

## 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

```
PS C:\Users\User\Desktop\projects_arduino\taller_6> npx npm init --yes
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
Need to install the following packages:
  npm
Ok to proceed? (y) y
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
Wrote to C:\Users\User\Desktop\projects_arduino\taller_6\package.json:
{
  "name": "taller_6",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" & exit 1"
  },
  "keywords": [],
  "author": "User",
  "license": "ISC"
}
```

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

```
PS C:\Users\User\Desktop\projects_arduino\taller_6> npm install mqtt --save
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
[#####] ... \ reify:concat-map: timing reifyNode:node modules/split2 Completed in 3970ms
```

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

added 44 packages, and audited 45 packages in 14s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\User\Desktop\projects\arduino\taller 6> node .\src\indexmgtt.js

```

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

```

    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:77:12)
    at node:internal/main/run_main_module:17:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}
PS C:\Users\User\Desktop\proyectos_arduino\taller_6> node src/indexmqtt.js
hola
hola_xd

```

Se realiza la publicacion al topico 1 con un mensaje

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.2006]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\User>cd ..
C:\Users>cd ..
C:\>cd "Program Files"
C:\Program Files>cd mosquito
C:\Program Files\mosquito>mosquito_pub -t topico1 -m "hola_xd"
C:\Program Files\mosquito>
```

Publicación en el tópico en formato json

```
C:\Program Files\mosquito>mosquito_pub -t topico1 -m {"op1":20,"op2":30}
C:\Program Files\mosquito>
```

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.

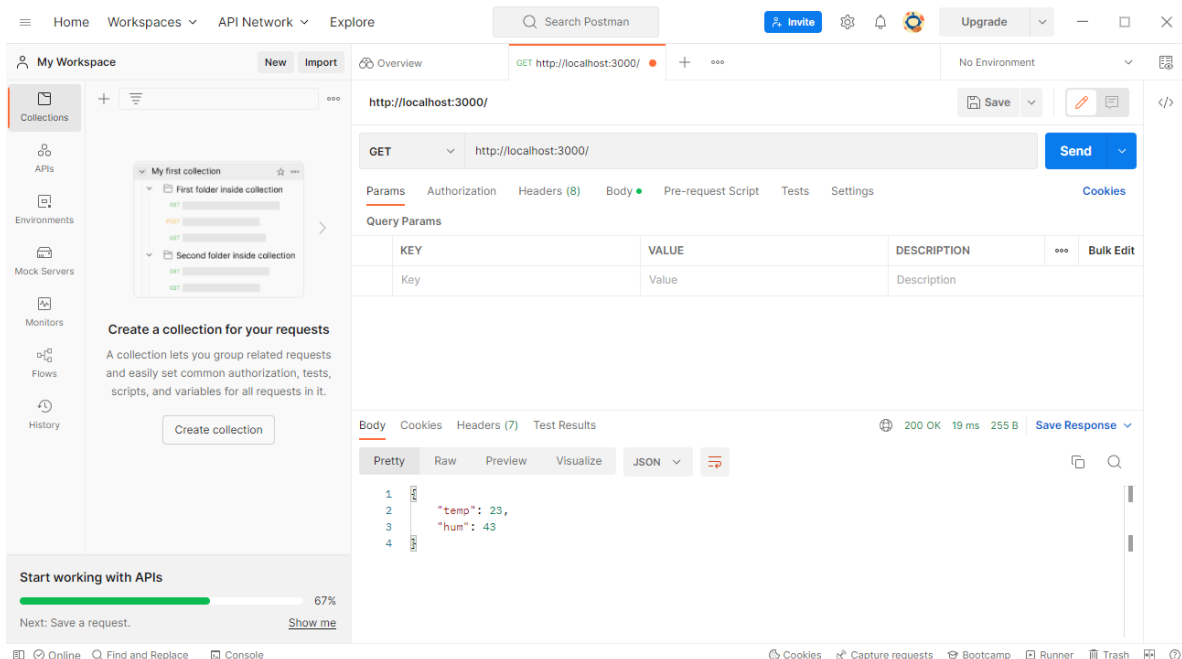
```
PS C:\Users\User\Desktop\proyectos_arduino\taller_6> npm i express morgan
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
[.....] | idealTree:taller_6: sill idealTree buildDeps
```

```
C++ main.cpp JS indexmqttjs JS indexrestjs JS ejemplojs X
taller_6 > src > rutas > JS ejemplojs > ...
1  const { Router } = require("express");
2  const router = Router();
3  router.get("/", (req, res) => {
4    res.json({ temp: 23, hum: 43 });
5  });
6  router.post("/", (req, res) => {
7    console.log(req.body);
8    res.send("datos recibidos...");
9  });
10 module.exports = router;
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
at Module._compile (node:internal/modules/cjs/loader:1105:14)
at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
at Module.load (node:internal/modules/cjs/loader:981:32)
at Function.Module._load (node:internal/modules/cjs/loader:822:12)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:77:12)
at node:internal/main/run_main_module:17:47
PS C:\Users\User\Desktop\proyectos_arduino\taller_6> node .\src\indexrest.js
Servidor funcionando
```

