Parcial 3 MQTT y Mongo BD

Diego Iván Perea Montealegre (2185751) diego.perea@uao.edu.co

Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Valle del Cauca

Siguiendo con el contexto de la problemática identificada que es el mal control de temperatura y humedad relativa en el almacenamiento de los productos de frutas y hortalizas; En donde el objeto de la aplicación son las frutas y hortalizas, y el objeto será una caja inteligente compuesto por sensores de temperatura y humedad relativa que permita recolectar información de la condición de almacenamiento.
La información de las condiciones de la fruta s visualizarán en una aplicación Web que debe entregar los datos recibidos de las frutas u hortalizas, en el que darán avisos de alerta según la personalización del usuario.
a) Temperatura (baja, optima, alta)
b) Humedad (baja, optima, alta)
*Se tomará como ejemplo el limón

Temperatura °C	Mensaje	Humedad relativa %	Mensaje
T<10	Temperatura Baja	H<85	Humedad Baja
10>=T<=13	Temperatura Optima	85>=T<=90	Humedad Optima
T>13	Temperatura Alta	H>90	Humedad Alta

Tabla 1 Visualización de mensaje según datos adquiridos

El mensaje y los datos adquiridos serán visualizados en una pagina web para que el usuario tenga la información clara y detallada de lo que esta pasando en el almacenamiento de este , este almacenamiento es dado por cajas , en cada caja estará un dispositivo en el que le enviará los datos adquiridos de temperatura y humedad al servidor para así visualizarlos como se menciono anteriormente en una página web.

Los datos que visualizará el usuario en la web son : nodo (Representa tipo de alimento) , la temperatura, humedad , fecha y hora.

La estructuración de almacenamiento de los datos se dará en el cual un usuario puede tener muchos nodos, estos nodos representan diferentes tipos de producto como frutas u hortalizas , en donde los diferentes niveles de temperatura y humedad serán guardados en tabla denominada Nodo y los datos adquiridos temperatura y humedad con sus respectivo nodo, fecha y hora estarán en la tabla llamada Dato que según estos procesamiento de datos ya definidos en la tabla 1 realizará un mensaje nombrado como estado que se almacenará en la tabla alerta al valor de dato alertado , nodo, fecha y hora en donde se produjo este estado. El protocolo utilizado para este parcial será el de MQTT y la ase de datos a utilizar es Mongo.

Una mejor interpretación del almacenamiento de la información y de estas conexiones que tienen esta en la siguiente figura:

