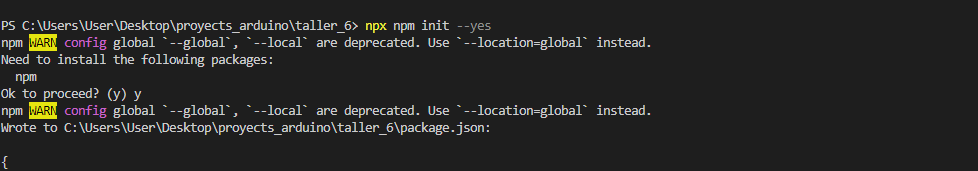
Taller 6. Servidor Local-IoT

Diego Iván Perea Montealegre (2185751) [diego.perea@uao.edu.co](mailto:diego.perea@uao.edu.co)

Facultad de Ingeniería, Universidad Autónoma de Occidente

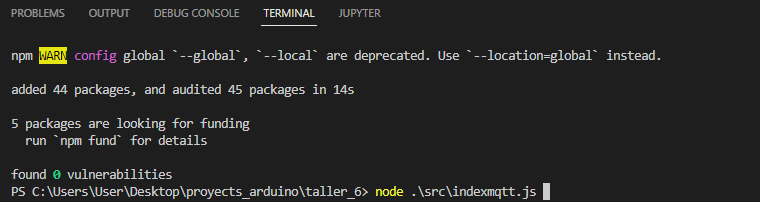
Cali, Valle del Cauca

Este comando lo que hace es crear un json (package.json) con toda la descripción de proyecto y además muestra los paquetes instalados

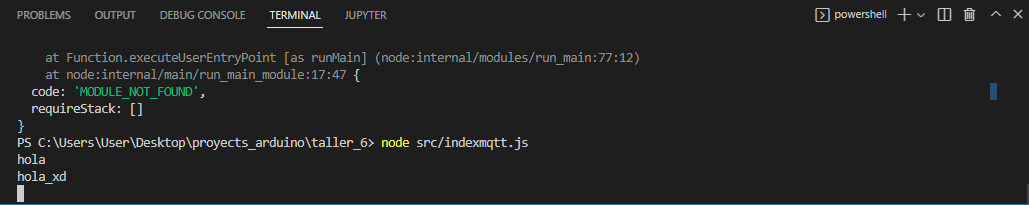


Se instala la librería de MQTT que nos permitirá conectarnos a un bróker MQTT y realizar publicaciones y subscripciones. Para esto usamos el comando

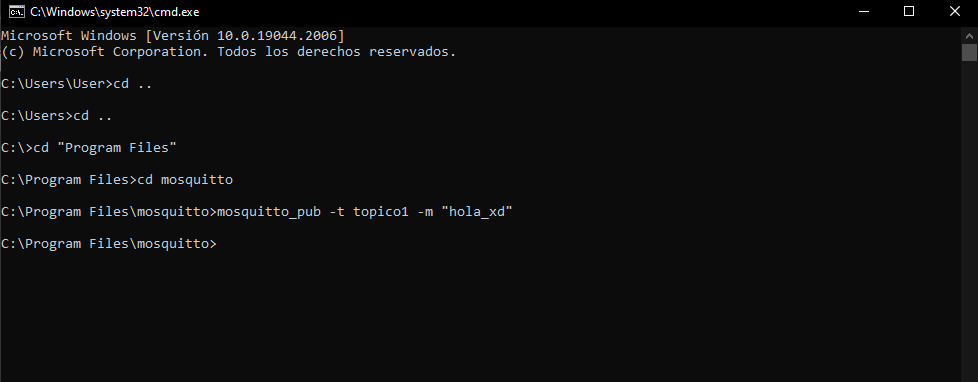
A continuación se realiza la ejecución de node.js con la forma mqqt de mosquitto.



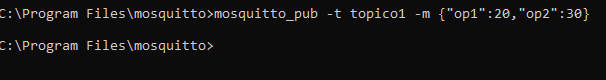
El nodo se queda esperando a que sea publicado en el tópico especificado



