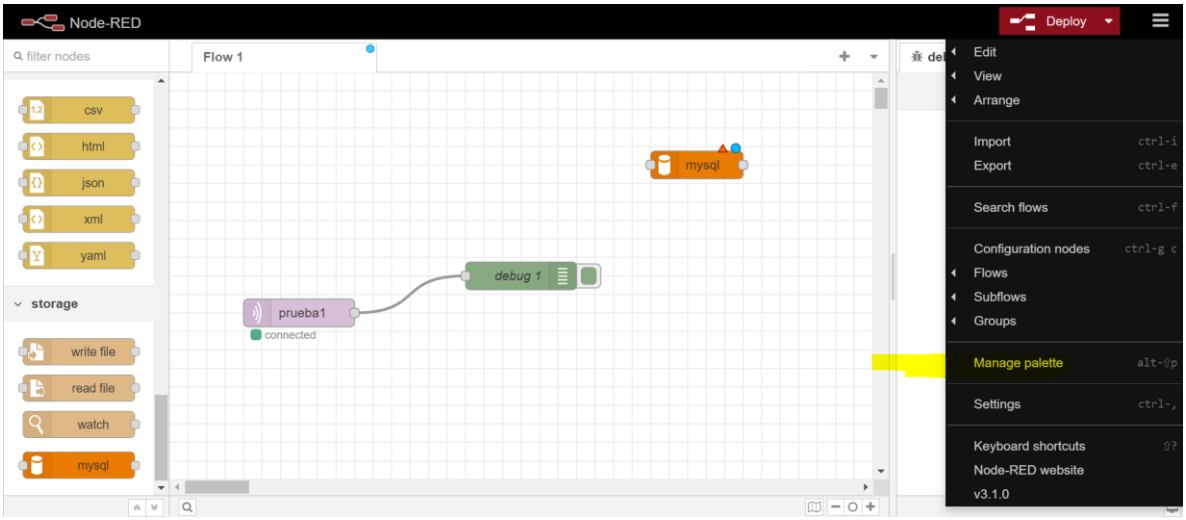
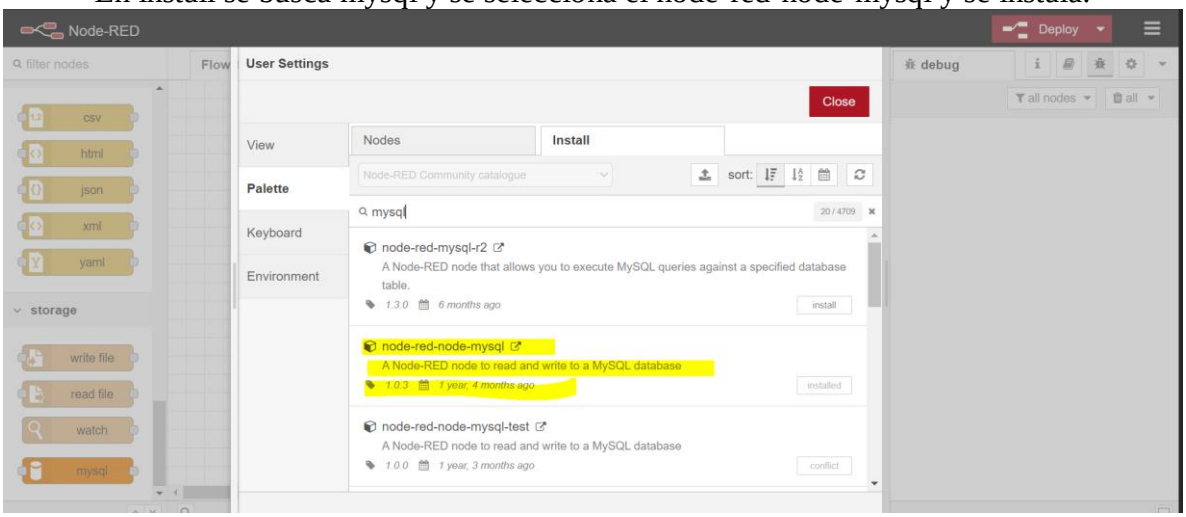
Actividad 5 IoT

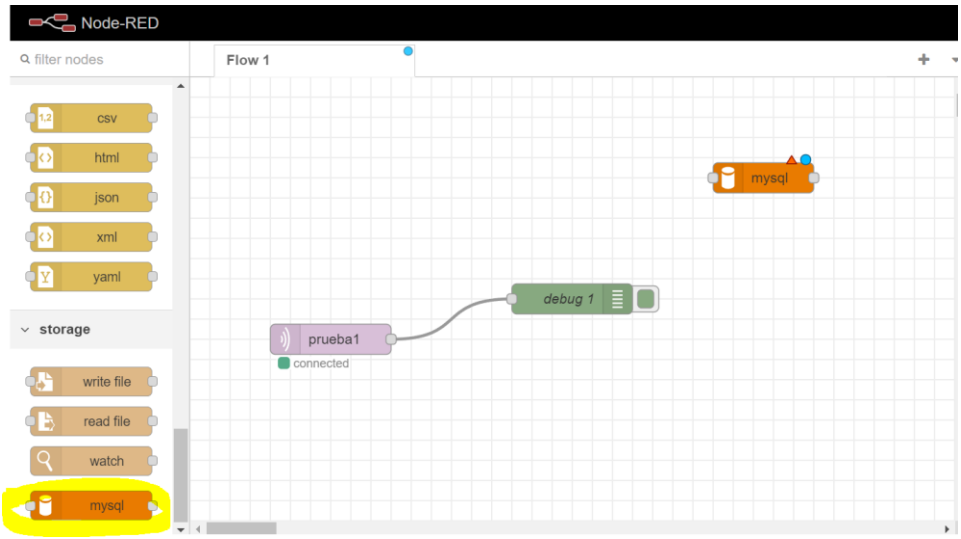
Diego Iván Perea Montealegre (2238513) diego.perea@uao.edu.co
Facultad de Ingeniería, Universidad Autónoma de Occidente
Cali, Valle del Cauca

Para usar mysql en node red dar en el menú en Manage palette :

