CHAPTER 1
INTRODUCTION

1.1 Home automation:

Home automation is building automation for a home, called a smart home or smart house. A home automation system will monitor and/or control home attributes such as lighting, climate, entertainment systems, and appliances. It may also include home security such as access control and alarm systems. When connected with the Internet, home devices are an important constituent of the Internet of Things ("IoT").

A home automation system typically connects controlled devices to a central smart home hub (sometimes called a "gateway"). The user interface for control of the system uses either wall-mounted terminals, tablet or desktop computers, a mobile phone application, or a Web interface that may also be accessible off-site through the Internet.

While there are many competing vendors, there are increasing efforts towards open source systems. However, there are issues with the current state of home automation including a lack of standardized security measures and deprecation of older devices without backwards compatibility.

Home automation has high potential for sharing data between family members or trusted individuals for personal security and could lead to energy saving measures with a positive environmental impact in the future.

The home automation market was worth US$5.77 billion in 2013, predicted to reach a market value of US$12.81 billion by 2020.

In other words, “Home automation” refers to the automatic and electronic control of household features, activity, and appliances. In simple terms, it means you can easily control the utilities and features of your home via the Internet to make life more convenient and secure, and even spend less on household bills.

1.1(a) How does home automation work?

Home automation is a network of hardware, communication, and electronic interfaces that work to integrate everyday devices with one another via the Internet. Each device has sensors and is connected through WiFi, so you can manage them from your smartphone or tablet whether you’re at home, or miles away. This allows you to turn on the lights, lock the front door, or even turn down the heat, no matter where you are.

Far from the idea of having a robot at home to do things, the purpose of home automation is that the functions of all electrical and electronic equipment in the house are controlled and automated locally or remotely through a central integrated system.

Luz e Som, a pioneer in Home Automation area in Portugal, installs this type solutions since around 1998, having had a very strong participation in the first Smart Home in Portugal, at the Concreta exhibition in Exponor.

Currently, with many hundreds of solutions already implemented, from simple home automation systems to sophisticated solutions in terms of the best things in the world especially from the United States, pioneer country in these solutions.

The fact that we operate for about 20 years in the professional market of audio, video and system integration allowed us to acquire a "know-how" that was easily transported to the home of home automation solutions. In addition, permanent contact with international markets and fairs with partners working in the sector, allowed Luz e Som to deploy firsthand in Portugal many innovative solutions of the world's leading brands of the sectors in which they operate.

1.1(b) What are the benefits of home automation?

The purpose of a home automation system is to streamline how your home functions. Consider some of these benefits:

- *Remote access:* Control your home from mobile devices, including your laptop, tablet, or smartphone.
- *Comfort:* Use home automation to make your home a more comfortable, liveable space. Pre-program your thermostat with your preferred settings so that your home is always at a comfortable temperature, set up smart speakers to play music when you get home from work, or adjust your lights to soften or brighten based on the time of day.
- *Convenience:* Program devices to turn on automatically at certain times, or access their settings remotely from anywhere with an Internet connection. When you don't have to remember to lock the door behind you or switch off the lights, you can turn your attention to more important things.
- *Increased safety:* Smart fire detectors, carbon monoxide monitors, pressure sensors, and other home automation security features can help protect your home from disaster.
- *Energy efficiency*: Home automation allows you to be more mindful of your power usage. For example, you can save on energy bills by reducing the length of time that lights stay on, or by lowering temperatures when you leave a room.

1.1(c)What do we do in this project?

In this project we will control an LED which is connected to ESP8266 with a Telegram bot. As we know about Telegram, it is a messenger app similar to Whatsapp. This enables users to control their ESP8266 via simply typing and sending commands in Telegram by creating a bot. By adding some relays or TRIAC you can make this a home automation project. We control the home automation by using that Telegram bot.

<u>1.2 Requirements:</u>

1.2(a) Hardware:

- ESP8266 NodeMCU
- Lamp
- Jumper wires
- Breadboard
- Relay module
- Battery / Power Supply

1.2(b) Software:

- Arduino IDE (v1.6 or above)
-
- Telegram Bot

We will see more details of each components in the next chapter.

CHAPTER 2

2.1 IoT:

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A thing in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as a pill to something as big as an aeroplane, into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us more smarter and more responsive, merging the digital and physical universes.

Over the past few years, IoT has become one of the most important technologies of the 21st century. Now that we can connect everyday objects—kitchen appliances, cars, thermostats, baby monitors—to the internet via embedded devices, seamless communication is possible between people, processes, and things.

By means of low-cost computing, the cloud, big data, analytics, and mobile technologies, physical things can share and collect data with minimal human intervention. In this hyperconnected world, digital systems can record, monitor, and adjust each interaction between connected things. The physical world meets the digital world—and they cooperate

2.1(a) How does IoT work?

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.

2.1(b)Advantages of IoT:

- Cost reduction
- Efficiency and productivity
- Business opportunities
- Customer experience
- Mobility and agility

2.2 Arduino IDE:

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the Arduino language, inspired by the Processing language and used with a modified version of the Processing IDE. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool developed in Go.

2.3 Node MCU ESP8266:

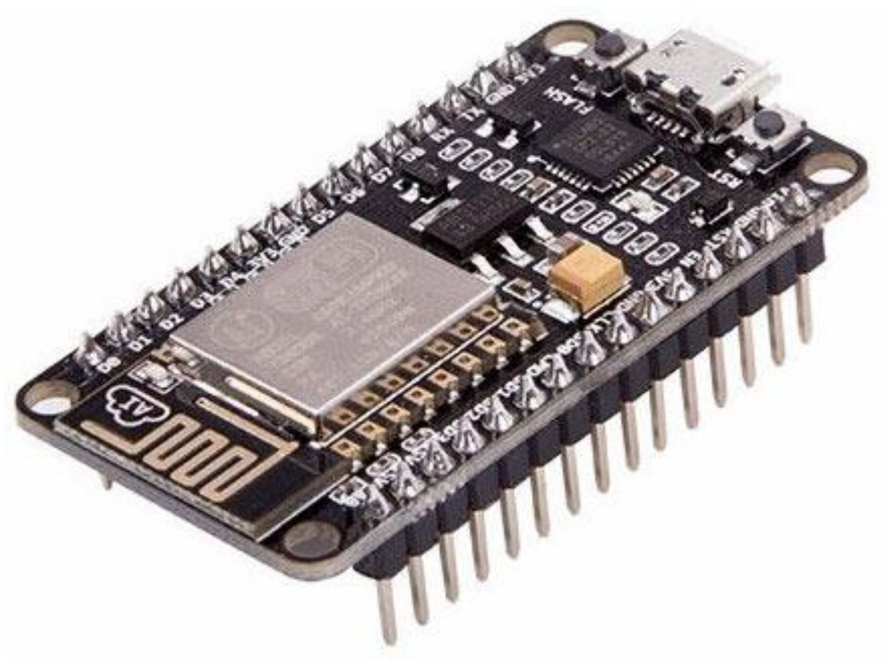
NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).[8] The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.[citation needed]

Both the firmware and prototyping board designs are open source.

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 and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a 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.



We use ESP-12 version in this project.

2.4 Relay Module:

The relay is the device that open or closes the contacts to cause the operation of the other electric control. It detects the undesirable condition with an assigned area and gives the commands to the circuit breaker to disconnect the affected area through ON or OFF.

Every electromechanical relay consists of

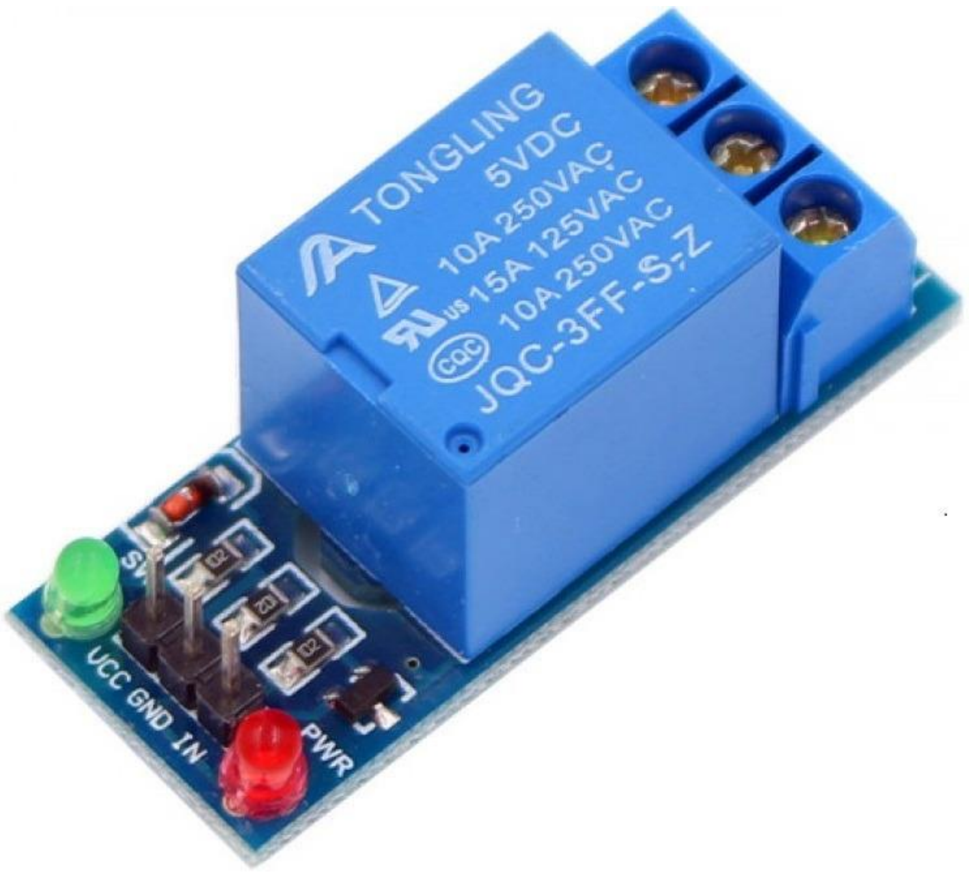
1. Electromagnet

3. Mechanically movable contact

3. Switching points and

4. Spring.

One Channel Relay Module:

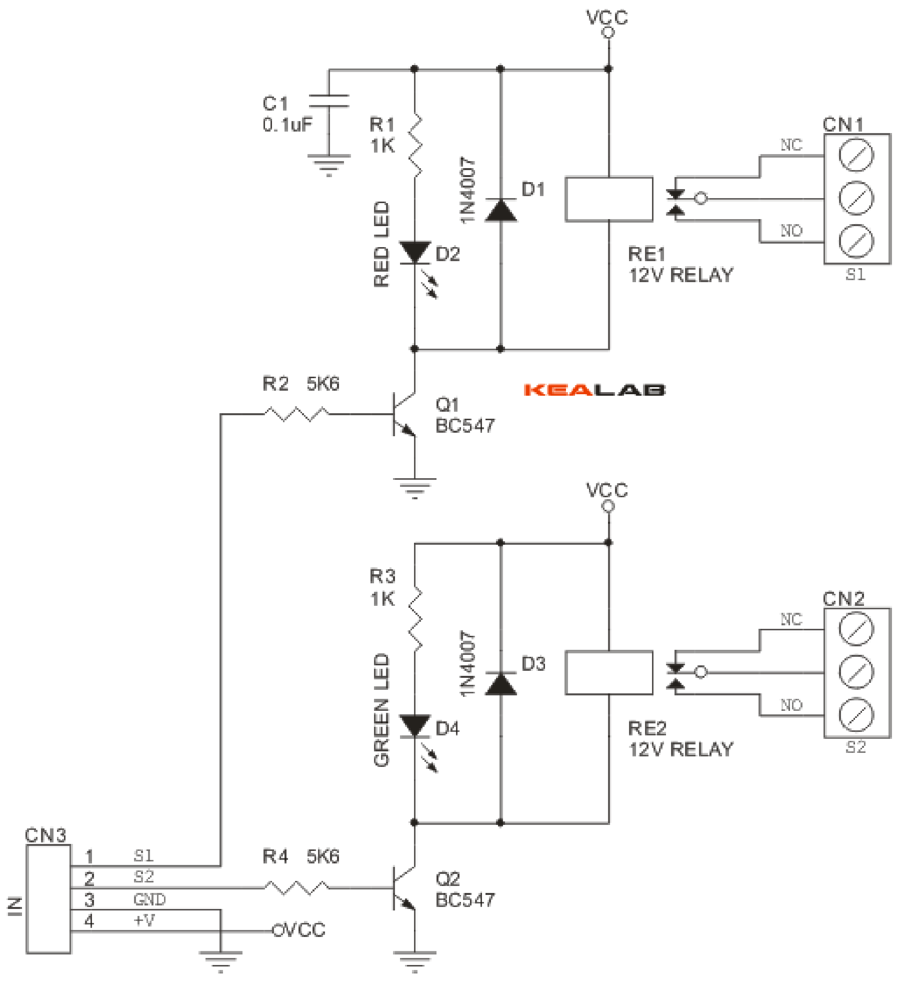


COM: common pin

NO: Normally open – there is no contact between the common pin and the normally open pin. So, when you trigger the relay, it connects to the COM pin and power is provided to the load.

NC: Normally closed – there is contact between the common pin and the normally closed pin. There is always connection between the COM and NC pins, even when the relay is turned off. When you trigger the relay, the circuit is opened and there is no supply provided to the load.

Relay Module 2-Channel Circuit Diagram:



.

2.4(a) APPLICATIONS OF RELAY:

A. They can be used for both ac and dc systems for protection of ac and dc equipment's

B. Electromagnetic relays operating speeds which has the ability to operate in milliseconds are also can be possible

C. They have the properties such as simple, robust, compact and most reliable

D. These relays are almost instantaneous. Though instantaneous the operating time of the relay varies with the current. With extra arrangements like dashpot, copper rings.

E. Electromagnetic relays have fast operation and fast reset.

2.4(b) DISADVANTAGES:

a. High burden level instrument transformers are required (CTs and PTs of high burden is required for operating the electromagnetic relays compared to static relays)

b. The directional feature is absent in electromagnetic relays

c. Requires periodic maintenance and testing unlike static relays

d. Relay operation can be affected due to ageing of the components and dust, pollution resulting in spurious trips

e. Operation speed for an electromagnetic relays is limited by the mechanical inertia of the component.

