MAJOR PROJECT SYNOPSIS

ON

"Smart Houses"

A Major Project Synopsis submitted In partial fulfillment of the award of the requirements for the degree of Bachelor of Technology in " Computer Science & Engineering"

Submitted By:

Anant Sharma (PU17-55004)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PACIFIC UNIVERSITY, UDAIPUR-313001 (RAJASTHAN)

OBJECTIVE

My project aims to produce a system that can integrate with current household appliances and electronic devices and bridge our homes to the internet with advanced controlling and optimizing features.

There are several methods with which one can proceed with this project.

Many hardware and software platforms are available to support developers and innovators. These technologies are primarily for prototyping purposes which may be implemented on an industrial scale in a much better and more efficient way.

For my project, I choose the Arduino platform because of its availability and flexibility. It provided me the advantage of bridging computer science and electronics engineering, at moments hiding the complexity in Computer science and vice versa.

As we proceed, we will get to know more and more benefits of using such technology and how it can make our lives easier.

We will also discuss the future of the project, how it could be further developed, and how the outcome project would benefit both the developer and the user. Hopefully, We will manage to answer most of the questions by the end of my project.

- **HARDWARE USED**

    - **Arduino interface board** - provide the engineers, artists, designers, hobbyists, and anyone who tinkers with technology with a low-cost, easy-to-use technology to create their creative, interactive objects, practical projects, etc.; a whole new breed of projects can now be built that can be controlled from a computer. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. It's an open-source physical computing platform based on a microcontroller board, and a development environment for writing software for the board. In simple words, Arduino is a small microcontroller board with a USB plug to connect to your computer and many connection sockets that can be wired up to external electronics, such as motors, relays, light sensors, laser diodes, loudspeakers, microphones, etc., They can either be powered through the USB connection from the computer or a 9V battery. They can be controlled or programmed by the computer, disconnected, and allowed to work independently.

    - **Arduino Ethernet Shield** -The Arduino Ethernet Shield V1 connects your Arduino to the internet in minutes. Just plug this module onto your Arduino board, connect it to your network with an RJ45 cable (not included) and follow a few simple instructions to start controlling your world through the internet. As always with Arduino, every element of the platform – hardware, software, and documentation – is freely available and open-source. This means you can learn exactly how it's made and use its design as the starting point for your circuits. Hundreds of thousands of Arduino boards are already fueling people's creativity daily. Join us now, Arduino is you!
        - ➢ Requires an Arduino board (not included)
        - ➢ Operating voltage 5V (supplied from the Arduino Board)
        - ➢ Ethernet Controller: W5100 with internal 16K buffer
        - ➢ Connection speed: 10/100Mb
        - ➢ Connection with Arduino on SPI port
    - **ESP8266 WiFi Module**-The ESP8266 is a low-cost Wi-Fi microchip with a full TCP/IP stack and microcontroller capability, produced by Espressif Systems in Shanghai, China. The chip first came to the attention of Western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker.
    - **Arduino relay**-A relay is an electrically operated switch that can be turned on or off, letting the current go through or not, and can be controlled with low voltages, like the 5V provided by the Arduino pins. Controlling a relay module with the Arduino is as simple as controlling any other output, as we'll see later.

- **SOFTWARE**

    - The software used by the Arduino is Arduino IDE. the Arduino IDE is a cross-platform application written in Java. It is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with syntax highlighting, brace matching, and automatic indentation. It can also compile and upload programs to the board with a single click. There is no need to edit makefiles or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ino. The Arduino IDE comes with a C/C++ library called "Wiring" (from the same name project), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need to define two functions to make a runnable program: setup() – a function run once at the start of a program that can initialize settings loop() – a function called repeatedly until the board powers off.

    - Software written using Arduino is called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment, including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

SMART HOUSES

A PROJECT REPORT

Submitted by

Anant Sharma

in partial fulfillment for the award of the degree of

B. Tech

IN

Computer Science Engineering

FACULTY OF ENGINEERING

PACIFIC UNIVERSITY

AUGUST, 2021

TABLE OF CONTENTS

PART I
INTRODUCTION

CHAPTER 1

1.1 ARDUINO

1.1.1 What is Arduino

Arduino interface boards provide the engineers, artists, designers, hobbyists, and anyone who tinker with technology with a low-cost, easy-to-use technology to create their creative, interactive objects, useful projects, etc., A whole new breed of projects can now be built that can be controlled from a computer.
Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. It's an open-source physical computing platform based on a microcontroller board and a development environment for writing software for the board.

In simple words, Arduino is a small microcontroller board with a USB plug to connect to your computer and a number of connection sockets that can be wired up to external electronics, such as motors, relays, light sensors, laser diodes, loudspeakers, microphones, etc., They can either be powered through the USB connection from the computer or from a 9V battery. They can be controlled from the computer or programmed by the computer and then disconnected and allowed to work independently.

Anyone can buy this device through an online auction site or search engine. Since Arduino is an open-source hardware designs and creates its own clones of the Arduino and sells them, so the market for the boards is competitive. An official Arduino costs about $30, and a clone often less than $20.

The name "Arduino" is reserved by the original makers. However, clone Arduino designs often have the letters "duino" on the end of their name, for example, Freeduino or DFRduino. The software for programming your Arduino is easy to use and freely available for Windows, Mac, and LINUX computers at no cost.

1.1.2 Microcontroller

Microcontroller can be described as a computer embedded on a rather small circuit board. To describe the function of a microcontroller more precisely, it is a single chip that can perform various calculations and tasks, and send/receive signals from other devices via the available pins. Precisely what tasks and communication with the world it does, is what is governed by what instructions we give to the Microcontroller. It is this job of telling the chip what to do, which is what we refer to as programming on it.

However, the uC by itself, cannot accomplish much; it needs several external inputs: power, for one; a steady clock signal, for another. Also, the job of programming it has to be accomplished by an external circuit. So typically, a uC is used along with a circuit that provides these things to it; this combination is called a microcontroller board. The Arduino Uno that you have received, is one such microcontroller board. The actual microcontroller at its heart is the chip called **Atmega328**. The advantages that Arduino offers over other microcontroller boards are largely in terms of the reliability of the circuit hardware as well as the ease of programming and using it.

1.1.3 Open Source Hardware

Open-source hardware shares much of the principles and approaches of free and open-source software. The founders of Arduino wanted people to study their hardware, understand how it works, make changes to it, and share those changes with the world. To facilitate this, they release all of the original design files(Eagle CAD)for the Arduino hardware. These files are licensed under a Creative Common Attribution Share-Alike license, which allows for both personal and commercial derivative works, as long as they(people) credit Arduino and release their designs under the same license.

The Arduino software is also open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL

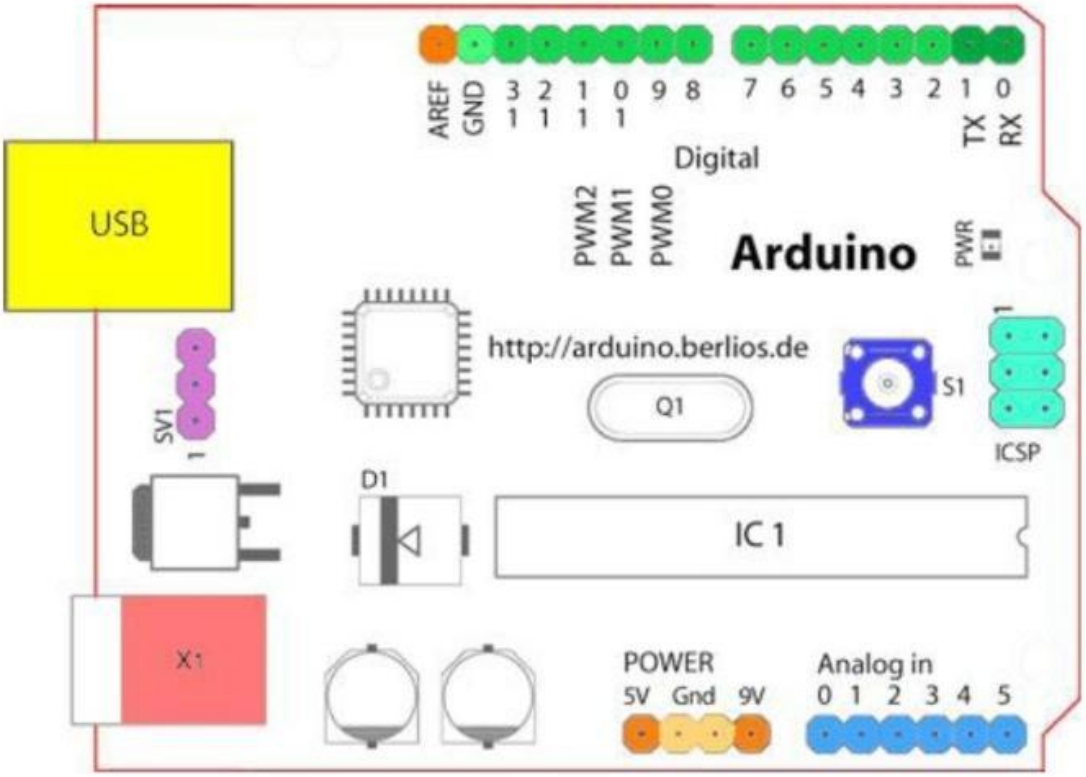
1.1.4 Community

As the project is aimed at students and hobbyists who may not have any formal electronics background, there are many excellent guides online covering everything from making a light blink to creating a laser harp. The official forum has almost 60 000 registered users, and along with helping users with their projects, is extremely active in developing new libraries to extend the functionality of the Arduino. The open-source share and share alike sentiment is very strong, and the vast majority of users freely publish the code to their projects.

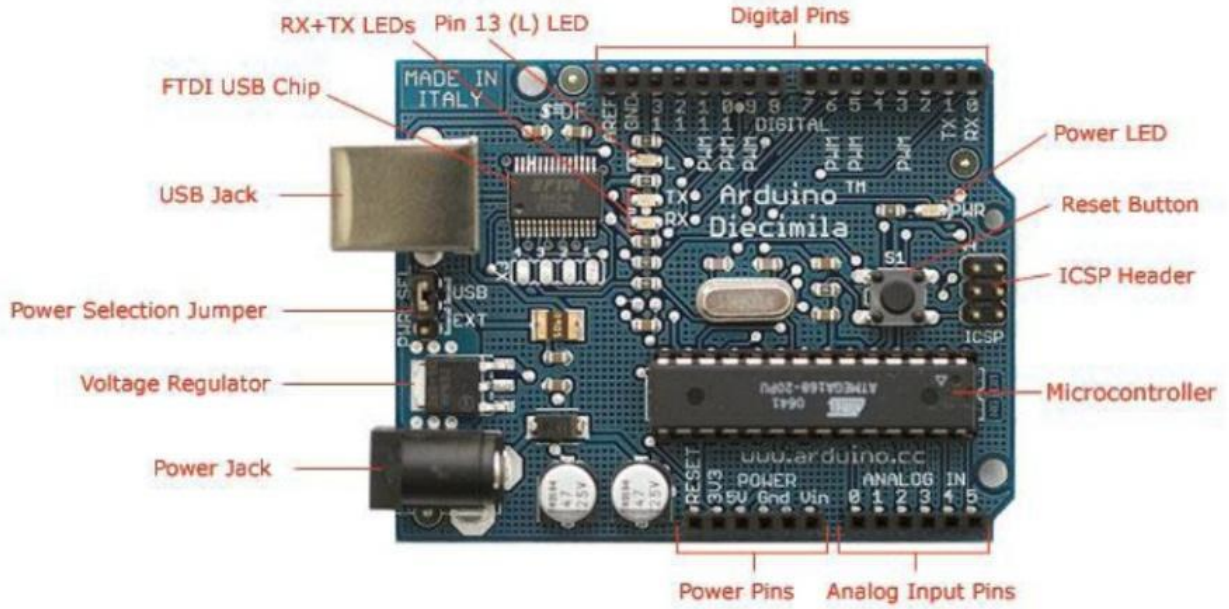
CHAPTER 2

2.1 PLATFORM

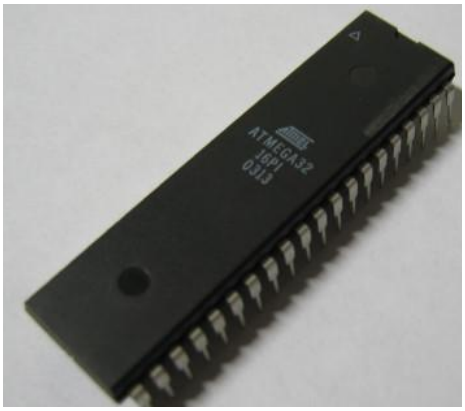
2.1.1 HARDWARE



BASIC LAYOUT OF ARDUINO BOARD & PIN DIAGRAM



2.1.2 ATMEGA 32 (MICROCONTROLLER)



FLASH MEMORY - 32 KB
FLASH TYPE - DYNAMIC (CAN ME FLASHED MULTIPLE TIMES)

2.1.3 Single chip USB to async. Serial data transfer interface

- USB 2.0 compatible
- Transmit and receive LED drive signals
- 256 Byte receive,128 Byte transmit buffer
- Data transfer rate from 300bits/sec to 2 Mb/sec

2.2 FAMOUS VARIANTS

2.2.1 ARDUINO UNO

This is the latest revision of the basic Arduino USB board. It connects to the computer with a standard USB cable and contains everything else you need to program and use the board. It can be extended with a variety of shields: custom daughter-boards with specific features. It is similar to the Duemilanove but has a different USB-to-serial chip the ATMega8U2, and newly designed labeling to make inputs and outputs easier to identify.