Se realiza la publicacion al topico 1 con un mensaje

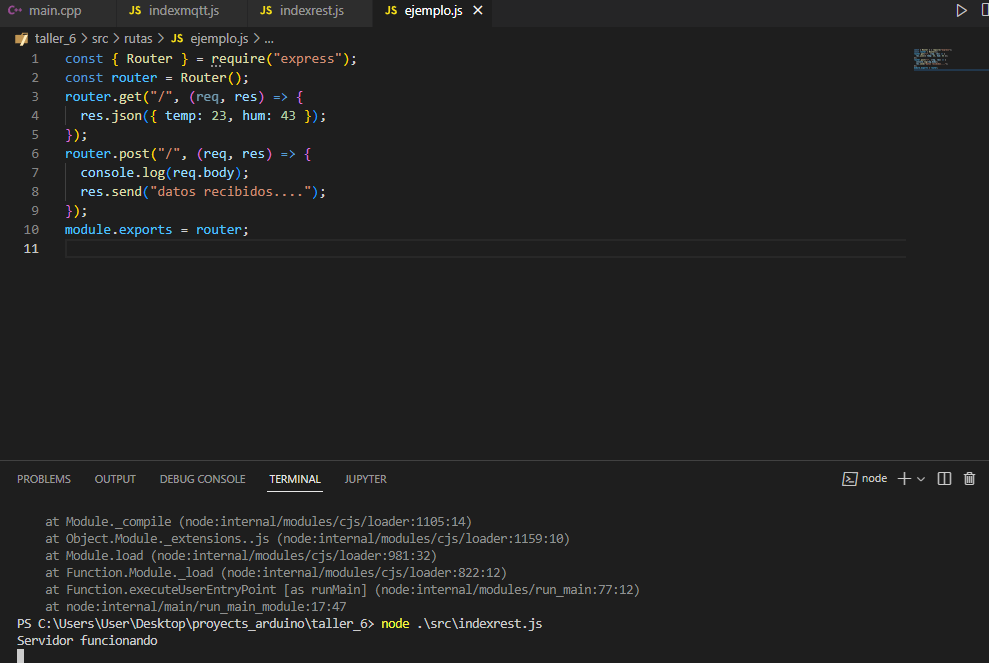


Publicación en el tópico en formato json

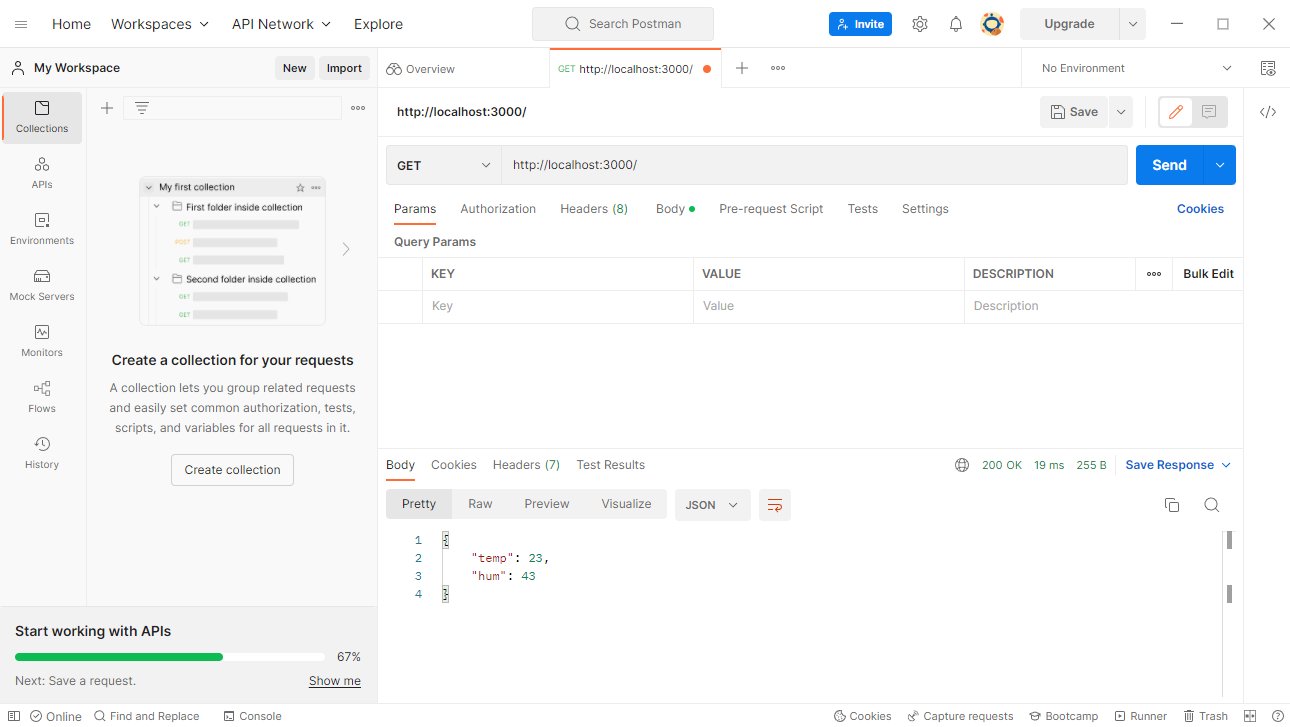


Se instala los paquetes express y morgan, Express facilita a creación del servicio rest desde node y morgan permite ver las peticiones y respuestas del servidor en la consola lo cual facilita el proceso de depuración, ya que nos permite ver los errores.