CHAPTER 3

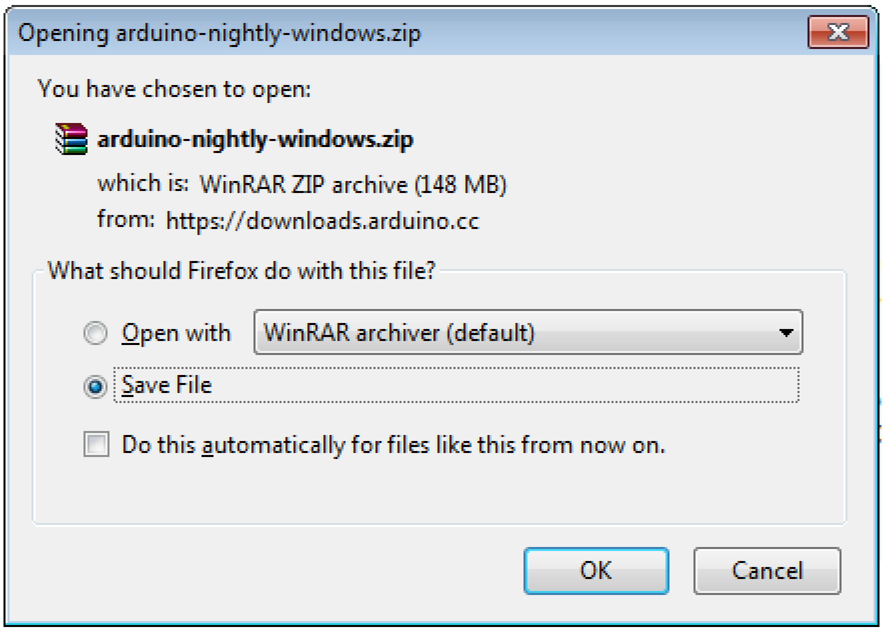
3.1 Installing the libraries in Arduino IDE:

Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)

from: https://downloads.arduino.cc

What should Firefox do with this file?

Open with [WinRAR archiver (default)]

[✓ Save File]

Do this automatically for files like this from now on.

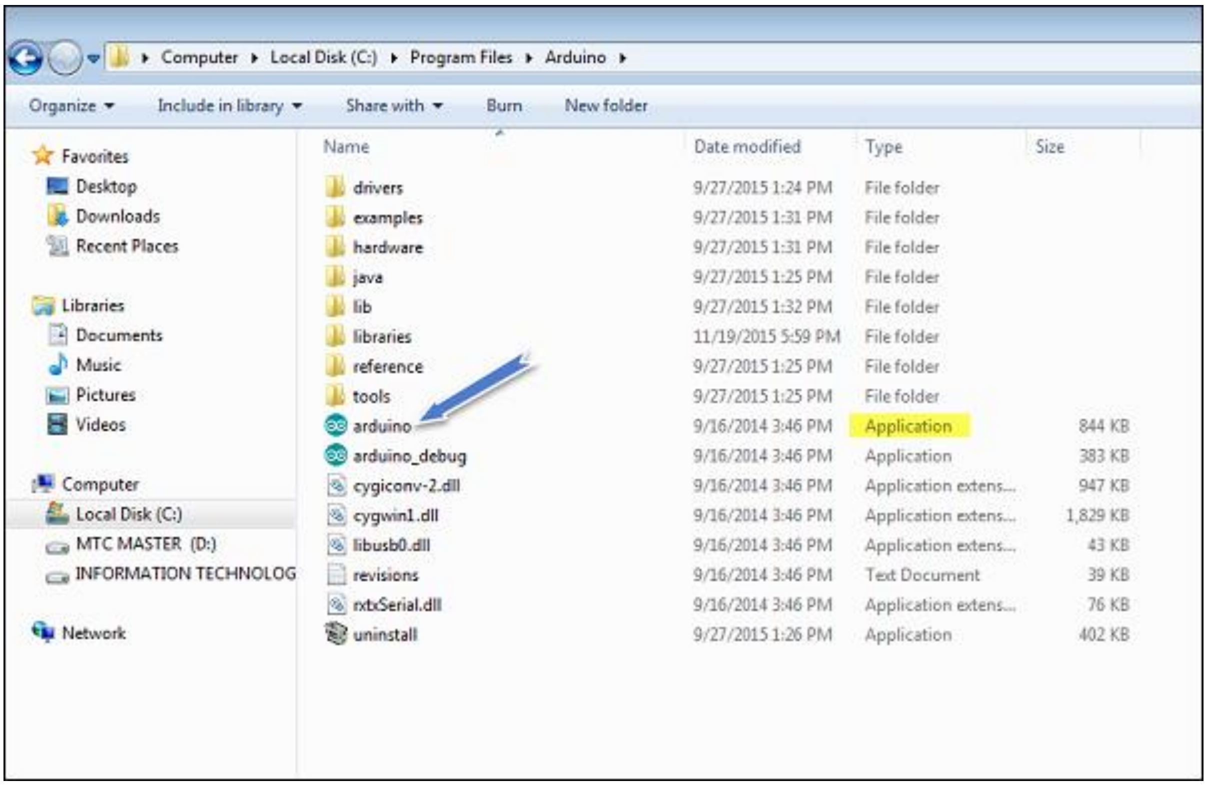
Step 3 − Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



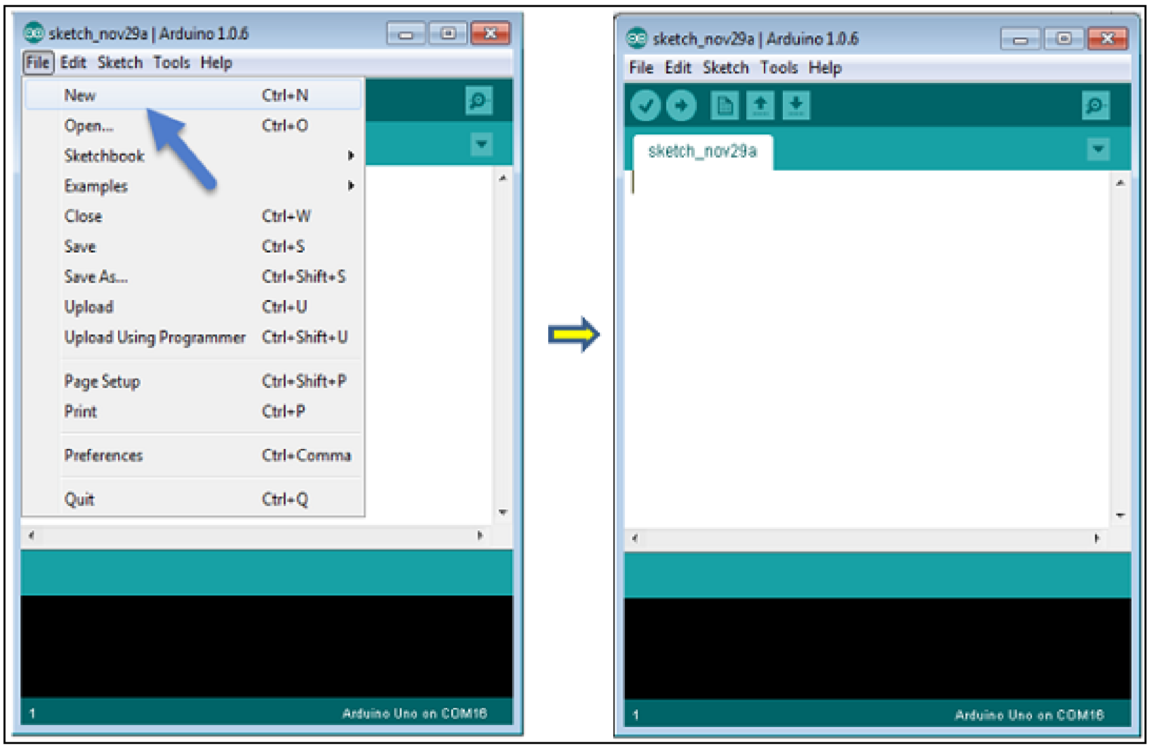
Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

Step 5 − Open your first project.

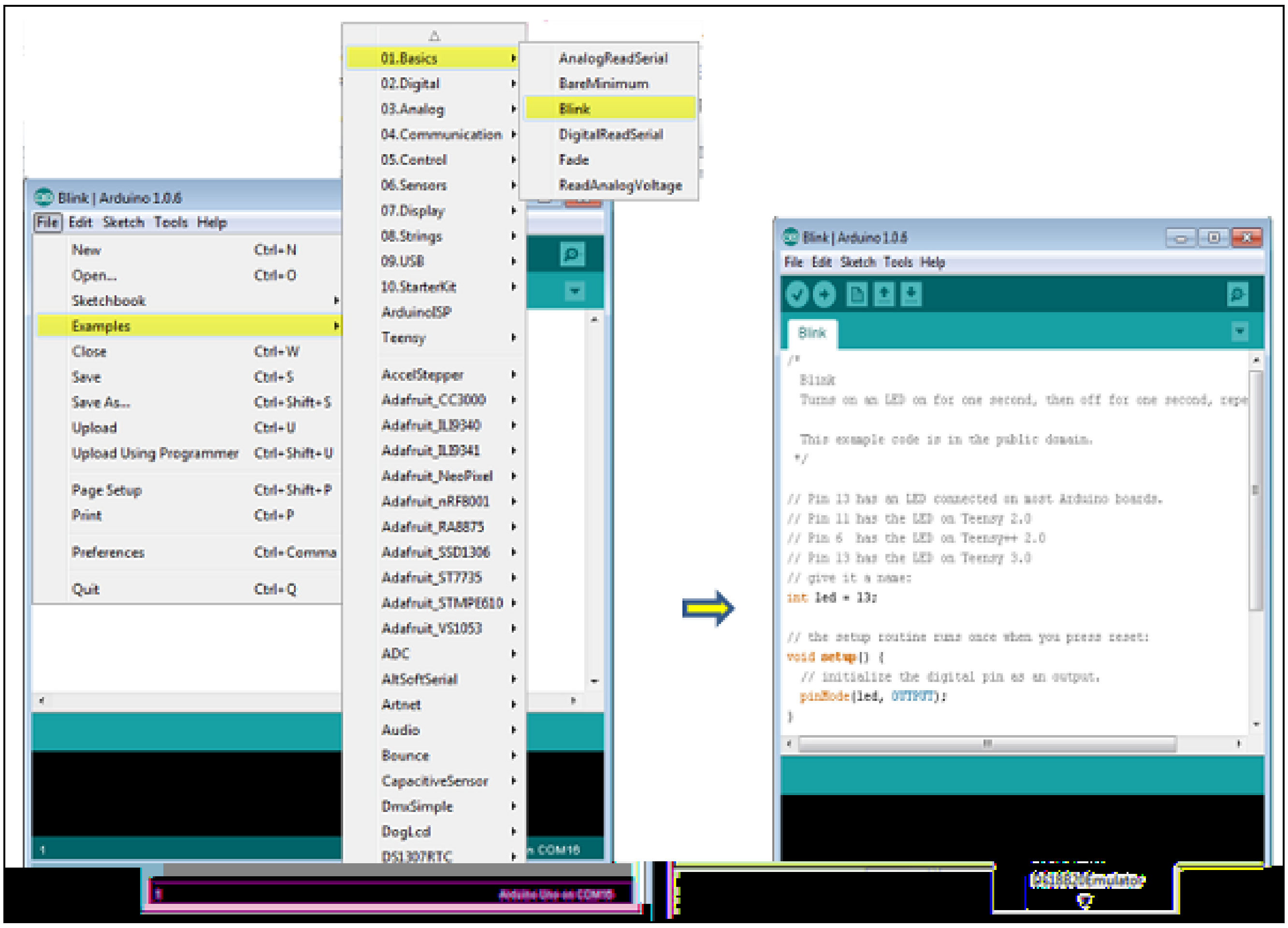
Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.



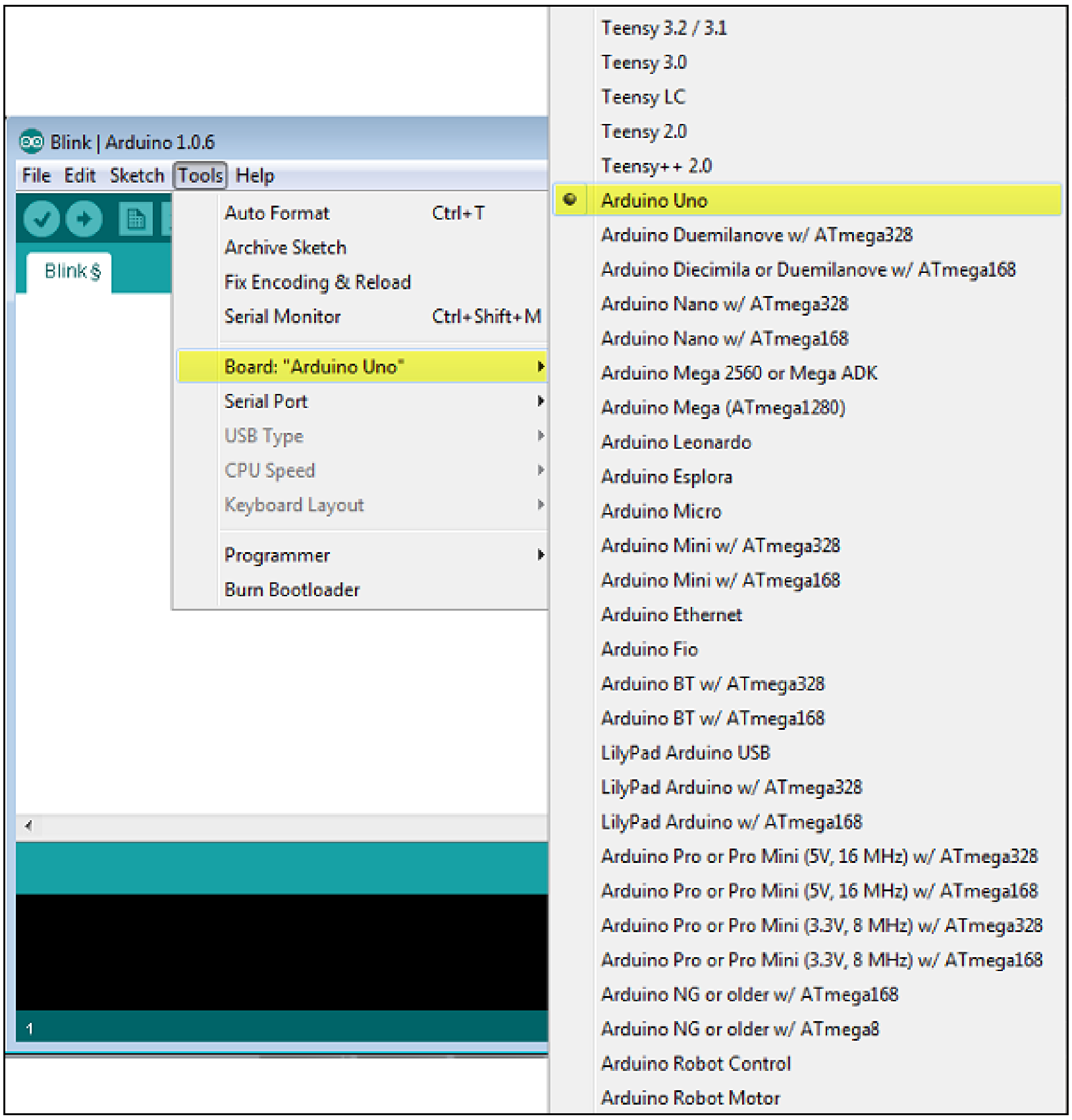
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

