Google Assistant Based Home Automation

A Project report submitted in partial fulfillment
of the requirements for the degree of B. Tech in Electrical Engineering

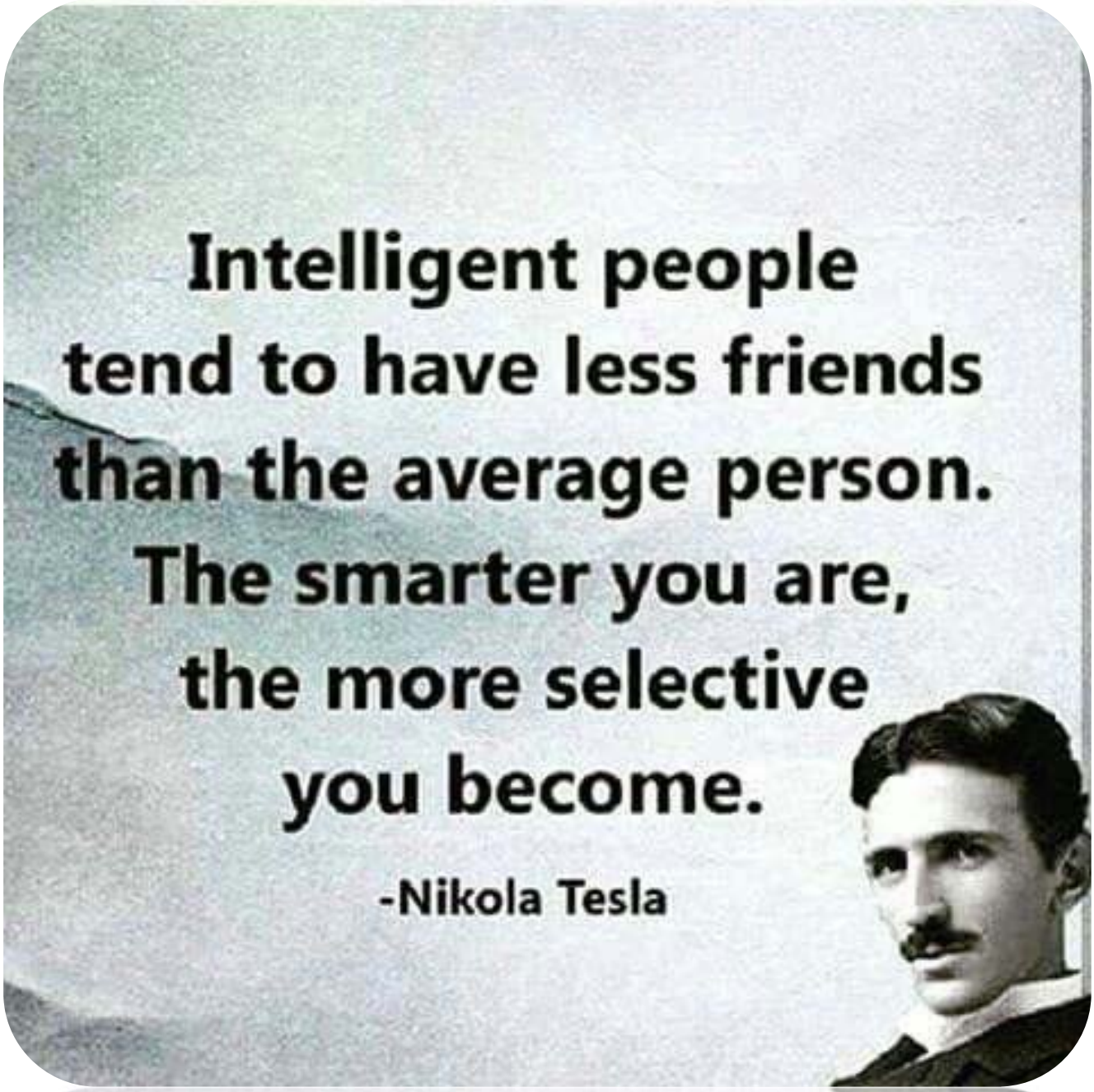
by

TUHINAA DEY (11701619038)
ANISH KOLEY (11701619017)
SUBHRADEB MONDAL (11701619020)

Under the supervision of

Mr. Budhaditya Biswas
Assistant Professor
Department of Electrical Engineering

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
Maulana Abul Kalam Azad University of Technology (MAKAUT)


Intelligent people
tend to have less friends
than the average person.
The smarter you are,
the more selective
you become.
-Nikola Tesla

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
PHONE: 033-2323-2463-154, FAX: 033-2323-4668
Email: hodeercciit@gmail.com, Website: http://www.rcciit.org/academic/ee.aspx

CERTIFICATE

To whom it may concern

This is to certify that the project work entitled **Google Assistant Based Home Automation** is the bonafide work carried out by **TUHINAA DEY (11701619038), ANISH KOLEY (11701619017), SUBHRADEB MONDAL (11701619020)** the students of B.Tech in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year 2022-23, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and that this project has not submitted previously for the award of any other degree, diploma and fellowship.

Budhaditya Biswas

(Budhaditya Biswas)
Assistant Professor
Department of Electrical Engineering
RCC Institute of Information Technology

Countersigned by

(Dr. Shilpi Bhattacharya)
HOD, Electrical Engineering Dept
RCC Institute of Information Technology

ACKNOWLEDGEMENT

It is our great fortune that we have got opportunity to carry out this project work under the supervision of **Mr. Budhaditya Biswas** in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (**MAKAUT**), West Bengal, India. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance and limitless encouragement.

We would also like to convey our gratitude to all the faculty members and staffs of the Department of Electrical Engineering, RCCIIT for their whole hearted cooperation to make this work turn into reality.

We are very thankful to our department and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research work.

Thanks to the fellow members of our group for working as a team.

TUHINAA DEY (11701619038)
ANISH KOLEY (11701619017)
SUBHRADEB MONDAL (11701619020)

To

The Head of the Department
Department of Electrical Engineering
RCC Institute of Information Technology
Canal South Rd. Beliaghata, Kolkata-700015

Respected Sir,

In accordance with the requirements of the degree of Bachelor of Technology in the Department of Electrical Engineering, RCC Institute of Information Technology, we present the following thesis entitled "**Google Assistant Based Home Automation**". This work was performed under the valuable guidance of Mr. Budhaditya Biswas, Assistant Professor in the Dept. of Electrical Engineering.

We declare that the thesis submitted is our own, expected as acknowledge in the test and reference and has not been previously submitted for a degree in any other Institution.

Yours Sincerely,

TUHINAA DEY (11701619038) Tuhinaa Dey

ANISH KOLEY (11701619017) Anish Koley

SUBHRADEB MONDAL (11701619020) Subhradeb Mondal

Contents

List of Figures

List of Tables

ABBREVIATIONS AND ACRONYMS

GA – *Google Assistant*
IoT – *Internet of Things*
IFTTT– If This Then That
IC - Integrated Circuit
PCB – Printed Circuit Board
µC – Micro Controller
BJT - Bi-polar Junction Transistor
SPDT - Single Pole Double Throw
NO - Normally Open
NC - Normally Closed
COM – Common
OLED – Organic Light Emitting Diode
LED - Light Emitting Diode
SMPS – Switch Mode Power Supply
ISM – Industrial, scientific and medical
USB – Universal serial bus
SPI – Serial Peripheral Interface
I^2C – Inter-Integrated Circuit
TXD – Transmitter
RXD – Receiver
GND – Ground

ABSTRACT

Nowadays Technology keeps on upgrading. The idea behind Google assistant-controlled home automation is to control home devices with voice. In the market there are many devices available to do that, but making our own is awesome. In this project, the Google assistant requires voice commands. BLYNK account which is a cloud based free IoT web server used to create virtual switches, is linking to IFTTT website abbreviated as "If This Than That" which is used to create if else conditional statements. The voice commands for Google assistant have been added through IFTTT website. In this home automation, as the user gives commands to the Google assistant, home appliances like Bulb, Fan and Motor etc., can be controlled accordingly. The commands given through the Google assistant are decoded and then sent to the microcontroller, the microcontroller in turn control the relays connected to it. The device connected to the respective relay can be turned ON or OFF as per the users request to the Google Assistant. The microcontroller used is ESP32 and the communication between the microcontroller and the application is established via Wi-Fi (Internet).

The load status is showing in a 0.96" OLED inbuilt in the circuit. The whole system is power up through a 5V 1A adapter.

CHAPTER 1

(Introduction)

INTRODUCTION

The aim of the proposed system is to develop a cost-effective solution that will provide controlling of home appliances remotely and enable home security against intrusion in the absence of homeowner. The system provides availability due to development of a low-cost system. The home appliances control system with an affordable cost was thought to be built that should be mobile providing remote access to the appliances and allowing home security. Though devices connected as home and office appliances consume electrical power. These devices should be controlled as well as turn on/off if required. Most of the times it was done manually. Now it is a necessity to control devices more effectively and efficiently at anytime from anywhere.

In this system, we developed a Google Assistant voice-controlled home/office appliance. This system is designed for controlling arbitrary devices, it receives the voice command through android mobile using google assistant to control the electrical load through a relay. The ESP32 microcontroller connected to the internet through the wifi. The microcontroller also connected to the blynk cloud. When a voice command is initiated through the android mobile, a API command is also generated through IFTTT cloud service provider. The API command communicate to the blynk cloud and blynk.cloud communicate to the device. The relay is switch ON/OFF based on the API command and control the electrical load connected. The load status is showing in a 0.96" OLED inbuilt in the circuit. The whole system is power up through a 5V 1A adapter.

1.1 HOME AUTOMATION

Home automation is the residential extension of building automation. It is automation of the home, housework or household activity. Home automation may include centralized

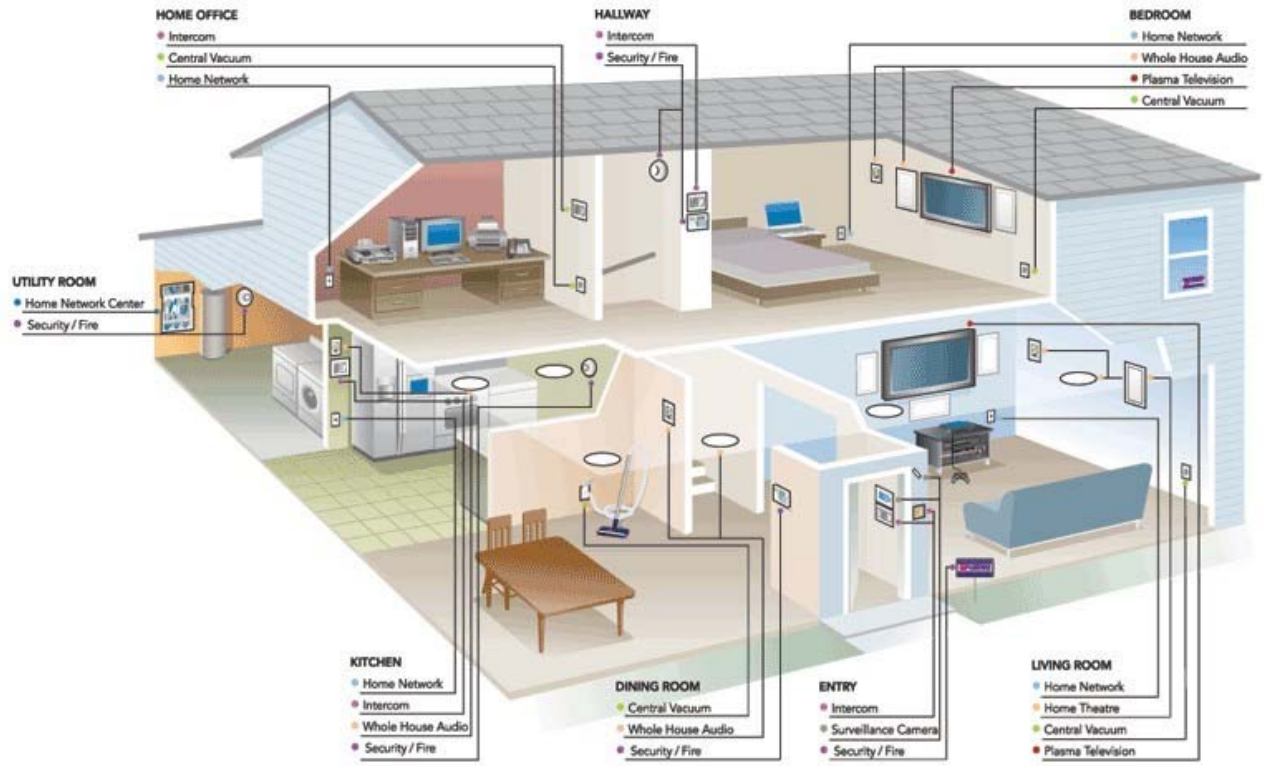


Figure 1: Concept of Home Automation

control of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security. Home automation for the elderly and disabled can provide increased quality of life for persons who might otherwise require caregivers or institutional care.

The popularity of home automation has been increasing greatly in recent years due to much higher affordability and simplicity through Smartphone and tablet connectivity. The concept of the "Internet of Things" has tied in closely with the popularization of home automation.

A home automation system integrates electrical devices in a house with each other. The techniques employed in home automation include those in building automation as well as the control of domestic activities, such as home entertainment systems, houseplant and yard watering, pet feeding, changing the ambiance "scenes" for different events (such as dinners or parties), lighting control system, and the use of domestic robots. Devices may be connected through a home network to allow control by a personal computer, and may allow remote access from the internet. Through the integration of information technologies with the home environment, systems and appliances can communicate in an integrated manner which results in convenience, energy efficiency, and safety benefits.

Automated "homes of the future" have been staple exhibits for World's Fairs and popular backgrounds in science fiction. However, problems with complexity, competition between vendors, multiple incompatible standards and the resulting expense have limited the penetration of home automation to homes of the wealth, or ambitious hobbyists. Possibly the first "home computer" was an experimental home automation system in 1966.

1.2 Overview and benefits

Home automation refers to the use of computer and information technology to control home appliances and features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/microcontroller-based networks with varying degrees of intelligence and automation. Home automation is adopted for reasons of ease, security and energy efficiency.

In modern construction in industrialized nations, most homes have been wired for electrical power, telephones, TV outlets (cable or antenna), and a doorbell. Many household tasks were automated by the development of specialized automated appliances. For instance, automatic washing machines were developed to reduce the manual labor of cleaning clothes, and water heaters reduced the labor necessary for bathing.

The use of gaseous or liquid fuels, and later the use of electricity enabled increased automation in heating, reducing the labor necessary to manually refuel heaters and stoves. Development of thermostats allowed more automated control of heating, and later cooling.

As the number of controllable devices in the home rises, interconnection and communication becomes a useful and desirable feature. For example, a furnace can send an alert message when it needs cleaning or a refrigerator when it needs service. If no one is supposed to be home and the alarm system is set, the home automation system could call the owner, or the neighbors, or an emergency number if an intruder is detected.

In simple installations, automation may be as straightforward as turning on the lights when a person enters the room. In advanced installations, rooms can sense not only the presence of a person inside but know who that person is and perhaps set appropriate lighting, temperature,

music levels or television channels, taking into account the day of the week, the time of day, and other factors.

Other automated tasks may include reduced setting of the heating or air conditioning when the house is unoccupied, and restoring the normal setting when an occupant is about to return. More sophisticated systems can maintain an inventory of products, recording their usage through bar codes, or an RFID tag, and prepare a shopping list or even automatically order replacements.

Home automation can also provide a remote interface to home appliances or the automation system itself, to provide control and monitoring on a Smartphone or web browser.

An example of remote monitoring in home automation could be triggered when a smoke detector detects a fire or smoke condition, causing all lights in the house to blink to alert any occupants of the house to the possible emergency. If the house is equipped with a home theater, a home automation system can shut down all audio and video components to avoid distractions, or make an audible announcement. The system could also call the home owner on their mobile phone to alert them, or call the fire department or alarm monitoring company.

1.3 Organisation of thesis

The thesis is organised into five chapters including the chapter of introduction. Each chapter is different from the other and is described along with the necessary theory required to comprehend it.

Chapter 2 deals with the literature reviews. From this chapter we can see before our project who else works on this topic and how our project is different and advance from those projects.

Chapter 3 deals with the theory required to do the project. The basic of Google Assistant, connect ESP32 with Blynk cloud, activate the Google Home to works with the voice command are described here. The overview of the project and software simulation of the project is also listed in this chapter.

Chapter 4 deals with the hardware modelling of the projects. The main features, photographs, step by step operation of the prototype, component listing and the hardware interfacing of the required components are described here.

Chapter 5 describes the basic operation of the circuit. A flow chart is presented on the actions that would take in the controller beginning from the voice command through google assistant in android device to generatee API command to the switching on and off of loads. Advantages and disadvantages and cost estimation are listed in this chapter.

Chapter 6 concludes the work performed so far. The possible limitations in proceeding research towards this work are discussed. The future work that can be done in improving the current scenario is mentioned. The future potential along the lines of this work is also discussed.

Chapter 7 References are listed in this chapter

Appendix A, B & C Hardware description, software coding and datasheets are listed here.

CHAPTER 2

(Literature Review)

[1] Md Sarwar Kamal in (2017) “Efficient low-cost supervisory system for Internet of Things enabled smart home.” This paper proposes an efficient low-cost supervisory system for smart home automation that can be managed using IoT. The proposed system is based on Apriority algorithm and will help to monitor and control all the home appliances and electronic devices through a supervisory system in a most efficient and reliable manner. Both the consumers and the suppliers will get the opportunity to manage the power distribution by monitoring the electricity consumption.

[2] Aayush Agarwal, Anshul Sharma, Asim Saket Samad and S Babeetha (2018) “UJALA-Home Automation System Using Google Assistant” This project presents a design and prototype of Home Automation system that will use ESP8266 Wi-Fi module as a network provider in connecting with other appliances. Further we will connect the specific home to our database and it can be accessed from anywhere through a specific IP address or website. Also, an app would be developed which will allow the user to control their devices using the Google Assistant.

[3] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, Subhas Chandra Mukhopadhyay (2013) “Towards the Implementation of IoT for Environmental Condition Monitoring in Homes” In this paper, we have reported an effective implementation for Internet of Things used for monitoring regular domestic conditions by means of low cost ubiquitous sensing system. The description about the integrated network architecture and the interconnecting mechanisms for the reliable measurement of parameters by smart sensors and transmission of data via internet is being presented. The longitudinal learning system was able to provide a self-control mechanism for better operation of the devices in monitoring stage. The framework of the monitoring system is based on a combination of pervasive distributed sensing units, information system for data aggregation, and reasoning and context awareness. Results are encouraging as the reliability of sensing information transmission through the proposed integrated network architecture is 97%. The prototype was tested to generate real-time graphical information rather than a test bed scenario.

[4] Sarthak Jain, Anant Vaibhav, Lovely Goyal in, explained the system that can be used to control home appliances by reading the commands the subject of an email received to the specifically programmed email address of the device.

[5] Ana Marie D. Celebre in, proposed a system that uses the Siri technology powered by Apple Inc., to control the system using in built voice commands provided with Siri. They used an unsupported server to get the functionality of Siri.

[6] Nikhil Rathod, Dr. P. D. Paikrao explained the proposed working of A Survey on Home Automation by Using Voice Command Based on IOT. It will increase the efficiency of this application. We control the entire home appliance over the internet.

[7] Jawarkar, Ahmed, Ladhake, and Thakare (2008) propose remote monitoring through mobile phone involving the use of spoken commands. The spoken commands are generated and sent in the form of text SMS to the control system and then the microcontroller on the basis of SMS takes a decision of a particular task.

[8] S. Meera, M. Ramya, L. Megala has designed Smart Hospitals Using Internet of Things (IoT). In this project, Smart hospital using Internet of Things (IoT) has been successfully designed. This project is highly energy efficient as it uses arduino board having microcontroller (AT mega Atmel 328PU) which having low power utilization. It also uses MQTT networking protocol which is a light weight protocol and helps in power saving. We do not need to manually turn ON or turn OFF the switch of the light. It is possible to control the switch from a webpage or from the mobile application. This system is a time consuming. It will save patient from the risk of "AIR EMBOLISM". It is user friendly system. Maintenance of this project is not costly.

[9] Ms. Preeti U. Melikatti, the maximum extensively proposed of numerous domestic automation gadgets confirmed that there are a range of technologies used to place into practice this type of structure. All the planned systems have been obtainable and compared in this paper which reveal some qualities and demerits of the gadget. This assessment explain delivers home automation gadget e.g. Bluetooth-based, Web based, mobile-based, ZigBee-based, SMS based, Adriano microcontroller based, Android app based, IOT based and cloud-based. Due to its recital, ease, low cost and dependability domestic automation gadget is making its place in international market.

[10] Nikhil Singh, Shambhu Shankar Bharti, Rupal Singh, Dushyant Kumar Singh (2014) "Remotely controlled home automation system", Advances in Engineering and Technology Research (ICAETR) This paper describes an investigation into the potential for remote controlled operation of home automation systems. It considers problems with their implementation, discusses possible solutions through various network technologies and indicates how to optimize the use of such systems. The home is an eternal, heterogeneous, distributed computing environment (Greaves, 2002) which certainly requires a careful study before developing any suitable Home Automation System (HAS) that will accomplish its requirements. Nevertheless, the latest attempts at introducing Home Automation Systems in actual homes for all kinds of users are starting to be successful thanks to the continuous standardization process that is lowering the prices and making devices more useful and easier to use for the end user. Even so several important issues are always to be handled strictly before developing and installing a Home Automation System; factors like security, reliability, usefulness, robustness and price are critical to determine if the final product will accomplish the expected requirements.

