Home automation system using NODE MCU and Relay

TABLE OF CONTENTS

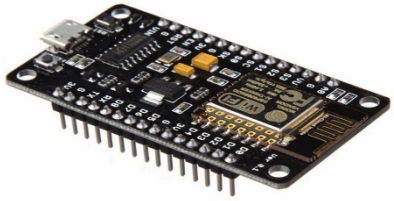
Components Required

Sl. No.	Name
1)	Node MCU
2)	2 Way Relay
3)	Lm7805 transistor
4)	9v Battery
5)	Battery cap
6)	Software Support: Adafruit.io, Google Assistant, IFTTT
7)	Breadboard
8)	Multimeter
9)	Vero Board

10)	Soldering Iron
11)	Soldering Flux
12)	Solder Wire
13)	Jumper Wire

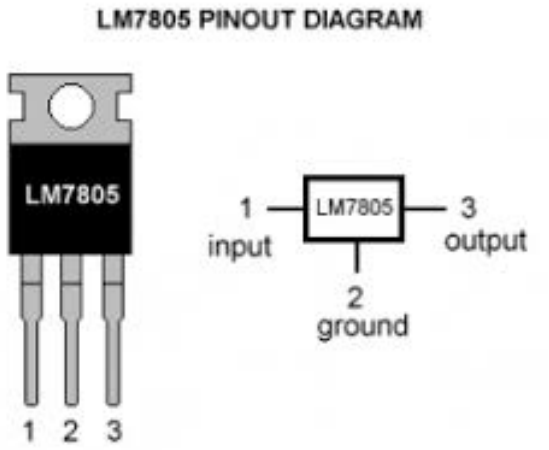
COMPONENT DESCRIPTION

1) **Node MCU:** NodeMCU is an open source IoT platform. It includes Firmware Which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and Hardware which is baesd on ESP-12 module. The term “NodeMCU” refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It 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.



2) **2WayRelay:** A Relay is an electrically operated switch. May relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal.

3) **LM7805 Transistor:** Voltage sources in a circuit may have fluctuations resulting in not providing fixed voltage outputs. A voltage regulator IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series pf fixed linear voltage regulators used to maintain such fluctuations, is a popular voltage regulator integrated circuit(IC). The xx in 78xx indicates the output voltage it provides. 7805 IC provides +5 voltage regulated power supply with provisions to add a heat sink.



4) Software Support:

a) **AdaFruit.io:** Adafruit IO is a system that makes data useful. IO includes client libraries that wrap our REST and MQTT APIs. IO is built on RUBY on Rails, and Node.js.

b) **Google Assistant:** The Google Assistant is a virtual personal assistant developed by google that is primarily available on smart phones. Unlike Google now, the Google Assistant can engage in two-way conversations.

c) **IFTTT (If This Then That):** The IFTTT is a free web-based service to create chains of simple conditional statements, called Applets.

INTRODUCTION

This project is implemented using NodeMCU, 2Way Relay Switch, LM7805 Transistor, a breadboard, a veroboard, couple of jumper wires, multimeter, and as for the software implementations we have used google assistant, Arduino ide for the coding, IFTTT applet launcher and Adafruit IO for the Wi-Fi connectivity.

Our goal with this project is very simple yet useful that we have implemented home automation. Basically we have controlled the home electronic appliances such as turning light on/off with voice recognition system.

From the google assistant we give voice recognition such as “turn on light/turn off light”. Then is responses “turning on/off light”. After that it connects with the IFTTT applet launcher which sends instructions through adafruit io to Node MCU wi-fi module (ESP8266) Which then connects to the relay and the relay works as a switch by turning the light on/off in terms of the instruction you give.