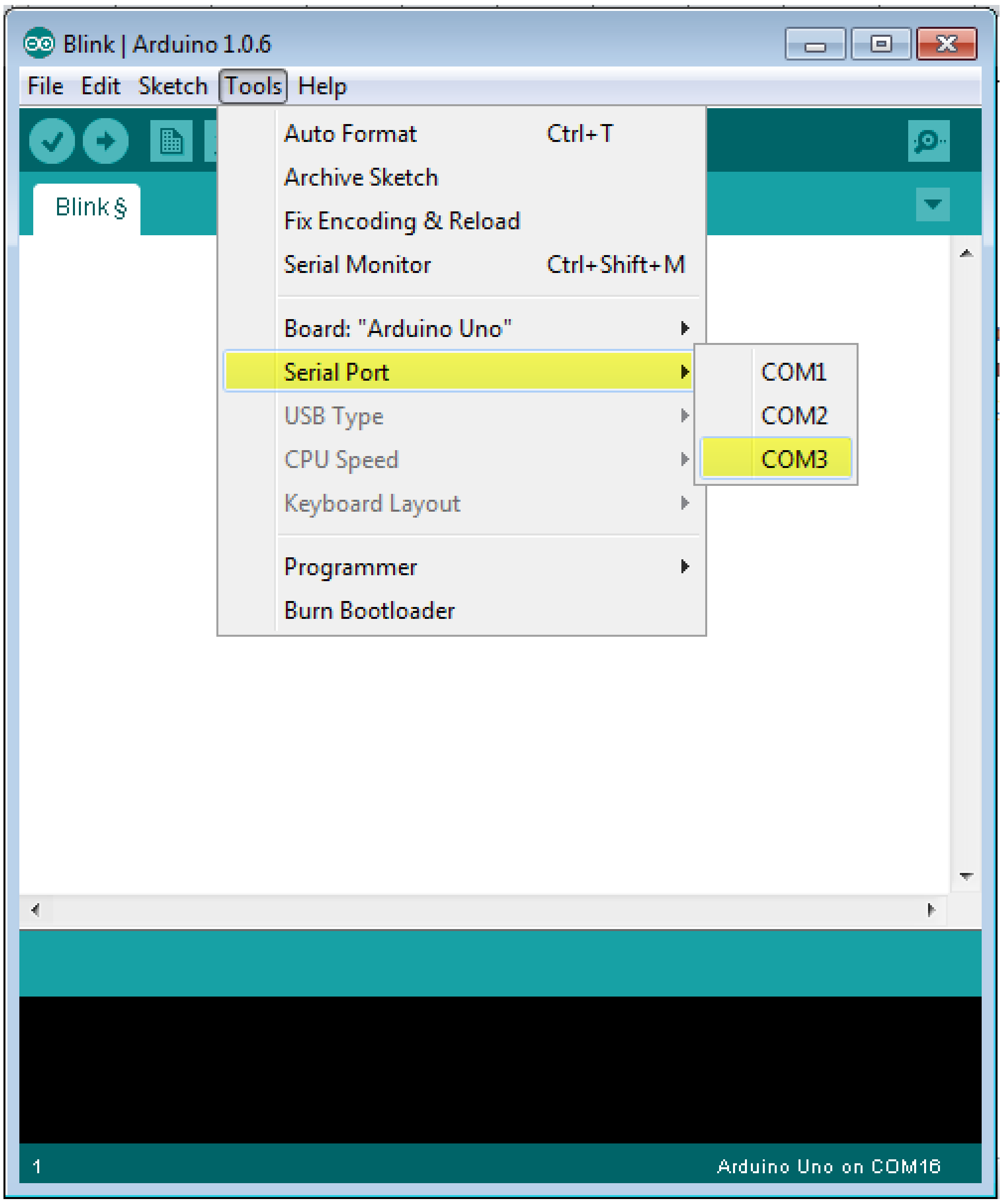
Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
● Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

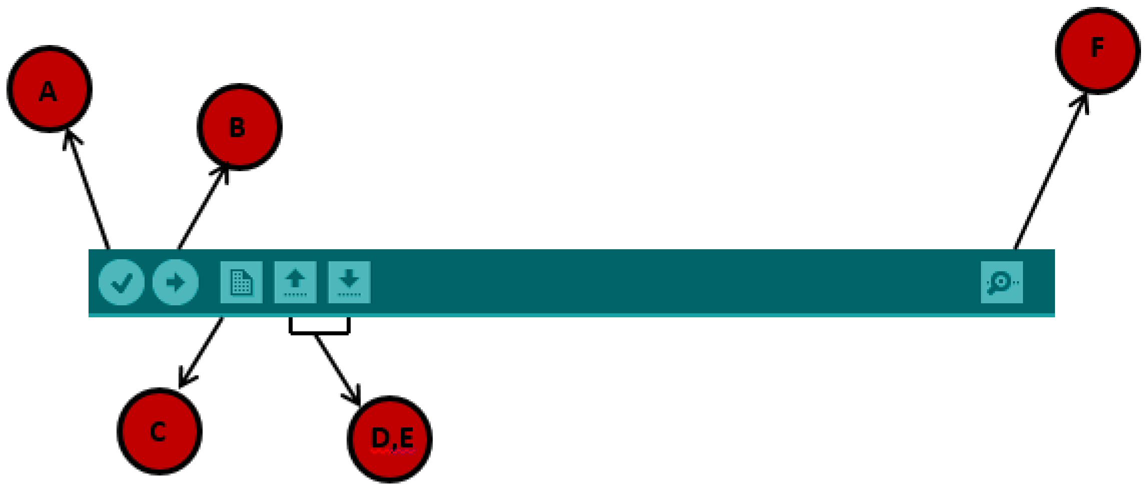
Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A – Used to check if there is any compilation error.

B – Used to upload a program to the Arduino board.

C – Shortcut used to create a new sketch.

D – Used to directly open one of the example sketch.

E – Used to save your sketch.

F – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

3.2 Setting Up NodeMCU ESP8266 Board into Arduino IDE:

Step 1: Connect Your NodeMCU to the Computer

Use the USB cable to connect your NodeMCU to the computer,you will see the blue onboard LED flicker when powered up, but they will not stay lit.

Step 2: Install the COM/Serial Port Driver

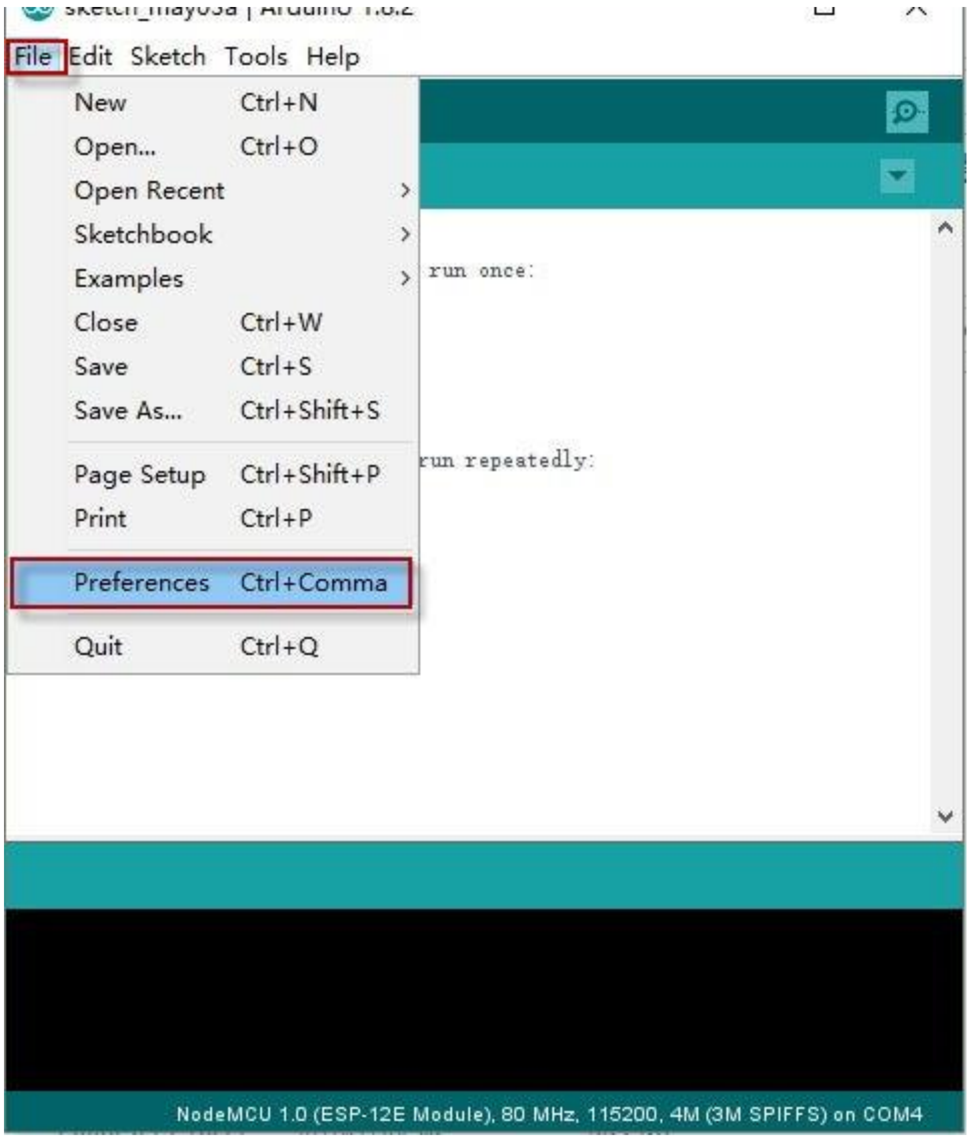
In order to upload code to the ESP8266 and use the serial console, connect any data-capable micro USB cable to ESP8266 IOT Board and the other side to your computer's USB port.

Step 3: Install the Arduino IDE 1.6.4 or Greater

Download Arduino IDE from Arduino.cc (1.6.4 or greater) – don't use 1.6.2! You can use your existing IDE if you have already installed it. You can also try downloading the ready-to-go package from the ESP8266-Arduino project, if the proxy is giving you problems

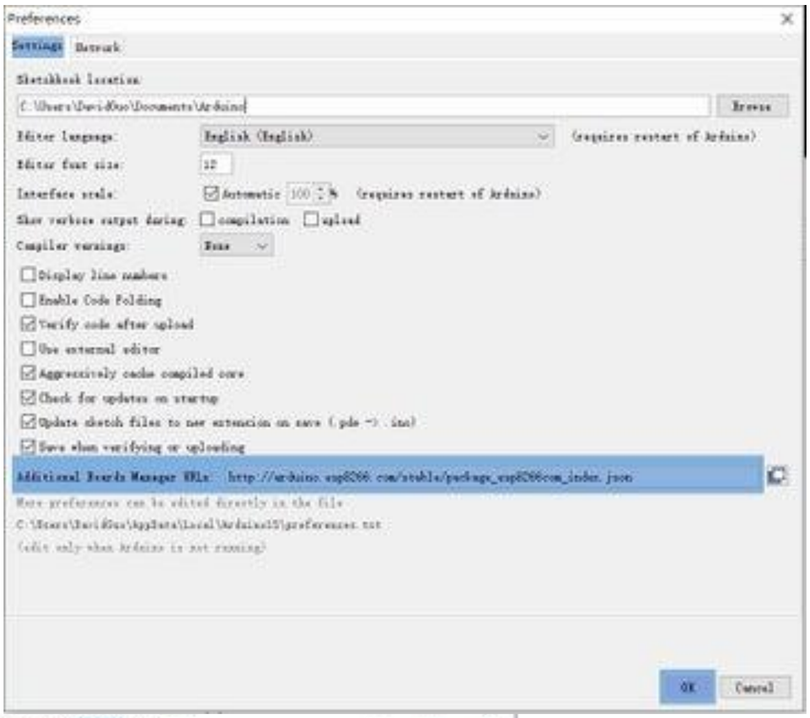
Step 4: Install the ESP8266 Board Package