.

CHAPTER 3

(Theory)

3.1 Google Assistant

Google Assistant is a virtual assistant software application developed by Google that is primarily available on mobile and home automation devices. Based on artificial intelligence, Google Assistant can engage in two-way conversations, unlike the company's previous virtual assistant, Google Now.

Figure 2: Google Assistant application in mobile

Google Assistant debuted in May 2016 as part of Google's messaging app Allo, and its voice-activated speaker Google Home. After a period of exclusivity on the Pixel and Pixel XL smartphones, it was deployed on other Android devices starting in February 2017, including third-party smartphones and Android Wear (now Wear OS), and was released as a standalone app on the iOS operating system in May 2017. Alongside the announcement of a software development kit in April 2017, Assistant has been further extended to support a large variety of devices, including cars and third-party smart home appliances. The functionality of the Assistant can also be enhanced by third-party developers.

Users primarily interact with the Google Assistant through natural voice, though keyboard input is also supported. Assistant is able to answer questions, schedule events and alarms, adjust hardware settings on the user's device, show information from the user's Google account, play games, and more. Google has also announced that Assistant will be able to identify objects and gather visual information through the device's camera, and support purchasing products and sending money.

At CES 2018, the first Assistant-powered smart displays (smart speakers with video screens) were announced, with the first one being released in July 2018. In 2020, Google Assistant is already available on more than 1 billion devices. Google Assistant is available in more than 90 countries and in over 30 languages, and is used by more than 500 million users monthly.

3.1.1 What can Google Assistant do?

Google Assistant offers voice commands, voice searching, and voice-activated device control, letting you complete a number of tasks after you've said the "OK Google" or "Hey Google" wake words. It is designed to give you conversational interactions.

Google Assistant will:

- Control your devices and your smart home
- Access information from your calendars and other personal information
- Find information online, from restaurant bookings to directions, weather and news
- Control your music

- Play content on your Chromecast or other compatible devices
- Run timers and reminders
- Make appointments and send messages
- Open apps on your phone
- Read your notifications to you
- Real-time spoken translations
- Play games

Continued Conversation means you don't have to say "Hey Google" for follow-up requests. Instead, once you've started talking to Google, it listens for a response without needing a trigger phrase all the time. Google can also recognise voice profiles for different people, so it knows who is talking to it and can tailor the responses accordingly. You can also ask for multiple things at the same time.

As Google Assistant knows you and understands context, it will react in an informed or smart way. That's important as it gives voice control a lot more power and moves it on from only reacting to specific phrases or commands. It's designed to be more than just reactive.

3.1.2 Which devices offer Google Assistant?

Google Assistant originally launched on the **Google Pixel smartphones** and **Google Home**, but it is now available to just about all modern Android devices, including **Wear OS devices**, Android TV, and **Nvidia Shield**, as well as any cars that support **Android Auto** and other devices too, like Nest cameras and the Lenovo Smart Clock.

Google Assistant is native to **Google Nest** (formerly Google Home) smart speakers, but it's also widely available on **other smart speakers** from third-party manufacturers including Sony, Sonos, LG and Panasonic. Similarly, it's widely supported by headphones (when connected to an Android phone).

Smart home devices like **Philips Hue** and **Ikea's Home Smart range**, for example, can be controlled by Google Assistant and not just through Google Nest, but wherever you happen to interact with Assistant.

3.1.3 IFTTT IoT Cloud service

IFTTT is a free web service and mobile app that helps users automate web-based tasks and boost productivity by making popular apps work together. IFTTT stands for “If This Then That,” an homage to the programming conditional statement. Using formulas called “recipes,” users can dictate task automations, so if something happens in one app, the event triggers an action in another app. For instance, you can set up an automation so that if you share a photo on Facebook, it triggers the action of automatically posting that photo to Twitter, Instagram, Flickr and other photo-sharing services.

- IFTTT stands for “If This Then That.” It’s a free web service that helps users automate web-based tasks and improve productivity.
- IFTTT connects various developers’ devices, services and apps to create “applets” that perform automations.

- The service is incredibly easy to use and includes guides for setting up myriad specific automations.

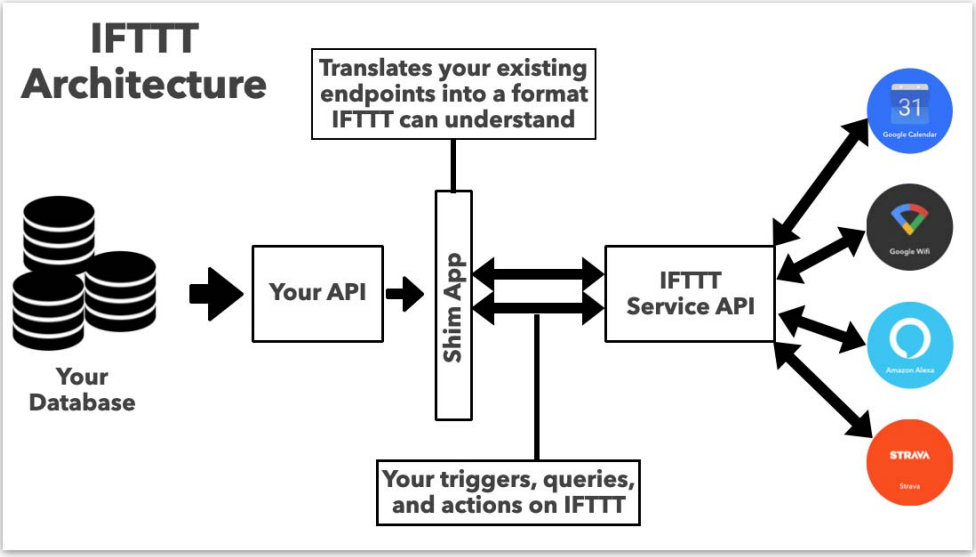


Figure 3: IFTTT architecture

3.2 Overview of the project

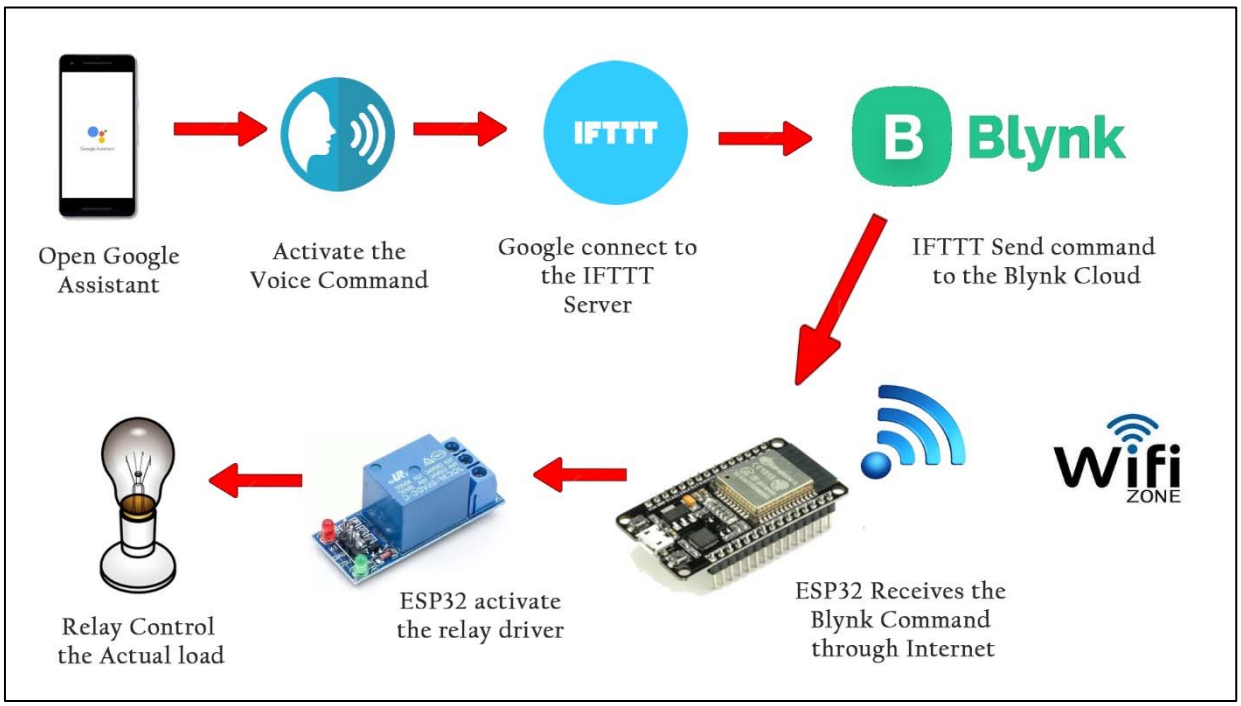


Figure 4: Overview of the complete project

According to fig. No. 4 i.e. overview of the project, first we need to place the ESP32 in the wifi zone and make sure that the microcontroller must be connected to the wifi. Open the Google Assistant application in the android mobile. Activate the pre-specified voice command through google assistant. The assistant will communicate to the IFTTT server and IFTTT will generate the blynk API command. Once the API command will be activated the Blynk server communicate with the ESP32 through internet. Receiving the signal from the Blynk server ESP32 send ON/OFF command to the relay driver. The relay activates the actual load based on the command received from the controller. The real time status of the load will be display over a 0.96 inch colour OLED screen which is install on the

prototype. The small buzzer also generates some audio feedback during load switching to ensure the load is turn on or off properly.

3.3 Circuit Diagram

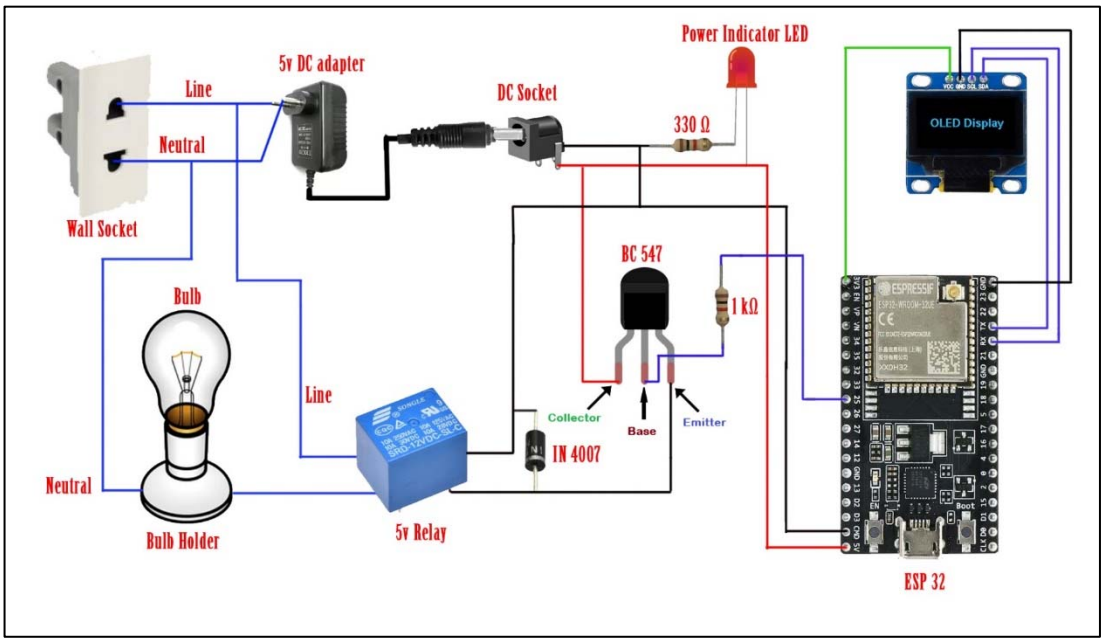


Figure 5: Complete circuit Diagram of the Project

The complete circuit is first simulated in the Wokwi simulator, the screenshot of the Wokwi simulation is shown in figure 6. The circuit is running well in this software. The software simulation includes microcontroller (ESP32), OLED and LED. Here we incorporate 1 load which can be controlled as per user choice. The load status is showing in a 0.96" OLED inbuilt in the circuit. The whole system is power up through a 5V 1A adapter.

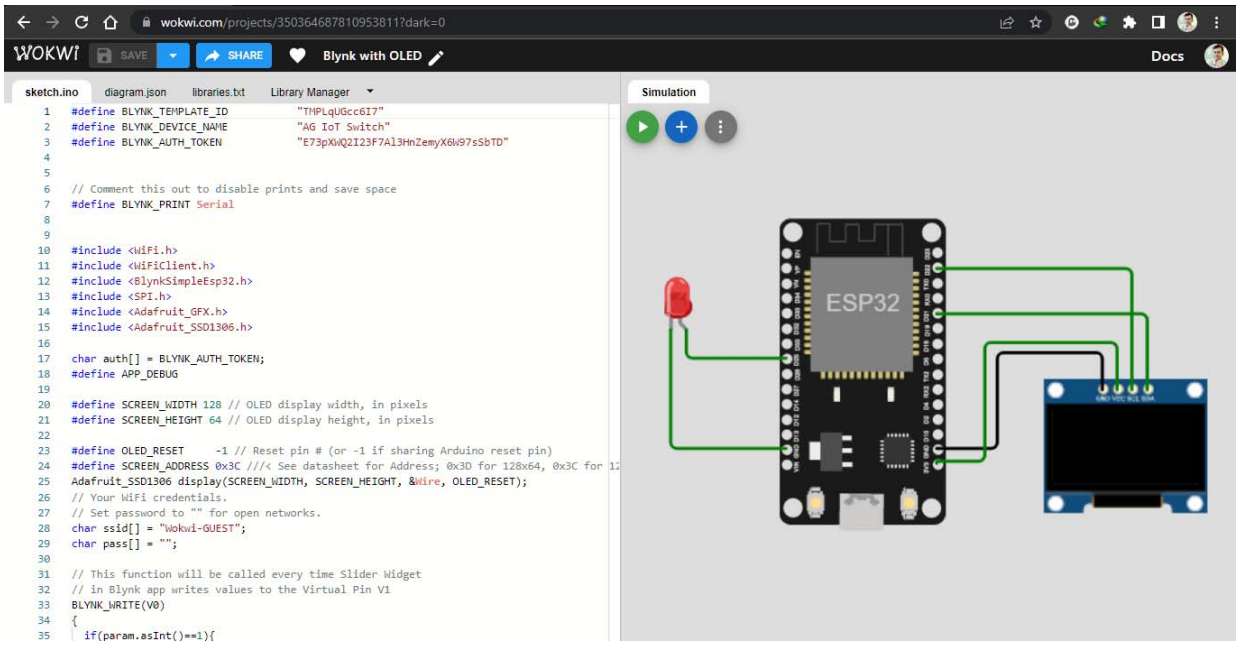


Figure 6: Simulation of the Project in Wokwi online simulator

CHAPTER 4

(Hardware Modeling)

4.1 Main features of the prototype

The features of the developed prototype are:

- Secure (The API command is generated through a particular mobile)
- OLED display (showing the condition of the load status and the status of the circuit)
- 1 electrical load controls (250 volt, 7 amp max, ON/OFF control)
- Inbuilt relay driver
- Buzzer indication during load switching
- Power indication LEDs
- 5 Volt operation (both control board and relay board)

4.2 Photographs of the prototype

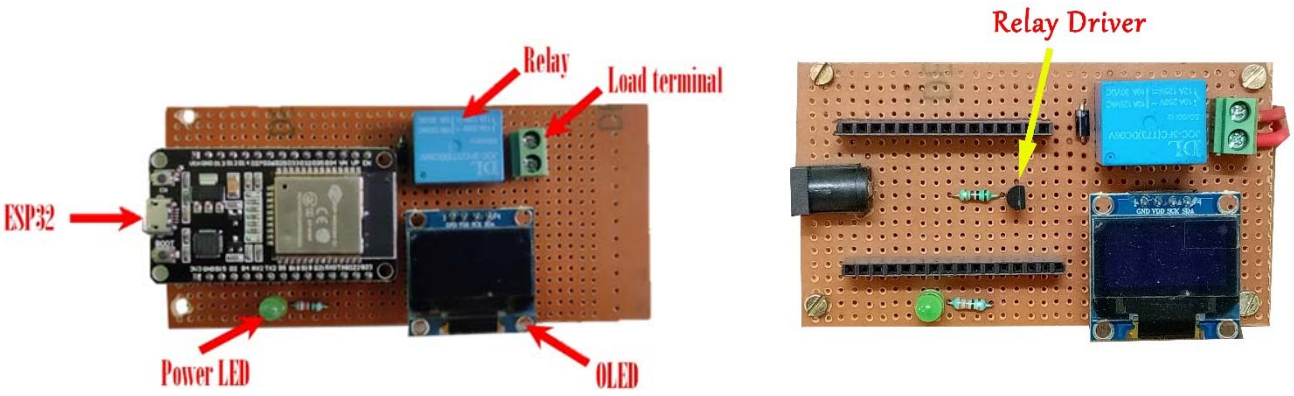


Figure 7: Main prototype with OLED and relay driver

4.3 Step by step operation of the prototype

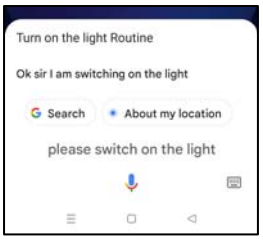
1. Connect the DC adapter (5V, 1A) to the DC jack.

2. Power On the circuit
3. Connect the wifi and make sure that the device is connected to the wifi.
4. Open the google assistant in the android mobile to initiate the voice command

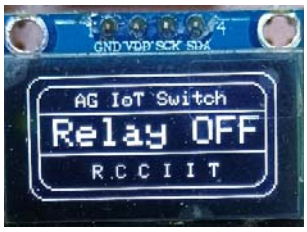
5. The Assistant will communicate with IFTTT and IFTTT will communicate to the blynk cloud to initiate the ON/OFF API command

6. Switch on the load by saying a particular pre-define phase like “Please switch ON the Light”.

7. Switch off the load by saying a particular pre-define phase like “Please switch OFF the Light”.



8. The status of the load will be seen in the OLED screen.



4.4 Components required

Sl. No.	Components	Quantity
1	ESP32 Microcontroller	1
2	330 Ω resistance	1
3	OLED Display	1
4	5 volt static Relay	1
5	Female pin header	1
6	Piezo Buzzer	1
7	Single stand wire	3m
8	Wire nipper	1
9	Wire striper	1
10	Soldering Iron	1
11	Soldering material	1
12	De-soldering pump	1
13	BC 547	1
14	DC Socket	1
15	IN 4007	1
16	Bulb Holder	1
17	Bulb	1

Table 1: Component listing

4.5 Hardware interfacing

4.5.1 ESP32 OLED Display with Arduino IDE

The combination of OLED with ESP32 is so popular that there are some boards of ESP32 with the OLED integrated. We'll, however, assume that you will be using a separate OLED module with your ESP32 board. If you have an OLED module, it perhaps looks like the image below.

Figure 8: 0.96 inch OLED display module

The OLED (Organic Light Emitting Diode) display that we'll use in this tutorial is the SSD1306 model: a mono color, 0.96-inch display with 128×64 pixels as shown in the above figure.

The OLED display doesn't require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared to other displays.

The model we're using has four pins and communicates with any microcontroller using I2C communication protocol. There are models that come with an extra RESET pin or that communicate using SPI communication protocol.

OLED Display SSD1306 Pin Wiring

Because the OLED display uses I2C communication protocol, wiring is very simple. Use the following table as a reference.

Pin	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

Table 2: OLED pin configuration

Alternatively, it can follow the next schematic diagram to wire the ESP32 to the OLED display.

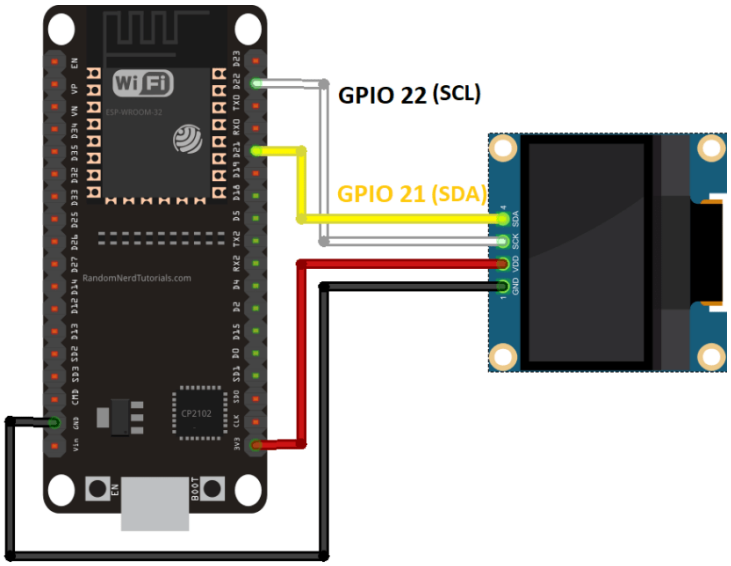


Figure 9: OLED interfacing with ESP32

In this example, we're using I2C communication protocol. The most suitable pins for I2C communication in the ESP32 are GPIO 22 (SCL) and GPIO 21 (SDA).