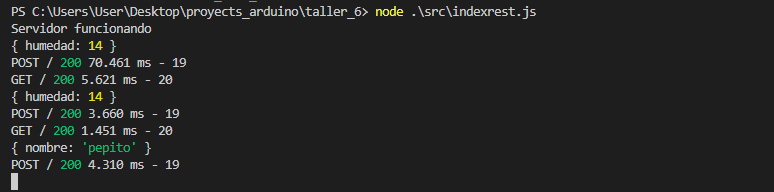




Postman en uso de GET para obtener la información enviada que se encuentra en ejemplo.js

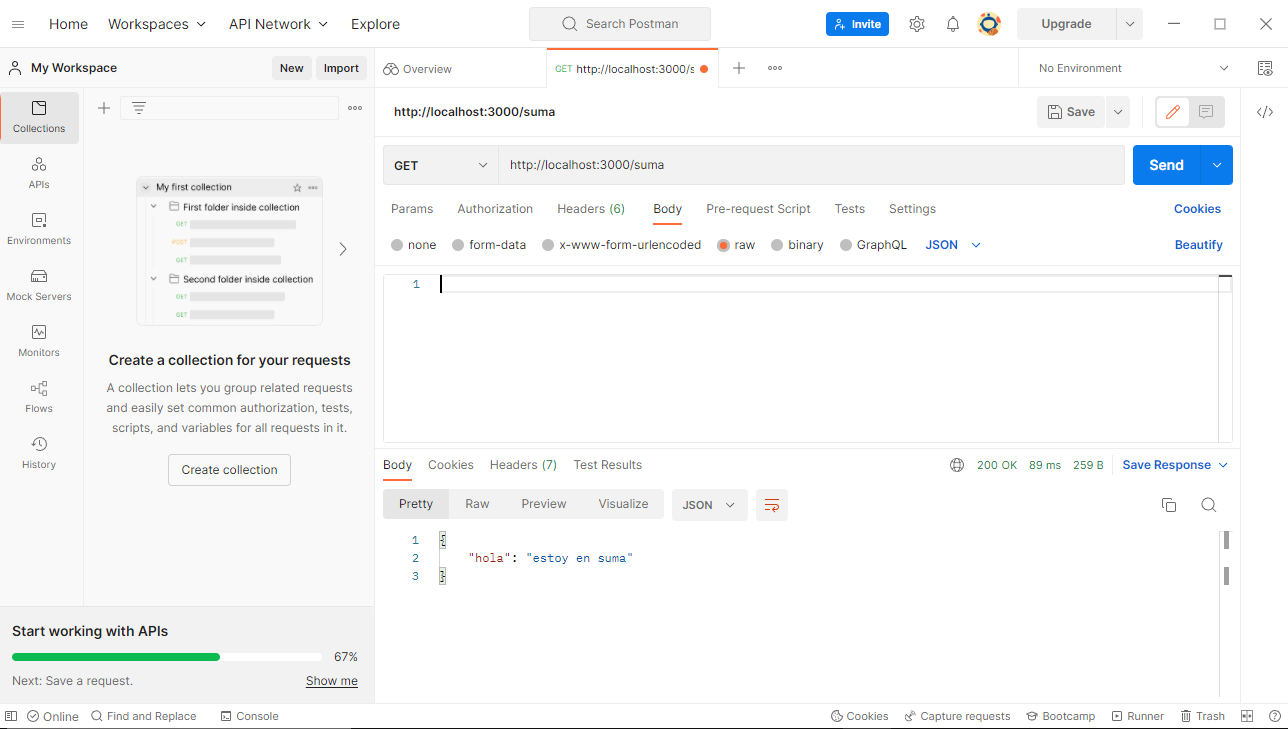


Visualizacion en la terminal en donde se realizo GET con postman en la extracción de datos.