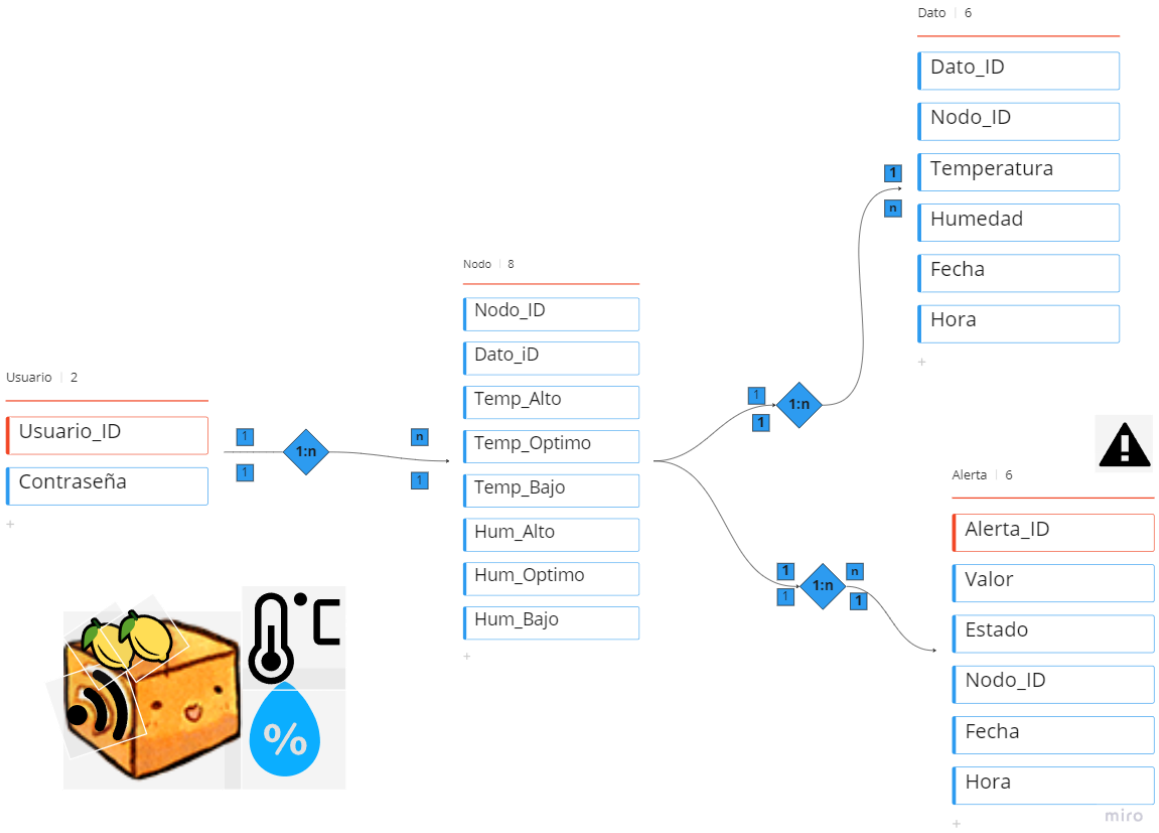


Figura 1 Tipos de conexiones y tablas en donde se almacenará la información

Aunque hay que aclarar que para este parcial solo se requiere el envió de datos usando un protocolo MQTT y sean almacenados en una base de datos Mongo , por lo que el registro de usuarios y demás pasos se requiere una implementación web.

Para crear la conexiones del envío de datos se crea el json esto se realiza creando una carpeta que en mi caso es llamada “taller_6”, y con la terminal establecerse en la carpeta y ejecutar el comando

npm init –yes

Este comando nos permite es crear un json (package.json) con toda la descripción de proyecto y además muestra los paquetes instalados , dado el caso que no funcione se requiere realizar el comando

npmx init –yes

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

npm install mqtt –save

Hecho esto se crea una carpeta dentro de “taller_6” llamada “src” , en el cual creamos un archivo “indexmqtt3.js” que será la parte de mqtt , que hará la conexión con la base de datos mongo .

En la siguiente figura ignorar carpeta “rutas” y archivos .js menos indesmqtt3.js ,debido a que estos no aplican para los requerimientos dados del este parcial.

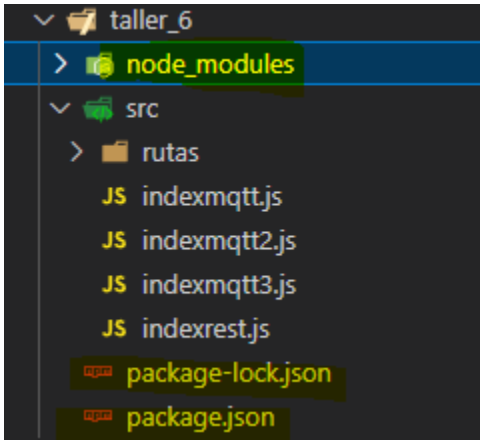


Figura 2 Paquetes instalados de json y mqtt

La url en donde se dará la conexión con mogo es "mongodb://localhost:27017/" y la publicación de los mensajes estarán dados en "topico1", el nombre de la base de datos fue de “ProyectoDiegoP” y su correspondiente tabla fue “datosNodo”.