If you're using an OLED display with SPI communication protocol, use the following GPIOs.

- GPIO 18: CLK
- GPIO 19: MISO
- GPIO 23: MOSI
- GPIO 5: CS

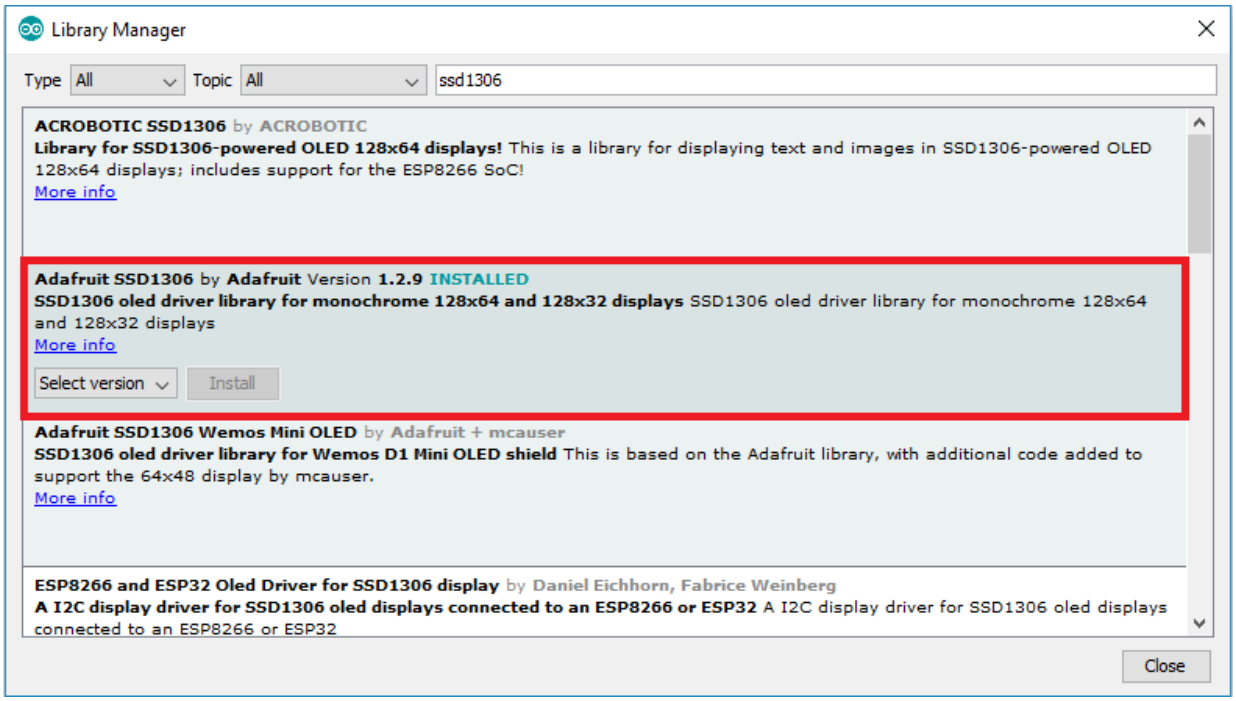
Installing SSD1306 OLED Library – ESP32

There are several libraries available to control the OLED display with the ESP32. In this project we'll use two Adafruit libraries: Adafruit_SSD1306 library and Adafruit_GFX library.

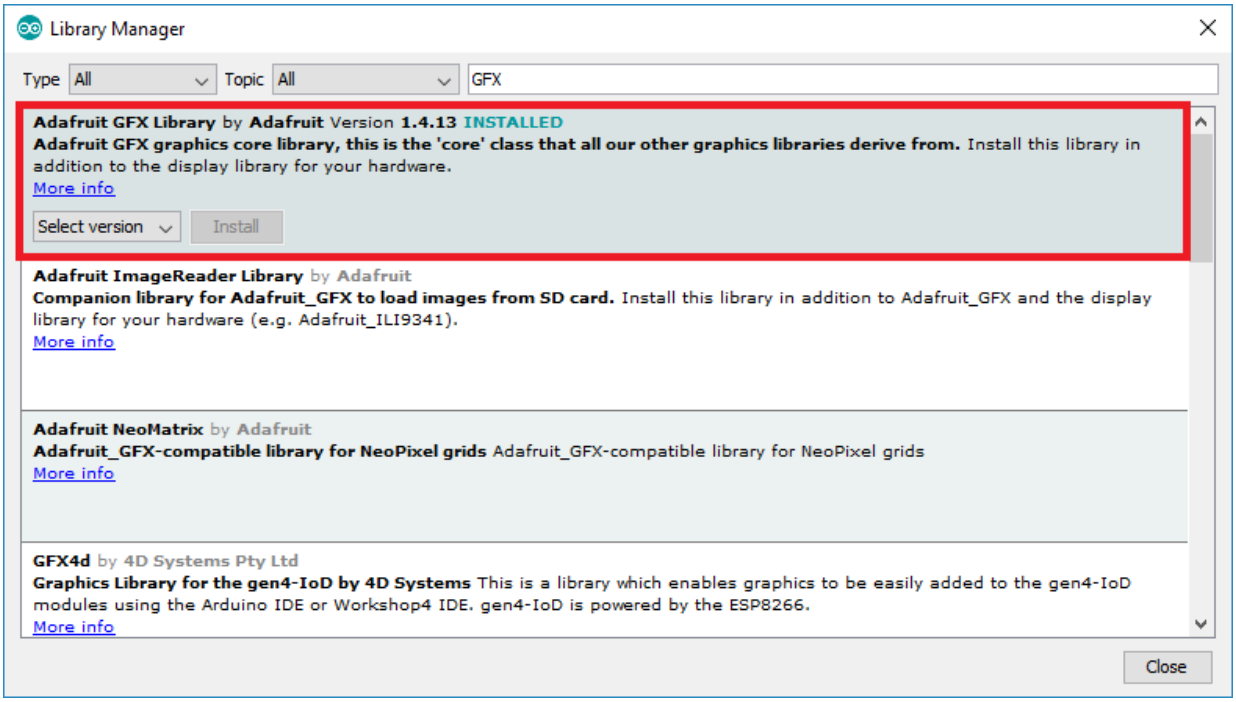
Follow the next steps to install those libraries.

1. Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries. The Library Manager should open.

2. Type "SSD1306" in the search box and install the SSD1306 library from Adafruit.



3. After installing the SSD1306 library from Adafruit, type “GFX” in the search box and install the library.

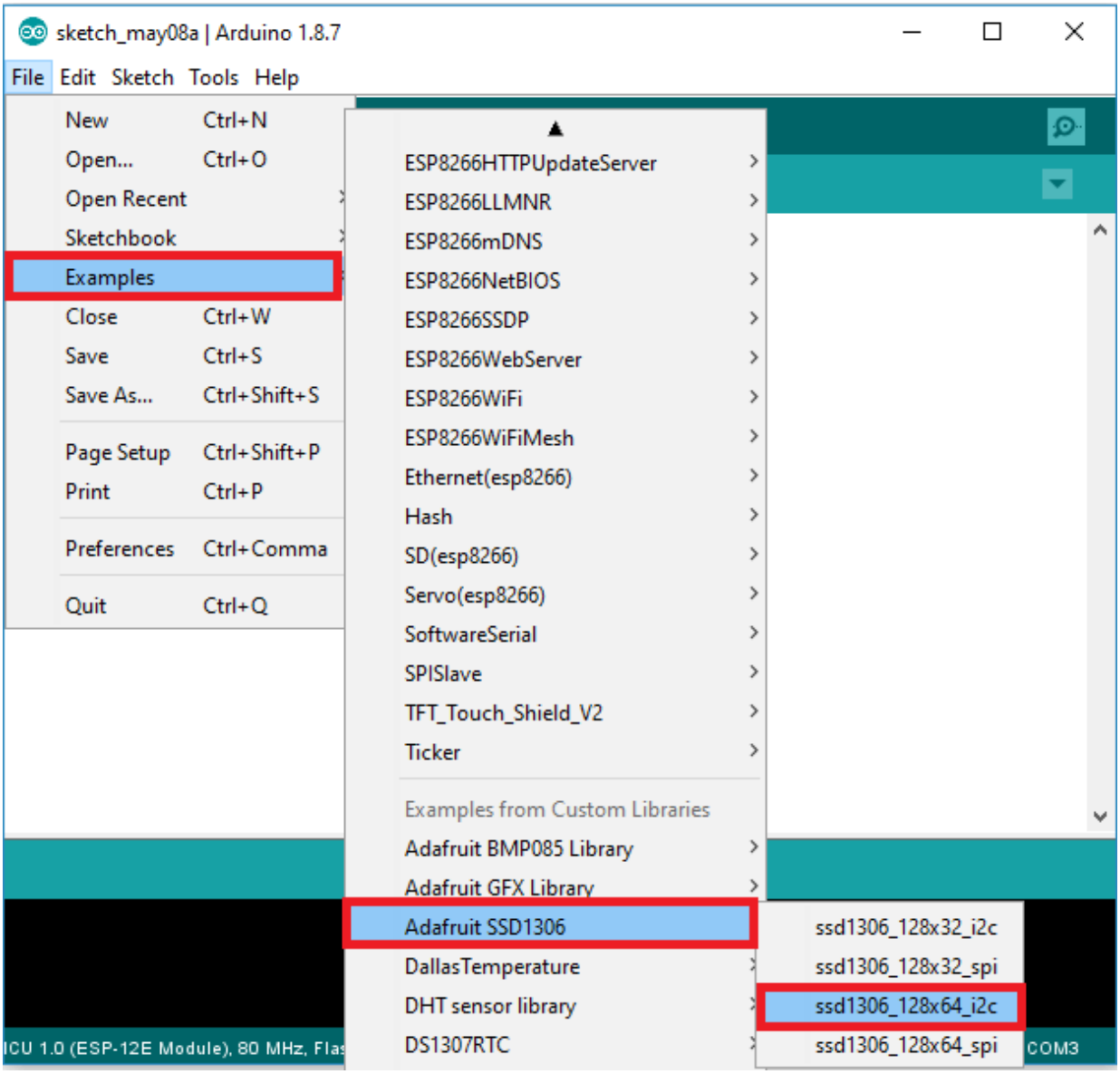


4. After installing the libraries, restart your Arduino IDE.

Testing OLED Display with ESP32

After wiring the OLED display to the ESP32 and installing all required libraries, you can use one example from the library to see if everything is working properly.

In your Arduino IDE, go to File > Examples > Adafruit SSD1306 and select the example for the display you’re using.



Write Text – OLED Display

The Adafruit library for the OLED display comes with several functions to write text. In this section, you'll learn how to write and scroll text using the library functions.

"Hello, world!" OLED Display

The following sketch displays Hello, world! message in the OLED display.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels


// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(115200);
```

```
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Hello, world!");
  display.display();
}

void loop() {

}
```

After uploading the code, this is what you'll get in your OLED:

Figure 10: Text message display on OLED

Let's take a quick look on how the code works.

Importing libraries

First, you need to import the necessary libraries. The Wire library to use I2C and the Adafruit libraries to write to the display: Adafruit_GFX and Adafruit_SSD1306.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Initialize the OLED display

Then, you define your OLED width and height. In this example, we're using a 128×64 OLED display. If you're using other sizes, you can change that in the SCREEN_WIDTH, and SCREEN_HEIGHT variables.

#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Then, initialize a display object with the width and height defined earlier with I2C communication protocol (&Wire).

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

The (-1) parameter means that your OLED display doesn't have a RESET pin. If your OLED display does have a RESET pin, it should be connected to a GPIO. In that case, you should pass the GPIO number as a parameter.

In the setup(), initialize the Serial Monitor at a baud raute of 115200 for debugging purposes.

Serial.begin(115200);
Initialize the OLED display with the begin() method as follows:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println("SSD1306 allocation failed");
  for(;;); // Don't proceed, loop forever
}
```

This snippet also prints a message on the Serial Monitor, in case we're not able to connect to the display.

Serial.println("SSD1306 allocation failed");

In case you're using a different OLED display, you may need to change the OLED address. In our case, the address is 0x3C.

if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {

After initializing the display, add a two second delay, so that the OLED has enough time to initialize before writing text:

delay(2000);

Clear display, set font size, color and write text

After initializing the display, clear the display buffer with the clearDisplay() method:

display.clearDisplay();

Before writing text, you need to set the text size, color and where the text will be displayed in the OLED.

Set the font size using the setTextSize() method:

display.setTextSize(1);

Set the font color with the setTextColor() method:
display.setTextColor(WHITE);
WHITE sets white font and black background.
Define the position where the text starts using the setCursor(x,y) method. In this case, we're setting the text to start at the (0,0) coordinates – at the top left corner.
display.setCursor(0,0);
Finally, you can send the text to the display using the println() method, as follows:
display.println("Hello, world!");
Then, you need to call the display() method to actually display the text on the screen.
display.display();

4.5.2 Relay Driver interfacing with microcontroller

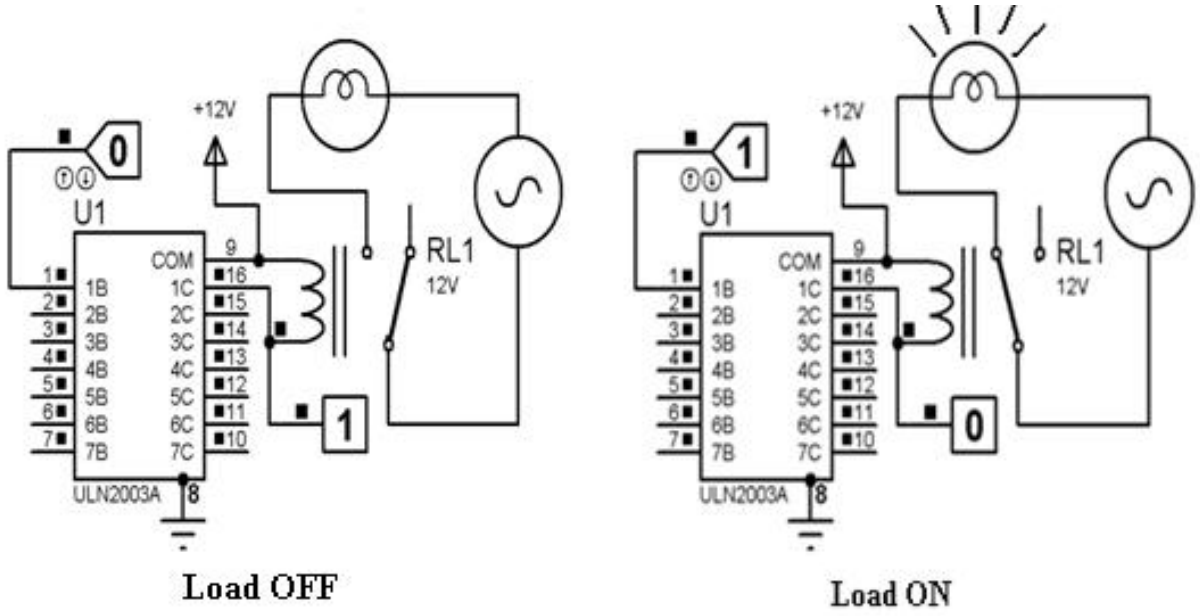


Figure 11: ULN2003A interfacing with microcontroller

The ULN2003A is a active high relay driver. 7 relays are controlled by this relay driver. Pin 1-7 are for controlling the relay which are connected to pin 10-16. For a '0' from microcontroller the corresponding relay is turned off and a '1' from microcontroller is turned on the relay.

CHAPTER 5

(Logic & Operation)

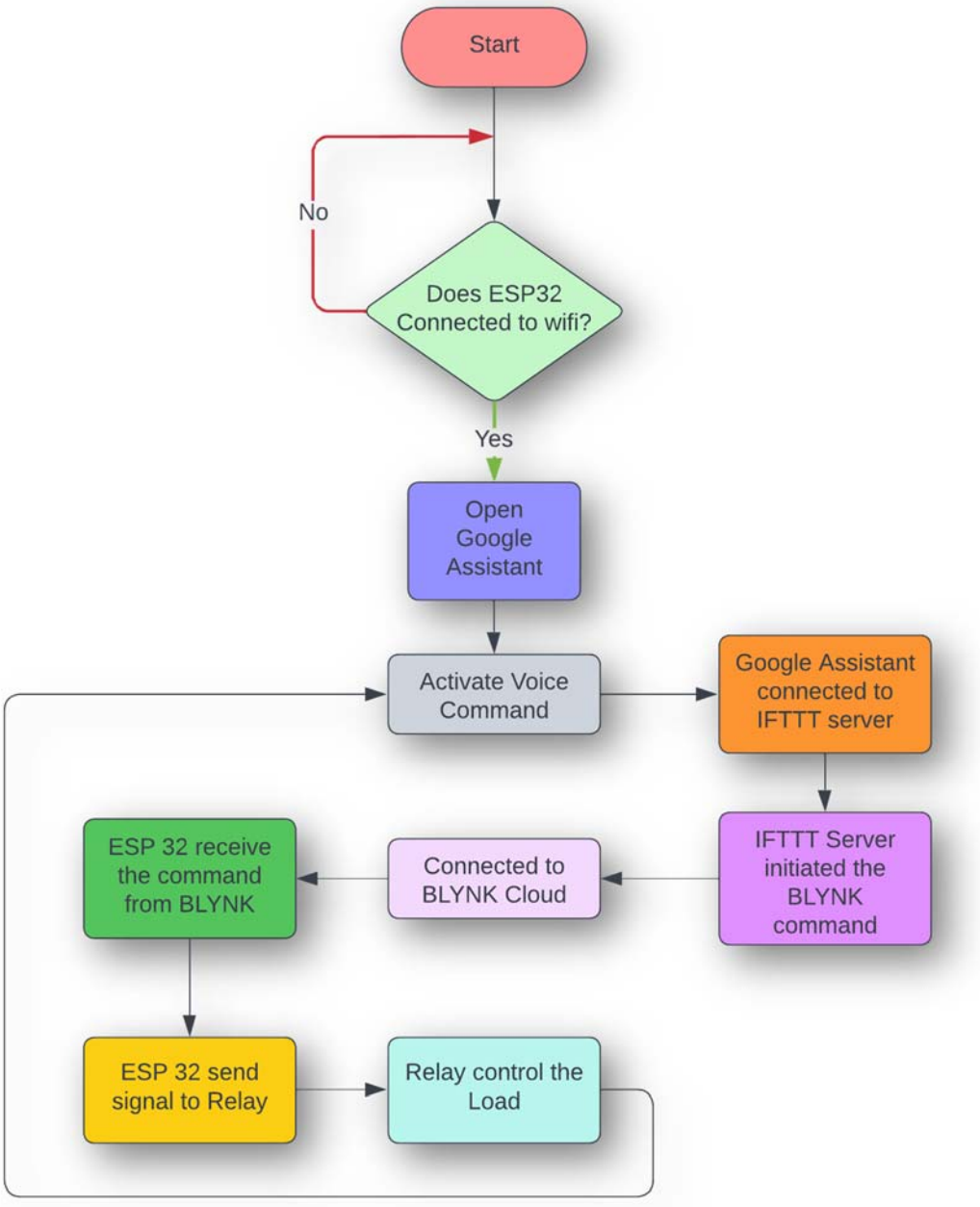
5.1 INTRODUCTION

After assembling the system, what remains is to observe its operation and efficiency of the system. The total system is divided in several sub systems, like

- Blynk cloud section
- Microcontroller section
- OLED section
- Audio feedback section
- Relay section
- IFTTT applet creation section
- Google Assistant setting section

The operation of the whole circuit is depending on every sections performance.

5.2 Flow Chart



5.3 Principle & Operations

First, we need to give supply to the prototype. The supply will be taken from 5 volt, 1amp adapter. As soon as we power up the circuit the indicator LED will glow to indicates that the board is powered up. Now the OLED will show the message “Please connect the wifi…” if the wifi is not connected to the circuit. Here we need to connect the microcontroller ESP32 to the wifi. Now the device is connected to the wifi.

After a successful connection user need to give commands to google assistant predefined phase like “**Please switch on the light**” to turn on the light and “**Please switch off the light**” to turn off the light. Then Google assistant will send the commands to **IFTTT** server. If the command match to the predefined command which is previously added in the IFTTT applets then IFTTT will initiated the BLYNK API command. Then Blynk will send API command to the ESP32.

BLYNK API’s

The on- API Command is

https://blr1.blynk.cloud/external/api/update?token=E73pXWQ2I23F7Al3HnZemyX6W97sSbTD&V0=1

The off-API command is

https://blr1.blynk.cloud/external/api/update?token=E73pXWQ2I23F7Al3HnZemyX6W97sSbTD&V0=0

receiving the API command from the cloud the ESP32 microcontroller will instruct the relay to switch ON or OFF respectively. The status of the relay will also be shown on a 0.96-inch OLED screen on board.

The Google Assistant based IoT switch is very much secure because the API command for controlling the load is generated only through the voice command in a particular google account. Except the concern person it is impossible for others to generate the same the voice command because this command are google account specific.