Enter http://arduino.esp8266.com/stable/package_esp8266... into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences (Open Arduino IDE–>File–>Preferences–>Settings). Enter the link and click “OK” to save your changes. Next, use the Board Manager to install the ESP8266 package Enter the Boards Manager and find the board type as below: Scroll the Broads Manager screen down to the bottom, you will see A module called “esp8266 by esp8266 Community” (see following picture), select the latest version and click “Install“. The ESP8266 package has been installed successfully.



```
run once:
run repeatedly:
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM4



Preferences

Settings | Network

Sketchbook location:

C:\Users\DavidGuo\Documents\Arduino

Editor language: English (English) (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Aggressively cache compiled core

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

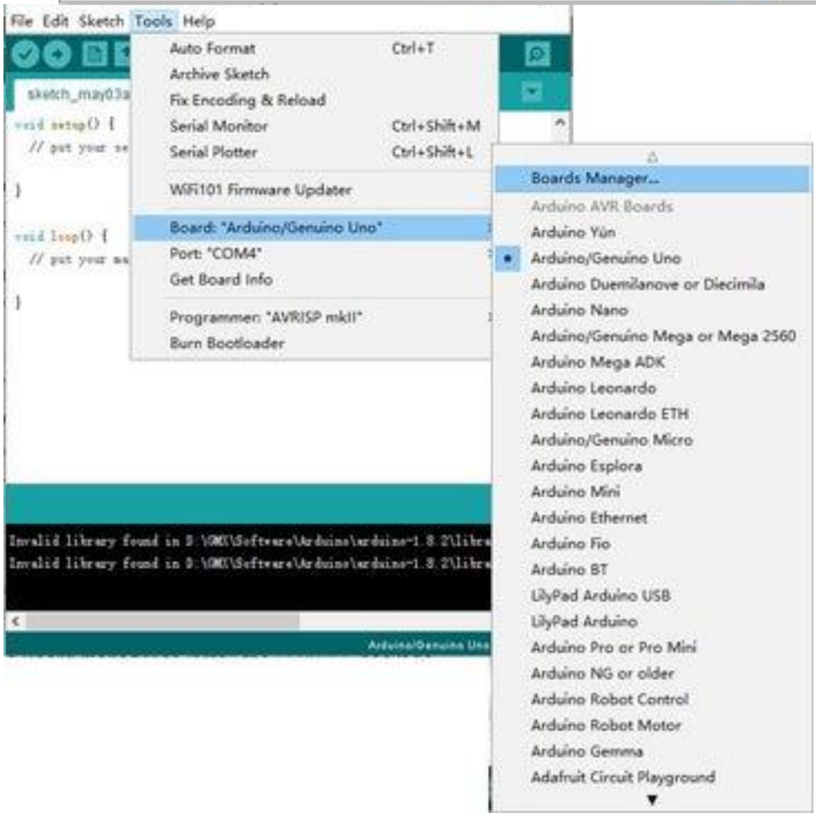
☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

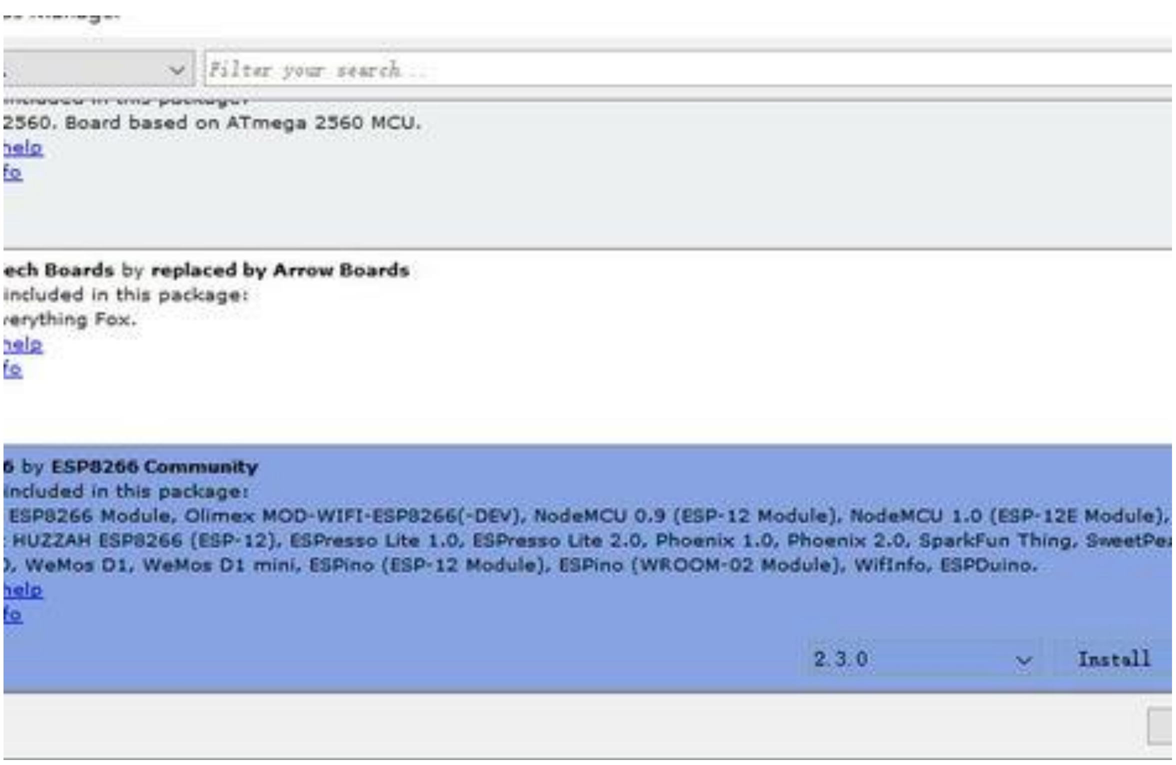
C:\Users\DavidGuo\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)



```
void setup() {
  // put your se
}

void loop() {
  // put your ma
}
```

```
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libra
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libra
```

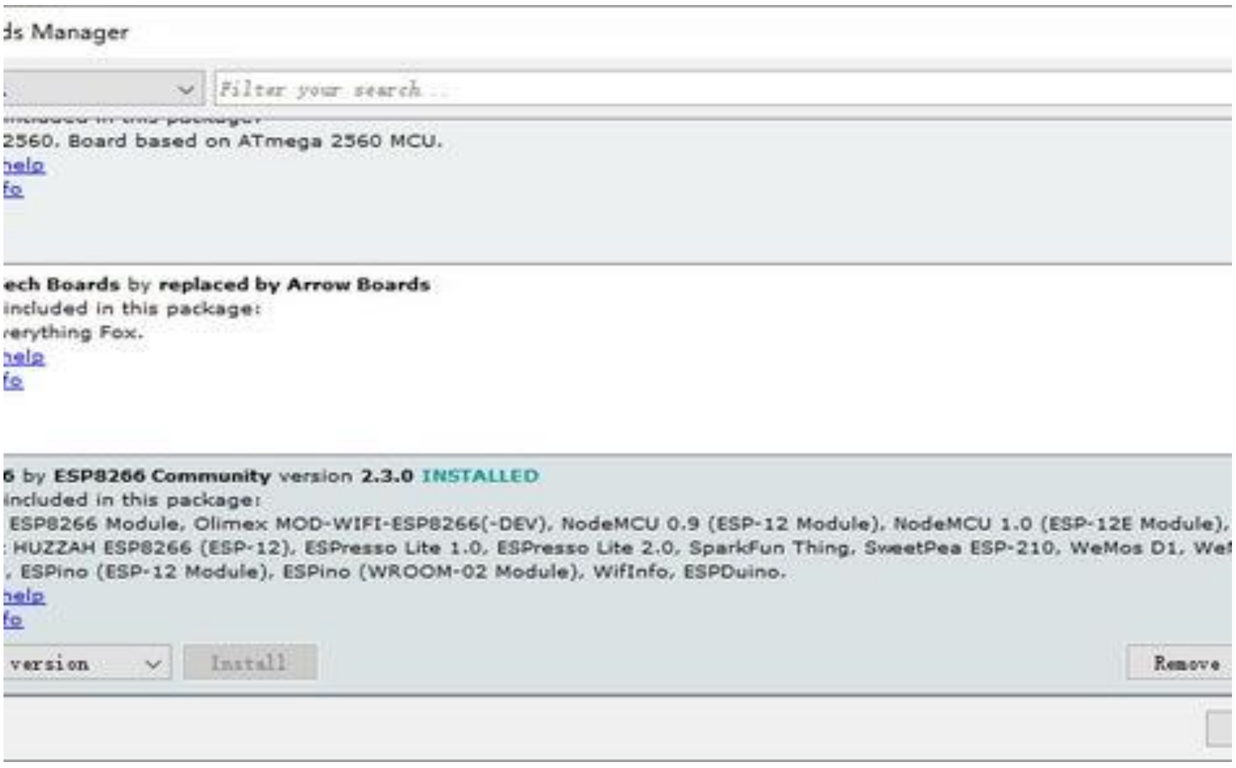

Filter your search ...

2560. Board based on ATmega 2560 MCU.
help
fo

ech Boards by replaced by Arrow Boards
included in this package:
erything Fox.
help
fo

6 by ESP8266 Community
included in this package:
ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module),
HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPe
), WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
help
fo

2.3.0 Install


ds Manager

Filter your search ...

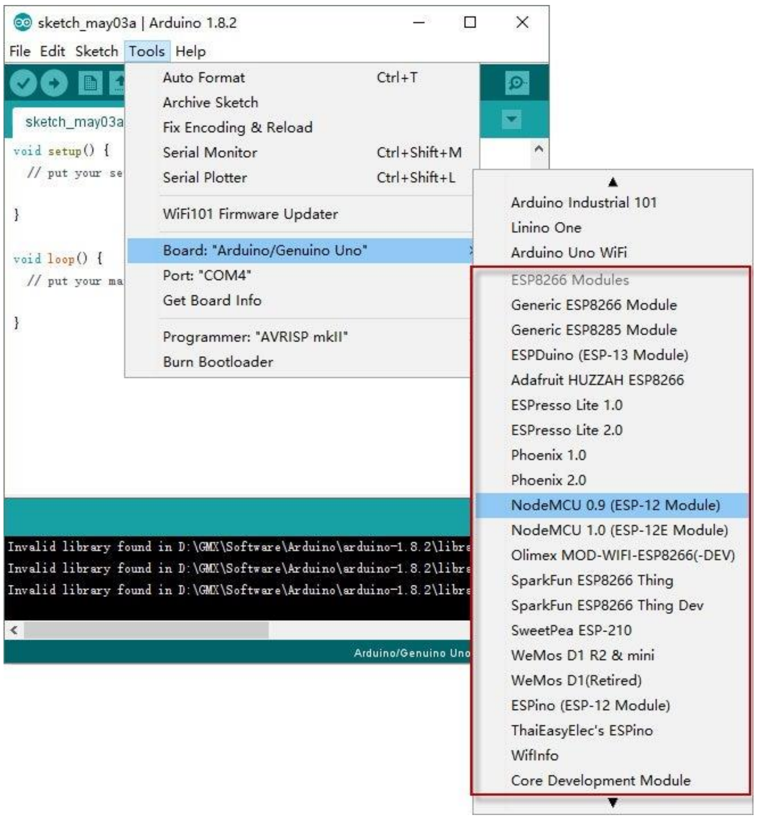
2560. Board based on ATmega 2560 MCU.
help
fo

ech Boards by replaced by Arrow Boards
included in this package:
erything Fox.
help
fo

6 by ESP8266 Community version 2.3.0 INSTALLED
included in this package:
ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module),
HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, We
, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
help
fo

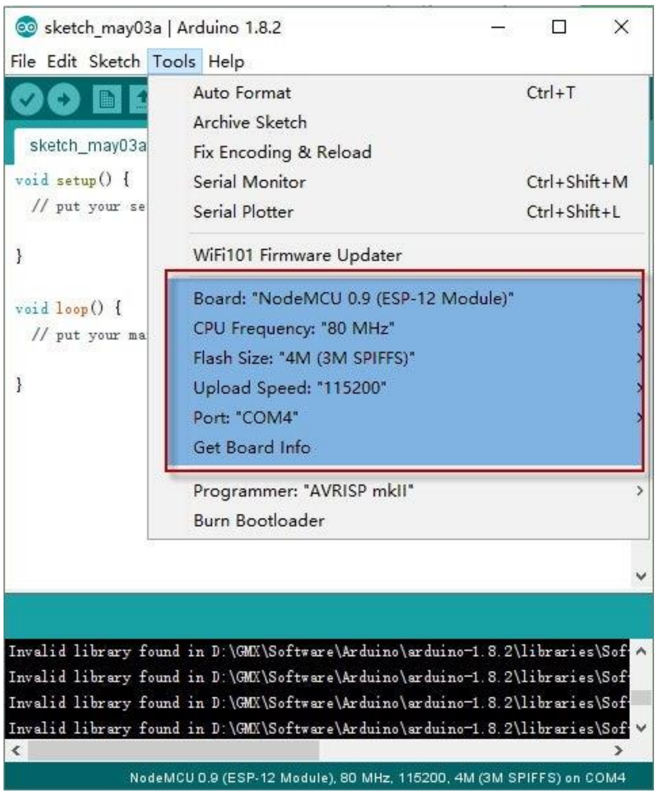
version Install Remove

Step 5: Setup ESP8266 Support



```
sketch_may03a
void setup() {
  // put your se

}

void loop() {
  // put your ma

}
```

```
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libra
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libra
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libra
```


sketch_may03a | Arduino 1.8.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Programmer: "AVRISP mkII"
Burn Bootloader

Board: "NodeMCU 0.9 (ESP-12 Module)"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Upload Speed: "115200"
Port: "COM4"
Get Board Info

