | Integrated Development Environment |
| UI | User Interface |
| WLAN | Wireles Local Area Network |
| DSSS | Direct-Sequence Spread Spectrum |
| MIMO | Multiple-Input Multiple-Output |
| QoS | Quality of Service |
| VoIP | Voice over Internet Protocol |
| API | Application Programming Interface |
| QR | Quick Response |
| LORAWAN | Long Range Wide Area Network |

# CHAPTER 1

# INTRODUCTION

With the advent of embedded systems Technology, we are at the doorstep of domesticating Advanced Technological concepts incorporated into products that include (but are not limited to) embedded systems, the internet of things, connected devices, computer networks ,mobile communication etc. This statement is justified by the rising trend in home automation and connected devices in the Western world ,the Eastern part of the globe seems to have been left behind in the race owing to lack of options in the market, high costs and maintenance charges, need to change in initial wiring bulky and monotonous systems sold in the name of Home Automation.

Our project aims at bringing Home Automation systems to the common household with absolute ease of use. Our project is based on the following USP's (unique selling points) that are Affordability, Versatility Retrofit and State of the art systems designed to provide seamless voice control Feather Touch "plug and play" home automation system with the least possible costs and fuss.

The idea here is to create a practical Home automation board that can fit into our AC power units on our walls and stay concealed inside it. The board should not interrupt the normal working of our power unit switches, that is they should turn ON or OFF with manual switches as well. And without being said, it should also be able to control the same load with voice commands too. Since this brings in various technologies, the project gets interesting and practical to use.

The system would also comprise of various other features such as voice control and will also have sensors integrated into the board, which would complement the home automation system. The aim is to build a prototype which would lead to a product solving a common need in home automation in households with pre-existing wirings and installed switchboards.

# LITERATURE SURVEY

Home Automation is a product or service that brings some level of action or message to the home environment, an event that was generated without the homeowner's direct intervention. The words "home" and "automation" fit together perfectly to describe how to get things done easier, better, and faster than ever before, which equates to convenience.

Home automation technology has changed a great deal over the decades. The way that home automation worked in whole-home solutions of the past was that a central controller was placed in the home and the user controlled it through wall-mounted keypads and remote control boxes.

These days, whole-home solutions are much improved and incorporate Internet and network-based technologies to connect you to your home. We can also remotely control our home using smartphones and tablets. The operating systems are typically customized for their clients, though, as is everything in a whole-home system. Modular Internet and network-based solutions that utilize apps and the web are taking the market by storm, due to affordability, convenience, and the capability to be added on to.

Another factor in their growth is that folks are familiar with their modes of usage. Lots and lots of people have smartphones and tablets today, and most of those devices are running on iOS or Android operating systems. The modular home automation solutions allow us to buy products that you interact with through other products (our smart devices and home network) that we are already familiar with.

Future for the Home Automation systems involves making homes even smarter. Homes can be interfaced with sensors including motion sensors, light sensors and temperature sensors and provide automated toggling of devices based on conditions. More energy can be conserved by ensuring occupation of the house before turning on devices and checking brightness and turning off lights if not necessary. The system can be integrated closely with home security solutions to allow greater control and safety for home owners.

# PROPOSED SOLUTION

Our project aims at bringing Home Automation systems to the common household with absolute ease of use. Our project is based on the following USP's (unique selling points) that are Affordability,Versatility Retrofit and State of the art systems designed to provide seamless voice control Feather Touch "plug and play" home automation system with the least possible costs and fuss.

The idea here is to create a practical Home automation board that can fit into our AC power unit son our walls and stay concealed inside it. The board should not interrupt the normal working of our power unit switches,that is they should turn ON or OFF with manual switches as well.And without being said, it should also be able to control the same load with voice commands too.Since this brings in various technologies,the project gets interesting and

Practical to use.

The system would also comprise of various other features such as voice control and will also have sensors integrated into the board, which would complement the home automation system.

Thus. by using technologies like IOT, cloud computing and mobile commu nications and Embedded systems we can make Home automation experience of customers smooth and easy as well as make empower other sectors where this systems can be incorporated .

# CHAPTER 2

# THEORY

## HOME & LIGHTING AUTOMATION

What is automation? It places the world of connected things at your fingertips. Easily control the everyday world & make it more intelligent, convenient, secure, safe, efficient, and fun. From control to convenience, office to living room, from your TV screen to the projector screen, you can control more, save more & simplify more effortlessly.
our Automation that is,
● Affordable
● Convenience rather than luxury
● Modular
● Non-Stagnant platform
● Energy Saving
● Cost effective
Also it provides you the freedom to Unify all of the below!
● Lighting – ON/OFF, Dimming, Fan speed control
● Dimming – Halogen, CFL, TL, PL, LED
● Motorized Mechanism – Curtains, Roller Blinds, Shutter, Projector Screen
● AV Device – TV, Set-top box, Music System, DVD Player, Blue-ray
Video Door Phone & Door lock
● Security System & sensors
● Integrate any make switch

Touch Switches
They are touch-sensitive Control Pads. Or, in other words, Intelligent Switches. That means the most basic models are the modern substitute for conventional switches and switchboards. Except, they are operated by touch. In addition, they come in sleek and stylish designs that can take the luxury quotient of your home a notch higher.

LIGHTING AUTOMATION
Lighting automation systems are a growing area of interest for homeowners because they have a measurable impact on quality of life and home value. Lighting automation provides programmable ON/OFF and Dimming control which transforms user's home and its lifestyle with a new comfort.

Features
● Capacitive Fan Dimming Technology - No more fan noise at low speed!
● Feather Touch Buttons
● Hybrid Module
● Retrofit
● Any non metallic surface can be made switch
● Touch,Bluetooth & Wifi Control
● Smartphone / Tablet controlled

## Unique Selling Points (USPs) of Project:

The following are the USPs of the project:

- **Retrofit:** The system will be installed in pre-existing boards in houses, and require no extra wiring for it to be installed.

- **State of Art:** With this system any Non-Metallic surface can be turned into a touch based switch. In this project we turned Sunmica into a touched based switch.

- **Affordability:** Efficient, effective and affordable home automation system as compared to other home automation systems available in the market.

- **Versatality:** Versatile in its operation like giving multiple features such as manual touch based switching, Switching over internet, Switching over Bluetooth, without internet using voice commands.

# COMPONENTS

## 1. RELAY MODULE:


*figure 2.1 : 8 channel Relay Module*

Relays are most commonly used **switching devices** used in electronics. It can be used to switch high current loads easily unlike transistors which are limited by the maximum current that can flow through them and also can't switch AC loads. The Relay Module is an Electromagnetic switch, when the coil inside is energized with a small current, it can switch ON or OFF the high current circuit. It has PCB screw terminals to directly connect. They can be used in Home automation to switch ON or OFF the appliances, in Electronic circuits to perform switching operations, in safety circuits to disconnect or connect the **heavy loads** in case of any dangerous situation, in Automobile applications like turning on windscreen wipers, power windows fuel pump, cooling fan etc.

The input port description of 8 channel Relay Module is as follows:

- VCC: Connected to positive supply voltage (supply power according to relay voltage)
- GND: Connected to negative supply voltage
- IN1: Signal(LOW Level) triggering terminal 1 of relay module
- IN2: Signal(LOW Level) triggering terminal 2 of relay module
- IN3: Signal(LOW Level) triggering terminal 3 of relay module
- IN4: Signal(LOW Level) triggering terminal 4 of relay module
- IN5: Signal(LOW Level) triggering terminal 5 of relay module
- IN6: Signal(LOW Level) triggering terminal 6 of relay module
- IN7: Signal(LOW Level) triggering terminal 7 of relay module
- IN8: Signal(LOW Level) triggering terminal 8 of relay module

INx stands for the contacts including IN1, IN2, IN3 and other pins in the following part.

Triggered at high level means a forward voltage exists between signal triggering terminal (INx) and negative supply voltage. It generally connects positive supply voltage and triggering terminal together. When triggering terminal has positive supply voltage or reaches triggering voltage, the relay will pull in.

While triggered at low level means the voltage between the signal triggering terminal (INx) and negative supply voltage is 0V, or the voltage at the triggering terminal is lower than positive supply voltage. If it lowers to a triggering voltage, the relay will pull in. It generally connects negative supply voltage and triggering terminal together.

Output port description:

Each submodular of the relay has one NC(nomalclose), one NO(nomalopen) and one COM(Common). So there are 8 NC, 8NO and 8 COM of the channel relay in total. NC stands for the normal close port contact and the state without power; No stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

**2. ESP 32:**

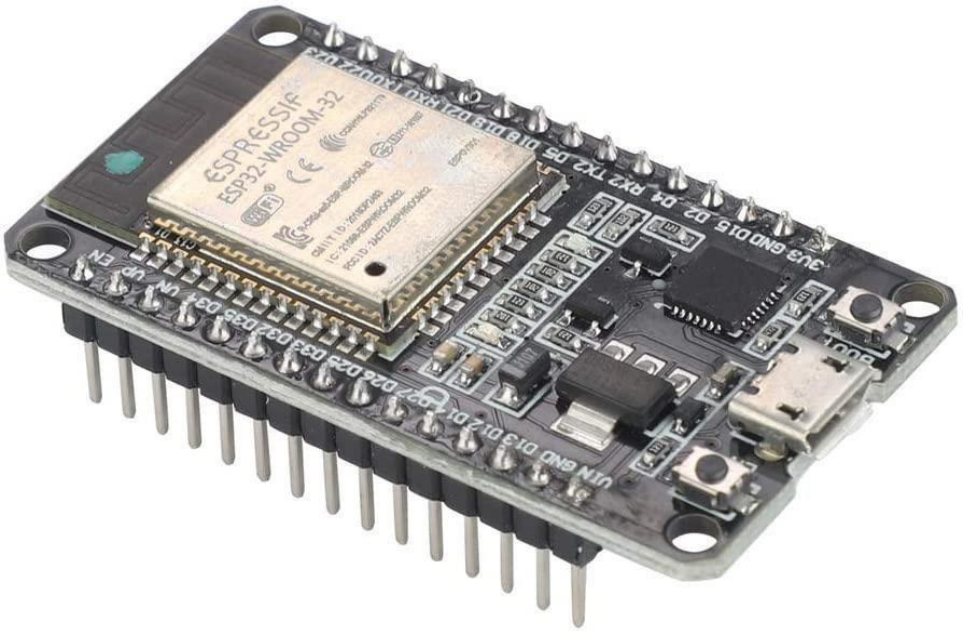



*Figure 2.2: ESP32*

ESP32 is a low-cost System on Chip (SoC) Microcontroller from Espressif Systems, the developers of the famous ESP8266 SoC. It is a successor to ESP8266 SoC and comes in both single-core and dual-core variations of the Tensilica's 32-bit Xtensa LX6 Microprocessor with integrated Wi-Fi and Bluetooth. The good thing about ESP32, like ESP8266 is its integrated RF components like Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filters and RF Balun. This makes designing hardware around ESP32 very easy as you require very few external components.

**Specifications of ESP32**

ESP32 has a lot more features than ESP8266 and it is difficult to include all the specifications here. So, we have made a list of some of the important specifications of ESP32 here.

- Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz.
- 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.
- Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.
- Support for both Classic Bluetooth v4.2 and BLE specifications.
- 34 Programmable GPIOs.
- Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity include 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
- Ethernet MAC for physical LAN Communication (requires external PHY).
- 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.
- Motor PWM and up to 16-channels of LED PWM.
- Secure Boot and Flash Encryption.

