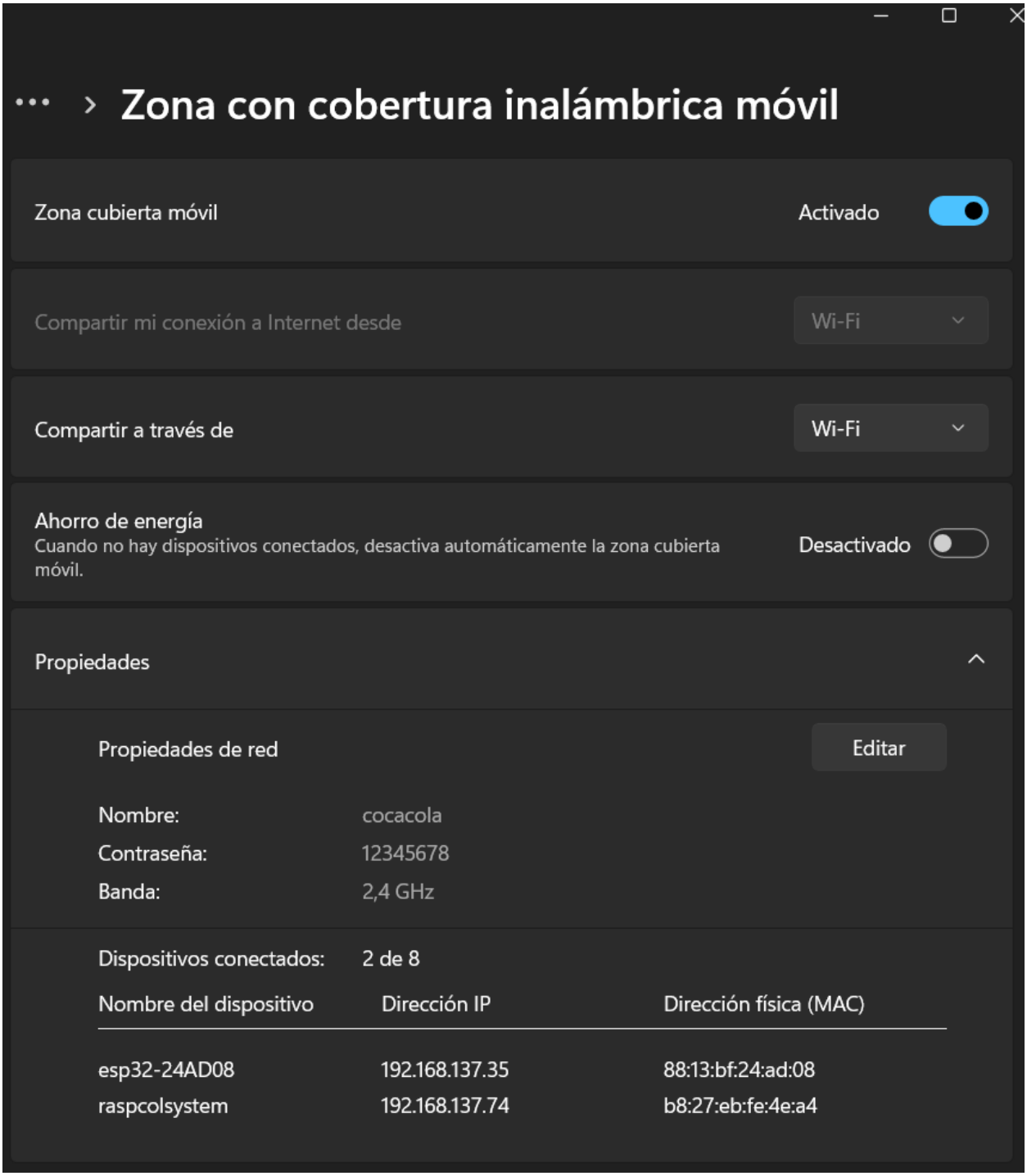
# RASPCOLSYSTEM
# SENSOR DE TEMPERATURA Y HUMEDAD

ANDRES FELIPE TORRES MORENO
JUAN JOSÉ MANJÁRRES COCA
JUAN JOSÉ SALGADO PEREZ
JUAN SEBASTIAN LENIS ACOSTA
SEBASTIÁN JARAMILLO GARCÍA