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

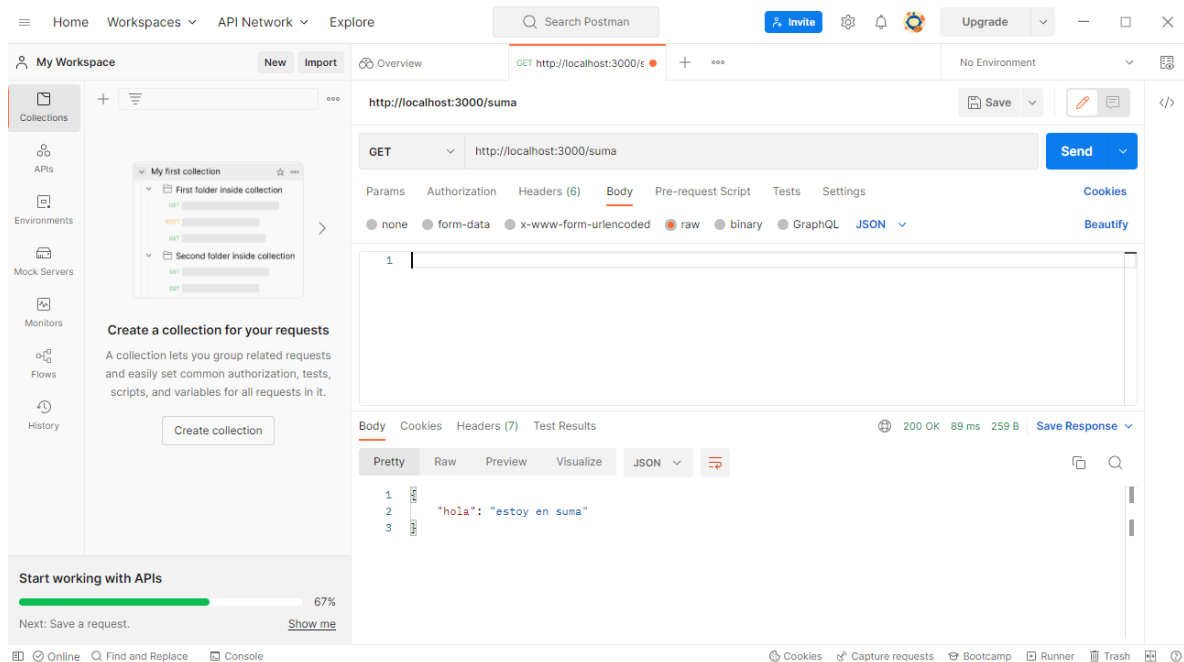


Visualización en la terminal en donde se realizó GET con postman en la extracción de datos.

```
PS C:\Users\User\Desktop\proyectos_arduino\taller_6> node .\src\indexrest.js
Servidor funcionando
{ humedad: 14 }
POST / 200 70.461 ms - 19
GET / 200 5.621 ms - 20
{ humedad: 14 }
POST / 200 3.660 ms - 19
GET / 200 1.451 ms - 20
{ nombre: 'pepito' }
POST / 200 4.310 ms - 19
```