```
sketch_may03a
void setup() {
  // put your se

}

void loop() {
  // put your ma

}
```

```
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libraries\Sof
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libraries\Sof
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libraries\Sof
Invalid library found in D:\GMX\Software\Arduino\arduino-1.8.2\libraries\Sof
```

NodeMCU 0.9 (ESP-12 Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM4

When you’ve restarted, select NodeMCU 0.9 (or NodeMCU 1.0) from the Tools->Board dropdown Config the Board menu and choose the right Port for your device. CPU Frequency ：80MHz,Flash Size：4M（3M SPIFFS）,Upload Speed：115200 Now just proceed as the Arduino: Start your sketching! Note: 115200 baud upload speed is a good place to start – later on you can try higher speeds but 115200 is a good safe place to start.

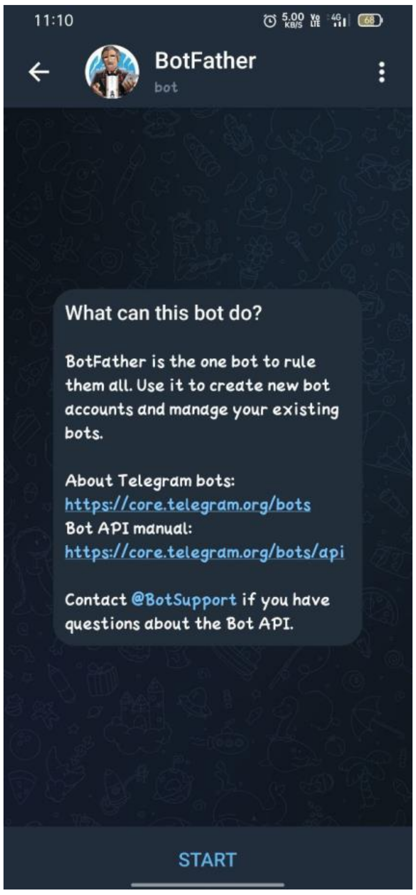
CHAPTER 4

Creation of Telegram Bot:

Telegram is a messaging application which is used to send text, images or video messages free of cost. It also allows using other API to create programs to integrate Telegram in their applications. There are special bots which do not need phone number to set up and can be interfaced with any embedded or software application to trigger some event using telegram text messages.

Install "Telegram" app on your smartphone from your play store or apple store. Sign Up if you are going to use it for the first time with your country code and mobile number. It will ask for OTP code for verification.

After successfully sign up, search for BotFather and open it. Open BotFather and click on RESTART for creating your bot as shown in screenshot below. It will show set of codes which can be used to create and edit bots. Type "/newbot" in the text section for creating new bot.

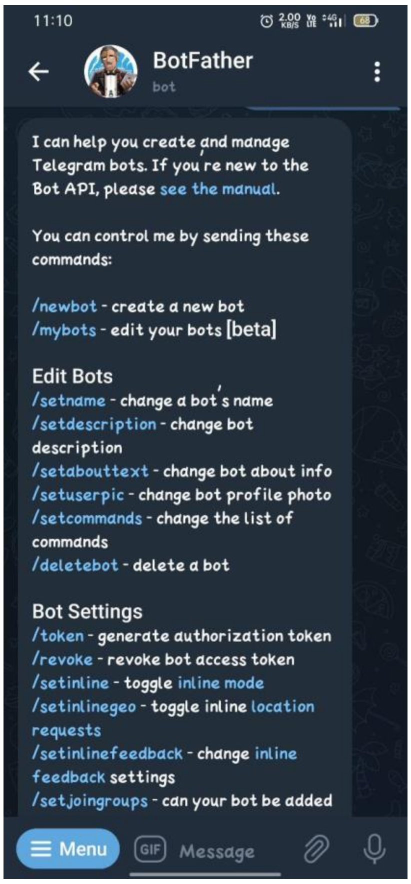


What can this bot do?

BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots.

About Telegram bots:
https://core.telegram.org/bots
Bot API manual:
https://core.telegram.org/bots/api

Contact @BotSupport if you have questions about the Bot API.



11:10

BotFather
bot

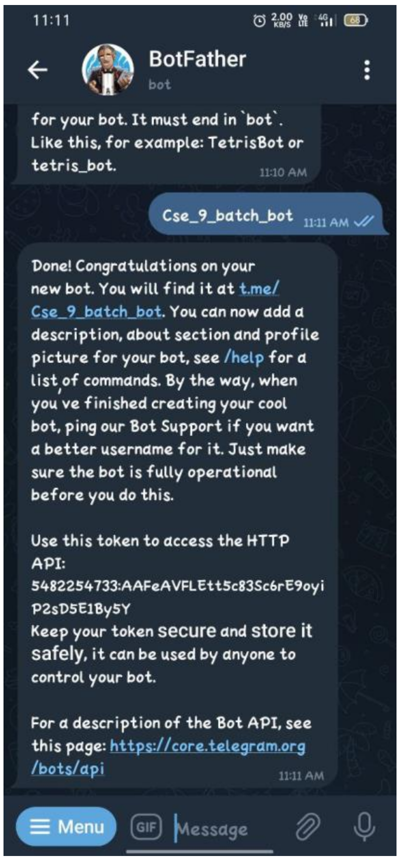
I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.

You can control me by sending these commands:

/newbot - create a new bot
/mybots - edit your bots [beta]

Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot

Bot Settings
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added



11:11

BotFather
bot

for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 11:10 AM

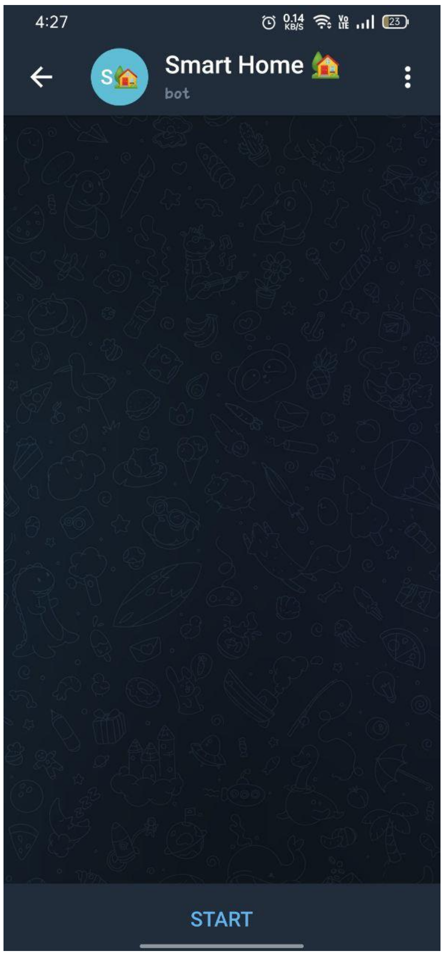
Cse_9_batch_bot 11:11 AM

Done! Congratulations on your new bot. You will find it at t.me/Cse_9_batch_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
5482254733:AAFeAVFLEtt5c83Sc6rE9oyiP2sD5E1By5Y
Keep your token secure and store it safely, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: https://core.telegram.org/bots/api 11:11 AM

Now you have to choose a name for your bot. Type the name of your bot and click on send. Here I gave “Smart Home 🏠” name to my bot. Now it’s time to choose a unique username for your bot. Here I have given “@Cse_9_batch_bot” username to my bot. This username should be unique. After giving username you will receive a text from BotFather about your new bot and will get a token for the new bot. This token will be used in code for NodeMCU. Click on the link given in the message to go to your new chat box of the bot and then click on Start. You can now send your commands by send a text message from Telegram app.



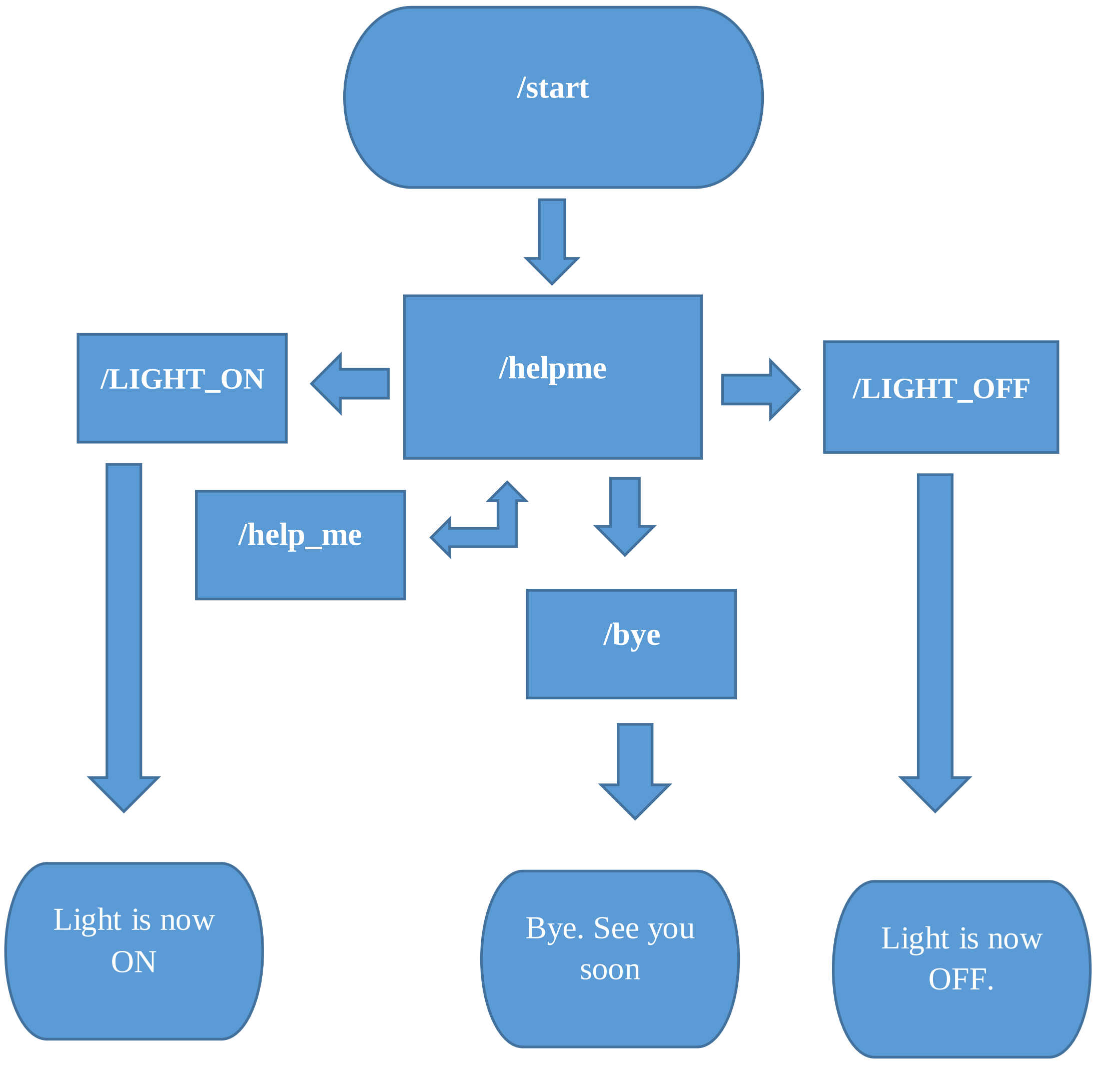
CHAPTER 5

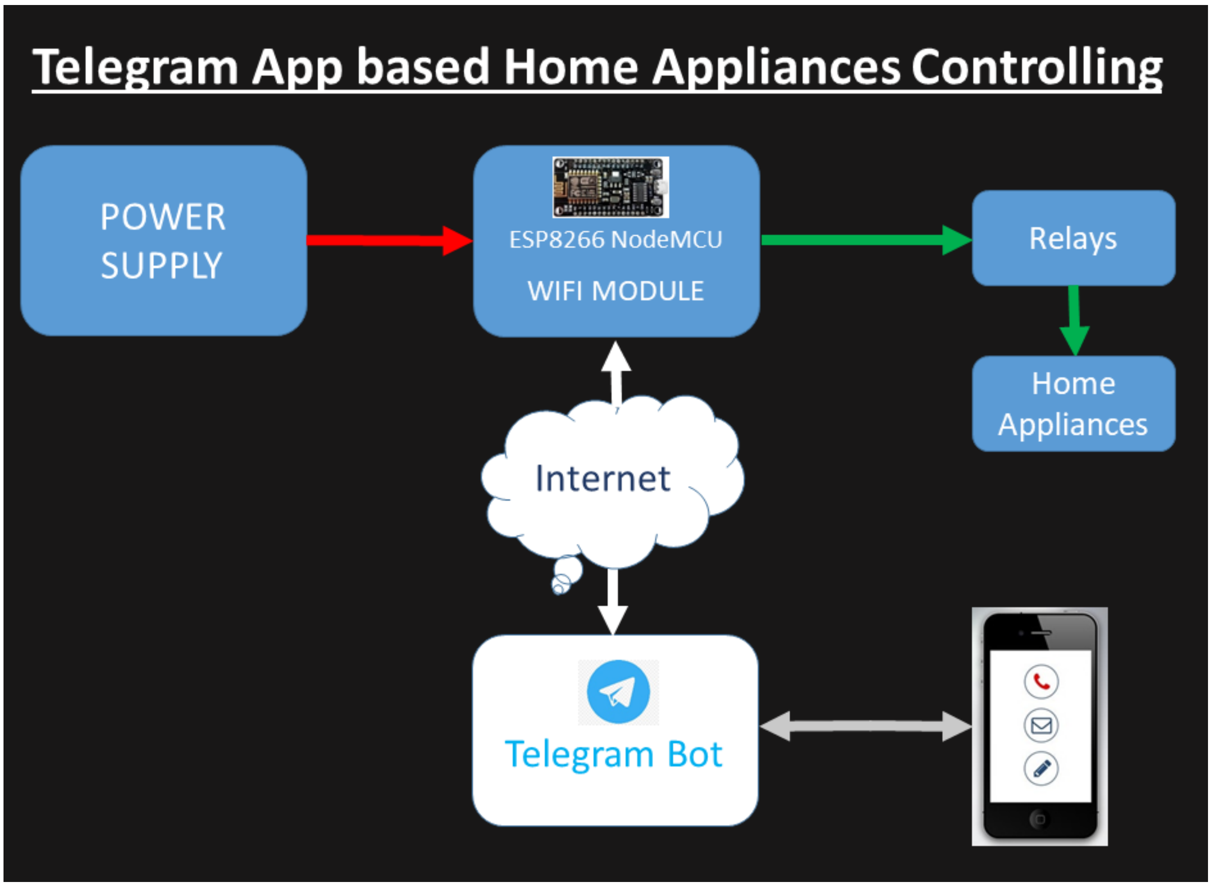
5.1 Algorithm:

```
Step 1: Start
Step 2: Check whether WIFI is connected to NODEMCU , if WIFI connected goto
  Step3. Else goto Step 9.
Step 3: Code in NodeMCU is processed and now telegram bot will be active.
Step 4: If the Telegram access token is true then goto Step 5.Else goto Step 9.
Step 5: /start
Step 6: /helpme
Step 7: Now 4 commands will be displayed. /LIGHT_ON,/LIGHT_OFF,/helpme,/bye
Step 8: If /LIGHT_ON:
                Light will on and bot replies as LIGHT will be ON.
        Else If /LIGHT_OFF:
                Light off and bot replies ad LIGHT will OFF.
        Else If /helpme :
                Goto Step 6.
        Else If /bye:
                Nothing happens and bot replies as Byee See you soon.
        Else:
                Goto Step 4.
Step 9: End.
```

5.2 Flow Chart:

The below is the flow chart designed for our project.




Telegram App based Home Appliances Controlling
POWER SUPPLY
ESP8266 NodeMCU
WIFI MODULE
Relays
Home Appliances
Internet
Telegram Bot

CHAPTER 6

6.1 Code Explanation

Complete code with a working video for this IoT controlled Home Automation is given at the end of this tutorial, here we are explaining the complete program to understand the working of the project.

Replace SSID and password with your Wi-Fi credentials. String token stores the unique token number you got after creating your bot. D0 is the digital output pin for changing the state of the relay.

```
String ssid = "**********";     // Replace with your ssid

String pass = "************"; // Replace with your password

String token = "66****885:A*G-X**dTYdSCt******aCQPCk***SL**b4";   // token number
of your bot

int led = D0;    // digital pin on NodeMCU
```

myBot.wifiConnect() function takes the SSID and password to connect ESP8266 with the Wi-Fi and function *myBot.setTelegramToken()* takes the unique token number which further establishes the connection between telegram bot and NodeMCU. On successful connection you get "*testConnection OK*" on the serial monitor.

```
myBot.wifiConnect(ssid, pass);

myBot.setTelegramToken(token);

if (myBot.testConnection())


                     Serial.println("\ntestConnection OK");

                Else

                     Serial.println("\ntestConnection NOK");
```

TBMessage stores the message received from the telegram bot. *msg.text.equalsIgnoreCase()* checks if the text received matches with string “light on” or “light off”. If the message received is “Light on” it changes the relay state to turn on the lamp and if the message received is “Light off” then it changes the relay state to turn off the lamp. If the text received is different from “light on” and “light off” then it sends welcome message back to the telegram.

```
TBMessage msg;

if (myBot.getNewMessage(msg)) {

   if (msg.text.equalsIgnoreCase("LIGHT ON")) {

       digitalWrite(led, HIGH);

          myBot.sendMessage(msg.sender.id, "Light is now ON");

            }

            else if (msg.text.equalsIgnoreCase("LIGHT OFF")) {

              digitalWrite(led, LOW);

                 myBot.sendMessage(msg.sender.id, "Light is now OFF");

                    }

         else {

           String reply;

            reply = (String)"Welcome " + msg.sender.username + (String)". Try LIGH
T ON or LIGHT OFF.";

          myBot.sendMessage(msg.sender.id, reply);

       }


  }
```

Now after completing the code its time to test the system.

6.2 Source Code:

```
#include "CTBot.h"
CTBot myBot;

String ssid = "VARSHA";     // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
 // REPLACE myPassword YOUR WIFI PASSWORD, IF ANY
String token = "5482254733:AAFeAVFLEtt5c83Sc6rE9oyiP2sD5E1By5Y";   // REPLACE
myToken WITH YOUR TELEGRAM BOT TOKEN
uint8_t led = D0;           // the onboard ESP8266 LED.
                        // If you have a NodeMCU you can use the BUILTIN_LED pin
                        // (replace 2 with BUILTIN_LED)

void setup() {
   // initialize the Serial
   Serial.begin(115200);
   Serial.println("Starting TelegramBot...");

   // connect the ESP8266 to the desired access point
   myBot.wifiConnect(ssid, pass);

   // set the telegram bot token
   myBot.setTelegramToken(token);

   // check if all things are ok
   if (myBot.testConnection())
      Serial.println("\ntestConnection OK");
   else
      Serial.println("\ntestConnection NOK");
```

```
    // set the pin connected to the LED to act as output pin
    pinMode(led, OUTPUT);
    digitalWrite(led, HIGH); // turn off the led (inverted logic!)