2.2.2 ARDUINO MEGA

A larger, more powerful Arduino board. Has extra digital pins, PWM pins, analog inputs, serial ports, etc. *The version of the Mega released with the Uno, this version features the Atmega2560, which has twice the memory, and uses the ATMega 8U2 for USB-to-serial communication.*

CHAPTER 3

3.1 Node MCU

3.1.1 Overview

NodeMCU is an open-source Lua-based firmware for the ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif NON-OS SDK.

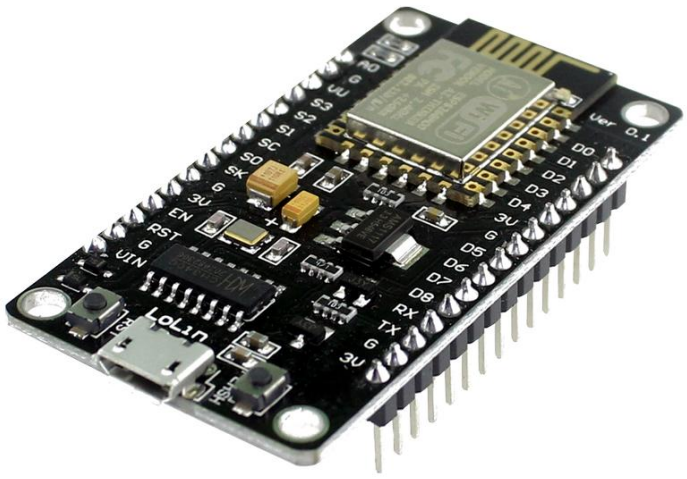
The firmware was initially developed as a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now be run on *any* ESP module.

The firmware uses the Lua scripting language. The firmware is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson[9] and SPIFFS.[10] Due to resource constraints, users need to select the modules relevant for their project and build firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects).

3.1.2 Types of Node MCU

There are two available versions of NodeMCU version 0.9 & 1.0 where the version 0.9 contains **ESP-12** and version 1.0 contains **ESP-12E** where E stands for "Enhanced"



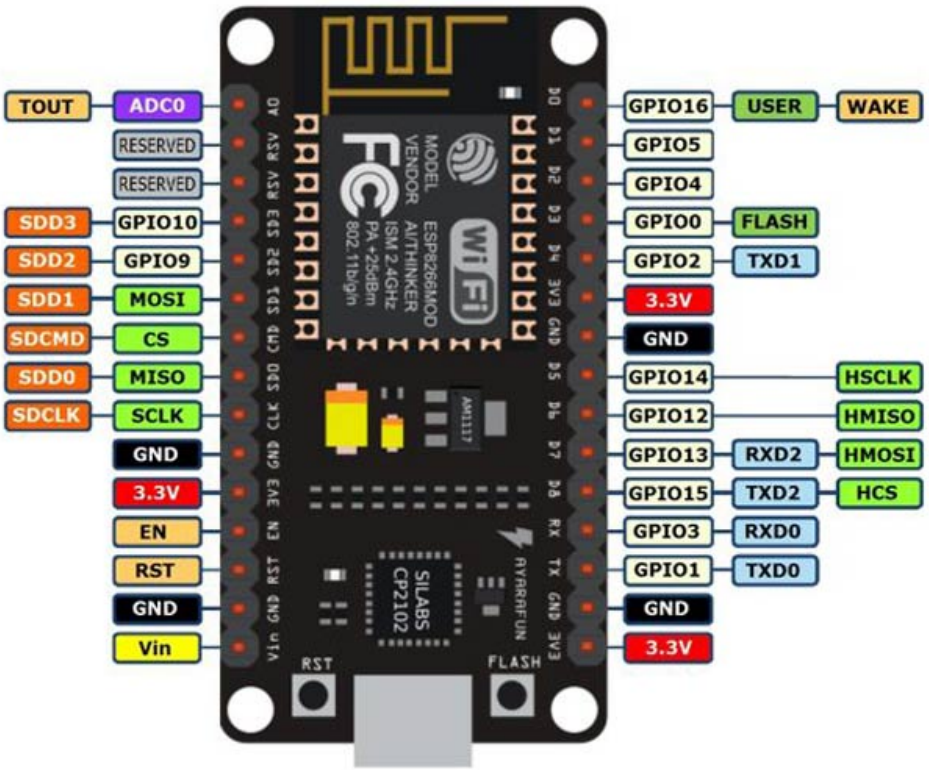
3.1.3 How to start with NodeMCU?

NodeMCU Development board is featured with wifi capability, analog pin, digital pins, and serial communication protocols.

To get started with using NodeMCU for IoT applications first we need to know how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

There are online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement.

3.1.4 NodeMCU Development Board Pinout Configuration



Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

3.1.5 NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

3.1.6 Programming NodeMCU ESP8266 with Arduino IDE

The NodeMCU Development Board can be easily programmed with Arduino IDE since it is easy to use. Programming NodeMCU with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable, and the NodeMCU board itself.

3.1.7 Uploading your first program

Once Arduino IDE is installed on the computer, connect the board with the computer using the USB cable. Now open the Arduino IDE and choose the correct board by selecting Tools>Boards>NodeMCU1.0 (ESP-12E Module), and choose the correct Port by selecting Tools>Port. To get it started with the NodeMCU board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code is loaded into your IDE, click on the 'upload' button given on the top bar. Once the upload is finished, you should see the built-in LED of the board blinking.

3.1.8 Applications of NodeMCU

- Prototyping of IoT devices
- Low power battery operated applications
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

CHAPTER 4

4.1 BASIC TERMINOLOGIES IN ARDUINO

4.1.1 Terms

microcontroller: this is a small computer designed for interacting with physical hardware; there's probably one in your microwave and your alarm clock

Arduino: this is a board with a microcontroller built in. It makes it easier to work with the microcontroller. There are several types

of Arduino Uno*:* this is the type of Arduino we show below. It's the most common type.

shield*:* this is a board that can be stacked on top of an Arduino; There are many kinds of Arduino shields

pin*:* an electrical connection to a microcontroller

digital pin*:* this pin can be used to read/write on/off (binary) values (typically 0V or 5V for an Arduino)

analog pin*:* this pin can be used to read voltages (typically 0V to 5V for an Arduino)

sketch: this is a program that runs on an Arduino

Grove: this is a hardware product line from Seeed Studio that makes it easy to connect sensors and other devices to an Arduino

WireGarden*:* this is a software system for communicating with an Arduino from your web browser

4.1.2 Analog to digital converter(ADC)

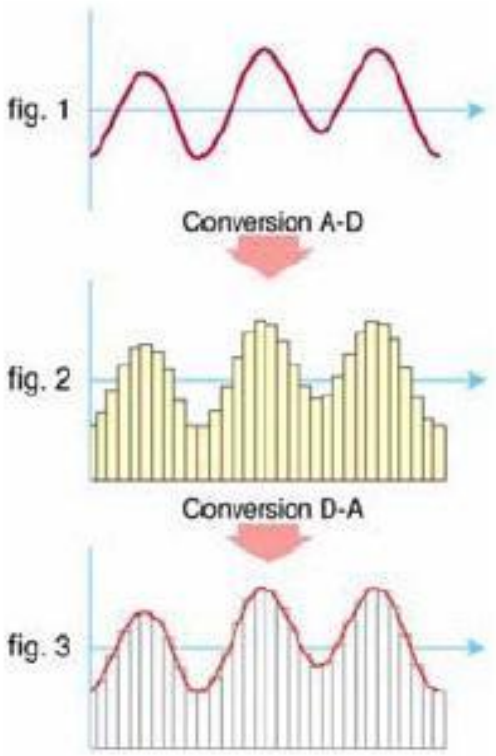
The process of Analog to digital conversion is shown in the figure.
The Arduino has 10 bits of Resolution when reading analog signals.
2 power 10=1024 increments
Influence also by how fast you sample

4.2 Pulse width modulation (PWM)

The Arduino has 8bit of resolution when outputting a signal using PWM. The range of output voltage is from 0 to 5 Volts

2power 8=255 Increments

Average of on/off(digital signals to make an average voltage), Duty cycle in 100% of 5Volts.



CHAPTER 5

5.1 LANGUAGE REFERENCES:

The Microcontroller on the board is programmed using the **Arduino programming language**(based on **wiring**) and the **Arduino development environment**(based on **processing**).

5.1.1 Arduino Programming Language(APL)(based on wiring)

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

5.1.2 Wiring

Wiring is an open-source programming framework for microcontrollers. Wiring allows writing cross-platform software to control devices attached to a wide range of microcontroller boards to create all kinds of creative coding, interactive objects, spaces or physical experiences. The framework is thoughtfully created with designers and artists in mind to encourage a community where beginners through experts from around the world share ideas, knowledge, and their collective experience. There are thousands of students, artists, designers, researchers, and hobbyists who use Wiring for learning, prototyping, and finished professional work production.

5.2 ARDUINO DEVELOPMENT ENVIRONMENT BASED ON PROCESSING

5.2.1 Processing

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

5.2.2 Software

The software used by the Arduino is Arduino IDE.

The Arduino IDE is a cross-platform application written in Java and is derived from the IDE for the Processing programming language and the Wiring Project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ino.

The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need to define two functions to make a runnable program:

setup() – a function run once at the start of a program that can initialize settings

loop() – a function called repeatedly until the board powers off

CHAPTER 6

6.1 ARDUINO DEVELOPMENT ENVIRONMENT

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

6.1.1 Writing Sketches

Software written using Arduino is called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment, including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the IDE before 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify Checks your code for errors.

Upload Compiles your code and uploads it to the Arduino I/O board.

See uploading below for details.Note: If you are using an external programmer, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New Creates a new sketch.

Open Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save Saves your sketch.

Serial Monitor Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive which means only those items relevant to the work currently being carried out are available.

Edit

- *Copy for Forum* Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy as HTML* Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

Sketch

Verify/Compile Checks your sketch for errors.

Show Sketch Folder Opens the current sketch folder.

Add File... Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu.

Import Library Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see the libraries below. Additionally, with version 1.0.5 and later of the IDE, you can import a library from a .zip file.

Tools

Auto Format This formats your code nicely: i.e., indents it so that opening and closing curly braces line up and that the statements inside curly braces are indented more.

Archive Sketch Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Board Selects the board that you're using. See below for descriptions of the various boards.

Serial Port This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Programmer For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

Burn Bootloader The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino board but is useful if you purchase a new ATmega microcontroller (which normally comes without a bootloader). Ensure you've selected the correct board from the Boards menu before burning the bootloader.

Sketchbook

The Arduino environment uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

'"Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

6.1.2 Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h).

6.1.3 Uploading

Before uploading your sketch, you must select the correct items from the Tools > Board and Tools > Serial Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyUSB0, /dev/ttyUSB1 or similar.

Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino environment will display a message when the upload is complete or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded onto the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then, it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the onboard (pin 13) LED when it starts (i.e., when the board resets).

6.1.4 Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

To write your own library, see this tutorial.

6.1.5 Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "Arduino" as the sub-directory name, or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the platforms page on the Arduino Google Code developers site.

