

ÍNDICE GENERAL

CAPÍTULO 1	INTRODUCCIÓN	5
CAPÍTULO 2	OBJETIVOS	7
2.1	OBJETIVOS TEÓRICOS.....	7
2.2	OBJETIVOS TÉCNICOS	7
CAPÍTULO 3	REQUISITOS	8
3.1	REQUISITOS FUNCIONALES.....	8
3.2	REQUISITOS NO FUNCIONALES.....	9
3.3	REQUISITOS DE INFORMACIÓN.....	10
CAPÍTULO 4	DISEÑO	11
4.1	ESQUEMA GENERAL	11
4.2	DISEÑO DEL SERVIDOR.....	12
4.3	DISEÑO DE LA OFICINA.....	14
4.3.1	<i>Climatización automática con MQTT</i>	<i>15</i>
4.3.2	<i>Comunicación UDP luces.....</i>	<i>17</i>
4.3.3	<i>Comunicación UDP puertas</i>	<i>18</i>
4.4	DISEÑO DEL RECINTO DE PISTAS	20
4.4.1	<i>Diseño de luces</i>	<i>20</i>
4.4.2	<i>Diseño de las puertas.....</i>	<i>25</i>
CAPÍTULO 5	PRUEBAS	28
5.1	ACCESO SERVIDOR IOT	28
5.2	CONTROL DE LUCES CON POTENCIÓMETROS	29
5.3	CONTROL DE PUERTAS CON INTERRUPTORES.....	31

5.4	CLIMATIZACIÓN AUTOMÁTICA.....	32
BIBLIOGRAFÍA.....		35

ÍNDICE DE FIGURAS

FIGURA 1: ENTORNOS DEL SISTEMA IoT	11
FIGURA 2: SERVIDOR IoT REMOTO	12
FIGURA 3: VISTA EXTERNA SISTEMA IoT	12
FIGURA 4: SERVICIO DHCP	13
FIGURA 5: CREANDO CUENTA SERVIDOR IoT.....	13
FIGURA 6: CREDENCIALES SERVIDOR IoT	14
FIGURA 7: DISEÑO OFICINA	14
FIGURA 8: PROYECTO MQTT BROKER.....	15
FIGURA 9: PROYECTO MQTT CLIENT	15
FIGURA 10: POTENCIÓMETROS SISTEMA IoT.....	17
FIGURA 11: INTERRUPTORES SISTEMA IoT.....	19
FIGURA 12: RECINTO DE PISTAS.....	20
FIGURA 13: LUCES PT-IOT-NM-1W	22
FIGURA 14: CONEXIÓN A HOME GATEWAY DE LUCES.....	23
FIGURA 15: LUCES CONEXIÓN IoT SERVER REMOTO	24
FIGURA 16: DEVICE CONDITIONS LUCES	25
FIGURA 17: NETWORK ADAPTER PUERTA	27
FIGURA 18: CONEXIÓN REMOTE SERVER PUERTAS	27
FIGURA 19: ACCESO A SERVIDOR REMOTO DESDE SMARTPHONE	28
FIGURA 20: LISTA DISPOSITIVOS IoT SERVER	29
FIGURA 21: POTENCIÓMETROS 1 Y 3 ACTIVADOS	29
FIGURA 22: OUTPUTS MCUs	30
FIGURA 23: LUCES PISTAS 3 Y 1 ENCENDIDAS.....	31
FIGURA 24: INTERRUPTORES 2 Y 4 ACTIVADOS	31

FIGURA 25: OUTPUTS MCUs PUERTAS	32
FIGURA 26: PUERTAS 2 Y 4 ABIERTAS	32
FIGURA 27: SBC CLIMATIZACIÓN	33
FIGURA 28: CALEFACCIÓN ACTIVADA	33
FIGURA 29: TEMPERATURA PUBLICADA	34

Capítulo 1

INTRODUCCIÓN

En los últimos años, el Internet de las Cosas ha revolucionado la manera en que interactuamos con nuestro entorno, permitiendo la automatización y el control remoto de dispositivos de forma eficiente. Así como la recolección de datos de forma masiva. Gracias a herramientas como Cisco Packet Tracer, es posible simular y probar soluciones IoT antes de su implementación en entornos reales.

En este proyecto, se ha desarrollado la simulación de un club de pádel inteligente en Packet Tracer, integrando múltiples dispositivos IoT para mejorar la gestión y el control del recinto. En las pistas encontramos dos elementos principales que debemos controlar, las puertas y las luces. A parte, se tendrá en cuenta una pequeña oficina del club que debe presentar buenas condiciones para que el encargado del recinto pueda tener un control adecuado de las pistas.

Otro punto muy que también resultaría muy interesante es poder interactuar con los elementos de las pistas y la oficina de forma online. Así cualquier problema en el que no se encontrase un trabajador de forma presencial en el recinto se podría solucionar desde el teléfono móvil de cualquier encargado.

Se han utilizado MCUs con sockets para permitir la gestión remota de luces y puertas de las pistas desde la oficina, proporcionando una automatización eficiente y centralizada. Además, todos los dispositivos han sido conectados a una red IoT remota, lo que permite su acceso y control desde cualquier ubicación, facilitando la administración del club de manera flexible y segura.

Otro aspecto clave de la simulación es la implementación de un servidor MQTT en la oficina, encargado de gestionar la climatización mediante SBCs. Esto permite una regulación inteligente de la temperatura, optimizando el consumo energético y mejorando el confort dentro del edificio.

Capítulo 2

OBJETIVOS

A continuación, se detallan los objetivos teóricos y técnicos que guían el desarrollo de este sistema IoT.

2.1 Objetivos Teóricos

Los objetivos del proyecto son los siguientes:

- Control remoto y presencial de las luces de las pistas.
- Control remoto y presencial de las puertas de las pistas.
- Climatización automática de la oficina.

2.2 Objetivos Técnicos

A nivel técnico se pretende implementar todas las tecnologías estudiadas en las prácticas de la asignatura.

- Comunicación UDP con múltiples destinos para controlar luces mediante potenciómetros.
- Comunicación UDP con múltiples destinos para controlar puertas mediante interruptores.
- Control externo de todos los elementos inteligentes de las pistas y oficina a través de servidor IoT remoto.
- Climatización automática de la oficina mediante MQTT.

Capítulo 3

REQUISITOS

En esta sección se procederá a detallar técnicamente lo que el sistema IoT a desarrollar deberá realizar. El objetivo es definir claramente la información que el sistema manipulará y las operaciones que realizará.

Dividiré los requisitos en tres categorías principales:

Requisitos funcionales: Estos describen las interacciones específicas entre el sistema y su entorno, detallando las funciones que el software debe ejecutar. Por ejemplo, acciones que el usuario espera que el sistema realice en respuesta a entradas específicas, como la creación de cuentas de usuario, la visualización de astros y la gestión de logros.

Requisitos no funcionales: Estos requisitos especifican criterios que pueden ser usados para juzgar la operación del sistema, en lugar de comportamientos específicos. Incluyen atributos como la seguridad, la usabilidad, la accesibilidad, y el rendimiento, entre otros.

Requisitos de la información: Se refieren a los tipos de información que el software debe manejar, cómo se almacena, accede y actualiza. Esto incluye la gestión de datos de los astros, los datos personales de los usuarios, y el seguimiento de los logros alcanzados por los usuarios dentro de la aplicación.

3.1 Requisitos Funcionales

- RF1. El sistema permitirá el control de la iluminación de las pistas mediante potenciómetros conectados a MCUs.

- RF2. El sistema permitirá el control de las puertas de las pistas mediante interruptores conectados a MCUs.
- RF3. Los microcontroladores estarán conectados a un hub central, el cual gestionará la comunicación con las luces y puertas a través de UDP.
- RF4. El usuario podrá acceder y controlar los dispositivos IoT de las pistas y la oficina a través de un servidor remoto.
- RF5. La red IoT permitirá el control de dispositivos a través de una conexión externa, asegurando acceso desde cualquier ubicación.
- RF6. El sistema contará con un servidor DHCP que asignará direcciones IP a los dispositivos IoT conectados.
- RF7. El sistema implementará un protocolo MQTT para la gestión automatizada de la climatización en la oficina.
- RF8. Un sensor de temperatura transmitirá datos periódicamente en el topic `/club1/sensor/temp1`.
- RF9. Una SBC actuará como suscriptor del topic y gestionará el encendido y apagado del aire acondicionado y calefactor según los valores de temperatura.
- RF10. El sistema permitirá la conexión de dispositivos móviles a la red a través de un punto de acceso inalámbrico.
- RF11. La comunicación UDP asegurará que cada MCU reciba y transmita los datos correspondientes para la correcta operación de luces y puertas.
- RF12. El sistema garantizará la recepción y ejecución de comandos en tiempo real para la automatización de los dispositivos.