}


void loop() {
    // a variable to store telegram message data
    TBMessage msg;


    // if there is an incoming message...
    if (myBot.getNewMessage(msg)) {


        if (msg.text.equalsIgnoreCase("/LIGHT_ON")) {              // if the received message is
"LIGHT ON"...
            digitalWrite(led, LOW);                               // turn on the LED (inverted logic!)
            myBot.sendMessage(msg.sender.id, "Light is now ON 💡 ");  // notify the sender
        }
        else if (msg.text.equalsIgnoreCase("/LIGHT_OFF")) {        // if the received message is
"LIGHT OFF"...
            digitalWrite(led, HIGH);                              // turn off the led (inverted logic!)
            myBot.sendMessage(msg.sender.id, "Light is now OFF"); // notify the sender
        }
        else if (msg.text.equals("/helpme")) {
      digitalWrite(led, HIGH);
      myBot.sendMessage(msg.sender.id, "Try sending following commands\n");
      myBot.sendMessage(msg.sender.id, "/LIGHT_ON\n");
      myBot.sendMessage(msg.sender.id, "/LIGHT_OFF\n");
      myBot.sendMessage(msg.sender.id, "/helpme\n");
```

```
     myBot.sendMessage(msg.sender.id, "/Byee\n");
   }
   else if (msg.text.equals("/Byee")){
            String r;
            r = (String)"Byeeee  " + msg.sender.username + (String)" .See you soon 👋  !!! ";
            myBot.sendMessage(msg.sender.id, r);
   }
      else {                                         // otherwise...
         // generate the message for the sender
         String reply;
         reply = (String)"Welcome  😉 " + msg.sender.username + (String)".Try /helpme";
         myBot.sendMessage(msg.sender.id, reply);            // and  send it
      }
   }
   // wait 500 milliseconds
   delay(500);
}
```

6.3 Implementation:

Here 1st we need to load the code into NodeMCU and then we need to connet the NodeMCU with power and WIFI connection so that it works fine.

After the code gets compiled successfully we need to open the telegram bot.

After opening the Telegram Bot we need to click on START. Now the bot starts replying the user. Bot keeps the commands which the user can use. The user needs to click on the command so that the command will work.

If it is Light On then the light which we connected will ON.

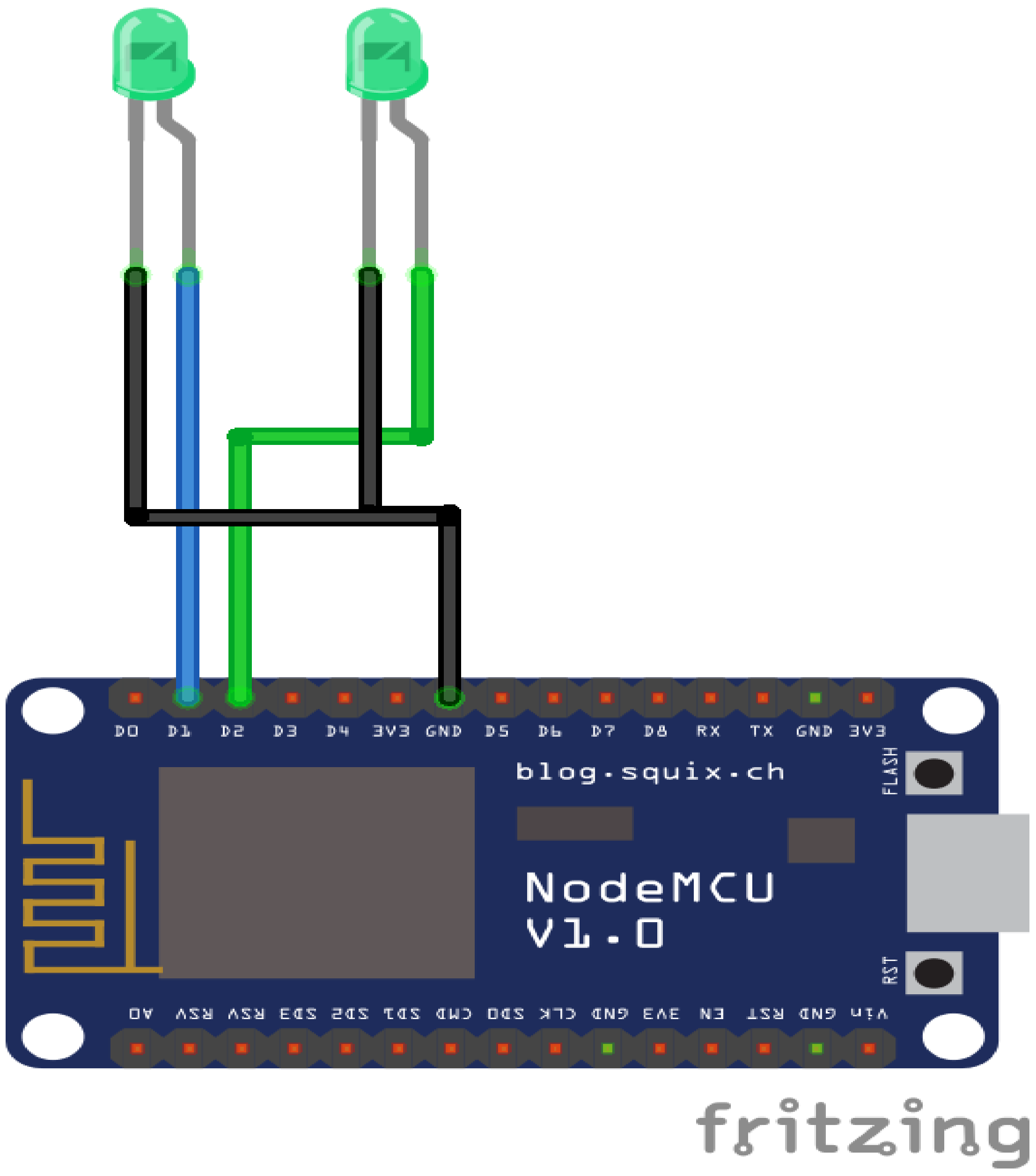
If it is Light Off then the light which we connected will go OFF.

If it is Byee then it Sends off the user with a message.

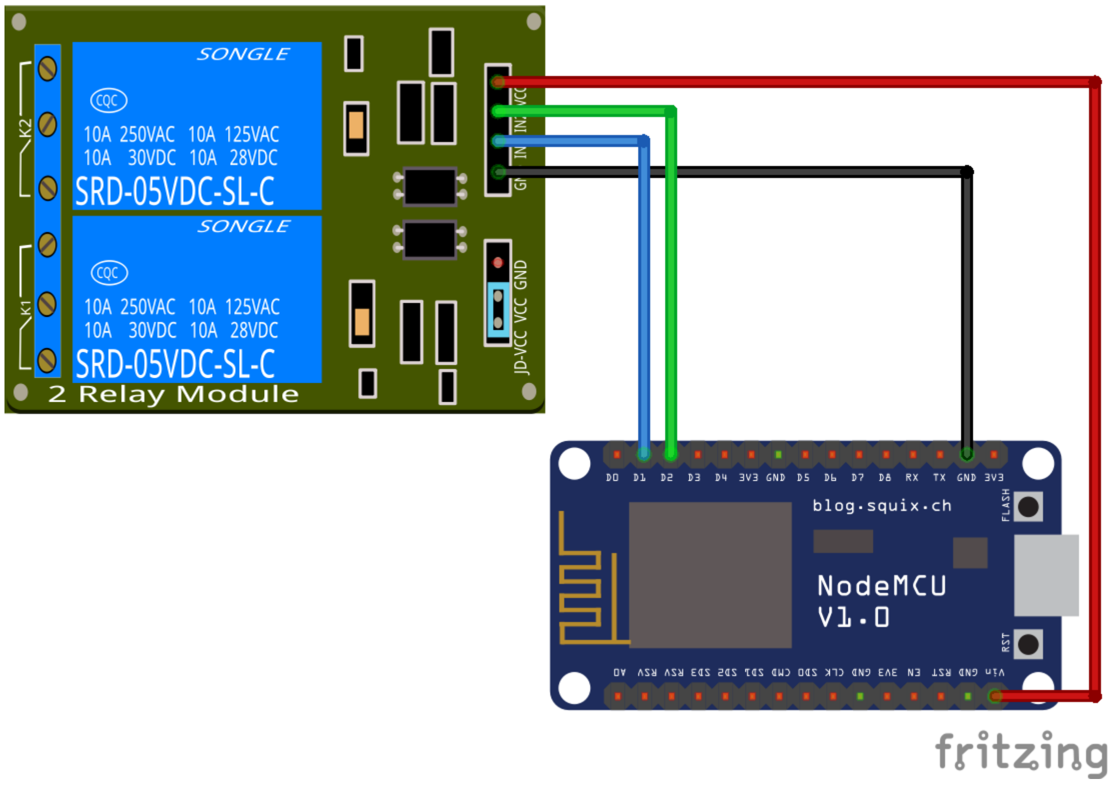
If it is Help me the loop will be continued.

Circuit Diagrams:

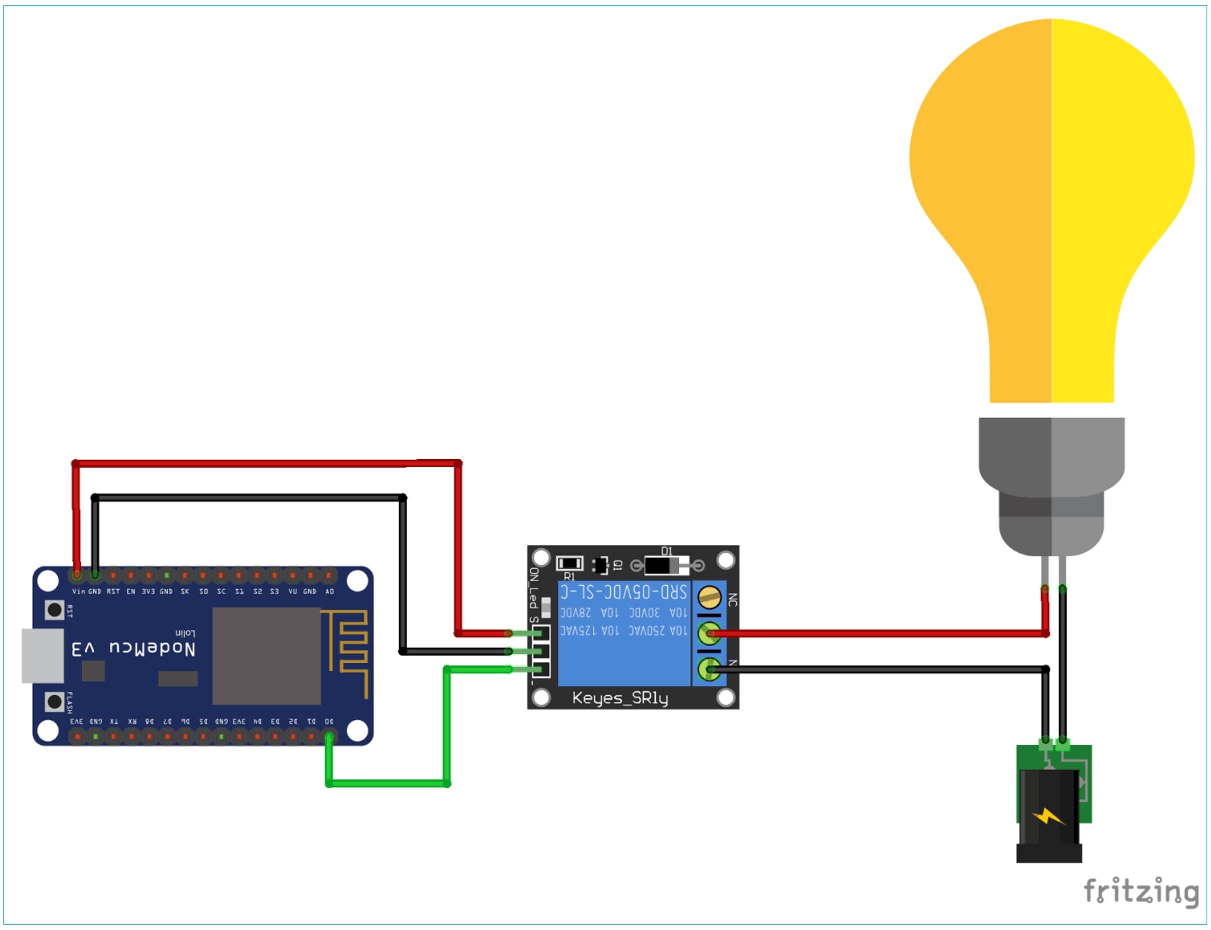
Node MCU



Relay Module:



Project Circuit Diagram:

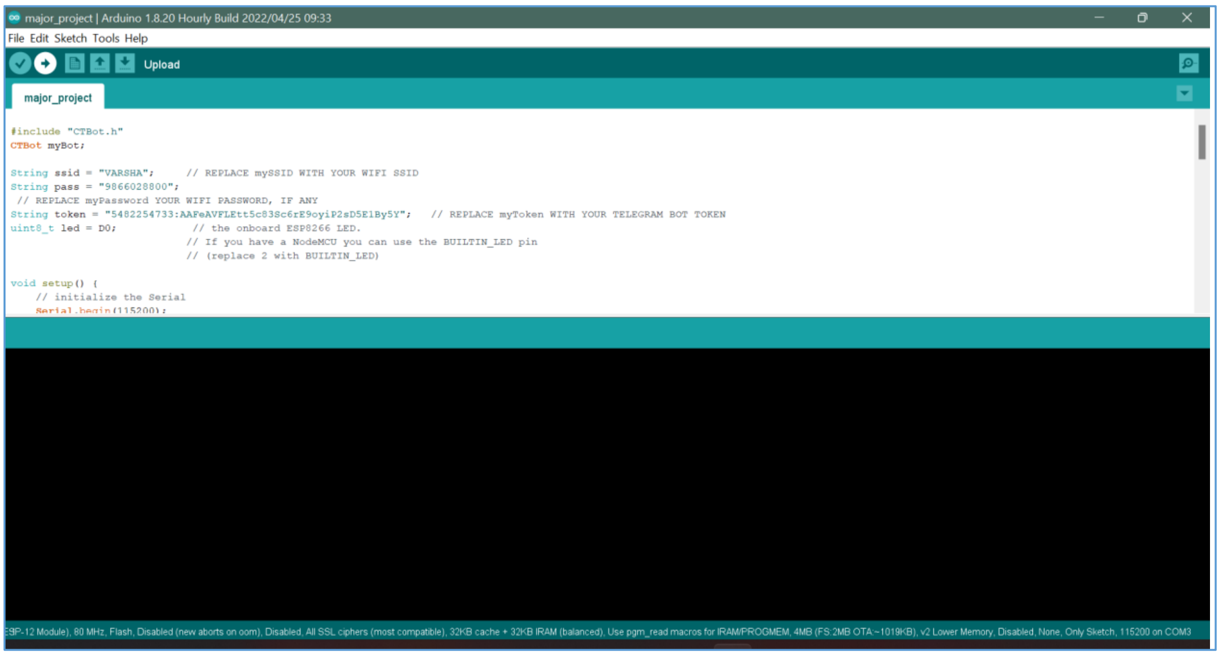


6.4 Output:

6.4(a) Software:

(i)Arduino IDE:

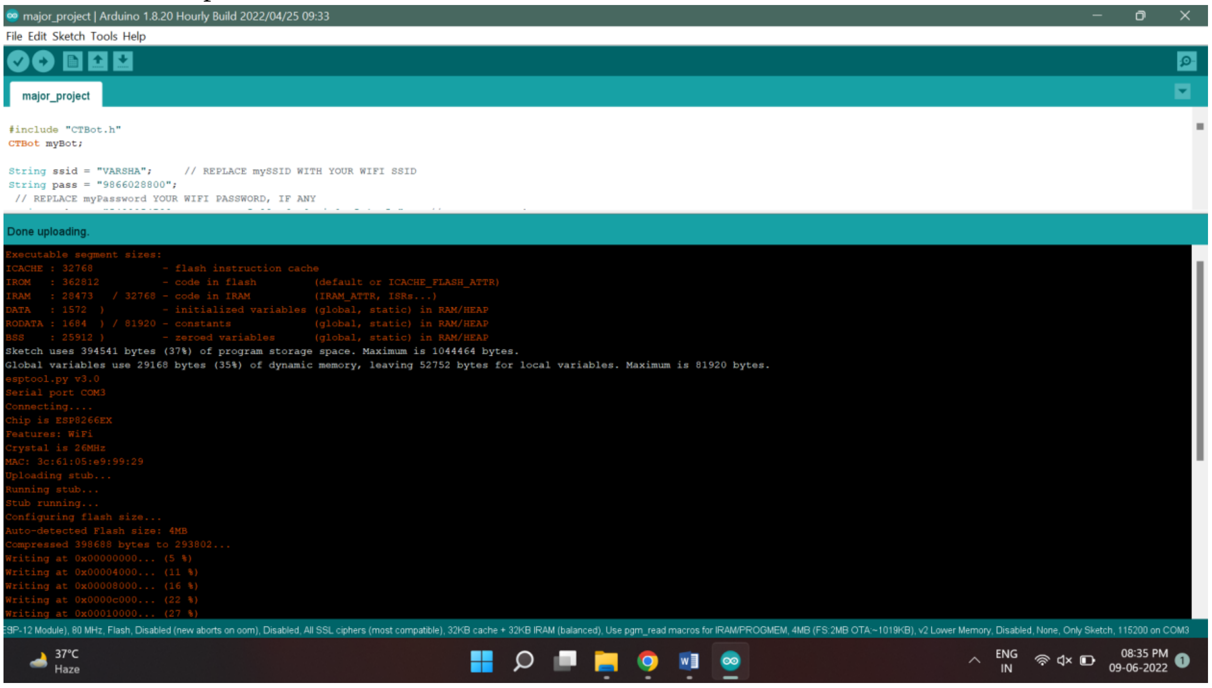
Firstly we need to select the board and tools in the arduino which we downloaded earlier.


major_project | Arduino 1.8.20 Hourly Build 2022/04/25 09:33
File Edit Sketch Tools Help
Upload
major_project