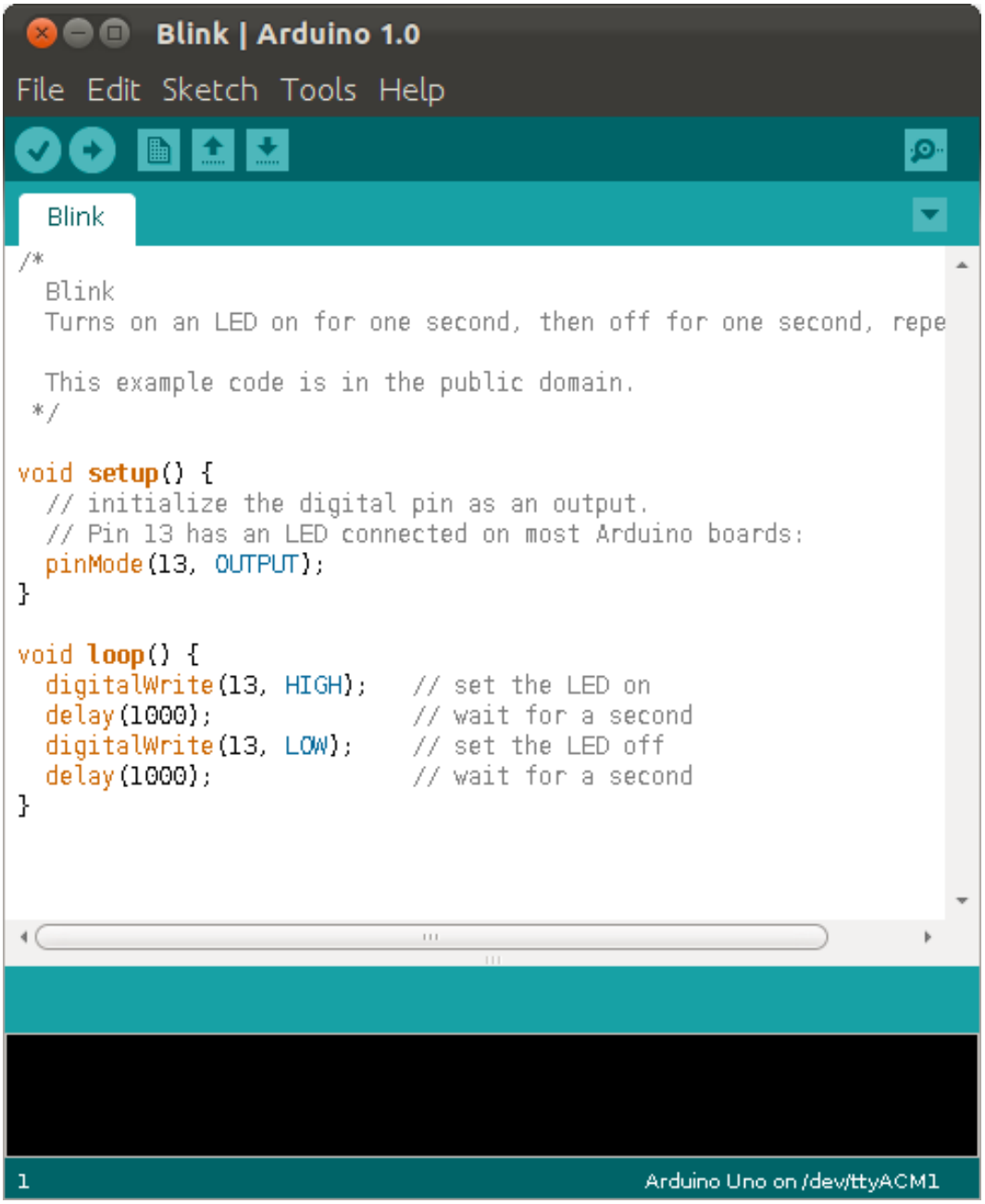
6.1.6 Serial Monitor

Displays serial data being sent from the Arduino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial.begin in your sketch. Note that on Mac or Linux, the Arduino board will reset (rerun your sketch from the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

6.1.7 Preferences

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.


Blink | Arduino 1.0
File Edit Sketch Tools Help
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repe
This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}
1
Arduino Uno on /dev/ttyACM1

CHAPTER 7

7.1 WHAT IS BLYNK

Blynk is an “Internet of Things” (IoT) platform that allows you to build your own apps to control certain devices over the internet. Blynk supports hardware platforms such as Arduino, Raspberry Pi, and similar microcontroller boards to build hardware for your projects. Blynk started off as a Kickstarter campaign in 2015, where it raised a total of 2,321 backers and raised $49,235 on a goal of just $10,000 (see the Kickstarter project here). Since then, the company has gained popularity year after year. It can control hardware remotely, it can display sensor data, can store data, visualize it, and do many other cool things.

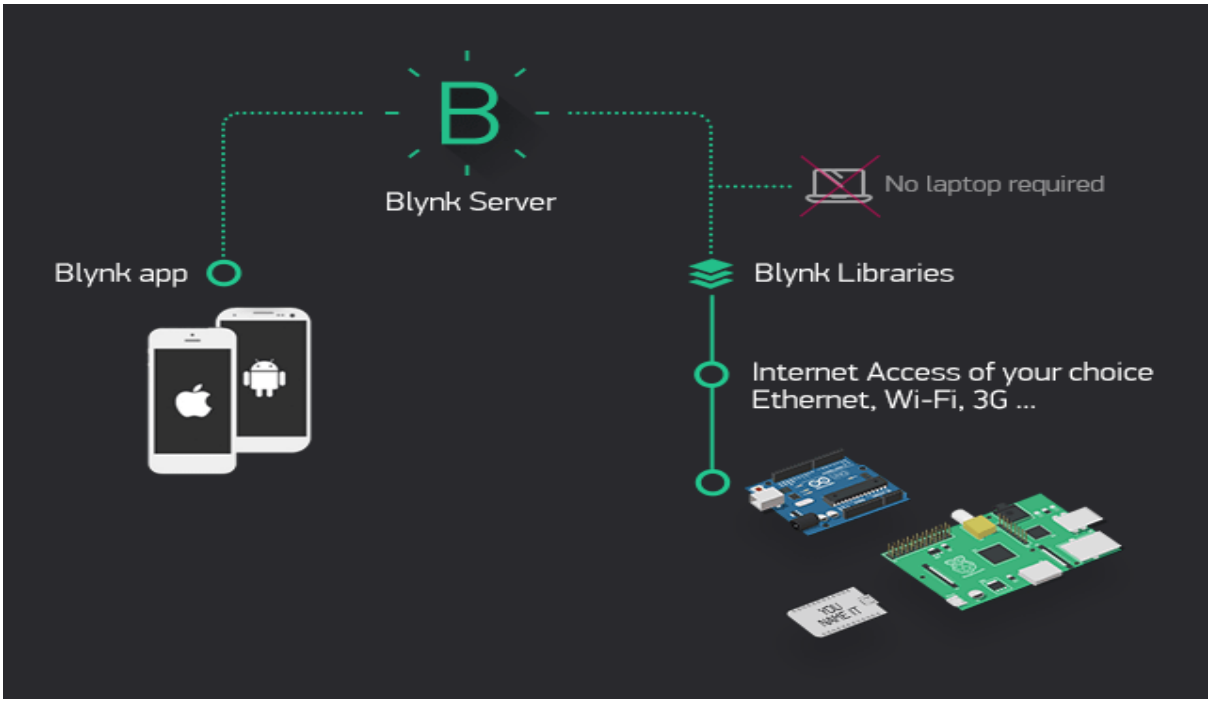
7.1.1 How Blynk Works

There are three major components in the platform:
Blynk App - allows to you create amazing interfaces for your projects using various widgets we provide.
Blynk Server - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It’s open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to space the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.

Blynk has several different groups of users:

1. their own commercial app based on BlynkHobbyists and electronics geeks who just want to play around with this technology
2. Individuals who want to make their life easier in some way by being able to control and monitor their Blynk-compatible devices through their phone
3. Companies who want to launch



7.1.2. CONNECTION TYPE

Blynk supports the following connection types to connect your microcontroller board (hardware) with the Blynk Cloud and Blynk's personal server:

- **Ethernet**
- **Wi-Fi**
- **Bluetooth**
- **Cellular**
- **Serial**

7.1.3 Blynk Architecture

The Blynk platform includes the following components:

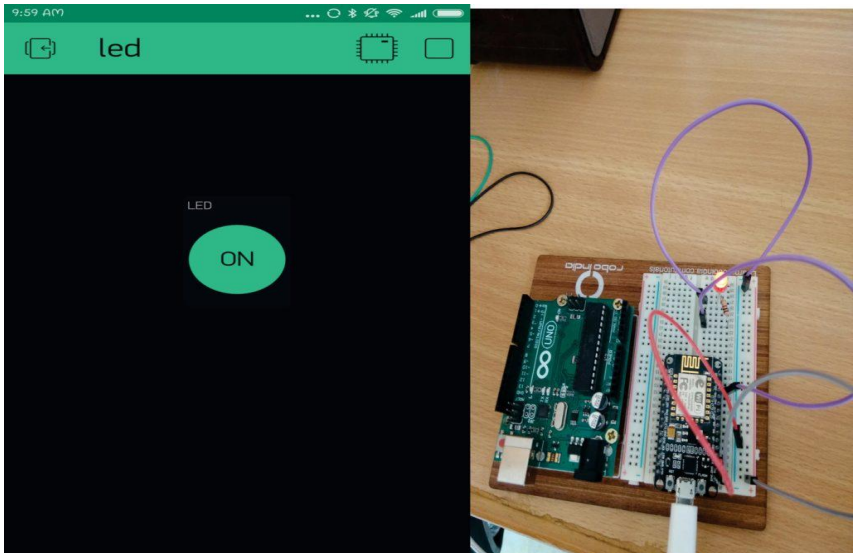
- Blynk app builder: Allows to you build apps for your projects using various widgets. It is available for Android and iOS platforms.
- Blynk server: Responsible for all the communications between your mobile device that's running the Blynk app and the hardware. You can use the Blynk Cloud or run your private Blynk server locally. It's open source, could easily handle thousands of devices, and can even be launched on a Raspberry Pi.
- Blynk libraries: Enables communication with the server and processes all the incoming and outcoming commands from your Blynk app and the hardware. They are available for all the popular hardware platforms.

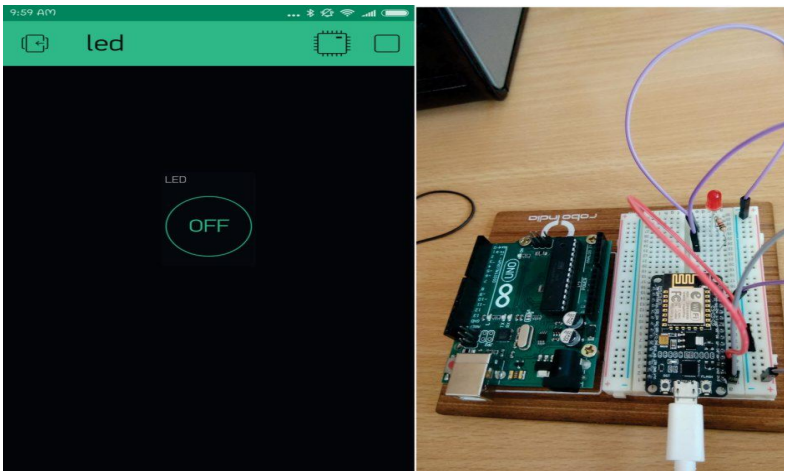
1. Hardware.

An Arduino, Raspberry Pi, or a similar development kit. Blynk works over the Internet. This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, Raspberry Pi with WiFi dongle, Particle Photon or SparkFun Blynk Board. But even if you don't have a shield, you can connect it over USB to your laptop or desktop (it's a bit more complicated for newbies, but we got you covered). What's cool, is that the list of hardware that works with Blynk is huge and will keep on growing.

2. A Smartphone.

The Blynk App is a well-designed interface builder. It works on both iOS and Android, so no holy wars here





CHAPTER 8

8.1. IFTTT

IFTTT stands for If This Then That. It is a free service that allows you to connect multiple services to your smart home devices and beyond. It uses simple conditional statements in a simple form to make it easy to create little, custom applets to enhance your life.IFTTT defines a clear and concise protocol that your service's API will implement. Each trigger and action for your service will map 1-to-1 to an API endpoint on your service built specifically for IFTTT. Trigger endpoints will be event streams that IFTTT will poll for new data. Conversely, action endpoints will be writable endpoints that IFTTT will send data. For services that use OAuth, IFTTT is fully compatible with a couple of the common flavors of OAuth 2.0.

8.1.1 How It Works

IFTTT accepts input. Then, based on the rules and guides you set up, perform an action on another system.

EXAMPLE

It's often easiest to understand IFTTT best through examples:

- You're coming home from work. Your phone recognizes that it is near your home and sends a message to IFTTT. IFTTT accepts that message, forwards it to your smart home lighting, and turns on the light in your kitchen and garage, welcoming you home.
- You lock your smart lock. Your smart lock sends a message stating it was locked to IFTTT. IFTTT accepts that message and sends a message to your smart thermostat to turn it down to save on heating costs.
- You're hosting a game night-watching party. Every time your team scores a goal, your smart lights change to match your team's color, making the event much more fun and special.
- The weather report says it'll rain today. About an hour before the rain should start, your smart windows have already closed themselves.

Beyond smart home applications, you can also harness many other sources of information, and the resulting actions can streamline your digital life.

- Every time your company is mentioned on Twitter, your group knows instantly because IFTTT forwards the tweet to your Slack group.
- Automatically mute your smartphone whenever you're in a meeting scheduled on your calendar.
- You're working out but waiting for an essential email message. When the message comes in, your smartphone reads out the sender and title, letting you know to stop your workout and respond.

8.1.2. Personal and Customizable

The whole idea behind IFTTT is that everyone has a different need when it comes to the integration of their various services. Rather than having just a few that may or may not work or work poorly, they allow you to 'write your own' integrated approaches, custom and personal for you.

There are example applets, but they exist more to give you an idea to base your works upon. It can take a bit of thought and careful consideration of your unique wants and needs, but the result is a tailored experience that works best for you.

CHAPTER 9

9.1. WEBHOOKS

Webhooks are one of a few ways web applications can communicate with each other. It allows you to send real-time data from one application to another whenever a given event occurs.
Webhooks are "user-defined HTTP callbacks". They are usually triggered by some event, such as pushing code to a repository or a comment being posted to a blog. When that event occurs, the source site makes an HTTP request to the URL configured for the webhook. What a webhook does is notify you any time someone checks in, so you'd be able to run any processes that you had in your application once this event is triggered.

The data is then sent over the web from the application where the event originally occurred to the receiving application that handles the data.
Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your service and Applets, check out the IFTTT platform.