3.2 Requisitos No Funcionales

- RNF1. La comunicación del sistema deberá tener una latencia mínima para garantizar una respuesta inmediata en el control de dispositivos.
-

- RNF2. La interfaz de usuario del sistema IoT remoto será intuitiva y fácil de usar para facilitar la gestión de los dispositivos.
- RNF3. El sistema deberá ser escalable para permitir la integración de nuevos dispositivos IoT en el futuro sin afectar el rendimiento.
- RNF4. La comunicación entre los dispositivos IoT utilizará protocolos seguros para evitar accesos no autorizados.
- RNF5. El sistema deberá garantizar la estabilidad de la red, evitando caídas o interrupciones en la comunicación entre los dispositivos.
- RNF6. La arquitectura del sistema permitirá la actualización y modificación de la configuración sin afectar la operatividad.
- RNF7. El procesamiento de datos deberá ser eficiente para garantizar un bajo consumo energético en los dispositivos IoT.

3.3 Requisitos de Información

- RI1. Estado de dispositivos de iluminación y puertas: el sistema debe registrar y mostrar información sobre el estado actual de las luces y puertas de cada pista, permitiendo verificar si están activas o inactivas.
 - RI2. Información del servicio MQTT: el servicio MQTT debe proporcionar información en tiempo real sobre la temperatura de la oficina a través del topic `/club1/sensor/temp1`, permitiendo la regulación automática del clima.
 - RI3. Información de la conexión de los dispositivos: el servidor DHCP debe mostrar información sobre las direcciones IP asignadas a cada dispositivo IoT conectado, permitiendo un monitoreo de la red.
-

Capítulo 4

DISEÑO

4.1 Esquema general

En la Figura 1 se presenta una representación esquemática de los principales entornos del sistema IoT implementado. Este sistema se compone de tres elementos clave:

- **Recinto de pistas:** Incluye las áreas donde se ubican las pistas de pádel, equipadas con iluminación y acceso controlado de manera remota o presencial.
- **Oficina:** Se encuentra dentro del recinto de las pistas. Actúa como punto de gestión y supervisión local, donde se encuentran los dispositivos encargados de la climatización automática y la administración de los sistemas de control de las pistas.
- **Servidor:** Permite el control remoto de luces, puertas y climatización a través de la infraestructura IoT.

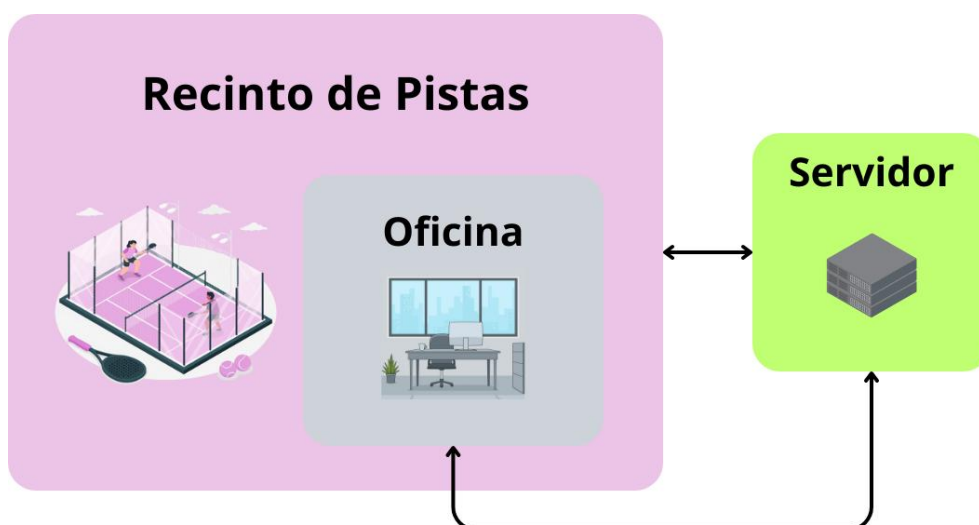


Figura 1: Entornos del sistema IoT

4.2 Diseño del Servidor

En el servidor tenemos simplemente el Server-PT y un switch al que tenemos conectado diferentes puntos de acceso al mismo. Concretamente el HomeGateway de la oficina un un AccessPoint-PT-N que se utilizará para simular un acceso desde un teléfono móvil desde cualquier lugar.

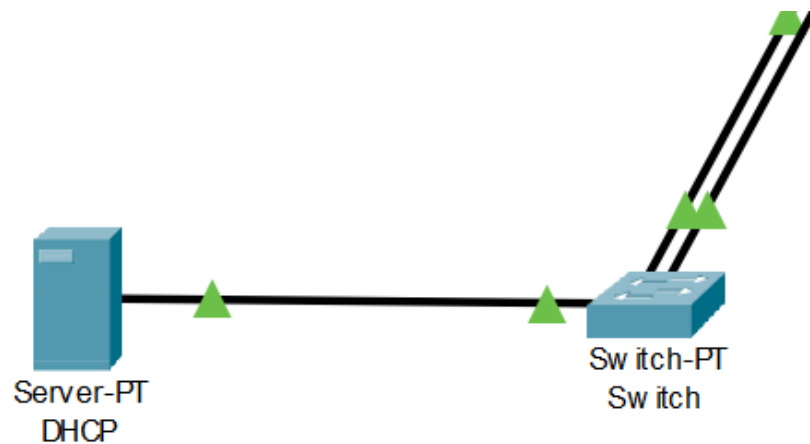


Figura 2: Servidor IoT Remoto

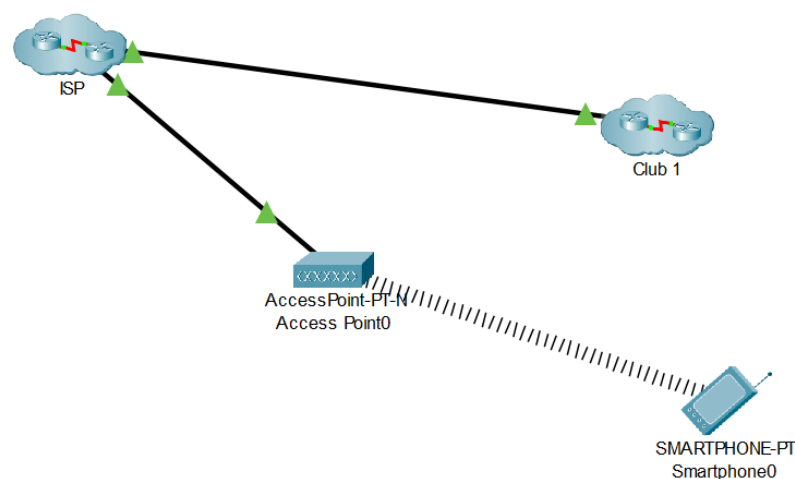


Figura 3: Vista Externa Sistema IoT

Primero, configuraremos el servidor DHCP [1] para asignar direcciones IP dentro del rango 200.100.1.1 hasta 200.100.1.254 a los dispositivos que lo soliciten, con una máscara de 255.255.255.0.

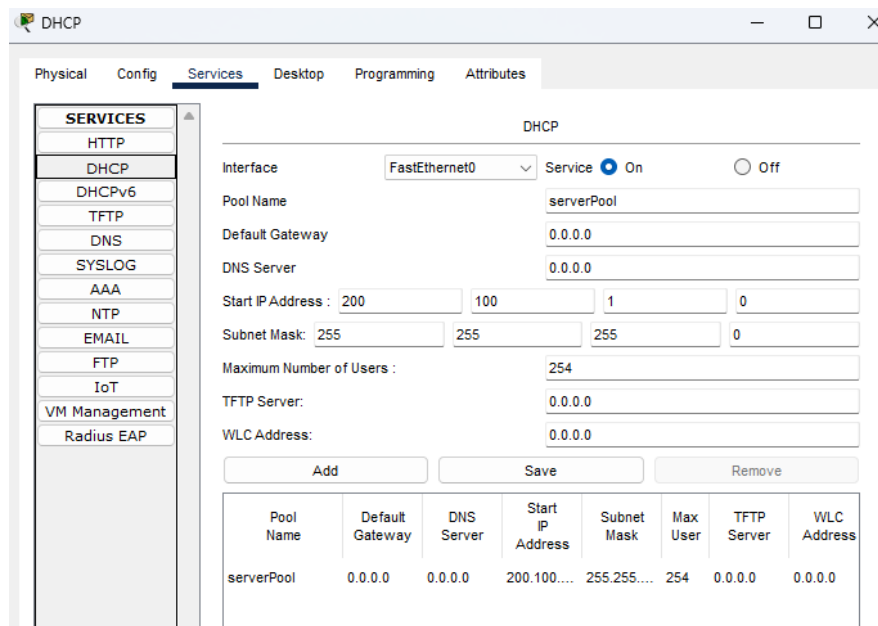


Figura 4: Servicio DHCP

A continuación, accederemos a la URL de 200.100.1.1 para crear la cuenta con la que los diferentes elementos inteligentes podrán conectarse al servidor IoT. Los credenciales serán usuario: club1, contraseña: club1.