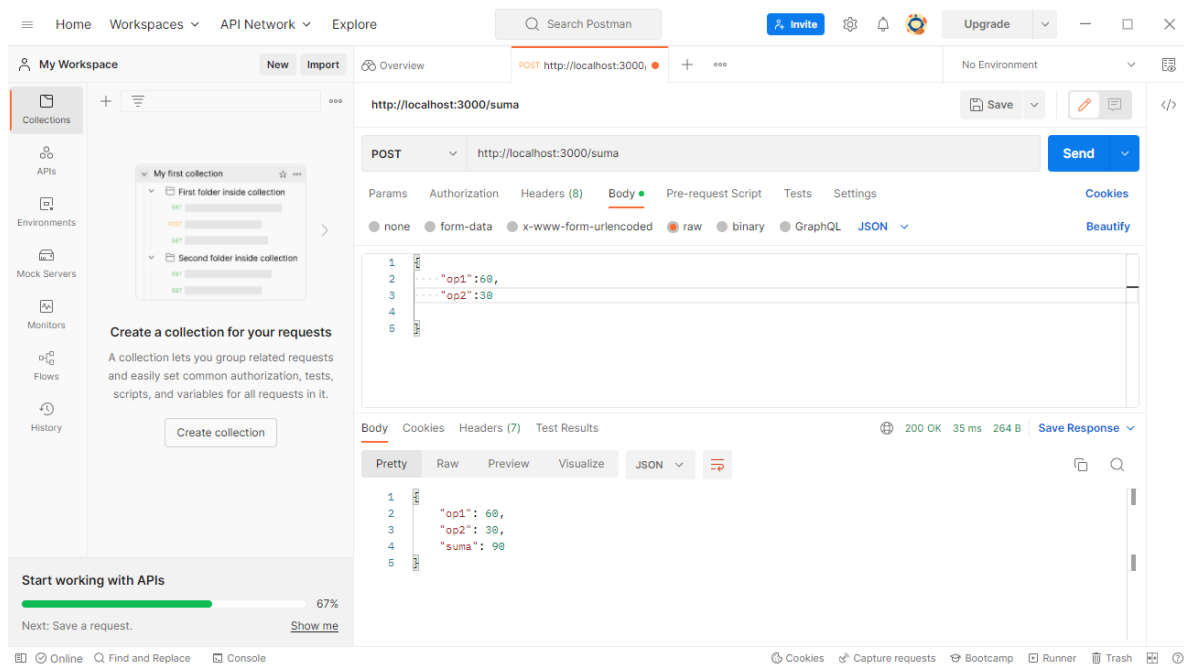
Se realiza la extracción de método GET en el dato que está en el archivo suma.js

