

# STEPS FOR EXECUTION

## **Implementation:**

Here 1<sup>st</sup> we need to load the code into NodeMCU and then we need to connect 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.

## **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.

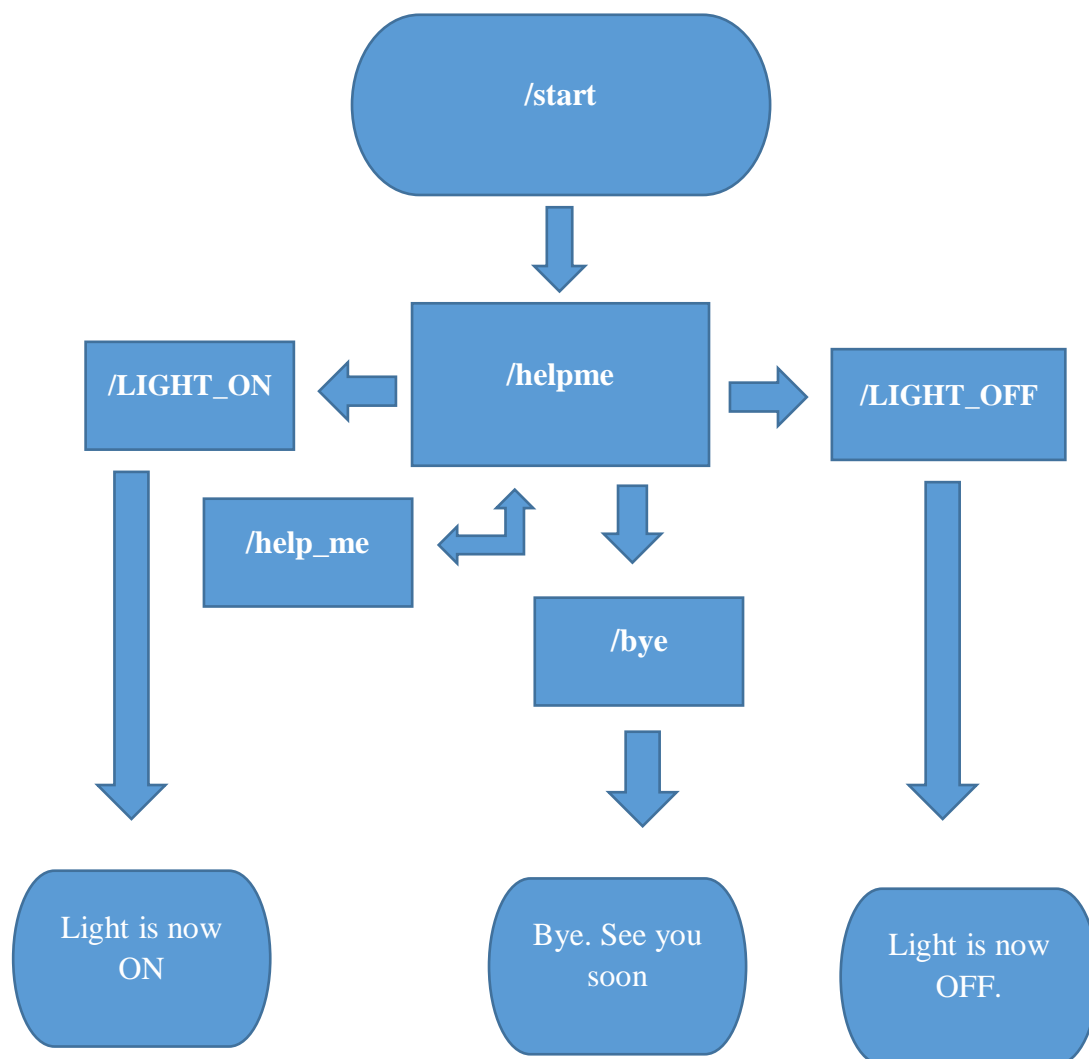
Else:

Goto Step 4.