9.1.1 FORMAT OF WEBHOOK

When users make an initial query or provide some subsequent input, the Assistant sends a request to your fulfillment. Conversation webhook requests from the Assistant contain data such as the intent that was triggered, the raw text of the user input, and the surface capabilities of the user's device.

The key fields for a request in the conversation webhook format are summarized below:

Field	Description
isInSandbox	This boolean variable is primarily used for the transactions feature, to indicate whether your webhook should handle this request in sandbox mode. In sandbox mode, your webhook should not charge or fulfill any purchase orders by users. By default, it is set to "true".
surface	Information about the Assistant surface the user is interacting with and its capabilities.
Inputs	Information about the invocation request. The triggering invocation includes an intent that maps to an action. For subsequent requests in a conversation, this object might also include arguments corresponding to the expected inputs specified by your fulfillment.

User	Information about the user who initiated the request. This information includes permissions granted by the user and the user's locale.
Conversation	Information about the conversation context, including the conversation ID, the type (for example, whether this request is initiating a new conversation), and a conversation token to store persistent data across the conversation lifespan.
availableSurfaces	This information is used for multi-surface conversations.

CHAPTER 10

10.1. GOOGLE ASSISTANT

Google Assistant is Google's voice assistant. When it launched, Google Assistant was an extension of Google Now, designed to be personal while expanding on Google's existing "OK Google" voice controls.

Originally, Google Now smartly pulled out relevant information for you. It knew where you worked, your meetings and travel plans, the sports teams you liked, and what interested you so that it could present you with information that mattered.

Google has long killed Google Now, but Assistant lives in the same space, fusing these personalized elements with a wide range of voice control. Google Assistant supports both text and voice entry, and it will follow the conversation whichever entry method you're using.

10.1.1 What can Google Assistant do(Applications)?

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 recognize voice profiles of 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.

Features include the ability to check in to your flight (airline and destination dependant), as well as the ability to book a room with some partners, and there's an Interpreter Mode on Google Nest devices and smart displays too. With it, you can ask Google Assistant to help you conduct a conversation in dozens of languages. Just say, "Hey Google, be my Spanish interpreter" to start Interpreter Mode and get real time spoken and (on Smart Displays) written translation to aid the conversation.

Google Assistant in Google Nest devices forms the foundation of smart home control. It's compatible with a wide range of devices, so you can control heating, lights, and a lots more with your voice.

To find out everything Google Assistant can do, visit the dedicated page on the Google website

10.1.2 Google Assistant on phones

Google Assistant is available on most Android phones, with all recent launches offering the AI system. Even devices that offer another AI system, like Samsung's Bixby, also offer Google Assistant. Essentially, if your phone has Android, your phone has Google Assistant.

It's possible to have Assistant respond to you even when your Android phone is locked, too, if you opt-in through your settings, and you can also opt in to see answers to personal queries.

Google Assistant is also available on iPhones, although there are some restrictions.

10.1.3 Smart home devices and appliances

A lot of connected devices are compatible with Google Assistant, from lightbulbs to fridges and everything else between. Assistant works with over 1000 home automation brands and more than 10,000 devices.
Google Assistant is also compatible with IFTTT, so custom recipes can be created. As we've said before, you also don't need to talk to Google Home; you can use Assistant on your phone to interact.

PART II

DEVELOPMENT

CHAPTER 11

11.1 GETTING STARTED

11.1.1 Requirements:-

a) arduino uno /mega (mega preferred for greater flexibility)
b) Node MCU
c) Breadboard
d) Jumper wire
e) Wifi connection
f) Relay modules (10A in - controllable with 5V)
g) Bulb and bulb holder
e) AC 220 Volt power supply
f) 12V DC Adapter

11.1.2 Available code:-

https://github.com/anant13sharma/Node-MCU-/tree/main

11.1.3 Procedure:-

- Download Arduino IDE from the website http://arduino.cc/
- Download the code from the link above.
- Normal connections to be done as mentioned in the readme file on git using jumper cables.
- pins D1 and D2 as outputs (LEDs) - connect to the relay module control pin - marked with IN1/2/3/4..
- Enter wifi SSID and password.
- flash the .ino code to your Node MCU.
- debugging/output stream at BAUD 9600.
- Preferred browser - chrome -
- Download Blynk app on the phone.

11.2 Relay module

A relay module is a circuit that is technically a part of our main circuit but it controls something else. It means that it can control on/off functions with the help of a simple input voltage from our Arduino board.

There are numerous relay modules available in the market. So we have to be aware of what we are doing and what we are buying for.
They are also available in a wide range of quality and use different input and output voltages to perform.

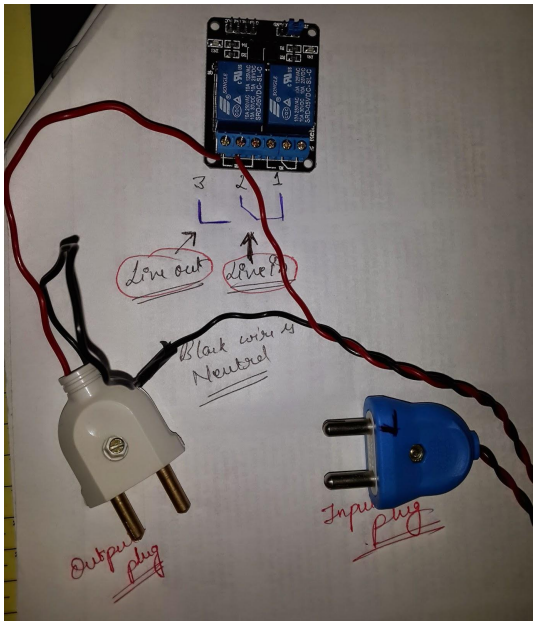
But on a whole relay modules are used to control circuits by breaking and joining the supply line.

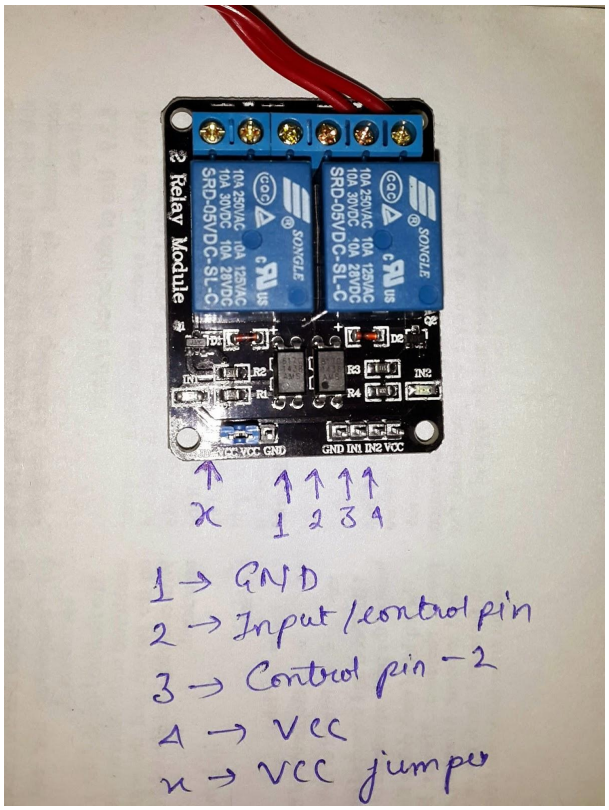
Arduino relays mostly use optocoupler for performing the function.

Relay for Arduino 5V control signal - 10A 220V input (household supply).

Relay provides the first major step towards automating the most fundamental things that we see in our house. You can control almost everything that turns on and off using a relay over Arduino. You can get 1-32 relay modules on a board, which means you can control 1 to up to 32 different things with an Arduino or raspberry pi.

For convenience and simplicity, I have used a 2-channel relay module for turning on and off a 100 W bulb or 18-30W CFL light.


3 2 1
Live out
Live in
Black wire is Neutral
Output plug
Input plug


2 Relay Module
SONGLE
10A 250VAC 10A 30VDC 10A 125VAC 10A 28VDC SRD-05VDC-SL-C
IN1
IN2
R1
R2
R3
R4
VCC GND
GND IN1 IN2 VCC
x 1 2 3 4
1 → GND
2 → Input/control pin
3 → Control pin - 2
4 → Vcc
x → Vcc jumper

11.3 Node MCU

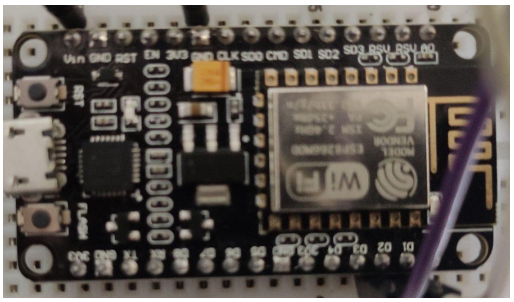
NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e., NodeMCU Development board. Here we are using node MCU . To control our home automation project by using internet.

So things you should keep in mind :-

- Node MCU works on 3.3v. It may get broken if you supply more than 3.3v power .
- **Pin diagram or data sheet :** make sure that you know which pin is Vcc, out and gnd. The pin diagrams varies a lot. It's better to get an Node MCU which have something or label marked on it so that you can search for it's data sheet on the internet.
- If you have used an NodeMCU in wrong way it may overheat and burn. In this case you need to change your NodeMCU because it's not going to work anyway.

Note : A data sheet is a file which contains information about the working your device.
you can search for data sheet using the model number or some marking on your receiver if any.

I am posting an image from the web which helped me in figuring out the pins.
(But always trust the data sheet if there are any conflicts)



- There are 17 GPIO (general-purpose input/output). In node MCU
- **17 GPIO pins** which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically.

CHAPTER 12

12.1 DESIGN CONCEPT

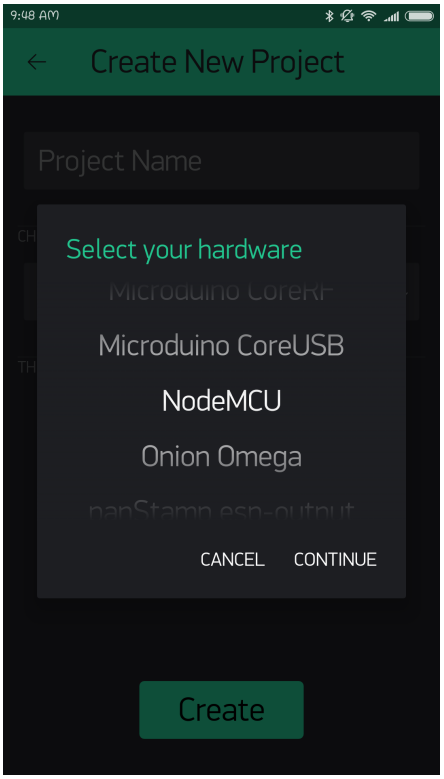
We are going to use Blynk. Blynk was designed **for the Internet of Things**. It can control hardware remotely; it can display sensor data, it can store data, visualize it, and do many other cool things. There are three major components in the platform. Blynk Server - responsible for all the communications between the smartphone and hardware.

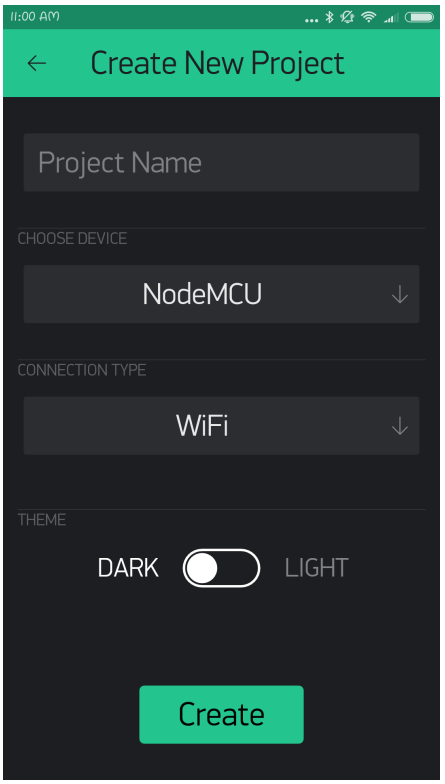
Another thing we are going to use is IFTTT. If This Then That is a service that allows a user to program a response to events in the world. IFTTT has partnerships with service providers that supply event notifications to IFTTT and execute commands that implement the responses. Some event and command interfaces are simply public APIs.

12.1.1 Blynk for NodeMCU –

1. Create a Blynk Project-

Click the "Create New Project" in the app to create a new Blynk app. Give it any name. Blynk works with hundreds of hardware models and connection types. Select the Hardware type. After this, select the connection type. In this project, we have selected WiFi connectivity. The *Auth Token* is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it to yourself.