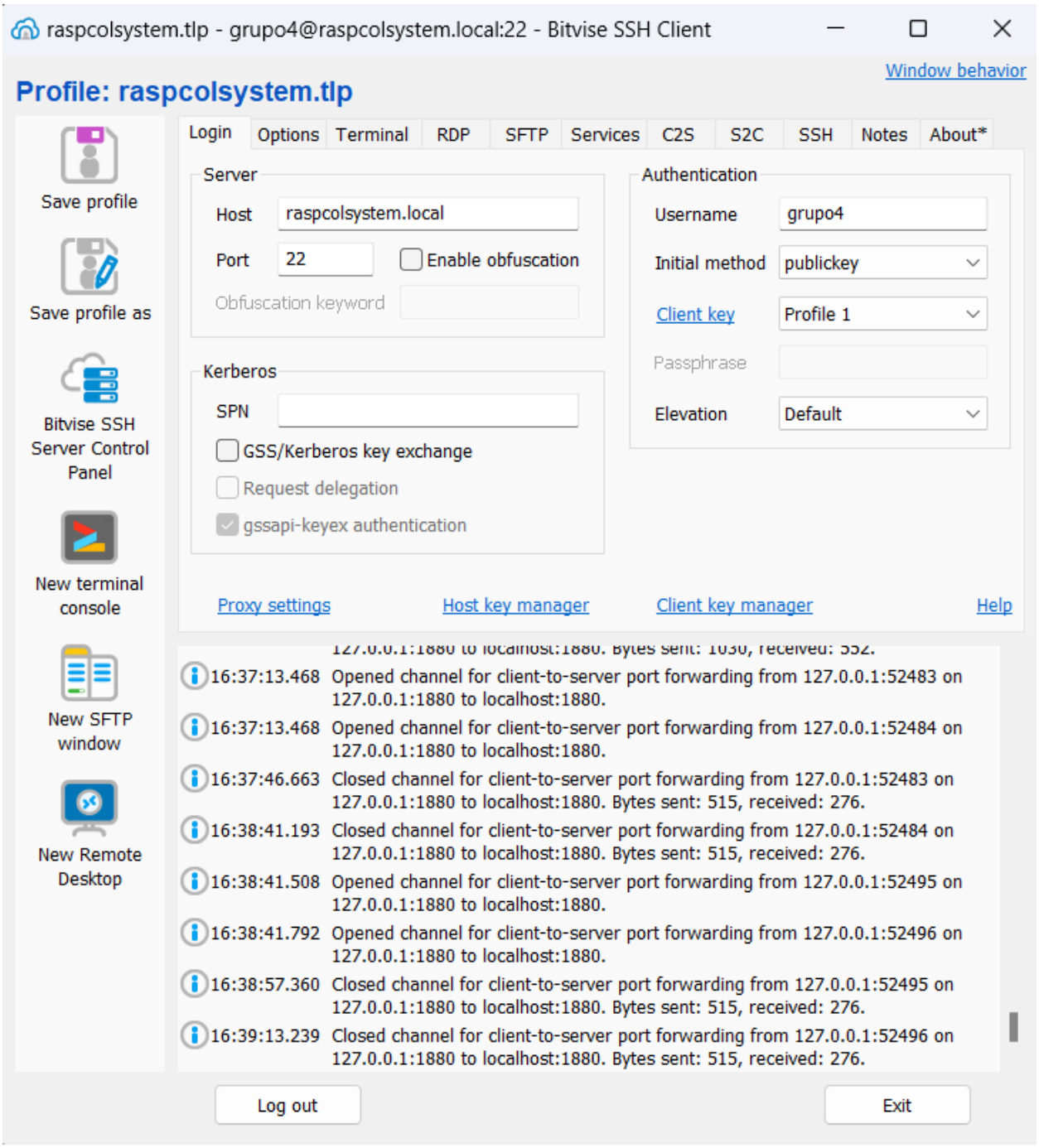
UNIDAD CENTRAL DEL VALLE DEL CAUCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
TULUÁ - VALLE DEL CAUCA
2024


2024

descarga de la imagen del raspberry y con la configuración se ve ahi y para la conexion de este dispositivo con el equipo se abre una zona portatil poniendo la siguiente configuración




Ahí se ve la conexión del Raspberry + el esp32 pero lo que más importa es la conexión del Raspberry ya que con la dirección IP o el nombre puesto en la imagen se va a ingresar a este por medio del Bitvise.




