Practical 11 (additional)

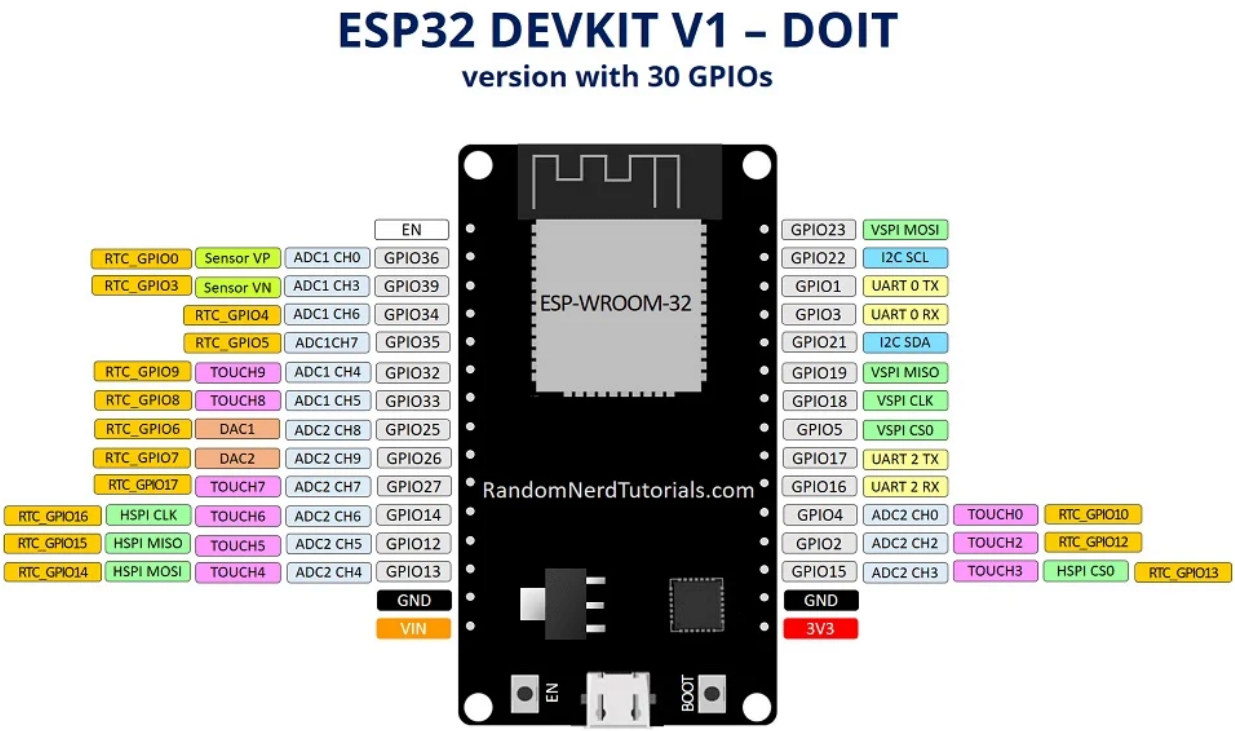
Set up the NodeMCU ESP32 embedded system.

Objective

Understand how to use NodeMCU ESP32 for setting up a simple IOT solution with LEDs, sensors, WiFi connection, MQTT communication and Google Firebase connection.

Background

NodeMCU ESP32 is an ESP-WROOM-32 module in breadboard friendly form factor, wecan develop our project by using this compact microcontroller on a breadboard. ESP32 is the big brother of ESP8266. It comes with a dual-core 32-bit processor, built-in WiFi and Bluetooth, more RAM and Flash memory, more GPIO, more ADC, and many other peripherals.



Features:

- NodeMCU based on ESP-WROOM-32 module
- Based on ESP32 DEVKIT DOIT
- 30 GPIO Version
- ESP32 is a dual-core 32-bit processor with built-in 2.4 GHz Wi-Fi and Bluetooth
- 4MByte flash memory
- 520KByte RAM
- 2.2 to 3.6V Operating voltage range
- In breadboard-friendly breakout

- USB micro B for power and Serial communication, use to load the program and serial debugging too