```
#include "CTBot.h"
CTBot myBot;

String ssid = "VARSHA";      // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
 // REPLACE myPassword YOUR WIFI PASSWORD, IF ANY
String token = "5482254733:AAFeAVFLEtt5c83Sc6rE9oyiP2sD5E1By5Y";   // REPLACE myToken WITH YOUR TELEGRAM BOT TOKEN
uint8_t led = D0;             // the onboard ESP8266 LED.
                             // If you have a NodeMCU you can use the BUILTIN_LED pin
                             // (replace 2 with BUILTIN_LED)


void setup() {
    // initialize the Serial
    Serial.begin(115200);
```

ESP-12 Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Here we have uploaded the code .

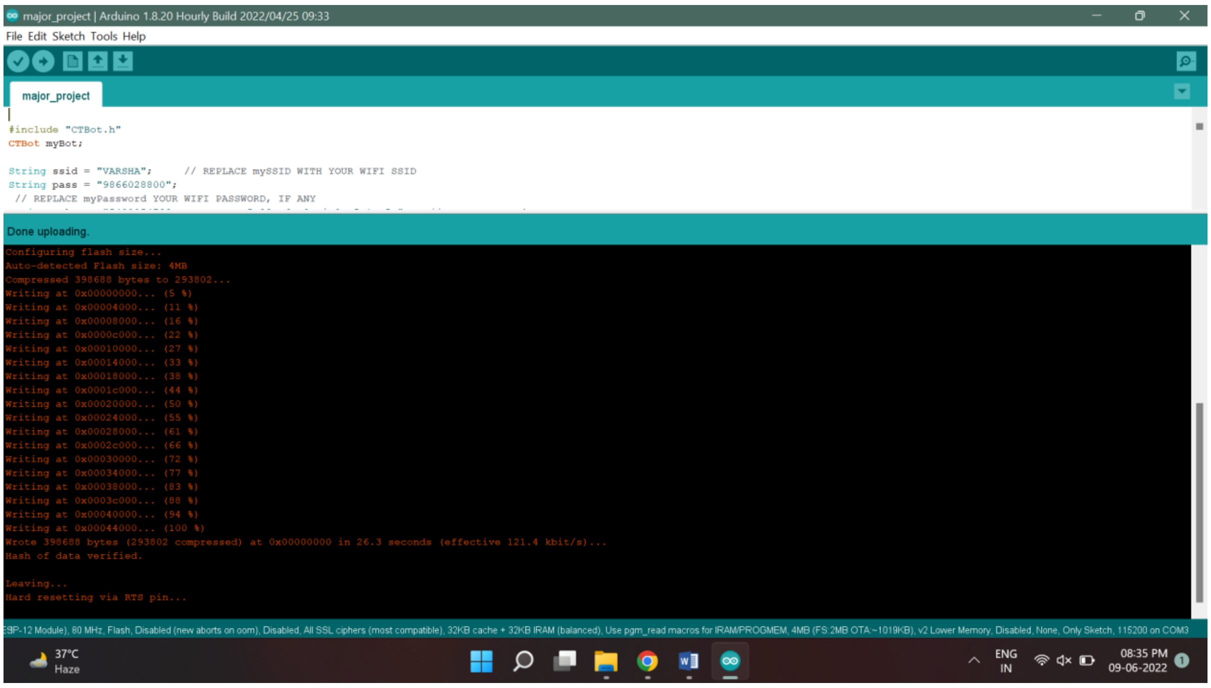

major_project | Arduino 1.8.20 Hourly Build 2022/04/25 09:33
File Edit Sketch Tools Help
major_project
37°C
Haze
ENG
IN
08:35 PM
09-06-2022

```
#include "CTBot.h"
CTBot myBot;

String ssid = "VARSHA";      // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
 // REPLACE myPassword YOUR WIFI PASSWORD, IF ANY
```

Done uploading.

```
Executable segment sizes:
ICACHE : 32768           - flash instruction cache
IROM   : 362812          - code in flash         (default or ICACHE_FLASH_ATTR)
IRAM   : 28473   / 32768 - code in IRAM          (IRAM_ATTR, ISRs...)
DATA   : 1572  )         - initialized variables (global, static) in RAM/HEAP
RODATA : 1684  ) / 81920 - constants             (global, static) in RAM/HEAP
BSS    : 25912 )         - zeroed variables      (global, static) in RAM/HEAP
Sketch uses 394541 bytes (37%) of program storage space. Maximum is 1044464 bytes.
Global variables use 29168 bytes (35%) of dynamic memory, leaving 52752 bytes for local variables. Maximum is 81920 bytes.
esptool.py v3.0
Serial port COM3
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 3c:61:05:e9:99:29
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 398688 bytes to 293802...
Writing at 0x00000000... (5 %)
Writing at 0x00004000... (11 %)
Writing at 0x00008000... (16 %)
Writing at 0x0000c000... (22 %)
Writing at 0x00010000... (27 %)
```

ESP-12 Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3



major_project | Arduino 1.8.20 Hourly Build 2022/04/25 09:33
File Edit Sketch Tools Help
major_project

```
#include "CTBot.h"
CTBot myBot;

String ssid = "VARSHA";      // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
 // REPLACE myPassword YOUR WIFI PASSWORD, IF ANY
```

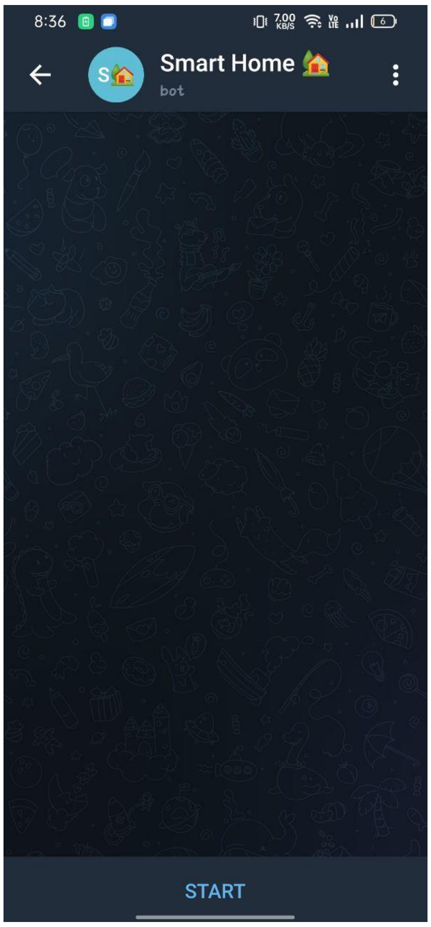
Done uploading.

```
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 398688 bytes to 293802...
Writing at 0x00000000... (5 %)
Writing at 0x00004000... (11 %)
Writing at 0x00008000... (16 %)
Writing at 0x0000c000... (22 %)
Writing at 0x00010000... (27 %)
Writing at 0x00014000... (33 %)
Writing at 0x00018000... (38 %)
Writing at 0x0001c000... (44 %)
Writing at 0x00020000... (50 %)
Writing at 0x00024000... (55 %)
Writing at 0x00028000... (61 %)
Writing at 0x0002c000... (66 %)
Writing at 0x00030000... (72 %)
Writing at 0x00034000... (77 %)
Writing at 0x00038000... (83 %)
Writing at 0x0003c000... (88 %)
Writing at 0x00040000... (94 %)
Writing at 0x00044000... (100 %)
Wrote 398688 bytes (293802 compressed) at 0x00000000 in 26.3 seconds (effective 121.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

ESP-12 Module), 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

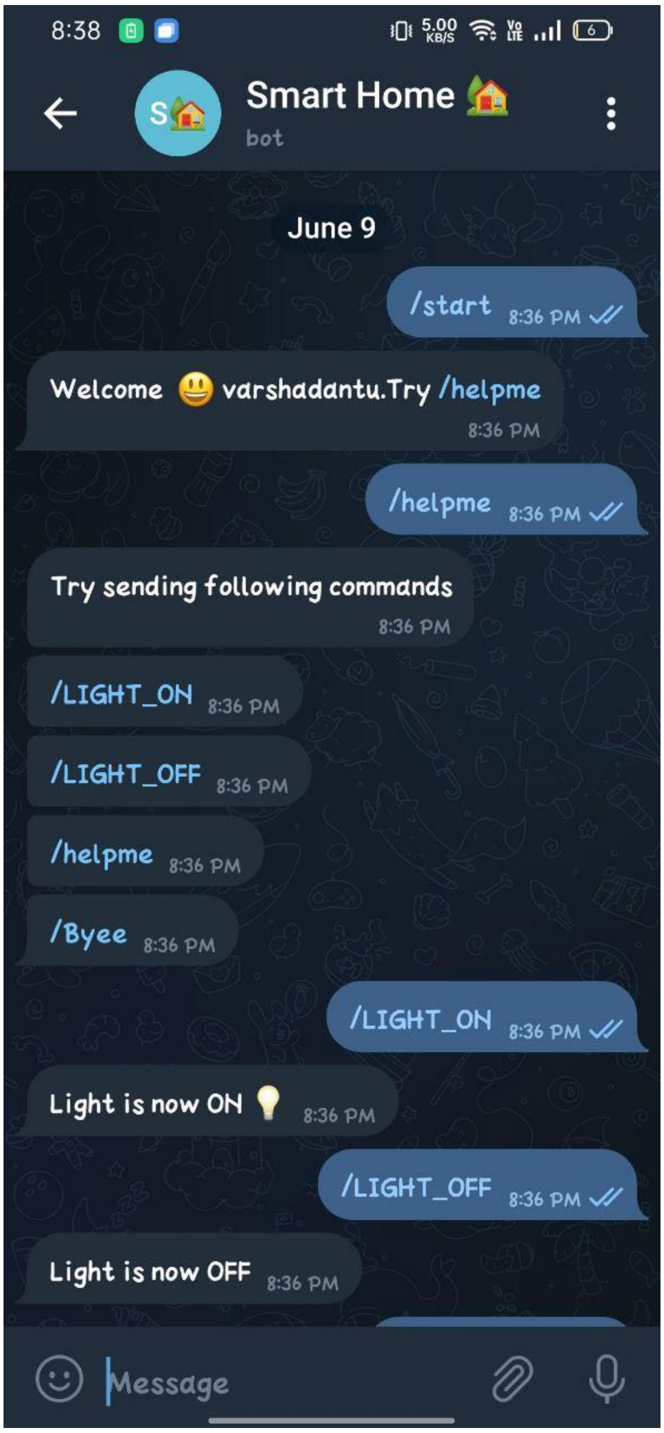
Now the code compiled successfully and done uploading into the NODEMCU.

(ii) Telegram Bot:



/start → /helpme → /LIGHT_ON → Light is now ON.

/start → /helpme → /LIGHT_OFF → Light is now OFF.