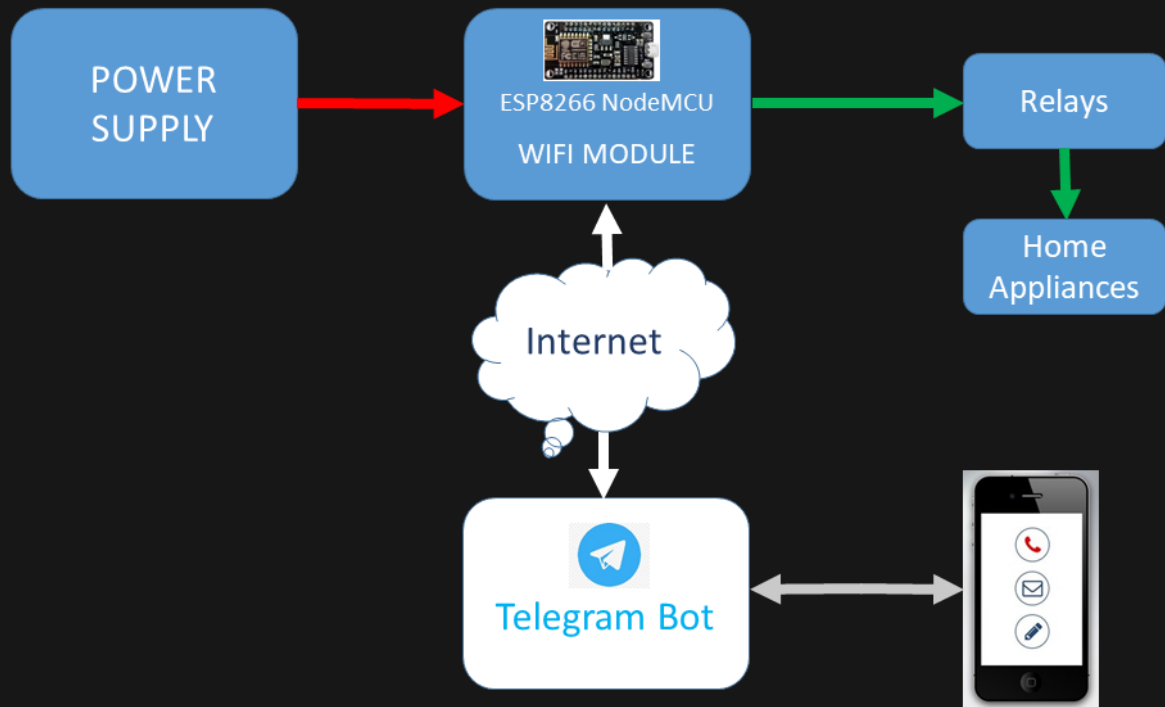
**Step 9:** End.

### **Flow Chart:**

The below is the flow chart designed for our project.



## Telegram App based Home Appliances Controlling



## 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*****aQPck***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 LIGHT ON or LIGHT OFF.";
        myBot.sendMessage(msg.sender.id, reply);
    }
}
```

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

## **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!)

}
```

[illegible]

```

// 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

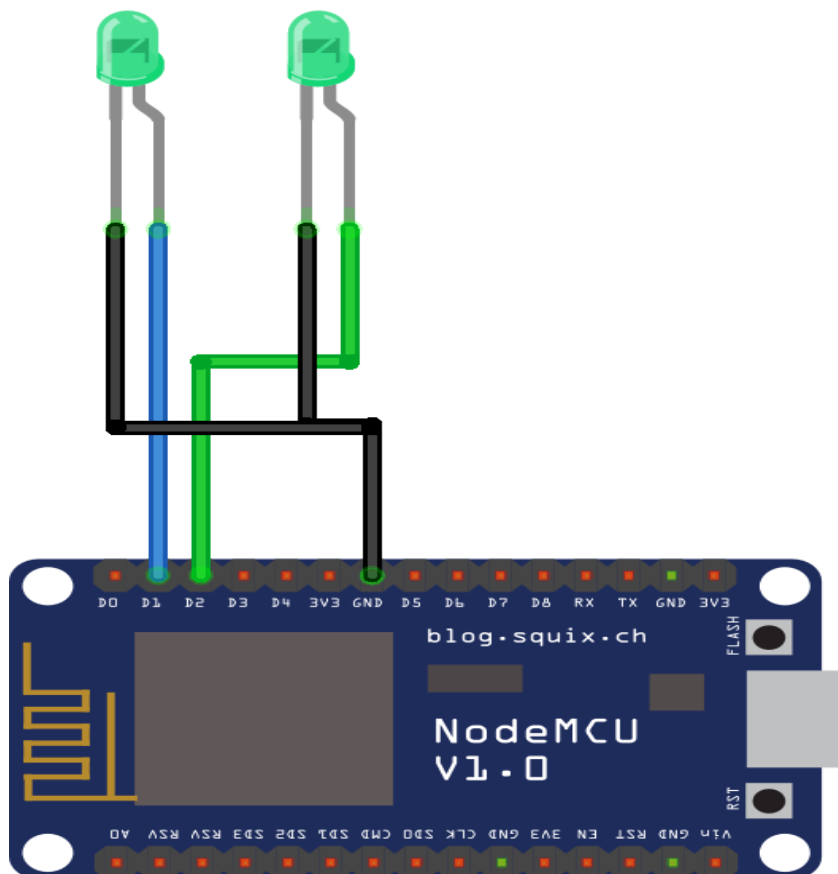
delay(500);

}