Different Ways to Program :

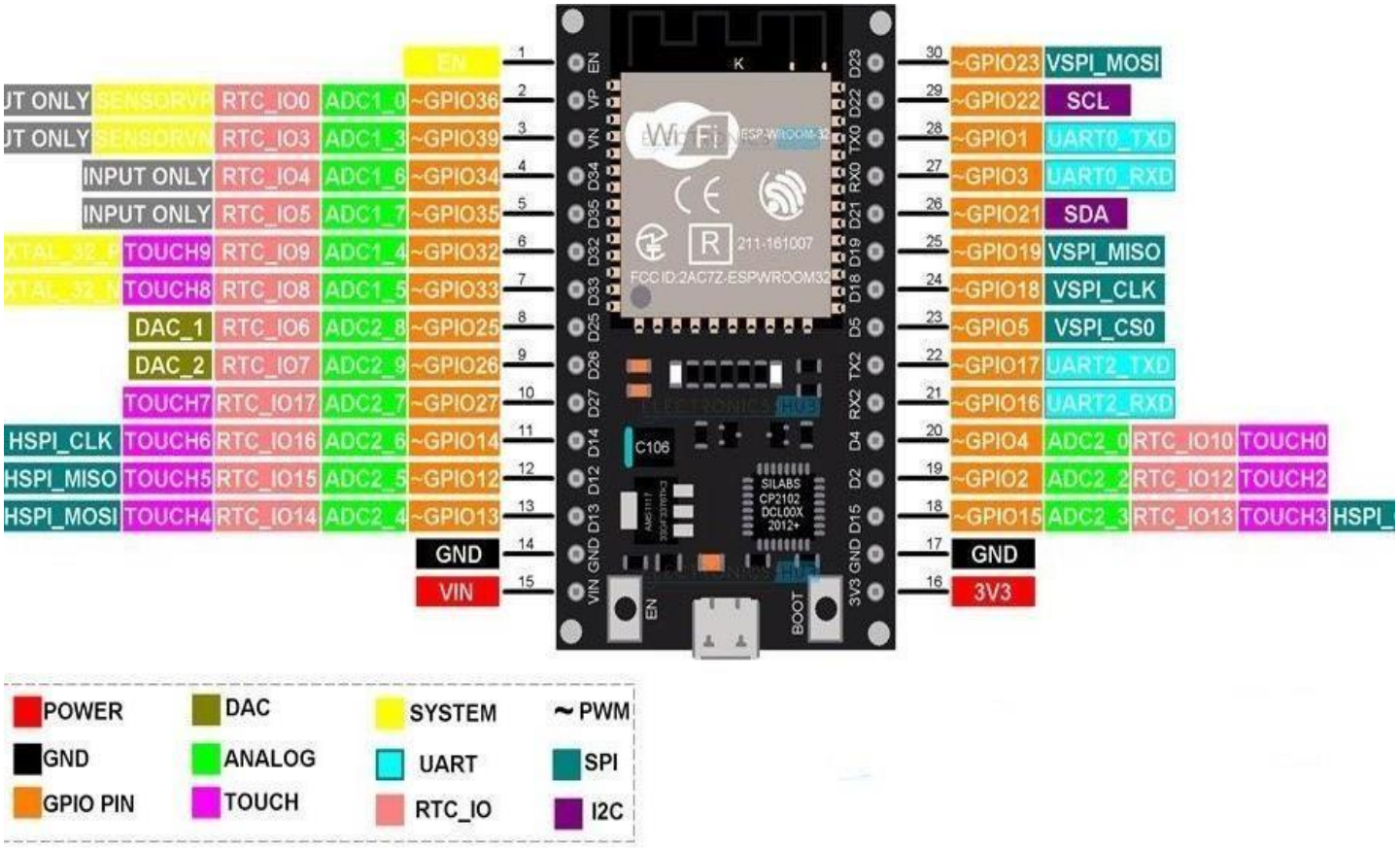
A good hardware like ESP32 will be more user friendly if it can be programmed (writing code) in more than one way. And not surprisingly, the ESP32 supports multiple programming environments. Some of the commonly used programming environments are:

1. Arduino IDE
2. PlatformIO IDE (VS Code)
3. LUA
4. MicroPython
5. Espressif IDF (IoT Development Framework)
6. JavaScript

As Arduino IDE is already a familiar environment, we have used the same to program ESP32 in our project.

**Pinout of ESP32 Board:**

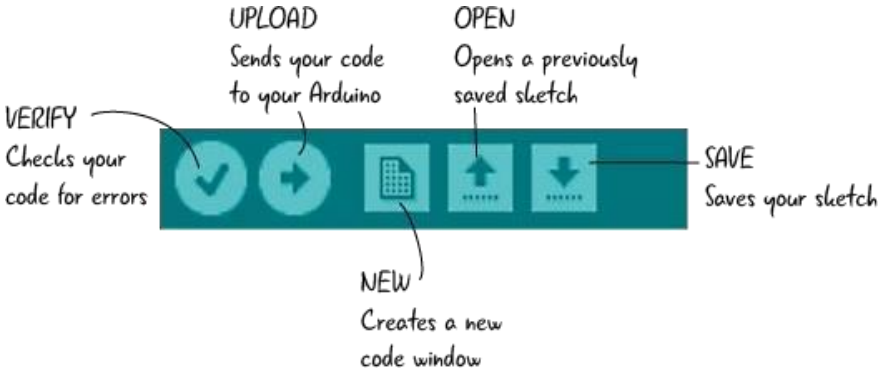
*Figure 2.3 : ESP32 Pin diagram*



**3. ARDUINO IDE:**

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'



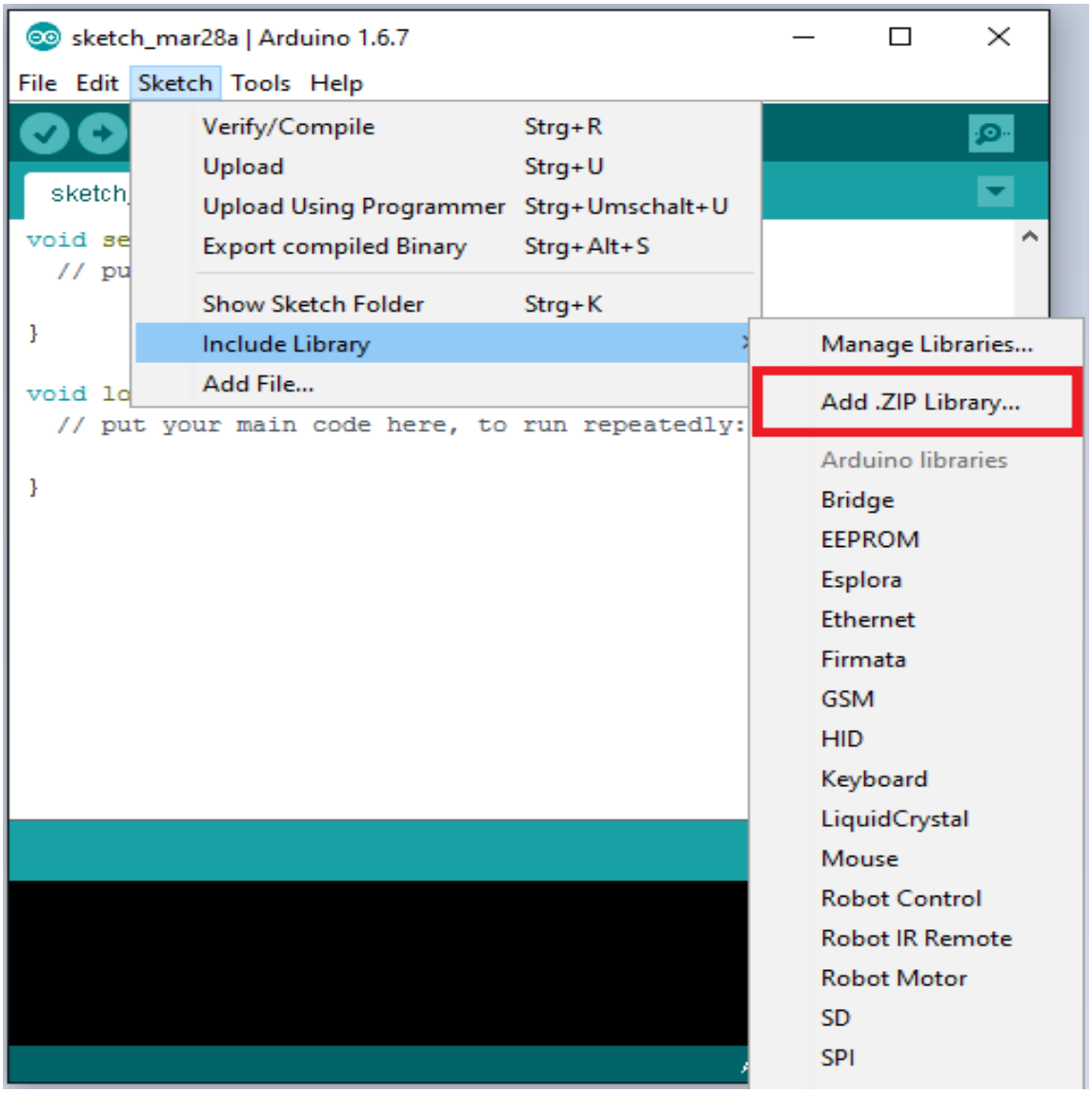



*Figure 2.4 : Including Libraries in Arduino IDE*

Libraries are files written in C or C++ (.c, .cpp) which provide your sketches with extra functionality (e.g. the ability to control an LED matrix, or read an encoder, etc.). They were introduced in Arduino 0004.

To use an existing library in a sketch simply go to the Sketch menu, choose "Import Library", and pick from the libraries available. This will insert an #include statement at the top of the sketch for each header (.h) file in the library's folder. These statements make the public functions and constants defined by the library available to your sketch. They also signal the Arduino environment to link that library's code with your sketch when it is compiled or uploaded.

To install your own library, create a folder inside ARDUINO/hardware/libraries with the name of your library. The folder should contain a C or C++ file with your code and a header file with your function and variable declarations. It will then appear in the Sketch | Import Library menu in the Arduino IDE.

Because libraries are uploaded to the board with your sketch, they increase the amount of space used by the ATmega8 on the board.

If a sketch no longer needs a library, simply delete its #include statements from the top of your code. This will stop the Arduino IDE from linking the library with your sketch and decrease the amount of space used on the Arduino board.