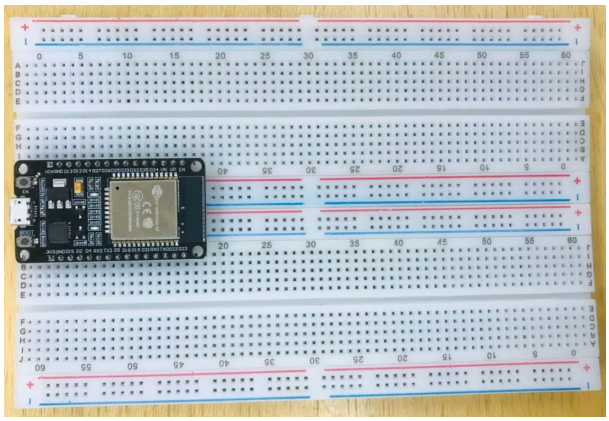
Document/Tutorials:

- Github, NodeMCU ESP32 Dev firmware
- Getting Started with NodeMCU 32 board
- Wiki page
- ESP-WROOM-32 Datasheet
- CP210x USB driver

Procedure

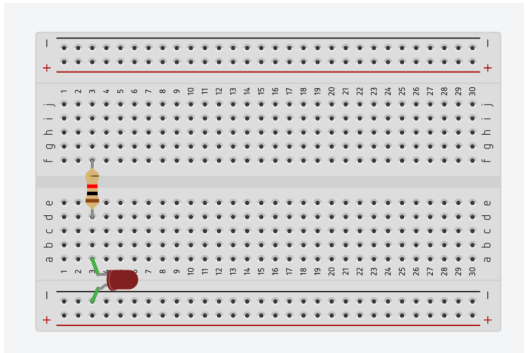
Step 1: Set up the NodeMCU ESP32 on two pieces of breadboard

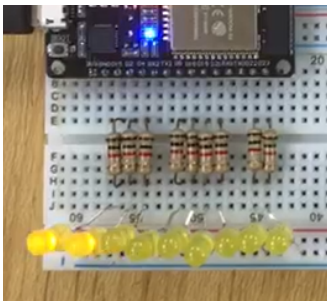
* Take note that the width size of NodeMCU ESP32 pins are not able to fit into one breadboard.

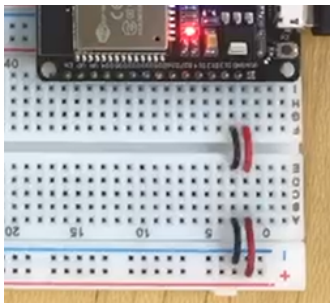


Step 2: Connect LEDs with resistors on NodeMCU ESP32 pins

- Use 9 LEDs and 1kΩ registers and connect them on Pin D15, D2, D4, D5, D18, D19, D21, D22 and D23. All -ve pins of LEDs connected to the common GND.
- You may use the USB powered Vin and GND to connect to the - + of the breadboard.







Step 3: Set up the Arduino IDE with NodeMCU ESP32 drivers

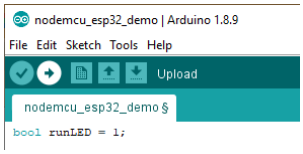
- Install Arduino IDE (download from here: https://www.arduino.cc/en/Main/Software_)
- Follow the steps in : https://www.aranacorp.com/en/programming-an-esp32-nodemcu-with-the-arduino-ide/ to set up the NodeMCU ESP32 driver (to ensure the communication port is connected), and necessary library (add json handler and include the board manager package) in Arduino IDE.

Step 4: Type Codes for initialization and running light functions.

```
bool runLED = 1;
void setup() {
  initialize();
}

void loop() {
  if (runLED){
    runningLED();
  }
}

void initialize(){
  pinMode(15, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(18, OUTPUT);
  pinMode(19, OUTPUT);
  pinMode(21, OUTPUT);
  pinMode(22, OUTPUT);
  pinMode(23, OUTPUT);
}
```

