

ACTIVIDAD 3. Protocolos de transmisión

Inicio

El objetivo de este taller es entender el funcionamiento y la configuración de los protocolos de transmisión tanto de capa de enlace como de aplicación en un objeto IoT.

El protocolo de capa de enlace será WiFi, ya que la plataforma hardware que estamos usando ya tiene incluido el módulo para este protocolo.

El protocolo de capa de aplicación será MQTT el cual probaremos usando un bróker simple y una herramienta de suscripción y publicación como MQTT Explorer.

Configuración del WiFi del ESP32

La conexión del ESP32 a la red WiFi es posible gracias a la librería WiFi que forma parte de la definición del ESP32 que se descarga con el gestor de placas al configurar el IDE de Arduino.

En el código del ESP32 añadimos lo siguiente:

```
#include <WiFi.h>

const char* ssid = "xxxxx"; //ssid de la red inalambrica
const char* password = "xxxxxxxx"; //password para conectarse a la red
```

Y creamos la siguiente función:

```
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());
}
```

Y en el setup se llama la función:

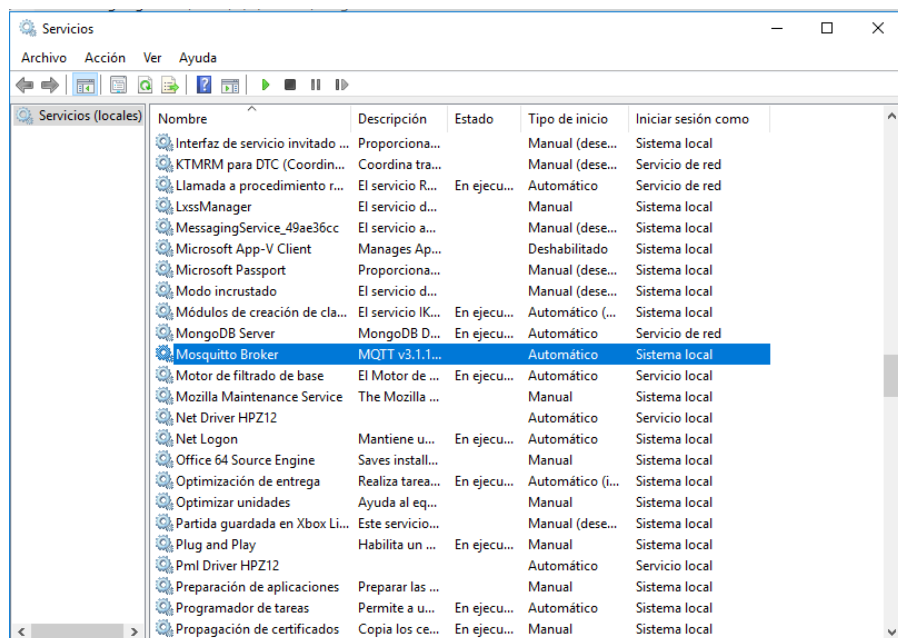
```
void setup()
{
    ...
    setup_wifi(); //WiFi connection
    ...
}
```

Configuración de la conexión al protocolo MQTT en el ESP32

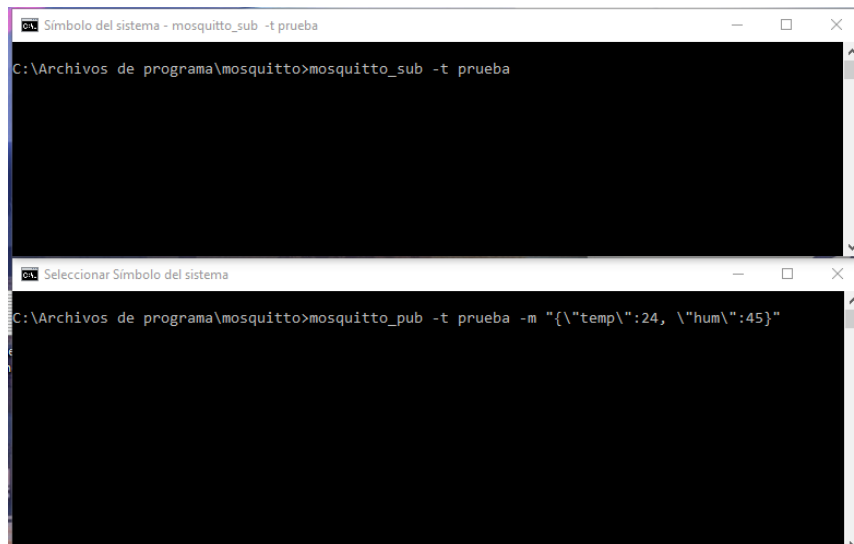
Para esto nos conectaremos a un broker público que no requiere cuentas ni registro de usuarios. La URL de este broker es: broker.mqtt-dashboard.com.

Podríamos también usar un broker local: mosquitto. Para ello seguimos los siguientes pasos:

Descargar mosquitto para Windows, del siguiente enlace: <https://mosquitto.org/download/>, instalarlo e iniciar el servicio, lo cual se puede hacer desde la ventana de Servicios de Windows.



Cientes Mosquitto. Abrir dos ventanas de comandos de Windows, una para ejecutar un publicador de mosquitto y la otra para ejecutar un subscriber.

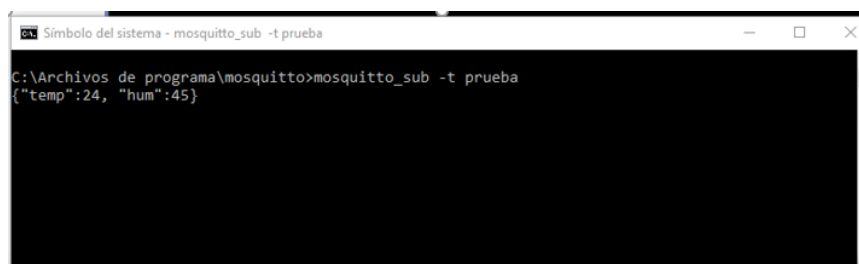


```
Símbolo del sistema - mosquitto_sub -t prueba
C:\Archivos de programa\mosquitto>mosquitto_sub -t prueba

Selecciónar Símbolo del sistema
C:\Archivos de programa\mosquitto>mosquitto_pub -t prueba -m "{\"temp\":24, \"hum\":45}"
```

El comando `mosquitto_sub` lanza un subscriptor el cual se suscribe al tópico `prueba`.

El comando `mosquitto_pub` lanza un publicador, que publicará un mensaje en el tópico `prueba`. Al ejecutar el publicador el mensaje aparecerá en la ventana del subscriptor



```
Símbolo del sistema - mosquitto_sub -t prueba
C:\Archivos de programa\mosquitto>mosquitto_sub -t prueba
{"temp":24, "hum":45}
```

Estos clientes también me permiten conectarnos con el broker público:

```
C:\Archivos de programa\mosquitto>mosquitto_pub -t zeida/prueba -h broker.mqtt-dashboard.com -m "mensaje de prueba"
C:\Archivos de programa\mosquitto>
```

```
C:\Archivos de programa\mosquitto>mosquitto_sub -t zeida/prueba -h broker.mqtt-dashboard.com
mensaje de prueba
```

Como publicadores y subscriptores también podemos usar la aplicación MQTT Explorer. La información de esta herramienta la podemos obtener de su página oficial: <https://mqtt-explorer.com/>

Plataforma hardware (ESP32). En nuestro hardware (plataforma+ sensores) incluir las librerías y el código que permite la conexión al bróker.

Para la conexión al bróker se debe usar la librería `pubsubclient`. Esta debe ser incluida en el IDE de Arduino.

A continuación, se muestran un sketch de ejemplo para publicar datos en el bróker en un tópico denominado **proyecto/topico1**

```
#include <ArduinoJson.h>
```

```

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

#define mqttUser ""
#define mqttPass ""
#define mqttPort 1883
#define Led 2

const char* ssid = "xxxx"; //ssid de la red inalambrica
const char* password = "xxxxxx"; //password para conectarse a la red

char mqttBroker[] = "broker.mqtt-dashboard.com"; //ip del servidor
char mqttClientId[] = "device1"; //cualquier nombre
char inTopic[] = "proyecto/topico1";

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

    String json=String((char *)payload);
    Serial.println();
    StaticJsonDocument<300> doc;
    DeserializationError error = deserializeJson(doc, json);
    if (error) { return; }
    int estado = doc["estado"];

    Serial.println(estado);
    if (estado == 1) {
        digitalWrite(Led, HIGH);
    }
    else {
        digitalWrite(Led, LOW);
    }
}

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(mqttClientId, mqttUser, mqttPass)) {
            Serial.println("connected");
            client.subscribe("topico2");
        } 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()
{
    pinMode(Led, OUTPUT);
    Serial.begin(115200); //Serial connection
    setup_wifi(); //WiFi connection
    client.setServer( mqttBroker, mqttPort );
    client.setCallback( callback );
    Serial.println("Setup done");
    delay(1500);
}

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());
}

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

    //aquí va el código asociado al sensor y generación del json

    int lon = json.length()+1;