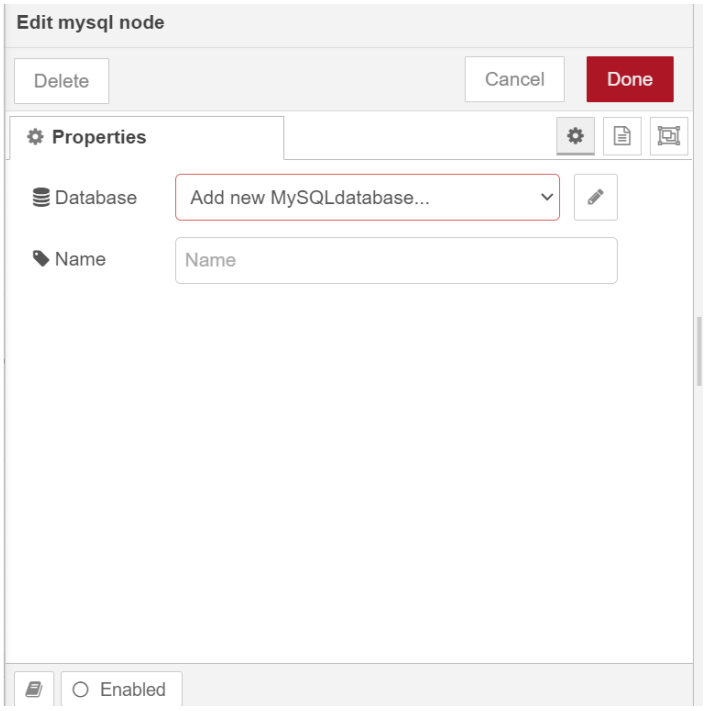


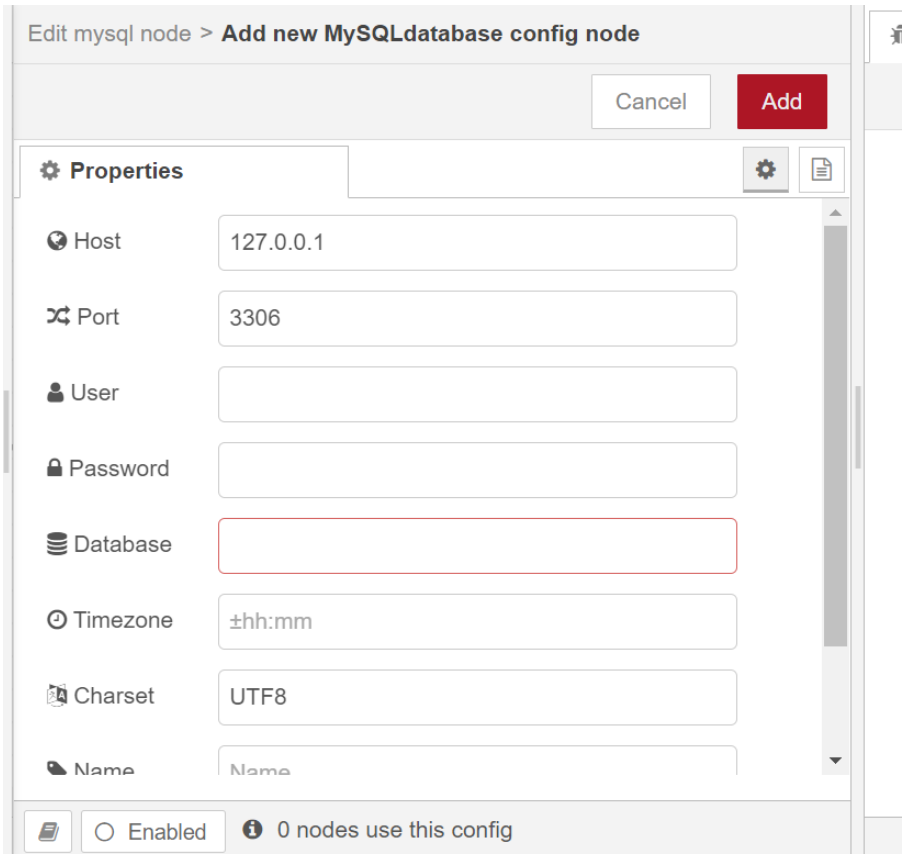
En install se busca mysql y se selecciona el node-red-node-mysql y se instala:





Se clickea mysql y se edita:





Se crea la base de datos , en este caso sea usa Docker para crear un contenedor de mysql, en el cual el el nombre del contenedor es mymysql , el usuario es root y se le da una contraseña :

docker run --name mymysql -e MYSQL_ROOT_PASSWORD=mypassword -p 3306:3306 -d mysql:latest

Se ingresa al contenedor mymysql:
docker exec -it mymysql bash

Se ingresa a mysql :
mysql -u root -p

Se pone la contraseña y se crea la base de datos llamada “iaiot” :

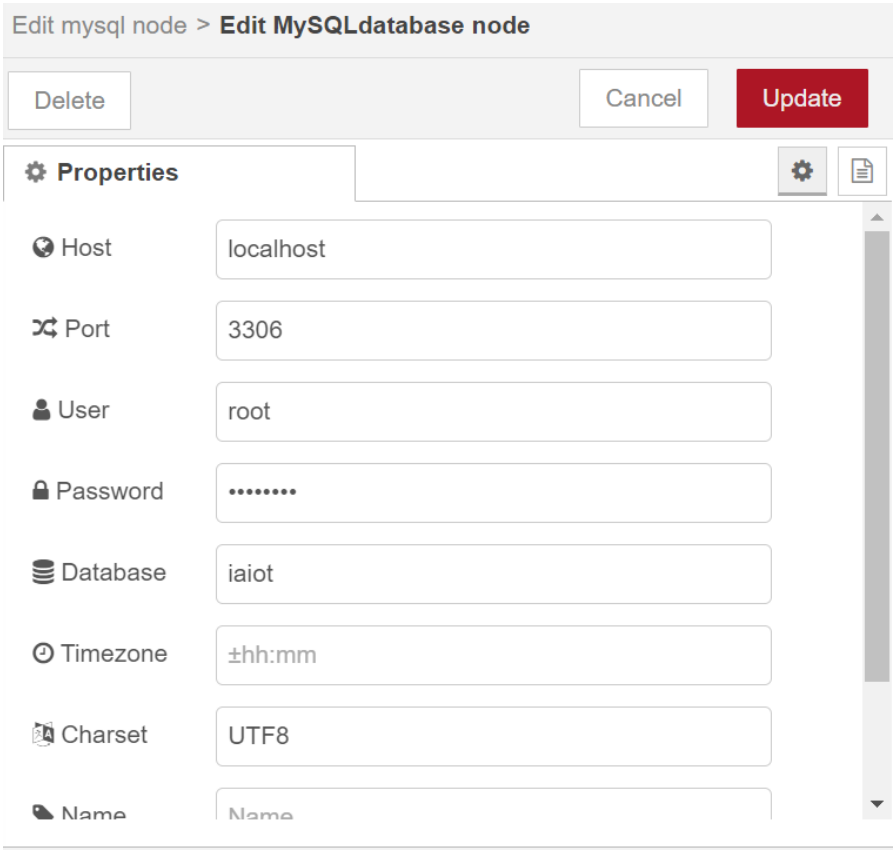
```
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database iaiot;
Query OK, 1 row affected (0.02 sec)
```

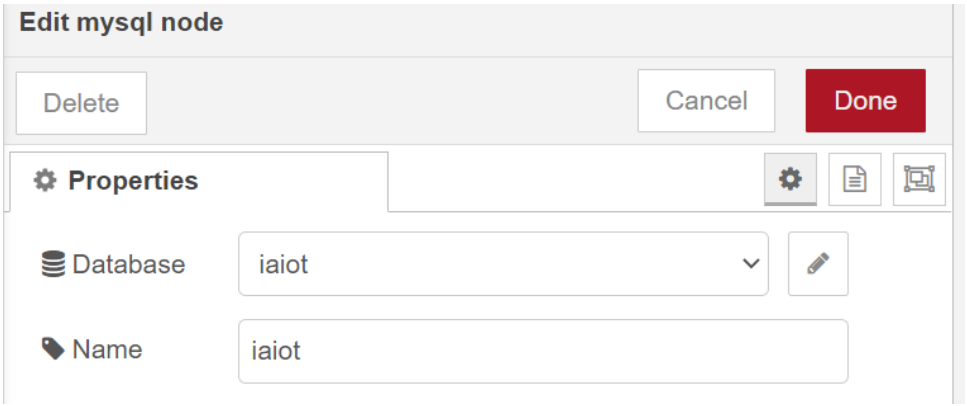
Se crea la tabla “datos” y sus parámetros :

```
mysql> use iaiot;
Database changed
mysql> create table datos (
    -> id int auto_increment,
    -> idnodo int,
    -> temperatura float,
    -> humedad float,
    -> fecha datetime default now(),
    -> primary key(id));
Query OK, 0 rows affected (0.09 sec)
```

