

# **INTRODUCTION**

## **1.1 Introduction**

The Internet of Things (IoT) is an emerging technology paradigm that is revolutionizing the way we interact with the world around us. It refers to the network of interconnected physical devices, vehicles, appliances, and other objects embedded with sensors, software, and network connectivity, enabling them to collect and exchange data over the internet. The IoT has the potential to transform various industries, enhance efficiency, and improve the quality of life for individuals and communities. This project aims to explore and leverage the power of IoT to address specific challenges or achieve particular objectives.

In this project, we are using ESP32 chipset for automate the system. The ESP32 module is a powerful and versatile microcontroller unit (MCU) that has gained significant popularity in the field of home automation. Developed by Espressif Systems, the ESP32 is a dual-core MCU with built-in Wi-Fi and Bluetooth capabilities, making it an ideal choice for creating smart and connected devices for home automation applications. This introduction will provide an overview of the ESP32 module and highlight its features that make it suitable for home automation systems.

## **1.2 Background Research**

The process of improving and upgrading the living standard of the house has been raised due to the advanced technology applied in this era society. The Home Automation System was implemented for the decades but due to higher cost of these kind of project, it still remains as a expensive product for consumers. The concept of home automation is being there around for a long time, but it can be established in short time. Home automation has emerged as a prominent feature of this generation, transforming the way we interact with our living spaces. With advancements in technology and the widespread adoption of IoT, home automation systems have become more accessible and sophisticated. Today, homeowners can seamlessly control and monitor various aspects of their homes with just a few taps on their smartphones or voice commands. From smart lighting and temperature control to automated security systems and entertainment setups, home automation offers convenience, energy efficiency, and enhanced security. It enables homeowners to create personalized and responsive environments that adapt to their preferences and schedules. Through

the integration of sensors, actuators, and connectivity, home automation systems have revolutionized the way we live, providing comfort, efficiency, and peace of mind in this digital age.

## 1.3 Project Aim

The main aims of a home automation system project is to design, develop, and implement an efficient and user-friendly system that enhances the overall living experience for homeowners. The project aims to integrate various technologies, devices, and sensors to create a smart and connected home environment. The primary objectives of the project include:

1. **Convenience and Control:** The aim is to provide homeowners with seamless control and management of various home devices and systems. This includes the ability to control lighting, temperature, security systems, entertainment devices, and other appliances remotely using smartphones, voice commands, or a centralized control interface.
2. **Energy Efficiency:** The project aims to promote energy efficiency by implementing automation features that optimize energy consumption. This includes features such as scheduling and automation of lighting, HVAC systems, and energy-consuming devices to minimize wastage and reduce utility costs.
3. **Enhanced Security and Safety:** The aim is to integrate security systems and sensors, such as surveillance cameras, motion detectors, and door/window sensors, to enhance home security and provide real-time monitoring. The project aims to enable homeowners to receive alerts and notifications regarding any security breaches or unusual activities.
4. **Personalization and Customization:** The aim is to provide homeowners with the ability to personalize and customize their automation settings based on their preferences and lifestyle. This includes the creation of personalized scenes, routines, and automation schedules that align with their daily routines and specific needs.
5. **Seamless Integration and Scalability:** The project aims to ensure seamless integration and interoperability of various devices and systems within the home automation ecosystem. It should be designed to accommodate future expansions and additions, allowing homeowners to easily integrate new devices or technologies into the existing system.
6. **User-Friendly Interface:** The aim is to develop a user-friendly interface, such as a mobile application or a centralized control panel, that is intuitive and easy to use. The interface should provide clear and straightforward controls, monitoring features, and notifications for effortless management of the home automation system.

7. **Cost-Effectiveness:** The project aims to strike a balance between upfront costs and long-term benefits. It aims to provide a cost-effective solution by considering the affordability of devices, energy savings resulting from automation, and maintenance requirements.

## 1.4 Project Objective

The objective of this project is to implement a low cost, reliable and scalable E-HOME system that can be used to remotely switch on or off any household appliance, using a microcontroller to achieve hardware simplicity, Low cost short messaging service for feedback voice dial from phone to toggle the switch state.

## 1.5 Scopes

This project aims for designing a prototype of controlling the home appliances wirelessly via a mobile application that provides the features switch mode. The application can be run on android devices. The system can be used in wide range of areas.

The system integrated with different features can be applied in the following fields.

- a) **Smart Switch Functionality:** The project aims to implement smart switch functionality, which allows homeowners to remotely control and automate the operation of electrical appliances and lighting fixtures. This includes the ability to turn devices on/off, adjust brightness or dimness, and schedule automated routines.
- b) **Connectivity:** The project scope involves integrating the smart switches with a home automation system or a dedicated smart home platform. This enables seamless communication and control through various connectivity options such as Wi-Fi, Zigbee, Z-Wave, or Bluetooth.
- c) **Mobile Application or Control Interface:** The project may include the development or utilization of a mobile application or a centralized control interface. This interface allows homeowners to conveniently control and monitor the smart switches from their smartphones, tablets, or a dedicated control panel.
- d) **Compatibility and Integration:** The project aims to ensure compatibility and integration with other devices and systems within the home automation ecosystem. This includes

integration with voice assistants (e.g., Amazon Alexa or Google Assistant), integration with other smart devices (e.g., thermostats, security systems), and interoperability with other automation protocols or platforms.

- e) **Automation and Scheduling:** The scope involves implementing automation and scheduling features for the smart switches. This allows homeowners to create customized automation routines or schedules, such as turning on lights at specific times or automating the operation of appliances based on occupancy or specific events.
- f) **Energy Monitoring and Efficiency:** The project may include energy monitoring capabilities, enabling homeowners to track and analyze the energy consumption of connected devices. This empowers homeowners to make informed decisions about energy usage and promotes energy efficiency.
- g) **Security and Safety:** The scope may encompass integrating security and safety features into the smart switch system. This can include features like overcurrent protection, surge protection, and integration with home security systems for enhanced safety and security.
- h) **User-Friendly Configuration and Setup:** The project aims to provide a user-friendly setup and configuration process for the smart switches. This includes clear instructions, intuitive user interfaces, and straightforward integration with the home automation system.
- i) **Scalability and Expansion:** The project should be designed with scalability in mind, allowing for easy expansion and addition of smart switches in the future. This enables homeowners to gradually expand their home automation system as needed.
- j) **Testing and Validation:** The project scope includes thorough testing and validation of the smart switch functionality, ensuring reliable and consistent operation. This involves testing different scenarios, performance evaluation, and user acceptance testing.

# **SOFTWARE /HARDWARE REQUIREMENTS**

## **2.1 HARDWARE**

1. ESP32
2. Relay module
3. Male female jumper wire
4. Power adapter

## **2.2 SOFTWARE**

1. Blynk
2. Arduino IDE

# DESCRIPTION OF SOFTWARE /HARDWARE USED

## 2.1 HARDWARE

**2.1.1. ESP32:-**The ESP32 module is a popular and versatile microcontroller module widely used in the field of IoT (Internet of Things) applications. It is developed by Espressif Systems and is based on the ESP32 system-on-a-chip (SoC). The ESP32 module combines a powerful microcontroller unit (MCU) with built-in Wi-Fi and Bluetooth capabilities, making it an ideal choice for wireless connectivity and IoT projects. Here are some key features and characteristics of the ESP32 module:

- a) **Dual-Core Processor:** The ESP32 module features a dual-core Xtensa LX6 microprocessor, which operates at clock frequencies of up to 240 MHz. This dual-core architecture enables multitasking and provides increased processing power for running complex applications.
- b) **Wi-Fi Connectivity:** The ESP32 module includes integrated Wi-Fi connectivity, supporting both 2.4 GHz and 5 GHz frequency bands. It supports various Wi-Fi protocols, including 802.11 b/g/n and 802.11ac, allowing seamless wireless connectivity for IoT applications.
- c) **Bluetooth Connectivity:** In addition to Wi-Fi, the ESP32 module also integrates Bluetooth technology, including Bluetooth Classic and Bluetooth Low Energy (BLE). This enables the module to connect and communicate with a wide range of Bluetooth-enabled devices, such as smartphones, sensors, and peripherals.
- d) **GPIO Pins:** The ESP32 module provides a large number of General Purpose Input/Output (GPIO) pins, allowing for easy interfacing with external sensors, actuators, and other electronic components. These GPIO pins can be configured for various purposes, including digital input/output, analog input, PWM (Pulse Width Modulation) output, and more.
- e) **Memory:** The ESP32 module offers ample memory resources for program storage and data handling. It typically comes with 4 MB flash memory for storing program code and data, as well as additional RAM for runtime operations and variables.
- f) **Peripherals:** The module features a wide range of peripherals, including SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), PWM, ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), and more. These peripherals enable seamless communication with external devices and sensors.
- g) **Development Environment:** The ESP32 module can be programmed using various development environments and programming languages, including the Arduino IDE

(Integrated Development Environment), ESP-IDF (Espressif IoT Development Framework), and MicroPython. This flexibility allows developers to choose the most suitable programming environment based on their familiarity and project requirements.

- h) Low Power Consumption:** The ESP32 module is designed to be power-efficient, offering various power-saving modes and sleep states. This makes it suitable for battery-powered or energy-conscious IoT applications, where minimizing power consumption is crucial.

The ESP32 module's versatility, robustness, and extensive features have made it a popular choice for a wide range of IoT projects, including home automation, smart devices, wearables, industrial automation, and more. Its combination of Wi-Fi, Bluetooth, and powerful processing capabilities provides a solid foundation for developing connected and wireless-enabled applications.