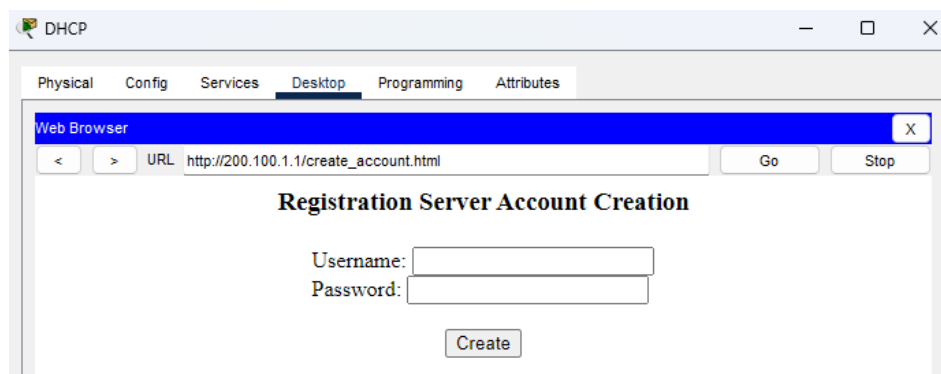


Figura 5: Creando Cuenta Servidor IoT

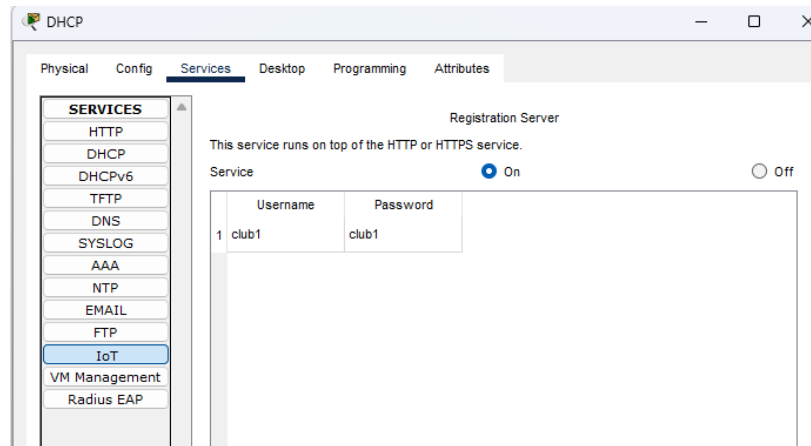


Figura 6: Credenciales Servidor IoT

4.3 Diseño de la Oficina

En la oficina encontramos la mayoría de los elementos del sistema. Encontramos tanto el router del recinto de pistas como los sistemas para controlar luces, puertas y climatización.

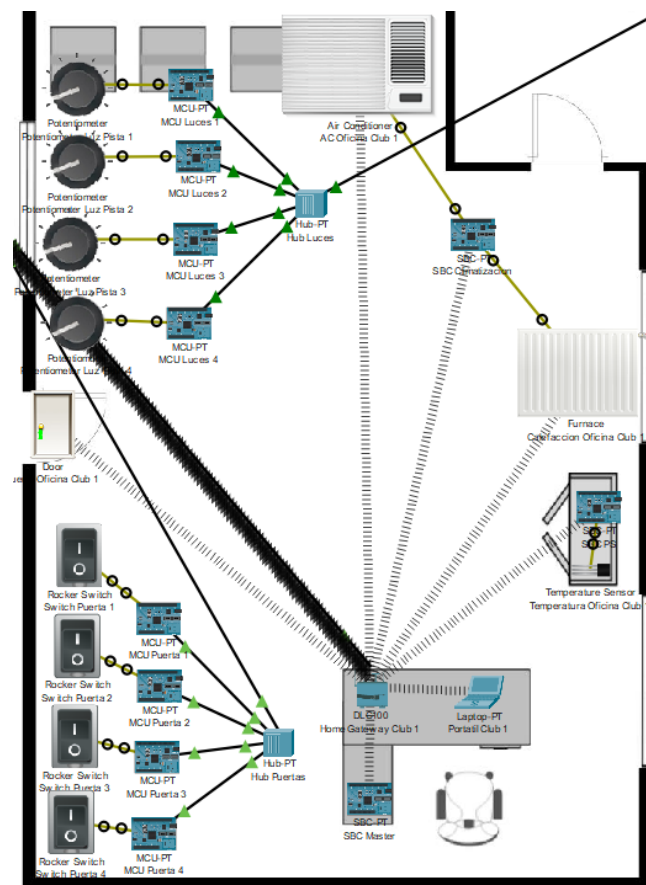


Figura 7: Diseño Oficina

4.3.1 Climatización automática con MQTT

Tenemos un total de 3 SBCs. En dos de ellos se ha instalado MQTT Client y en el restante MQTT Broker [4].

Al bróker ha habido simplemente que instalarle el proyecto de MQTT Broker.

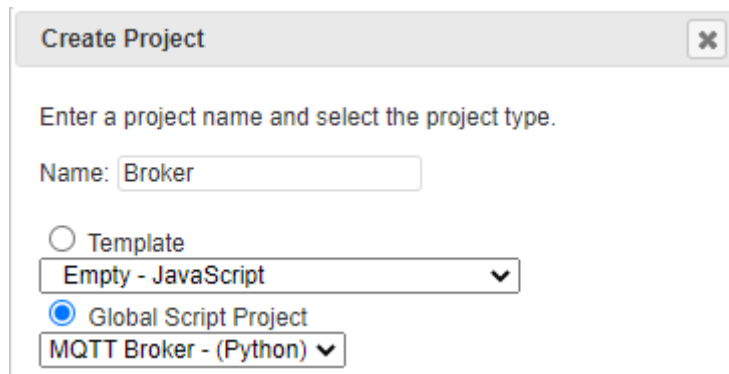


Figura 8: Proyecto MQTT Broker

Uno de los cliente está conectado a un sensor de temperatura y se encarga de recoger las medidas del sensor y publicarlas en el topic /club/sensor/temp1. Para esta tarea, primero se le ha instalado el proyecto MQTT Client.

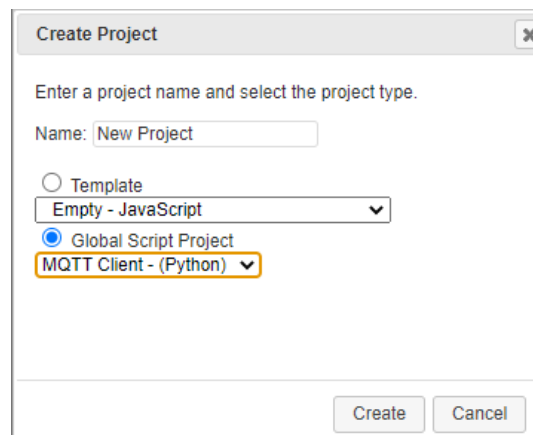


Figura 9: Proyecto MQTT Client

Después se ha añadido el siguiente código al main.py:

```
def publishTemperature():
    value = ( analogRead(0) * (200.0 / 1023.0) ) - 100.0
    topic = "/club1/sensor/temp1"
    payload = str("%.2f"% (value,))
    qos = "0"
```

```

mqttclient.init()
mqttclient.publish(topic, payload, qos)
CLI.exit()

def autoConnect():
    brokerIP = "192.168.25.102"
    user = ""
    password = ""
    mqttclient.init()
    mqttclient.connect(brokerIP, user, password)
    CLI.exit()

def main():
    pinMode(0, IN)

    while True:
        delay(6000)
        publishTemperature()
        autoConnect()

```

El SBC restante es el encargado de mantener un control del aire acondicionado y de la calefacción. A este se la ha añadido el código de autoConnect, pero añadiendo la siguiente línea “mqttclient.subscribe("/club1/sensor/temp1")” para suscribir este dispositivo al topic creado por la otra SBC. Además, para gestionar los mensajes recibidos:

```

def on_message_received(status, msg, packet):
    if status == "Success" or status == "Error":
        print status + ": " + msg
    elif status == "":
        print msg
    # Gestion de temperatura en mensaje recibido
    try:
        message_parts = msg.split(" ")
        if len(message_parts) > 1:
            temperature_str = message_parts[1].split()[0] # Obtener temperatura
            temperature = float(temperature_str)
            if temperature > 25:
                print("Temperatura mayor de 25 C. Activando aire acondicionado.")
                digitalWrite(0, HIGH)
            elif temperature < 25 and temperature > 15:
                print("Temperature entre 25 y 15 grados. Desactivando ac y calefaccion.")
                digitalWrite(0, LOW)
                digitalWrite(1, LOW)
            else:
                print("Temperature menor de 15 grados. Activando calefaccion.")
                digitalWrite(1, HIGH)
        except Exception as e:
            print "Error processing message: " + str(e)
    CLI.exit()

```

De esta forma, si la temperatura es mayor de 25°C, el código activa el aire acondicionado (por medio de la salida digital correspondiente). Si la temperatura está entre 15°C y 25°C, se apagan tanto el aire acondicionado como la calefacción. En el caso de que la temperatura sea inferior a 15°C, se activa la calefacción.

4.3.2 Comunicación UDP luces

La luces reaccionarán a un potenciómetro. Según el valor de este, la luces se encontrarán apagadas o encendidas. A cada potenciómetro le corresponde una pista, es decir, tenemos 4 potenciómetros, uno para cada pista. Será necesario conectar una MCU a todos los potenciómetros para conseguir que se comuniquen con una MCU que se encontrará en las pistas que será la encargada de interactuar con las luces [2].