```

## Circuit Diagrams:

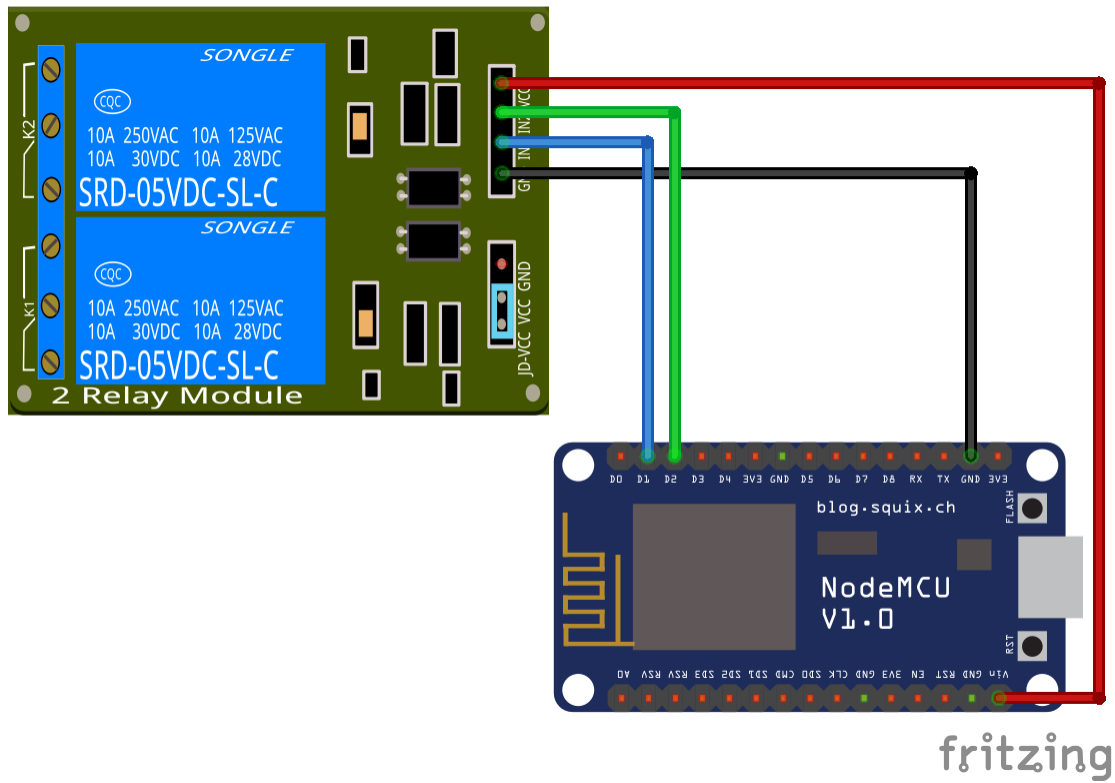
### Node MCU



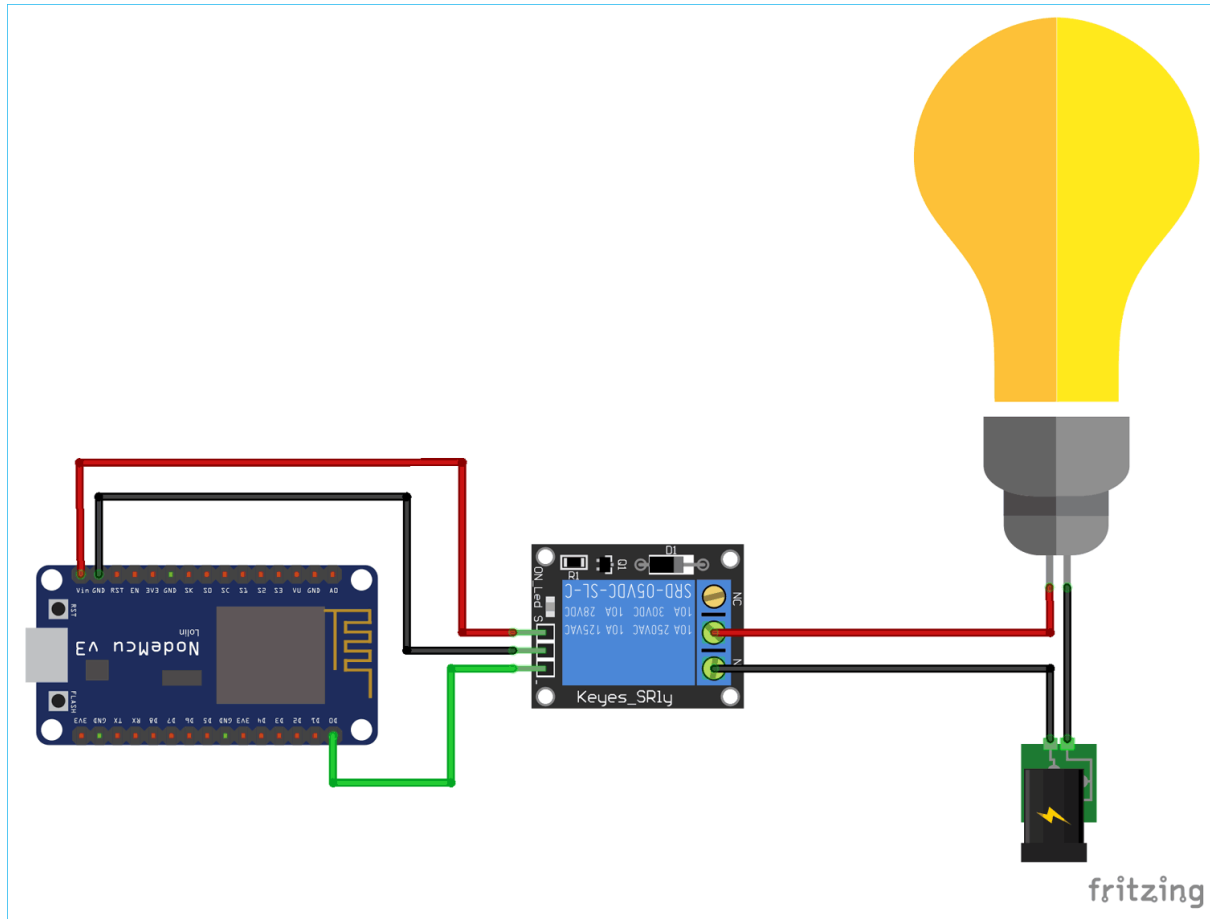
fritzing



## Relay Module:



## Project Circuit Diagram:

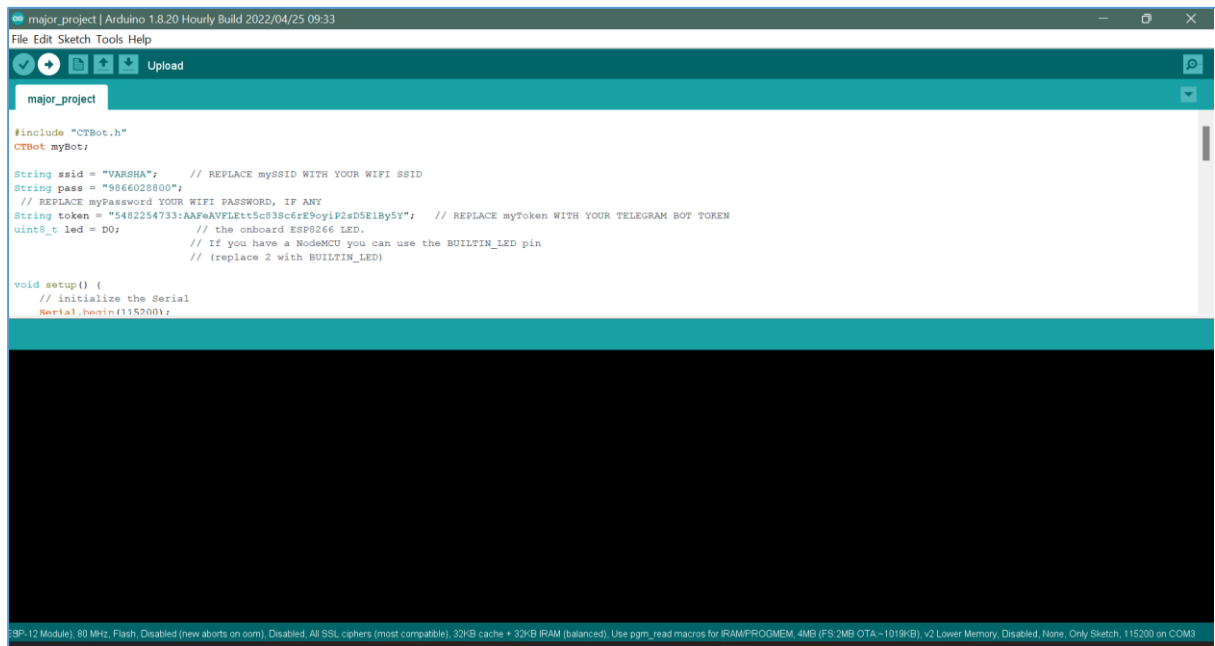


## Output:

### (a) Software:

#### (i)Arduino IDE:

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



The screenshot shows the Arduino IDE interface with the following code in the editor:

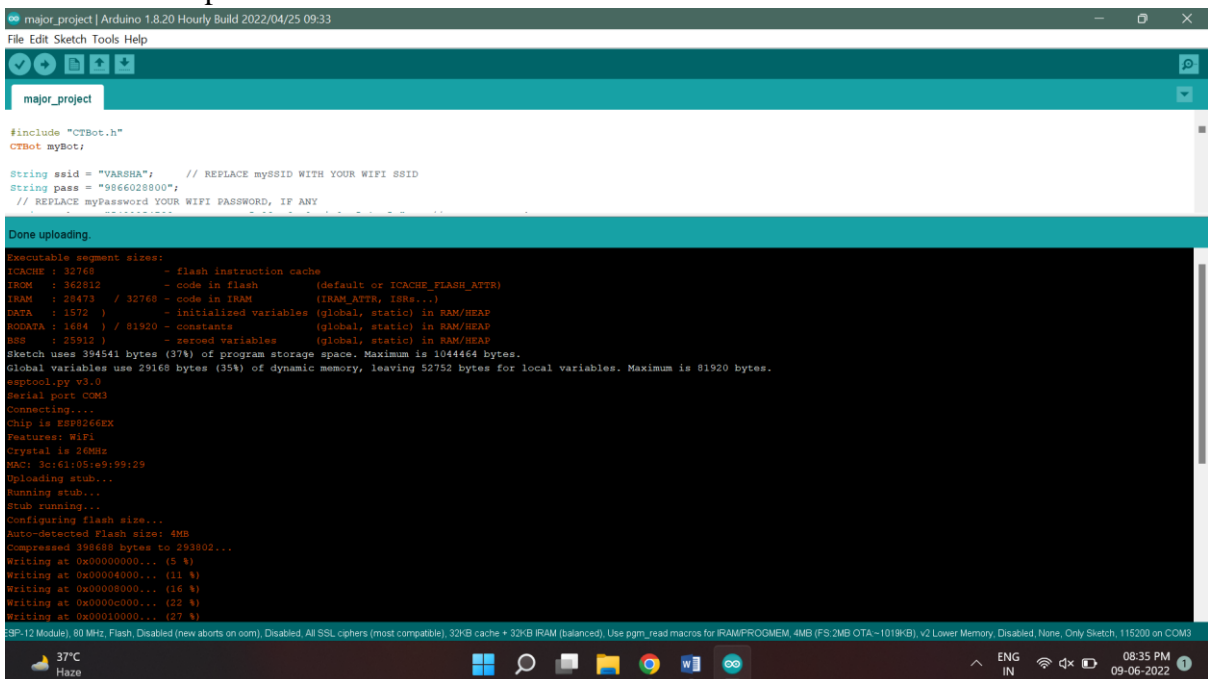
```
#include "CTBot.h"
CTBot myBot;

String ssid = "VARSHA"; // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
// REPLACE myPassword YOUR WIFI PASSWORD, IF ANY
String token = "5492254733:AAFeAVLEt5c93Sc6r89yI92aD5E1By5Y"; // REPLACE myToken WITH YOUR TELEGRAM BOT TOKEN
uint8_t led = D0; // the onboard ESP8266 LED. // REPLACE myToken WITH YOUR TELEGRAM BOT TOKEN
// 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);
}
```

The status bar at the bottom indicates: "ESP-12 Module, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB RAM (balanced), Use pgm\_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3".