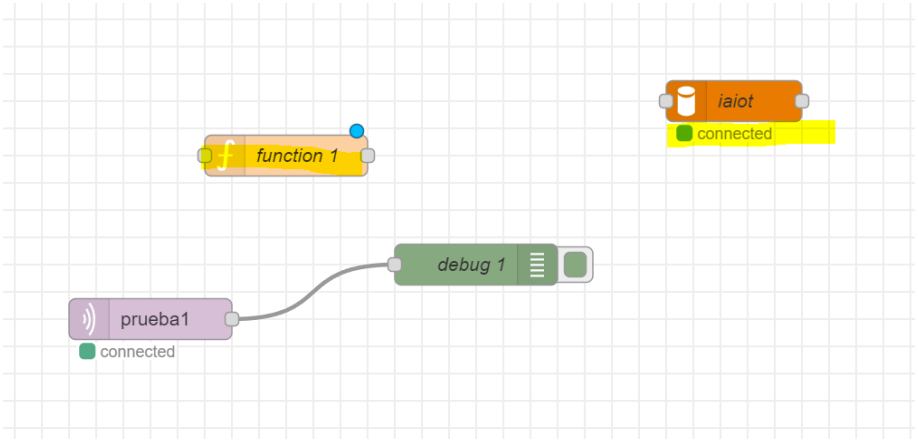
```
mysql> desc datos;
+-------------+----------+------+-----+-------------------+-------------------+
| Field       | Type     | Null | Key | Default           | Extra             |
+-------------+----------+------+-----+-------------------+-------------------+
| id          | int      | NO   | PRI | NULL              | auto_increment    |
| idnodo      | int      | YES  |     | NULL              |                   |
| temperatura | float    | YES  |     | NULL              |                   |
| humedad     | float    | YES  |     | NULL              |                   |
| fecha       | datetime | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+----------+------+-----+-------------------+-------------------+
5 rows in set (0.02 sec)
```

Para node red con mysql :





Se pone desploy para observar si se conectó , y se agrega la función :

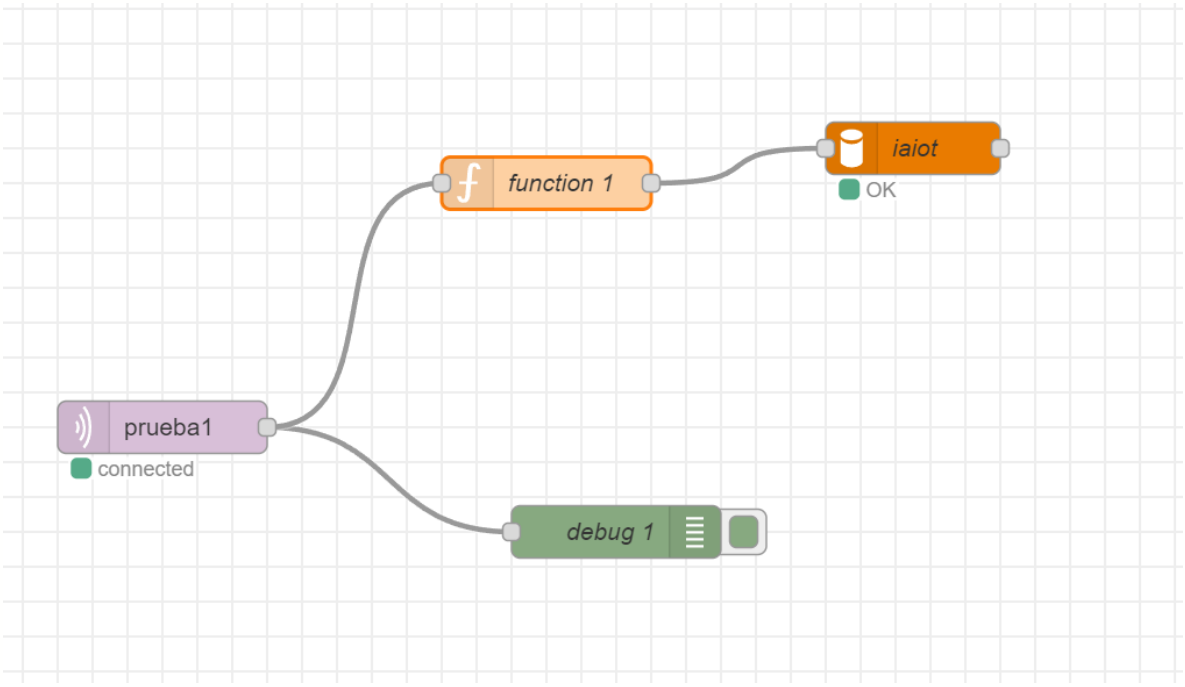


Se edita la función :

El código es el siguiente:

```
var json = msg.payload;
msg.topic = "insert into datos values (null," + json.idnodo + ","  +
json.temperatura + "," + json.humedad +
",now())";

return msg;
```

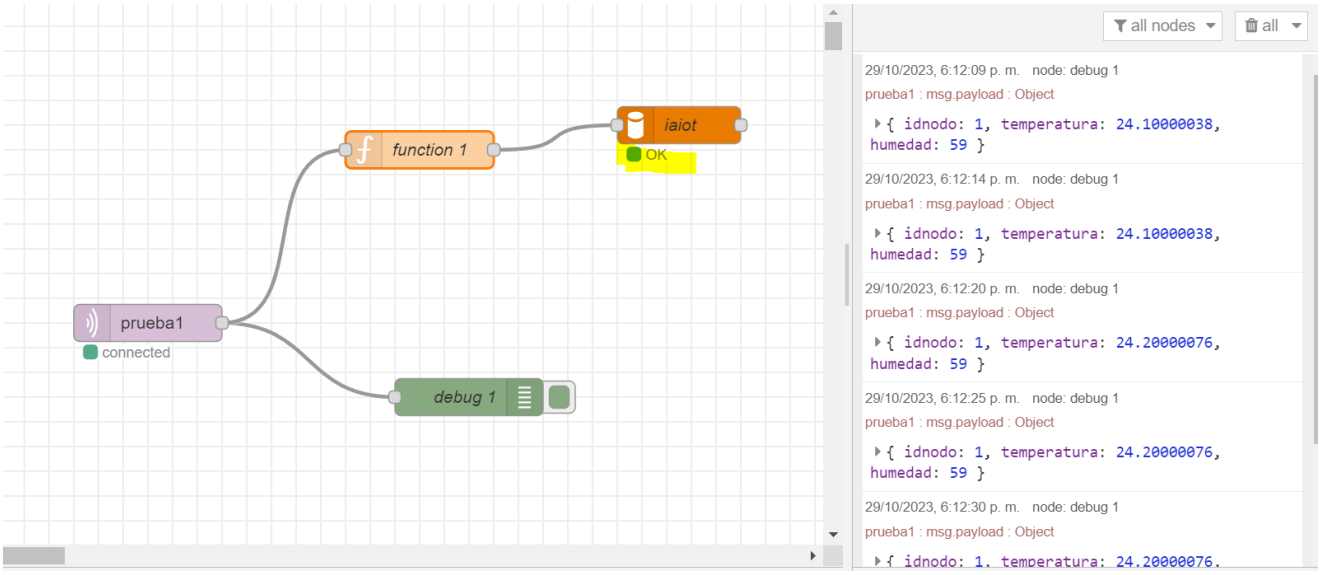
Se conecta la función



Se ejecuta código ESP32:

```
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.10000038,"humedad":59}
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.10000038,"humedad":59}
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.20000076,"humedad":59}
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.20000076,"humedad":59}
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.20000076,"humedad":59}
Attempting MQTT connection...connected
{"idnodo":1,"temperatura":24.29999924,"humedad":59}
```

Y se puede visualizar el buen funcionamiento en Node-RED dando en la base de datos un OK :