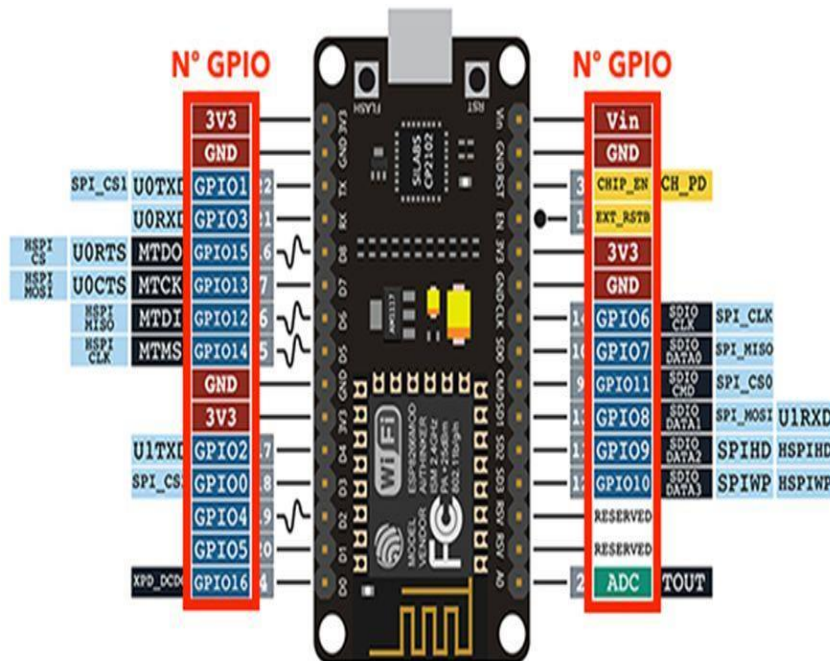


Figure 1. ESP32 Pinout

Pins used:

1. Vin is connected to power supply output 5VDC.
2. GND is ground.
3. D1,D2,D3 and D4 are used as digital outputs.
4. A0 is used as analogue signal input to input sensor signal.

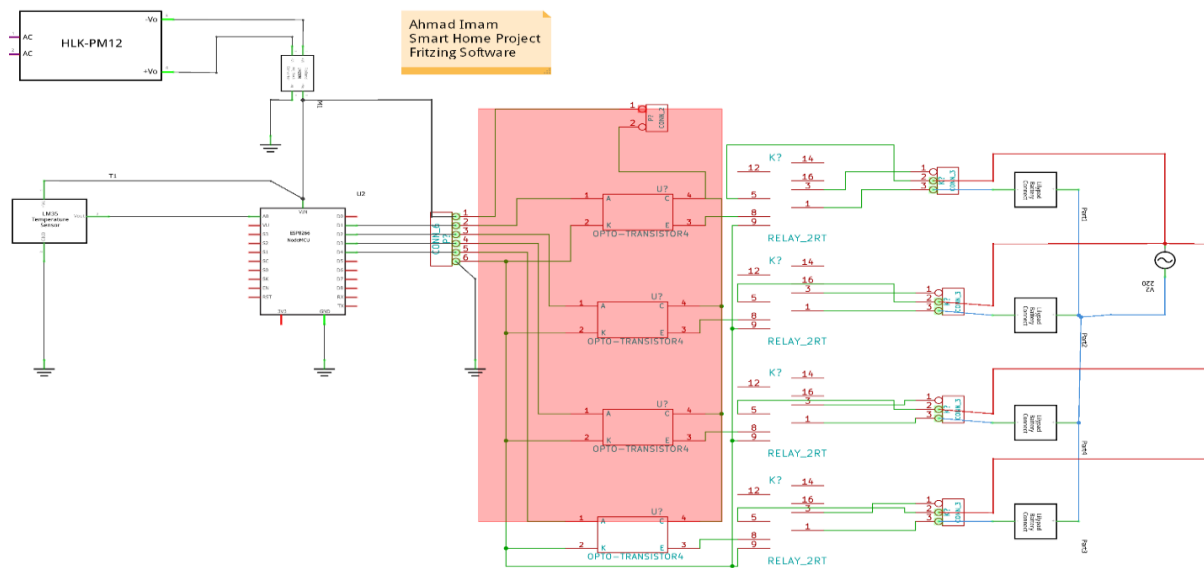


Figure 2. The Circuit Diagram

**2.1.2. Relay Module:** - The relay module is an electronic switch that can be turned on or off deciding external low power electric pulse in as input. They are designed to be controlled with low voltages like 3.3V like the ESP-32, ESP8266, etc, or 5V like Arduino.

On top of relay modules, you will see channels which are the black cube as seen above. You may see relay modules with more channels from two channels, four channels to also eight channels.

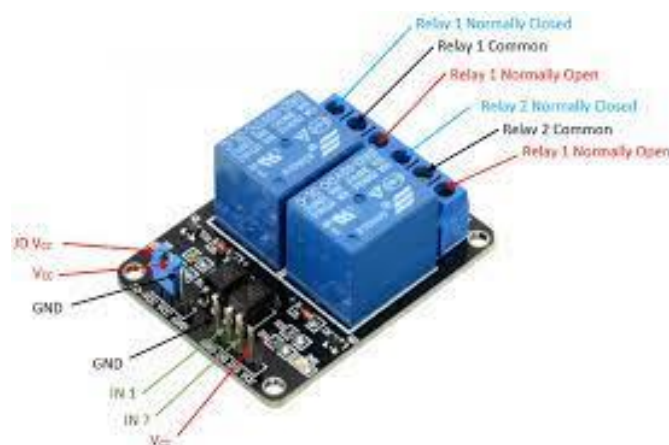


Figure 3 relay module



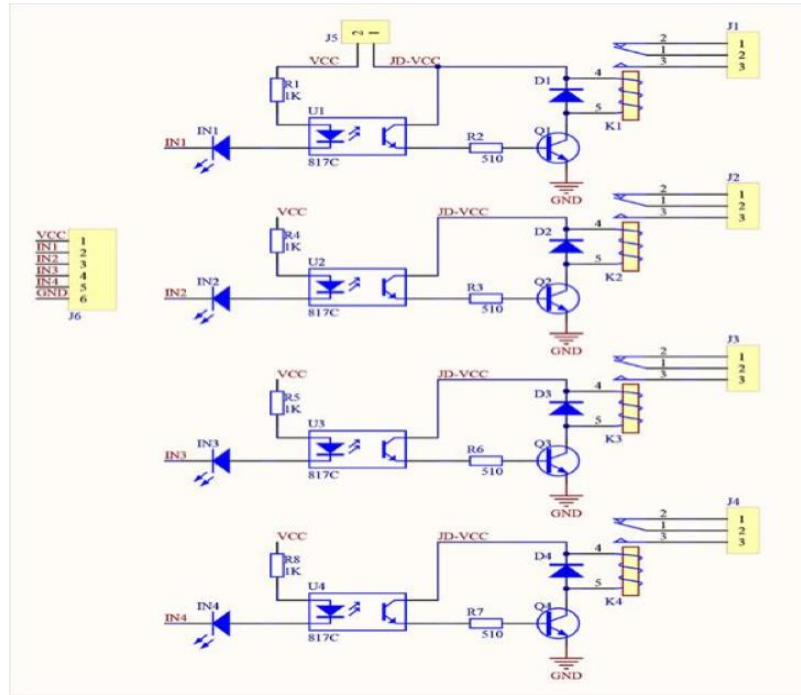


Figure 4. Relay Module Circuit Diagram

**2.1.3. Jumper wires:-** A **jump wire** (also known as **jumper**, **jumper wire**, **DuPont wire**) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots in a breadboard, the header connector of a circuit board, or a piece of test equipment.

**There are different types of jumper wires. Some are.**

- a) **Male-to-Male (M-M) Jumper Wires:** These jumper wires have male pins on both ends, making them suitable for connecting two female headers or male pins together. They are widely used for breadboard connections or connecting components to microcontrollers, sensors, and other electronic modules.
- b) **Male-to-Female (M-F) Jumper Wires:** M-F jumper wires have a male pin on one end and a female socket on the other end. They are used to connect female headers or sockets to male pins. These wires are commonly used to connect components to a breadboard, Arduino boards, Raspberry Pi, or other development boards.
- c) **Female-to-Female (F-F) Jumper Wires:** F-F jumper wires feature female sockets on both ends. They are primarily used for connecting two male pins or headers together, without the

need for a breadboard. F-F jumper wires are useful for direct connections between components, prototyping with module breakout boards, or extending the reach of existing cables.

- d) **Alligator Clip Jumper Wires:** These jumper wires come with alligator clips on both ends. They are particularly useful for creating temporary connections to large components or non-standard connection points. The alligator clips provide a firm grip and can easily attach to terminals, wires, or test points.
- e) **Ribbon Cable Jumper Wires:** Ribbon cables consist of multiple wires arranged in a flat ribbon-like configuration. They typically have female sockets on one end and can be split into individual wires or groups of wires for specific connections. Ribbon cables are commonly used for connecting devices with standard IDC (Insulation-Displacement Connector) connectors, such as connecting displays, LCDs, or modules to a control board.
- f) **Dupont Jumper Wires:** Dupont jumper wires are a popular type of jumper wires commonly used in prototyping and breadboarding. They feature plastic housing with male pins on one end and are available in various lengths. Dupont wires are easy to insert and remove from breadboards and are compatible with standard 0.1-inch (2.54mm) spacing.



Figure 5. Jumper wire

**2.1.4. Power Adapter:** - An "AC-adapter" or a "charger," power adapters plug into a wall switch board and convert AC to a DC voltage. Computers use multiple DC voltages, and the power adapter is the external part of the power supply for a laptop. The additional DC voltages are created by internal circuits. Desktop computer power supplies are in one internal unit, which converts AC to all DC voltages.

Power adapters also exist for other purposes; for example, to output a different AC voltage, rather than DC. See transformer, power supply, power distribution unit, wall wart, wall tap, power strip and USB charger.



Figure 6. Power Adapter

## 2.2 SOFTWARE

### 2.2.1. Arduino IDE :-

The Arduino IDE (Integrated Development Environment) is a software application that serves as a central tool for programming and developing applications for Arduino boards. It provides a user-friendly interface and a comprehensive set of features to simplify the process of writing, compiling, and uploading code to Arduino microcontrollers. The IDE consists of various components and functionalities that aid in the development process.

First and foremost, the Arduino IDE includes a code editor with features like syntax highlighting, auto-completion, and indentation. This editor supports the C++ programming language, which is the primary language used for Arduino development. It allows users to write and edit Arduino sketches, which are the programs or applications for Arduino boards. These sketches typically contain two main functions: **setup()** and **loop()**. The **setup()** function is executed once when the Arduino board starts, while the **loop()** function runs continuously in a loop.

Additionally, the Arduino IDE provides a library manager that offers a collection of pre-written libraries. These libraries contain useful functions and code snippets that can be easily included in sketches to extend their functionality and simplify development. Users can also manage and install additional libraries from external sources as per their project requirements.