Para el Bitvise se pone el nombre o IP del dispositivo y el usuario creado en la imagen y siendo la primera vez se ingresa con la contraseña con el usuario
después entramos a la terminal y creamos y configuramos la llave publica para dejar de usar la contraseña ya que así lo vuelve más seguro.


Para la configuración de la Raspberry solamente se le va instalar el servicio de node-red y el Mosquitto que es lo que principalmente vamos a utilizar para hacer el empalme con el sensor y el esp32, después de terminar con la configuración del Raspberry

continuaremos con la ejecución y creación del código para la esp32 que es donde vamos a recibir los datos que nos envié el sensor pm7003.

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

// Configuración del sensor PMS
HardwareSerial mySerial(1); // UART1 para el sensor
PMS pms(mySerial);
PMS::DATA data;

// Configuración Wi-Fi
const char* ssid = "cocacola";           // Cambia a tu red Wi-Fi
const char* password = "12345678";   // Cambia a tu contraseña

// Configuración del broker MQTT
const char* mqtt_server = "192.168.137.74"; // IP de tu Raspberry Pi
const int mqtt_port = 1883;              // Puerto del broker MQTT (por defecto es 1883)
const char* mqtt_user = "admin";        // Usuario MQTT (opcional)
const char* mqtt_password = "a1b2c3d4"; // Contraseña MQTT (opcional)
```

Aquí se hace la configuración del esp32 y el Raspberry para conectarse entre si para recibir y almacenar los datos que se irán enviando.

```
// Tópico MQTT
const char* topic_pm = "sensor/pm";

// Cliente Wi-Fi y MQTT
WiFiClient espClient;
PubSubClient client(espClient);

// Función para conectar al Wi-Fi
void setupWiFi() {
    delay(10);
    Serial.println("Conectando a Wi-Fi...");
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.print(".");
    }
    Serial.println("\nWi-Fi conectado.");
    Serial.println(WiFi.localIP());
}
```

verificación de la conexión del esp32.

—----------

```
// Función para conectar al broker MQTT
void reconnectMQTT() {
    while (!client.connected()) {
        Serial.println("Reconectando al broker MQTT...");
        if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
            Serial.println("Conectado al broker MQTT.");
        } else {
            Serial.println("Error de conexión MQTT. Reintentando...");
            delay(5000);
        }
    }
}
```

Función para conectar el Raspberry con esp32.

```
void setup() {
    Serial.begin(115200);
    mySerial.begin(9600, SERIAL_8N1, 16, 17); // RX = 16, TX = 17
    pms.passiveMode(); // Configura el sensor en modo pasivo

    setupWiFi();
    client.setServer(mqtt_server, mqtt_port); // Configura el cliente MQTT
}
```

verificación y conexión física del sensor y esp32 en la Protoboard y función para su correcto funcionamiento en el código.

```
void loop() {
    if (!client.connected()) {
        reconnectMQTT();
    }
    client.loop();
```

pequeña función para la conexión del cliente mqtt-mosquitto (raspberry).

---

```
    // Solicitar y leer datos del sensor
    pms.wakeUp();
    delay(3000);
    pms.requestRead();
    if (pms.readUntil(data)) {
        // Crear un JSON con los datos
        char payload[128];
       if (data.PM_AE_UG_1_0 >= 0 && data.PM_AE_UG_2_5 >= 0 && data.PM_AE_UG_10_0 >= 0) {
    snprintf(payload, sizeof(payload),
         "{\"PM1_0\":%d,\"PM2_5\":%d,\"PM10\":%d}",
         data.PM_AE_UG_1_0, data.PM_AE_UG_2_5, data.PM_AE_UG_10_0);

    if (client.publish(topic_pm, payload)) {
        Serial.println("Datos enviados al tópico MQTT:");
        Serial.println(payload);
    } else {
        Serial.println("Error al enviar datos al broker MQTT.");
    }
} else {
    Serial.println("Datos inválidos leídos del sensor.");
}

        // Publicar datos en el tópico MQTT
        if (client.publish(topic_pm, payload)) {
            Serial.println("Datos enviados al tópico MQTT:");
            Serial.println(payload);
        } else {
            Serial.println("Error al enviar datos al broker MQTT.");
        }
    }
    pms.sleep();
    delay(10000); // Esperar 10 segundos antes de la siguiente lectura
}
```

Aquí se reciben los datos y posteriormente se mandan al Node-red para ser imprimidos en un depurador siendo 10seg el Coldown para cada actualización de los datos.