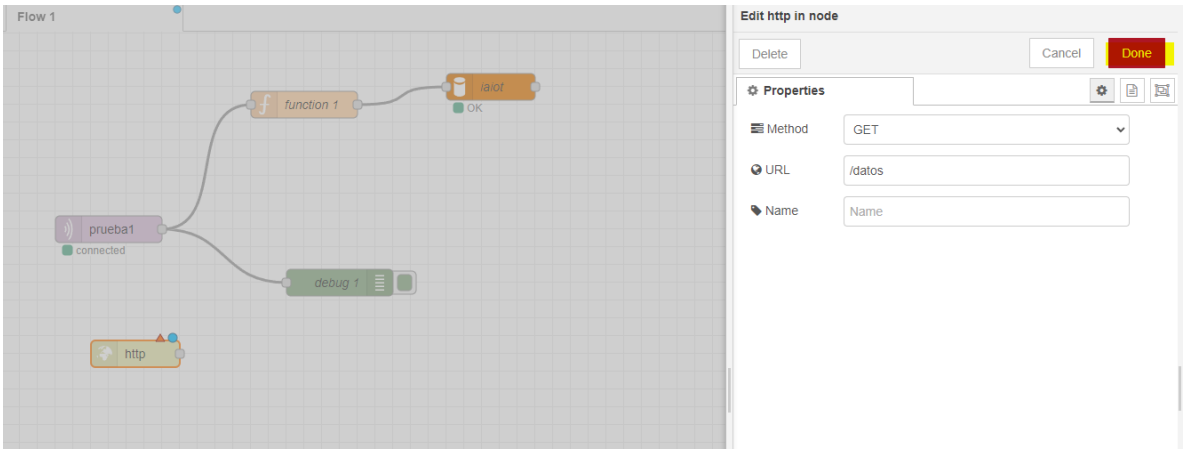


Se visualiza en la base de datos , que fue exitosamente agregado:

```
mysql> select * from datos;
+----+--------+-------------+---------+---------------------+
| id | idnodo | temperatura | humedad | fecha               |
+----+--------+-------------+---------+---------------------+
|  1 |      1 |        24.1 |      59 | 2023-10-29 23:12:09 |
|  2 |      1 |        24.1 |      59 | 2023-10-29 23:12:14 |
|  3 |      1 |        24.2 |      59 | 2023-10-29 23:12:20 |
|  4 |      1 |        24.2 |      59 | 2023-10-29 23:12:25 |
|  5 |      1 |        24.2 |      59 | 2023-10-29 23:12:30 |
|  6 |      1 |        24.3 |      59 | 2023-10-29 23:12:36 |
+----+--------+-------------+---------+---------------------+
6 rows in set (0.00 sec)
```

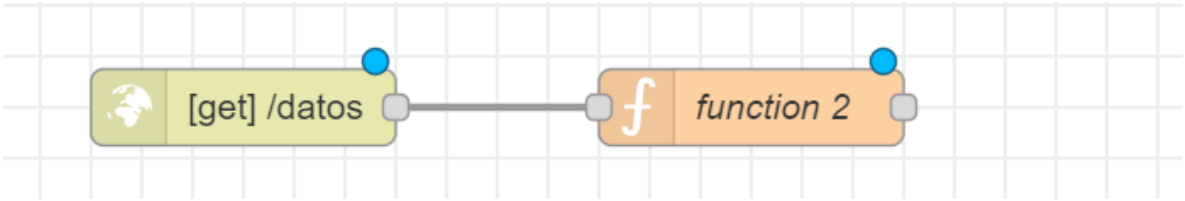
Usando API REST en Node-RED

Se agrega el bloque ‘http in’ y dar ruta “/datos” :

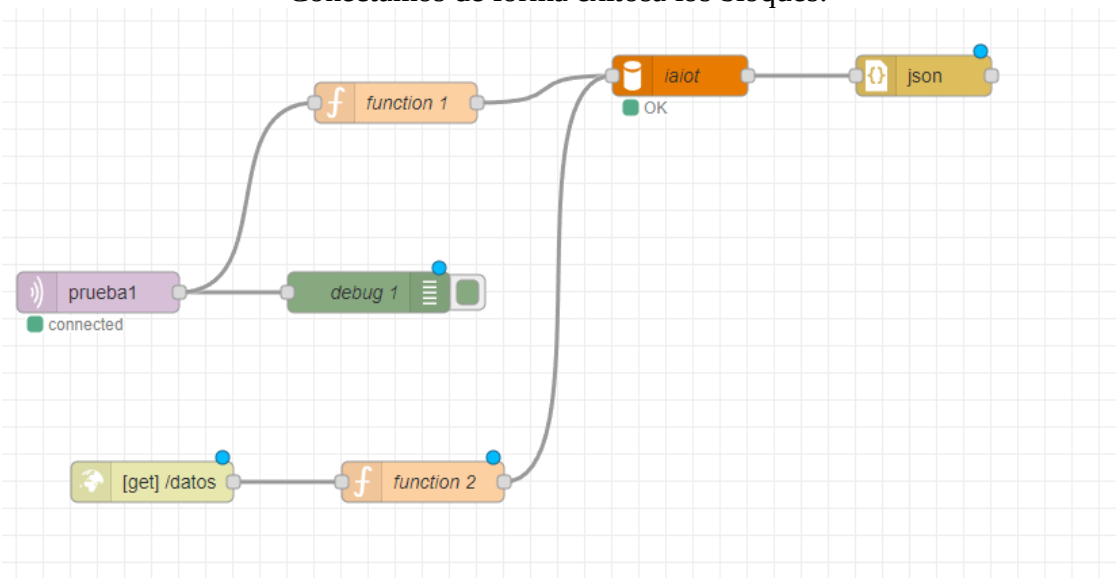


Se agrega una función para esa ruta :
Se da con el siguiente código debido a GET :

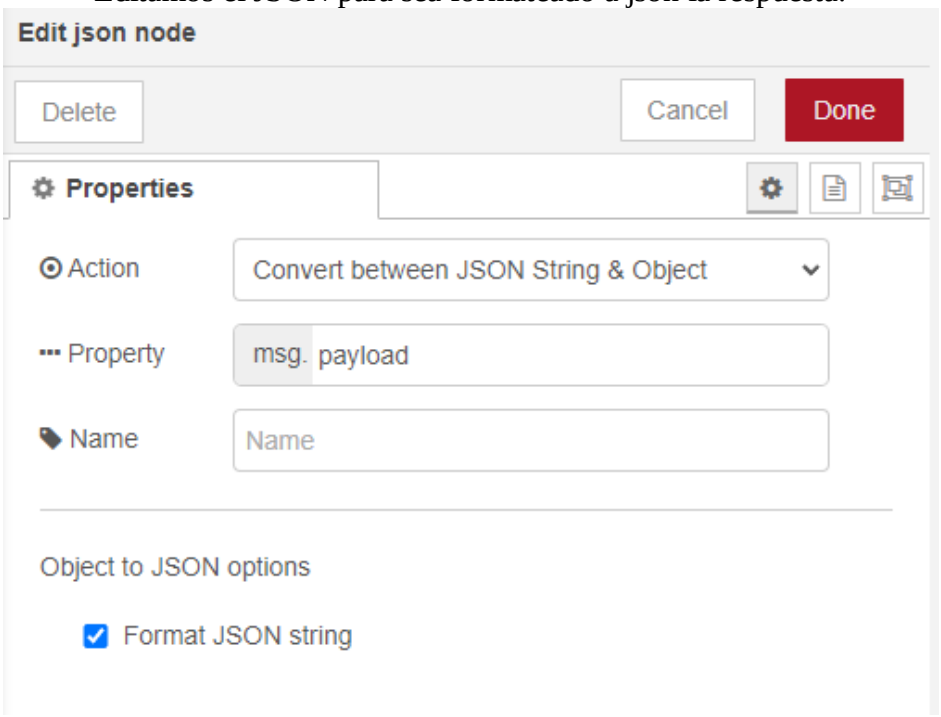
```
msg.topic = "select * from datos";
return msg;
```