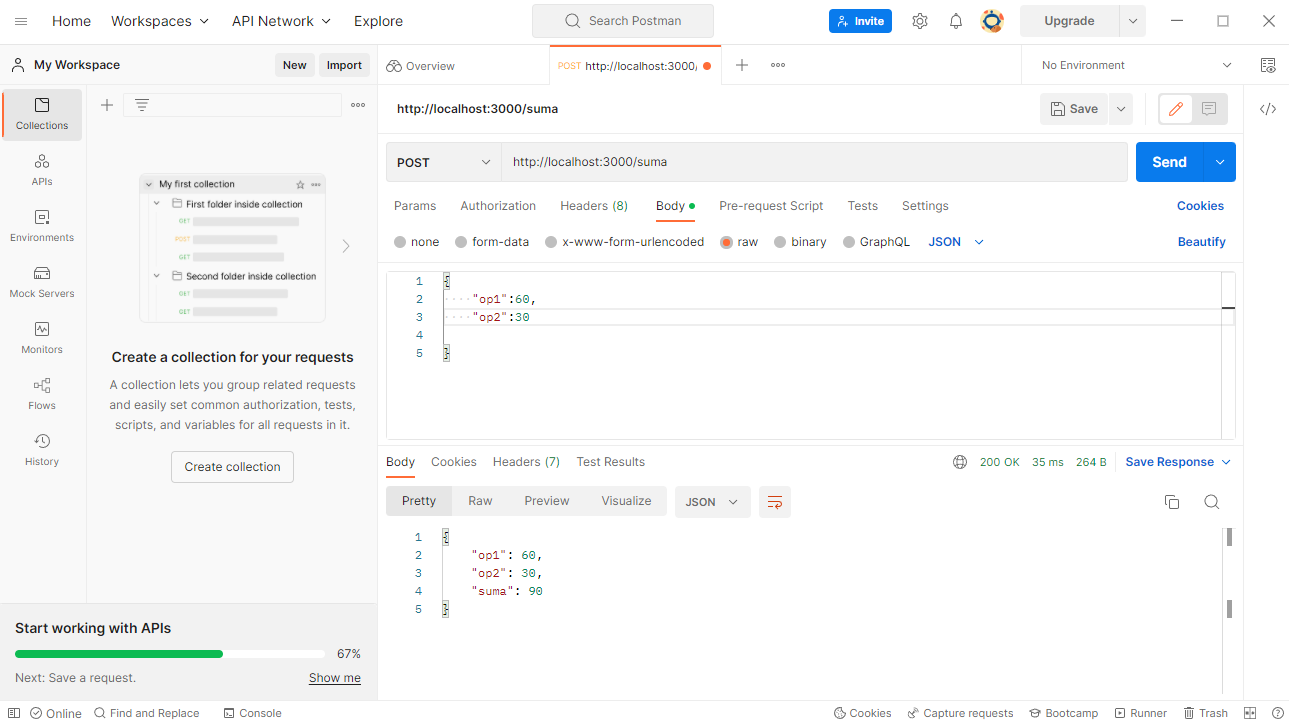
Se realiza la extracción de método GET en el dato que está en el archivo suma.js