5.4 Advantages of the GA load switching

A. Maintenance: It is an economical system that requires very less maintenance as compared to conventional system as it has no complicated circuits and delicate mechanisms. This saves the additional maintenance cost.

***B. Cost*:** The main advantage of this project is it has very low cost than the conventional one available in markets. For example, some commercial controllers use microcontrollers which alone costs around Rs.900. Some controllers even have a price range of Rs.2000-Rs. 4000. But for our system, the components used are less in number and easily available. Hence losses will be less leading to a better efficiency.

C. Construction: The construction of a Google Assistant based IoT load switching system is very simple as it requires only a few components. The circuit involved is also relatively simpler. The space and power requirement to operate this system is very less.

D. Skill Required: Since the system we implement is simpler than the ones conventionally available, it can be easily made at home. The controller can also be easily operated by anyone.

5.5 Disadvantages

- The actual status of the load is unknown.
- No backup action will take for any false switching by controller itself.

5.6 Cost estimation of the project

Sl. No.	Components	Quantity	Cost (in ৳,
1	ESP32 Microcontroller	1	370
2	330 Ω resistance	1	2
3	OLED Display	1	450
4	5-volt static Relay	1	25
5	Female pin header	1	10
6	Piezo Buzzer	1	15
7	Single stand wire	3m	30
8	Wire nipper	1	180
9	Wire striper	1	100
10	Soldering Iron	1	250
11	Soldering material	1	50
12	De-soldering pump	1	130
13	BC 547	1	2
14	DC Socket	1	10
15	IN 4007	1	2
16	Bulb Holder	1	35
17	Bulb	1	75
18	Blank PCB (KS 100)	1	40
Total			**1776/-**

Table 3: Costing of the projects

5.7 Photographs of the protype

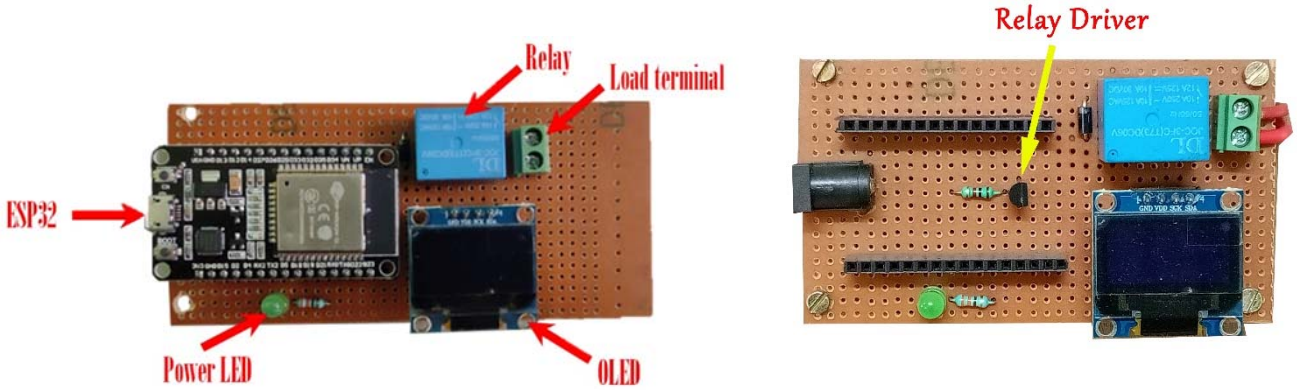


Figure 12: Main prototype and relay driver

Figure 13: Load is switch off and switch on condition

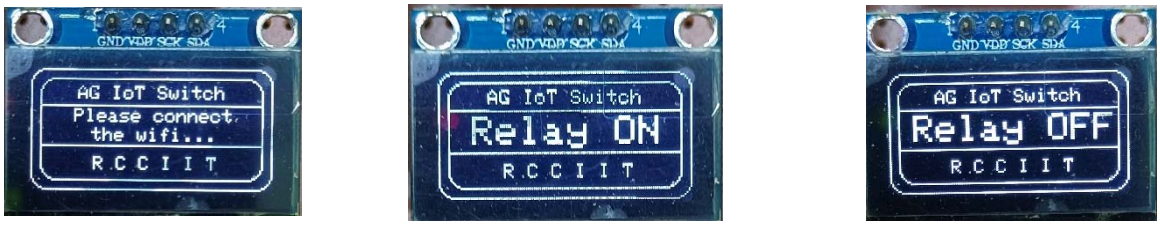


Figure 14: Different messages on the OLED screen

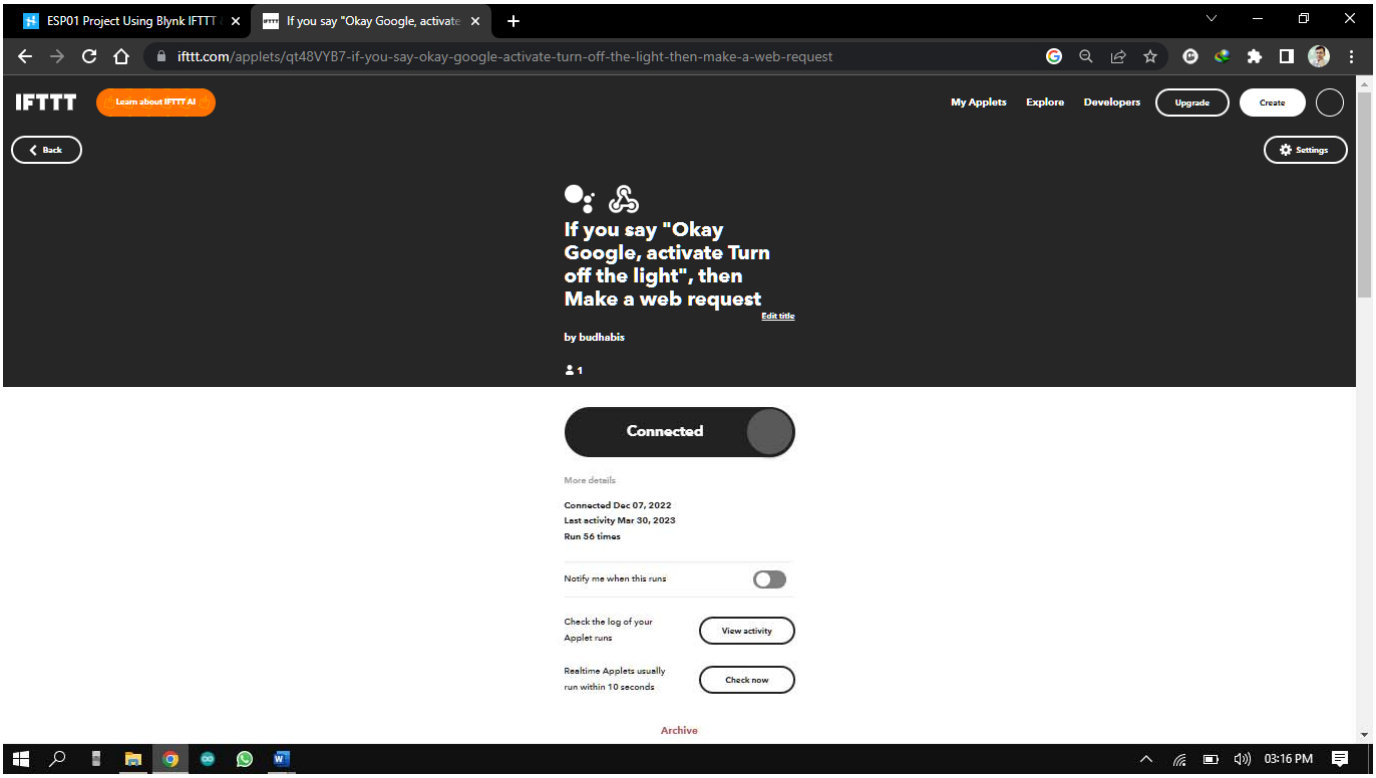


Figure 15: Making of applets in IFTTT

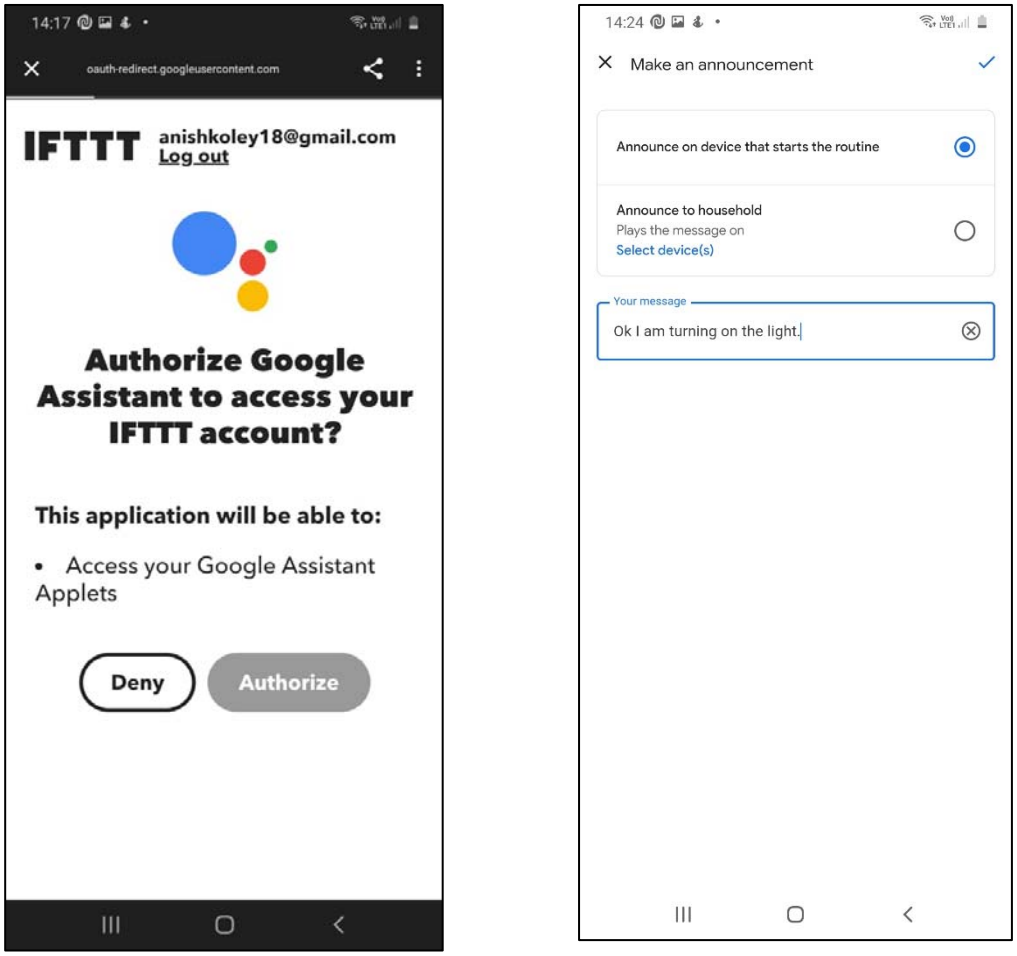


Figure 16 and 17: Using the GA app to control the load

Chapter 6

(Conclusion & Future Scope)

6.1 CONCLUSION

Here we are developed a '**Google Assistant base Home Automation**' circuit which could be used for domestic load switching. It's a secure way to control the load. The circuit mainly consists of four parts such as IoT section, IFTTT and Google assistant seting, audio and visual feedback, switching on and off the loads. When a command is given to the microcontroller ESP32 through blynk cloud it immediately changes the state of the load. The commands that are being passed through the microcontroller are generated through Google assistant voice command service which will connected to the IFTTT IoT service platform. If the command match to the pre-defined command which is previously added in the IFTTT applets then IFTTT will initiated the BLYNK API command. Then Blynk will send API command to the ESP32. Receiving the API command from the cloud the ESP32 microcontroller will instruct the relay to switch ON or OFF. The status of the relay will also be shown on a 0.96-inch OLED screen on board.

6.2 RESULTS

The prototype was made according to the circuit diagram and the results were as expected. The loads are switched on when the API command is received by the controller through the IFTTT connected blynk server. The loads are switched off only when the respective commands are generated through the google assistant app using a mobile device.

6.3 FUTURE WORK

In our present prototype the actual status of the load is not showing. It only shows the status of the ON/OFF command is generated by the controller. In future we will connect a feedback system to give the actual status of the load to the remote user. Also, the current control is ON OFF control. In future we also develop the continuous control of the load.

Chapter 7

(References)

[1] Md Sarwar Kamal in (2017) “Efficient low-cost supervisory system for Internet of Things enabled smart home.” Publisher: IEEE International Conference on Communication (ICC 2017).

[2] Aayush Agarwal, Anshul Sharma, Asim Saket Samad and S Babeetha (2018) “UJALA- Home Automation System Using Google Assistant” Volume: 04 Issue: 02 | 2018

[3] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, Subhas Chandra Mukhopadhyay (2013)“Towards the Implementation of IoT for Environmental Condition Monitoring in Homes” Publisher: IEEE Sensors Journal 13 |October-2013

[4] Anant Vaibhav, Sarthak Jain, Lovely Goyal “Raspberry Pi based Interactive Home Automation System through E-mail” 2014 International Conference on Reliability, Optimization and Information Technology - ICROIT 2014, India, Feb 6- 8 2014

[5] Ana Marie. D Celebre, Ian Benedict A. Medina, Alec Zandrae D. Dubouzet, Adrian Neil M. Surposa, Engr. Reggie C. Gustilo “Home Automation Using Raspberry Pi through Siri Enabled Mobile Devices” 8th IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM)

[6] Nikhil Rathod1, Dr. P. D. Paikrao2, “A Survey on Home Automation by Using Voice Command Based on IOT”, IJSART - Volume 5 Issue 11, pp. 110- 113, NOVEMBER 2019

[7] N. P Jawarkar, V. Ahmed, S.A. Ladhake, and R.D Thakare – ‘Microcontroller based Remote monitoring using mobile phone through spoken commands’,- Journal of networks, Publisher: World Journal control science and engineering, Place: Lagos, Country: Nigeria, Vol. No.:3, Iss. No.2, pp.58 to 83, Year: 2008.

[8] S.Meera, M.Ramya, L.Megala,” Smart Hospitals Using Internet Of Things (Iot)”, International Journal Of Advanced Research Trends In Engineering And Technology (Ijartet), Vol. 6, Pp. 9-13, December 2019.

[9] Ms. Preeti U. Melikatti, Research Scholar, Department of Computer Science and Technology, V.V.P.I.E.T. Solapur, Maharashtra, India. IJIERT] ISSN: 2394-3696 Website: ijiert.org VOLUME 8, ISSUE 12, Dec. -2021.

[10] Nikhil Singh, Shambhu Shankar Bharti, Rupal Singh, Dushyant Kumar Singh “Remotely controlled home automation system”, Publisher: IEEE International Conference on Advances in Engineering and Technology Research (ICAETR 2014).

Appendix A

(Hardware description)

Transformer less AC to DC power supply circuit using dropping capacitor

Production of low voltage DC power supply from AC power is the most important problem faced by many electronics developers and hobbyists. The straight forward technique is the use of a step down transformer to reduce the 230 V or 110V AC to a preferred level of low voltage AC. But *SMPS* power supply comes with the most appropriate method to create a low cost power supply by avoiding the use of bulky transformer. This circuit is so simple and it uses a voltage dropping capacitor in series with the phase line. Transformer less power supply is also called as capacitor power supply. It can generate 5V, 6V, 12V 150mA from 230V or 110V AC by using appropriate zener diodes.

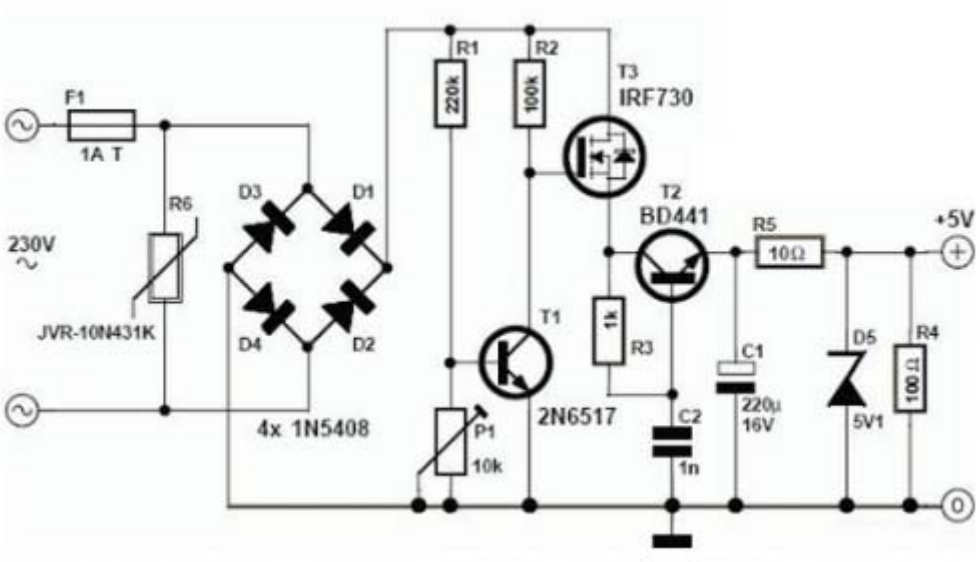


Figure 18: Transformer less SMPS 5-volt power supply

Working of Transformer less capacitor power supply

- This transformer less power supply circuit is also named as capacitor power supply since it uses a special type of AC capacitor in series with the main power line.
- A common capacitor will not do the work because the mains spikes will generate holes in the dielectric and the capacitor will be cracked by passing of current from the mains through the capacitor.
- X rated capacitor suitable for the use in AC mains is vital for reducing AC voltage.
- A X rated dropping capacitor is intended for 250V, 400V, 600V AC. Higher voltage versions are also obtainable. The dropping capacitor is non polarized so that it can be connected any way in the circuit.
- The 470kΩ resistor is a bleeder resistor that removes the stored current from the capacitor when the circuit is unplugged. It avoids the possibility of electric shock.
- Reduced AC voltage is rectified by bridge rectifier circuit. We have already discussed about bridge rectifiers. Then the ripples are removed by the 1000µF capacitor.
- This circuit provides 24 volts at 160 mA current at the output. This 24 volt DC can be regulated to necessary output voltage using an appropriate 1 watt or above zener diode.

- Here we are using 6.2V zener. You can use any type of zener diode in order to get the required output voltage.

ESP32 microcontroller

ESP32 is the SoC (System on Chip) microcontroller which has gained massive popularity recently. Whether the popularity of ESP32 grew because of the growth of IoT or whether IoT grew because of the introduction of ESP32 is debatable. If you know 10 people who have been part of the firmware development for any IoT device, chances are that 7−8 of them would have worked on ESP32 at some point. So what is the hype all about? Why has ESP32 become so popular so quickly? Let's find out.

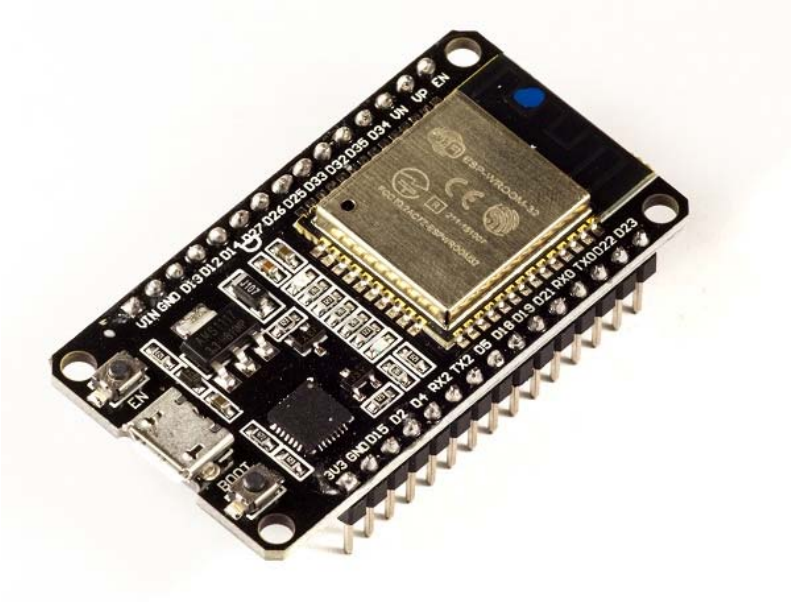


Figure 19: ESP32 Development module

Before we delve into the actual reasons for the popularity of ESP32, let's take a look at some of its important specifications. The specs listed below belong to the ESP32 WROOM 32 variant.−

- Integrated Crystal− 40 MHz
- Module Interfaces− UART, SPI, I2C, PWM, ADC, DAC, GPIO, pulse counter, capacitive touch sensor
- Integrated SPI flash− 4 MB
- ROM− 448 KB (for booting and core functions)
- SRAM− 520 KB
- Integrated Connectivity Protocols− WiFi, Bluetooth, BLE
- On−chip sensor− Hall sensor
- Operating temperature range− −40 − 85 degrees Celsius
- Operating Voltage− 3.3V
- Operating Current− 80 mA (average)

With the above specifications in front of you, it is very easy to decipher the reasons for ESP32's popularity. Consider the requirements an IoT device would have from its microcontroller (μC).