**4. TOUCH SENSOR** :

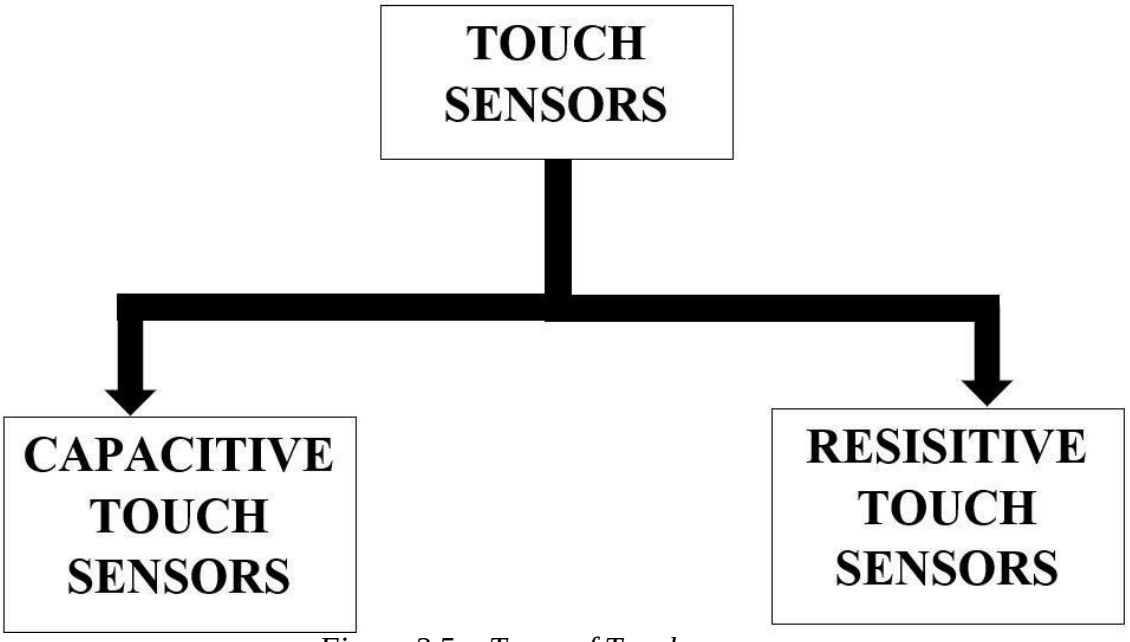



*Figure 2.5 : Types of Touch sensors*

Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc… Touch sensors offer an intuitive user interface Touch sensors are also known as Tactile sensors. These are simple to design, low cost and are produced in large scale. With the advance in technology, these sensors are rapidly replacing the mechanical switches. Based on their functions there are two types of touch sensors- Capacitive sensor and Resistive sensor

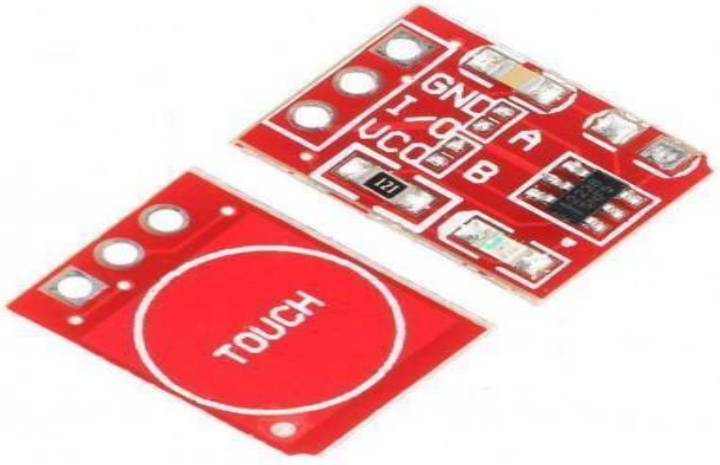



*Figure 2.6 : TTP223 Capacitive Touch Sensor*

Capacitive sensors work by measuring capacitance and are seen in portable devices. These aredurable, robust and attractive with low cost. Resistive sensors don’t depend on any electrical properties for operation. These sensors work by measuring the pressure applied to their surface. Capactive touch sensor working:Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they get activated and acts as a closed switch. When the pressure or contact is removed they act as an open switch. Capacitive touch sensor containstwo parallel conductors with an insulator between them. These conductors plates act as a capacitor with a capacitance value C0.

When these conductor plates come in contact with our fingers, our finger acts as a conductiveobject. Due to this, there will be an uncertain increase in the capacitance. A capacitance measuring circuit continuously measures the capacitance C0 of the sensor. When this circuit detects a change in capacitance it generates a signal.

## 5. BLUETOOTH APP

**We have used the Android App available on Google play store**

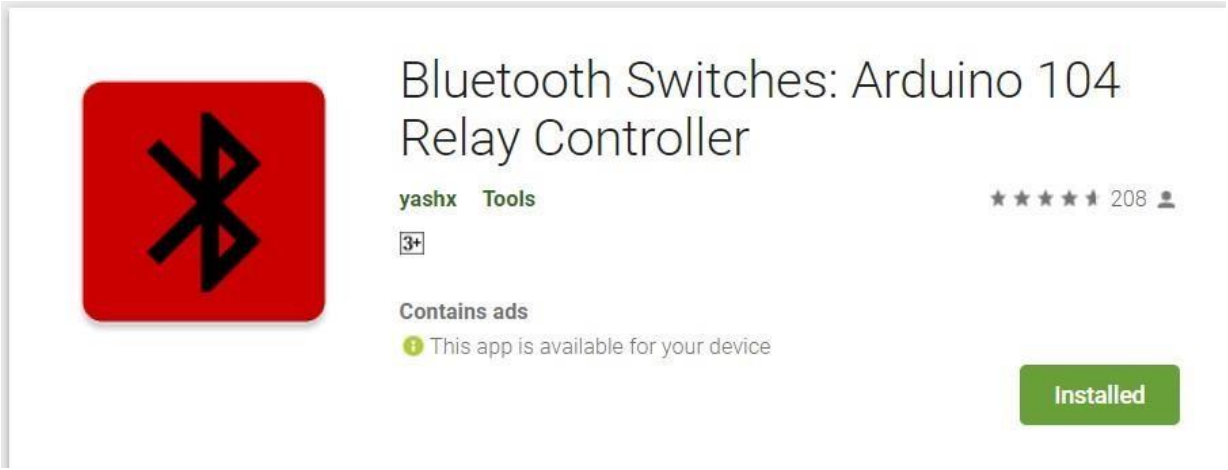



*Figure 2.7 : Bluetooth third party App*

**Link for above app :**

**https://play.google.com/store/apps/details?id=com.github.yashx.arduinobluetoothswitcheshc_05andhc_06**

**The app has been built using MIT APP INVENTOR for sending Data Serially over Bluetooth.**
**Features:**
- You can choose the number of buttons you want to use. (Up to 104 buttons for now)
- Supports both push buttons and normal switches.
- Supports basic voice control
- You can change values sent by the buttons
- You can control leds, relay or start your own home automation project.
- You can set the app to auto connect to your devices (like HC 05 or HC 06) for more convenience
- You can change name of any switch by long clicking it
- You can change the layout

This app uses serial communication over Bluetooth.
The app will send a unique character whenever a switch is turned on or turned off.
The app has enough buttons to satisfy any of the customer needs. With the freedom to choose how many buttons are required, we can have a UI tailored as per customer needs.It Use the built-in auto connect feature to quickly get up and running.

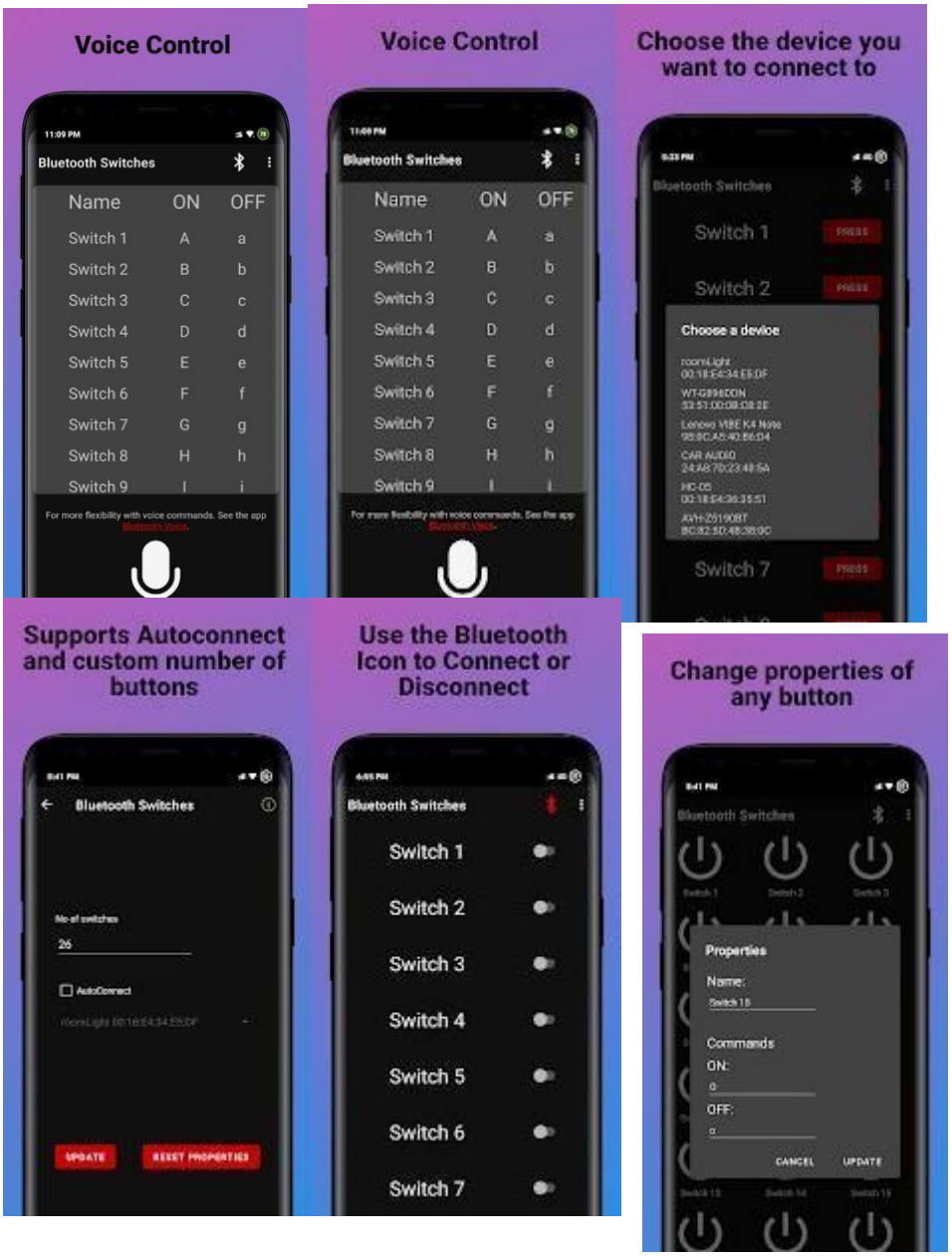



*Figure 2.8: Bluetooth App interface*

**6. Wifi Protocol:**

WiFi stands for Wireless Fidelity. WiFiIt is based on the IEEE 802.11 family of standards and is primarily a local area networking (LAN) technology designed to provide in-building broadband coverage.

WiFi is a universal wireless networking technology that utilizes radio frequencies to transfer data. WiFi allows high-speed Internet connections without the use of cables.

The term WiFi is a contraction of "wireless fidelity" and commonly used to refer to wireless networking technology.

WiFi is a freedom – freedom from wires. It allows you to connect to the Internet from just about anywhere — a coffee shop, a hotel room, or a conference room at work. What's more – it is almost 10 times faster than a regular dial-up connection. WiFi networks operate in the unlicensed 2.4 radio bands, with an 11 Mbps (802.11b) or 54 Mbps (802.11a) data rate, respectively. WiFi has become the de facto standard for last mile broadband connectivity in homes, offices, and public hotspot locations. Systems can typically provide a coverage range of only about 1,000 feet from the access point.