## Suma.js en GET

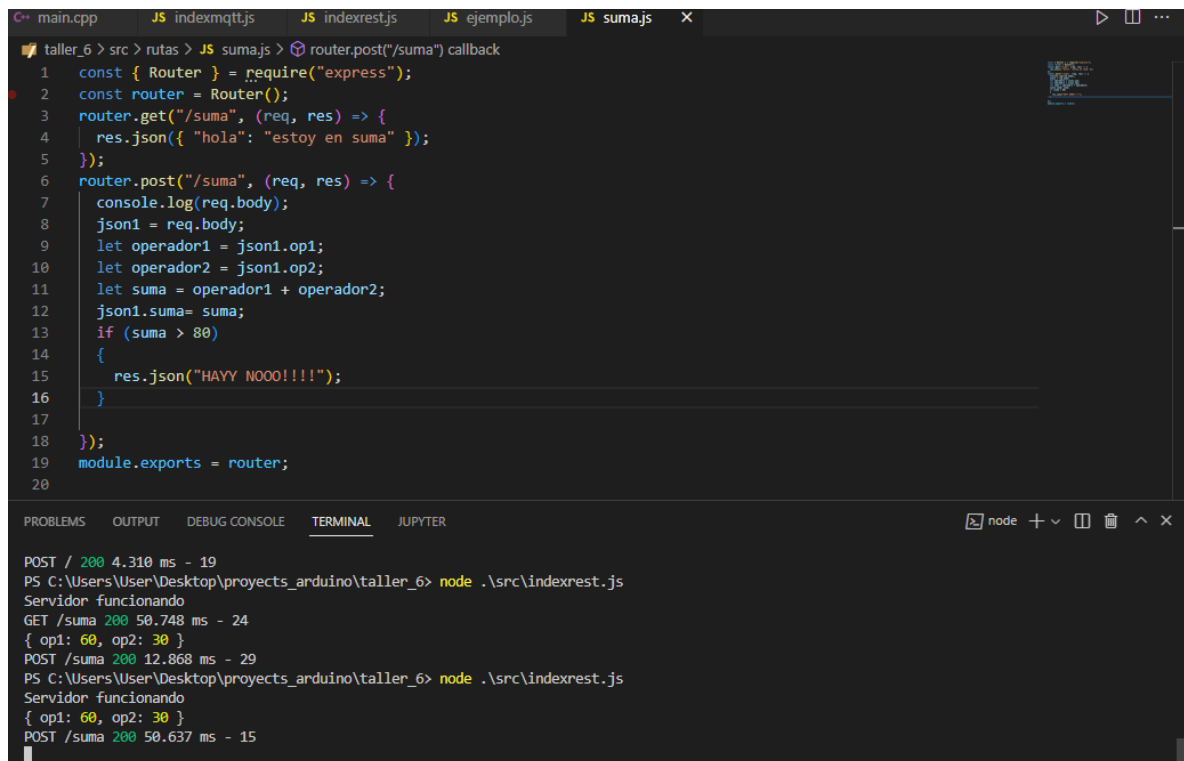


Se realiza la extracción de método POST en el dato que esta en el archivo suma.js