If you've gone through the previous chapter, you'd have realized that the major operational blocks of any IoT device are sensing, processing, storage, and transmitting. Therefore, to begin with, the μC should be able to interface with a variety of sensors. It should support all the common communication protocols required for sensor interface: UART, I2C, SPI. It should have ADC and pulse counting capabilities. ESP32 fulfills all of these requirements. On top of that, it also can interface with capacitive touch sensors. Therefore, most common sensors can interface seamlessly with ESP32.

Secondly, the μC should be able to perform basic processing of the incoming sensor data, sometimes at high speeds, and have sufficient memory to store the data. ESP32 has a max operating frequency of 40 MHz, which is sufficiently high. It has two cores, allowing parallel processing, which is a further add-on. Finally, its 520 KB SRAM is sufficiently large for processing a large array of data onboard. Many popular processes and transforms, like FFT, peak detection, RMS calculation, etc. can be performed onboard ESP32. On the storage front, ESP32 goes a step ahead of the conventional microcontrollers and provides a file system within the flash. Out of the 4 MB of onboard flash, by default, 1.5 MB is reserved as SPIFFS (SPI Flash File System). Think of it as a mini−SD Card that lies within the chip itself. You can not only store data, but also text files, images, HTML and CSS files, and a lot more within SPIFFS. People have displayed beautiful Webpages on WiFi servers created using ESP32, by storing HTML files within SPIFFS.

Finally, for transmitting data, ESP32 has integrated WiFi and Bluetooth stacks, which have proven to be a game-changer. No need to connect a separate module (like a GSM module or an LTE module) for testing cloud communication. Just have the ESP32 board and a running WiFi, and you can get started. ESP32 allows you to use WiFi in Access Point as well as Station Mode. While it supports TCP/IP, HTTP, MQTT, and other traditional communication protocols, it also supports HTTPS. Yep, you heard that right. It has a crypto−core or a crypto-accelerator, a dedicated piece of hardware whose job is to accelerate the encryption process. So you cannot only communicate with your web server, you can do so securely. BLE support is also critical for several applications. Of course, you can interface LTE or GSM or LoRa modules with ESP32. Therefore, on the 'transmitting data' front as well, ESP32 exceeds expectations.

With so many features, ESP32 would be costing a fortune, right? That's the best part. ESP32 dev modules cost in the ballpark of ₹ 500. Not only that, the chip dimensions are quite small (25 mm x 18 mm, including the antenna area), allowing its use in devices requiring a very small form factor.

Finally, ESP32 can be programmed using the Arduino IDE, making the learning curve much less steep. Isn't that great? Are you excited to get your hands dirty with ESP32? Then let's start by installing the ESP32 board in the Arduino IDE in the next chapter. See you there.

OLED

OLED displays are available in a range of sizes (such as 128×64, 128×32) and colors (such as white, blue, and dual-color OLEDs). Some OLED displays have an I2C interface, while others have an SPI interface.

One thing they all have in common, however, is that at their core is a powerful single-chip CMOS OLED driver controller – SSD1306, which handles all RAM buffering, requiring very little work from your Arduino.
In this tutorial, we'll be using both I2C and SPI 0.96-inch 128×64 OLED displays. Don't worry if your module is a different size or color; the information on this page is still useful.

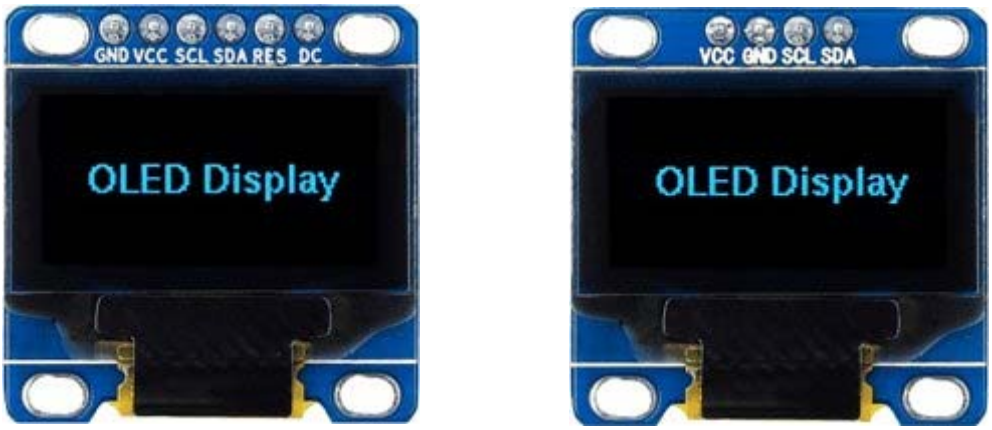


Figure 19: SPI and IIC OLED display module

Relay Driver

- The ULN2003 is a monolithic high voltage and high current Darlington transistor arrays.
- It consists of seven NPN Darlington pairs that features high-voltage outputs with common-cathode clamp diode for switching inductive loads.
- The collector-current rating of a single Darlington pair is 500mA.
- The ULN functions as an inverter.
- If the logic at input 1B is high then the output at its corresponding pin 1C will be low.

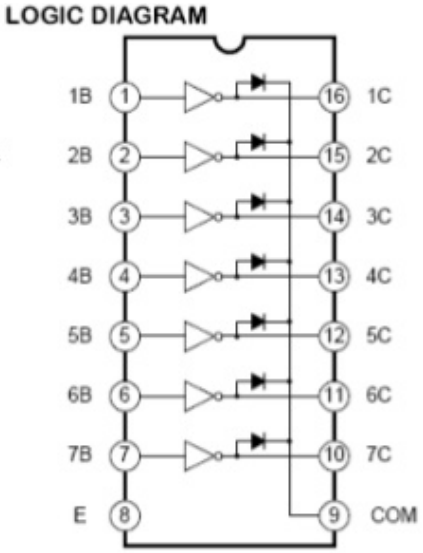


Figure 20: ULN2003A Internal Block Diagram

Resistor

Figure 21: Resistor

Resistance is the opposition of a material to the current. It is measured in Ohms Ω. All conductors represent a certain amount of resistance, since no conductor is 100% efficient. To control the electron flow (current) in a predictable manner, we use resistors. Electronic circuits use calibrated lumped resistance to control the flow of current. Broadly speaking, resistor can be divided into two groups viz. fixed & adjustable (variable) resistors. In fixed resistors, the value is fixed & cannot be varied. In variable resistors, the resistance value can be varied by an adjuster knob. It can be divided into (a) Carbon composition (b) Wire wound (c) Special type. The most common type of resistors used in our projects is carbon type. The resistance value is normally indicated by color bands. Each resistance has four colors, one of the band on either side will be gold or silver, this is called fourth band and indicates the tolerance, others three band will give the value of resistance (see table). For example if a resistor has the following marking on it say red, violet, gold. Comparing these colored rings with the color code, its value is 27000 ohms or 27 kilo ohms and its tolerance is ±5%. Resistor comes in various sizes (Power rating).The bigger the size, the more power rating of 1/4 watts. The four color rings on its body tells us the value of resistor value.

Color Code of the resistor

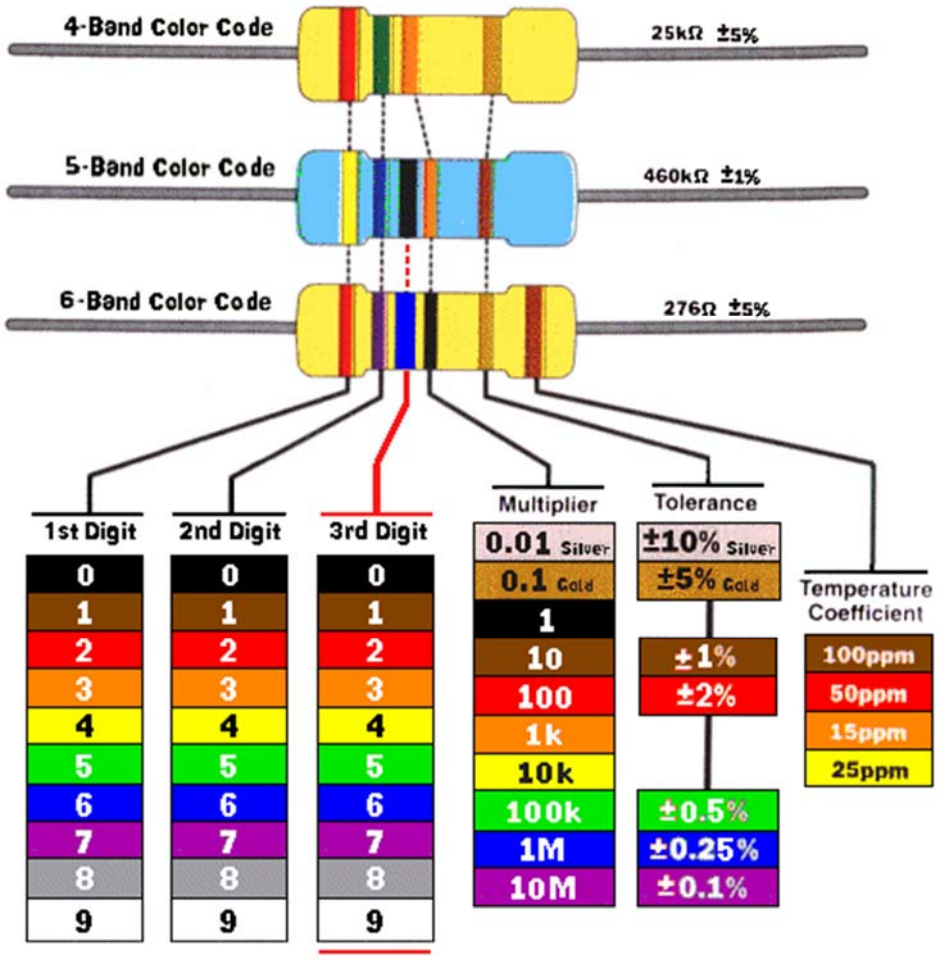


Figure 22: Color Code for resistance

RELAY

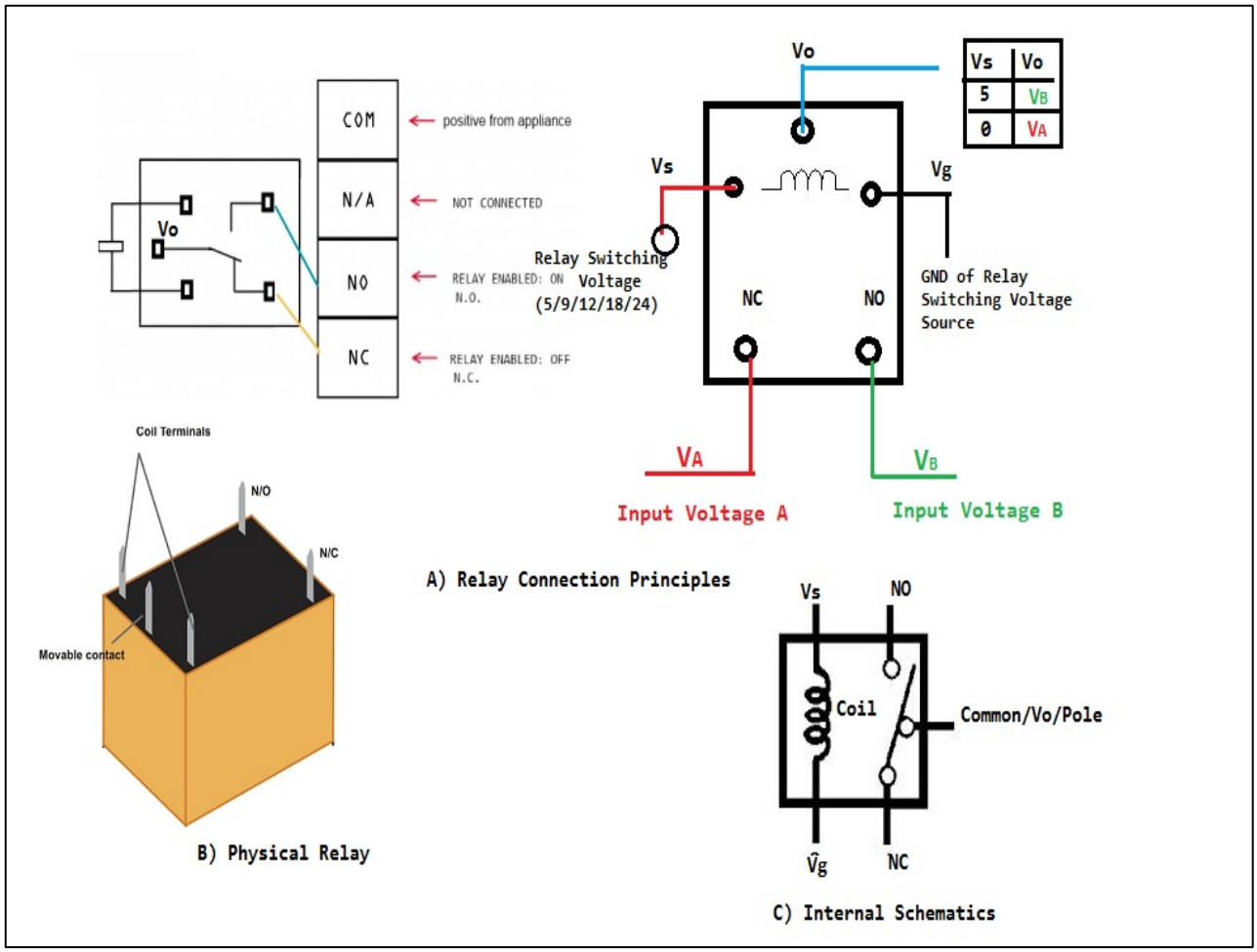


Figure 23: 6 volt Cube Relay

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches.

The relay's switch connections are usually labeled COM (POLE), NC and NO:

COM/POLE= Common, NC and NO always connect to this, it is the moving part of the switch.

NC = Normally Closed, COM/POLE is connected to this when the relay coil is not magnetized.

NO = Normally Open, COM/POLE is connected to this when the relay coil is MAGNETIZED and vice versa.

Capacitors

It is an electronic component whose function is to accumulate charges and then release it.

To understand the concept of capacitance, consider a pair of metal plates which all are placed near to each other without touching. If a battery is connected to these plates the positive pole to one and the negative pole to the other, electrons from the

Figure 24: Types of capacitors

battery will be attracted from the plate connected to the positive terminal of the battery. If the battery is then disconnected, one plate will be left with an excess of electrons, the other with a shortage, and a potential or voltage difference will exists between them. These plates will be acting as capacitors. Capacitors are of two types: - (1) **fixed type** like ceramic, polyester, electrolytic capacitors - these names refer to the material they are made of aluminum foil. (2) **Variable type** like gang condenser in radio or trimmer. In fixed type capacitors, it has two leads and its value is written over its body and variable type has three leads. Unit of measurement of a capacitor is farad denoted by the symbol F. It is a very big unit of capacitance. Small unit capacitor are pico-farad denoted by pf (1pf=1/1000,000,000,000 f) Above all, in case of electrolytic capacitors, it's two terminal are marked as (-) and (+).

Piezo buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke. A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Figure 25: Piezo Buzzer

Blank PCB

A **printed circuit board** (**PCB**) mechanically supports and electrically connects electronic components using conductive tracks, pads and other features etched from copper sheets laminated onto a non-conductive substrate. PCBs can be *single sided* (one copper layer), *double sided* (two copper layers) or *multi-layer* (outer and inner layers). Multi-layer PCBs allow for much higher component density. Conductors on different layers are connected with plated-through holes called vias. Advanced PCBs may contain components - capacitors, resistors or active devices - embedded in the substrate.

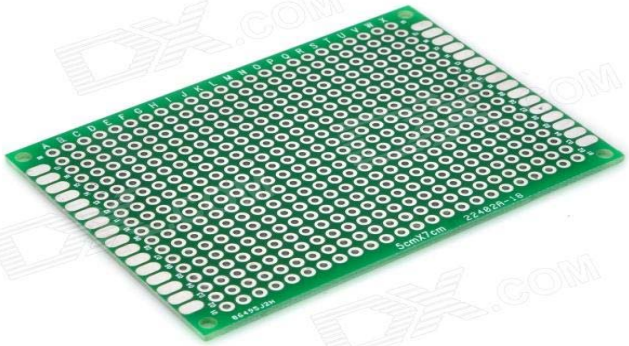


Figure 26: Blank glass epoxy PCB Board

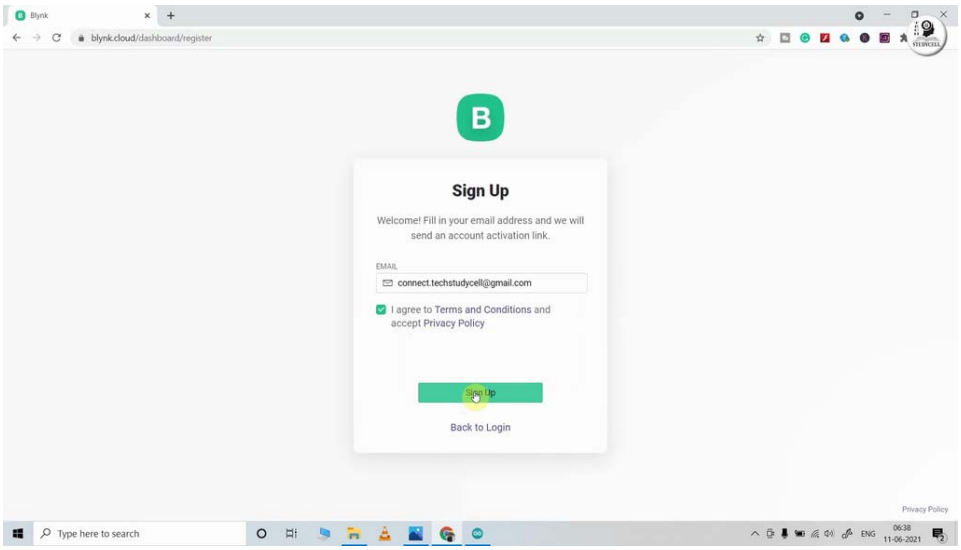
FR-4 glass epoxy is the primary insulating substrate upon which the vast majority of rigid PCBs are produced. A thin layer of copper foil is laminated to one or both sides of an FR-4 panel. Circuitry interconnections are etched into copper layers to produce printed circuit boards. Complex circuits are produced in multiple layers.

Printed circuit boards are used in all but the simplest electronic products. Alternatives to PCBs include wire wrap and point-to-point construction. PCBs require the additional design effort to lay out the circuit, but manufacturing and assembly can be automated. Manufacturing circuits with PCBs is cheaper and faster than with other wiring methods as components are mounted and wired with one single part. Furthermore, operator wiring errors are eliminated.

Appendix B

(Creating applets in IFTTT and connect it with blynk and Google Assistant)

Create Blynk Cloud FREE Account



For this Google Assistant based home automation project, we have used the Blynk IoT Cloud Free plan.

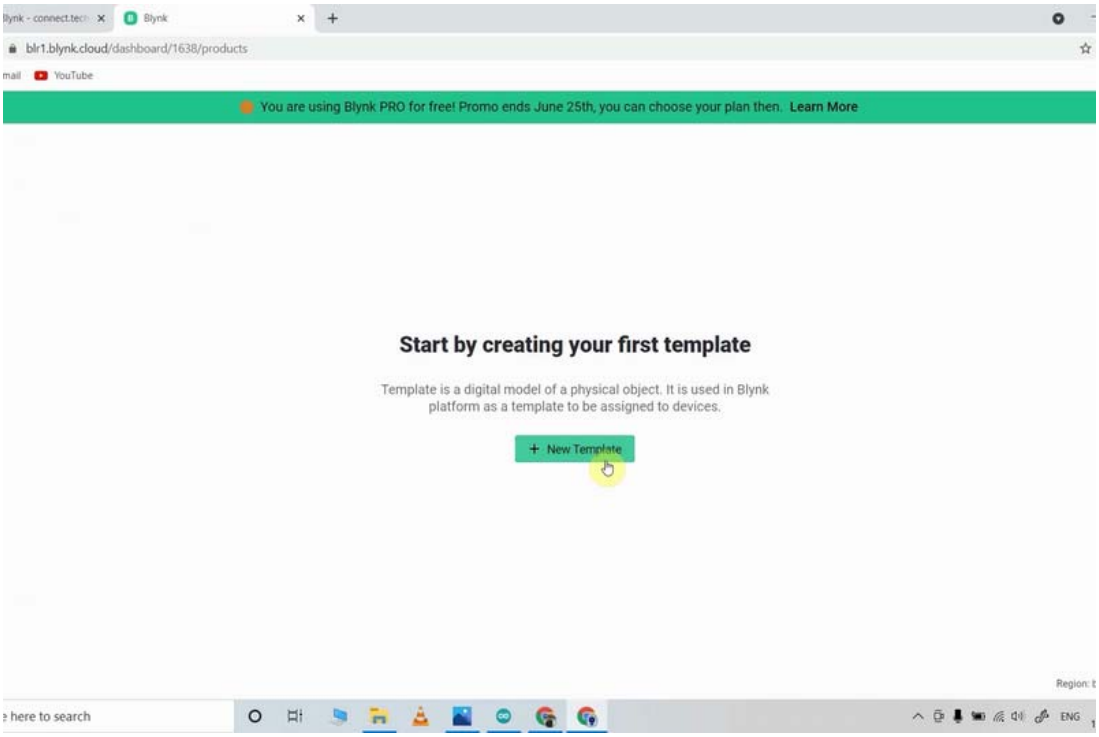
Click on the following link to create a Blynk Cloud account.

https://blynk.cloud/dashboard/register

1. Enter email ID, then click on "Sign Up". A verification email will send to the mail.
2. Click on Create Password in the email, then set the password, click on Next.
3. Enter your first name, click on Done.