ESP32 uses Full 802.11 b/g/n/e/i WLAN Protocol. The description of these different kinds of 802.11 protocols are as follows:

- 802.11 b: 802.11b uses DSSS (Direct-Sequence Spread Spectrum) - a modulation method used to reduce signal interference - in the 2.4 GHz band, allowing it to have speeds up to 11 Mbps. The 2.4 band does a good job at penetrating **obstacles** to provide more WiFi coverage. Unfortunately, the data travels at a much slower rate, especially when it's coupled with network interferences caused by devices operating on the same frequency, such as baby monitors, microwave ovens, cordless phones, appliances, and Bluetooth devices.

- 802.11 g: To fulfill a growing demand for faster internet under the 2.4 GHz band, 802.11g joined the 802.11 family in 2003.The developers took the best qualities of 802.11a and 802.11b to create the 802.11g standard. It supports a networking bandwidth up to 54 Mbps and operates under the 2.4 GHz band.

- 802.11 n: Wireless-N was developed in 2009 to improve speeds, reliability, and extend the range of wireless transmissions.It was the first standard to use MIMO (Multiple-Input Multiple-Output) technology. MIMO products use a series of antennas to receive more data from one device at a time, which results in faster data transmissions. In addition, it was the first to allow the usage of two radio frequencies – 2.4 GHz and 5 GHz. The use of both frequencies makes the 802.11n standard compatible with 802.11a/b/g devices.

- 802.11 e: 802.11e is an amendment to 802.11 standards, which defines a set of quality of service (QoS) for wireless LAN applications through alterations to the media access control layer. It also provides essential services for delay sensitive applications, such as streaming video and Voice over Internet Protocol (VoIP). It corresponds to the Wi-Fi multimedia certification of the Wi-Fi Alliance.

- 802.11e offers subscribers high speed internet access with VoIP, full motion video and high quality audio. Networks using 802.11e operate at radio frequencies ranging between 2.4 GHz and 2.4835 GHz or between 5.75 GHz and 5.850 GHz. This higher frequency ranges account for advantages such as more channels, fast data transfer speeds and less chances of interference.

- 802.11 i: 802.11i is a standard for wireless local area networks (WLANs) that provides improved encryption for networks that use the popular 802.11a, 802.11b(which inclues Wi-Fi) and 802.11g standards.

| Feature | WiFi (802.11b) | WiFi (802.11a/g) |
|---|---|---|
| **PrimaryApplication** | Wireless LAN | Wireless LAN |
| **Frequency Band** | 2.4 GHz ISM | 2.4 GHz ISM (g)<br>5 GHz U-NII (a) |
| **Channel Bandwidth** | 25 MHz | 20 MHz |
| **Half/Full Duplex** | Half | Half |
| **Radio Technology** | Direct Sequence<br>Spread Spectrum | OFDM<br>(64-channels) |
| **Bandwidth** | <=0.44 bps/Hz | ≤=2.7 bps/Hz |
| **Efficiency** | | |
| **Modulation** | QPSK | BPSK, QPSK, 16-, 64-QAM |
| **FEC** | None | Convolutional Code |
| **Encryption** | Optional- RC4m (AES in 802.11i) | Optional- RC4(AES in 802.11i) |
| **Mobility** | In development | In development |
| **Mesh** | Vendor Proprietary | Vendor Proprietary |
| **Access Protocol** | CSMA/CA | CSMA/CA |

*Table 2.1 : Technical Comparison Between Wi-Fi standards*

**7. Blynk App:**


*Figure 2.9 : Blynk App*

Blynk is an Internet of things (IoT) company which provides a platform for building mobile (IOS and Android) applications that can connect electronic devices to the Internet and remotely monitor and control these devices.

There are three major components in the platform:

- **Blynk App** - It allows to us create amazing interfaces for our projects using various widgets provided.
- **Blynk Server** -It responsible for all the communications between the smartphone and hardware. We can use Blynk Cloud or run private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Features of Blynk are as follows:

- Similar API & UI for all supported hardware & devices
- Connection to the cloud using:
    - WiFi
    - Bluetooth and BLE

- Ethernet
- USB (Serial)
- GSM

- Set of easy-to-use Widgets
- Direct pin manipulation with no code writing
- Easy to integrate and add new functionality using virtual pins
- History data monitoring via SuperChart widget
- Device-to-Device communication using Bridge Widget
- Sending emails, tweets, push notifications, etc.

**Blynk works over the Internet.** This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, Raspberri Pi with WiFi dongle, Particle Photon or SparkFun Blynk Board.

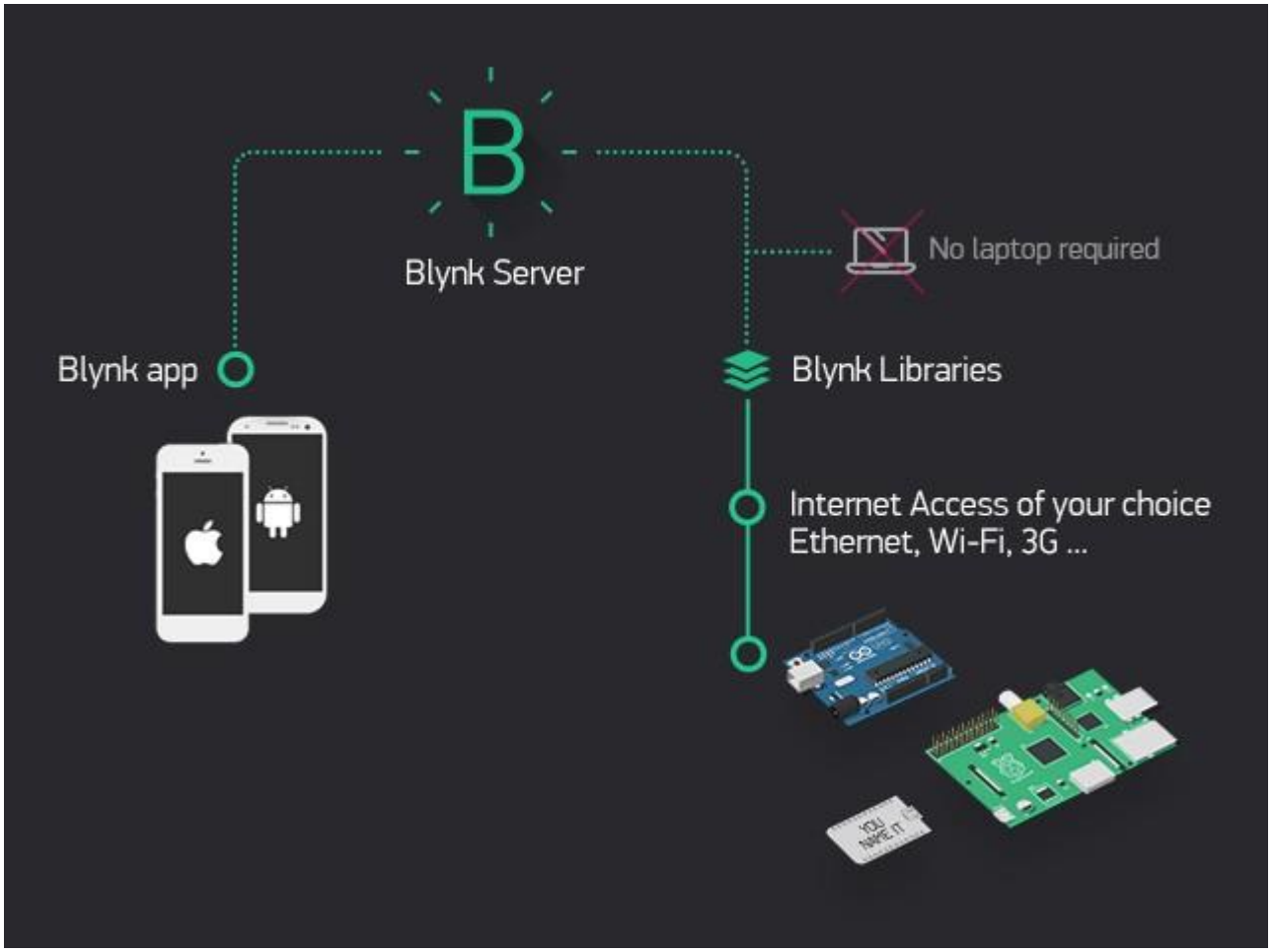



*Figure 2.10 : Blynk App Operation*

**8. Bluetooth Protocol:**

*Figure 2.11 : Bluetooth*

Bluetooth is a standardized protocol for sending and receiving data via a 2.4GHz wireless link. It's a secure protocol, and it's perfect for short-range, low-power, low-cost, wireless transmissions between electronic devices. Bluetooth embedded into a great variety of consumer products, like headsets, video gamecontrollers, or livestock trackers. Bluetooth serves as an excellent protocol for wirelessly transmitting relatively small amounts of data over a short range (<100m). It's perfectly suited as a wireless replacement for serial communication interfaces. The speed of the gross data rate is 1bit/s, and the speed of the second generation is increased up to 2bit/s. One-to-one Bluetooth connections are allowed for a maximum speed of data transfer, which is 723 kbit/s. The standby mode is only 0.3 ma and it has low power consumption.

The different versions of Bluetooth are as follows:

| Bluetooth versions | specification |
|---|---|
| Bluetooth v1.0 to v1.08 | Mandatory Bluetooth hardware device and address |
| Bluetooth v1.1 | IEEE standard 802.15.1-2002 |
| Bluetooth v1.2 | Faster connection |
| Bluetooth v2.0+EDR | Enhanced data rate |
| Bluetooth v2.1 | Secure simple pairing |
| Bluetooth v3.0 | High-speed data transfer |
| Bluetooth v4.0 | Low energy consumption; recently in use in apple i-phone 4s |

*Table 2.2 : Different versions of Bluetooth and their specifications*

Some of the characteristics of Bluetooth protocols are given below:

- Up to eight devices can be networked in the Piconet by using Bluetooth.
- Devices need not to be pointed at each other, as signals are Omnidirectional.
- Governments regulated worldwide because it is possible to utilize the same standard.
- Signals can be transmitted even through walls and briefcases.

# CHAPTER 3

# WORKING PRINCIPLE

In our project we have used ESP32 microcontroller to give signals for switching, 8 channel Relay module for carrying out switching. TTP223 capacitive touch sensors working as push buttons and also for turning any non-metallic surface as switch using it. We have also used Blynk App and Bluetooth third-party app for controlling switching of appliances over the internet and without the internet respectively. There is also a capacitive fan dimmer circuit installed for controlling fan speeds without humming.

After uploading code in ESP32 microcontroller, it sends signals to Relay module to do switching via touch sensors, Blynk app over Wi-Fi, Bluetooth app to operate without internet etc.The Relay Module have 8 input pins to activate each relay and are connected to GPIO pins of ESP32 according to the circuit diagram. Capacitive touch sensors are also connected to GPIO pins of ESP32 as per circuit diagram.

For manual switching using touch, sensors senses the touch and will operate switching in relay module through ESP32. They are also attached with Sunmica, a non metallic surface which replaces traditional switch buttons, gives it an aesthetic look , turn a non metallic surface into a switch.

For switching appliances over the internet, we used Blynk app to configure Blynk app with ESP32 which have in-built Wi-Fi module. We have to enter unique authentication code given by Blynk, Wi-Fi name (SSID) and password inside the code and after uploading the code in ESP32, its connection is established with Wi-Fi. Virtual Buttons are created in the app to switch ON/OFF the appliances. Also, even if we do switching using any means other than Blynk app, the ON/OFF status of switches are updated in real time in the Blynk app.