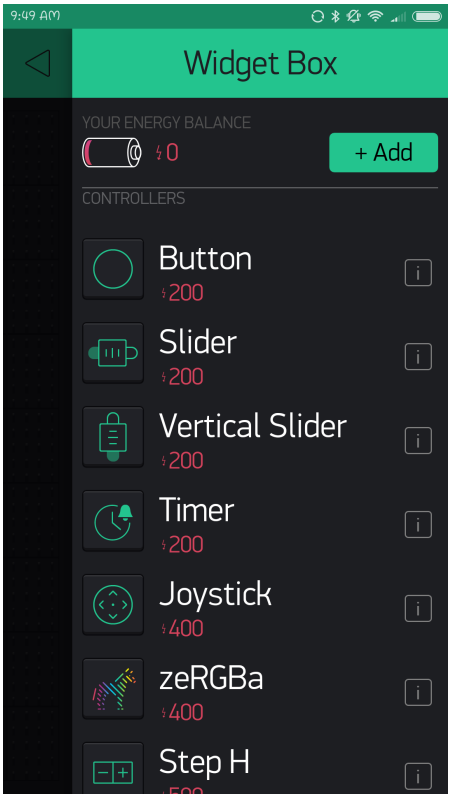


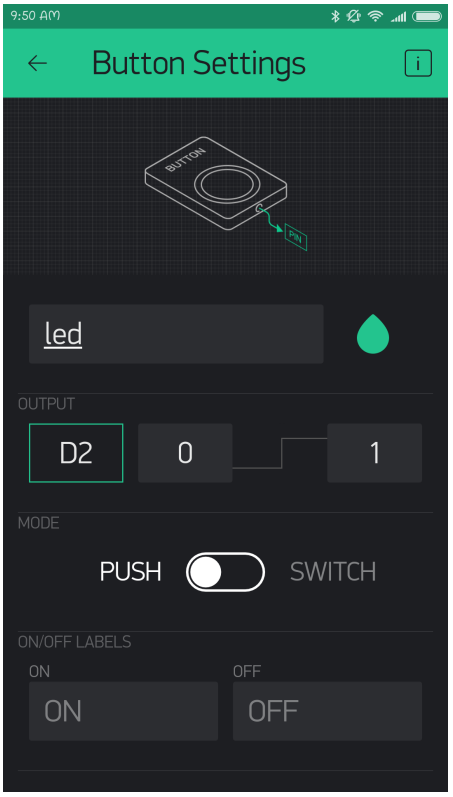
2. Add Widgets To The Project

then you'll be presented with a blank new project. To open the widget box, click in the project window to open. We are selecting a button to control Led connected with NodeMCU.

1. Click on the Button.
2. Give a name to Button say led.
3. Under the OUTPUT tab- Click the pin and select the pin to which led is connected to NodeMCU, here it is digital pin 2, hence select digital and underpin D2. And Click continues.

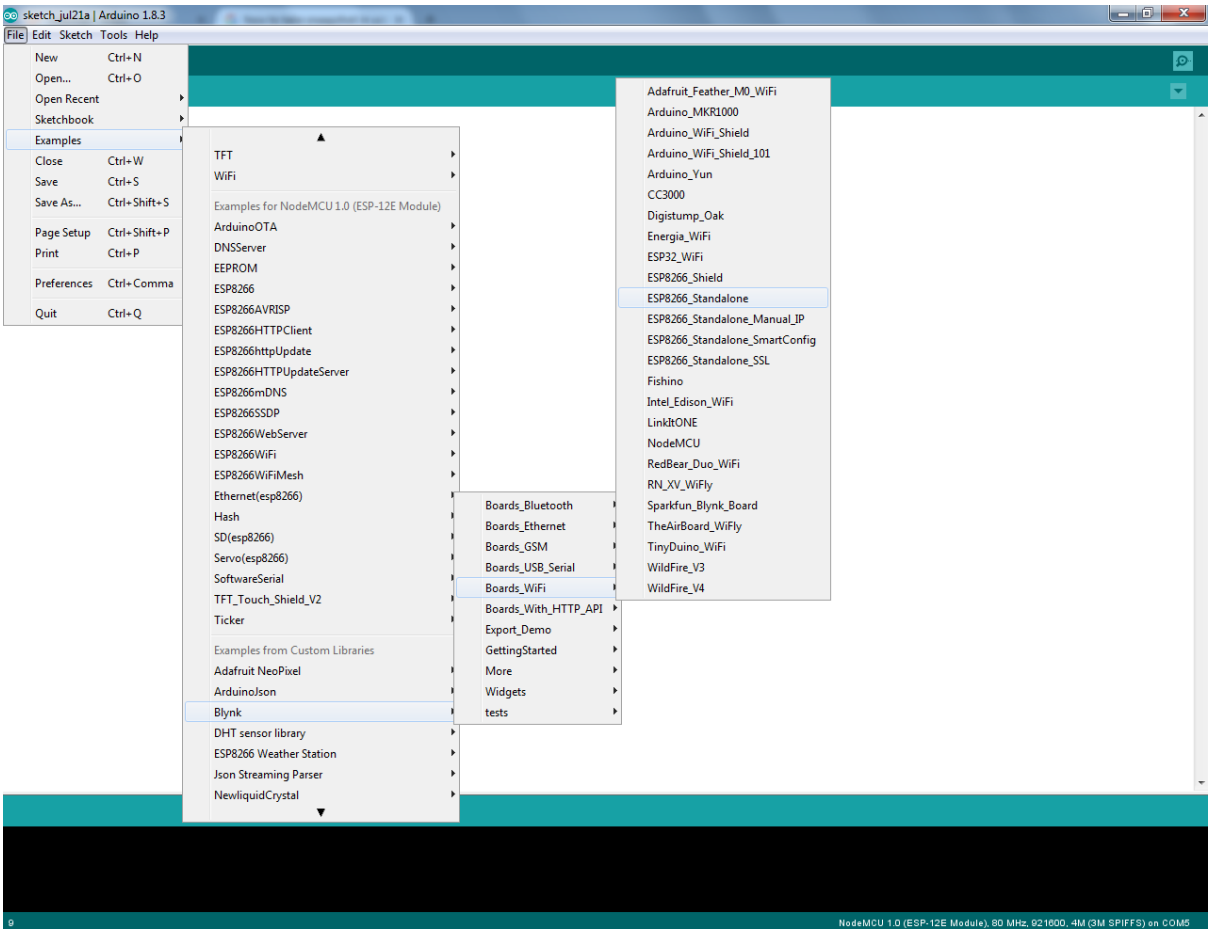
Under the MODE tab- Select whether you want this button as a "push button" or "Switch".





3. Upload The Firmware

Now that your Blynk project is set-up, open Arduino and navigate to the ESP8266_Standalone example in the File > Examples > Blynk > Boards_WiFi> ESP8266_Standalone menu.



4. Stand Alone Programming Code:

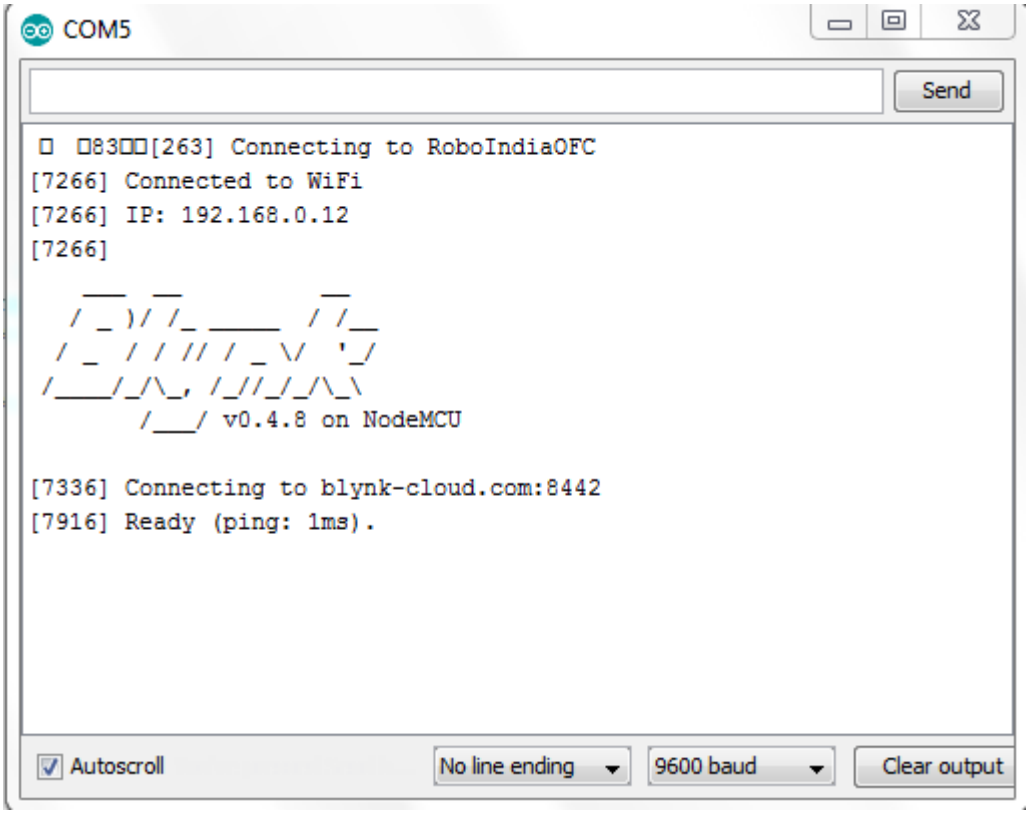
Before uploading, make sure to paste your authorization token into the auth [] variable. Also, make sure to load your Wifi network settings into the Blynk.begin(auth, "ssid", "pass") function.