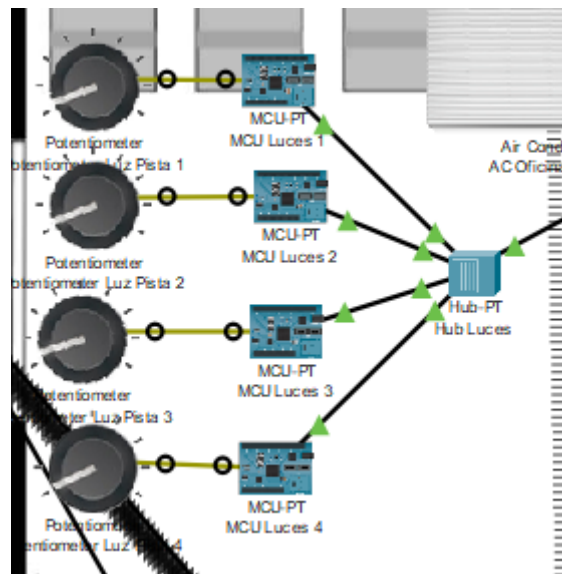


Figura 10: Potenciómetros Sistema IoT

Los MCU contienen el siguiente código:

```
// Luces pista 1
var port = 1234;
var dstIP = "192.168.1.1";
var socket;
var state;

function setup()
{
  pinMode(2, INPUT);
  state = 0;
  socket = new UDPSocket();
  // Recepcion UDP
  socket.onReceive = function(ip, port, data)
  {
    Serial.println("Recibido de "
      + ip + ":" + port + ": " + data);
  };
  // Activa el socket UDP en el puerto
  Serial.println(socket.begin(port));
}

function loop()
```

```
{
  if (analogRead(A0) < 1023 / 3) // Apagada de 0 a 1023 / 3
  {
    socket.send(dstIP, port, "1 0");
    Serial.println("Apagada");
    customWrite(A1, "0");
  }
  else if (analogRead(A0) < 1023 / 3 * 2) // Luz suave de 1023 / 3 a 1023 / 3 * 2
  {
    socket.send(dstIP, port, "1 1");
    Serial.println("Luz Suave");
    customWrite(A1, "1");
  }
  else // Encendida de 1023 / 3 * 2 a 1023
  {
    socket.send(dstIP, port, "1 2");
    Serial.println("Encendida");
    customWrite(A1, "2");
  }
}
```

El código establece comunicación a través del puerto 1234 con la MCU encargada de las luces (192.168.1.1), permitiendo enviar y recibir datos. La MCU lee el valor del potenciómetro y, dependiendo del nivel detectado, envía un mensaje UDP para apagar o encender la luz.

El mensaje que se envía tiene dos números, el primero indica el número de la pista cuyas luces se deben modificar. El segundo número indica el estado de la luz: 0 apagada, 1 suave y 2 encendida. Esta información la usará la MCU que se encuentra en el exterior para saber con qué luces debe interactuar.

4.3.3 Comunicación UDP puertas

Para las puertas se sigue exactamente el mismo diseño que con las luces, con el único cambio de que usamos unos interruptores en vez de los potenciómetros.

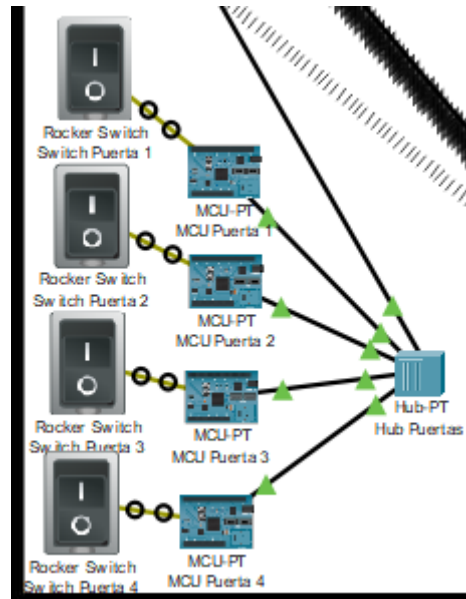


Figura 11: Interruptores Sistema IoT

```
// Puerta pista 1
var port = 1234;
var dstIP = "192.168.1.1";
var socket;
var state;

function setup()
{
  pinMode(2,INPUT);
  state = 0;
  socket = new UDPSocket();
  // Recepcion UDP
  socket.onReceive = function(ip, port, data)
  {
    Serial.println("Recibido de "
      + ip + ":" + port + ": " + data);
  };
  // Activa el socket UDP en el puerto
  Serial.println(socket.begin(port));
}

function loop()
{
  if (digitalRead(0))
  {
    if (state === 0)
    {
      state = 1;
      socket.send(dstIP, port, "1 1");
      Serial.println("PUERTA 1 ON");
    }
  }
  else
  {
    if (state === 1)
    {
      state = 0;
      socket.send(dstIP, port, "1 0");
      Serial.println("PUERTA 1 OFF");
    }
  }
}
```

```
}
}
```

4.4 Diseño del Recinto de Pistas

Por último, en la siguiente figura se muestra el diseño detallado del recinto de pistas.

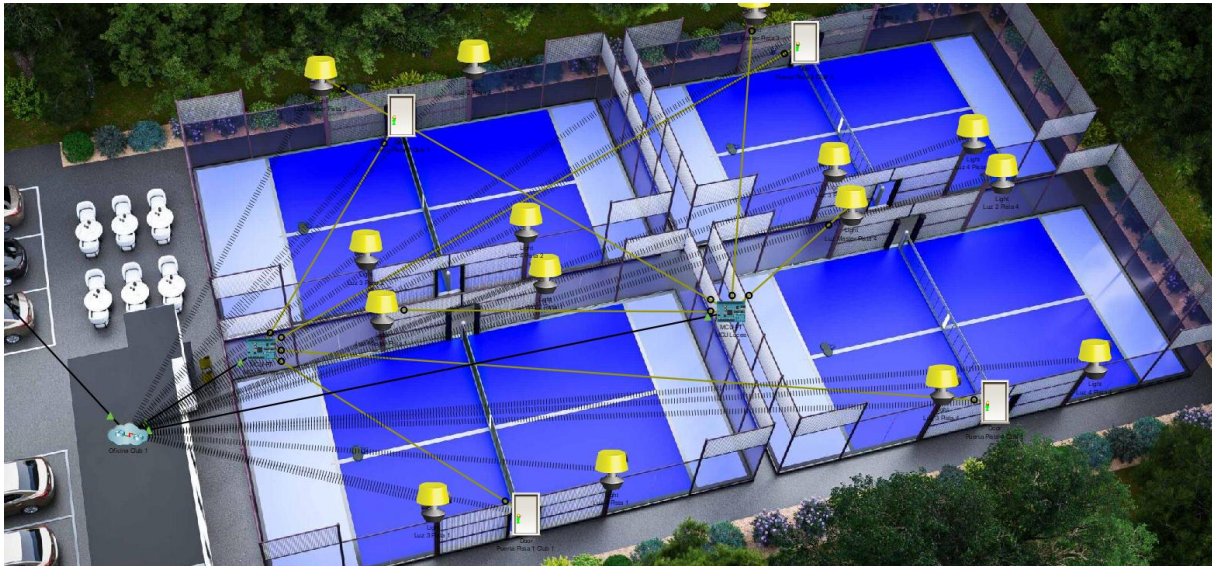


Figura 12: Recinto de pistas

4.4.1 Diseño de luces

4.4.1.1 Acceso presencial a las luces

Para esta tarea tenemos el MCU que recibe mediante UDP los valores de los potenciómetros. Primero, debemos tener en cuenta que en cada pista hay 4 luces. He decidido establecer una luz como *master* en cada pista, de tal forma que todas las demás luces de la misma pista copiarán el estado de la luz master de su pista. De esta forma, el MCU modificará solo el estado de la luz declarada como master que es la que tendrá conexión con cable con la placa.

El cómo se consigue que las demás luces copien el estado de la luz master se explica en el siguiente punto **Acceso remoto a las luces**.

```
var port = 1234;
var socket;

function setup()
{
```

```
socket = new UDPSocket();
customWrite(0,"0");

// Recepcion
socket.onReceive = function(ip, port, data)
{
    Serial.println("Recibido de "
        + ip + ":" + port + ": " + data);
    data = data.split(" ");
    if (data[0] == "1")
    {
        Serial.println("Recibido de "
            + ip + ":" + port + ": " + data);
        if(data[1]=="2")
        {
            Serial.println("LUZ ENCENDIDA");
            customWrite(0,"2");
        }
        else if(data[1]=="1")
        {
            Serial.println("LUZ SUAVE");
            customWrite(0,"0");
        }else
        {
            Serial.println("LUZ APAGADA");
            customWrite(0,"0");
        }
    }
    else if (data[0] == "2")
    {
        Serial.println("Recibido de "
            + ip + ":" + port + ": " + data);
        if(data[1]=="2")
        {
            Serial.println("LUZ ENCENDIDA");
            customWrite(1,"2");
        }
        else if(data[1]=="1")
        {
            Serial.println("LUZ SUAVE");
            customWrite(1,"0");
        }else
        {
            Serial.println("LUZ APAGADA");
            customWrite(1,"0");
        }
    }
    else if (data[0] == "3")
    {
        Serial.println("Recibido de "
            + ip + ":" + port + ": " + data);
        if(data[1]=="2")
        {
            Serial.println("LUZ ENCENDIDA");
            customWrite(2,"2");
        }
        else if(data[1]=="1")
        {
            Serial.println("LUZ SUAVE");
            customWrite(2,"0");
        }else
        {
            Serial.println("LUZ APAGADA");
            customWrite(2,"0");
        }
    }
}
```

```

else if (data[0] == "4")
{
  Serial.println("Recibido de "
+ ip + ":" + port + ": " + data);
  if(data[1]=="2")
  {
    Serial.println("LUZ ENCENDIDA");
    customWrite(3,"2");
  }
  else if(data[1]=="1")
  {
    Serial.println("LUZ SUAVE");
    customWrite(3,"0");
  }else
  {
    Serial.println("LUZ APAGADA");
    customWrite(3,"0");
  }
}
};
// Activa el socket UDP en el puerto
Serial.println(socket.begin(port));
}

function loop()
{
  // Nada
}

```

En este código primero vemos de qué pista se debe realizar el cambio y a continuación se modifica el estado de la luz.