By using Bluetooth third party app, voice commands can be given to do switching of loads.

ESP32 has an in-built Bluetooth module which can be configured with the Bluetooth app and

voice commands can be given to turn load ON/OFF. For example, Saying “Turn light ON” will give signal to the ESP32 and which will make relay module to do switching and turn the light ON. There is another additional feature of switching using virtual buttons in the Bluetooth app. Turning ON/OFF virtual buttons within the app will do switching of appliances. This is helpful when there are some issues with Wi-Fi connection, so that we can control appliances using Bluetooth, even without internet.

Capacitive Fan dimmer circuit is installed in the system to control the speed of fan. This project allows changing speed of fan using touch based switching, Bluetooth and Wi-Fi. We have three levels of speeds in this project.

## Block Diagram

The block diagram of our system is given below.

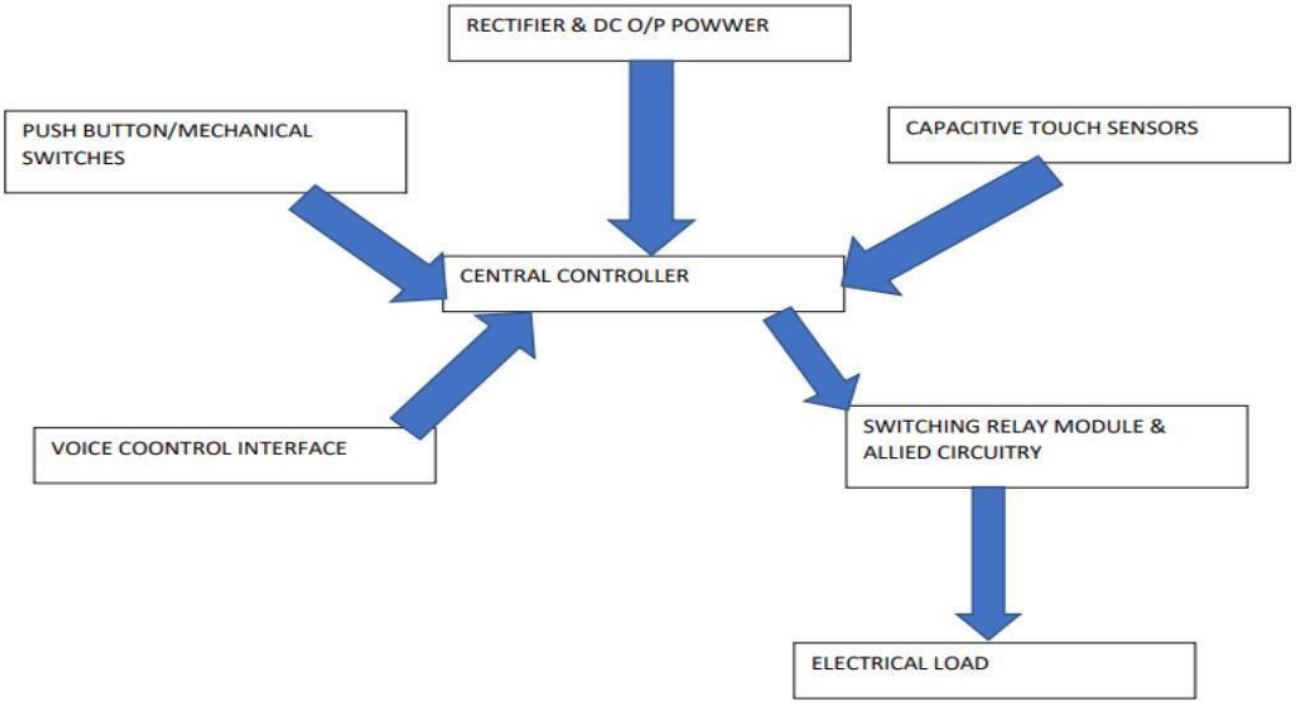



*Figure 3.1: Block Diagram*

**Circuit Diagram:**

The following figure represents the circuit diagram of our system displaying the circuit connections of each component in our system.

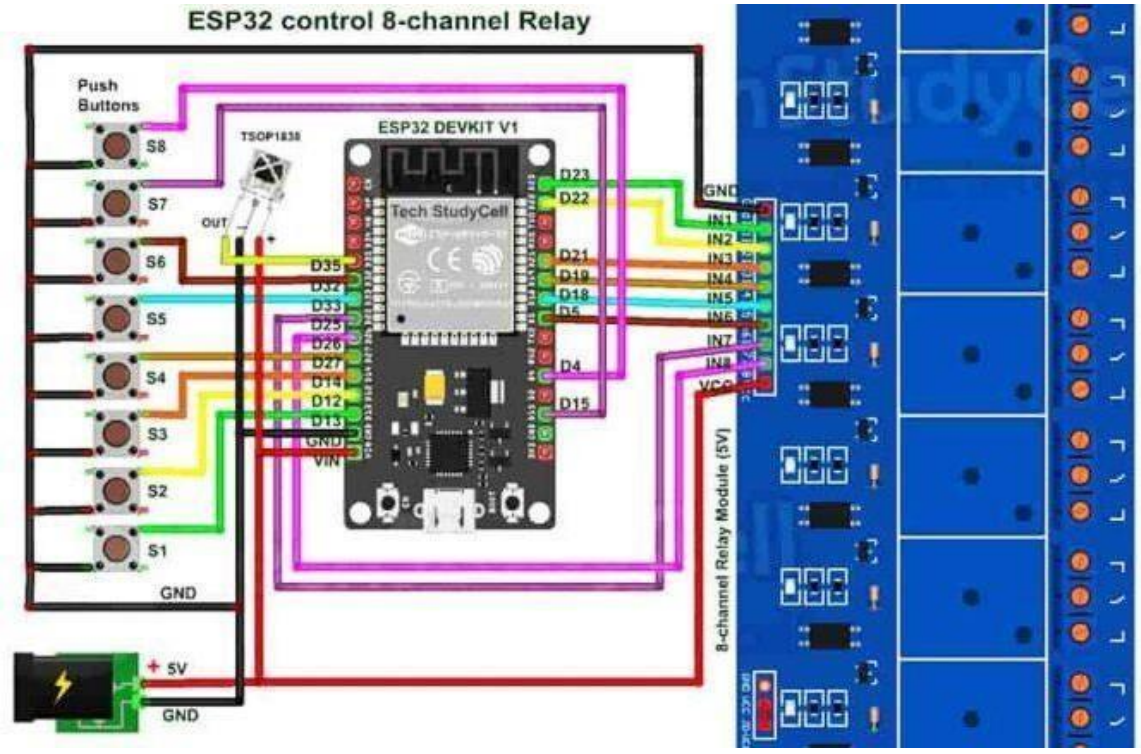



*Figure 3.2: Circuit Diagram*

**Fan Dimmer Circuit:**

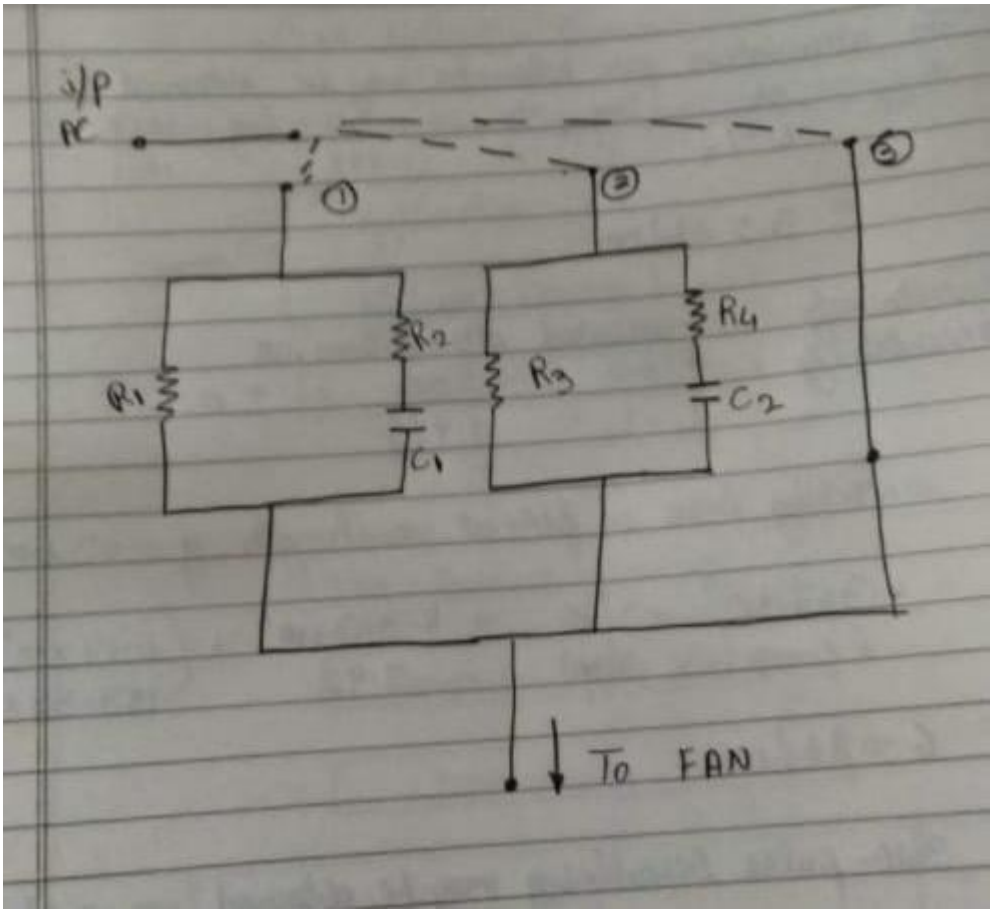



*Figure 3.3 : Fan dimmer circuit Diagram*

In the circuit we have two RC circuits. The values of resistors and capacitors in the circuit are as follows:

- R1 = 220kΩ , ¼ W
- R3 = 220kΩ , ¼ W
- R2 = 2.2Ω , ½ W
- R4 = 2.2Ω , ½ W
- C1 = 2.2 uF, 250V
- C2 = 3.3 uF, 250V

These RC circuits will limit the current flowing through it. The common terminal is connected to knob 1, the current flows through R1, R2 and C1. Here the reactance is more, so less current flows through it which will result in fan rotating with less intensity or slow speed. The common terminal is connected to knob 2, the current flows through R3, R4 and C2. Here the reactance is less, so more current flows through it which will result in fan rotating with more intensity or higher speed. When common terminal is connected to knob 3, it is short and the current flowing through it will be maximum, which will result in fan rotating in its highest speed.