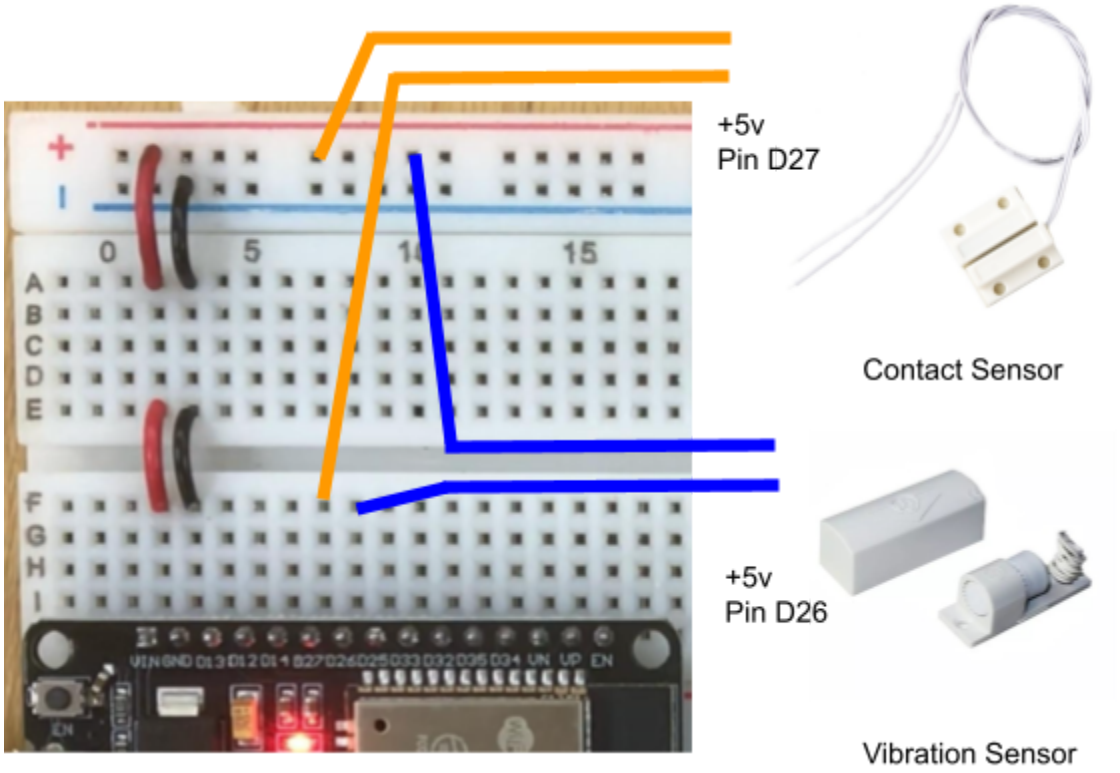
```
void runningLED() {
  toggle(2, 15);
  toggle(4, 2);
  toggle(5, 4);
  toggle(18, 5);
  toggle(19, 18);
  toggle(21, 19);
  toggle(22, 21);
  toggle(23, 22);
  toggle(22, 23);
  toggle(21, 22);
  toggle(19, 21);
  toggle(18, 19);
  toggle(5, 18);
  toggle(4, 5);
  toggle(2, 4);
  toggle(15, 2);
}

void toggle(byte on_pin, byte off_pin) {
  digitalWrite(on_pin, 1);
  digitalWrite(off_pin, 0);
  delay(25);
}
```

Press the second button (Right Arrow) to upload the code to NodeMCU ESP32



Step 5: (Optional) Connect contact and vibration sensors on NodeMCU ESP32 pins

- Connect the contact sensor to +5v and pin D27. Contact sensor has to always close contact with the magnetic bar, to ensure the input of D27 is high (door closed).
- Connect the vibration sensor to +5v and pin D26.



- In this example of work, we added three additional LEDs at D25, D33 and D32 for indicating WiFi connectivity status, MQTT connectivity status, and command executed status accordingly.

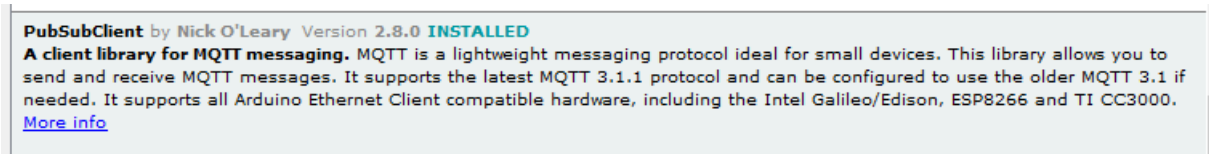
* Please follow Step 2 for setting up the three LEDs.

Step 6: (Optional) Type Codes for initialization and MQTT functions.