4.4.1.2 Acceso remoto a las luces

A cada una de las luces se les ha añadido el *network adapter* PT-IOT-NM-1W para poder acceder a ellas de forma remota [3].

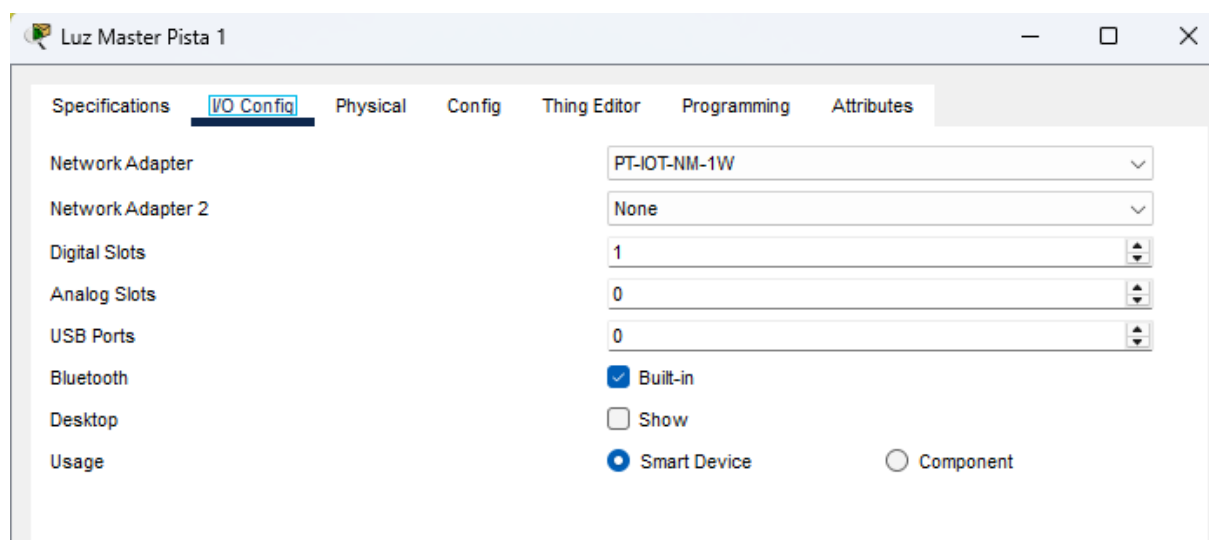


Figura 13: Luces PT-IOT-NM-1W

Una vez añadido el adaptador se han conectado todas al Home Gateway de la oficina el cual se describe en el punto 4.4 Diseño de la Oficina.

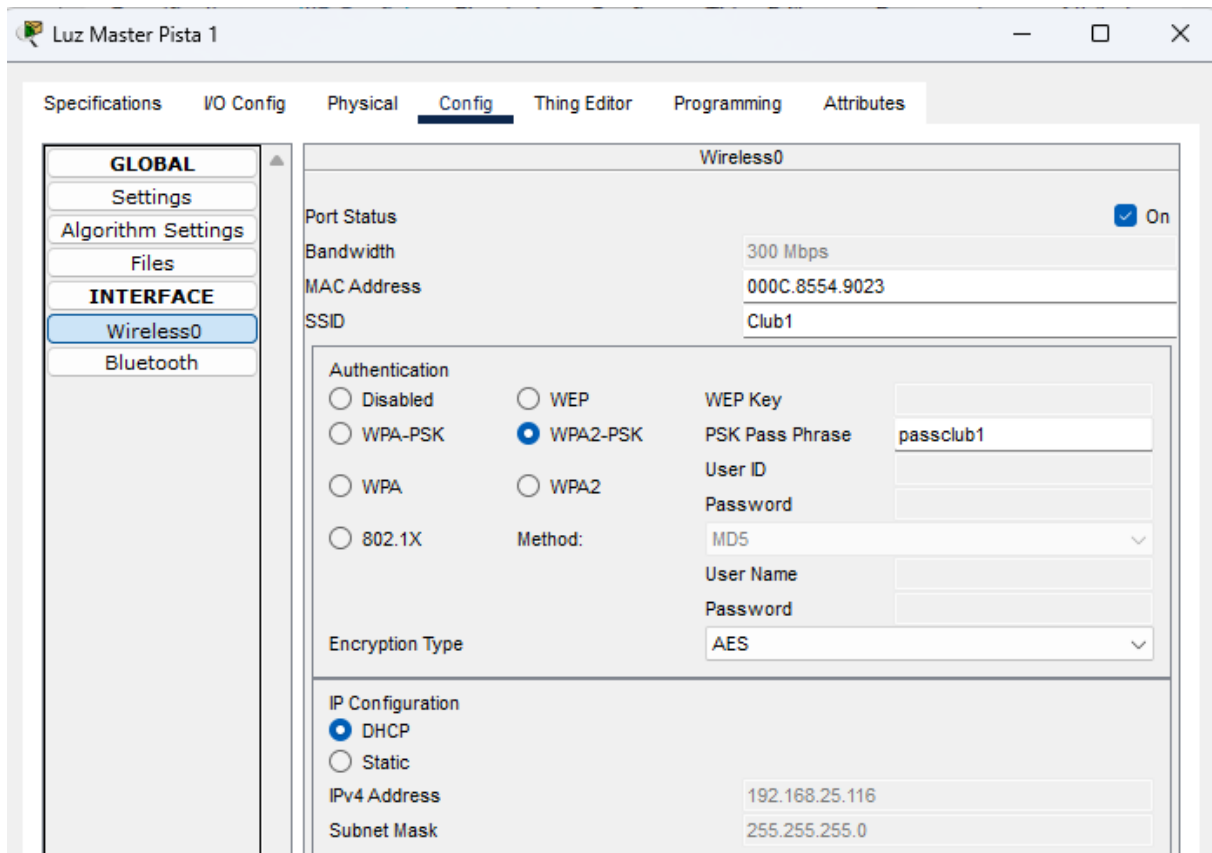


Figura 14: Conexión a Home Gateway de Luces

Y se han conectado al servidor IoT remoto descrito en el anterior punto indicando la dirección 200.100.1.1 y los credenciales creados (club1, club1).