Suma.js en GET

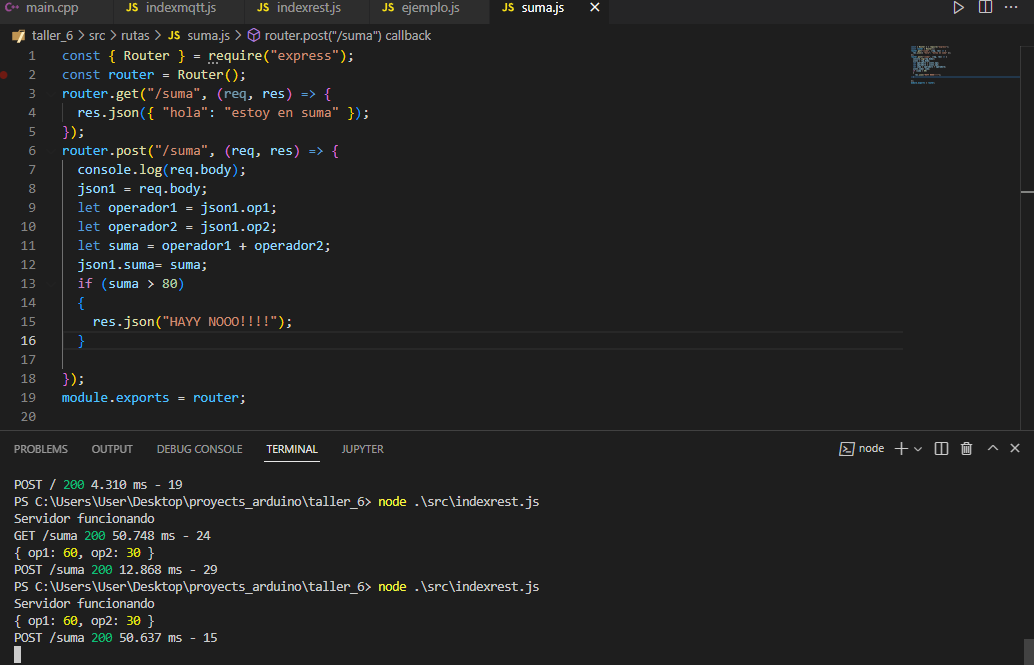


Se realiza la extracción de método POST en el dato que esta en el archivo suma.js