El código de indexmqt3.js con base de datos Mongo es el siguiente:

```
var mqtt = require("mqtt");
var client = mqtt.connect("mqtt://localhost");
const mysql = require("mongodb");
var MongoClient = require("mongodb").MongoClient;
var url = "mongodb://localhost:27017/";
client.on("connect", function () {
  client.subscribe("topico1", function (err) {
    if (err) {
      console.log("error en la subscripcion");
    }
  });
});
client.on("message", function (topic, message) {
  // message is Buffer
  json1 = JSON.parse(message.toString());
  console.log(json1);
  //client.publish('topico2', 'mensaje recibido')
  MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    var dbo = db.db("ProyectoDiegoP");
    dbo.collection("datosNodo").insertOne(json1, function (err, res) {
      if (err) throw err;
      console.log("1 document inserted");
```

```
      db.close();
    });
  });
  //client.end() //si se habilita esta opción el servicio termina
});
```

En el ESP32 se realiza las conexiones con el puerto 1883 debido a que realizara el envío de datos con el protocolo MQTT y demás configuración de creación de JSON ,fecha y hora y posterior funcionamiento de sensor con el DHT11 que censará la temperatura y humedad relativa.

Código de ESP32 con Protocolo MQTT:

```
#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
const char* password = "******"; // clave de wifi
char mqttBroker[] = "192.1**.*.**"; //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();

  //----------------------
  String variable;
  StaticJsonDocument<256> doc;

  doc["nodo"] = 1;
  doc["temperatura"] = t;
  doc["humedad"] = h;
  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();
}
```

La realización del envió de datos se realizará de dos formas diferentes desde dos nodos distintos. Un nodo será desarrollado con el hardware ESP32 y el otro nodo será simulado a través de un cliente MQTTLens.

Comenzando con el MQTTLens se configuró para el protocolo MQTT.

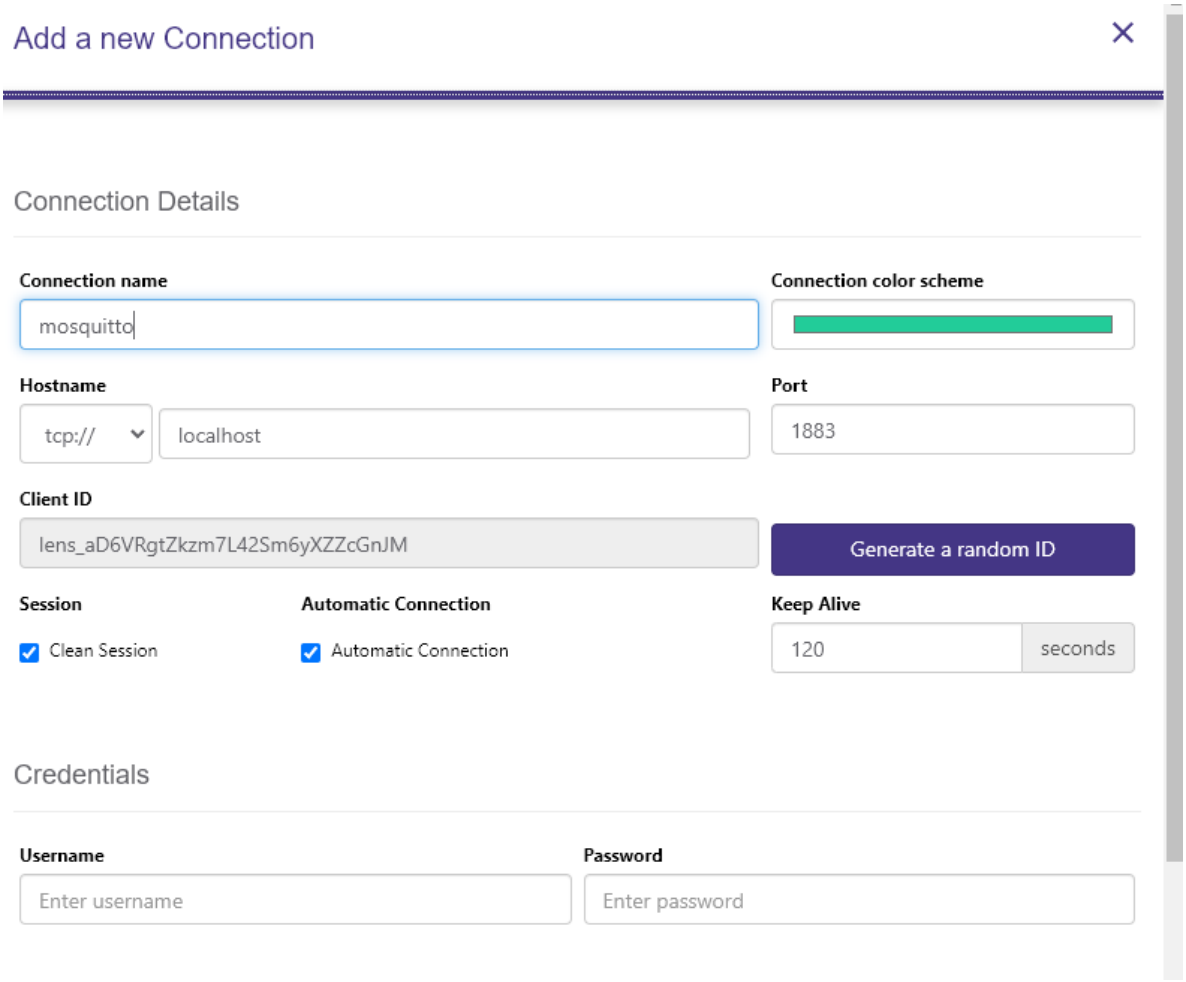
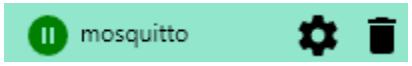