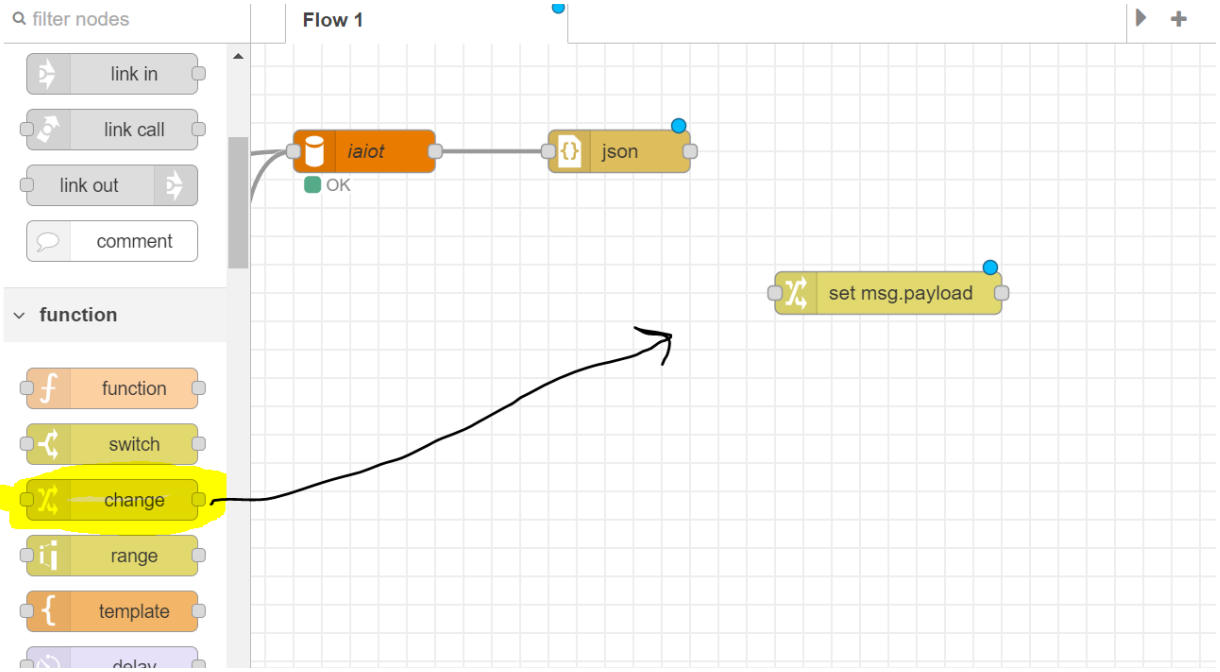
Conectamos de forma exitosa los bloques:

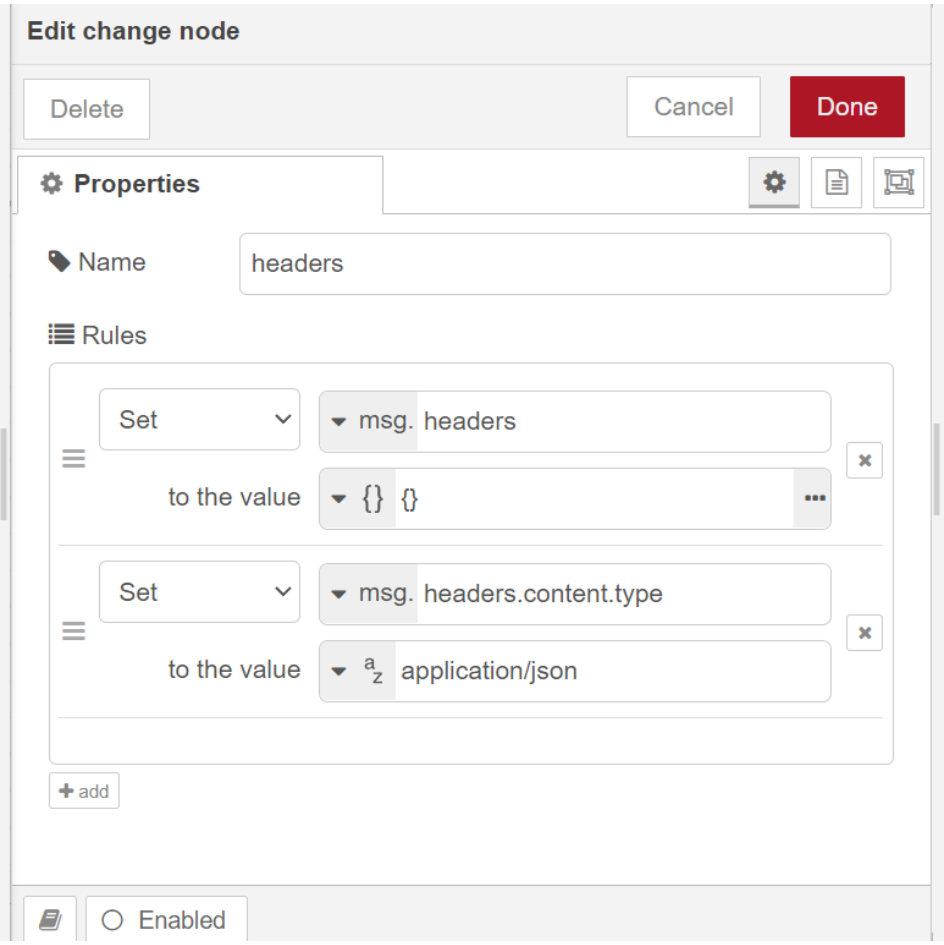


Editamos el JSON para sea formateado a json la respuesta:

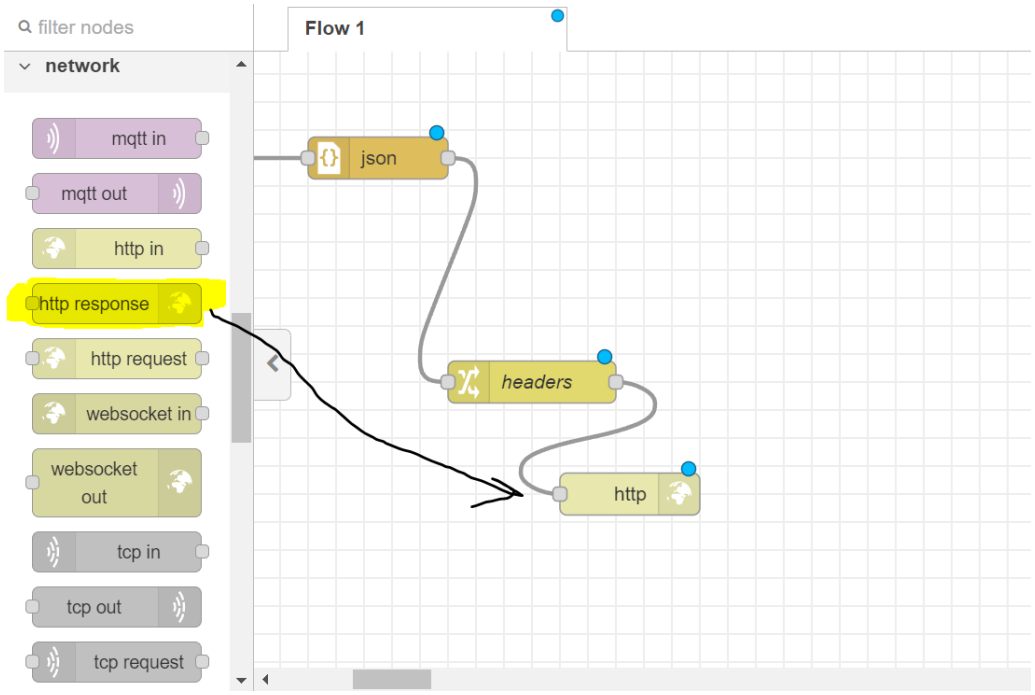


Se agrega “change” para la respuesta http y se edita ‘change’ :