The IDE includes a Board Manager feature that allows users to select the specific Arduino board they are working with or install additional board definitions if necessary. Each board has unique characteristics such as the type of microcontroller, clock speed, and available memory. Specifying the board in the IDE ensures that the code is compiled and uploaded correctly.

Furthermore, the Arduino IDE offers a Serial Monitor, which is a built-in tool for communication with the Arduino board via the serial port. This feature allows users to send and receive data, making it valuable for debugging and monitoring the behavior of the program. It is especially useful for displaying sensor readings, debugging output, or interacting with the Arduino in real-time.

The Arduino IDE simplifies the process of uploading code to Arduino boards. With a single click, users can compile their code and upload it to the connected board via a USB cable. The IDE takes care of the compilation process and communicates with the board's bootloader to transfer the compiled code.

In addition to its core functionalities, the Arduino IDE benefits from a vibrant community of Arduino enthusiasts and developers. The official Arduino website provides extensive documentation, tutorials, and examples to support users in learning and exploring Arduino programming. There are also numerous online forums, communities, and resources where users can seek help, share their projects, and collaborate with others.



Figure 7. Arduino IDE

**2.2.2 BASIC ARDUINO SKETCH:** - Arduino programs are called sketches. this matches the Arduino philosophy of prototyping and creativity. Arduino programs and builds are similar to an artist's sketches on paper. the only difference being that we use a computer and electrical components to create.

Creating and programming in the Arduino environment follows these simple steps:

1. Write the sketches in the IDE
2. Upload the sketch to the Arduino board
3. The sketch runs on its own in the board



Figure 8. Connecting Arduino to PC

### 2.2.3 STRUCTURE

Every basic Arduino sketch is fairly simple in structure and is composed of two basic parts:

- void setup() function
- void loop() function

Both functions are required for a program to work properly.

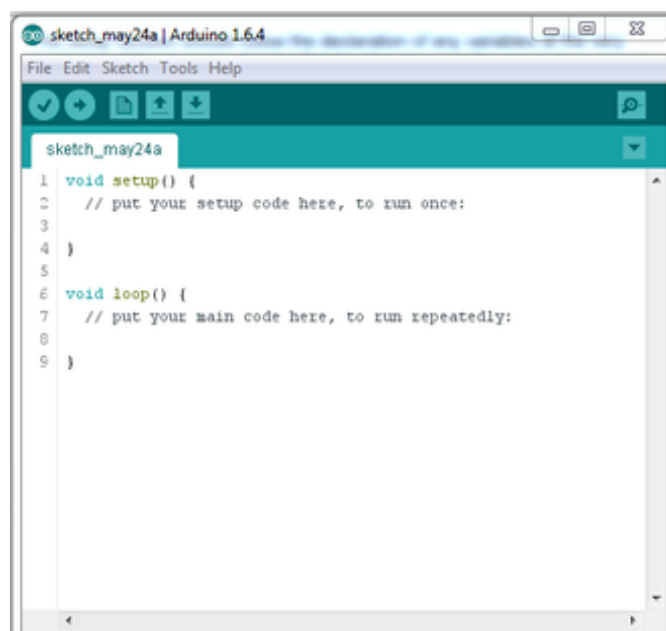


Figure 9. Arduino Basic syntax

### 2.2.4 VOID SETUP()

The void setup() function is the first function to run in a program and is only run once. It is used to set the pin modes or to begin the serial communication. It must be included in every sketch, even if there is no code to run.

```
void setup() {  
  // put your setup code here, to run once:  
  
}
```