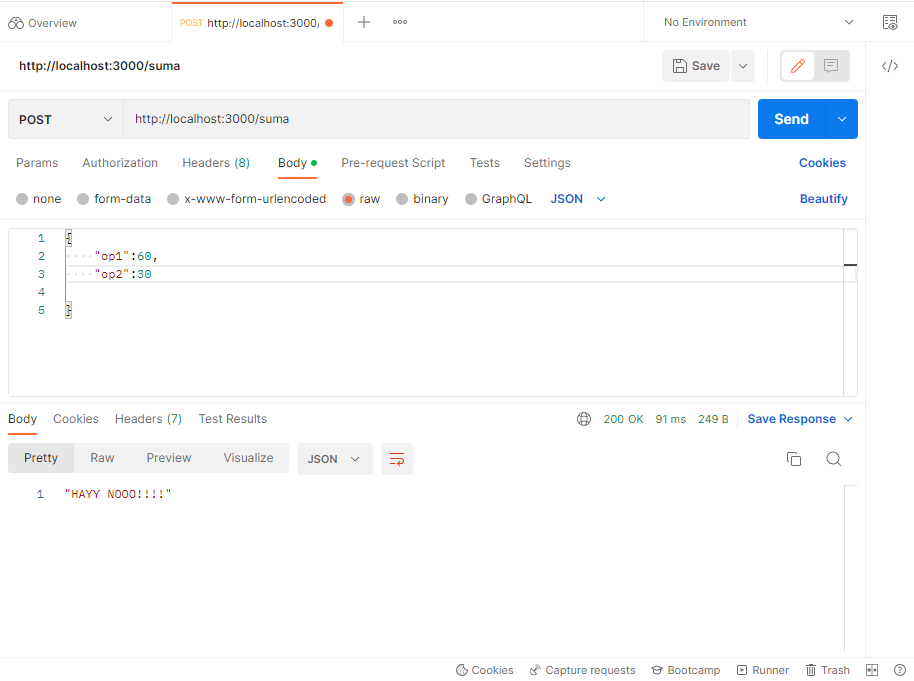
Suma.js en POST



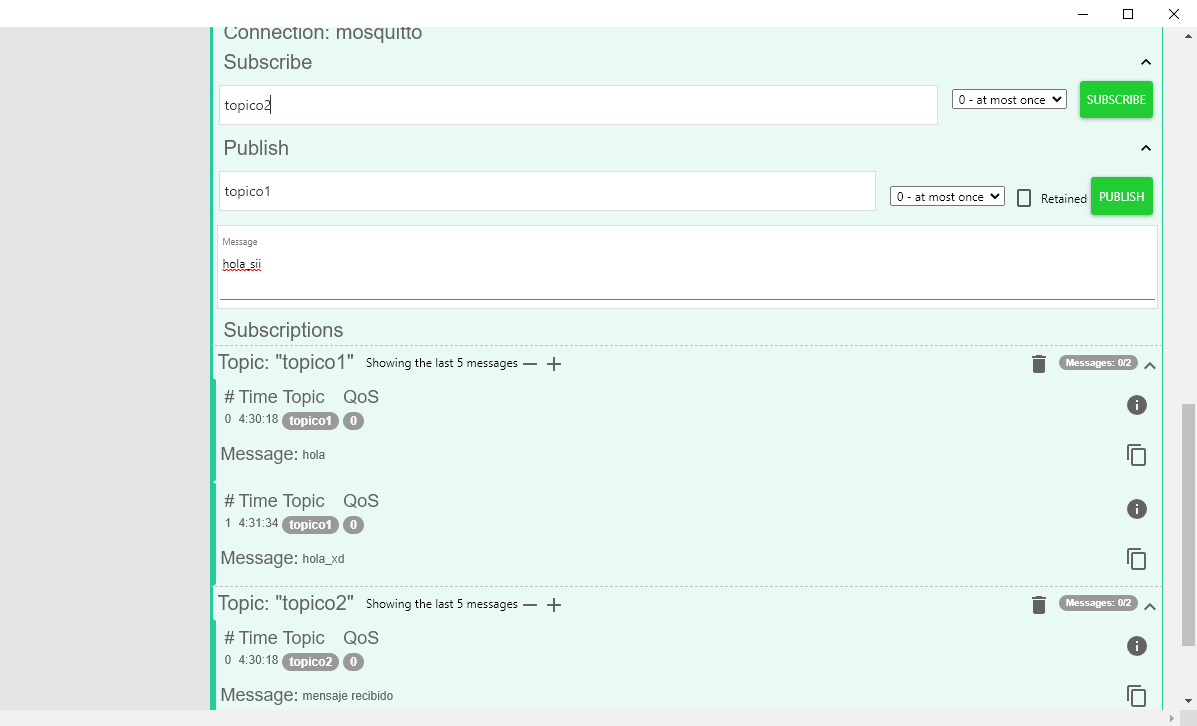
EJEMPLO DE REALIZACION CON IF



POST DE EJEMPLO IF



Suscripción de topico2 en donde va a pública en el topico1 utilizando mqttlens , en donde la url del servidor en confirguacion del mqttlens es “localhost”



Ahora con esp32

MOSQUITO ESP32

El código usado para mosquito es:

#include <Arduino.h>

#include <ArduinoJson.h>

#include <WiFi.h>

#include <PubSubClient.h>

//LIBRERIAS PARA DHT11 (TEMPERATURA Y HUMEDAD)

#include <Adafruit\_Sensor.h>

#include <DHT.h>

//LIBRERIAS PARA FECHA Y HORA

#include <WiFi.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

//DEFINICION DE PINES DHT11

#define DHTPIN 4   // 4 = PIN D4

#define DHTTYPE    DHT11

DHT dht(DHTPIN, DHTTYPE);

//potenciometro ph

const int portPin=34;

int valorPh=0;

// Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP);

// Variables to save date and time

String formattedDate;

String dayStamp;

String timeStamp;

#define mqttUser ""

#define mqttPass ""

#define mqttPort 1883

const char\* ssid = "\*\*\*\*\*\*\*";//name wifi

const char\* password = "\*\*\*\*\*\*\*"; // clave de wifi

char mqttBroker[] = "192.168.\*.\*"; //ip del servidor

char mqttClientId[] = "topico1"; //cualquier nombre

char inTopic[] = "topico1";//topcico a suscribirse

void callback(char\* topic, byte\* payload, unsigned int length) {

  Serial.print("Message arrived [");

  Serial.print(topic);

  Serial.print("] ");

  for (int i=0;i<length;i++) {

  Serial.print((char)payload[i]);

}

  Serial.println();

}

WiFiClient BClient;

PubSubClient client(BClient);

void reconnect() {

// Loop until we're reconnected

  while (!client.connected()) {

  Serial.print("Attempting MQTT connection...");

  // Attempt to connect

  if (client.connect("", mqttUser, mqttPass)) {

  Serial.println("connected");

  // Once connected, publish an announcement...

 // Once connected, publish an announcement...

  float h= dht.readHumidity();

  float t =dht.readTemperature();

  //potenciometro ph

  valorPh=analogRead(portPin)/292.5;

  //----------------------

  String variable;

  StaticJsonDocument<256> doc;

  doc["nodo"] = 2;

  doc["temperatura"] = t;

  doc["humedad"] = h;

  doc["ph"]=valorPh;

  doc["fecha"] = dayStamp;

  doc["hora"] = timeStamp;

  serializeJson(doc, variable);

  int lon = variable.length()+1;

  Serial.println(variable);

  char datojson[lon];

  variable.toCharArray(datojson, lon);

  client.publish(inTopic,datojson);

  client.disconnect();

  delay(5000);

  // ... and resubscribe

  //client.subscribe("topic2");

  } else {

  Serial.print("failed, rc=");

  Serial.print(client.state());

  Serial.println(" try again in 5 seconds");

  // Wait 5 seconds before retrying

  delay(5000);

}

}

}

void setup\_wifi() {

  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

  delay(500);

  Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  // Initialize a NTPClient to get time

  timeClient.begin();

  // Set offset time in seconds to adjust for your timezone, for example:

  // COLOMBIA -5 , entonces -5\*3600 ->  -18000

  timeClient.setTimeOffset(-18000); //Thailand +7 = 25200

}

void setup()

{

  Serial.begin(9600); //Serial connection

  setup\_wifi(); //WiFi connection

  client.setServer(mqttBroker, mqttPort );

  client.setCallback( callback );

  Serial.println("Setup done");

  delay(1500);

}

void loop(){

    while(!timeClient.update()) {

    timeClient.forceUpdate();

    }

    // The formattedDate comes with the following format:

    // 2018-05-28T16:00:13Z

    // We need to extract date and time

    formattedDate = timeClient.getFormattedDate();

    // Extract date

    int splitT = formattedDate.indexOf("T");

    dayStamp = formattedDate.substring(0, splitT);

    //Serial.print("DATE: ");

    //Serial.println(dayStamp);

    // Extract time

    timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);

    if (!client.connected()) {

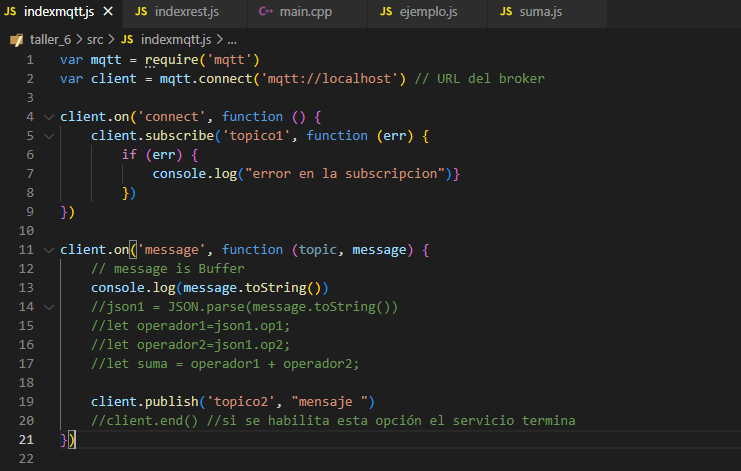
    reconnect();

    }

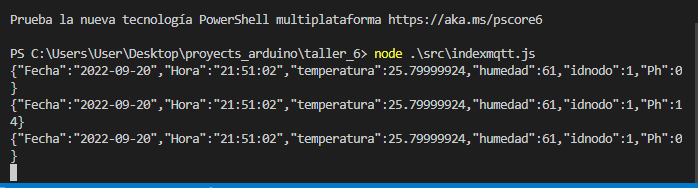
    client.loop();

}

Codigo correspondiente al archivo index\_mqtt.js



Se ejecuta el archivo indexmqtt.js poder leer lo que el esp32 mande los datos json , se visualiza en la terminal de la ubicacion del archivo.



Ahora con Rest EN ESP32:

Código:

#include <Arduino.h>

#include <ArduinoJson.h>

#include <HTTPClient.h>

//LIBRERIAS PARA DHT11 (TEMPERATURA Y HUMEDAD)

#include <Adafruit\_Sensor.h>

#include <DHT.h>

//LIBRERIAS PARA FECHA Y HORA

#include <WiFi.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

//DEFINICION DE PINES DHT11

#define DHTPIN 14  // 4 = PIN D4

#define DHTTYPE    DHT11

DHT dht(DHTPIN, DHTTYPE);

//potenciometro ph

const int portPin=34;

int valorPh=0;

// Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP);

// Variables to save date and time

String formattedDate;

String dayStamp;

String timeStamp;

const char\* ssid = "\*\*\*\*\*";//name wifi

const char\* password = "\*\*\*\*\*\*\*"; // clave de wifi

void setup\_wifi() {

  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

  delay(500);

  Serial.print(".");

}

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  // Initialize a NTPClient to get time

  timeClient.begin();

  // Set offset time in seconds to adjust for your timezone, for example:

  // COLOMBIA -5 , entonces -5\*3600 ->  -18000

  timeClient.setTimeOffset(-18000); //Thailand +7 = 25200

}

void setup() {

  Serial.begin(9600); //Serial connection

  setup\_wifi(); //WiFi connection

  delay(1500);

}

void loop() {

  // The formattedDate comes with the following format:

    // 2018-05-28T16:00:13Z

    // We need to extract date and time

  formattedDate = timeClient.getFormattedDate();