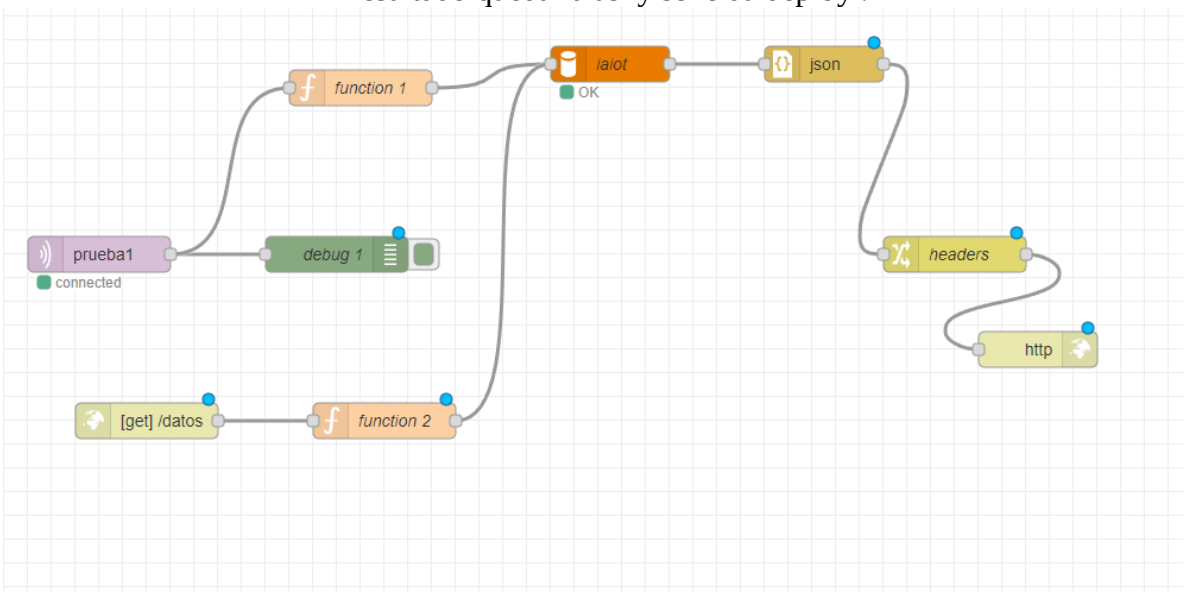


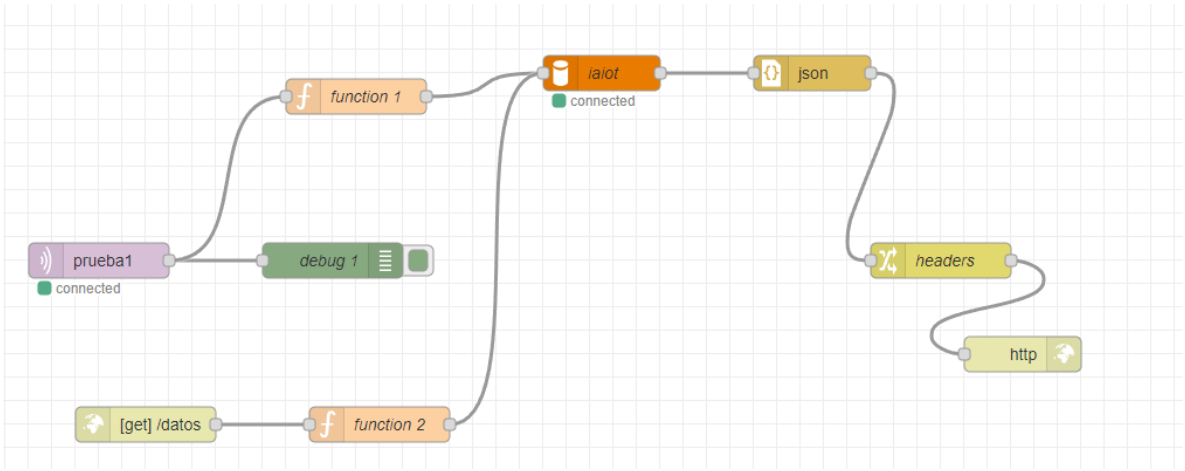


Se agrega por ultimo el http response :

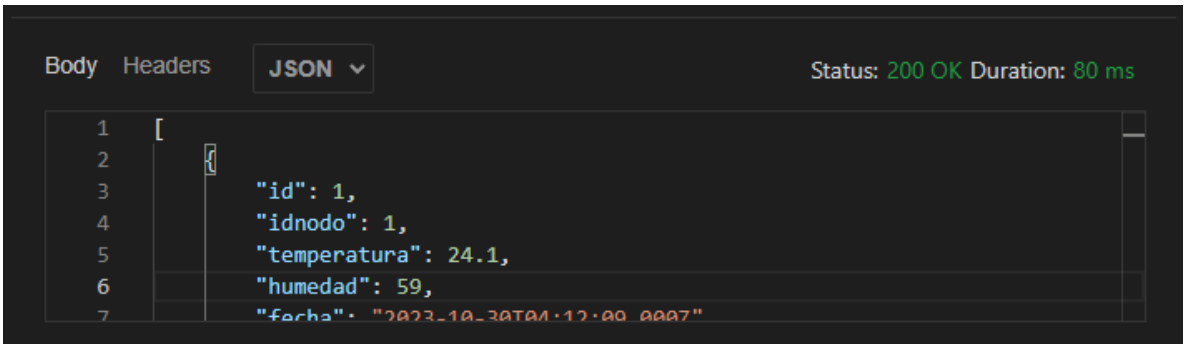


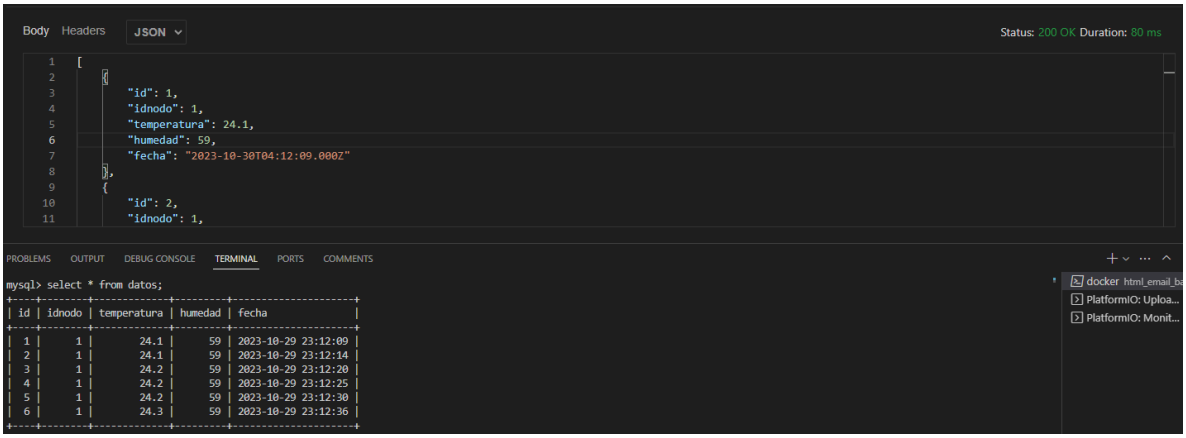
El resultado quedaría así y se le da deploy :