Figura 3 Configuración de protocolo MQTT

Sabemos que la conexión ha sido exitosa con el indicador en verde



Para publicar el dato JSON se el tópico al cual publicar en el cual este tópico se asignó en el indexmqtt3.js que fue el “topico1” ;En el espaciado del mensaje se crea el JSON que son los datos en que se van a enviar hacia la base de datos de Mongo ya definida.

En los datos a enviar esta el nodo , temperatura , humedad , fecha y hora , el nodo esta definido como el numero dos por lo que el otro nodo uno va ser asignado en el ESP32.

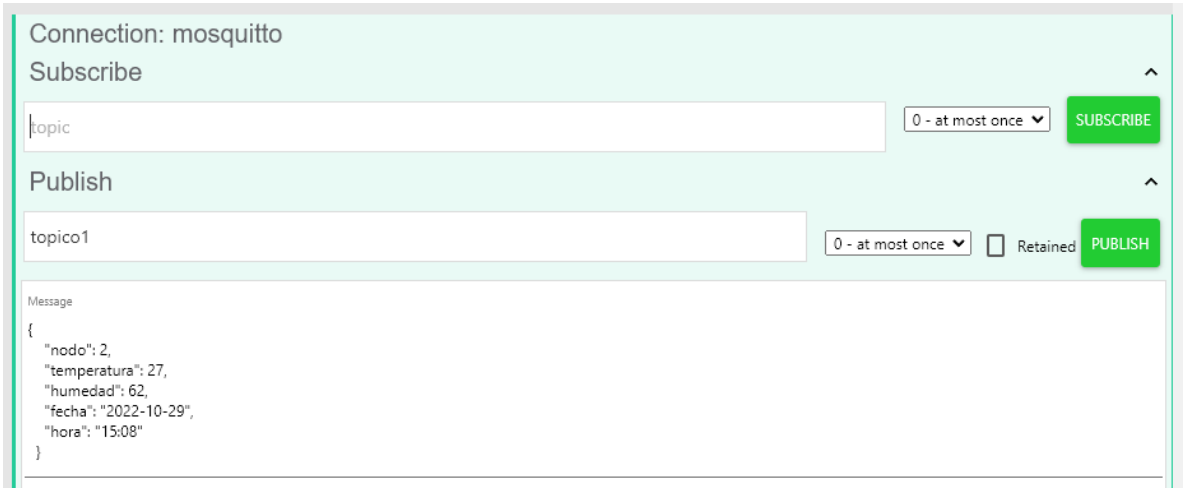


Figura 4 Envió de datos con protocolo MQTT desde MQTTLens

Para que se reciba los datos se lanza el servidor , ubicándose en la carpeta y archivo , dando el siguiente comando:

Node .\src\indexmqtt3.js

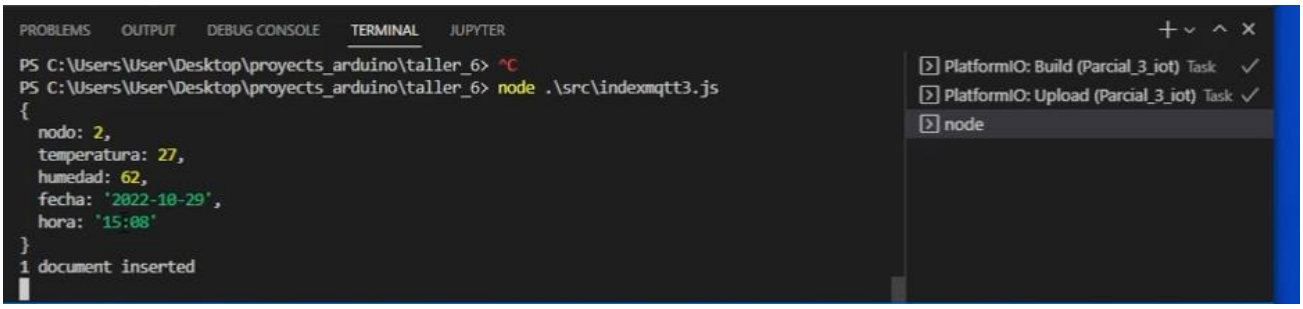


Figura 5 Recibimiento y almacenamiento de datos en Mongo desde MQTTLens

Ahora con el ESP32 se sube el código a través de Platformio

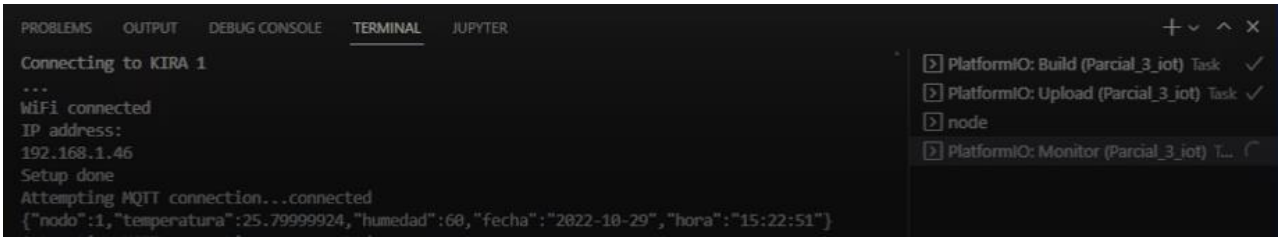


Figura 6 Envió de datos con protocolo MQTT desde ESP32

Figura 7 Recibimiento y almacenamiento de datos en Mongo desde ESP32

Para visualizar que se hallan almacenados los datos en la base de datos “PoyectoDiegoP” en Mongo , se ejecuta la cmd , que en Windows es tecla Windows + r.