### 2.2.5 VOID LOOP()

The void loop() function is run after the setup function. It functions as its name suggests. It continuously runs the code written inside its curly brackets. Once it reaches the end of the code, it loops back to the beginning and runs the code again. This allows the sketch to continuously change, respond, and control the Arduino board. The loop function will continue to run until the reset button on the board is pressed or the power source has been removed.

```
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

### 2.2.6 VERIFY, COMPILE, AND UPLOAD THE SKETCH

After you have written your code you need to check whether it is correct. Click the verify icon (check mark) in the menu bar.



Figure 10. Uploading code

The Arduino IDE will check your code for mistakes in syntax. If there are no mistakes, a message will appear in the message box as shown below.

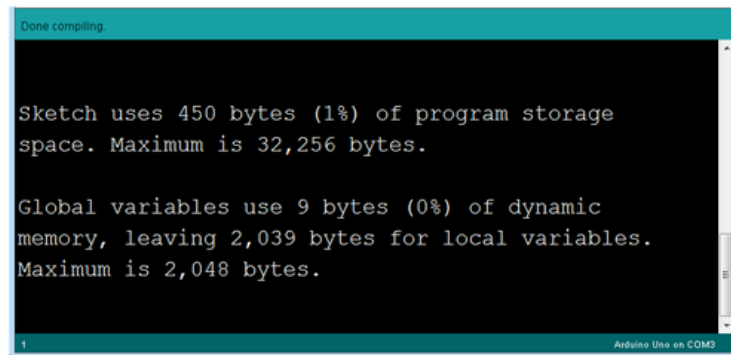


Figure 11. Compilation of code

Successful compilation of a sketch. It also tells you how much memory the sketch will use.

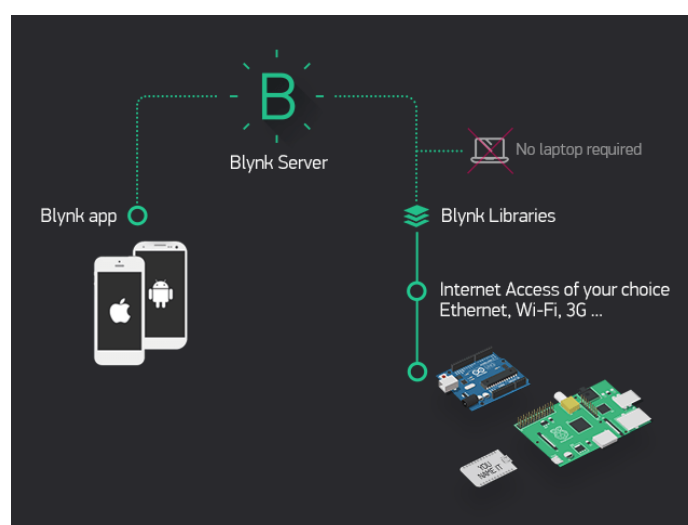
Once you have verified and compiled your sketch upload it your Arduino board by clicking on the upload button (white arrow) in the menu.

### 2.3.1 BLYNK APPLICATION: -

Blynk is a popular mobile application that enables easy and intuitive control and monitoring of IoT (Internet of Things) devices. It provides a user-friendly interface for creating custom mobile apps and connecting them to various hardware platforms, including Arduino, Raspberry Pi, ESP8266, and others. Blynk simplifies the process of building IoT projects by providing pre-built app components and seamless cloud connectivity. Here are the key components of the Blynk application:

1. **Blynk App:** The Blynk mobile app is available for both iOS and Android devices. It serves as the main interface for users to interact with their IoT devices. The app allows users to create custom dashboards with a variety of widgets and controls to monitor and control connected devices. It provides a drag-and-drop interface for designing the app layout and configuring the functionality of individual widgets.
2. **Blynk Cloud:** Blynk utilizes a cloud-based infrastructure that acts as a bridge between the mobile app and the IoT devices. The Blynk Cloud securely handles the communication and data transfer between the app and the connected hardware. It enables real-time updates and synchronization of data between the mobile app and the devices, ensuring seamless control and monitoring.

3. **Widgets:** Blynk offers a wide range of widgets that can be added to the app's dashboard to create interactive controls and visualizations. Some commonly used widgets include buttons, sliders, gauges, graphs, LED displays, and LCD displays. These widgets can be customized and configured to send commands to the connected devices, receive sensor data, display real-time values, and visualize data trends.
4. **Virtual Pins:** Blynk uses virtual pins to establish communication between the app and the hardware devices. Virtual pins are virtual placeholders that allow data to be sent and received between the app and the devices. They enable bi-directional communication, allowing the app to control hardware and receive data from sensors or other sources.
5. **Blynk Libraries:** Blynk provides software libraries and development tools that facilitate the integration of Blynk functionality into the firmware of IoT devices. These libraries are available for popular platforms such as Arduino, Raspberry Pi, ESP8266, and others. They simplify the process of establishing a connection with the Blynk Cloud, handling data exchange, and responding to commands from the app.
6. **Blynk API:** Blynk offers a RESTful API that allows developers to interact with the Blynk Cloud programmatically. The API enables advanced integrations and customizations, such as retrieving device data, controlling devices remotely, and managing app configurations.