Usando request (ejemplo Postman u otros) en la url con GET :
http://localhost:1880/datos





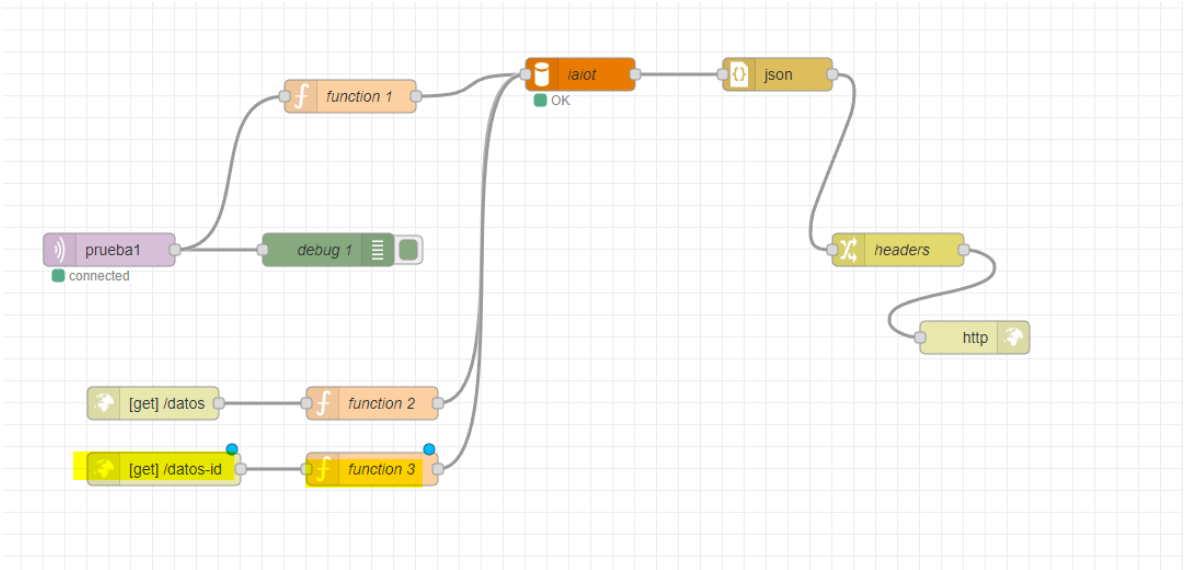
Aquí se muestra toda la respuesta del GET:

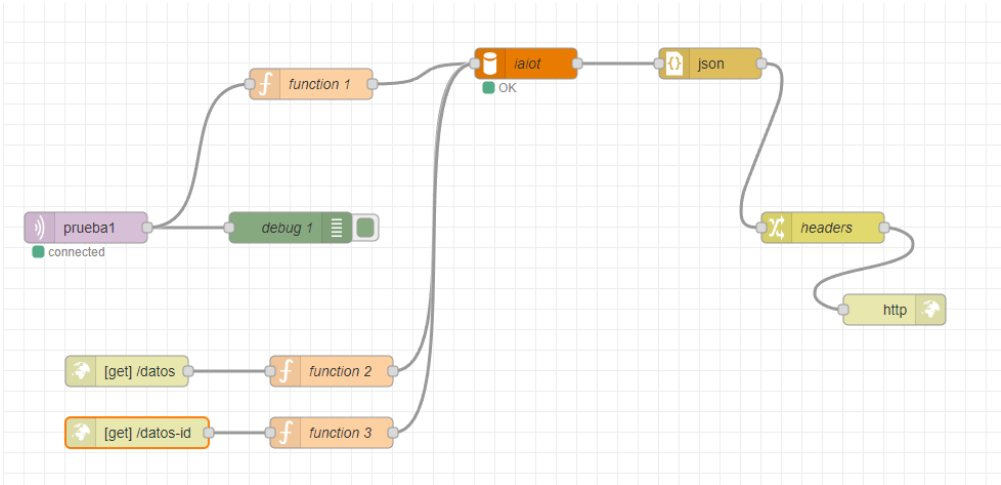

```
[
    {
        "id": 1,
        "idnodo": 1,
        "temperatura": 24.1,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:09.000Z"
    },
    {
        "id": 2,
        "idnodo": 1,
        "temperatura": 24.1,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:14.000Z"
    },
    {
        "id": 3,
        "idnodo": 1,
        "temperatura": 24.2,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:20.000Z"
    },
```

```
    {
        "id": 4,
        "idnodo": 1,
        "temperatura": 24.2,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:25.000Z"
    },
    {
        "id": 5,
        "idnodo": 1,
        "temperatura": 24.2,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:30.000Z"
    },
    {
        "id": 6,
        "idnodo": 1,
        "temperatura": 24.3,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:36.000Z"
    }
]
```

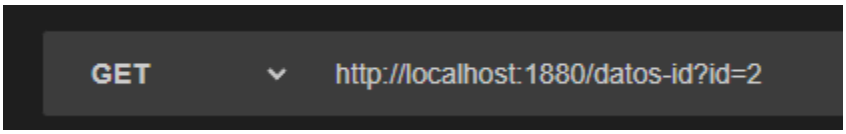

Se agrega otra ruta /datos-id con la función del siguiente código :

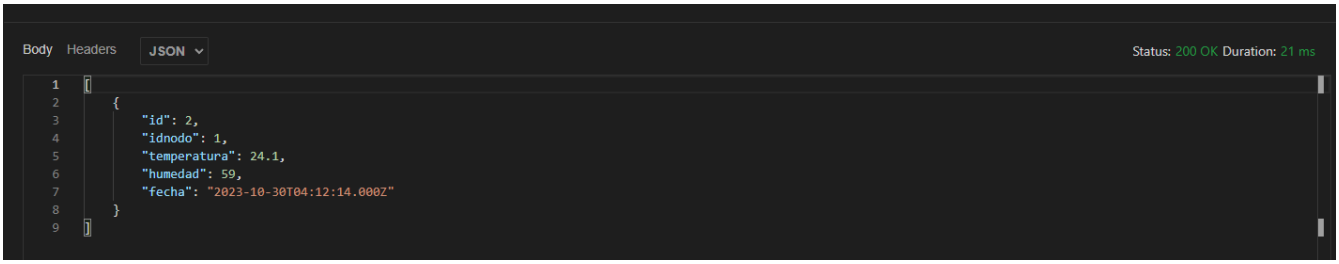
```
var id = msg.payload.id;
msg.topic = "select * from datos where id =" + id;
return msg;
```