WiFi.h library: https://www.arduino.cc/en/Reference/WiFi

PubSubClient.h library:

In Arduino IDE, from top menu bar, select Sketch > include library > Manage Libraries > Key in topic : PubSubClient and look for this (ONLY):




```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "D203"; //YOUR WIFI SSID
const char* password = "blockd203"; //YOUR WIFI PASSWORD
const char* mqtt_server = "broker.hivemq.com"; // MQTT BROKER
const char* mqtt_topic = "TARUC/BAIT2123/DEMO"; // UNIQUE TOPIC

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
int ledState = 0;
bool runLED = 1;

void setup() {
  initialize();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  reconnect();
  client.subscribe(mqtt_topic);
}

void loop() {
  if (runLED){
    runningLED();
  }
  if (digitalRead(27)== 0){
```

```
    client.publish(mqtt_topic,"Contact Sensor = 0");
    delay(1000);
  }
  if (digitalRead(26)== 0){
    client.publish(mqtt_topic,"Vibration Sensor = 0");
    delay(1000);
  }
  client.loop();
}

void initialize(){
  pinMode(27, INPUT); // Contact Sensor
  pinMode(26, INPUT); // Vibration Sensor

  pinMode(25, OUTPUT); // WIFI indicator
  pinMode(33, OUTPUT); // MQTT indicator
  pinMode(32, OUTPUT); // Command executed
  pinMode(15, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(18, OUTPUT);
  pinMode(19, OUTPUT);
  pinMode(21, OUTPUT);
  pinMode(22, OUTPUT);
  pinMode(23, OUTPUT);
}

void setup_wifi(){
  delay(10);
  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED){
    ledState = !ledState;
    digitalWrite(25, ledState);
    delay(500);
  }
  digitalWrite(25, 1);
}
```

```
void reconnect(){
  while(!client.connected()){
    if (client.connect("", NULL, NULL)){
      ledState = !ledState;
      digitalWrite(33, ledState);
      delay(500);
    }
  }
  digitalWrite(33, 1);
}

void callback(char* topic, byte* message, unsigned int length){
  digitalWrite(32, !digitalRead(32));
  String messageTemp = "";
  for (int i = 0; i < length; i++){
    messageTemp += (char)message[i];
  }
  if(messageTemp == "run=0"){
    runLED = 0;
  }
  if(messageTemp == "run=1"){
    runLED = 1;
  }
}
```

* You may need to disconnect the input pins for Vibration and Contact sensors in order to upload the code to NodeMCU ESP32. Then reconnect these input pins after the loading process is completed.

* Turn on another MQTT publisher (you can use Chrome MQTTBox), under the same topic, to publish a message: run=0 for stopping the running LEDs, run=1 for starting the running LEDs.

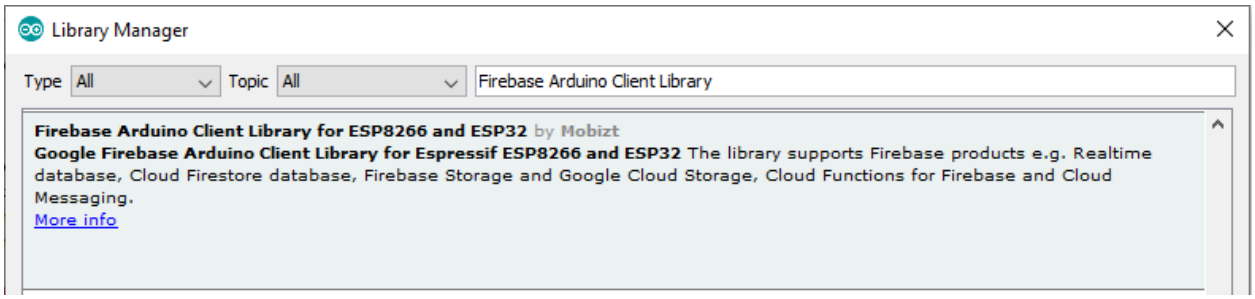
* Turn on another MQTT subscriber (you can use Chrome MQTTBox), to subscribe to the same topic on the same broker. Test the contact sensor (by removing the magnetic bar) and vibration sensor (by knocking the sensor) to see the MQTT report published by NodeMCU ESP32.

Step 7: (Optional) Type Codes for initialization and Google Firebase functions.