   After that Blynk cloud dashboard will open.

   Create a New Template in Blynk Cloud



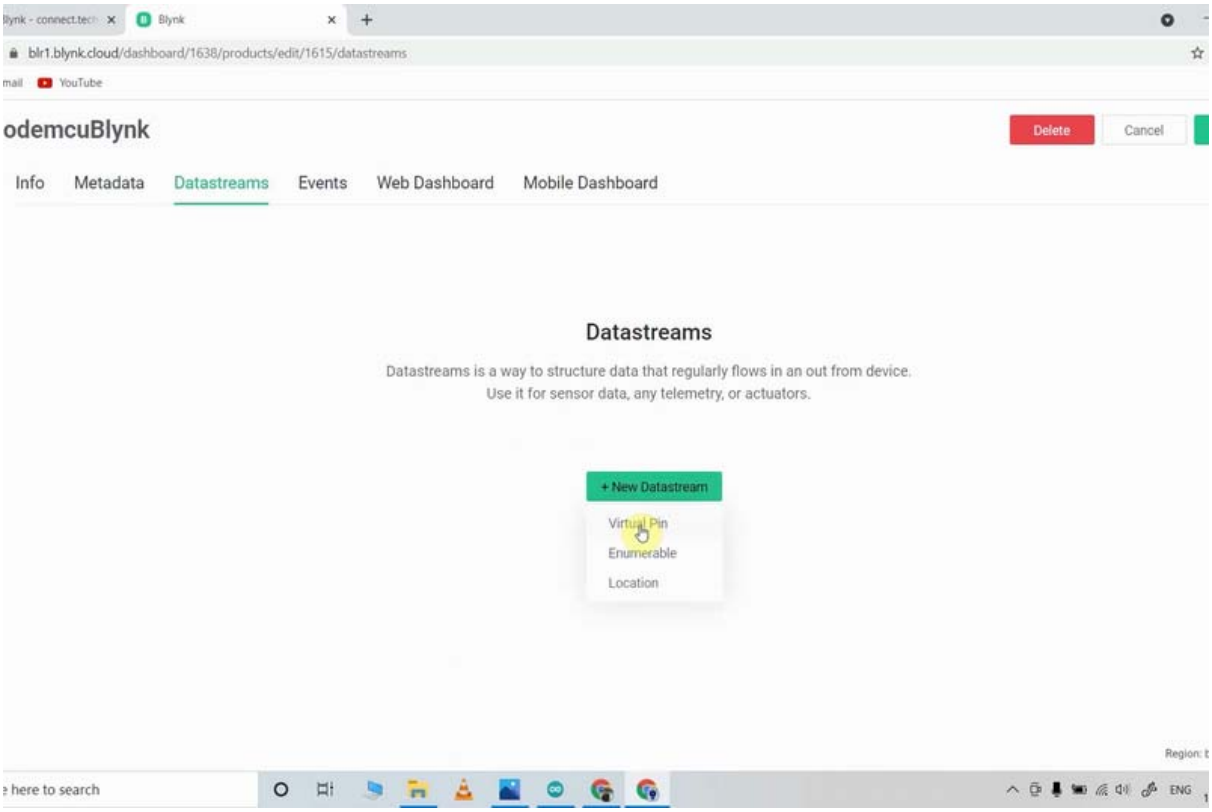
First, we have to create a template in the Blynk cloud.

1. Click on New Template.
2. Enter a template name, select the hardware as ESP32, and the connection type will WiFi.
3. Then click on DONE.

You will get the BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME after creating the temple.

The BLYNK_TEMPLATE_ID and BLYNK_DEVICE_NAME will be required while programming the ESP32.

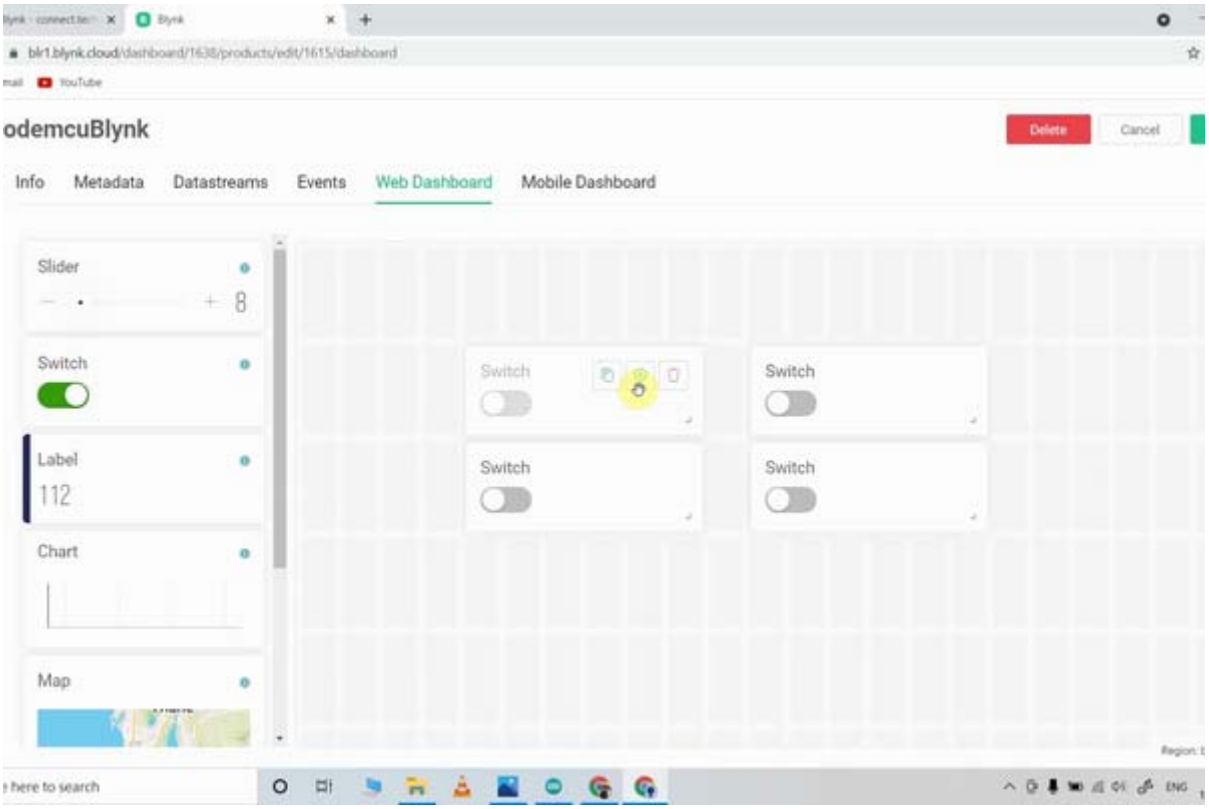
Create a Datastream in Blynk Cloud

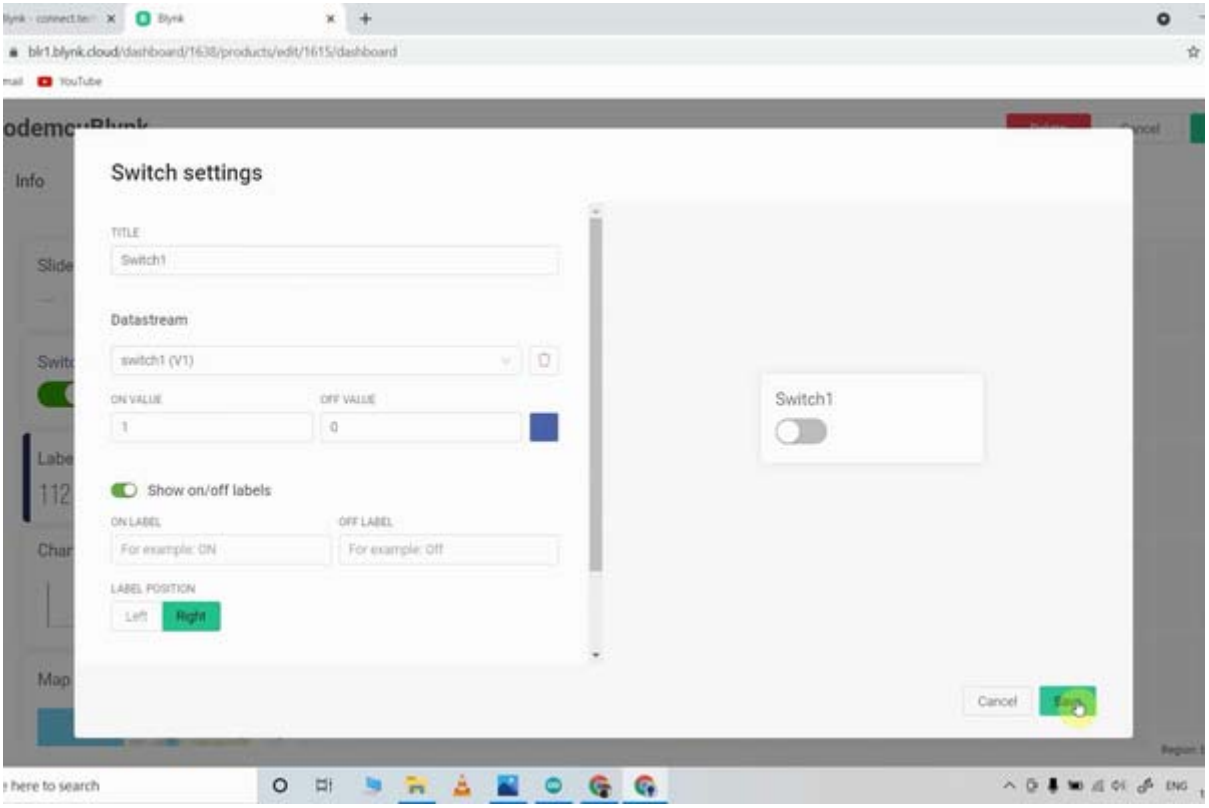


After that, you have to create Datastream. Here we will control 1 relay, so we have to create 1 Datastream.

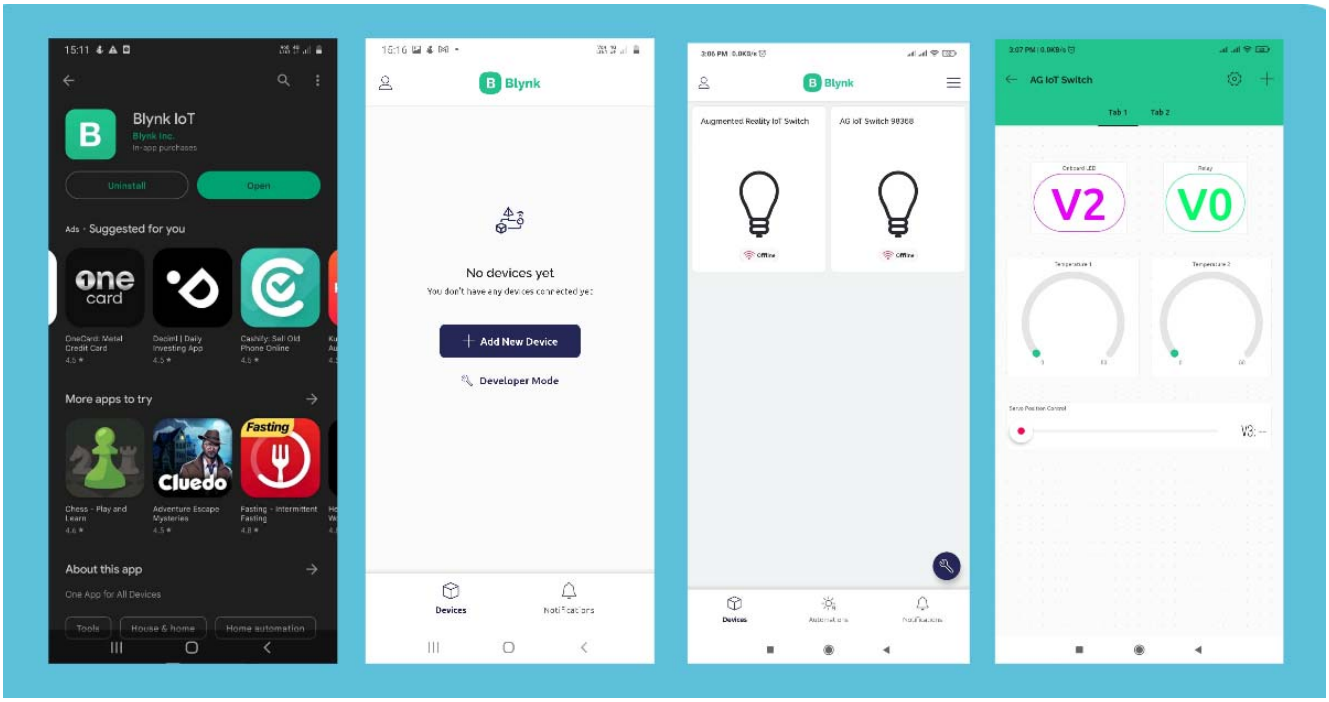
1. Go to the Datastreams tab.
2. Click on New Datastream and select Virtual Pin.
3. Enter a name, select the virtual pin V0, and the datatype will be Integer.
4. Then click on Create.

Set Up Blynk Cloud Web Dashboard



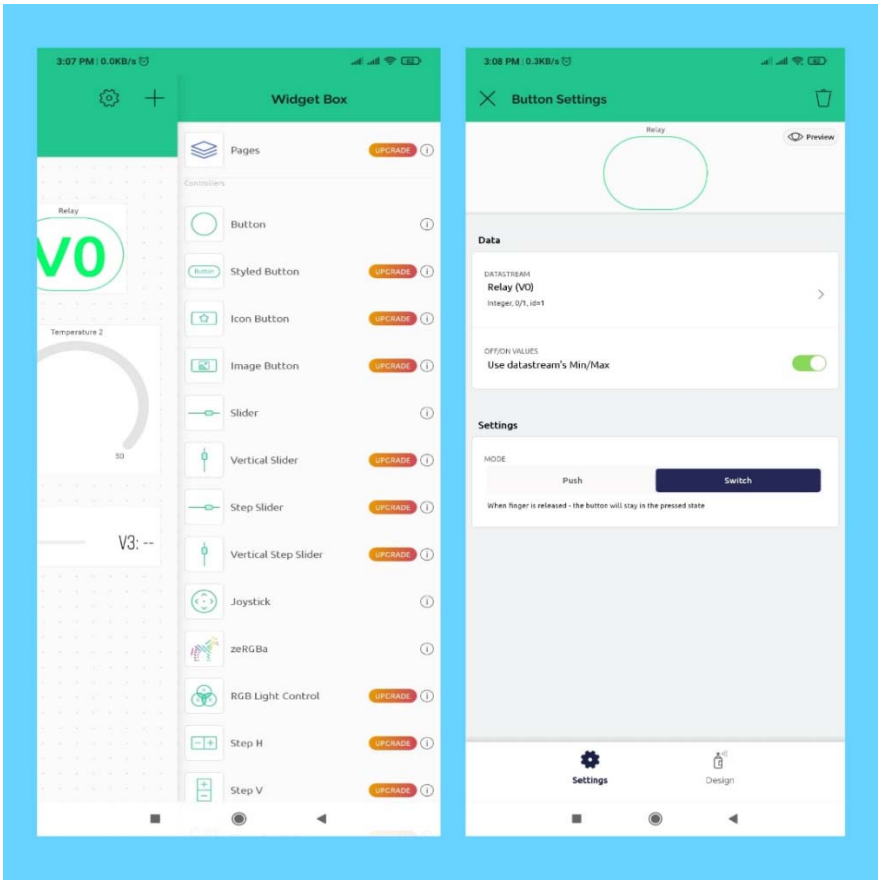


Now go to the web dashboard tab.
Drag and drop 1 Switch widgets.
Go to the settings of switch widget, and select the particular Datastream we have created earlier.
Install Blynk IoT App to Configure Mobile Dashboard



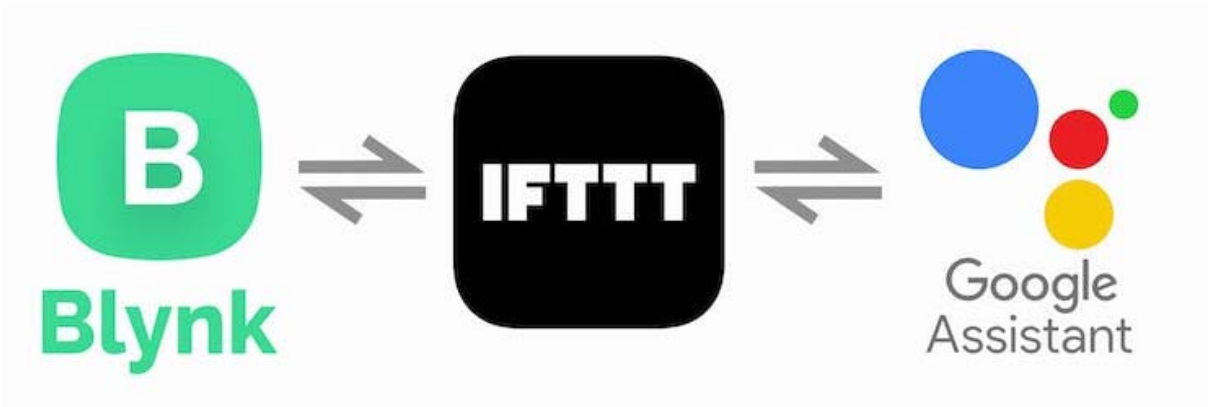
1. Install the Blynk IoT app from Google Play Store or App Store. Then log in.
2. Go to Developer Mode.
3. Tap on the template that you have already made.
4. Now go to the Widget box (on the right) to add widgets.

Add Widgets in Blynk IoT App

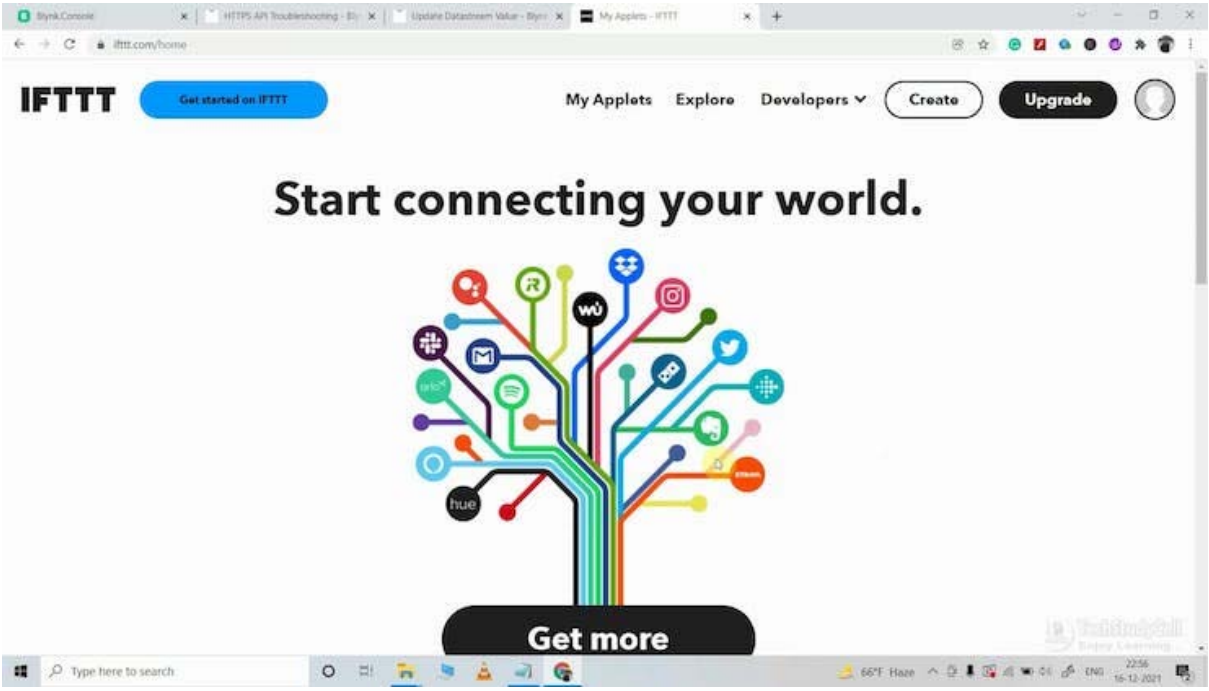


1. Add 1 Button widgets from Widget Box.
2. Go to Button widget settings.
3. Enter the name, select Datastream, Mode will be Switch. Then exit.
4. After setting all the Buttons tap on exit.

Now we will connect the Google Assistant with Blynk using IFTTT.