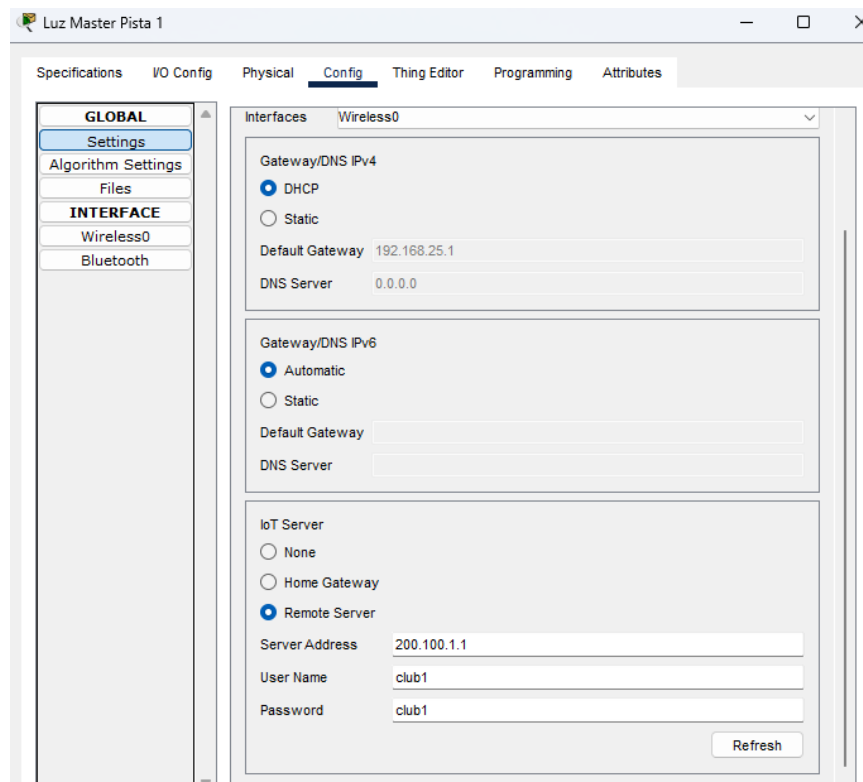
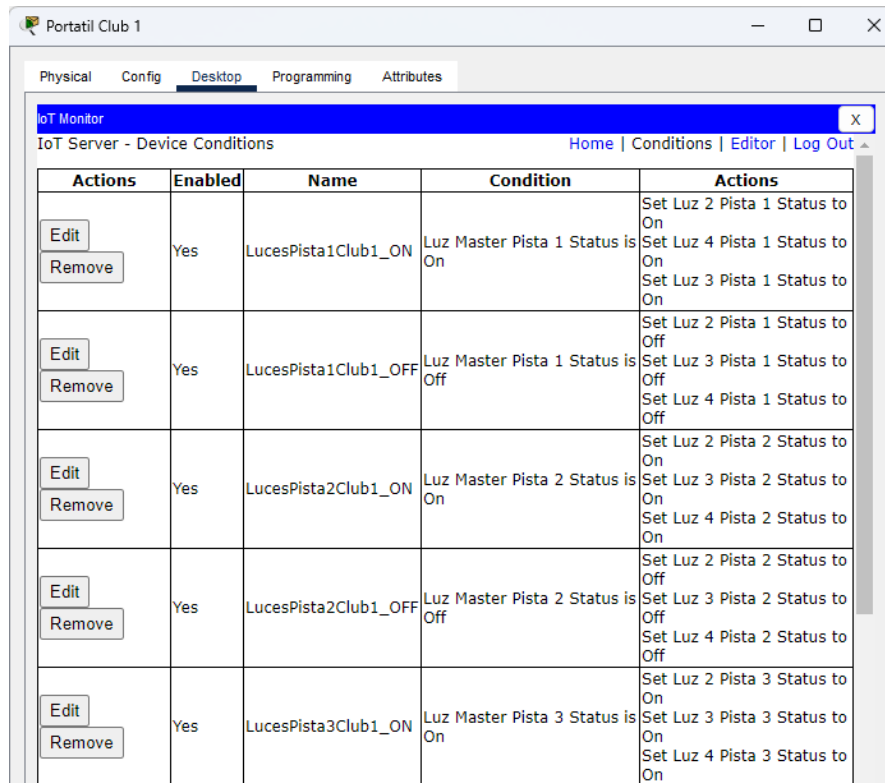


Figura 15: Luces Conexión IoT Server Remoto

Para conseguir que las luces tengan siempre el mismo estado que la luz master se crearán reglas en el servidor IoT. Si se cumple la condición de que la luz que no interese está encendida, entonces todas las demás luces de la pista deberán estarlo también. Sucederá de manera idéntica si están apagadas.



Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	LucesPista1Club1_ON	Luz Master Pista 1 Status is On	Set Luz 2 Pista 1 Status to On Set Luz 4 Pista 1 Status to On Set Luz 3 Pista 1 Status to On
Edit Remove	Yes	LucesPista1Club1_OFF	Luz Master Pista 1 Status is Off	Set Luz 2 Pista 1 Status to Off Set Luz 3 Pista 1 Status to Off Set Luz 4 Pista 1 Status to Off
Edit Remove	Yes	LucesPista2Club1_ON	Luz Master Pista 2 Status is On	Set Luz 2 Pista 2 Status to On Set Luz 3 Pista 2 Status to On Set Luz 4 Pista 2 Status to On
Edit Remove	Yes	LucesPista2Club1_OFF	Luz Master Pista 2 Status is Off	Set Luz 2 Pista 2 Status to Off Set Luz 3 Pista 2 Status to Off Set Luz 4 Pista 2 Status to Off
Edit Remove	Yes	LucesPista3Club1_ON	Luz Master Pista 3 Status is On	Set Luz 2 Pista 3 Status to On Set Luz 3 Pista 3 Status to On Set Luz 4 Pista 3 Status to On

Figura 16: Device Conditions Luces

4.4.2 Diseño de las puertas

4.4.2.1 Acceso presencial a las puertas

Para conseguir un acceso desde los interruptores a las puertas se ha realizado el mismo procedimiento que con las luces. Sin embargo, en este caso solo tenemos una puerta por pista, por lo que no será necesario establecer las condiciones en el servidor IoT.

```
var port = 1234;
var socket;

function setup()
{
  socket = new UDPSocket();
  customWrite(0, "0");

  // Recepcion
  socket.onReceive = function(ip, port, data)
  {
    Serial.println("Recibido de "
      + ip + ":" + port + ": " + data);
    data = data.split(" ");
    if (data[0] == "1")
    {
      Serial.println("Recibido de "
        + ip + ":" + port + ": " + data);
      if (data[1] == "1")
      {

```

```

        Serial.println("PUERTA 1 ABIERTA");
        customWrite(0,"1");
    }
    else
    {
        Serial.println("PUERTA 1 CERRADA");
        customWrite(0,"0");
    }
}
else if (data[0] == "2")
{
    Serial.println("Recibido de "
+ ip + ":" + port + ": " + data);
    if(data[1]=="1")
    {
        Serial.println("PUERTA 2 ABIERTA");
        customWrite(1,"1");
    }
    else
    {
        Serial.println("PUERTA 2 CERRADA");
        customWrite(1,"0");
    }
}
else if (data[0] == "3")
{
    Serial.println("Recibido de "
+ ip + ":" + port + ": " + data);
    if(data[1]=="1")
    {
        Serial.println("PUERTA 3 ABIERTA");
        customWrite(2,"1");
    }
    else {
        Serial.println("PUERTA 3 CERRADA");
        customWrite(2,"0");
    }
}
else if (data[0] == "4")
{
    Serial.println("Recibido de "
+ ip + ":" + port + ": " + data);
    if(data[1]=="1")
    {
        Serial.println("PUERTA 4 ABIERTA");
        customWrite(3,"1");
    }
    else {
        Serial.println("PUERTA 4 CERRADA");
        customWrite(3,"0");
    }
}
}
};
// Activa el socket UDP en el puerto
Serial.println(socket.begin(port));
}

function loop()
{
    // Nada
}

```

4.4.2.2 Acceso remoto a las puertas

Por último, para conseguir tener un acceso remoto a las puertas tendremos que añadir el mismo adaptador que a las luces.

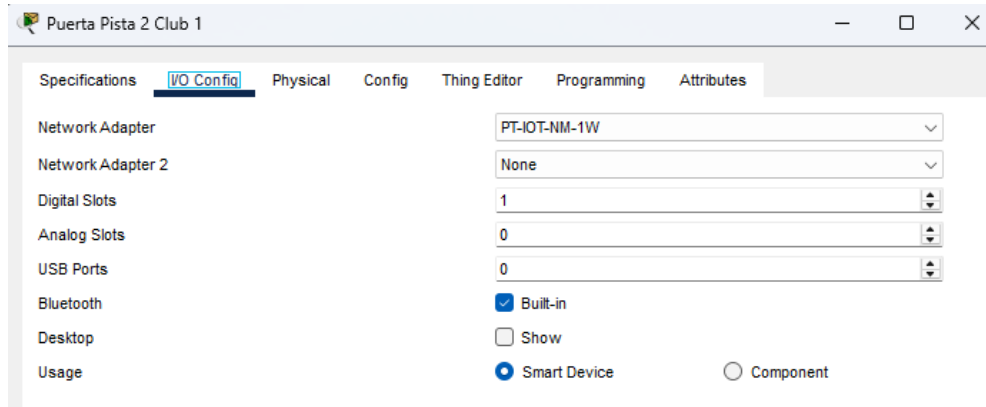


Figura 17: Network Adapter Puerta

Establecerán una conexión con router de la oficina y se conectarán al servidor IoT remoto también.

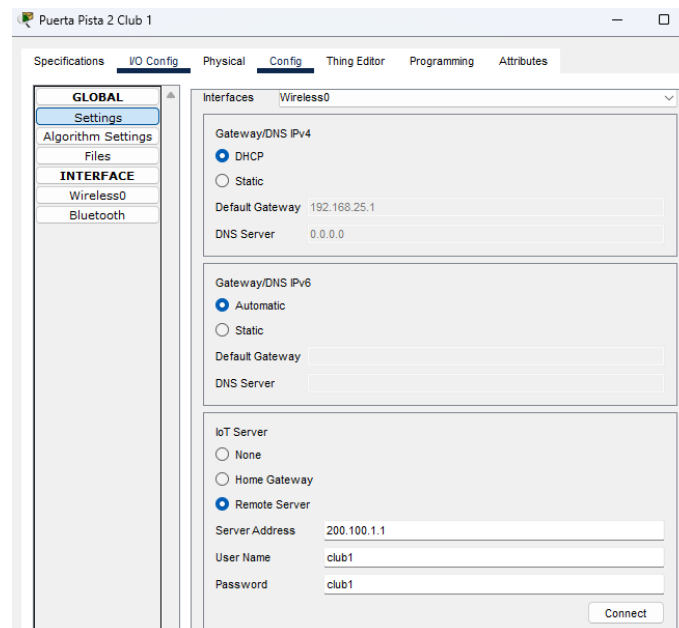


Figura 18: Conexión Remote Server Puertas

Capítulo 5

PRUEBAS

5.1 Acceso Servidor IoT

Primero, se accederá al servidor IoT remoto desde el smartphone para comprobar que todos los dispositivos son accesibles.