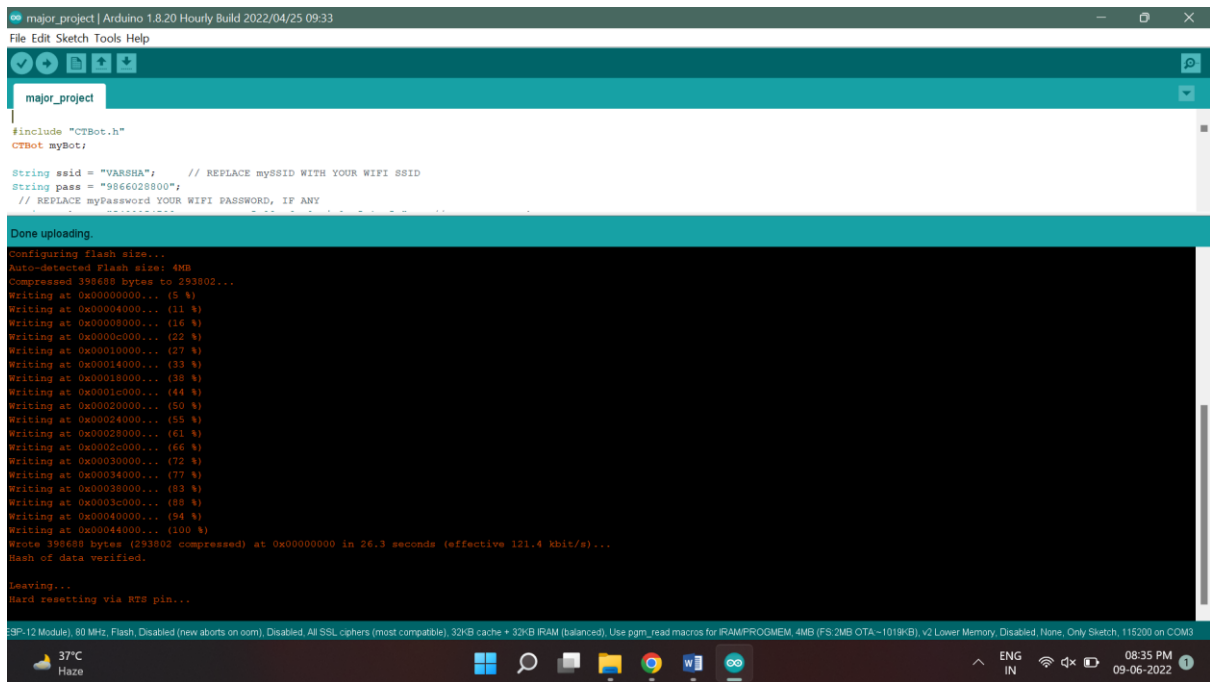
Here we have uploaded the code .



The screenshot shows the Arduino IDE interface with the upload progress and completion. The code in the editor is the same as in the previous screenshot. The status bar at the bottom indicates: "ESP-12 Module, 80 MHz, Flash, Disabled (new aborts on oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB RAM (balanced), Use pgm\_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3".

The upload progress shows:

```
Done uploading.
Executable segment sizes:
ICACHE : 32768 - flash instruction cache
IRAM : 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
BSSDATA : 1684 - 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:c6:105:e9:99:29
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected flash size: 4MB
Compressed 398688 bytes to 292802...
Writing at 0x00000000... (5 %)
Writing at 0x00004000... (11 %)
Writing at 0x00008000... (16 %)
Writing at 0x0000c000... (22 %)
Writing at 0x00010000... (27 %)
```



```
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;

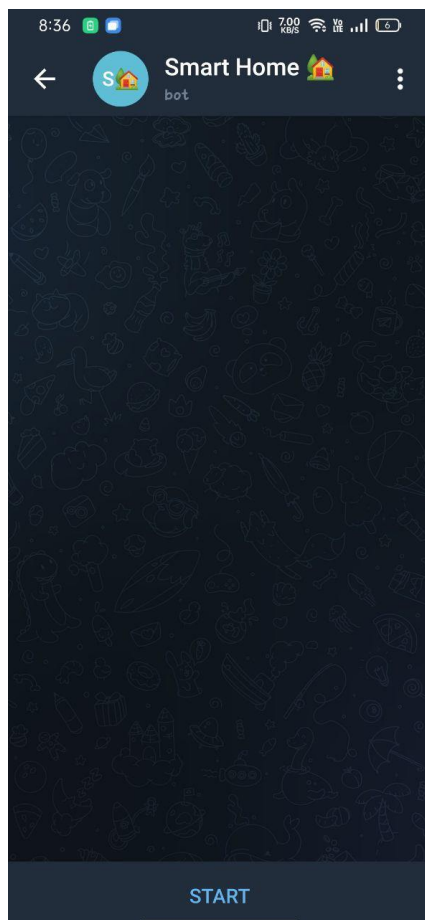
String ssid = "VARSHA"; // REPLACE mySSID WITH YOUR WIFI SSID
String pass = "9866028800";
// REPLACE myPassword YOUR WIFI PASSWORD, IF ANY

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 RAM (balanced), Use pgm_read macros for IRAM/FROMEM, 4MB (FS:2MB OTA~1019kB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3
37°C
Haze
ENG
IN
08:35 PM
09-06-2022
```