NodeMCU

```
/*************************************************************
  Download latest Blynk library here:
    https://github.com/blynkkk/blynk-library/releases/latest

  Blynk is a platform with iOS and Android apps to control
  Arduino, Raspberry Pi and the likes over the Internet.
  You can easily build graphic interfaces for all your
  projects by simply dragging and dropping widgets.

    Downloads, docs, tutorials: http://www.blynk.cc
    Sketch generator:           http://examples.blynk.cc
    Blynk community:            http://community.blynk.cc
    Follow us:                  http://www.fb.com/blynkapp
                                http://twitter.com/blynk_app

  Blynk library is licensed under MIT license
  This example code is in public domain.

 *************************************************************
  This example runs directly on NodeMCU.

  Note: This requires ESP8266 support package:
    https://github.com/esp8266/Arduino

  Please be sure to select the right NodeMCU module
  in the Tools -> Board menu!

  For advanced settings please follow ESP examples :
   - ESP8266_Standalone_Manual_IP.ino
   - ESP8266_Standalone_SmartConfig.ino
   - ESP8266_Standalone_SSL.ino

  Change WiFi ssid, pass, and Blynk auth token to run :)
  Feel free to apply it to any other example. It's simple!
 *************************************************************/

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

/* Fill-in your Template ID (only if using Blynk.Cloud) */
//#define BLYNK_TEMPLATE_ID   "YourTemplateID"


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "Lk0HbSr6NEt10c_6HShXgv_ZNhAue2dR";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "asdfg";
char pass[] = "anant512";

void setup()
{
  // Debug console
  Serial.begin(9600);

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

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

5. Execution

After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the “Ready” message. Then click the “Run” button in the top right corner of the Blynk app. Press the button and watch the LEDThen add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware.

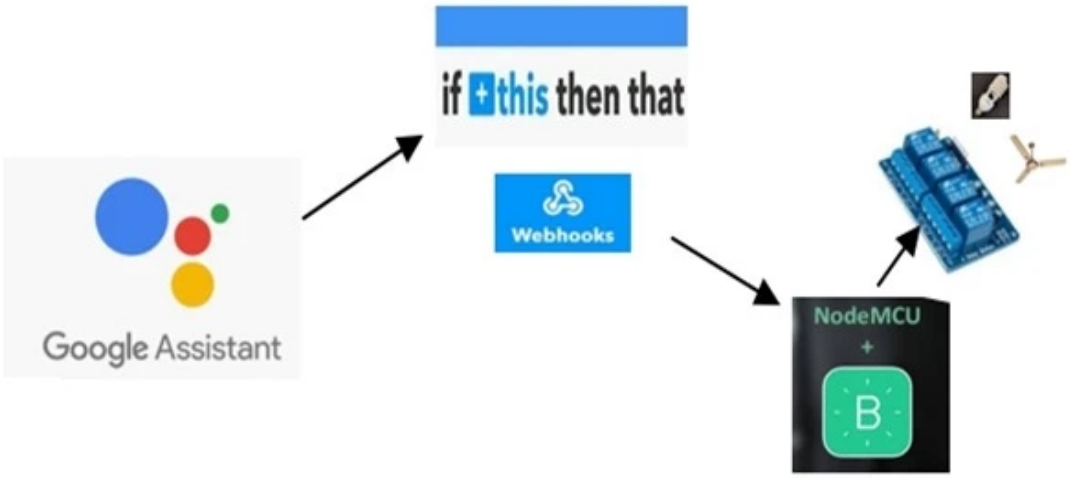


12.1.2 Blynk And IFTTT

Instead of controlling manually we can use voice commands to operate devices.For this, we use the GOOGLE ASSISTANT on mobile phones. Google Assistant cannot directly communicate with BLYNK. For making a chain link between Google assistant and Blynk, we use IFTTT.

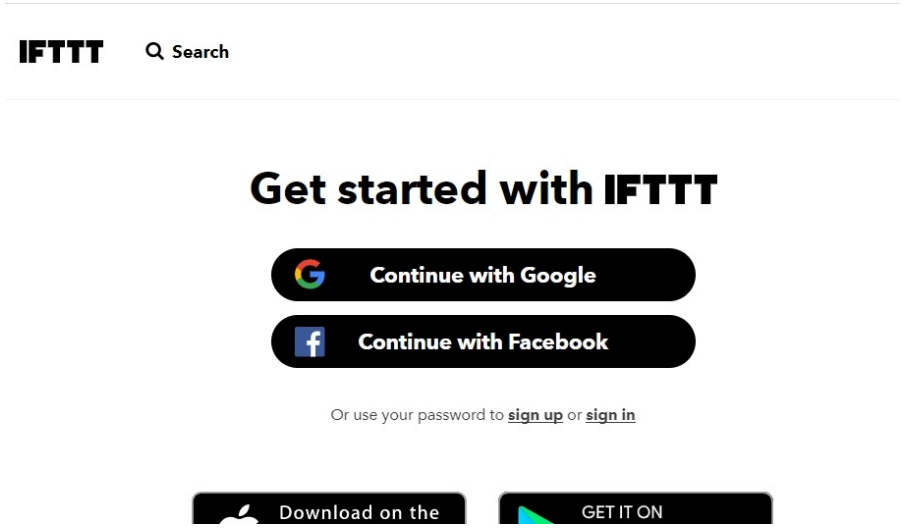
If This Then That, also known as IFTTT, is a free web-based service to create chains of simple conditional statements, called applets. An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, etc.

In this project, we make use of WEBHOOKS on IFTTT to handle a web service, which is BLYNK.



12.1.3 Using IFTTT

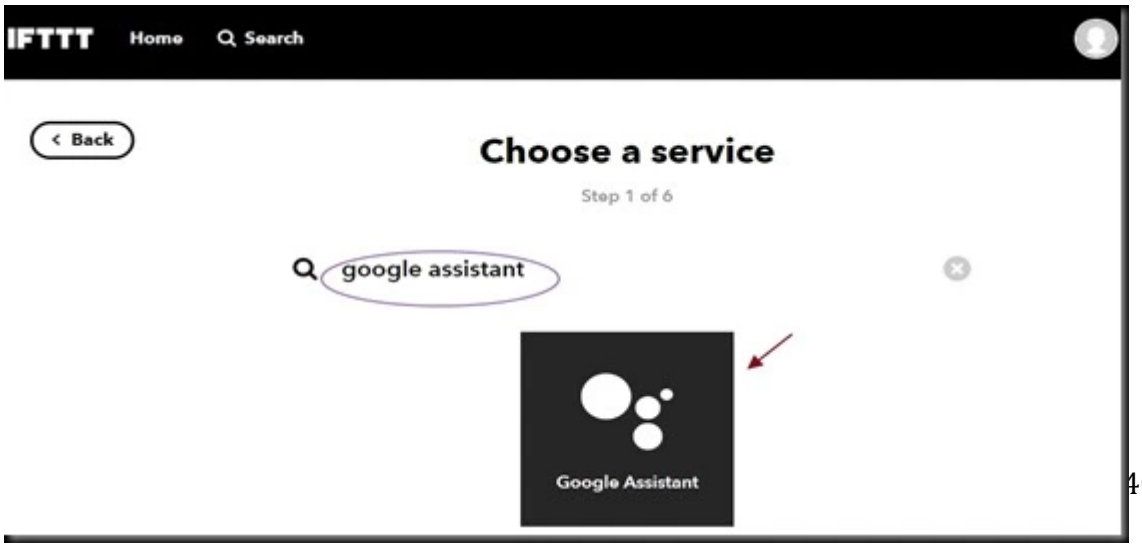
Once the settings on the BLYNK application are completed we can close it. Need not be operated manually. We shall use IFTTT to make a chain between Google assistant and Blynk.



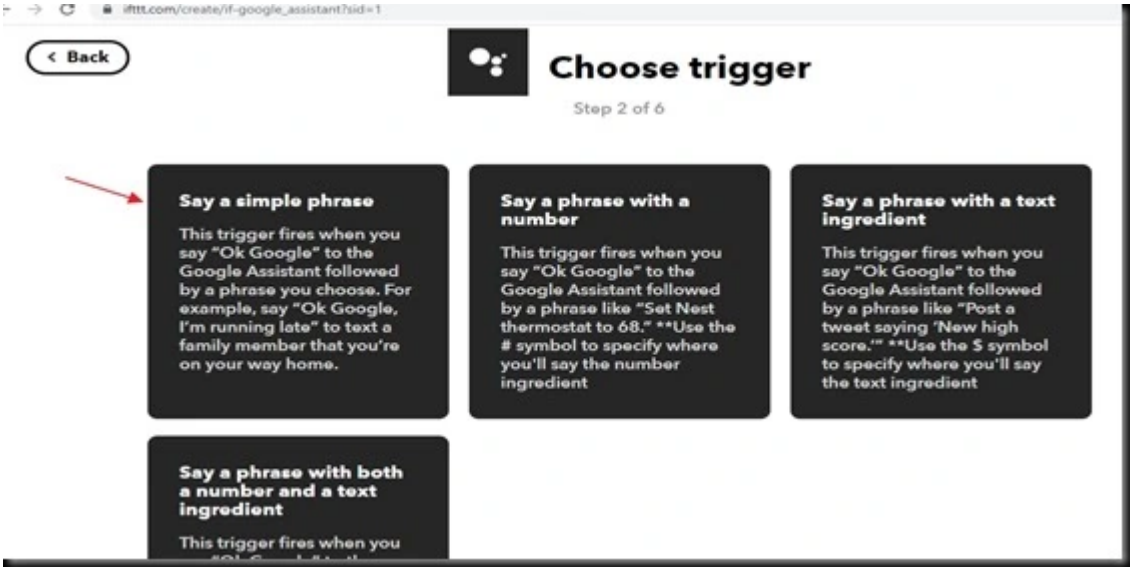
- Click on create

- You see a plus symbol (+) Before this
  Click on that

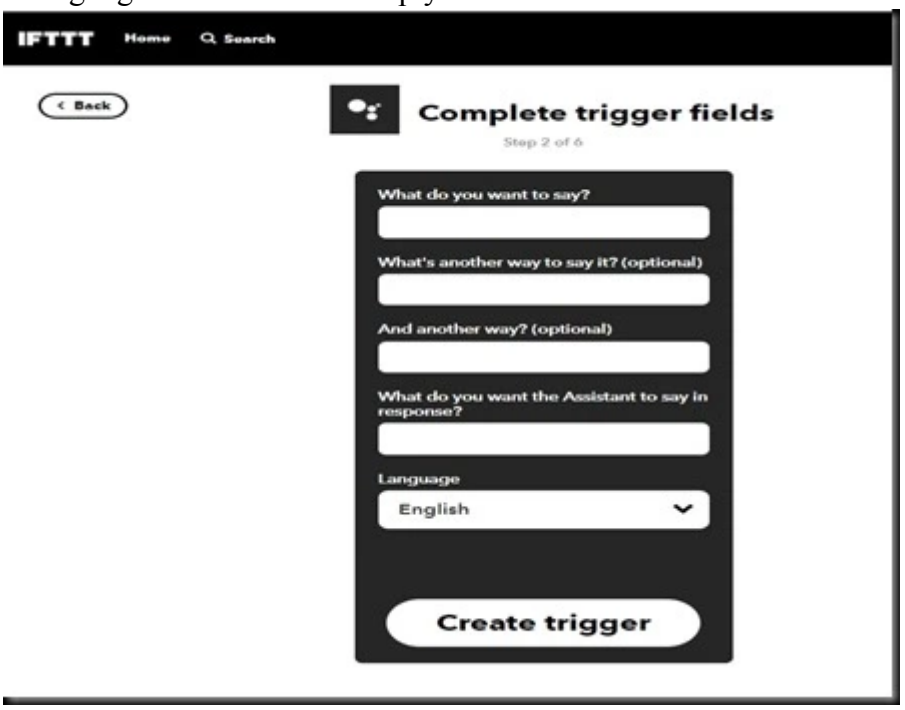
- Choose service Google assistant .



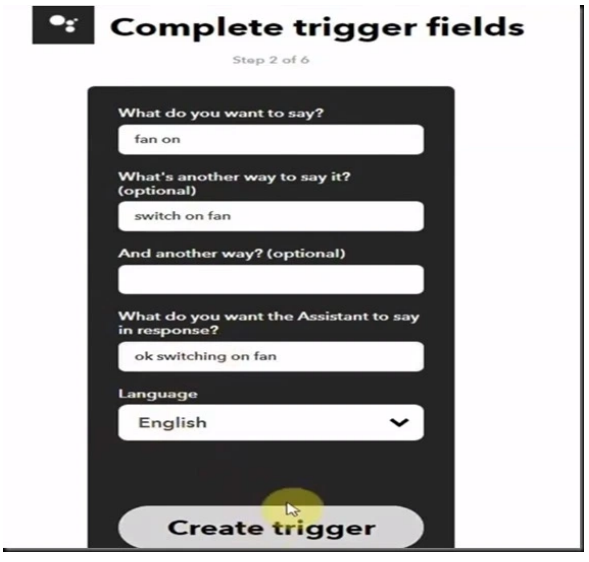
- Choose a method to trigger, say a simple phrase



- Next phase is to complete trigger fields with options like what you want to say& what google assistant should reply



- Complete the Field for fan/bulb / light, whatever you want

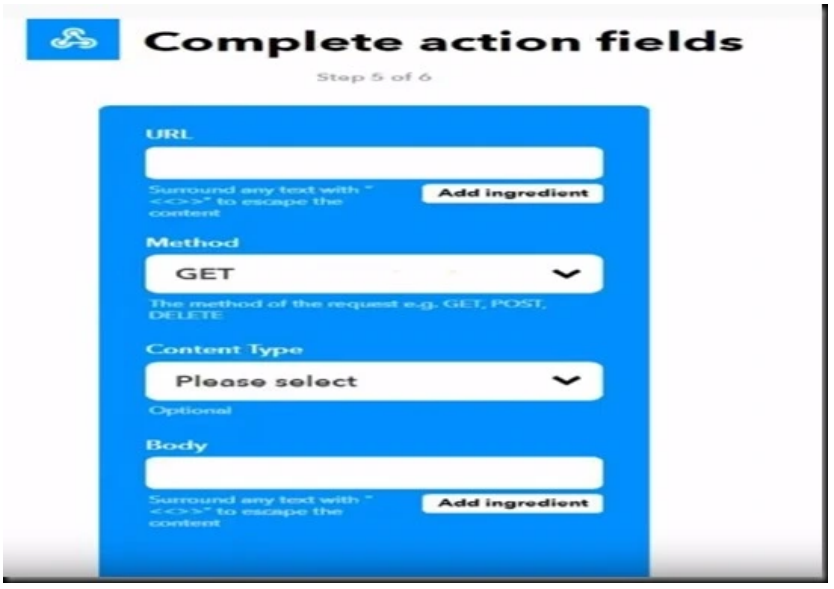


- Now you can see google assistant logo at this position

- Search for webhooks and select it

- Click on “make a web request “

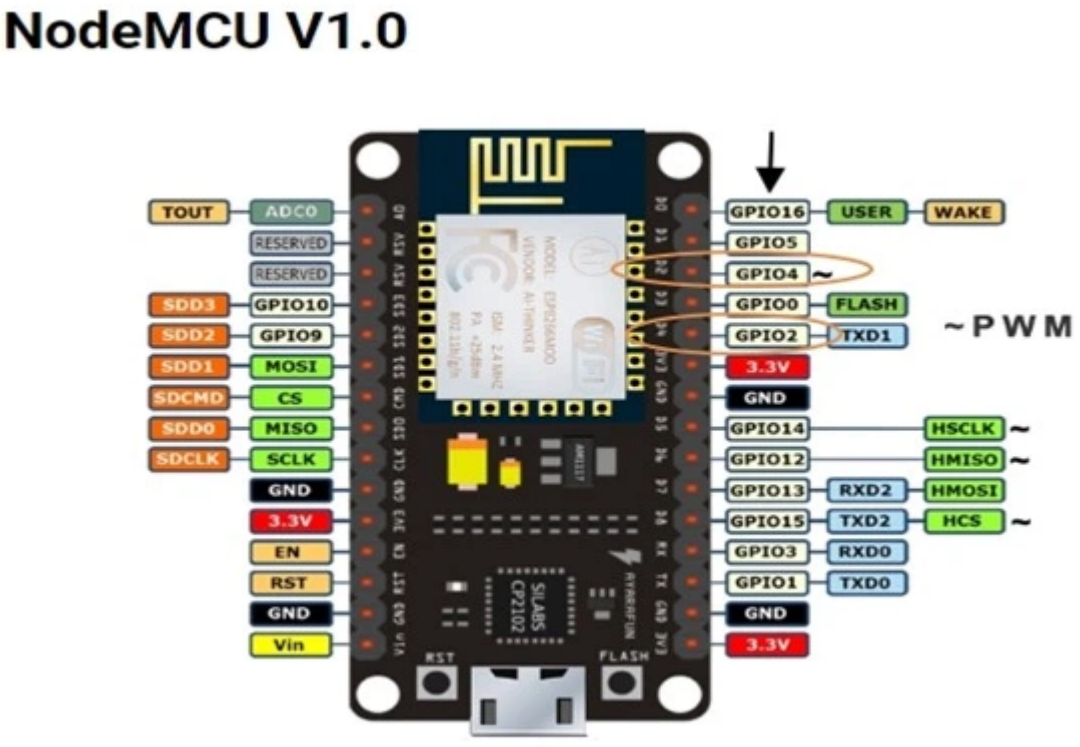
- On next page you have to complete the action field



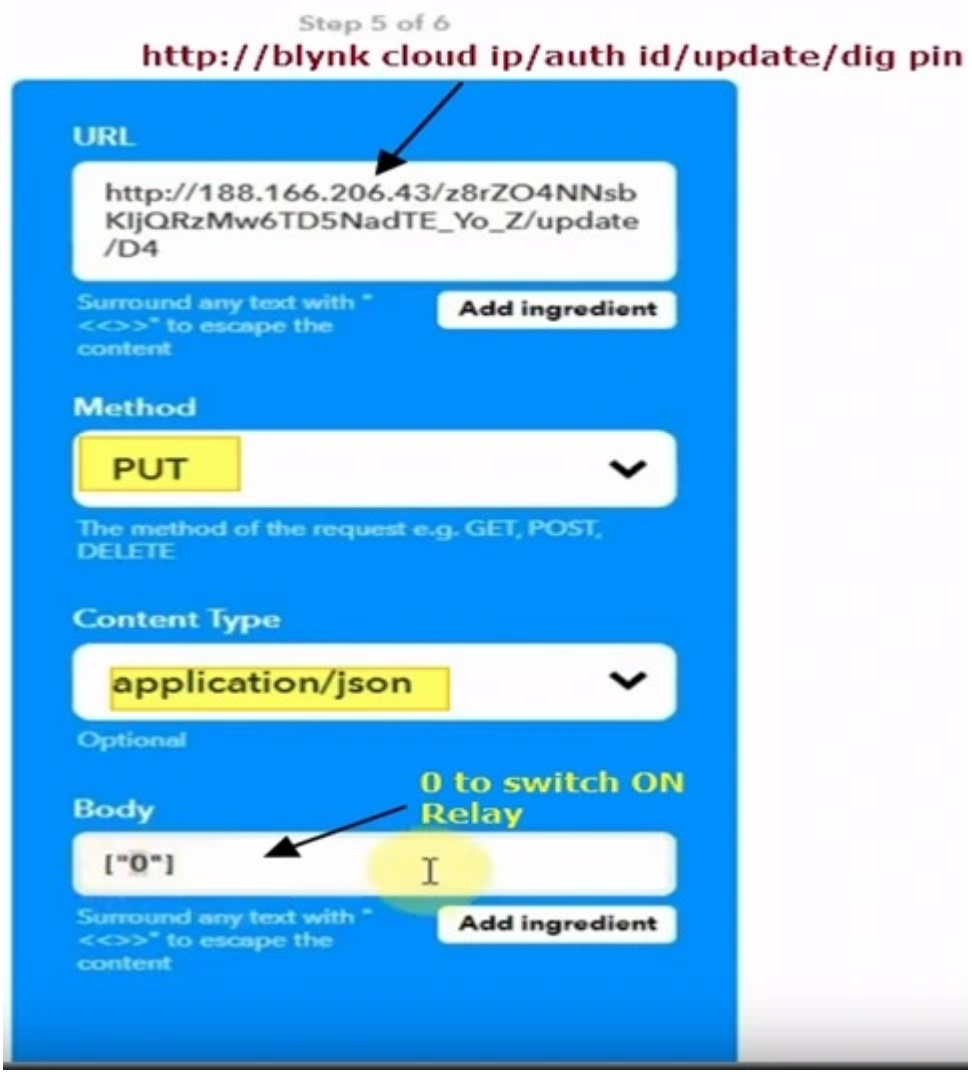
- Under URL you need to provide the IP address of Blynk-cloud.com
  This can be achieved by opening a command prompt & ping blynk-cloud.com