THEORY

At first we have created an applet using IFTTT (If This Then That) app which takes two string as an input i.e. “turn on/off light”. Then the google assistant connects with the IFTTT which gives these two actions to Adafruit IO. The adafruit io at first searches whether these actions are authentic or not. Then it sends these actions interms of 0’s and 1’s to the NodeMCU which is connected with the internet through a specified port. The ESP8266 inbuilt Wi-Fi module of the NodeMCU fetches these actions i.e. the 0’s and 1’s as in inverted that means 1 means

SOURCE CODE

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#define relay 16

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "your_SSID"
#define WLAN_PASS       "paswword"

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "PROJECT_NAME"
#define AIO_KEY         "AIO_KEy"

/************ Global State (you don't need to change this!) ******************/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiFlientSecure for SSL
//WiFiClientSecure client;
```

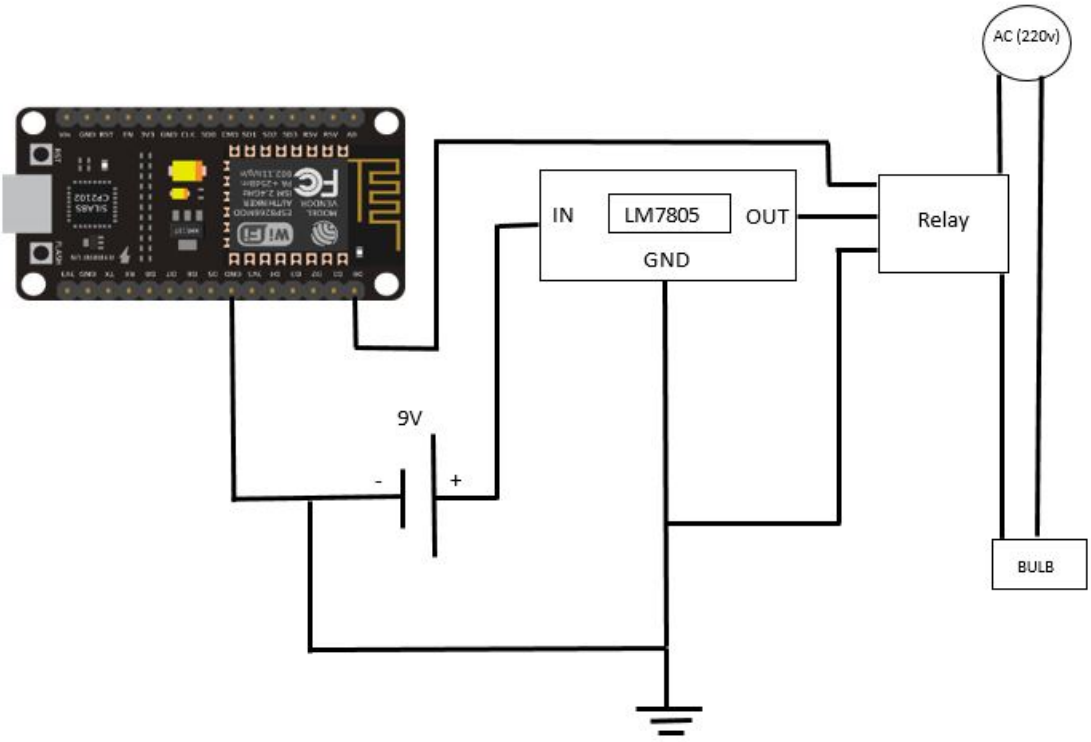
```
// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/************************** Feeds ***********************************/

Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/feeds/Light");

/************************* Sketch Code ***********************************/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration
// for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();

void setup() {
 Serial.begin(115200);
 delay(10);
 pinMode(relay,OUTPUT);

 Serial.println(F("Adafruit MQTT demo"));

 // Connect to WiFi access point.
 Serial.println(); Serial.println();
 Serial.print("Connecting to ");
 Serial.println(WLAN_SSID);

 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
```

```
  Serial.println();

  Serial.println("WiFi connected");
  Serial.println("IP address: "); Serial.println(WiFi.localIP());

  // Setup MQTT subscription for onoff feed.
  mqtt.subscribe(&onoffbutton);
}

uint32_t x=0;

void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).  See the MQTT_connect
  // function definition further below.
  MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop


  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &onoffbutton) {
      Serial.print(F("Got: "));
      Serial.println((char *)onoffbutton.lastread);
      uint16_t state = atoi((char *)onoffbutton.lastread);
      digitalWrite(relay,state);