7. **Energy System:** Blynk incorporates an energy system that assigns a certain amount of energy to each user account. Energy is the currency within the Blynk platform and is used to activate widgets, create projects, and access premium features. Different widgets consume varying amounts of energy, and additional energy can be purchased if needed.



**Figure 12. Working principle of Blynk application.**



Every time a radio button is accessed in the Blynk application, the message travels to the Blynk Cloud, where it finds the specific hardware by the unique generated authentication token. It works in the same way for the opposite direction.

## Proposed work and Coding

### 3.1 Proposed work

#### 3.1.1 Blynk System Principle

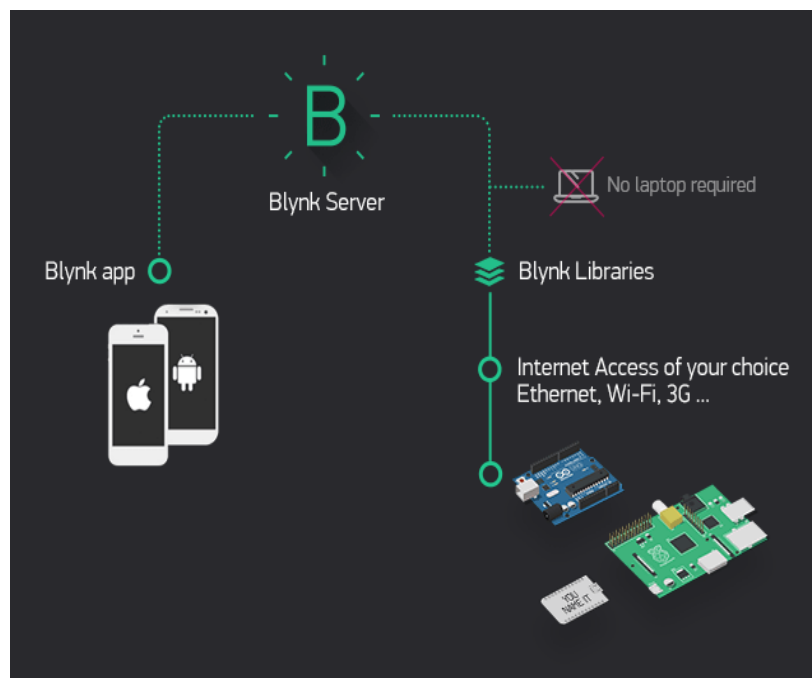


Figure 13. Blynk System Principle

The system is based on ESP32 board as an internet of things system. The ESP32 is connected to the internet from the hotspot of the smart phone via WIFI connection as the ESP32 has ESP 32 circuit to connect with the internet.

ESP32 to be connected to the hotspot of the smart phone, needs to be identified to the name of hotspot, the password and token code letting the server of Blynk connects them together. You may need the computer once to transfer code from Arduino IDE to the ESP32 kit to prepare the software part of the project. Figure 1 shows that the server of Blynk application will process the smartphone-ESP32 connection. Blynk libraries are ZIP files can be downloaded from GitHub website to be imported to the Arduino IDE library.

Blynk server will check for internet connection, ESP32 with android hotspot, the ESP32 code includes the token code, the name of hotspot and it's password. The information included to the code must be match with the hotspot information to allow ESP 32 connect with the WIFI to be as a channel to exchange commands between smart phone and ESP32. Remaining processes are just commands sent from Blynk application to ESP32 to control loads those are connected to the relay kit as shown in Figure 2. And sensor output value is sent reverse to the Blynk application from ESP32 kit

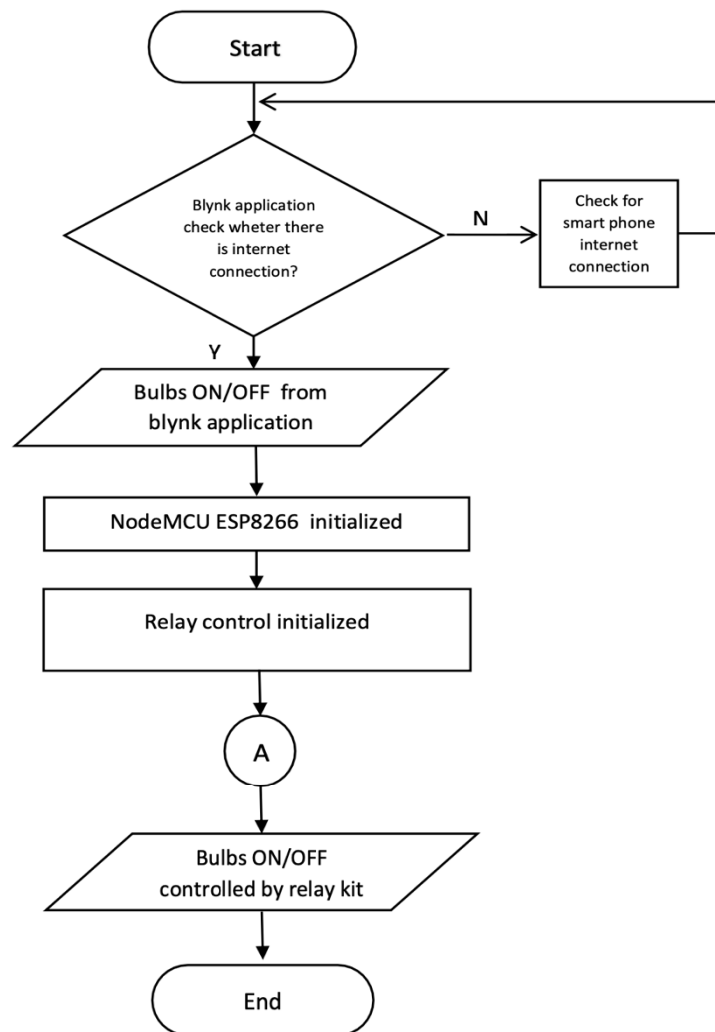


Figure 14. Flowchart of Load ON/OFF

### 3.1.2 The Block Diagram of the System

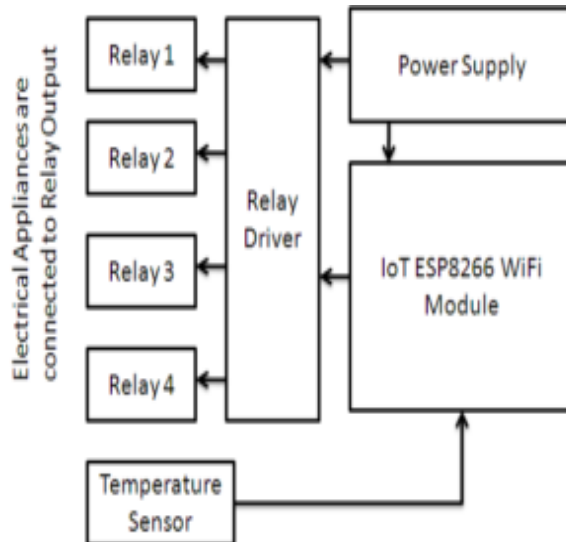


Figure 15. System Block Diagram

The block diagram of an ESP32-based wireless switch using a two-way relay module typically consists of several key components. Here's a breakdown of the main elements:

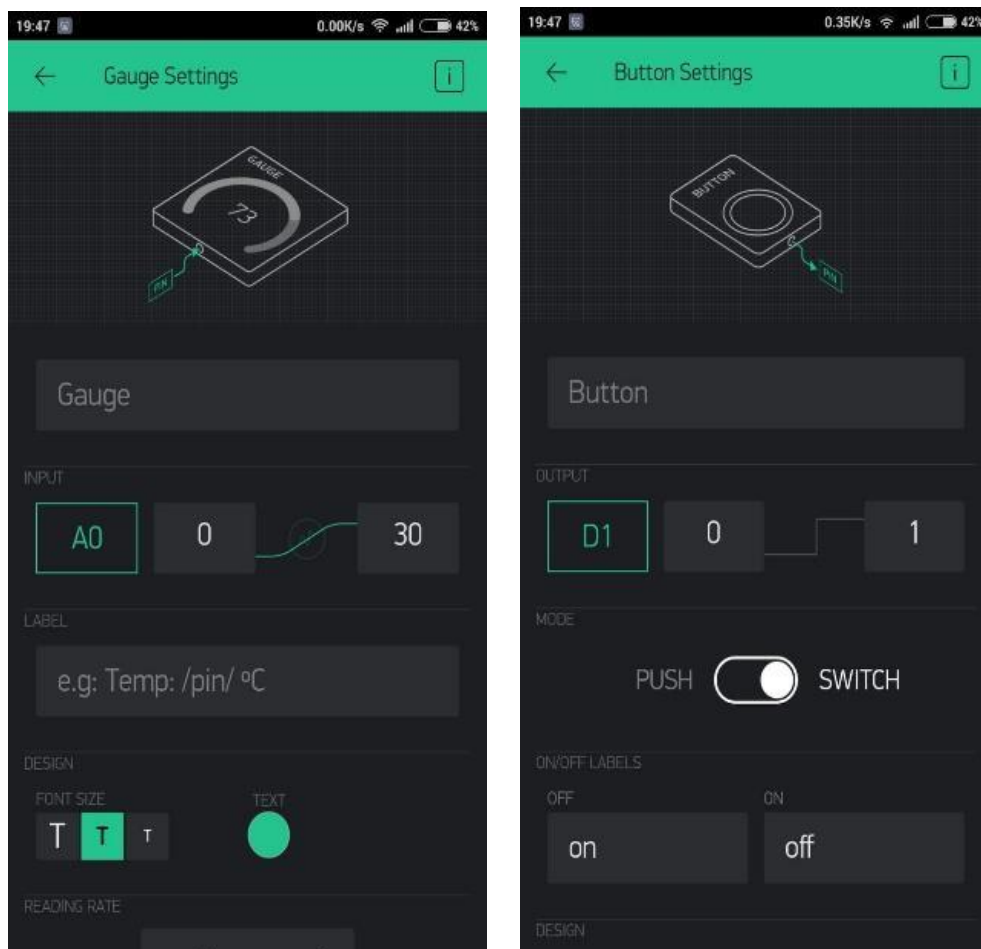
- ESP32 Microcontroller:** The ESP32 is a powerful microcontroller module that serves as the brain of the system. It integrates a dual-core processor, Wi-Fi and Bluetooth connectivity, and various input/output (I/O) pins. The ESP32 is responsible for running the software, handling wireless communication, and controlling the relay module.
- Two-Way Relay Module:** The two-way relay module acts as a switch to control the electrical circuits connected to it. It usually contains two relays, each with a set of normally open (NO) and normally closed (NC) contacts. These relays can be used to turn on or off different electrical loads, such as lights, fans, or appliances.
- Power Supply:** The power supply provides the necessary electrical energy to operate the system. It typically involves a suitable voltage source, such as a DC power adapter or a battery. The ESP32 and relay module require appropriate power levels to function correctly.
- Input Devices:** Input devices are used to trigger the switch operation. They can include physical buttons, touch sensors, or even virtual switches integrated into a mobile application.

When an input device is activated, it sends a signal to the ESP32, indicating the desired action to be taken.