```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\HP>ping blynk-cloud.com

Pinging blynk-cloud.com [188.166.206.43] with 32 bytes of data:
Reply from 188.166.206.43: bytes=32 time=48ms TTL=57
Reply from 188.166.206.43: bytes=32 time=49ms TTL=57
Reply from 188.166.206.43: bytes=32 time=53ms TTL=57
Reply from 188.166.206.43: bytes=32 time=49ms TTL=57

Ping statistics for 188.166.206.43:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 48ms, Maximum = 53ms, Average = 49ms

C:\Users\HP>
```

- URL Format should be like this ***http://188.166.206.43/auth_code/update/digitalpin***
  Authorization ID is the one which we received on Email from Blynk. The digital pin to be controlled is that of Arduino. As we use NODEMCU hardware, check the pinout below to select the correct digital pin. For e,g we used D2 to connect Fan control. In NODEMCU pinout D2 is GPIO4, so you need to input D4.

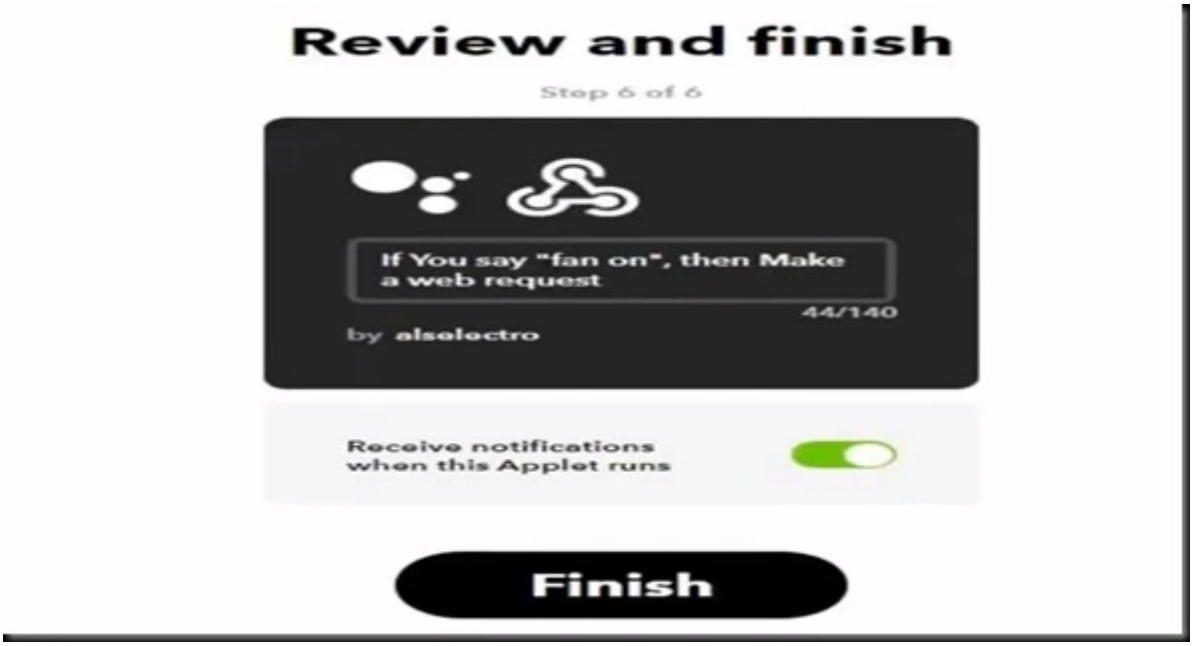


- Method to be selected is PUT.
  The content type is: application.json
  & the body is [“0”]or[“1”]. A 0 within double quotes & square brackets are used to switch on the relay. As the 4-channel relay board, we used is LOW enabled (LOW at input switches ON the relay) we need to feed 0 for ON.

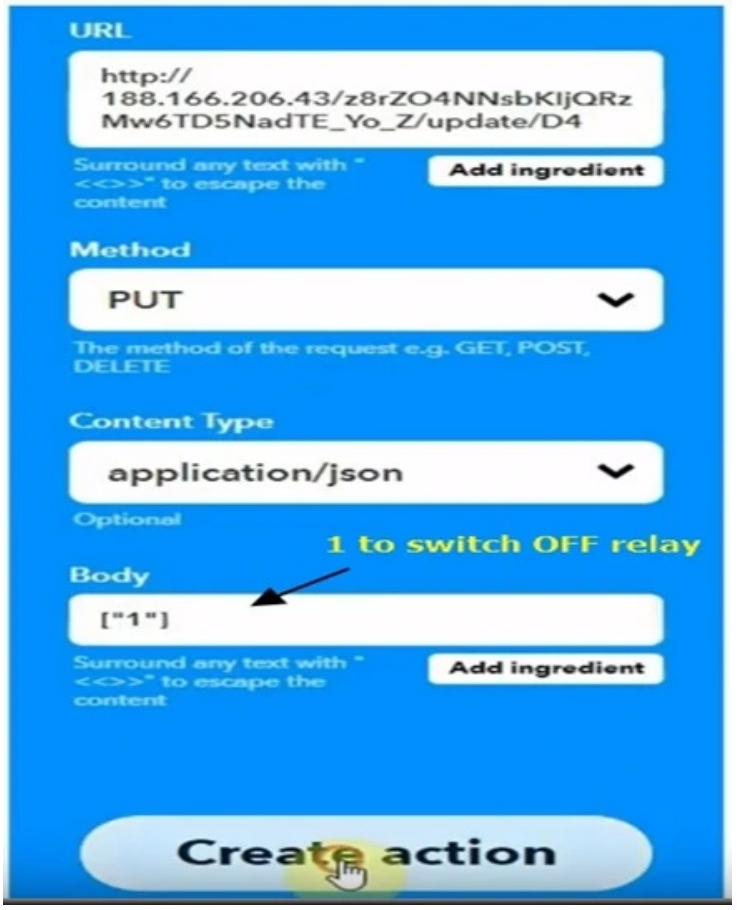


- Now the first applet is created and connected

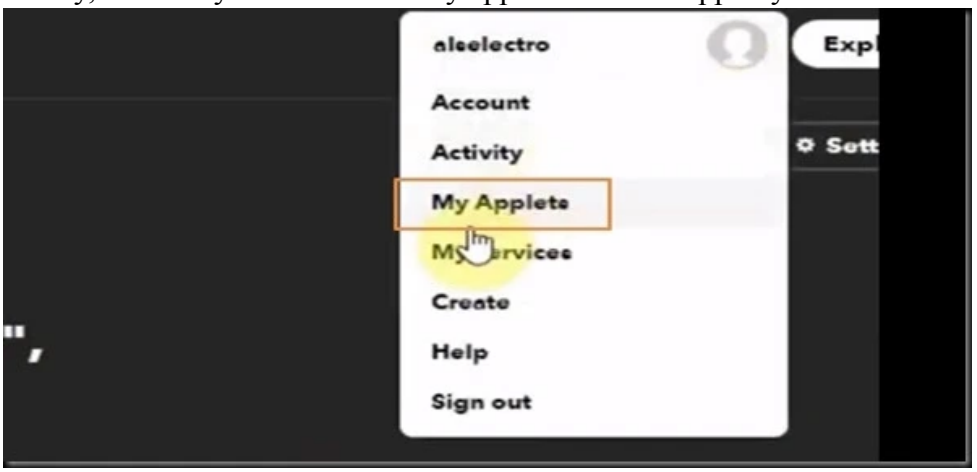
- Click on finish

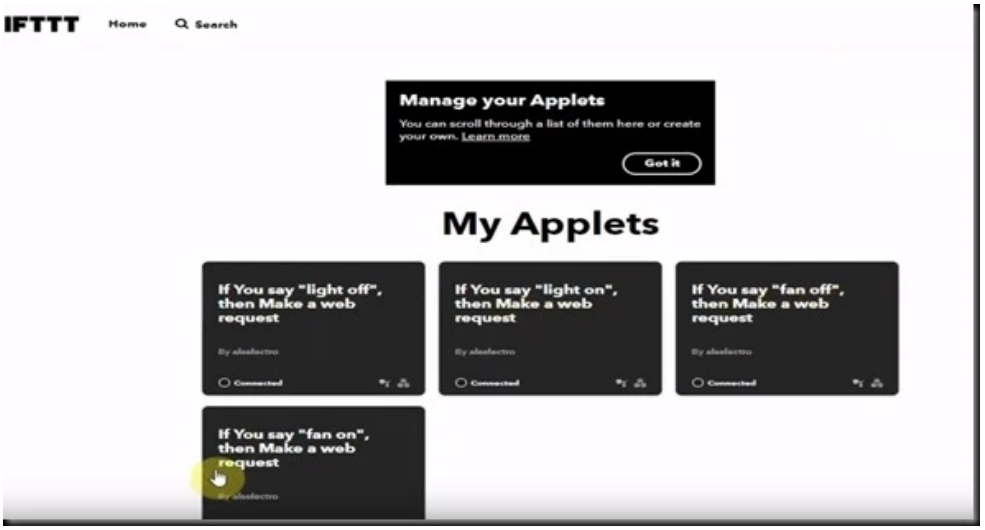


- In the same way we can create many other applets for FAN, LIGHT, and other appliances . Note that in Body you should send a 1 under double quotes & square brackets for switching OFF the relay.