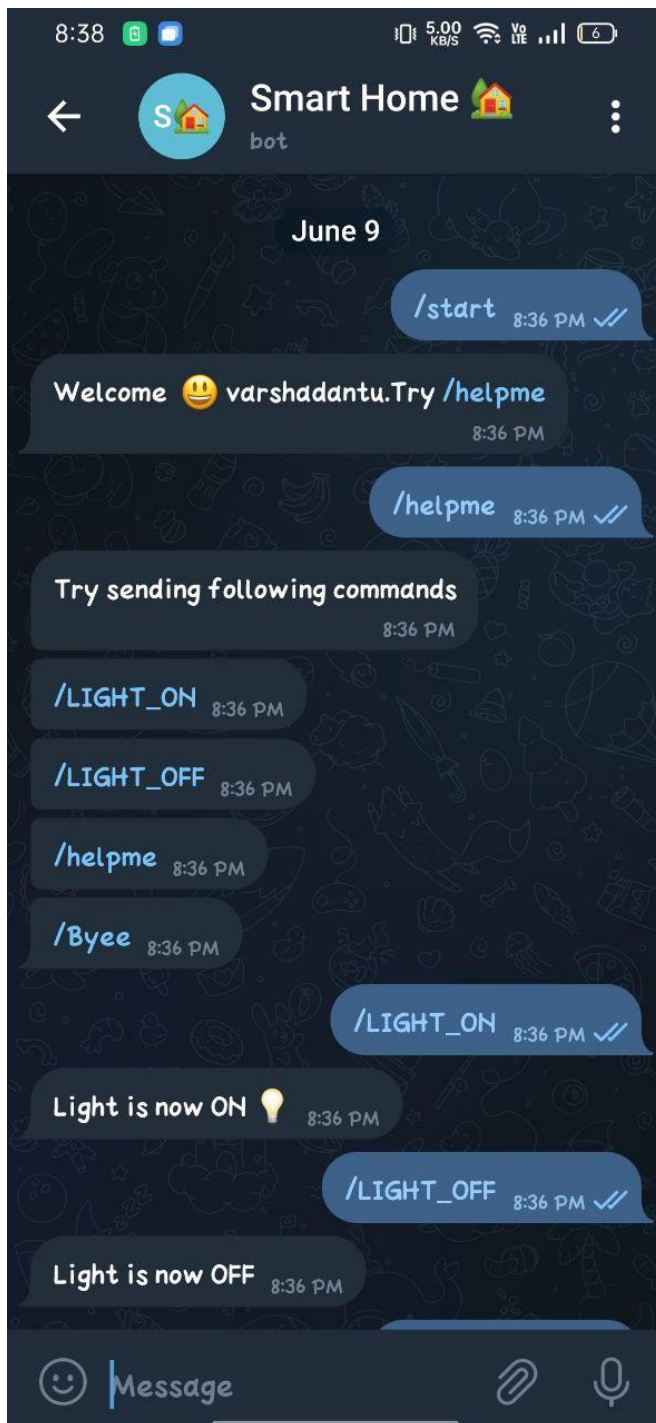
Now the code compiled successfully and done uploading into the NODEMCU.