    }
  }
```

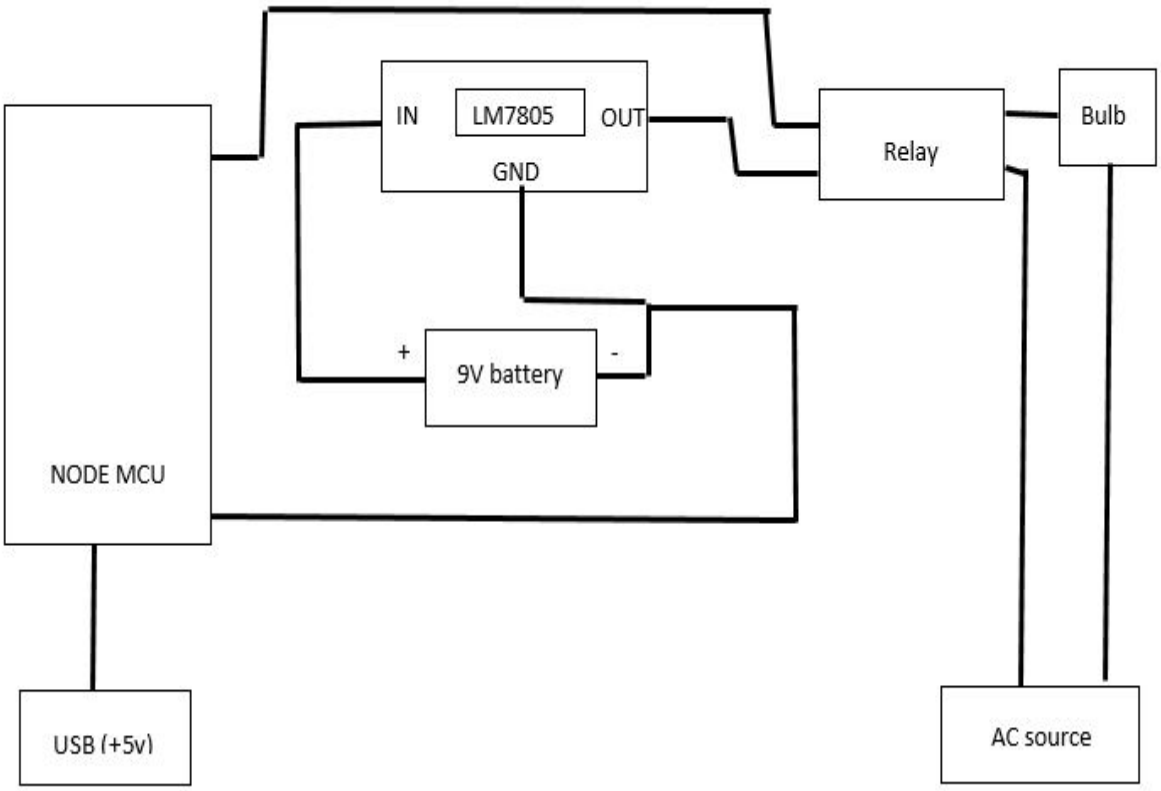
```
 // ping the server to keep the mqtt connection alive
 // NOT required if you are publishing once every KEEPALIVE seconds
 /*
 if(! mqtt.ping()) {
   mqtt.disconnect();
 }
 */
}

// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
 int8_t ret;

 // Stop if already connected.
 if (mqtt.connected()) {
   return;
 }

 Serial.print("Connecting to MQTT... ");

 uint8_t retries = 3;
 while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
      Serial.println(mqtt.connectErrorString(ret));
      Serial.println("Retrying MQTT connection in 5 seconds...");
      mqtt.disconnect();
      delay(5000);  // wait 5 seconds
      retries--;
      if (retries == 0) {
```

```
      // basically die and wait for WDT to reset me
      while (1);
    }
  }
  Serial.println("MQTT Connected!");
}
```

CIRCUIT DIAGRAM



BLOCK DIAGRAM



REMARKS

1) The NodeMCU should be connected with the usb only when all the connection have been made.

2) The ground for the NodeMCU and 2WayRelay should be common ground.

3) Uttermost precautions must be taken while making the connections.

CITATIONS

1) www.electronicshub.com
2) www.wikipedia.com