- e) **Control Logic:** The control logic is implemented in the software running on the ESP32 microcontroller. It interprets the input signals received from the input devices and decides the corresponding action to be taken. For example, when a button is pressed, the control logic determines whether to activate or deactivate the relay module based on the current state.
- f) **External Loads:** The external loads are the electrical devices or circuits connected to the two-way relay module. These can be lights, appliances, or any other electrical equipment that requires remote control or automation.

### 3.1.3 Blynk application and Arduino IDE Preparation and Running

This project is running by Blynk application. Down load the application to a smart phone from Google play store and then create a project on it with four switches and one gauge to be as a temperature scale. Set buttons to be switches on D1, D2, D3 and D4. Then set gauge on A0 because the sensor output is on A0 in ESP32 board. Figure 11 shows screenshots from Blynk application



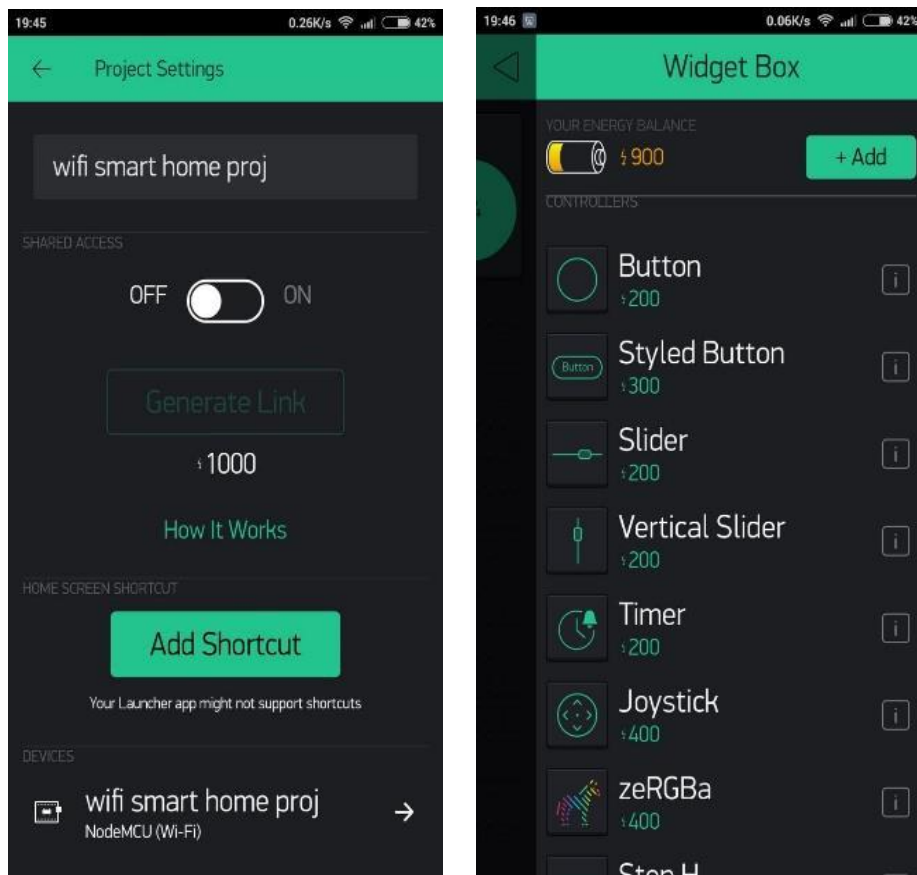


Figure 16. Screenshots from Blynk Application

# ARDUNIO IDE PREPARATION

## 4.1 Installing ESP32 Add-on in Arduino IDE

For installing the ESP32 board in Arduino IDE, follow these next instructions:

a) In your Arduino IDE, go to **File> Preferences**.

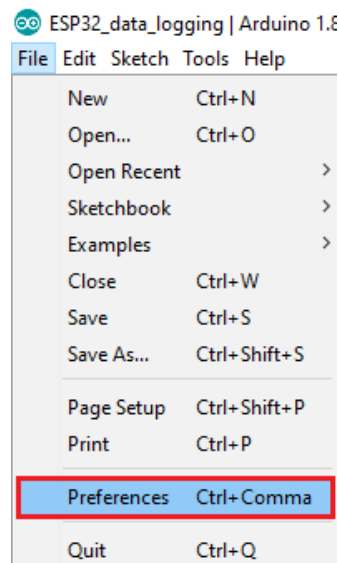


Figure 17.File> Preference

b) Enter the following into the “Additional Board Manager URLs” field:

[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

Then, click the “OK” button:

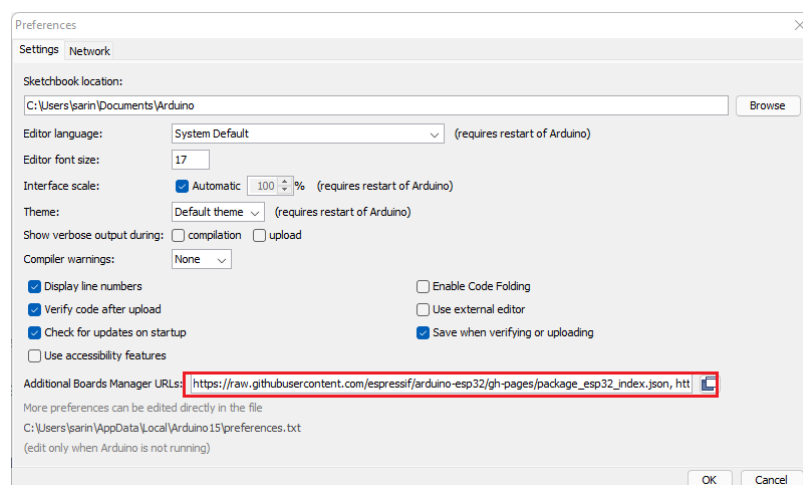


Figure 18 Integrating URL for ESP32.

**Note:** if we already have the ESP-32 boards URL, we can separate the URLs with a comma as follows:

[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

[http://arduino.esp32.com/stable/package\\_ESP32com\\_index.json](http://arduino.esp32.com/stable/package_ESP32com_index.json)

c) Open the Boards Manager. Go to **Tools > Board > Boards Manager...**

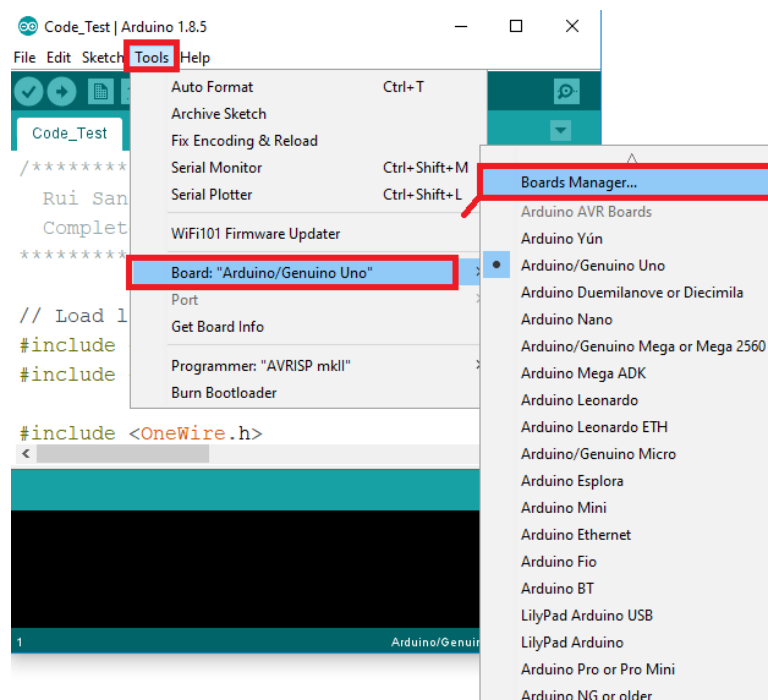


Figure 19. Opening Boards Manager

d) Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**“:

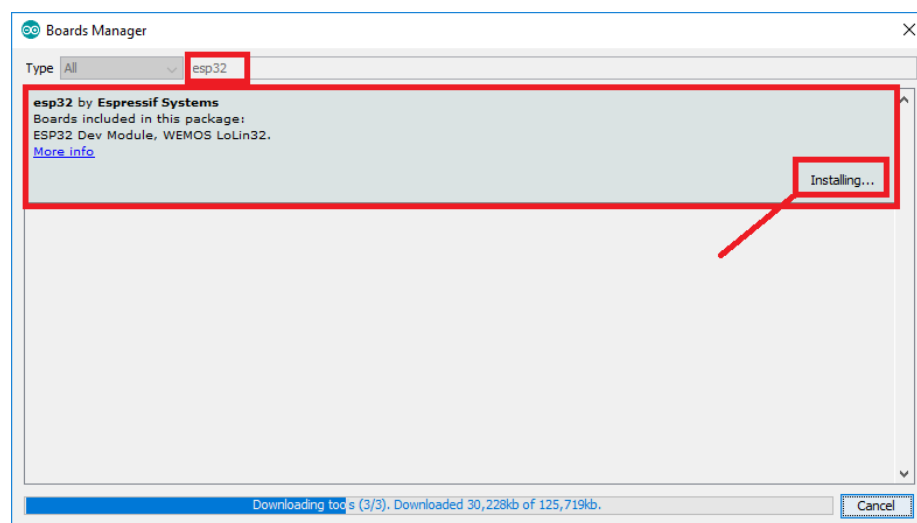


Figure 20. Searching/Installing ESP32 In IDE

e) That's it. It should be installed after a few seconds.

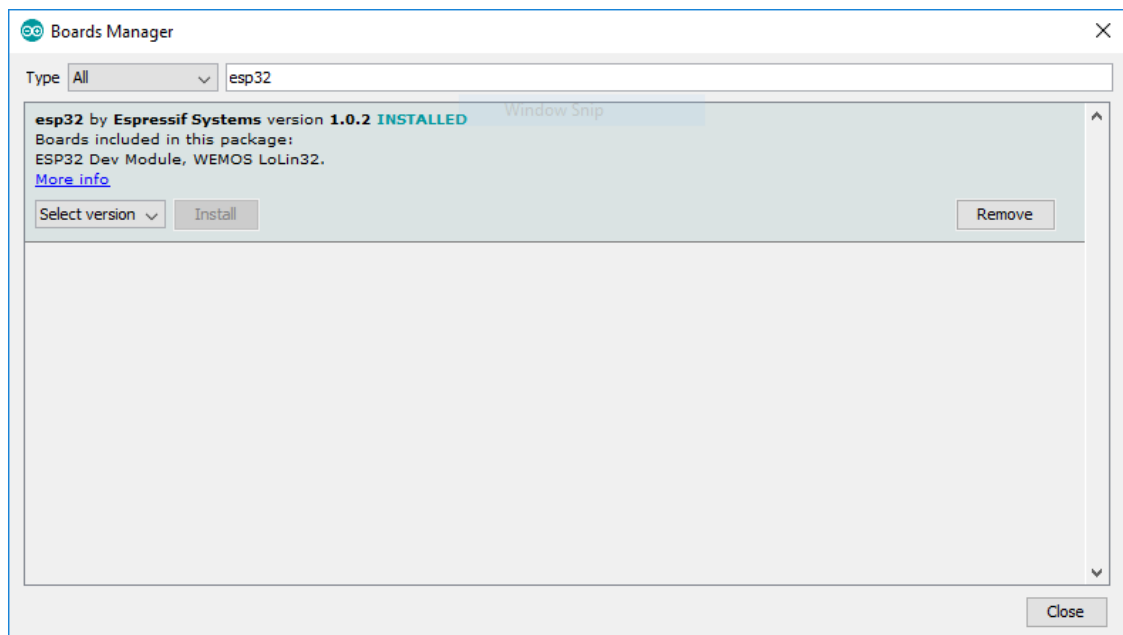


Figure 21. Installation Complete



## 4.2 Testing the Installation

Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

1. Select your Board in **Tools > Board** menu (in my case it's the **DOIT ESP32 DEVKIT V1**)

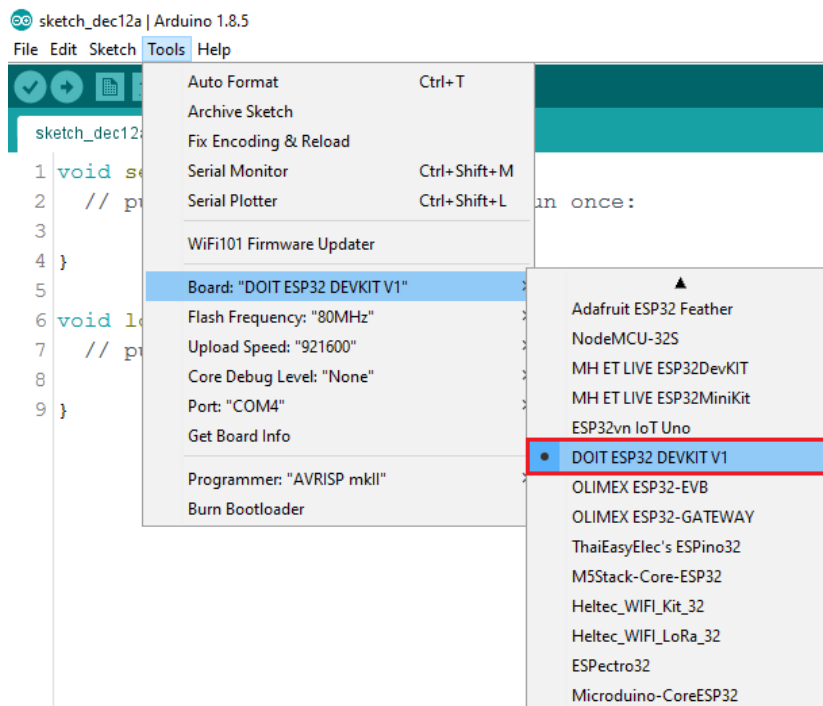


Figure 22. Selecting Board(DOIT ESP32 DEVKIT V1)

2. Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers):

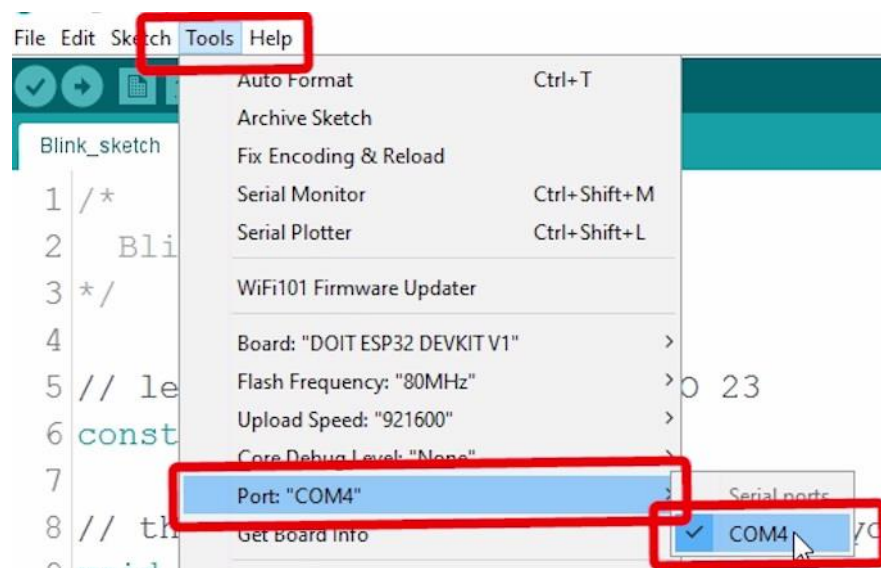


Figure 23. Select the Port

3. Open the following example under **File > Examples > WiFi (ESP32) > WiFiScan**

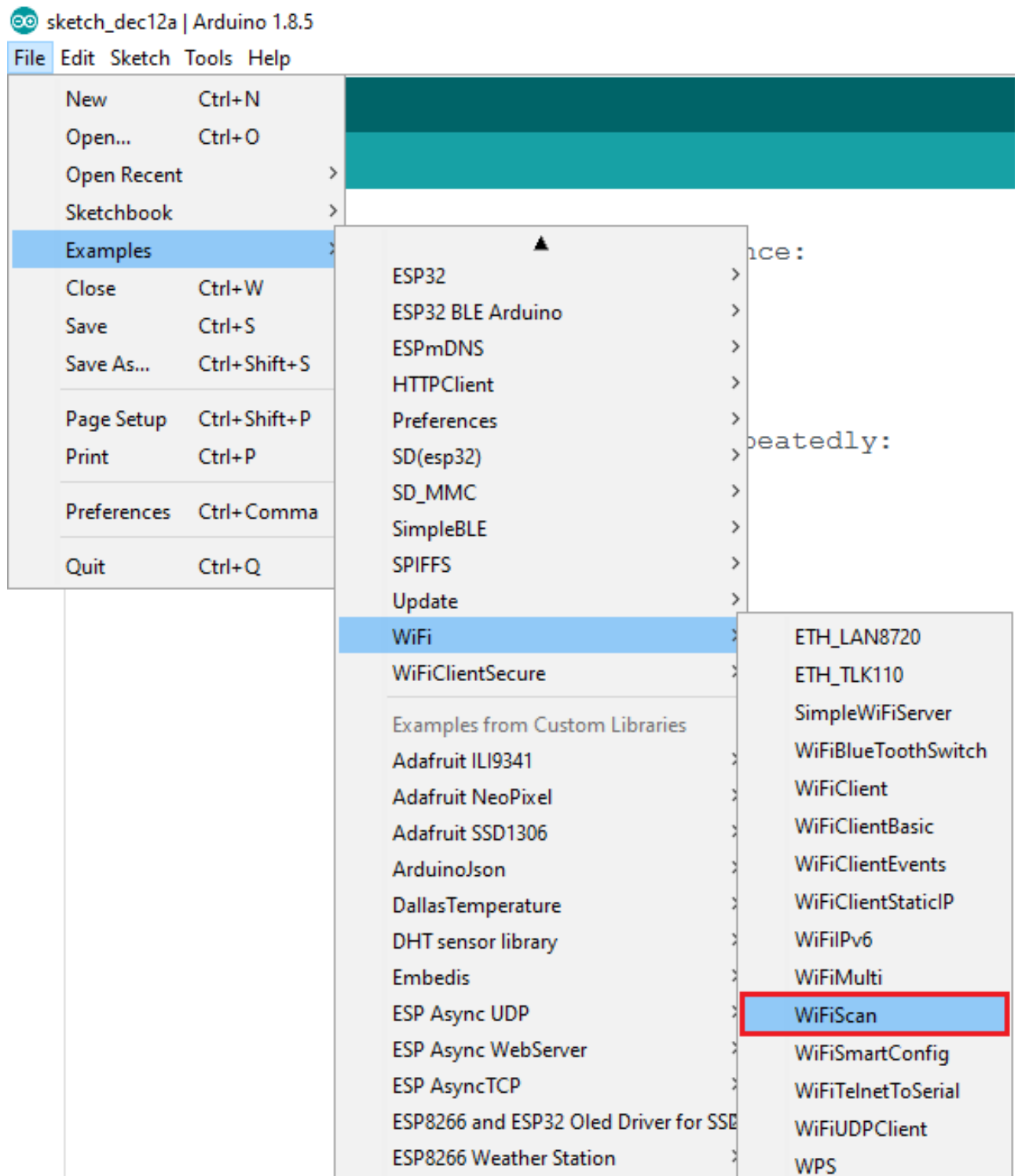


Figure 24. Selecting WiFiScan

4. A new sketch opens in your Arduino IDE:

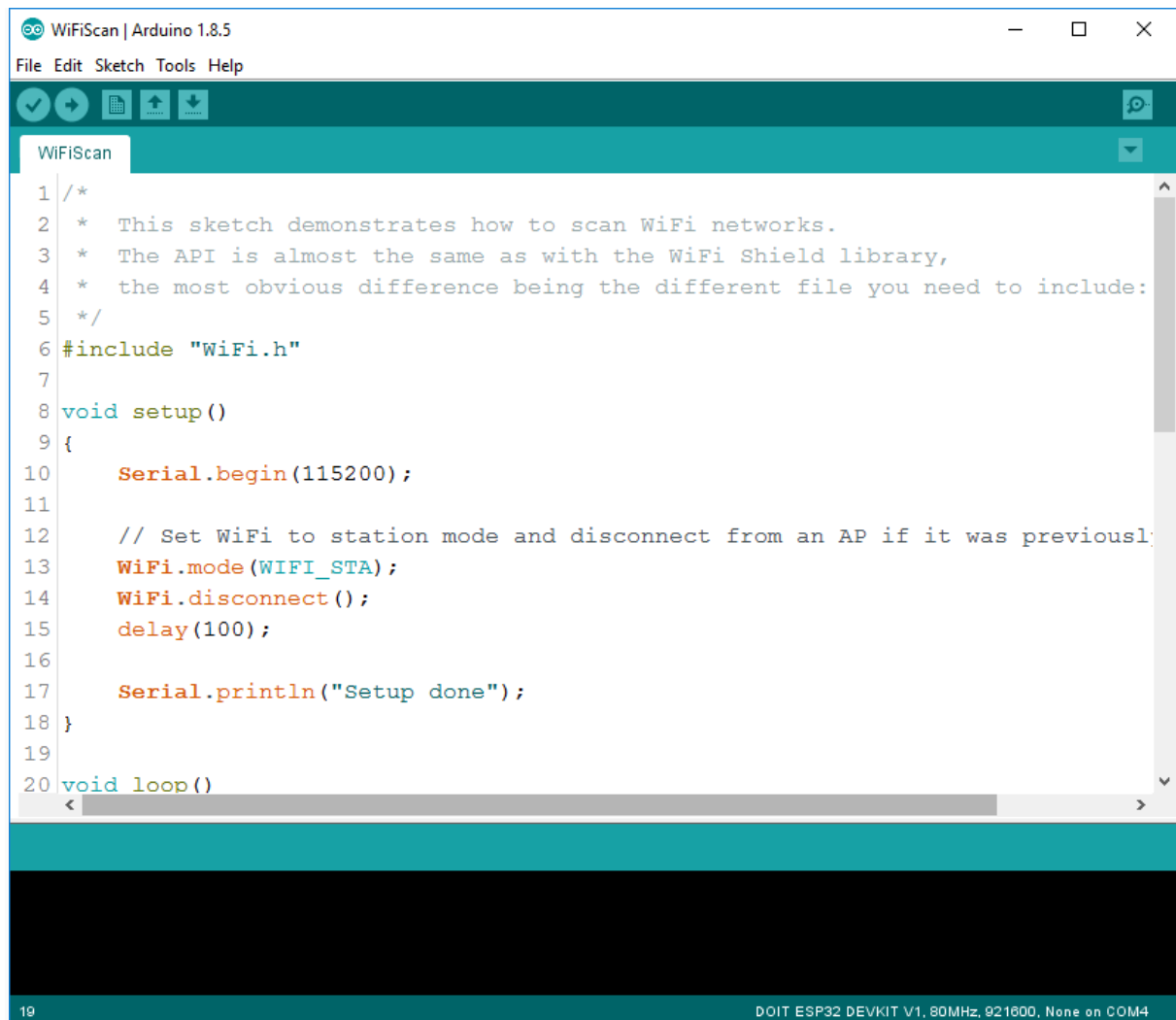
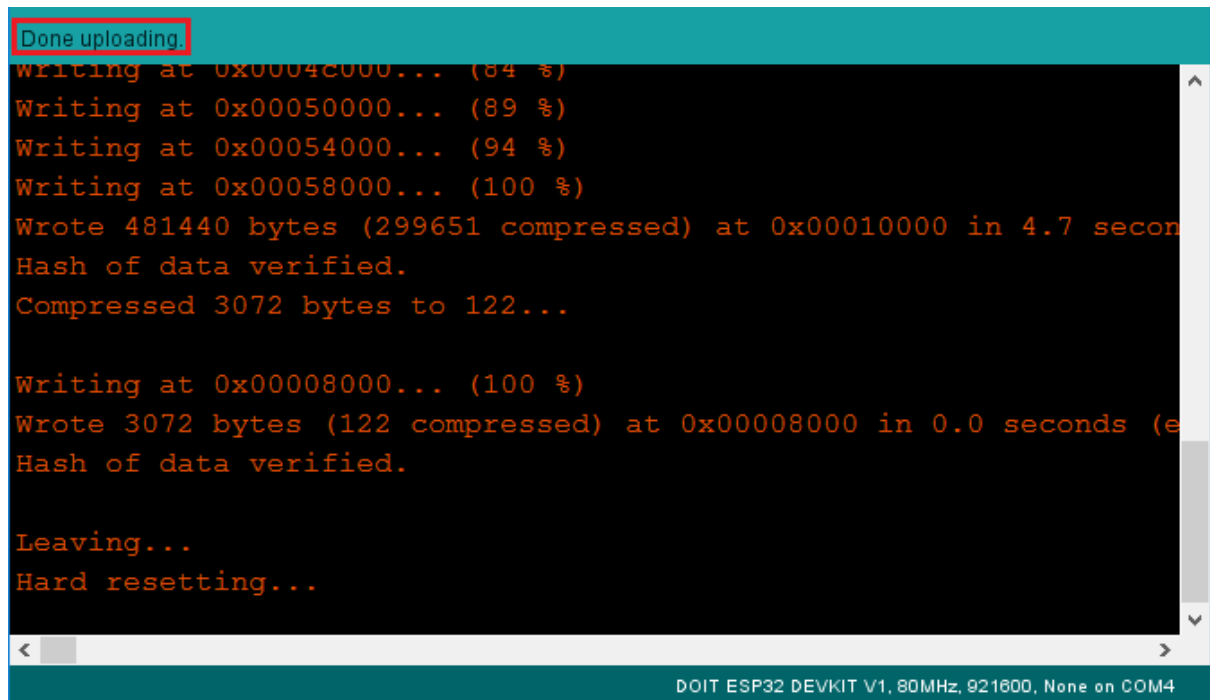


Figure 25. Uploading Code

5. Press the **Upload** button in the Arduino IDE. Wait for few seconds while the code compiles and uploads to the board.



- f) If everything went as expected, we should see a “**Done uploading.**” message.



The screenshot shows a serial monitor window with a black background and orange text. The text indicates the upload process is complete. A red box highlights the text "Done uploading." at the top. The output shows the progress of writing data to memory addresses, the total bytes written (481440), the compressed size (122 bytes), and the time taken (4.7 seconds). It also shows the hash of the data being verified. The process ends with "Leaving..." and "Hard resetting...". The bottom status bar of the window reads "DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM4".