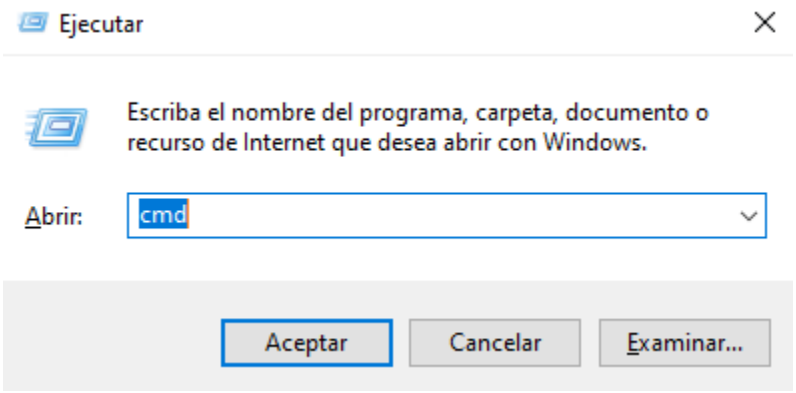


Figura 8.Entrando a la cmd

En la cmd se ejecuta el comando mongosh , que nos permite entrar a mongo

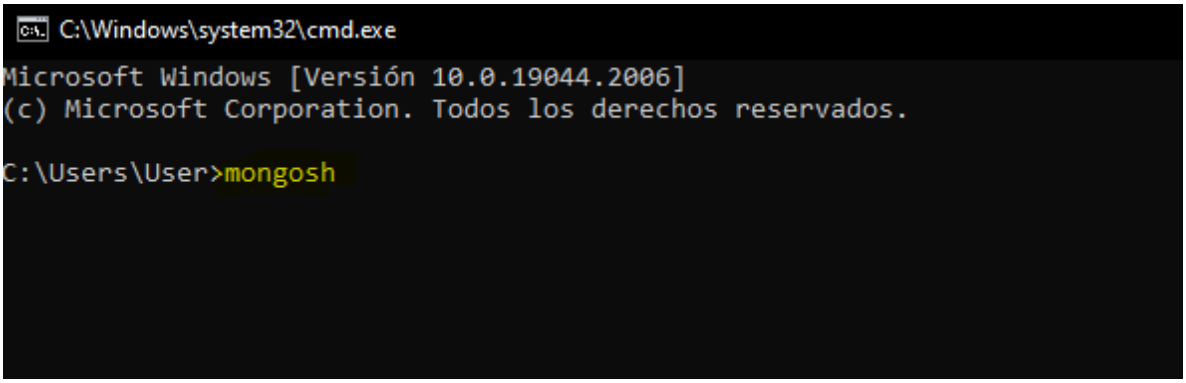


Figura 9.Entrando a Mongo con la cmd

Usamos la base de datos “PoyectoDiegoP” , con el comando use PoyectoDiegoP

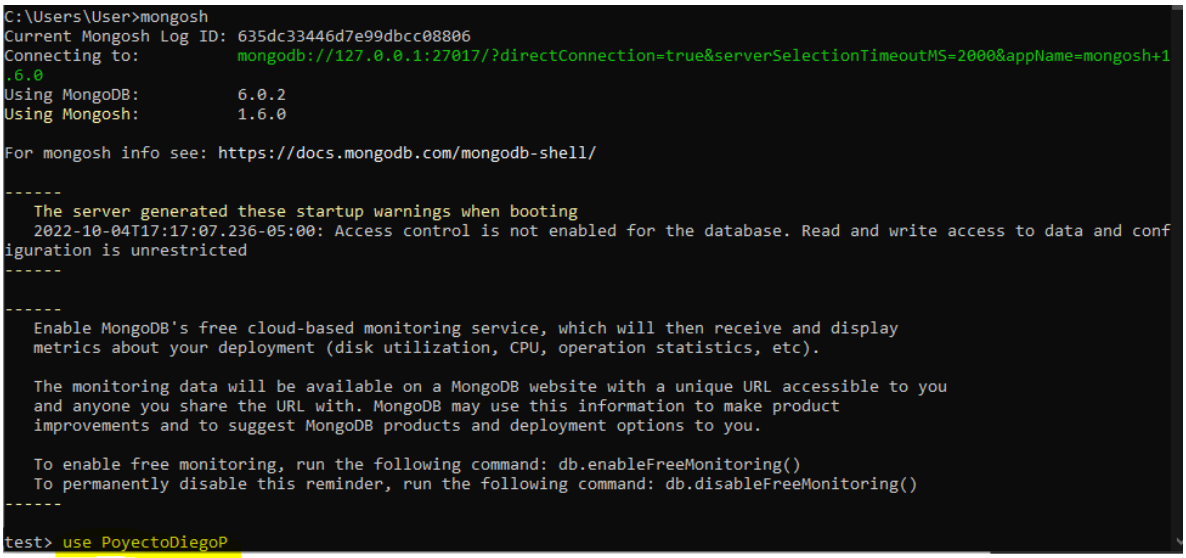


Figura 10.Entrando a base de datos en Mongo.

Para acceder a la visualización de los datos almacenados se dispone a realizar el comando db.datosNodo.find()

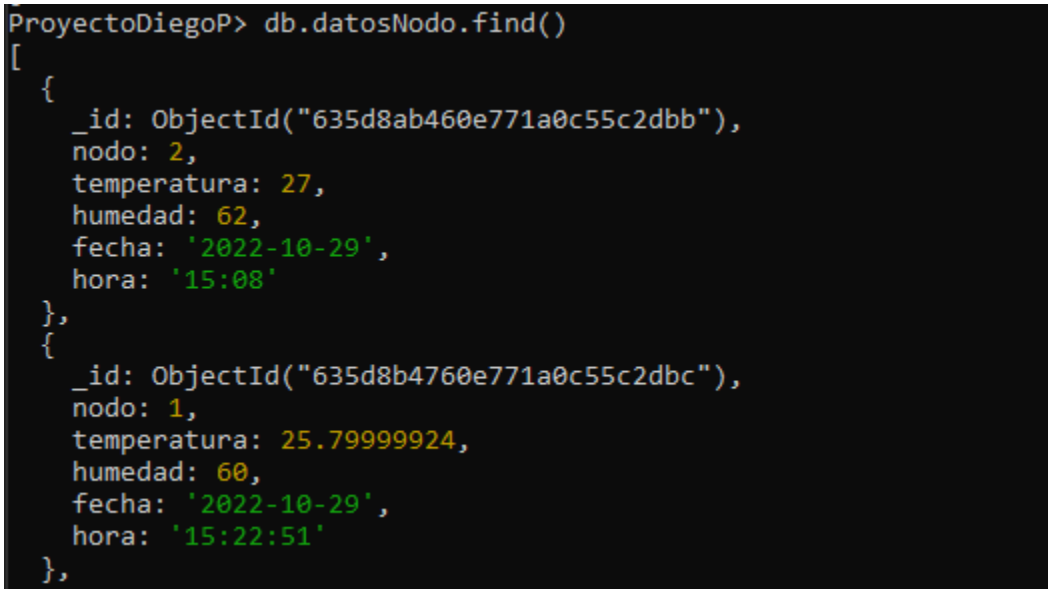


Figura 11.Datos almacenados en la base de datos PoyectoDiegoP en Mongo.

En la anterior figura 11 se puede observar que todos los datos del nodo 2 que se realizó con MQTTLens ha sido almacenados y del nodo 1 que se realizó con ESP32 ha sido almacenados todos utilizando el protocolo de MQTT , por lo que ha sido satisfactorio el envío de los datos y almacenaje de ellos en la base de datos.

Referencias

[1]“La Plataforma de Datos Para Aplicaciones,” MongoDB. [Online]. Available: https://www.mongodb.com/es.