8:38

Smart Home 🏡
bot

June 9

/start 8:36 PM

Welcome 😃 varshadantu.Try /helpme 8:36 PM

/helpme 8:36 PM

Try sending following commands 8:36 PM

/LIGHT_ON 8:36 PM

/LIGHT_OFF 8:36 PM

/helpme 8:36 PM

/Byee 8:36 PM

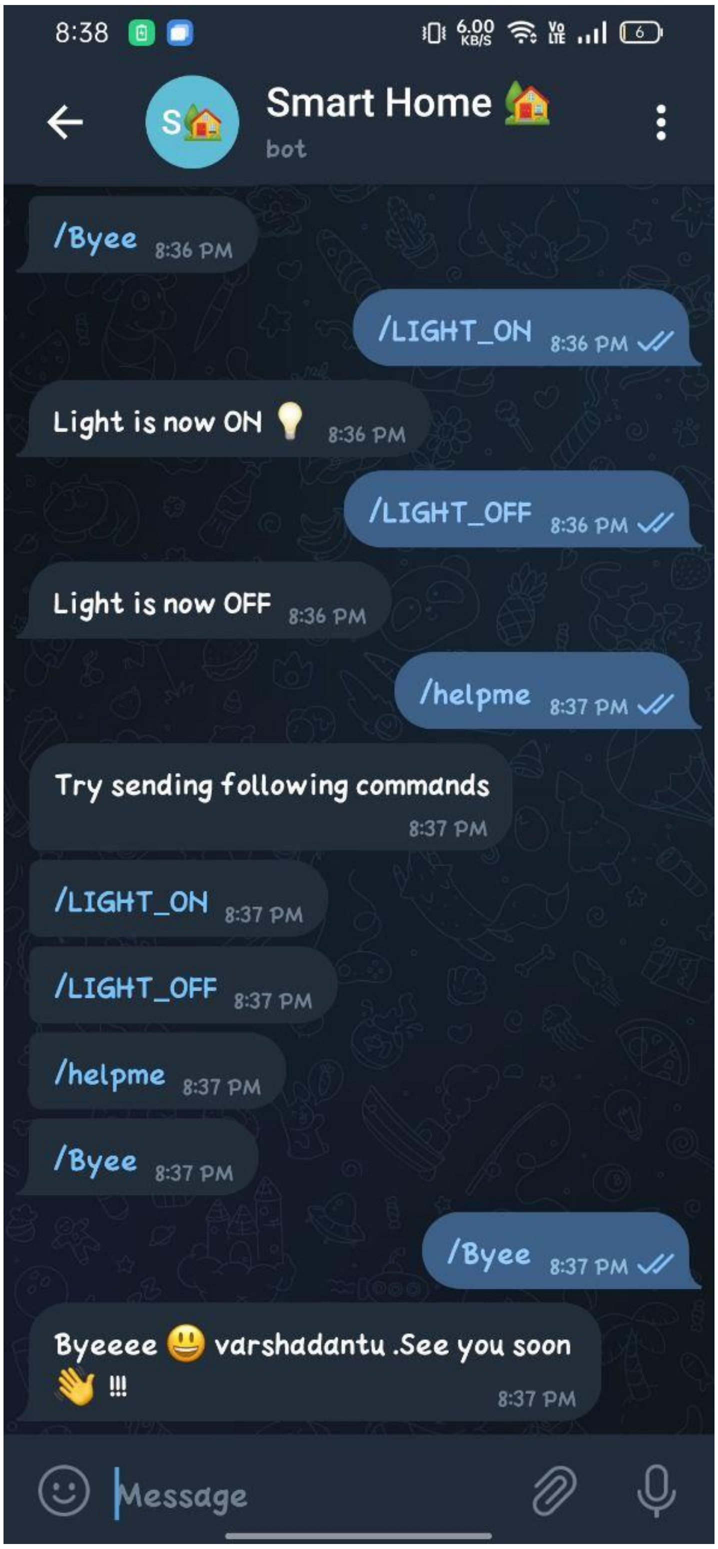
/LIGHT_ON 8:36 PM

Light is now ON 💡 8:36 PM

/LIGHT_OFF 8:36 PM

Light is now OFF 8:36 PM

/start→ /helpme→/Byee → (username). See you soon.



8:38

Smart Home 🏡
bot

/Byee 8:36 PM

/LIGHT_ON 8:36 PM

Light is now ON 💡 8:36 PM

/LIGHT_OFF 8:36 PM

Light is now OFF 8:36 PM

/helpme 8:37 PM

Try sending following commands 8:37 PM

/LIGHT_ON 8:37 PM

/LIGHT_OFF 8:37 PM

/helpme 8:37 PM

/Byee 8:37 PM

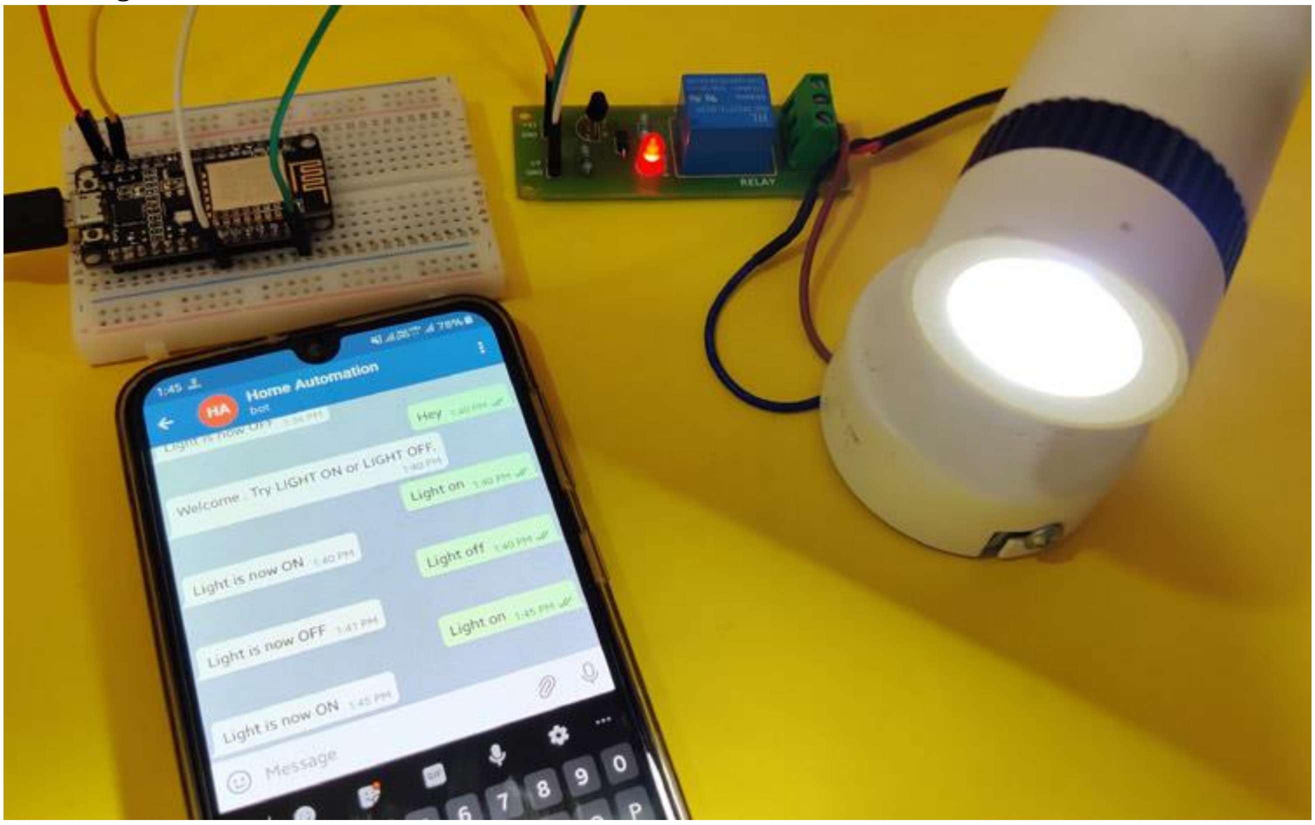
/Byee 8:37 PM

Byeeee 😃 varshadantu .See you soon 👋 !!! 8:37 PM

Message

(iii) Hardware:

Whole figure.



Hey 1:40 PM

Welcome . Try LIGHT ON or LIGHT OFF. 1:40 PM

Light on 1:40 PM

Light is now ON 1:40 PM

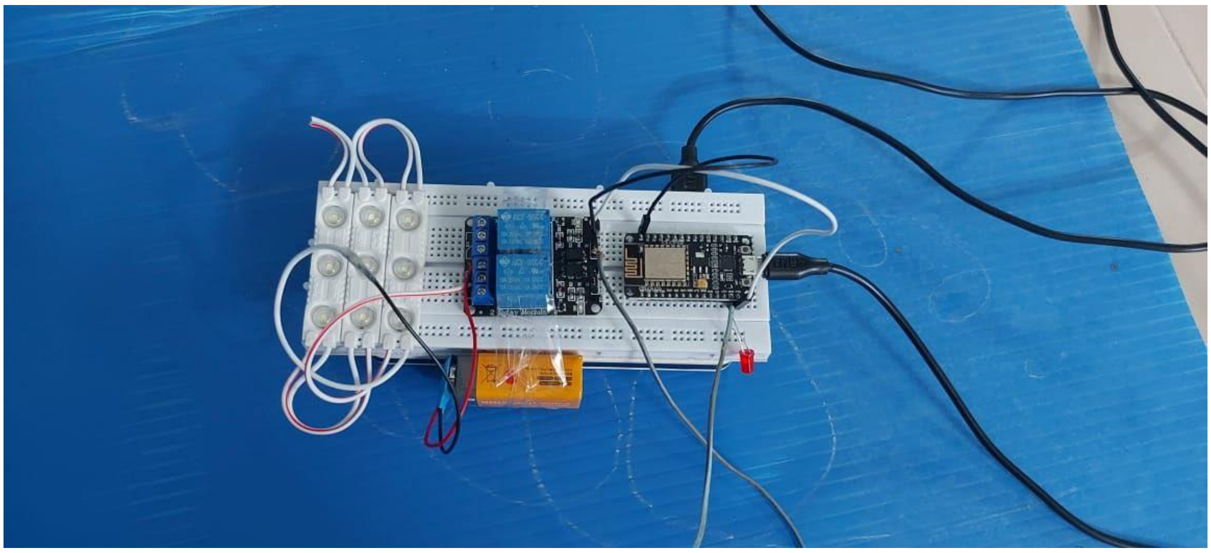
Light off 1:40 PM

Light is now OFF 1:41 PM

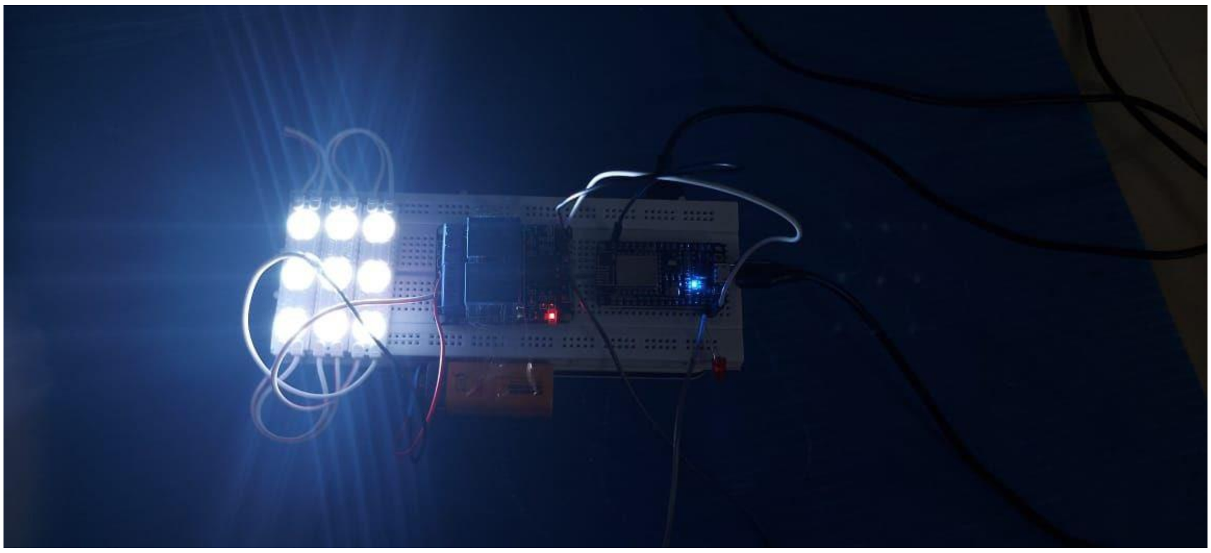
Light on 1:45 PM

Light is now ON 1:45 PM

Light Off:



Light On:



CHAPTER 7

Conclusion

This project is easy to build and can be used effectively. This project responds slowly and this is a major drawback of this project. Else this project is good as you don’t have to install any other app to control your devices. You can edit this code and control 4 or 6 devices.

In a intense study of Internet of Things, I found it to be hypothetical based on its purpose of application. It means IoT provides a lot of automation by connecting things. Connecting things has been made easy by the various sensors and embedding them them to the devices. In my instance of application with telegram messenger, I have used bots to communicate with the connected things in a house. Later by the statistical report given by the bot any user can take the decision on automation of house. With the emerging technologies, I have been successful in exploring the connectivity of various things in a house and also as a network specialist I also reported the pros and cons of connecting things. In future I would see the connected things at a different applicative arena, where it could be used in all contingency situations. It is found that IoT can be implemented and integrated with any software application. So this thesis is just an instance of IoT's implementation using a messenger application.

Now we know how to create a Telegram Bot to interact with the ESP32 or ESP8266. With this bot, you can use your Telegram account to send messages to the ESP and control its outputs. The ESP can also interact with the bot to send responses.

We’ve shown you a simple example on how to control an output. The idea is to modify the project to add more commands to execute other tasks. For example, you can request sensor readings or send a message when motion is detected.

The great thing about using Telegram to control your ESP boards, is that as long as you have an internet connection (and your boards too), you can control and monitor them from anywhere in the world.

REFERENCES

1. Ishan Krishna, K. Lavanya,"Intelligent Home Automation System using BitVoicer", 11th International Conference on Intelligent Systems and Control, 2017.
2. Ramón Alcarria, Diego Martín de Andrés, "A Service-Oriented Monitoring System Based on Rule Evaluation for Home Automation", IEEE 2016.
3. Hattie Clougherty, Alec Brown, Margaret Stonerock, "Home Automation and Personalization through Individual Location Determination",IEEE 978-1-5386-1848-6/17/$31.00 2017.
4. ShibliNisar, Muhammad Asadullah, "Home Automation Using Spoken Pashto Digits Recognition", IEEE 978-1-5090-3310-2/17/$3\.00 2017.
5. Sukhen Das, souvikghosh, RishirajSarker, "A Bluetooth Based Sophisticated Home Automation system Using Smartphone", international conference on intelligent power and instrumentation, 2016.
6. Juan Carlos de Oliveira, Danilo Henrique Santos," Chatting with Arduino Platform through Telegram Bot", IEEE International Symposium on Consumer Electronics, 2016.
7. Vedan rattan vasta , Gopal singh, "Raspberry Pi based Implementation of Internet of Things using Mobile Messaging Application -„Telegram‟", International Journal of Computer Applications (0975 – 8887), Volume 145 – No.14, July 2016.
8. Ankush B.powar, shasikanth gumhre, "A SURVEY ON IoT APPLICATIONS, SECURITY CHALLENGES AND COUNTER MEASURES",international conference on computing, dec 19-21, 2016.