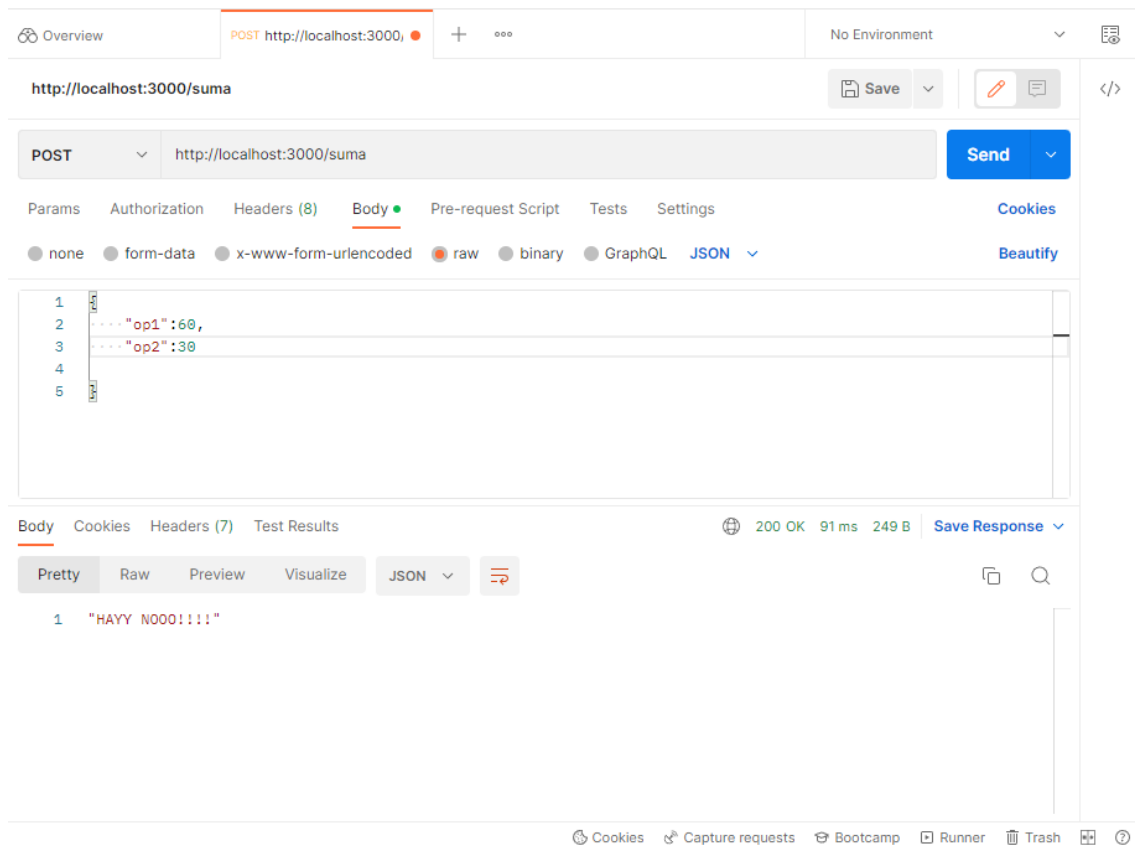
## Suma.js en POST



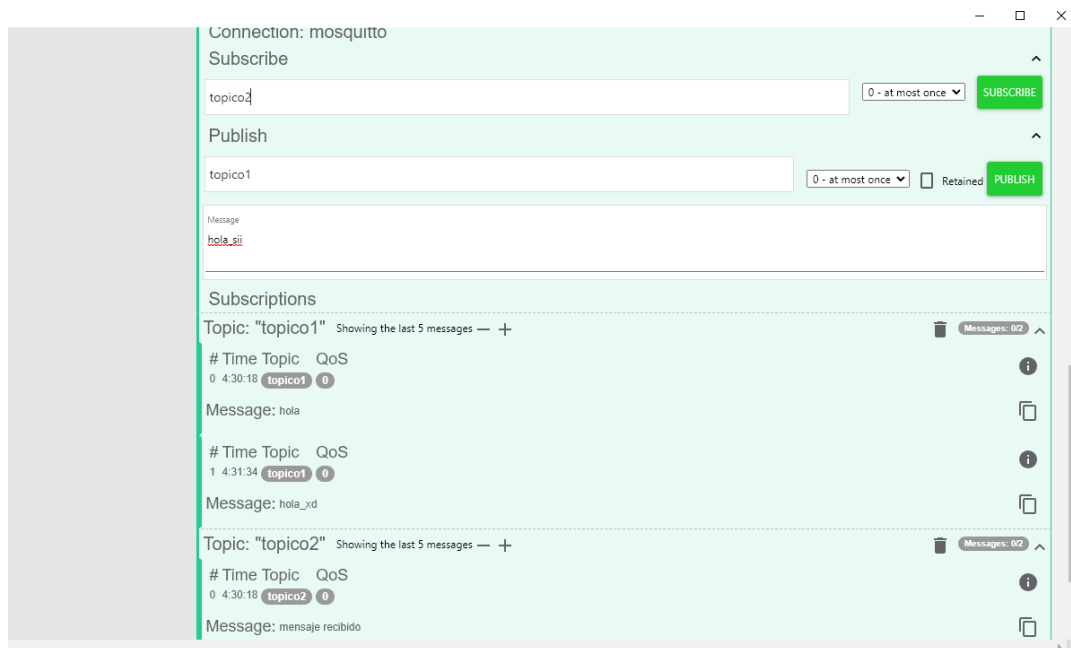
## EJEMPLO DE REALIZACION CON IF



## POST DE EJEMPLO IF



Suscripción de `topico2` en donde va a pública en el `topico1` utilizando `mqttlens`, en donde la url del servidor en configuración 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"; //topico 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();
    }
}