Create FREE Account in IFTTT

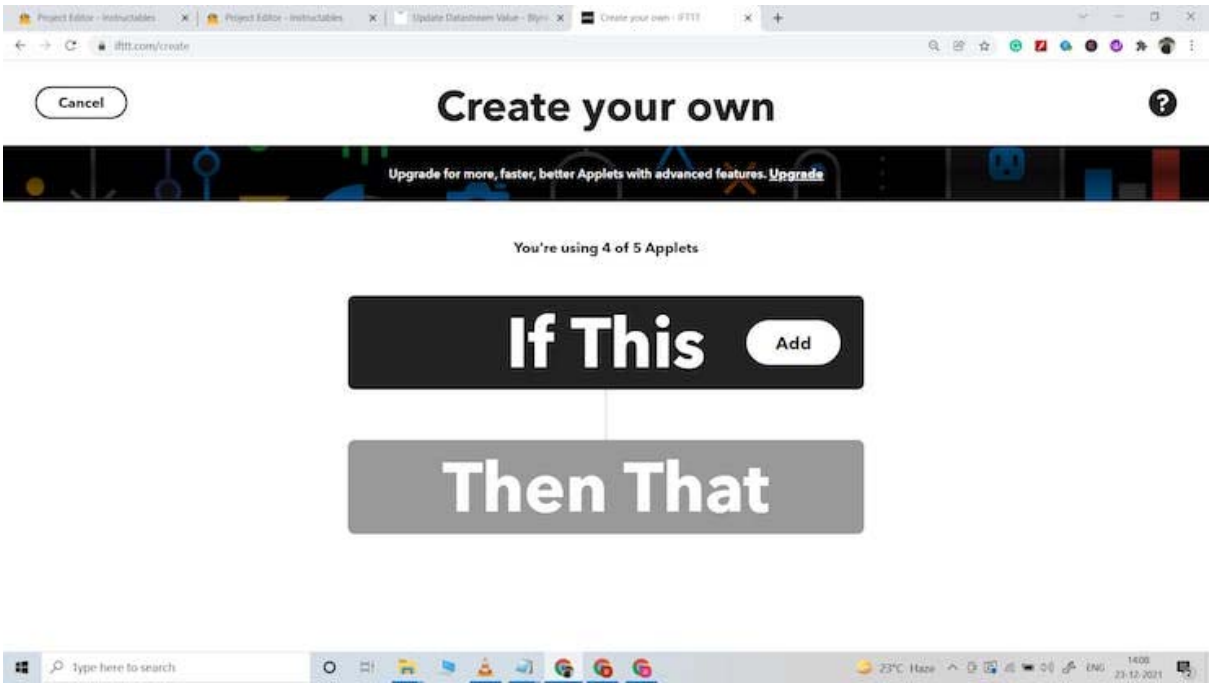


Create an IFTTT account, then log in.https://ifttt.com/

In the FREE plan, user can create 5 Applets. To control each relay you need 2 Applets, so to control 2 relays you need 4 Applets.

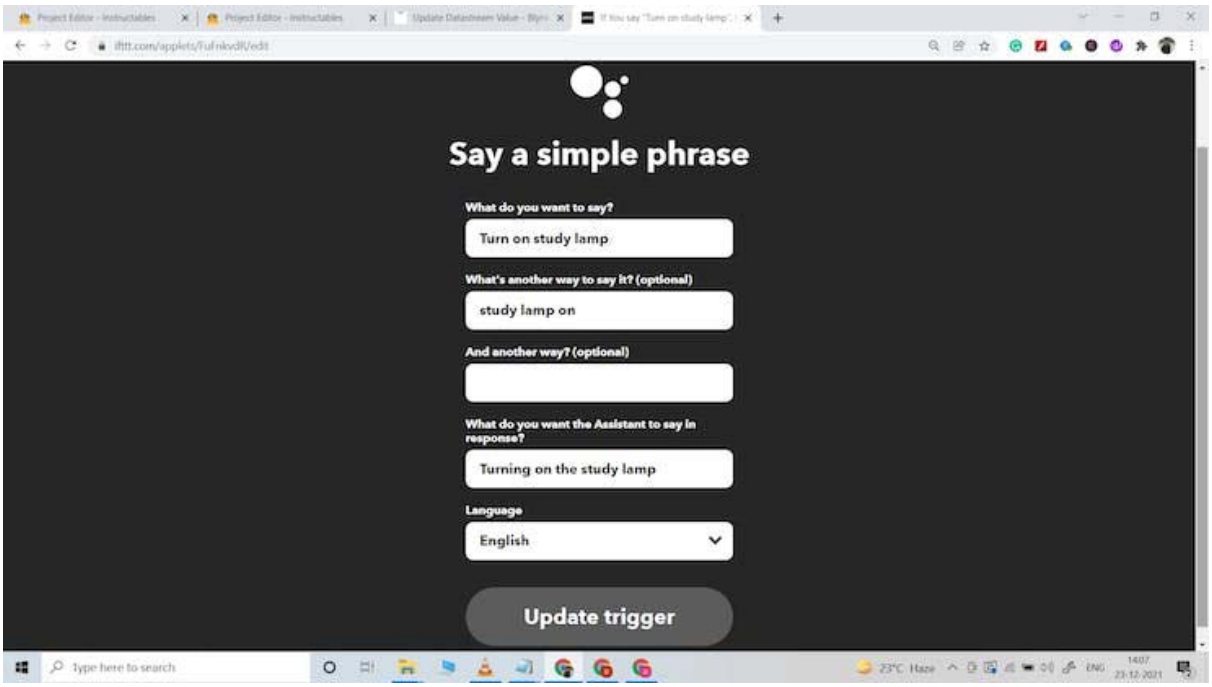
For each applet, user have to define a trigger and related action. In this project, if user says any pre-define commands in Google assistant, then the related Webhook request will be sent to the Blynk Cloud server.

Create Google Assistant Trigger for Applet in IFTTT



Steps to add Google Assistant Trigger in Applet

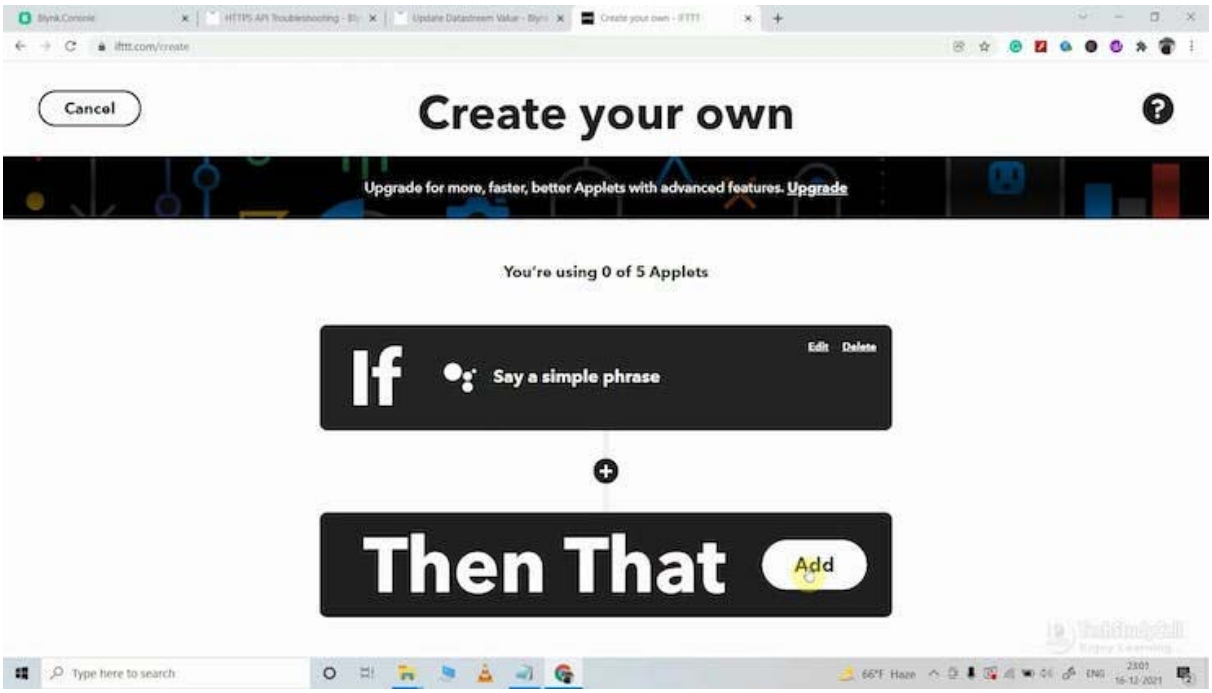
- Click on Create (on the top).
- Click on Add.
- Search for "Google Assistant" and click on it.
- Click on "Say a Simple phrase".
- Click on Connect and give the required permission.



- Then enter "What you want to say" and "response" for Google Assistant (as shown in the picture).

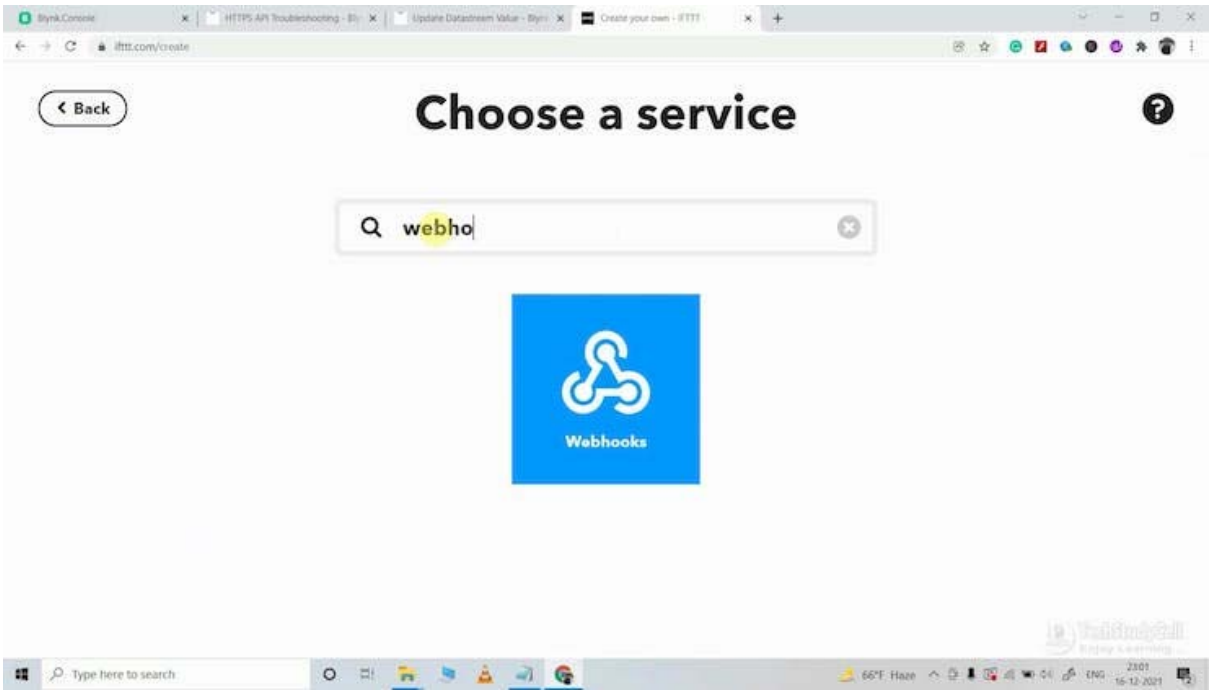
- Click on "Create trigger".

Create Webhooks Action for Applet in IFTTT



Here in the action, we will add Webhooks to send web requests to update the Datastream value in the Blynk server.

- Now click on the next Add button.



- Search for Webhooks and click on it.
- In Webhooks you have to mention the Blynk API URL.

Please refer to the following Blynk API URL syntax to update the Datastream value.

URL syntax to send web requests in New Blynk IoT platform

Syntax: https://{server_address}/external/api/update?token={token}&{pin}={value}

Example:

https://blr1.blynk.cloud/external/api/update?token=jptnDO4nPjlJvJtswlgMXB6BHdOhj9V6&v1=1

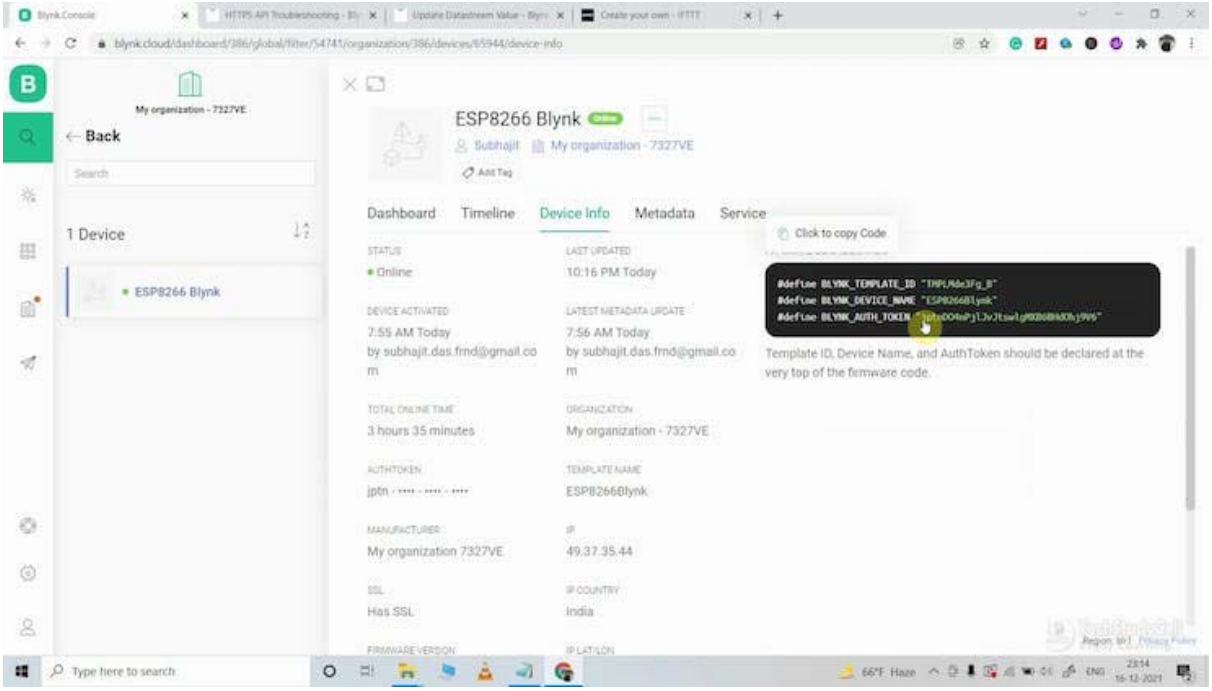
{server_address} {token} {pin} {value}

The **server region** could be found in the right bottom corner of the web interface in the Blynk account.

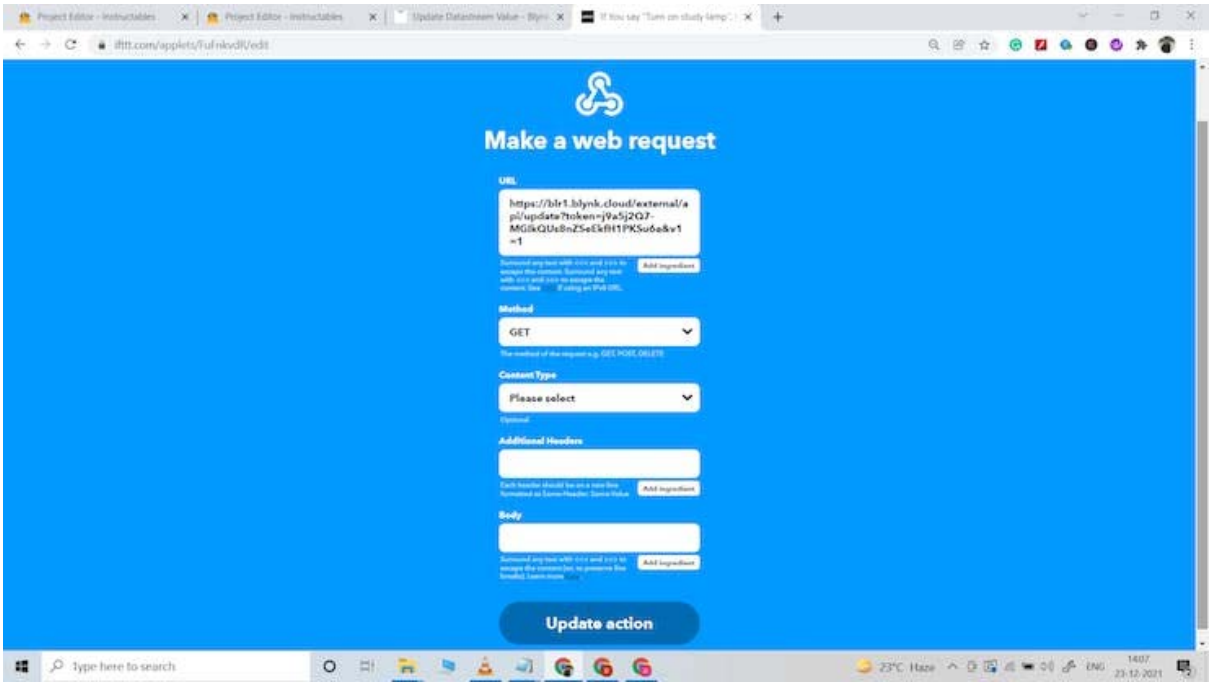
Syntax: https://{server_address}/external/api/update?token={token}&{pin}={value}

- fra1.blynk.cloud – Frankfurt
- lon1.blynk.cloud – London
- ny3.blynk.cloud – New York
- sgp1.blynk.cloud – Singapore
- blr1.blynk.cloud – Bangalore

The server region could be found in the right bottom corner of the web interface.



You can get the Auth Token from the Device Info tab. (Refer to tutorial video on top)



Now, enter the related Blynk URL to update the Datastream value. The method will be "GET". Keep other details as it is. (Refer to the picture)

- Click on Create Action.
- Click on Continue.
- Click on Finish.

If the ESP32 is connected with Wi-Fi then you can monitor the real-time feedback in the Blynk IoT App.

Appendix C

(Software Coding)

```
#define BLYNK_TEMPLATE_ID           "TMPLqUGcc6I7"
#define BLYNK_DEVICE_NAME           "AG IoT Switch"
#define BLYNK_AUTH_TOKEN            "E73pXWQ2I23F7Al3HnZemyX6W97sSbTD"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

char auth[] = BLYNK_AUTH_TOKEN;
#define APP_DEBUG

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin V1
BLYNK_WRITE(V0)
{
  if(param.asInt()==1){
    digitalWrite(25, HIGH);
    display.clearDisplay();
    display.setCursor(0,0);
    display.drawRoundRect(0, 0, 128, 64, 8, WHITE);
    display.drawRoundRect(5, 5, 118, 54, 8, WHITE);
    // Sets the color to black with a white background
    display.setTextColor(WHITE);
    display.setTextSize(1);
    display.setCursor(25,9);
    display.println("AG IoT Switch");
    display.drawLine(6,19,121,19, WHITE);
    display.setCursor(17,22);
    display.setTextSize(2);
    display.println("Relay ON");
    display.drawLine(6,40,121,40, WHITE);
    display.setCursor(33,46);
    display.setTextSize(1);
    display.println("R C C I I T");
    display.display();
  }
  else{
    digitalWrite(25, LOW); // assigning incoming value from pin V1 to a variable
    display.setTextColor(BLACK);
    display.setCursor(17,22);
    display.setTextSize(2);
    display.println("Relay ON");
    display.setTextColor(WHITE);
    display.setCursor(13,22);
    display.setTextSize(2);
    display.println("Relay OFF");
    display.display();
  }
```

```
  // process received value
}

void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(25,OUTPUT);
  Blynk.begin(auth, ssid, pass);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3D (for the 128x64)

  display.clearDisplay();
  display.setCursor(0,0);
  display.drawRoundRect(0, 0, 128, 64, 8, WHITE);
  display.drawRoundRect(5, 5, 118, 54, 8, WHITE);
  // Sets the color to black with a white background
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(25,9);
  display.println("AG IoT Switch");
  display.drawLine(6,19,121,19, WHITE);
  //display.setCursor(23,21);
  //display.println("Please connect");
  //display.setCursor(32,31);
  //display.println("the wifi...");
  display.drawLine(6,40,121,40, WHITE);
  display.setCursor(33,46);
  display.println("R C C I I T");
  display.display();

  delay (500);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
}

void loop()
{
  Blynk.run();
}
```

Appendix D

(Data Sheets)

ESP32 Datasheet

Espressif Systems

October 8, 2016

About This Guide

This document provides introduction to the specifications of ESP32 hardware.