```
Done uploading.  
Writing at 0x0004c000... (84 %)  
Writing at 0x00050000... (89 %)  
Writing at 0x00054000... (94 %)  
Writing at 0x00058000... (100 %)  
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 seconds  
Hash of data verified.  
Compressed 3072 bytes to 122...  
  
Writing at 0x00008000... (100 %)  
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (e  
Hash of data verified.  
  
Leaving...  
Hard resetting...
```

Figure 26. Uploading Done

- g) Open the Arduino IDE Serial Monitor at a baud rate of 115200:
- h) Press the ESP32 on-board Enable button and you should see the networks available near your ESP32:

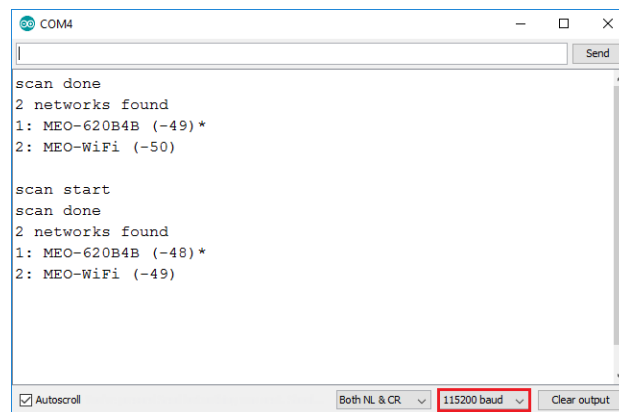


Figure 27. List of Network Available

## CODE

```
//Include the library files
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

//Define the relay pins
#define relay1 D0
#define relay2 D1

#define BLYNK_TEMPLATE_ID "TMPL3v0ZuJLrc"
#define BLYNK_TEMPLATE_NAME "Home Automation System 2"
#define BLYNK_AUTH_TOKEN "4nSTciq9UwXW-FioZIFesjxf3TAEHVpc"

// #define BLYNK_AUTH_TOKEN "" //Enter your blynk auth token

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "demo";//Enter your WIFI name
char pass[] = "12345678";//Enter your WIFI password

//Get the button values
BLYNK_WRITE(V0) {
    bool value1 = param.asInt();
    // Check these values and turn the relay1 ON and OFF
    if (value1 == 1) {
        digitalWrite(relay1, LOW);
    } else {
        digitalWrite(relay1, HIGH);
    }
}

//Get the button values
BLYNK_WRITE(V1) {
```

```

    bool value2 = param.asInt();
    // Check these values and turn the relay2 ON and OFF
    if (value2 == 1) {
        digitalWrite(relay2, LOW);
    } else {
        digitalWrite(relay2, HIGH);
    }
}

void setup() {
    //Set the relay pins as output pins
    pinMode(relay1, OUTPUT);
    pinMode(relay2, OUTPUT);

    // Turn OFF the relay
    digitalWrite(relay1, HIGH);
    digitalWrite(relay2, HIGH);

    //Initialize the Blynk library
    Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