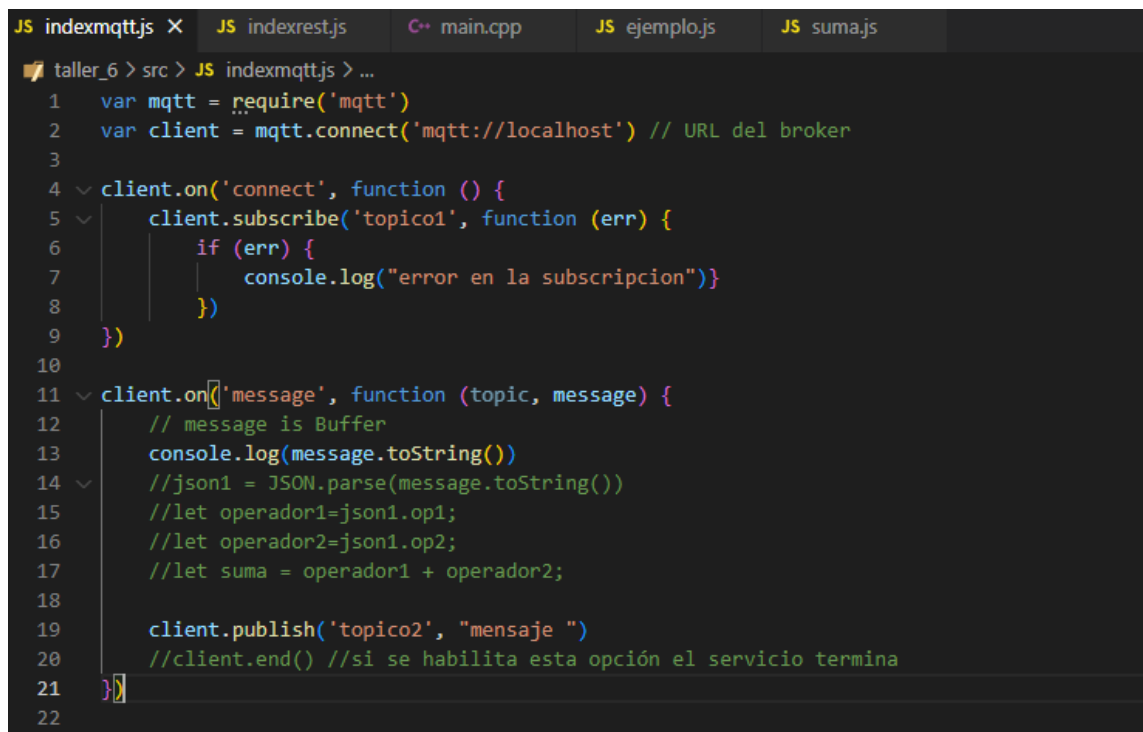
```

```

    // 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



```

JS indexmqtt.js X JS indexrest.js C++ main.cpp JS ejemplo.js JS suma.js
taller_6 > src > JS indexmqtt.js > ...
1  var mqtt = require('mqtt')
2  var client = mqtt.connect('mqtt://localhost') // URL del broker
3
4  client.on('connect', function () {
5      client.subscribe('topico1', function (err) {
6          if (err) {
7              console.log("error en la subscripcion")
8          }
9      })
10
11  client.on('message', function (topic, message) {
12      // message is Buffer
13      console.log(message.toString())
14      //json1 = JSON.parse(message.toString())
15      //let operador1=json1.op1;
16      //let operador2=json1.op2;
17      //let suma = operador1 + operador2;
18
19      client.publish('topico2', "mensaje ")
20      //client.end() //si se habilita esta opción el servicio termina
21  })
22

```

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.

```

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\User\Desktop\projects_arduino\taller_6> node .\src\indexmqtt.js
{"Fecha":"2022-09-20","Hora":"21:51:02","temperatura":25.79999924,"humedad":61,"idnodo":1,"Ph":0
}
{"Fecha":"2022-09-20","Hora":"21:51:02","temperatura":25.79999924,"humedad":61,"idnodo":1,"Ph":1
4}
{"Fecha":"2022-09-20","Hora":"21:51:02","temperatura":25.79999924,"humedad":61,"idnodo":1,"Ph":0
}

```

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

```
JS indexmqttjs  C++ main.cpp taller_6_rest + src  C++ main.cpp taller_6_mosquito + src  JS indexrestjs X  JS ejemplojs  JS suma.js  ▶ □ ...
taller_6 > src > JS indexrestjs > ...
1  const express = require('express'); //se indica que se requiere express
2  const app = express(); // se inicia express y se instancia en una constante de nombre app.
3  const morgan = require('morgan'); //se indica que se requiere morgan
4  // settings
5  app.set('port', 3000); //se define el puerto en el cual va a funcionar el
6
7  // Utilities
8  app.use(morgan('dev')); //se indica que se va a usar morgan en modo dev
9  app.use(express.json()); //se indica que se va a usar la funcionalidad para manejo de json de express
10 //Routes
11 app.use(require('./rutas/ejemplo.js'));
12 app.use(require('./rutas/suma.js'));
13 //Start server
14 app.listen(app.get('port'), ()=> {
15   console.log("Servidor funcionando");
16 }); //se inicia el servidor en el puerto definido y se pone un mensaje en la  console.
17
18
```

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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\User\Desktop\proyectos_arduino\taller_6> node .\src\indexrest.js
Servidor funcionando
{
  Fecha: '2106-02-07',
  Hora: '01:28:58',
  temperatura: 25.79999924,
  humedad: 60,
  Ph: 0
}
POST / 200 239.519 ms - 19
{
  Fecha: '2106-02-07',
  Hora: '01:29:04',
  temperatura: 25.79999924,
  humedad: 60,
  Ph: 0
}
POST / 200 133.512 ms - 19
```

Visualización del serial monitor del ESP 32

```
dato a enviar: {"Fecha":"2106-02-07","Hora":"01:28:58","temperatura":25.79999924,"humedad":60,"Ph":0}
200
datos recibidos....
dato a enviar: {"Fecha":"2106-02-07","Hora":"01:29:04","temperatura":25.79999924,"humedad":60,"Ph":0}
200
datos recibidos....
```