    // Extract date

  int splitT = formattedDate.indexOf("T");

  dayStamp = formattedDate.substring(0, splitT);

    //Serial.print("DATE: ");

    //Serial.println(dayStamp);

    // Extract time

  timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);

  //temperatura y humedad

  float h= dht.readHumidity();

  float t =dht.readTemperature();

   //potenciometro ph

  valorPh=analogRead(portPin)/292.5;

  //----------------------

  String variable;

  int nodo\_numero = 1;

  DynamicJsonDocument doc(1024); //creacion del json

  doc["idnodo"] = nodo\_numero;

  doc["temperatura"] = t;

  doc["humedad"] = h;

  doc["ph"]=valorPh;

  doc["fecha"] = dayStamp;

  doc["hora"] = timeStamp;

  serializeJson(doc, variable);

  Serial.println("dato a enviar: "+ variable);

  HTTPClient http; //Declare object of class HTTPClient

  WiFiClient client;

  //Specify request destination

  //http.begin(client,"URL DEL SERVIDOR");

  //http.begin(client,"http://192.168.\*\*:3000/"); //para mosquito o mqtt

  http.begin(client,"http://192.168.\*.\*:3000/datos");// para rest mysql

   //http.begin(client,"http://192.168.\*.\*:3000/datosm");// mongo rest

  http.addHeader("Content-Type", "application/json"); //Specify contenttype header

  int httpCode = http.POST(variable); //Send the request

  String payload = http.getString(); //Get the response payload

  Serial.println(httpCode); //Print HTTP return code

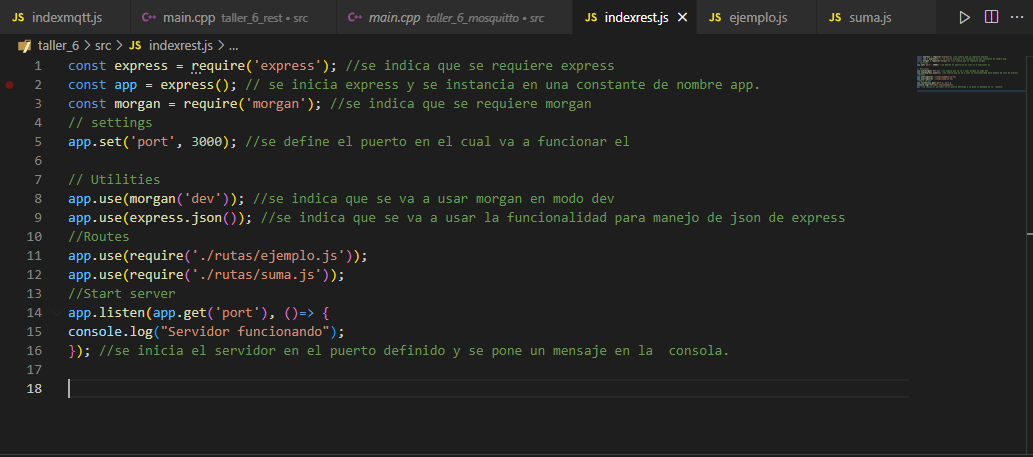
  Serial.println(payload); //Print request response payload

  http.end(); //Close connection

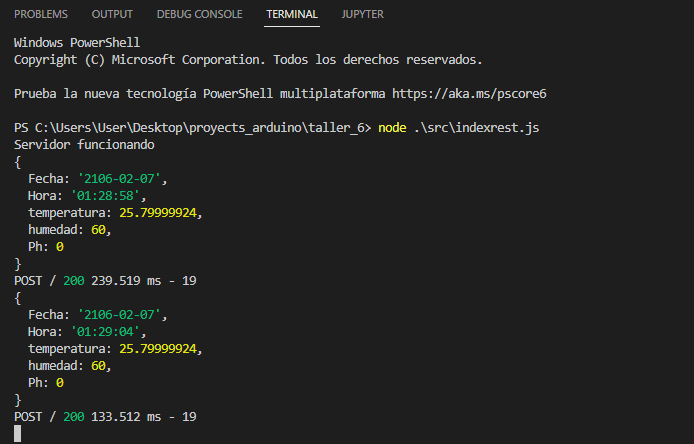
  delay(5000); //Send a request every 5 seconds

}

Código correspondiente al archivo index\_rest.js



Se ejecuta el archivo indexrest.js poder leer lo que el esp32 mande los datos json , se visualiza en la terminal de la ubicación del archivo.



Visualización del serial monitor del ESP 32