```

```

char datojson[lon];
json.toCharArray(datojson, lon);
client.publish(inTopic,datojson);
delay (5000);
}

```

Configure los suscriptores para que cuando se ejecute el sketch reciban los datos publicados.

Anexo:

Adjunto el código completo con la configuración del sensor MPU6050

```

#include <ArduinoJson.h>
#include <PubSubClient.h>
#include <WiFi.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>

#define mqttUser ""
#define mqttPass ""
#define mqttPort 1883
#define Led 2

Adafruit_MPU6050 mpu;

const char* ssid = "zeida"; //ssid de la red inalambrica
const char* password = "12345678"; //password para conectarse a la red

char mqttBroker[] = "broker.mqtt-dashboard.com"; //ip del servidor
char mqttClientId[] = "device1"; //cualquier nombre
char inTopic[] = "proyecto/topico1";

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

    String json=String((char *)payload);
    Serial.println();
    StaticJsonDocument<300> doc;
    DeserializationError error = deserializeJson(doc, json);
    if (error) { return; }
    int estado = doc["estado"];

    Serial.println(estado);
    if (estado == 1) {
        digitalWrite(Led, HIGH);
    }
    else {
        digitalWrite(Led, LOW);
    }
}

```

```

}

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(mqttClientId, mqttUser, mqttPass)) {
            Serial.println("connected");
            client.subscribe("topico2");
        } 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()
{
    pinMode(Led, OUTPUT);
    Serial.begin(115200); //Serial connection
    setup_wifi(); //WiFi connection
    client.setServer( mqttBroker, mqttPort );
    client.setCallback( callback );
    Serial.println("Setup done");
    delay(1500);

    Serial.println("Adafruit MPU6050 test!");
    // Try to initialize!
    if (!mpu.begin()) {
        Serial.println("Failed to find MPU6050 chip");
        while (1) {
            delay(10);
        }
    }
}
Serial.println("MPU6050 Found!");

mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
Serial.print("Accelerometer range set to: ");
switch (mpu.getAccelerometerRange()) {
case MPU6050_RANGE_2_G:

```

```

    Serial.println("+2G");
    break;
case MPU6050_RANGE_4_G:
    Serial.println("+4G");
    break;
case MPU6050_RANGE_8_G:
    Serial.println("+8G");
    break;
case MPU6050_RANGE_16_G:
    Serial.println("+16G");
    break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
    Serial.println("+ 250 deg/s");
    break;
case MPU6050_RANGE_500_DEG:
    Serial.println("+ 500 deg/s");
    break;
case MPU6050_RANGE_1000_DEG:
    Serial.println("+ 1000 deg/s");
    break;
case MPU6050_RANGE_2000_DEG:
    Serial.println("+ 2000 deg/s");
    break;
}

mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");

```



```

        break;
    case MPU6050_BAND_5_HZ:
        Serial.println("5 Hz");
        break;
    }

    Serial.println("");
    delay(100);
}

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());
}

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

    //aquí va el código asociado al sensor y generación del json
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    String json;
    StaticJsonDocument<300> doc;

    doc["accx"] = a.acceleration.x;
    doc["accy"] = a.acceleration.y;
    doc["accz"] = a.acceleration.z;
    doc["rotx"] = g.gyro.x;

```

```
doc["roty"] = g.gyro.y;
doc["rotz"] = g.gyro.z;
doc["temp"] = temp.temperature;

serializeJson(doc, json);
Serial.println(json);
int lon = json.length()+1;
char datojson[lon];
json.toCharArray(datojson, lon);
client.publish(inTopic,datojson);
delay (5000);
}
```