The document structure is as follows:

Chapter	Title	Subject
Chapter 1	Overview	An overview of ESP32, including featured solutions, basic and advanced features, applications and development support
Chapter 2	Pin Definitions	Introduction to the pin layout and descriptions
Chapter 3	Functional Description	Description of the major functional modules
Chapter 4	Peripheral Interface	Description of the peripheral interfaces integrated on ESP32
Chapter 5	Electrical Characteristics	The electrical characteristics and data of ESP32
Chapter 6	Package Information	The package details of ESP32
Chapter 7	Supported Resources	The related documents and community resources for ESP32
Appendix	Touch Sensor	The touch sensor design and layout guidelines

Release Notes

Date	Version	Release notes
2016.08	V1.0	First release

Disclaimer and Copyright Notice

Name	No.	Type	Function
SD_DATA_0	32	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SD_DATA_1	33	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
VDD3P3_CPU			
GPIO5	34	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
GPIO18	35	I/O	GPIO18, VSPICLK, HS1_DATA7
GPIO23	36	I/O	GPIO23, VSPID, HS1_STROBE
VDD3P3_CPU	37	P	CPU IO power supply input (1.8V - 3.3V)
GPIO19	38	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
GPIO22	39	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
U0RXD	40	I/O	GPIO3, U0RXD, CLK_OUT2
U0TXD	41	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
GPIO21	42	I/O	GPIO21, VSPIHD, EMAC_TX_EN
Analog			
VDDA	43	I/O	Analog power supply (2.3V - 3.6V)
XTAL_N	44	O	External crystal output
XTAL_P	45	I	External crystal input
VDDA	46	P	Digital power supply for PLL (2.3V - 3.6V)
CAP2	47	I	Connects with a 3 nF capacitor and 20 kΩ resistor in parallel to CAP1
CAP1	48	I	Connects with a 10 nF series capacitor to ground

2.3 Power Scheme

ESP32 digital pins are divided into three different power domains:

- VDD3P3_RTC
- VDD3P3_CPU
- VDD_SDIO

VDD3P3_RTC is also the input power supply for RTC and CPU. **VDD3P3_CPU** is also the input power supply for CPU.

VDD_SDIO connects to the output of an internal LDO, whose input is **VDD3P3_RTC**. When **VDD_SDIO** is connected to the same PCB net together with **VDD3P3_RTC**; the internal LDO is disabled automatically.

The internal LDO can be configured as 1.8V, or the same voltage as **VDD3P3_RTC**. It can be powered off via software to minimize the current of Flash/SRAM during the Deep-sleep mode.

Note:
It is required that the power supply of **VDD3P3_RTC**, **VDD3P3_CPU** and analog must be stable before the pin **CHIP_PU** is set at high level.

2.4 Strapping Pins

ESP32 has 6 strapping pins:

- MTDI/GPIO12: internal pull-down
- GPIO0: internal pull-up
- GPIO2: internal pull-down
- GPIO4: internal pull-down
- MTDO/GPIO15: internal pull-up
- GPIO5: internal pull-up

Software can read the value of these 6 bits from the register "GPIO_STRAPPING".

During the chip power-on reset, the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device boot mode, the operating voltage of VDD_SDIO and other system initial settings.

Each strapping pin is connected with its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impendence, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or apply the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset, the strapping pins work as the normal functions pins.

Refer to Table 2 for detailed boot modes configuration by strapping pins.

Table 2: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3V		1.8V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Debugging Log on U0TXD During Booting					
Pin	Default	U0TXD Toggling		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Input Falling-edge Output	Falling-edge Input Rising-edge Output	Rising-edge Input Falling-edge Output	Rising-edge Input Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:
Firmware can configure register bits to change the setting of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.

3. Functional Description

This chapter describes the functions implemented in ESP32.

3.1 CPU and Memory

3.1.1 CPU

ESP32 contains two low-power Xtensa® 32-bit LX6 microprocessors with the following features.

- 7-stage pipeline to support the clock frequency of up to 240 MHz
- 16/24-bit Instruction Set provides high code-density
- Support Floating Point Unit
- Support DSP instructions, such as 32-bit Multiplier, 32-bit Divider, and 40-bit MAC
- Support 32 interrupt vectors from about 70 interrupt sources

The dual CPUs interface through:

- Xtensa RAM/ROM Interface for instruction and data
- Xtensa Local Memory Interface for fast peripheral register access
- Interrupt with external and internal sources
- JTAG interface for debugging

3.1.2 Internal Memory

ESP32's internal memory includes:

- 448 KBytes ROM for booting and core functions
- 520 KBytes on-chip SRAM for data and instruction
- 8 KBytes SRAM in RTC, which is called RTC SLOW Memory and can be used for co-processor accessing during the Deep-sleep mode
- 8 KBytes SRAM in RTC, which is called RTC FAST Memory and can be used for data storage and main CPU during RTC Boot from the Deep-sleep mode
- 1 Kbit of EFUSE, of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID

3.1.3 External Flash and SRAM

ESP32 supports 4 x 16 MBytes of external QSPI Flash and SRAM with hardware encryption based on AES to protect developer's programs and data.

ESP32 accesses external QSPI Flash and SRAM by the high-speed caches

- Up to 16 MBytes of external Flash are memory mapped into the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.

- Up to 8 MBytes of external Flash/SRAM are memory mapped into the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data read is supported on the Flash and SRAM. Data write is supported on the SRAM.

3.1.4 Memory Map

The structure of address mapping is shown in Figure 3. The memory and peripherals mapping of ESP32 is shown in Table 3.

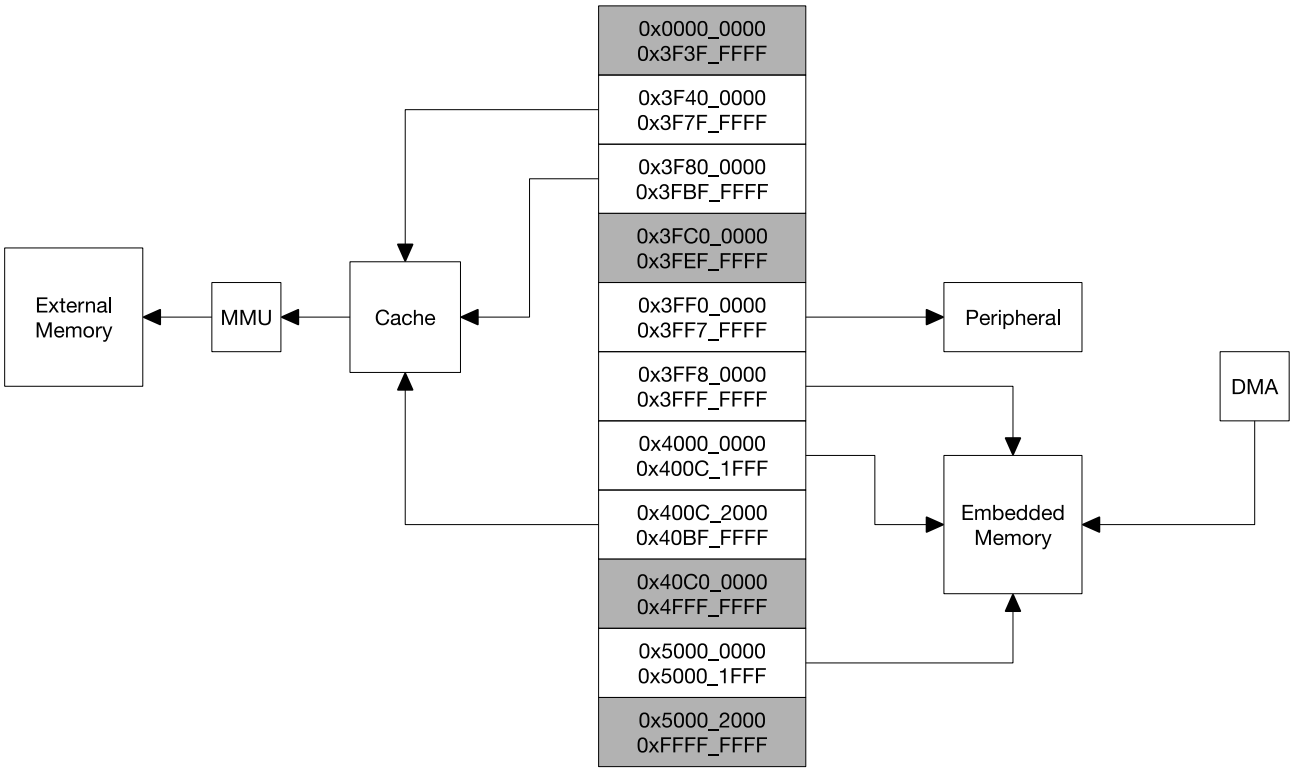


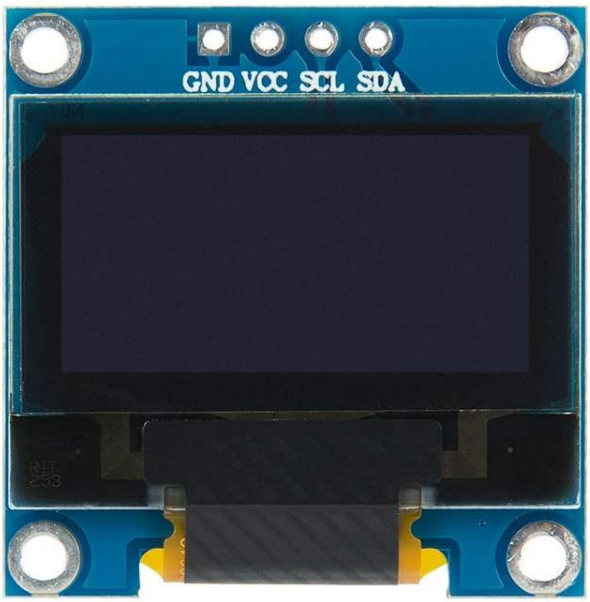
Figure 3: Address Mapping Structure

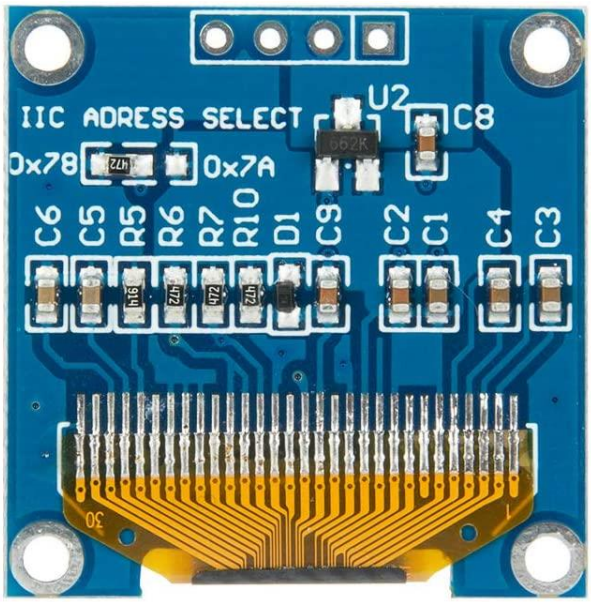
Table 3: Memory and Peripheral Mapping

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
		0x400A_0000	0x400B_FFFF	
	Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB
	RTC FAST Memory	0x3FF8_0000	0x3FF8_1FFF	8 KB
		0x400C_0000	0x400C_1FFF	
	RTC SLOW Memory	0x5000_0000	0x5000_1FFF	8 KB
External Memory	External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
		0x400C_2000	0x40BF_FFFF	11 MB
				248 KB
	External SRAM	0x3F80_0000	0x3FBF_FFFF	4 MB

Category	Target	Start Address	End Address	Size
Peripheral	DPort Register	0x3FF0_0000	0x3FF0_0FFF	4 KB
	AES Accelerator	0x3FF0_1000	0x3FF0_1FFF	4 KB
	RSA Accelerator	0x3FF0_2000	0x3FF0_2FFF	4 KB
	SHA Accelerator	0x3FF0_3000	0x3FF0_3FFF	4 KB
	Secure Boot	0x3FF0_4000	0x3FF0_4FFF	4 KB
	Cache MMU Table	0x3FF1_0000	0x3FF1_3FFF	16 KB
	PID Controller	0x3FF1_F000	0x3FF1_FFFF	4 KB
	UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
	SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
	SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
	GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
	RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
	IO MUX	0x3FF4_9000	0x3FF4_9FFF	4 KB
	SDIO Slave	0x3FF4_B000	0x3FF4_BFFF	4 KB
	UDMA1	0x3FF4_C000	0x3FF4_CFFF	4 KB
	I2S0	0x3FF4_F000	0x3FF4_FFFF	4 KB
	UART1	0x3FF5_0000	0x3FF5_0FFF	4 KB
	I2C0	0x3FF5_3000	0x3FF5_3FFF	4 KB
	UDMA0	0x3FF5_4000	0x3FF5_4FFF	4 KB
	SDIO Slave	0x3FF5_5000	0x3FF5_5FFF	4 KB
	RMT	0x3FF5_6000	0x3FF5_6FFF	4 KB
	PCNT	0x3FF5_7000	0x3FF5_7FFF	4 KB
	SDIO Slave	0x3FF5_8000	0x3FF5_8FFF	4 KB
	LED PWM	0x3FF5_9000	0x3FF5_9FFF	4 KB
	Efuse Controller	0x3FF5_A000	0x3FF5_AFFF	4 KB
	Flash Encryption	0x3FF5_B000	0x3FF5_BFFF	4 KB
	PWM0	0x3FF5_E000	0x3FF5_EFFF	4 KB
	TIMG0	0x3FF5_F000	0x3FF5_FFFF	4 KB
	TIMG1	0x3FF6_0000	0x3FF6_0FFF	4 KB
	SPI2	0x3FF6_4000	0x3FF6_4FFF	4 KB
	SPI3	0x3FF6_5000	0x3FF6_5FFF	4 KB
	SYSCON	0x3FF6_6000	0x3FF6_6FFF	4 KB
	I2C1	0x3FF6_7000	0x3FF6_7FFF	4 KB
	SDMMC	0x3FF6_8000	0x3FF6_8FFF	4 KB
	EMAC	0x3FF6_9000	0x3FF6_AFFF	8 KB
	PWM1	0x3FF6_C000	0x3FF6_CFFF	4 KB
	I2S1	0x3FF6_D000	0x3FF6_DFFF	4 KB
	UART2	0x3FF6_E000	0x3FF6_EFFF	4 KB
	PWM2	0x3FF6_F000	0x3FF6_FFFF	4 KB
	PWM3	0x3FF7_0000	0x3FF7_0FFF	4 KB
	RNG	0x3FF7_5000	0x3FF7_5FFF	4 KB

OLED 4 Pin 128*64 Display Module 0.96” Blue Color





In contrast to LCD technology, Organic Light-Emitting Diode (OLED) displays do not require a backlight and are regarded as the ultimate technology for the next generation of flat-panel displays.

OLED displays are composed of a thin, multi-layered organic film placed between an anode and cathode, which are made up of electric conductive transparent Indium Tin Oxide.

The multi-layered organic film includes a Hole Transporting Layer, Emission Layer and Electron Transporting Layer.

By applying an appropriate electrical voltage, the holes and electrons are injected into the Emission Layer from the anode and cathode respectively and combine to form excitons, after which electroluminescence occurs.

This 0.96” 128*64 Blue OLED Module offers 128*64-pixel resolution. They are featuring much less thickness than LCD Displays with good brightness and produce better and true colors.

This OLED Display Module is very compact and will add a great ever user interface experience to your Arduino project. The connection of this display with Arduino is made through the I2C (also called as IIC) serial interface.

The 0.96” 4 pin 128*64 Blue OLED Display Module produces blue text on black background with very good contrast when supplied with 3.3V-5V Supply. The OLED Display Modules also offers a very wide viewing angle.

FEATURES:

- Supply voltage: 3.3V-5V
- Pixel: 128*64
- Display size- 0.96 inch

- Operating temperature range: -40^{0}C - +80^{0}C
- Use I2C Interface
- Chip No: SSD1306
- Color: Blue
- Drive Duty: 1/64 Duty
- Only need 2 I/O port to control
- Supported platforms: For Arduino,51 series, MSP430 series, STIM32/2, SCR chips
- Super high contrast and brightness(adjustable)
- Low power consumption
- High contrast, thus supporting clear display with no need of backlight
- For OLED SSD1306, a more elaborate and beautiful screen than LCD with more functions

PIN DESCRIPTION:

Pin No.	Pin Name	Description
1.	Supply Voltage (Vcc, 5V)	Can be powered by either 3.3V or 5V
2.	Ground (GND)	Pin Ground
3.	Serial Clock(SCL)	Pin SCL of I2C interface
4.	Serial Data(SDA)	Pin SDA of I2C interface

MECHANICAL SPECIFICATIONS:

ITEM	NORMAL DIMENSION
Module Dimension	27.30*27.30*2.37
Active Area	21.74*10.86
Pixel Size	0.148*0.148
Pixel Pitch	0.17*0.17

ABSOLUTE MAXIMUM RATING:

PARAMETER	SYMBOL	MIN	MAX	UNIT
Supply voltage for logic	VCC	1.65	5.5	V
Operating temperature	TOP	-40	+80	ºC
Storage temperature	TSTG	-40	+80	ºC

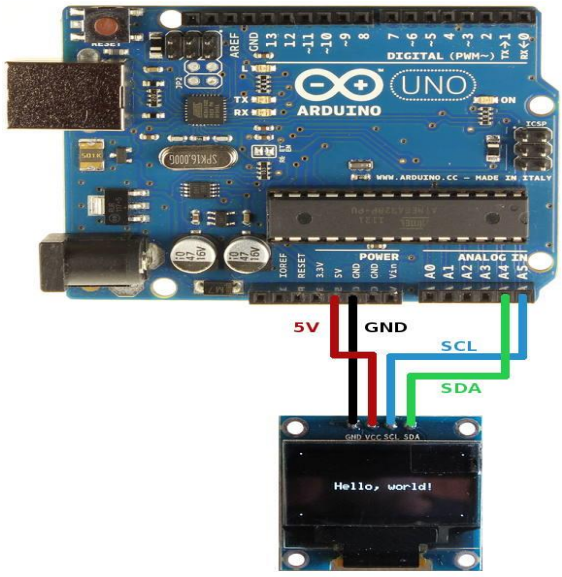
ELECTRONICAL CHARACTERISTICS:

ITEM	SYMBOL	CONDITION	MIN	TYP	MAX	UNIT
Supply voltage for logic	VCC	------	2.8	3.3	5.2	V
Input high voltage	VIH	------	0.8*VCC	-----	VCC	V
Input low voltage	VIL	------	0	-----	0.2*VCC	V
Output high voltage	VOH	------	0.9*VCC	-----	VCC	V
Output low voltage	VOL	------	0	-----	0.1*VCC	V
50%check board operating current	ICC	VCC=3.3	------	12.0	20.0	mA

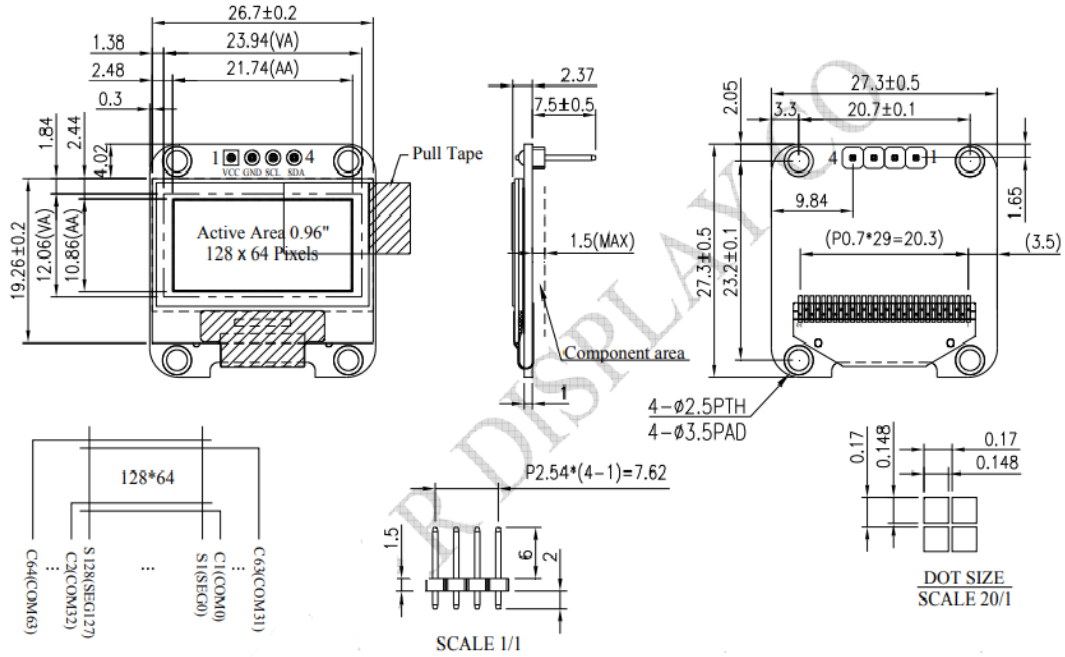
CONNECTION DIAGRAM OF ARDUINO UNO TO 4 PIN 0.96 INCH I2C OLED DISPLAY TO ARDUINO UNO:

Arduino Uno OLED Wiring
The image below shows how to connect the 0.96inch OLED I2C display to Arduino. Pin connections are as follows for wiring the OLED display to an Arduino Uno.

- OLED GND – Arduino GND
- OLED VCC – Arduino 5V
- OLED SCL – Arduino Uno A5
- OLED SDA – Arduino Uno A4



OUTER DIMENSION:



APPLICATIONS:

Due to its capability in displaying, it is often used in various application for instances, smart watch, MP3, function cellphone, portable health device and many others.