## (ii) Telegram Bot:

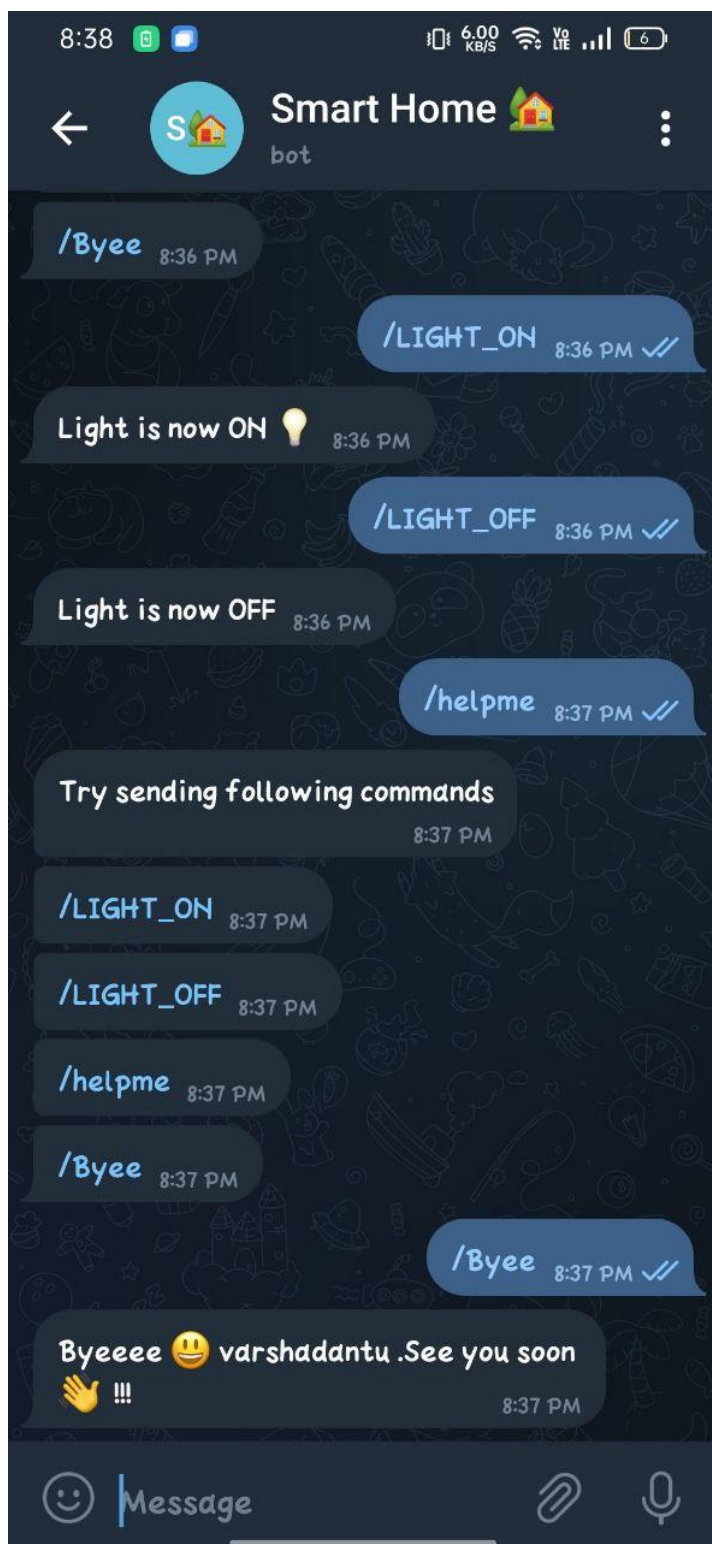


/start → /helpme → /LIGHT\_ON → Light is now ON.

/start → /helpme → /LIGHT\_OFF → Light is now OFF.



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



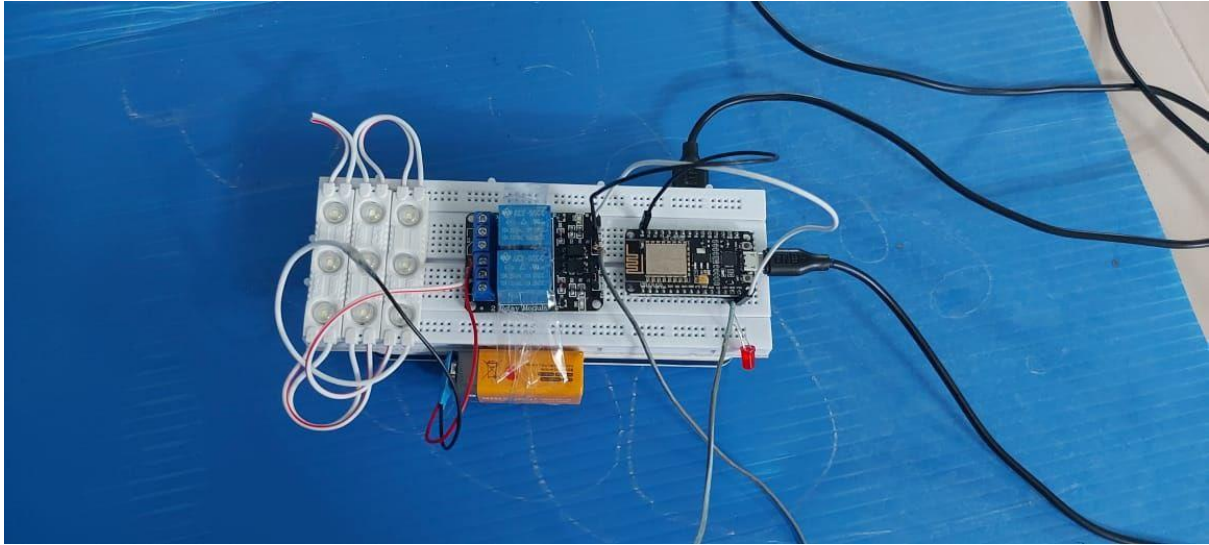
### **(iii) Hardware:**

Whole figure.





Light Off:



Light On:

