Funciones prototipo

Procedemos a explicar el prototipo realizado. El prototipo consta de 2 ESP32 con sensores y actuadores, un servidor MQTT y un ordenador con Node-Red.

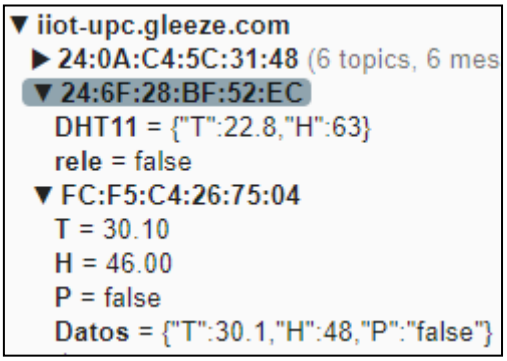
La 1era ESP (FC:F5:C4:26:75:04)simula un camión y una vaca. Como vaca envía la información de la temperatura. Como camión, dispone de un botón que se utiliza para pedir abrir la puerta.

La 2da ESP (24:6F:28:BF:52:EC) simula las oficinas y la fábrica. Como oficinas envía datos de temperatura y humedad. Como fábrica tiene el control de apertura de la puerta. Para abrir la puerta, es necesario que el camión pulse el botón de petición de apertura de puerta y que el sensor de presencia de camión esté activo. Cuando se dan estas dos condiciones, la puerta se abre. Además tiene una señal luminosa que cambia de color en función de la situación:

- Rojo: No se ha detectado nada con el sensor de presencia.
- Azul: se ha detectado presencia de camión.
- Verde: El camión ha pulsado el botón para acceder a la fábrica.

El servidor de MQTT, se utiliza para publicar y subscribir los topics de los distintos dispositivos.

El ordenador con Node-Red se encarga de la comunicación entre los dispositivos.



Descripción Topics:

- T: temperatura.
- H: humedad.
- P: pulsador petición apertura puerta.
- Rele: se utiliza para recibir la petición de apertura de puerta.

Ejemplo de código en ESP:

El código contiene la conexión a redes Wi-Fi, bróker MQTT, lectura de sensores y publicación de datos en json.

Ejemplo de lectura de la temperatura.

```
// DHT temperature update
void updatetemperature() {
  float t = dhtSensor.readTemperature(); // Updates temperature
  if(!isnan(t)){
    temperature = t; // Updates the value to global variable
    Serial.println("Updated temperature: " + String(temperature) + "°C"); // Prints in a new line the result
  } else {
    Serial.println("Sensor had a bad reading");
  }
}
```

Ejemplo archivo json:

```
// Datos del DHT y del pulsador en JSON
void publishDatos() {
  static const String topicStr = createTopic("Datos");
  static const char *topic = topicStr.c_str();

  StaticJsonDocument<128> doc; // Create JSON document of 128 bytes
  char buffer[128]; // Create the buffer where we will print the JSON document
                    // to publish through MQTT
  doc["T"] = temperature;
  doc["H"] = humidity;
  doc["P"] = pulsador;

  // Serialize the JSON document to a buffer in order to publish it
  serializeJson(doc, buffer);
  mqttClient.publish(topic, buffer, RETAINED);
  Serial.println(" <= " + String(topic) + ": " + String(buffer));
}
String createTopic(char *topic) {
  String topicStr = String(macAddress) + "/" + topic;
  return topicStr;
}
```

Ejemplo subscribir a un tópico:

```
void callback(char *topic, byte *payload, unsigned int length) {
  // Register all subscription topics
  static const String cmdTopicStr = createTopic(COMMAND_TOPIC);
  static const String testTopicStr = createTopic(TEST_TOPIC);

  String msg = unwrapMessage(payload, length);
  Serial.println(" => " + String(topic) + ": " + msg);

  // What to do in each topic case?
  if (String(topic) == cmdTopicStr) {
    if (msg== "true" and Pres == HIGH){
      digitalWrite(Motor,HIGH);
      digitalWrite(LED_VERDE,HIGH);
      digitalWrite(LED_ROJO,LOW);
      digitalWrite(LED_AZUL,LOW);
      delay(2000);
      Pres=LOW;
      digitalWrite(Motor,LOW);
      }else if (msg=="false"){
        digitalWrite(Motor,LOW);
        Pres=LOW;
        }
  } else if (String(topic) == testTopicStr) {
    // Do some other stuff
  } else {
    Serial.println("[WARN] - '" + String(topic) +
                   "' topic was correctly subscribed but not defined in the "
                   "callback function");
  }
}
```

Esquemas prototipo

Para los esquemas de conexionado se ha utilizado Tinkercard, por lo que en vez de una ESP32 se utiliza un Arduino UNO y en vez del sensor DHT11, se utiliza el sensor de temperatura de la herramienta.