**OUTPUT:**

*Figure 3.4 : SMART hub for domestic appliance switching*

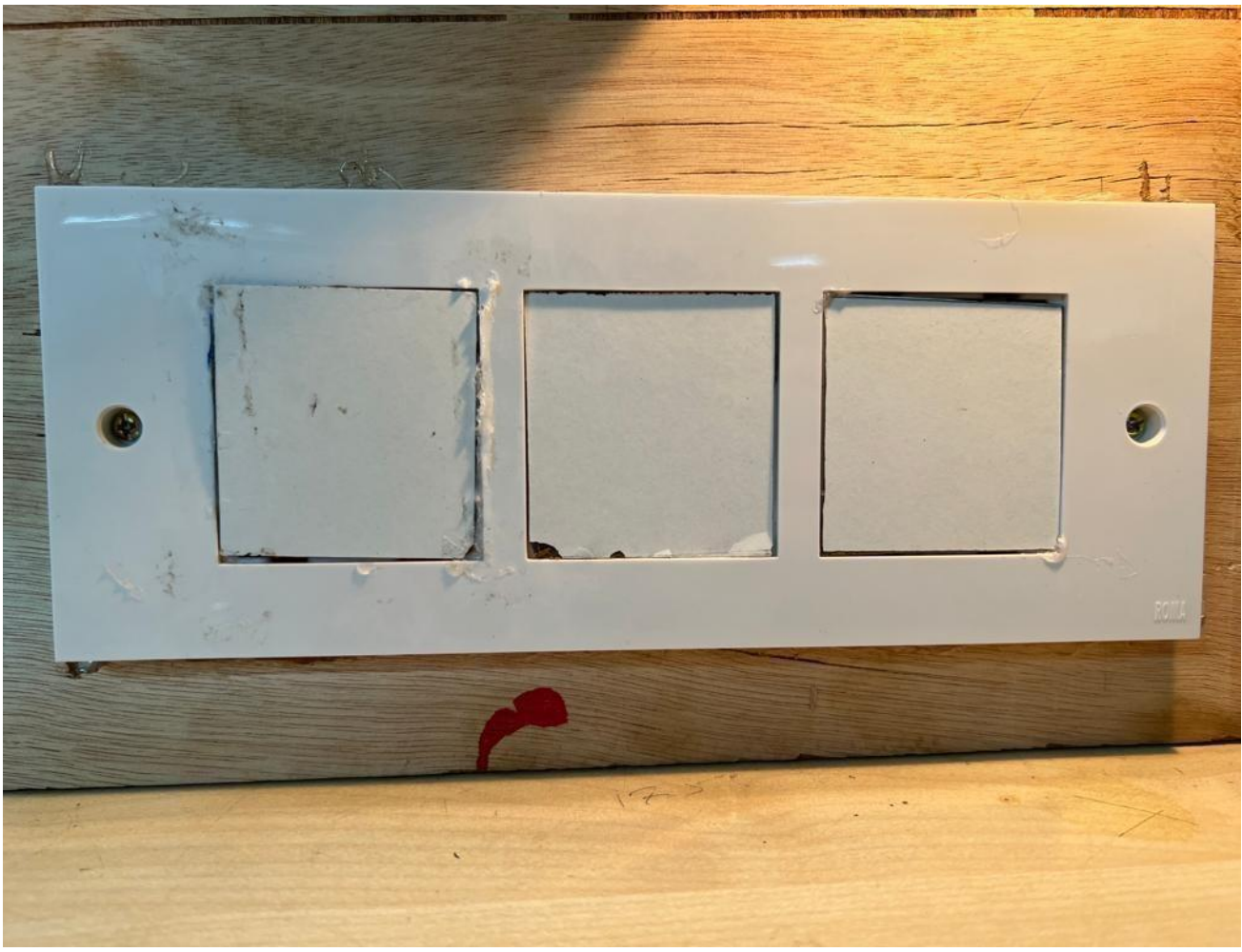


*Figure 3.5 : Touch based switch board*

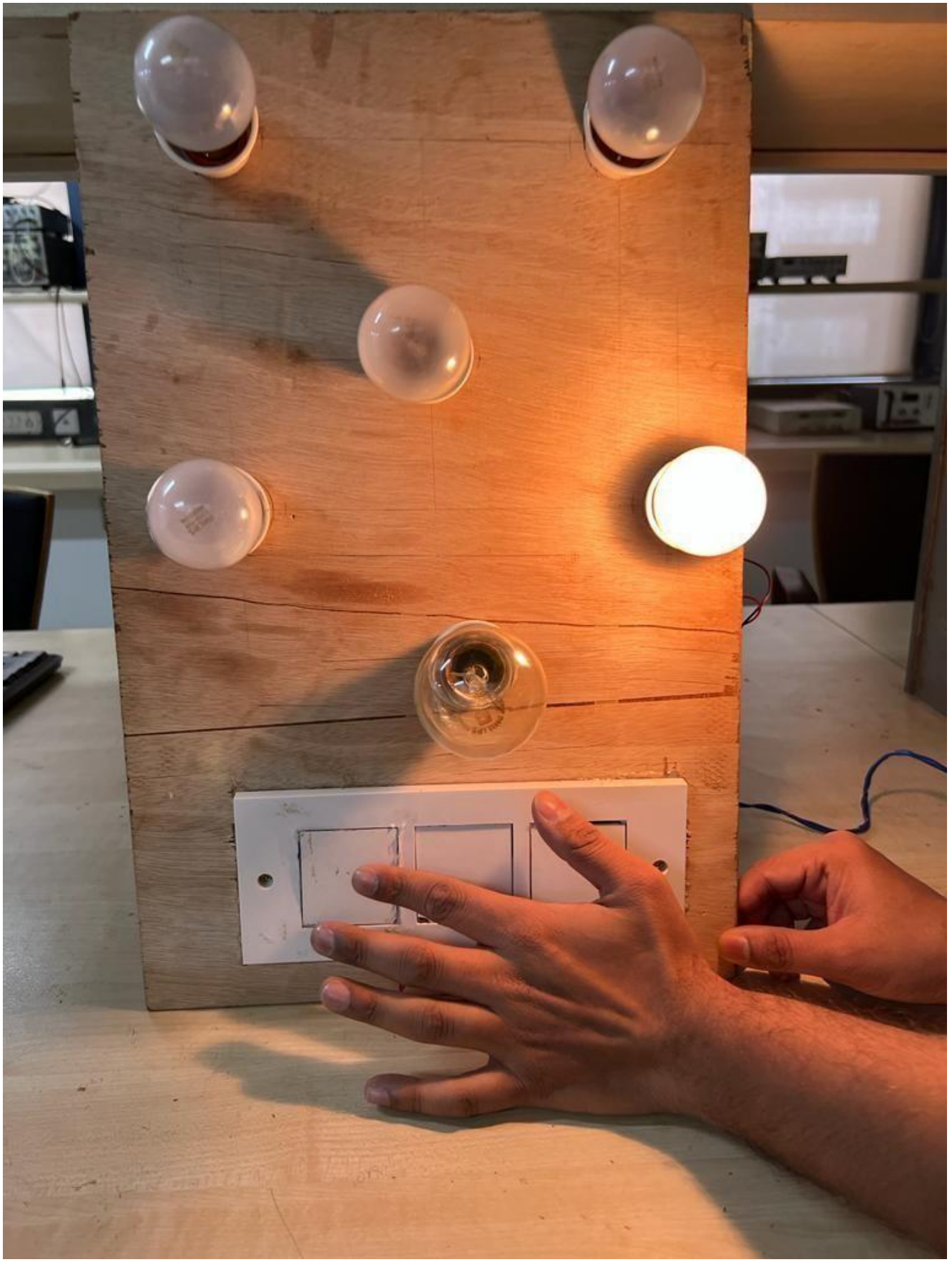


*Figure 3.6 : Switching of load by touch*

*Figure 3.7 : Switching of load using Blynk app over the internet*

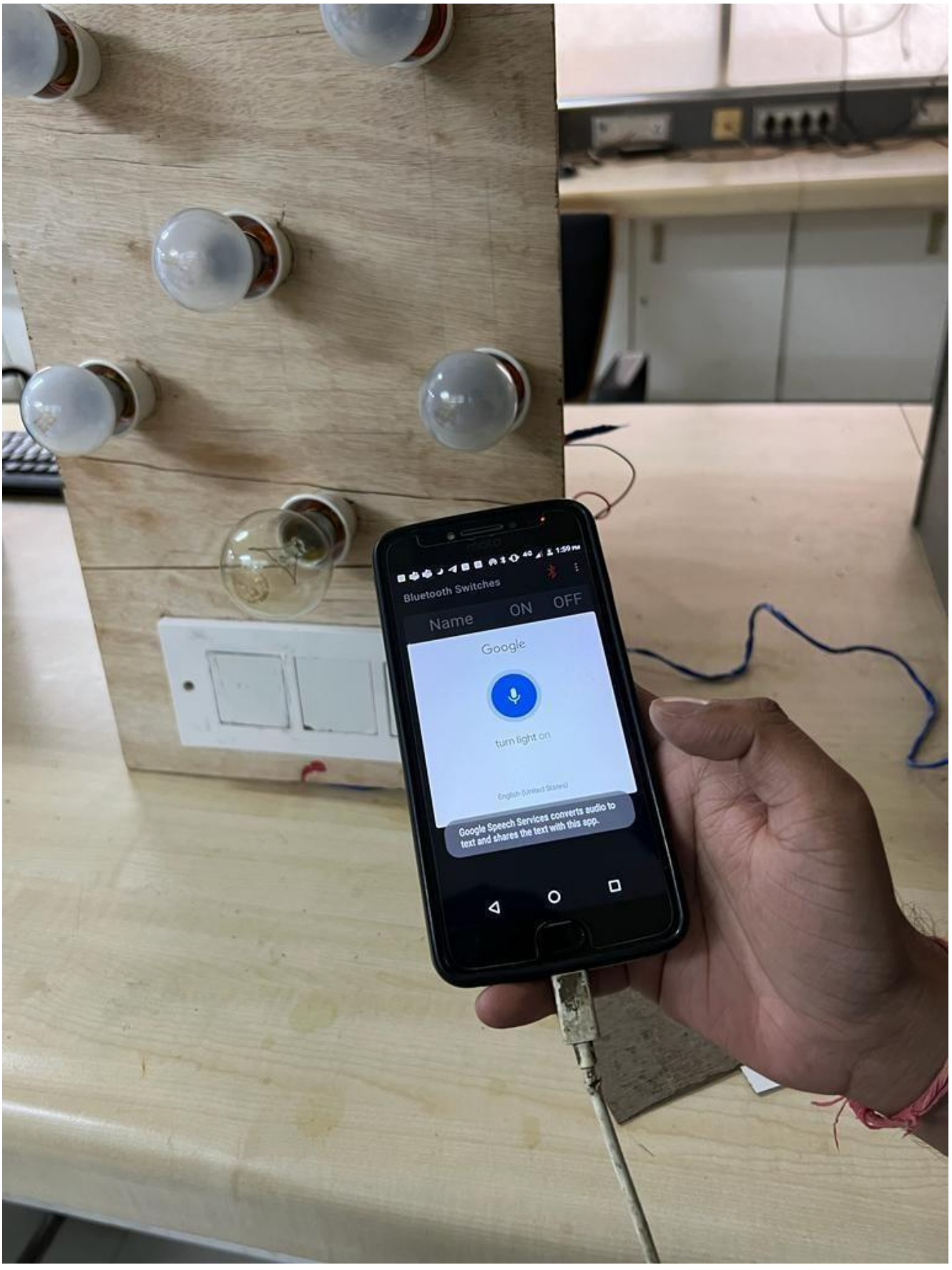



*Figure 3.8 : Switching of load by Voice command in Bluetooth app*

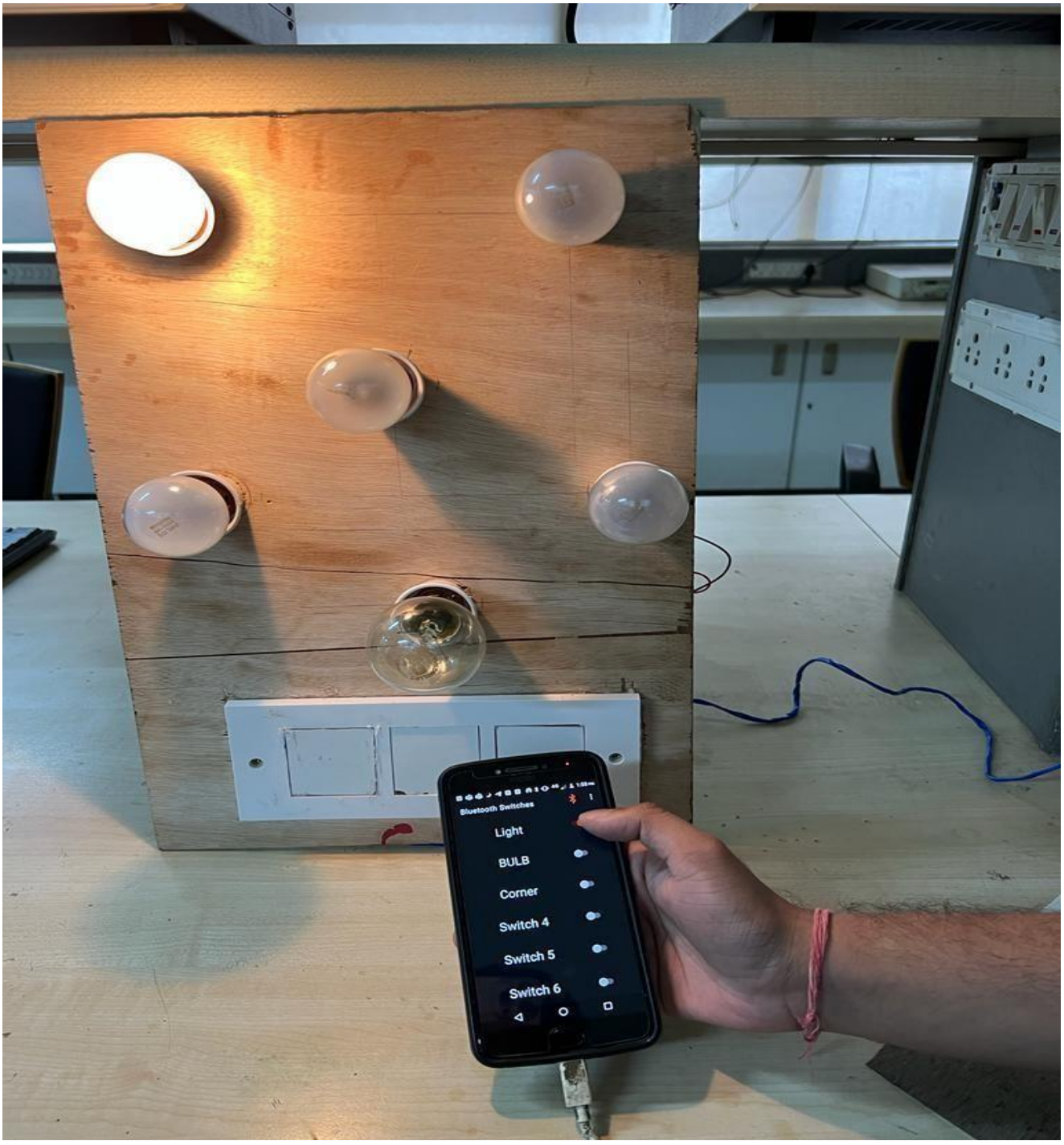



*Figure 3.9 : Switching of load by virtual switches in Bluetooth App*

# CHAPTER 4

# CONCLUSION

We have been successful in building our SMART HUB for domestic appliance switching. We have achieved the objectives of our project.

With this system we are able to turn a non-metallic surface (Sunmica) into a touch based switch board using capacitive touch sensors providing sparkless touch based switching. We are also able to control appliances using a Bluetooth app, configuring it with ESP32 microcontroller and switching using Voice commands and virtual switches without internet. With internet, we are able to control switching of appliances using virtual switches in Blynk App. We successfully build our SMART Hub for domestic appliance switching making it Affordable, Retrofit, versatile without requiring any extra wirings.

# APPLICATIONS

The system also comprises of other features like voice control, and have sensors integrated into the board to complement our system.
We would be able to carry out the basic functions of the system using Mobile application.

It will enable us to control switches by sending signals using Bluetooth module and also through voice control which provides convenience to control switches and appliances without need to manually turn ON/OFF switches.

Our system will provide sparkless switching and will be seamless in working , cost efficient, technologically advanced, Retrofit.

Key applications of our system are:

- Capacitive touch switching
- Swtiching in sync with mechanical switches
- Voice control
- Remote Operation
- Sparkless Switching

# FUTURE SCOPE

With the continuous advances in home automation technology this field is poised to grow with leaps and bounds and as we bring in more and more Connected devices together the scope and reach of home automation systems will continue to grow.

As far as our project is concerned we can propose a few clearly foreseeable upgrades for the same,Namely:

- Interfacing with available voice assistants like Amazon Alexa, Google Home kit, Apple's Hey Siri etc.
- More sensors can be incorporated so as to increase the reach of the system such as temperature sensor, Day/night Sensor, Motion(person) Sensor,Heat Sensor etc.
- Scene configuration, ie COnfiguring status of various switches into a single switch blended with conditions on time,season or mood.
- Switch Application through QR CODE or AUGMENTED REALITY
- Use of Long Distance communication alternatives over the one incorporated such as LORAWan etc.
- Surface materials for the switches can be changed to bring out more aesthetically appealing switch.