- For Google Firebase to work with NodeMCU ESP32, we can refer to this online reference: https://randomnerdtutorials.com/esp32-firebase-realtime-database/

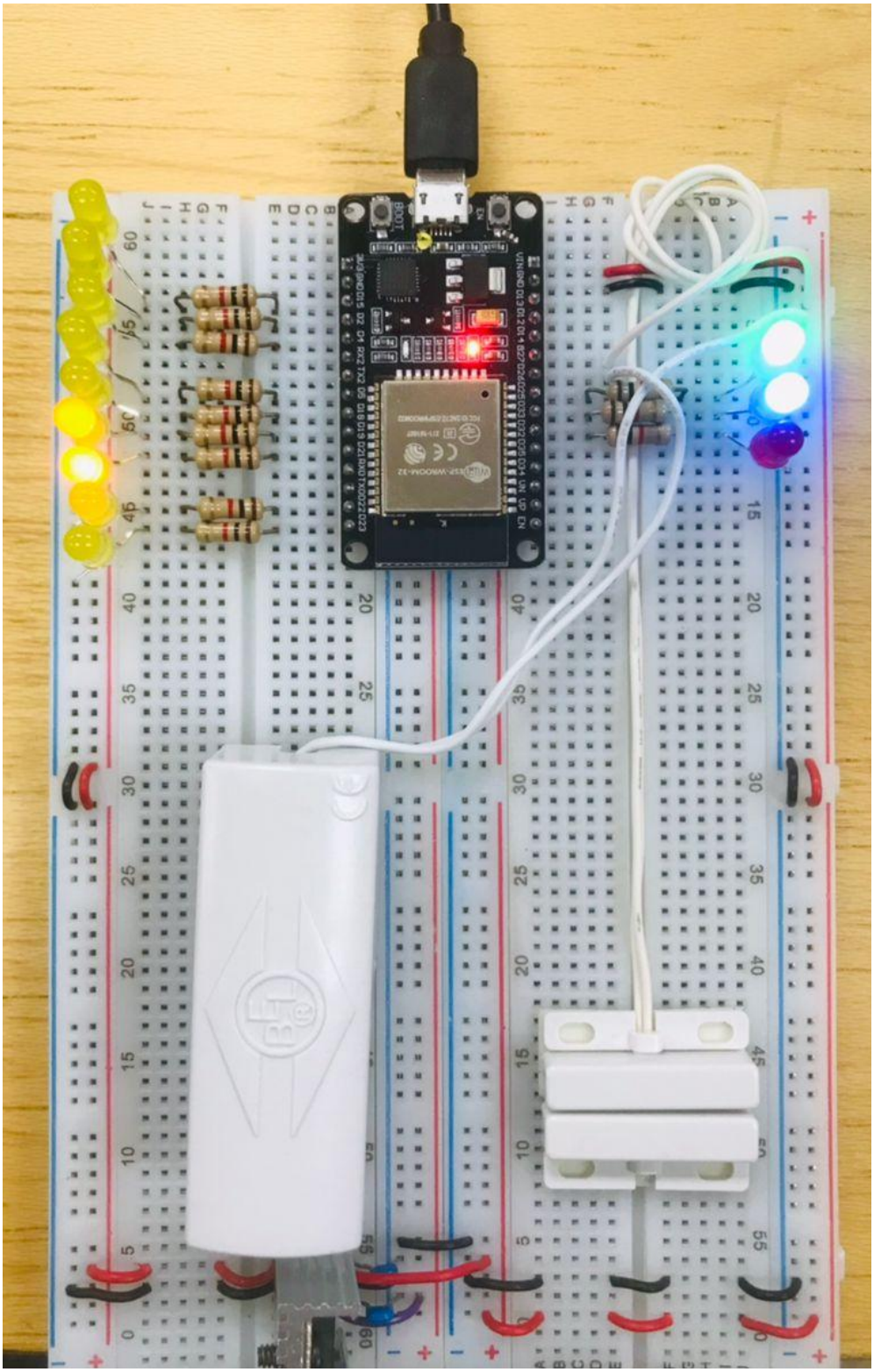
In Arduino IDE, follow the next steps to install the library.

1. Go to Sketch > Include Library > Manage Libraries
2. Search for Firebase ESP Client and install the Firebase Arduino Client Library for ESP8266 and ESP32 by Mobitz.



* Self exploration is required.

Sample of constructed circuits on breadboard for running LEDs controlled by MQTT commands, contact and vibration sensor reporting via MQTT.



Additional:

We have added in a small circuit (7805 voltage regulator) for external power input, e.g., +12VDC adapter.

* Remember to disconnect the Vin from the NodeMCU ESP32 board to the breadboard. The GND from NodeMCU ESP32 should remain connected to the breadboard as a common GND.