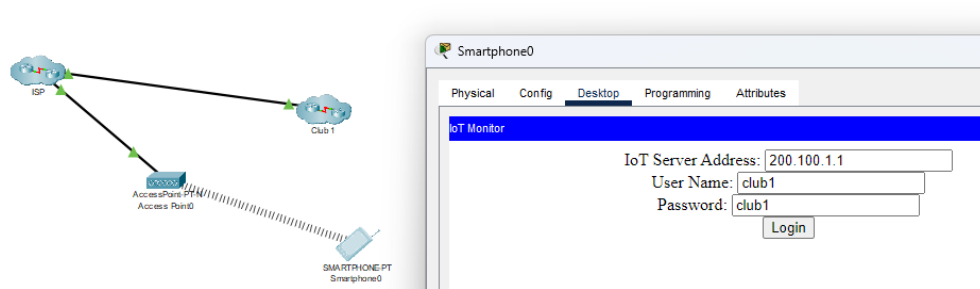


Figura 19: Acceso a Servidor Remoto desde Smartphone

Podemos observar cómo obtenemos una lista con todos los elementos del proyecto. Desde las luces, puertas, calefacción, etc.



Figura 20: Lista Dispositivos IoT Server

5.2 Control de Luces con Potenciómetros

Activaré los potenciómetros de la pista 3 y 1.

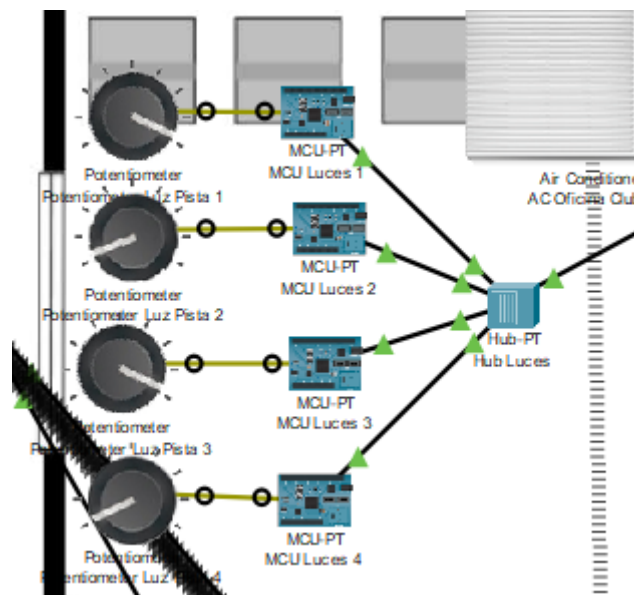


Figura 21: Potenciómetros 1 y 3 Activados

En los MCUs podemos observar como recibimos el feedback de las luces encendidas y apagadas.

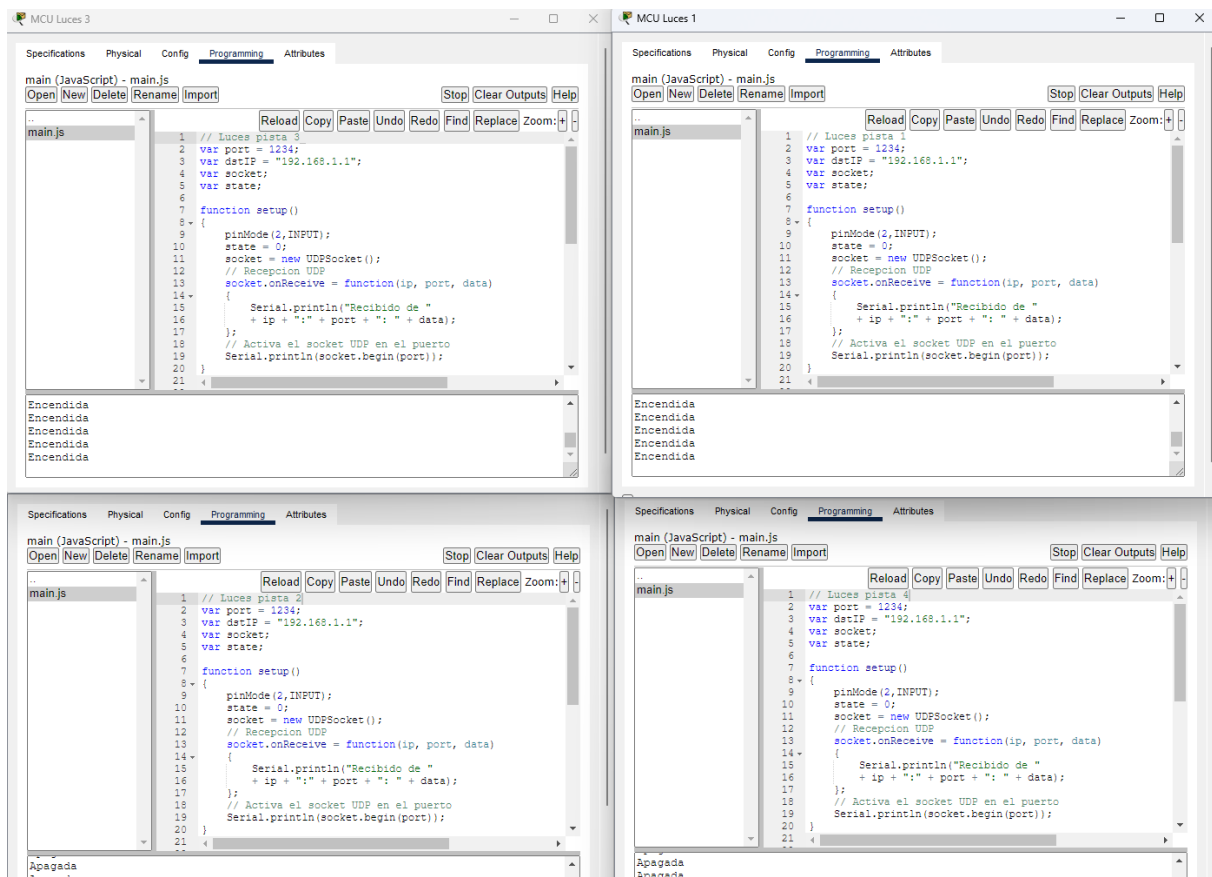


Figura 22: Outputs MCUs

Y si salimos de la oficina a las pistas podremos ver como están encendidas las luces de las pistas 1 y 3.