Se realiza el GET con http://localhost:1880/datos-id?id=2



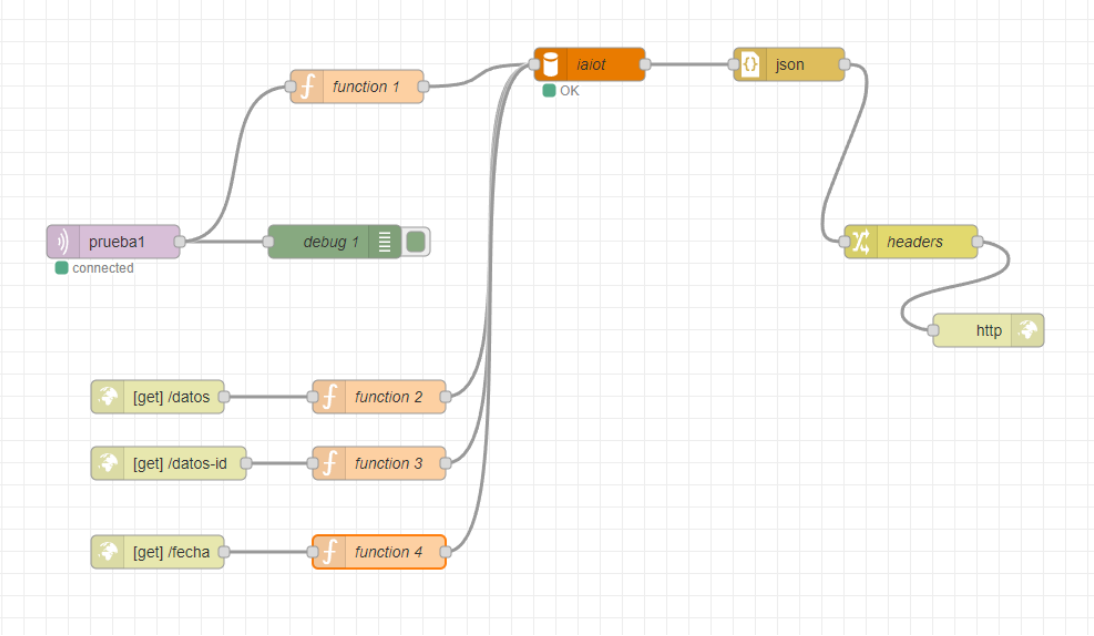



```
[
    {
        "id": 2,
        "idnodo": 1,
        "temperatura": 24.1,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:14.000Z"
    }
]
```


Por ultimo se hace un filtro GET de fecha de dos parámetros, de fecha de inicio y fecha de fin .

Por lo cual se crea ruta “/fecha” y se agrega función con el siguiente código :

```
var fechaInicio = msg.req.query.fechaInicio;
var fechaFin = msg.req.query.fechaFin;

if (fechaInicio && fechaFin) {
    msg.topic = "SELECT * FROM datos WHERE fecha BETWEEN '" + fechaInicio +
"' AND '" + fechaFin + "'";
} else {

    msg.topic = "SELECT * FROM datos";
}

return msg;
```

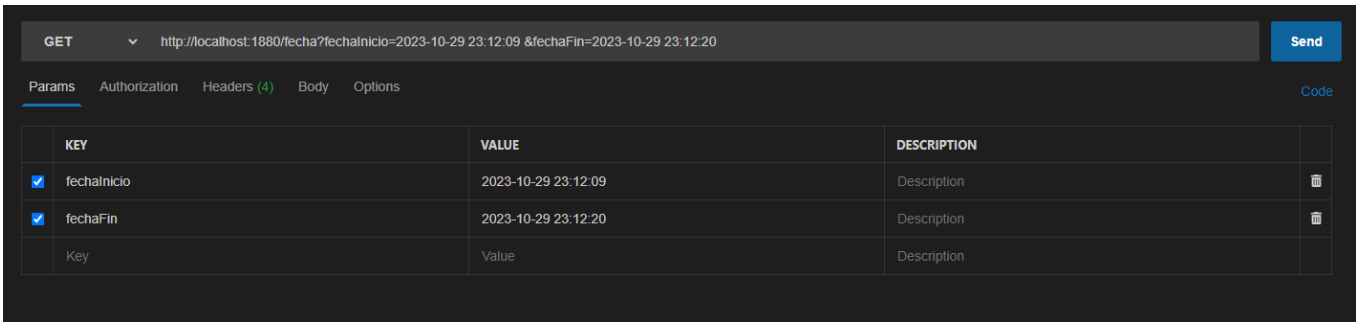


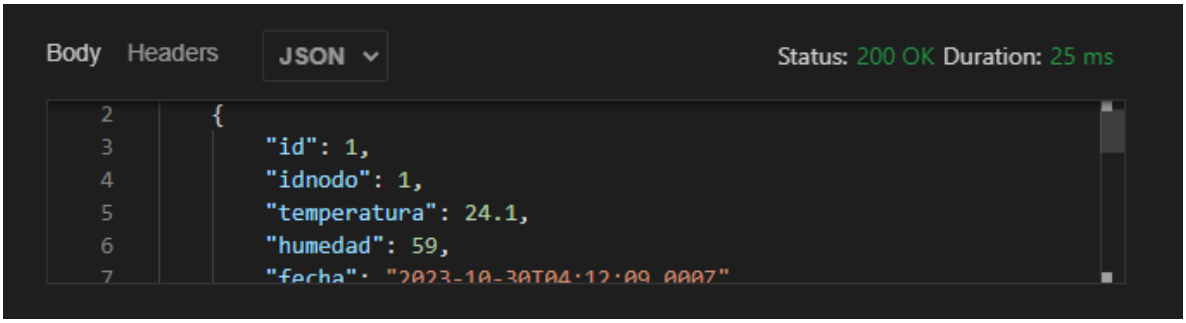
Sabemos que la información guardada es :

```
mysql> select * from datos;
+----+--------+-------------+---------+---------------------+
| id | idnodo | temperatura | humedad | fecha               |
+----+--------+-------------+---------+---------------------+
|  1 |      1 |        24.1 |      59 | 2023-10-29 23:12:09 |
|  2 |      1 |        24.1 |      59 | 2023-10-29 23:12:14 |
|  3 |      1 |        24.2 |      59 | 2023-10-29 23:12:20 |
|  4 |      1 |        24.2 |      59 | 2023-10-29 23:12:25 |
|  5 |      1 |        24.2 |      59 | 2023-10-29 23:12:30 |
|  6 |      1 |        24.3 |      59 | 2023-10-29 23:12:36 |
+----+--------+-------------+---------+---------------------+
```

En la url se pone lso parámetros de la fecha y hora , por ejmeplo si queremos de fecha inicio 2023-10-29 23:12:09 y hasta 2023-10-29 23:12:20 , se pone en el request :

http://localhost:1880/fecha?fechaInicio=2023-10-29 23:12:09 &fechaFin=2023-10-29 23:12:20





La salida es :


```
[
    {
        "id": 1,
        "idnodo": 1,
        "temperatura": 24.1,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:09.000Z"
    },
    {
        "id": 2,
        "idnodo": 1,
        "temperatura": 24.1,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:14.000Z"
    },
    {
        "id": 3,
        "idnodo": 1,
```

```
        "temperatura": 24.2,
        "humedad": 59,
        "fecha": "2023-10-30T04:12:20.000Z"
    }
]
```

Que es correctamente a los datos

```
mysql> select * from datos;
+----+--------+-------------+---------+---------------------+
| id | idnodo | temperatura | humedad | fecha               |
+----+--------+-------------+---------+---------------------+
|  1 |      1 |        24.1 |      59 | 2023-10-29 23:12:09 |
|  2 |      1 |        24.1 |      59 | 2023-10-29 23:12:14 |
|  3 |      1 |        24.2 |      59 | 2023-10-29 23:12:20 |
|  4 |      1 |        24.2 |      59 | 2023-10-29 23:12:25 |
|  5 |      1 |        24.2 |      59 | 2023-10-29 23:12:30 |
|  6 |      1 |        24.3 |      59 | 2023-10-29 23:12:36 |
+----+--------+-------------+---------+---------------------+
```

Codigo ESP32 Platformio :

```
#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 14   // 4 = PIN D4
#define DHTTYPE    DHT11
DHT dht(DHTPIN, DHTTYPE);


// 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*";//name wifi
const char* password = "*PASSWORD_WIFI*"; // clave de wifi
char mqttBroker[] = "192.168.**.*"; //ip del servidor


char mqttClientId[] = "prueba1"; //cualquier nombre
char inTopic[] = "prueba1";//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();

  String variable;
  StaticJsonDocument<256> doc;

 /*  doc["Fecha"] = dayStamp;
  doc["Hora"] = timeStamp; */
```

```
  doc["idnodo"] = 1;
  doc["temperatura"] = t;
  doc["humedad"] = h;




  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();
}
```