void loop() {
    //Run the Blynk library
    Blynk.run();
}

```

# DESIGN AND IMPLEMENTATION

## 5.1 DESIGN AND IMPLEMENTATION

A low cost and efficient E-HOME system is presented in our design. This system has two modules. The hardware interface and the software communication module. At the heart of this system is the ESP32 microcontroller which is also capable of functioning as a micro web server and the system pass through the microcontroller.

## 5.2 CIRCUIT DIAGRAM

The figure blow show the circuit diagram of our project in this we have connector ESP32 with 4way relay with the help of male-female jumper wire .The 4way relay is then further connected with our home appliances. The home appliances is connected in parallel series with 4 way relay

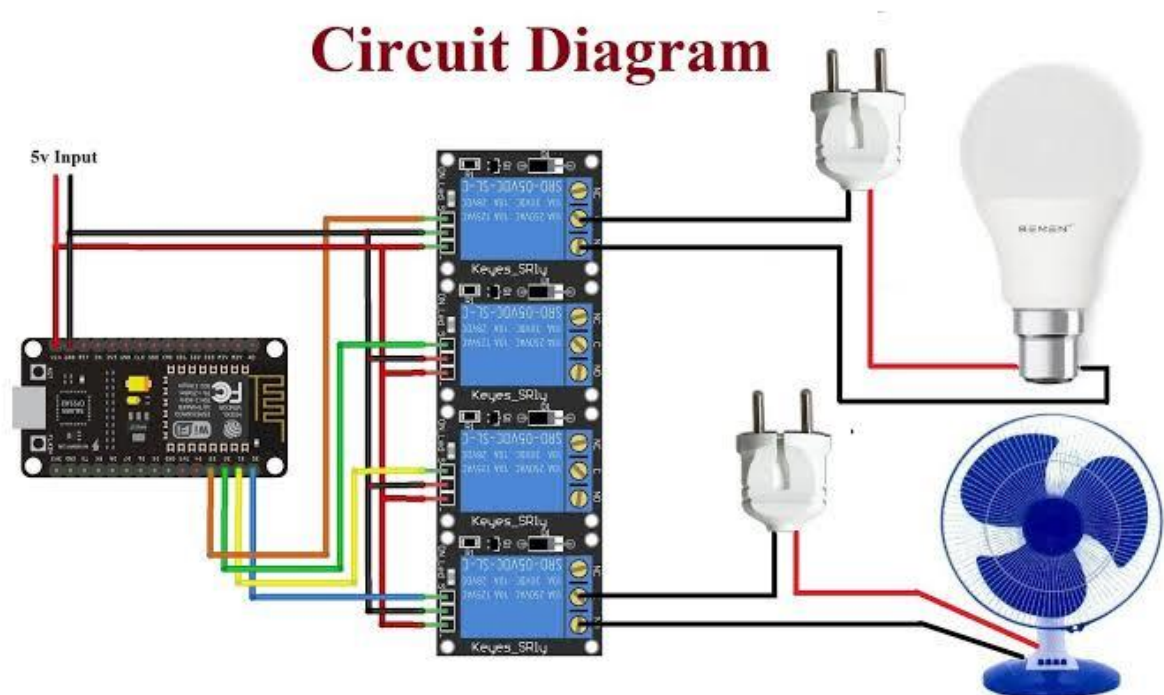


Figure 26.Circuit Diagram

### 5.3 NETWORK DIAGRAM

As IOT is all about internet so we need a network to connect our device for communication and share information . So The figure 27 show the Network Diagram of our model here our smartphone is connected to a different network and send a request to the Arduino which is connected to a router or hotspot .The Arduino receive the request form the Smartphone and the send it to the relay module which work is to ON/OFF the flow off current .So relay module is father connected to the home appliances as the request received by the relay is simply turn the home appliance ON/OFF (based on the request giving by the user) .

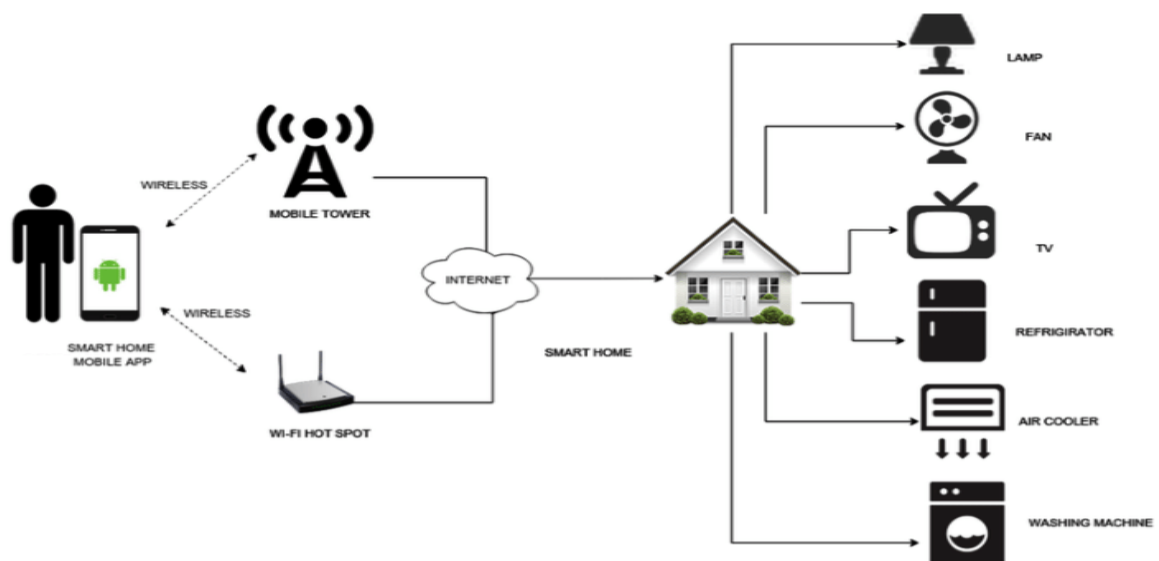


Figure 27.Network Diagram



# Pros of E-HOME

## 6.1 Pros of E-HOME

There are several advantages to implementing an e-home system. Here are some pros of e-home:

- a) **Convenience:** E-home systems provide convenience by allowing homeowners to control and automate various aspects of their homes with ease. Through a centralized control interface, such as a smartphone app or voice command, users can manage lighting, temperature, security systems, entertainment devices, and more, from anywhere inside or outside the home. This convenience saves time and effort by eliminating the need to manually operate individual devices.
- b) **Energy Efficiency:** E-home systems enable energy-efficient management of home appliances and devices. Smart thermostats can optimize heating and cooling based on occupancy and weather conditions, reducing energy waste. Lighting systems can be automated or controlled remotely to turn off when not needed, resulting in energy savings. Smart power outlets can monitor and manage power consumption, helping users identify energy-hungry devices and adjust their usage accordingly.
- c) **Enhanced Security:** E-home systems offer advanced security features for homes. With connected security cameras, motion sensors, and smart door locks, homeowners can monitor their property remotely and receive real-time alerts in case of unauthorized access or suspicious activities. Integration with alarm systems and security service providers further enhances home security.
- d) **Increased Comfort:** E-home systems enhance overall comfort by allowing homeowners to personalize their living environment. Temperature and humidity control can be adjusted remotely or set on schedules, ensuring optimal comfort throughout the day. Motorized window blinds or curtains can be automated to open or close based on time of day or user preferences. Smart audio and entertainment systems can be integrated for seamless music streaming or home theater experiences.
- e) **Remote Monitoring and Management:** E-home systems provide the ability to remotely monitor and manage various aspects of the home. Whether it's checking security camera feeds, adjusting thermostat settings, or receiving notifications about potential issues, homeowners have greater control and awareness even when they are away. This remote

access adds a layer of convenience, peace of mind, and enables timely response to emergencies or maintenance needs.

- f) Integration and Scalability:** E-home systems are designed to be scalable and adaptable to changing needs. They can integrate with a wide range of devices and platforms, allowing homeowners to expand their system as desired. Integration with voice assistants and smart speakers enables voice control, while interoperability with other smart home devices ensures seamless compatibility and expands the possibilities for automation and control.

## Conclusion

Based on the results of analysis of all data obtained by testing the smart home with the Internet of Things based ESP32 ESP6288 module, the following conclusions can be drawn:

1. Smart Home with Internet of Things (IoT) based ESP32 Module can be designed with various components hardware and software support so that it can be arranged into a smart home system that is controlled with the Blynk android application according to what is intended.
2. The Smart Home with this Internet of Things (IoT) based ESP32 Module can be implemented to control some of the home electronics performance including lighting controls, fan control, temperature monitoring, early warning systems and etc.
3. developing a smart switch system using the ESP32 chipset and a two-way relay module offers an exciting and efficient solution for home automation. The ESP32's powerful capabilities, including its dual-core processor and built-in Wi-Fi and Bluetooth connectivity, make it an ideal choice for controlling and managing the smart switch functionality. The two-way relay module allows for seamless control of electrical loads, providing the flexibility to turn devices on or off remotely.
4. By integrating these components, homeowners can experience the convenience of controlling their home's lighting, appliances, and other electrical devices from anywhere using a smartphone or other compatible devices. This brings enhanced convenience, as users no longer need to physically operate switches or worry about forgetting to turn off lights or devices when leaving the house.
5. Moreover, the smart switch system utilizing the ESP32 chipset and two-way relay module promotes energy efficiency by enabling optimized usage of electrical loads. Users can schedule the operation of devices, automate lighting based on occupancy or time of day, and monitor power consumption. This not only reduces energy waste but also contributes to cost savings and a greener, more sustainable living environment.
6. Furthermore, the system enhances security by providing the ability to remotely control and monitor the status of switches and electrical devices. Homeowners can ensure that lights are turned on/off at specified times, creating an illusion of occupancy to deter potential intruders. Additionally, the system can be integrated with other security features such as cameras and sensors, allowing users to receive real-time notifications and take immediate action in the event of suspicious activities.

## Suggestions

In the design and manufacture of this final project there are still deficiencies that need to be corrected in order to perfect this final project, including:

1. Optimizing the power control consumption of the ESP32 module to be further developed in wireless-based technology application, considering the current technology prioritizes low cost but efficient.
2. The development of an internet-based smart home system of things needs to be tested on other electronic devices in everyday life.
3. Choose a suitable ESP32 development board that meets your project requirements. Consider factors such as available I/O pins, onboard peripherals, power requirements, and form factor. Popular options include the ESP32 NodeMCU, ESP32 DevKit, or ESP32-WROOM-32.
4. Test the system thoroughly to ensure proper functionality and reliability. Verify the communication between the ESP32 and the relay module, check the responsiveness of the switches, and test different load scenarios. Monitor for any issues or bugs and debug as necessary.
5. Once the system is tested and working correctly, install the smart wireless switches in the desired locations in your home or building. Ensure proper electrical connections and safety measures are in place. Deploy the user interface to the desired devices (smartphones, tablets, etc.) for remote access.

## REFERENCES

1. Arduino Temperature Sensor Using LM35. Groups of Electronics Hobbyist, Roboticist. We Developed Electronics Project Tutorials Make Open for Everyone.
2. “BOHORA”, “Bharat; MAHARJAN”, “Sunil; SHRESTHA”, “Bibek Raj”. IoT Based Smart Home by Using the Blynk Framework. “Zerone Scholar”, [S.l.], v. 1, n. 1, p. 26-30, dec. 2022. ISSN 2542- 2774. google scholar.
3. DC-DC Step Down Converter Power Supply Provides Regulated 5VDC Output with Range Input of 10-32VDC, Model GTD21088L-1505-T2.
4. E-HOME Using Internet of Thing 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) Published: 2016. Google Scholar.
5. Internet of Things in E-HOME and Energy Efficient Smart Home Technologies Simon G. M. Koo Department of Computer Engineering, Santa Clara University, CA 95053, USA
6. Low Cost Implementation of Smart E-HOME Ravi Kishore Kodali, “Department of Electronics and Communication Engineering” NIT, Warangal , 506004 India
7. Mobile based E-HOME using Internet of Things (IoT) 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) Published: 2015
8. ESP32 Features and Pinout. A Brief Tutorial on the Introduction to ESP32 V3.
9. Yoyosteven in Circuits Microcontrollers. ESP32 1.0 (ESP 32) CONTROLLED RELAY USING BLYNK (OVER THE WEB).
10. 5V 4-Channel Relay Interface Board, Standard Interface that can be Controlled Directly by