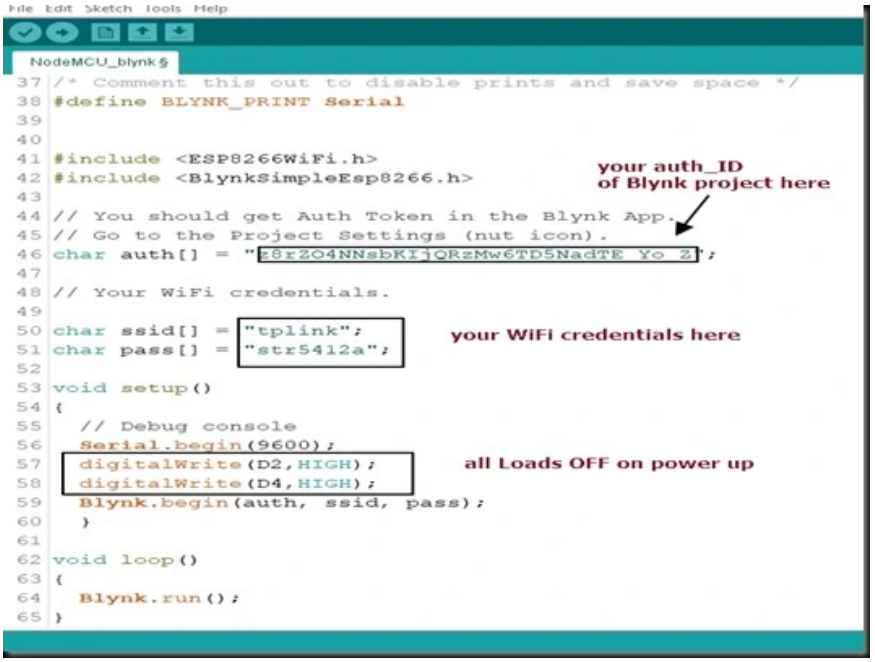


- Finally, click on your account & my applet to see the applet you created.





- Now open your Arduino IDE & under examples –> BLYNK –> BOARDS WIFI –> NODEMCU.Do the following modifications in code & upload to NODEMCU.



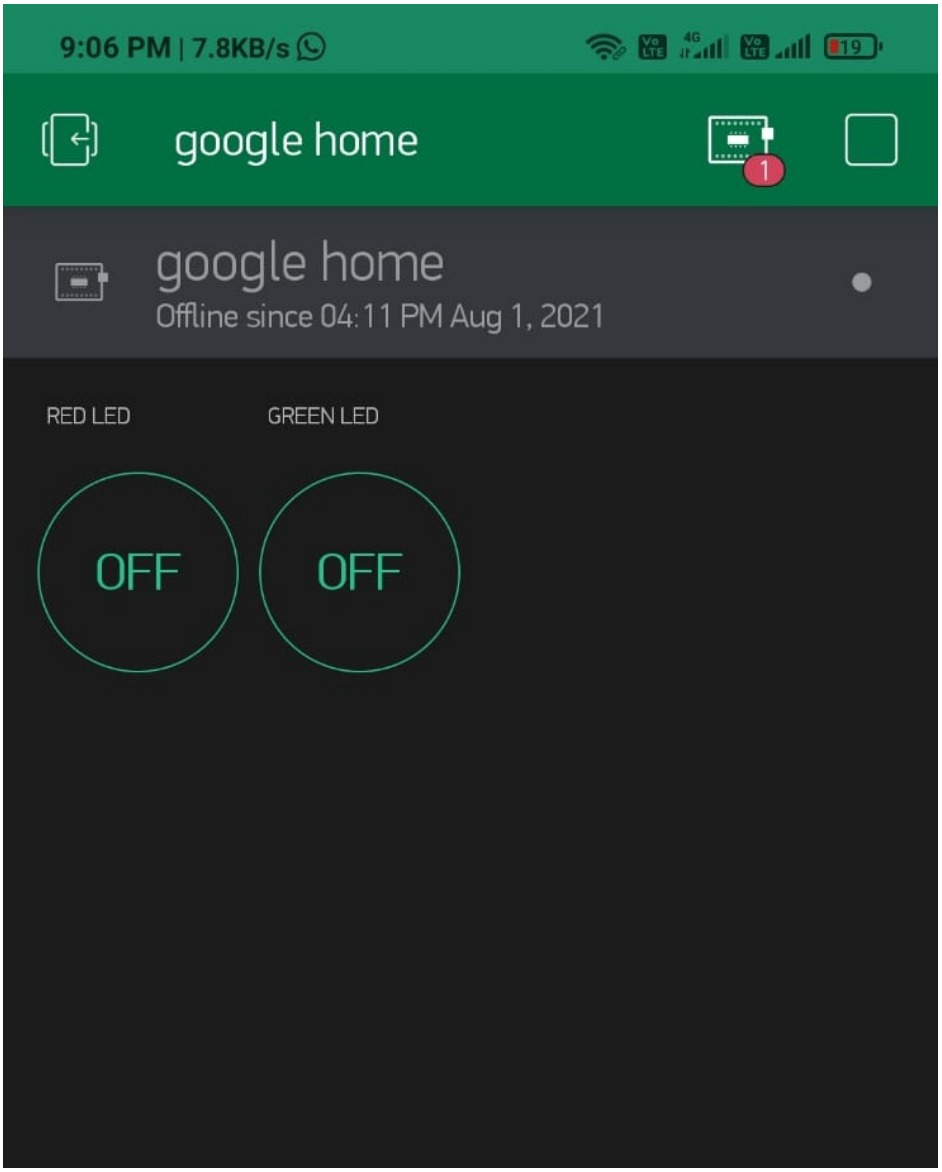
PART III

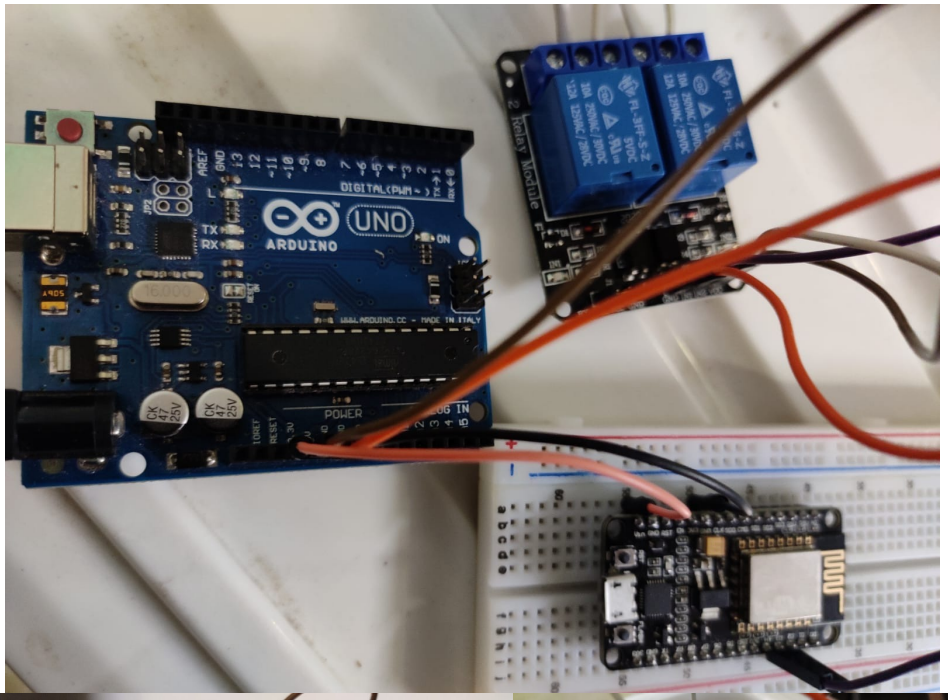
CONCLUSION

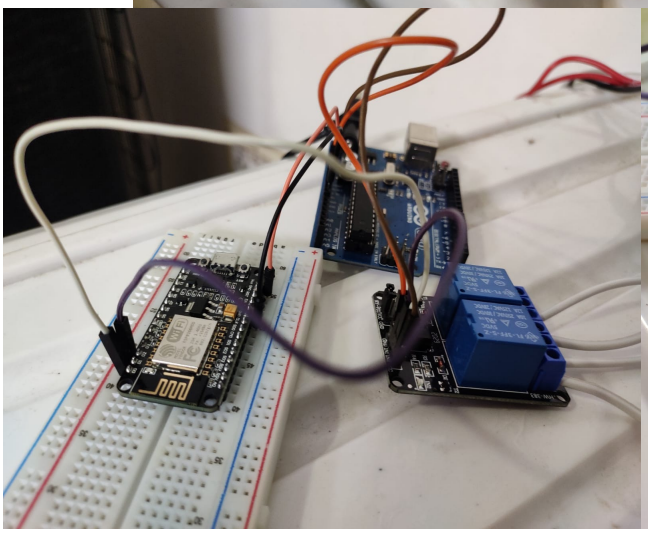
CHAPTER 13

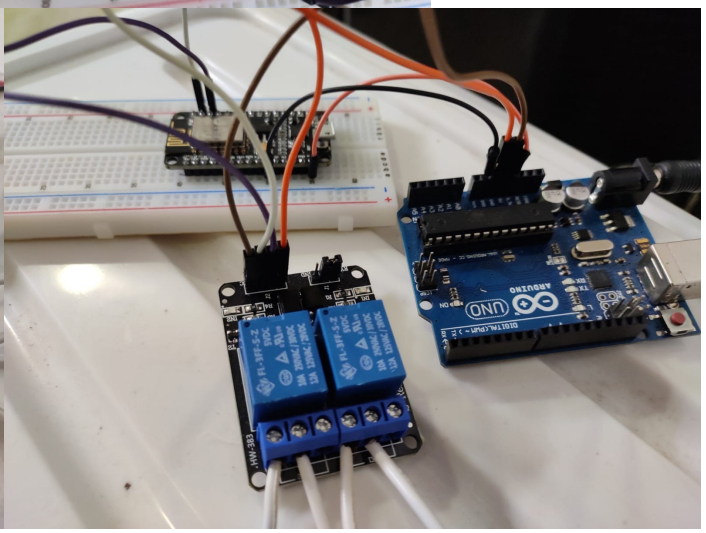
After this, webhook will make a request to the Blynk server to turn on and off the relays which we used. The application screen of the Blynk app will also get updated each time we turn it on and off,

The screen that the Blynk app will show is here:




UNO
ARDUINO
Relay Module





13.1 CONTROLS AVAILABLE

13.1.1 Version 1.1 (Future)

DISTANCE MONITOR:- shows the distance observed by the SR-04 ultrasonic sensor.

LED lights:- control the bulbs and other high voltage devices via relay module via checkboxes and buttons.

13.1.2 Version 2 (future)

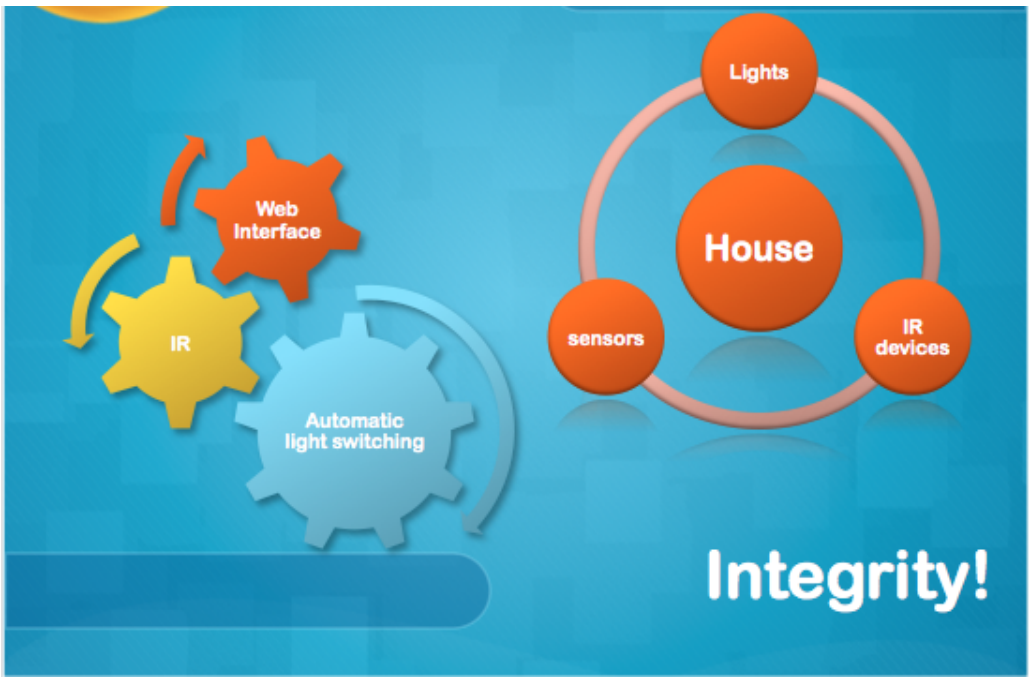
IR received data:- will be used to receive data from IR in future

IR data shooter:- able to copy, paste and store data delivered by IR receiver

IR shooter:- shoot the code via IR led. Uses stored procedures.

13.2 Future scope

To be able to control electronic and electric devices. Receiving and shooting IR signals to appliances such as TV and primarily Air conditioners using ARDUINO MEGA or Bridged ARDUINO UNO.




THANK
YOU!