# APPENDIX

## CODE:

```
#include <BlynkSimpleEsp32.h>
#include "BluetoothSerial.h"


#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run
`make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

BlynkTimer timer;

// define the GPIO connected with Relays and switches
#define RelayPin1 23  //D23
#define RelayPin2 22 //D22
#define RelayPin3 21 //D21
#define RelayPin4 19 //D19
#define RelayPin5 18 //D18
#define RelayPin6 5 //D5
#define RelayPin7 25 //D25
#define RelayPin8 26  //D26

#define  SwitchPin1  13  //D13
#define  SwitchPin2  12  //D12
#define  SwitchPin3  14  //D14
#define  SwitchPin4  27  //D27
#define  SwitchPin5  33  //D33
#define  SwitchPin6  32  //D32
#define  SwitchPin7  15  //D15
#define SwitchPin8 4  //D4

#define wifiLed 2 //D2

#define VPIN_BUTTON_1    V1
#define VPIN_BUTTON_2    V2
#define VPIN_BUTTON_3    V3
#define VPIN_BUTTON_4    V4
#define VPIN_BUTTON_5    V5
#define VPIN_BUTTON_6    V6
#define VPIN_BUTTON_7    V7
#define VPIN_BUTTON_8    V8

int toggleState_1 = 1; //Define integer to remember the toggle state for relay 1
int toggleState_2 = 1; //Define integer to remember the toggle state for relay 2
int toggleState_3 = 1; //Define integer to remember the toggle state for relay 3
int toggleState_4 = 1; //Define integer to remember the toggle state for relay 4
int toggleState_5 = 1; //Define integer to remember the toggle state for relay 5
int toggleState_6 = 1; //Define integer to remember the toggle state for relay 6
int toggleState_7 = 1; //Define integer to remember the toggle state for relay 7
int toggleState_8 = 1; //Define integer to remember the toggle state for relay 8
```

```
int wifiFlag = 0;
char bt_data; // variable for storing bluetooth data

#define AUTH "PUT BLYNK  TOKEN HERE"                    // You should get Auth Token in the Blynk App.
#define WIFI_SSID "PUT WIFI NAME HER" //Enter Wifi Name
#define WIFI_PASS "PUT PASS WORD OF WIFI HERE"                    //Enter wifi Password

// When App button is pushed - switch the state

BLYNK_WRITE(VPIN_BUTTON_1) {
  toggleState_1 = param.asInt();
  digitalWrite(RelayPin1, toggleState_1);
}

BLYNK_WRITE(VPIN_BUTTON_2) {
  toggleState_2 = param.asInt();
  digitalWrite(RelayPin2, toggleState_2);
}

BLYNK_WRITE(VPIN_BUTTON_3) {
  toggleState_3 = param.asInt();
  digitalWrite(RelayPin3, toggleState_3);
}

BLYNK_WRITE(VPIN_BUTTON_4) {
  toggleState_4 = param.asInt();
  digitalWrite(RelayPin4, toggleState_4);
}

BLYNK_WRITE(VPIN_BUTTON_5) {
  toggleState_5 = param.asInt();
  digitalWrite(RelayPin5, toggleState_5);
}

BLYNK_WRITE(VPIN_BUTTON_6) {
  toggleState_6 = param.asInt();
  digitalWrite(RelayPin6, toggleState_6);
}

BLYNK_WRITE(VPIN_BUTTON_7) {
  toggleState_7 = param.asInt();
  digitalWrite(RelayPin7, toggleState_7);
}

BLYNK_WRITE(VPIN_BUTTON_8) {
  toggleState_8 = param.asInt();
  digitalWrite(RelayPin8, toggleState_8);
}

BLYNK_CONNECTED() {
  // Request the latest state from the server
  Blynk.syncVirtual(VPIN_BUTTON_1);
  Blynk.syncVirtual(VPIN_BUTTON_2);
  Blynk.syncVirtual(VPIN_BUTTON_3);
  Blynk.syncVirtual(VPIN_BUTTON_4);
  Blynk.syncVirtual(VPIN_BUTTON_5);
  Blynk.syncVirtual(VPIN_BUTTON_6);
  Blynk.syncVirtual(VPIN_BUTTON_7);
  Blynk.syncVirtual(VPIN_BUTTON_8);
```

```
}

void relayOnOff(int relay){

   switch(relay){
     case 1:
          if(toggleState_1 == 1){
           digitalWrite(RelayPin1, LOW); // turn on relay 1
           toggleState_1 = 0;
           Serial.println("Device1 ON");
           }
          else{
           digitalWrite(RelayPin1, HIGH); // turn off relay 1
           toggleState_1 = 1;
           Serial.println("Device1 OFF");
           }
          delay(100);
     break;
     case 2:
          if(toggleState_2 == 1){
           digitalWrite(RelayPin2, LOW); // turn on relay 2
           toggleState_2 = 0;
           Serial.println("Device2 ON");
           }
          else{
           digitalWrite(RelayPin2, HIGH); // turn off relay 2
           toggleState_2 = 1;
           Serial.println("Device2 OFF");
           }
          delay(100);
     break;
     case 3:
          if(toggleState_3 == 1){
           digitalWrite(RelayPin3, LOW); // turn on relay 3
           toggleState_3 = 0;
           Serial.println("Device3 ON");
           }
          else{
           digitalWrite(RelayPin3, HIGH); // turn off relay 3
           toggleState_3 = 1;
           Serial.println("Device3 OFF");
           }
          delay(100);
     break;
     case 4:
          if(toggleState_4 == 1){
           digitalWrite(RelayPin4, LOW); // turn on relay 4
           toggleState_4 = 0;
           Serial.println("Device4 ON");
           }
          else{
           digitalWrite(RelayPin4, HIGH); // turn off relay 4
           toggleState_4 = 1;
           Serial.println("Device4 OFF");
           }
          delay(100);
     break;
     case 5:
          if(toggleState_5 == 1){
           digitalWrite(RelayPin5, LOW); // turn on relay 5
           toggleState_5 = 0;
```

```
          Serial.println("Device5 ON");
          }
         else{
          digitalWrite(RelayPin5, HIGH); // turn off relay 5
          toggleState_5 = 1;
          Serial.println("Device5 OFF");
          }
         delay(100);
    break;
    case 6:
         if(toggleState_6 == 1){
          digitalWrite(RelayPin6, LOW); // turn on relay 6
          toggleState_6 = 0;
          Serial.println("Device6 ON");
          }
         else{
          digitalWrite(RelayPin6, HIGH); // turn off relay 6
          toggleState_6 = 1;
          Serial.println("Device6 OFF");
          }
         delay(100);
    break;
    case 7:
         if(toggleState_7 == 1){
          digitalWrite(RelayPin7, LOW); // turn on relay 7
          toggleState_7 = 0;
          Serial.println("Device7 ON");
          }
         else{
          digitalWrite(RelayPin7, HIGH); // turn off relay 7
          toggleState_7 = 1;
          Serial.println("Device7 OFF");
          }
         delay(100);
    break;
    case 8:
         if(toggleState_8 == 1){
          digitalWrite(RelayPin8, LOW); // turn on relay 8
          toggleState_8 = 0;
          Serial.println("Device8 ON");
          }
         else{
          digitalWrite(RelayPin8, HIGH); // turn off relay 8
          toggleState_8 = 1;
          Serial.println("Device8 OFF");
          }
         delay(100);
    break;
    default : break;
    }
}

void with_internet(){
   //Manual Switch Control
   if (digitalRead(SwitchPin1) == LOW){
    delay(200);
    relayOnOff(1);
    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1); // Update Button Widget
   }
   else if (digitalRead(SwitchPin2) == LOW){
    delay(200);
```

```
    relayOnOff(2);
    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2); // Update Button Widget
   }
   else if (digitalRead(SwitchPin3) == LOW){
    delay(200);
    relayOnOff(3);
    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3); // Update Button Widget
   }
   else if (digitalRead(SwitchPin4) == LOW){
    delay(200);
    relayOnOff(4);
    Blynk.virtualWrite(VPIN_BUTTON_4, toggleState_4); // Update Button Widget
   }
   else if (digitalRead(SwitchPin5) == LOW){
    delay(200);
    relayOnOff(5);
    Blynk.virtualWrite(VPIN_BUTTON_5, toggleState_5); // Update Button Widget
   }
   else if (digitalRead(SwitchPin6) == LOW){
    delay(200);
    relayOnOff(6);
    Blynk.virtualWrite(VPIN_BUTTON_6, toggleState_6); // Update Button Widget
   }
   else if (digitalRead(SwitchPin7) == LOW){
    delay(200);
    relayOnOff(7);
    Blynk.virtualWrite(VPIN_BUTTON_7, toggleState_7); // Update Button Widget
   }
   else if (digitalRead(SwitchPin8) == LOW){
    delay(200);
    relayOnOff(8);
    Blynk.virtualWrite(VPIN_BUTTON_8, toggleState_8); // Update Button Widget
   }
}
void without_internet(){
   //Manual Switch Control
   if (digitalRead(SwitchPin1) == LOW){
    delay(200);
    relayOnOff(1);
   }
   else if (digitalRead(SwitchPin2) == LOW){
    delay(200);
    relayOnOff(2);
   }
   else if (digitalRead(SwitchPin3) == LOW){
    delay(200);
    relayOnOff(3);
   }
   else if (digitalRead(SwitchPin4) == LOW){
    delay(200);
    relayOnOff(4);
   }
   else if (digitalRead(SwitchPin5) == LOW){
    delay(200);
    relayOnOff(5);
   }
   else if (digitalRead(SwitchPin6) == LOW){
    delay(200);
    relayOnOff(6);
   }
   else if (digitalRead(SwitchPin7) == LOW){
```

```
    delay(200);
    relayOnOff(7);
   }
   else if (digitalRead(SwitchPin8) == LOW){
    delay(200);
    relayOnOff(8);
   }
}

void all_Switch_ON(){
 digitalWrite(RelayPin1, LOW); toggleState_1 = 0; delay(100);
 digitalWrite(RelayPin2, LOW); toggleState_2 = 0; delay(100);
 digitalWrite(RelayPin3, LOW); toggleState_3 = 0; delay(100);
 digitalWrite(RelayPin4, LOW); toggleState_4 = 0; delay(100);
 digitalWrite(RelayPin5, LOW); toggleState_5 = 0; delay(100);
 digitalWrite(RelayPin6, LOW); toggleState_6 = 0; delay(100);
 digitalWrite(RelayPin7, LOW); toggleState_7 = 0; delay(100);
 digitalWrite(RelayPin8, LOW); toggleState_8 = 0; delay(100);
}

void all_Switch_OFF(){
 digitalWrite(RelayPin1, HIGH); toggleState_1 = 1; delay(100);
 digitalWrite(RelayPin2, HIGH); toggleState_2 = 1; delay(100);
 digitalWrite(RelayPin3, HIGH); toggleState_3 = 1; delay(100);
 digitalWrite(RelayPin4, HIGH); toggleState_4 = 1; delay(100);
 digitalWrite(RelayPin5, HIGH); toggleState_5 = 1; delay(100);
 digitalWrite(RelayPin6, HIGH); toggleState_6 = 1; delay(100);
 digitalWrite(RelayPin7, HIGH); toggleState_7 = 1; delay(100);
 digitalWrite(RelayPin8, HIGH); toggleState_8 = 1; delay(100);
}

void Bluetooth_handle()
{
 bt_data = SerialBT.read();
// Serial.println(bt_data);
 delay(20);

 switch(bt_data)
    {
          case 'A': digitalWrite(RelayPin1, LOW); toggleState_1 = 0; break; // if 'A' received Turn on Relay1
      case 'a': digitalWrite(RelayPin1, HIGH); toggleState_1 = 1; break; // if 'a' received Turn off Relay1
      case 'B': digitalWrite(RelayPin2, LOW); toggleState_2 = 0; break; // if 'B' received Turn on Relay2
      case 'b': digitalWrite(RelayPin2, HIGH); toggleState_2 = 1; break; // if 'b' received Turn off Relay2
      case 'C': digitalWrite(RelayPin3, LOW); toggleState_3 = 0; break; // if 'C' received Turn on Relay3
      case 'c': digitalWrite(RelayPin3, HIGH); toggleState_3 = 1; break; // if 'c' received Turn off Relay3
      case 'D': digitalWrite(RelayPin4, LOW); toggleState_4 = 0; break; // if 'D' received Turn on Relay4
      case 'd': digitalWrite(RelayPin4, HIGH); toggleState_4 = 1; break; // if 'd' received Turn off Relay4
      case 'E': digitalWrite(RelayPin5, LOW); toggleState_5 = 0; break; // if 'E' received Turn on Relay5
      case 'e': digitalWrite(RelayPin5, HIGH); toggleState_5 = 1; break; // if 'e' received Turn off Relay5
      case 'F': digitalWrite(RelayPin6, LOW); toggleState_6 = 0; break; // if 'F' received Turn on Relay6
      case 'f': digitalWrite(RelayPin6, HIGH); toggleState_6 = 1; break; // if 'f' received Turn off Relay6
      case 'G': digitalWrite(RelayPin7, LOW); toggleState_7 = 0; break; // if 'G' received Turn on Relay7
      case 'g': digitalWrite(RelayPin7, HIGH); toggleState_7 = 1; break; // if 'g' received Turn off Relay7
      case 'H': digitalWrite(RelayPin8, LOW); toggleState_8 = 0; break; // if 'H' received Turn on Relay8
      case 'h': digitalWrite(RelayPin8, HIGH); toggleState_8 = 1; break; // if 'h' received Turn off Relay8
      case 'Z': all_Switch_ON(); break; // if 'Z' received Turn on all Relays
      case 'z': all_Switch_OFF(); break; // if 'z' received Turn off all Relays
      default : break;
    }
}
```

```
void checkBlynkStatus() { // called every 3 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    wifiFlag = 1;
    digitalWrite(wifiLed, LOW); //Turn off WiFi LED
  }
  if (isconnected == true) {
    wifiFlag = 0;
    digitalWrite(wifiLed, HIGH); //Turn on WiFi LED
  }
}

void setup()
{
  Serial.begin(9600);

  btStart(); //Serial.println("Bluetooth On");

  SerialBT.begin("ESP32_BT"); //Bluetooth device name
  Serial.println("The device started, now you can pair it with bluetooth!");
  delay(5000);

  pinMode(RelayPin1, OUTPUT);
  pinMode(RelayPin2, OUTPUT);
  pinMode(RelayPin3, OUTPUT);
  pinMode(RelayPin4, OUTPUT);
  pinMode(RelayPin5, OUTPUT);
  pinMode(RelayPin6, OUTPUT);
  pinMode(RelayPin7, OUTPUT);
  pinMode(RelayPin8, OUTPUT);

  pinMode(wifiLed, OUTPUT);

  pinMode(SwitchPin1, INPUT_PULLUP);
  pinMode(SwitchPin2, INPUT_PULLUP);
  pinMode(SwitchPin3, INPUT_PULLUP);
  pinMode(SwitchPin4, INPUT_PULLUP);
  pinMode(SwitchPin5, INPUT_PULLUP);
  pinMode(SwitchPin6, INPUT_PULLUP);
  pinMode(SwitchPin7, INPUT_PULLUP);
  pinMode(SwitchPin8, INPUT_PULLUP);

  //During Starting all Relays should TURN OFF
  digitalWrite(RelayPin1, toggleState_1);
  digitalWrite(RelayPin2, toggleState_2);
  digitalWrite(RelayPin3, toggleState_3);
  digitalWrite(RelayPin4, toggleState_4);
  digitalWrite(RelayPin5, toggleState_5);
  digitalWrite(RelayPin6, toggleState_6);
  digitalWrite(RelayPin7, toggleState_7);
  digitalWrite(RelayPin8, toggleState_8);

  WiFi.begin(WIFI_SSID, WIFI_PASS);
  timer.setInterval(3000L, checkBlynkStatus); // check if Blynk server is connected every 3 seconds
  Blynk.config(AUTH);
  delay(2000);
}

void loop()
{
```

```
if (WiFi.status() != WL_CONNECTED)
{
 // Serial.println("WiFi Not Connected");

}
else
{
  //Serial.println("WiFi Connected");
  Blynk.run();
}

timer.run(); // Initiates SimpleTimer
if (wifiFlag == 0){
  with_internet();
  }
else{
  without_internet();
  if (SerialBT.available()){
   Bluetooth_handle();
 }
}
```

# REFERENCES


1. T. Chaurasia and P. K. Jain, "Enhanced Smart Home Automation System based on Internet of Things," 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, pp. 709-713, doi: 10.1109/I-SMAC47947.2019.9032685.

2. M. Asadullah and K. Ullah, "Smart home automation system using Bluetooth technology," 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), 2017, pp. 1-6, doi: 10.1109/ICIEECT.2017.7916544.

3. Riley, Mike. (2012) Programming Your Home, The Pragmatic Programmers,LLC

4. Spivey, Dwight. (2015) Home Automation for Dummies, New Jersey: John Willey & Sons Inc.

5. Angel Deborah Suseelan, Naveen Hariharan, Satish Palaniappan. Home Automation Systems - A Study. International Journal of Computer Applications. April 2015