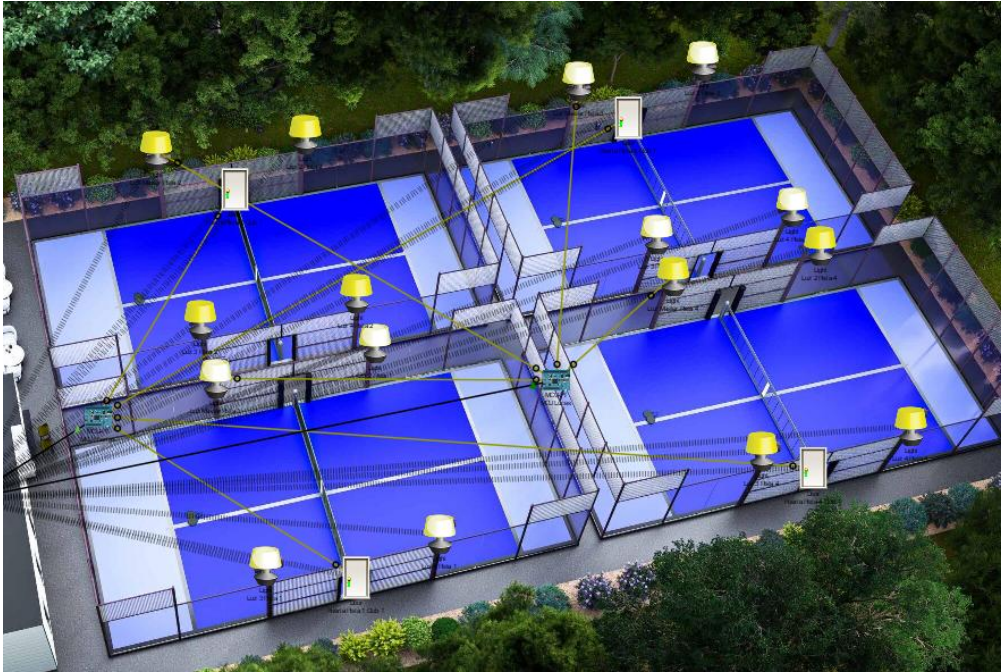


Figura 23: Luces Pistas 3 y 1 Encendidas

5.3 Control de Puertas con Interruptores

Ahora activaré los interruptores de las puertas 2 y 4

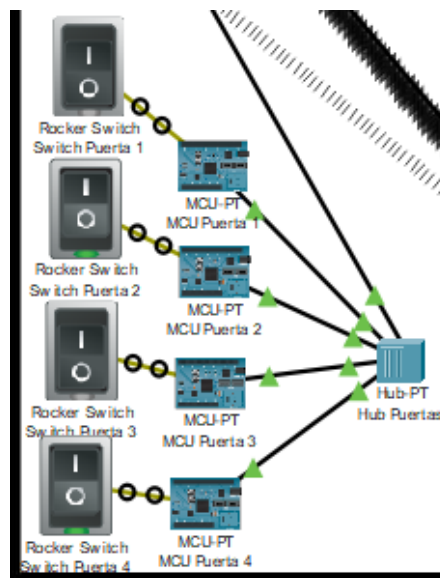


Figura 24: Interruptores 2 y 4 Activados


```

// MCU Puerta 2
main (JavaScript) - main.js
1 // Puerta pista 1
2 var port = 1234;
3 var destIP = "192.168.1.1";
4 var socket;
5 var state;
6
7 function setup()
8 {
9   pinMode(2, INPUT);
10  state = 0;
11  socket = new UDPSocket();
12  // Reception UDP
13  socket.onReceive = function(ip, port, data)
14  {
15    Serial.println("Recibido de "
16      + ip + ":" + port + ": " + data);
17  };
18  // Activa el socket UDP en el puerto
19  Serial.println(socket.begin(port));
20  }
21
22 function loop()
23 {
24   if (digitalRead(0))
25   {
26     if (state == 0)
27     {
28       state = 1;
29       socket.send(destIP, port, "3 1");
30       Serial.println("PUERTA 1 ON");
31     }
32   }
33 }

Starting main (JavaScript)...
true
PUERTA 1 ON

// MCU Puerta 4
main (JavaScript) - main.js
1 // Puerta pista 1
2 var port = 1234;
3 var destIP = "192.168.1.1";
4 var socket;
5 var state;
6
7 function setup()
8 {
9   pinMode(2, INPUT);
10  state = 0;
11  socket = new UDPSocket();
12  // Reception UDP
13  socket.onReceive = function(ip, port, data)
14  {
15    Serial.println("Recibido de "
16      + ip + ":" + port + ": " + data);
17  };
18  // Activa el socket UDP en el puerto
19  Serial.println(socket.begin(port));
20  }
21
22 function loop()
23 {
24   if (digitalRead(0))
25   {
26     if (state == 0)
27     {
28       state = 1;
29       socket.send(destIP, port, "4 1");
30       Serial.println("PUERTA 4 ON");
31     }
32   }
33 }

Starting main (JavaScript)...
true
PUERTA 4 ON

```

Figura 25: Outputs MCUs Puertas

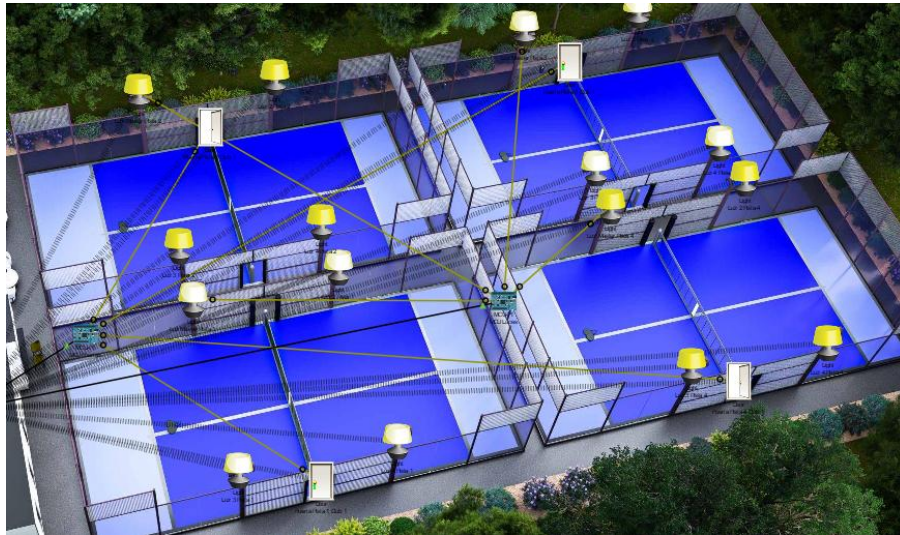


Figura 26: Puertas 2 y 4 Abiertas

5.4 Climatización automática

En el SBC conectado a la calefacción y el aire acondicionado podemos observar como está recibiendo en todo momento la temperatura de la sala. Como era 14,57°C se activó la calefacción hasta que la temperatura superase los 15 grados, entonces se desactivó.


```
Success: Received message '14.57' from topic '/club1/sensor/temp1'.  
Temperature menor de 15 grados. Activando calefaccion.  
Success: Received message '14.57' from topic '/club1/sensor/temp1'.  
Temperature menor de 15 grados. Activando calefaccion.  
Success: Received message '15.35' from topic '/club1/sensor/temp1'.  
Temperature entre 25 y 15 grados. Desactivando ac y calefaccion.  
Success: Received message '15.35' from topic '/club1/sensor/temp1'.  
Temperature entre 25 y 15 grados. Desactivando ac y calefaccion.
```

Figura 27: SBC Climatización

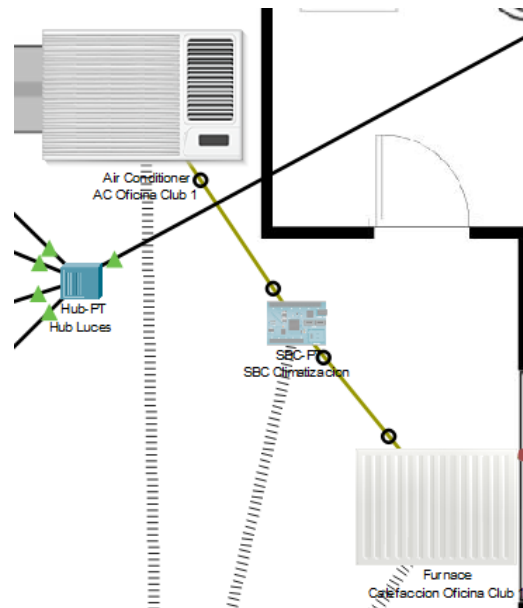


Figura 28: Calefacción Activada

En el SBC con el sensor de temperatura podemos observar como se está publicando en todo momento la temperatura.

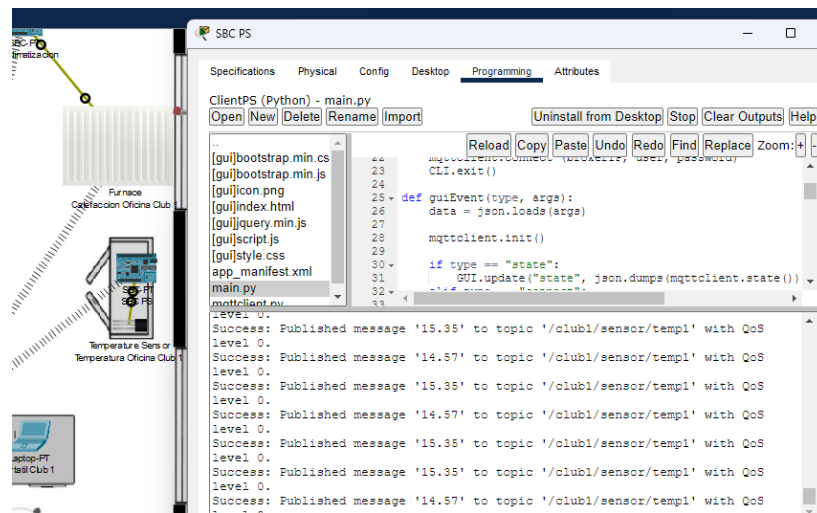


Figura 29: Temperatura Publicada

BIBLIOGRAFÍA

- [1] ClassVirtual. (s.f.). Configuración de Servidores DNS y DHCP en Packet Tracer. Recuperado en enero de 2025 de <https://eclassvirtual.com/configuracion-de-servidores-dns-y-dhcp-en-packet-tracer/>
- [2] Palomares Muñoz, J. M., León García, F. (2024). Interconectando dos MCU. Universidad de Córdoba. Máster Universitario en Inteligencia Computacional e Internet de las Cosas. https://moodle.uco.es/m2425/pluginfile.php/212733/mod_resource/content/2/PracticaPacketTracer-2.1.pdf
- [3] Palomares Muñoz, J. M., León García, F. (2024). SmartHome - Domótica. Universidad de Córdoba. Máster Universitario en Inteligencia Computacional e Internet de las Cosas. https://moodle.uco.es/m2425/pluginfile.php/212734/mod_resource/content/2/PracticaPacketTracer-3.1.pdf
- [4] Palomares Muñoz, J. M., León García, F. (2024). MQTT Básico. Universidad de Córdoba. Máster Universitario en Inteligencia Computacional e Internet de las Cosas. https://moodle.uco.es/m2425/pluginfile.php/212736/mod_resource/content/2/PracticaPacketTracer-4.1.pdf