Control Climático en alcobas Inteligentes a través del Análisis Meteorológico IoT, implementando una atención con colas de mensajes. – Práctica #2

María Isabel Masaya Córdova - 201800565, Cristian Hernández - 202010905, Ángel Geovany Aragón Pérez – 201901055, Juan Pablo González Leal – 201901374, Karen Michelle Gatica Arriola - 201900603

Resumen

Se propone desarrollar una estación meteorológica de IoT para una habitación inteligente. Utiliza sensores para medir temperatura, luz, calidad del aire y proximidad, conectados a un Arduino con módulo Wi-Fi. Los datos se envían a una plataforma en la nube utilizando MQTT, y se almacenan en una base de datos. Dos aplicaciones, web y móvil, permiten visualizar y controlar dispositivos. Incluye funciones como control de iluminación según la presencia humana, mejora de la calidad del aire, control de temperatura y seguridad para autorizar el acceso a la habitación. La seguridad y privacidad de los datos son consideraciones clave en el proyecto.

***Abstract*: The project aims to develop an IoT weather station for a smart room. It utilizes sensors to measure temperature, light, air quality, and proximity, all connected to an Arduino with a Wi-Fi module. Data is transmitted to a cloud-based platform using MQTT and stored in a database. Two applications, web and mobile, enable data visualization and device control. It includes functions like lighting control based on human presence, air quality enhancement, temperature control, and security to authorize room access. Data security and privacy are pivotal considerations in the project.**

I. Introducción

En un mundo cada vez más orientado hacia la automatización y la interconexión de dispositivos, la creación de un entorno inteligente y eficiente se ha convertido en una prioridad. Este proyecto se centra en el diseño y desarrollo de una estación meteorológica de Internet de las cosas (IoT) que no solo recopila datos meteorológicos esenciales, como la temperatura, la luz y la calidad del aire, sino que también permite un control inteligente de una habitación.

II. OBJETIVOS

1. Diseñar un dispositivo destinado a medir registrar regularmente diversas variables meteorológicas.

2. Aprender a desarrollar una solución mediante la correcta implementación del framework de IoT.

3. Diseñar un algoritmo de análisis de datos que interprete la información meteorológica recopilada y prediga patrones climáticos relevantes para optimizar el control climático en la habitación inteligente.

4. Implementar una plataforma de gestión centralizada que permita la integración de datos meteorológicos y el control de dispositivos en la habitación inteligente, ofreciendo una experiencia de usuario intuitiva y accesible.

III. FUNCIONAMIENTO, USOS E IMPACTO AMBIENTAL

Funcionamiento:

- ✓ **Sensores de Medición: l**os sensores de temperatura, luz, calidad el aire y proximidad recopilan datos en tiempo real.
- ✓ **Arduino y Conexión Inalámbrica:** un Arduino procesa los datos de los sensores y los transmite a través utilizando el protocolo MQTT.
- ✓ **Plataforma en la Nube:** los datos se envían a una plataforma centralizada en la nube, donde se almacenan de manera segura en una base de datos.
- ✓ **Análisis de Datos:** se aplican algoritmos de análisis de datos para interpretar la información meteorológica y predecir patrones climáticos relevantes.
- ✓ **Aplicación Web y Móvil:** los usuarios pueden acceder a los datos y controlar dispositivos a través de aplicaciones web y móviles. Pueden ajustar la iluminación y autorizar el acceso a la habitación.
- ✓ **Gestión de Datos:** se implementa un sistema de atención de datos para garantizar que todos los datos generados se procesen de manera eficiente.
- ✓ **Usos:** monitoreo Meteorológico en Interiores: proporciona datos precisos sobre la temperatura, la calidad del aire y la luz en una habitación.
- ✓ **Gestión de la Comodidad:** permite a los usuarios ajustar la iluminación y la temperatura según las condiciones y su presencia en la habitación.
- ✓ **Eficiencia Energética**: ayuda a reducir el consumo de energía al optimizar la iluminación y la climatización en función de las condiciones y la ocupación.
- ✓ **Seguridad:** controla el acceso a la habitación y ofrece notificaciones en caso de condiciones adversas.

Beneficios:

- ✓ **Comodidad Personalizada**: los usuarios pueden disfrutar de un entorno más cómodo y personalizado, cuenta con control sobre la luz ya que esta se apaga cuando no se encuentra alguien en la habitación, además de purificación del aire.
- ✓ **Ahorro de Energía:** la gestión eficiente de la iluminación y la climatización ayuda a reducir el consumo de energía y costos.
- ✓ **Salud y Bienestar:** el monitoreo de la calidad del aire contribuye a un ambiente más saludable, reduciendo la exposición a niveles altos de CO_2.
- ✓ **Seguridad Mejorada**: el sistema de seguridad mejora la seguridad al autorizar o denegar el acceso a la habitación y notificar a los usuarios sobre situaciones críticas.

Impacto Ambiental:

- ✓ **Reducción de Emisiones:** La optimización de la iluminación y la climatización reduce el consumo de energía, lo que disminuye las emisiones de gases de efecto invernadero.
- ✓ **Menos uso de Recursos:** La gestión eficiente de la energía y el aire contribuye a la conservación de los recursos naturales.
- ✓ **Mejora de la Calidad del Aire:** El monitoreo y la limpieza del aire en la habitación pueden mejorar la calidad del aire interior y reducir la exposición a contaminantes.

IV.BOCETOS DEL PROTOTIPO

Delantera

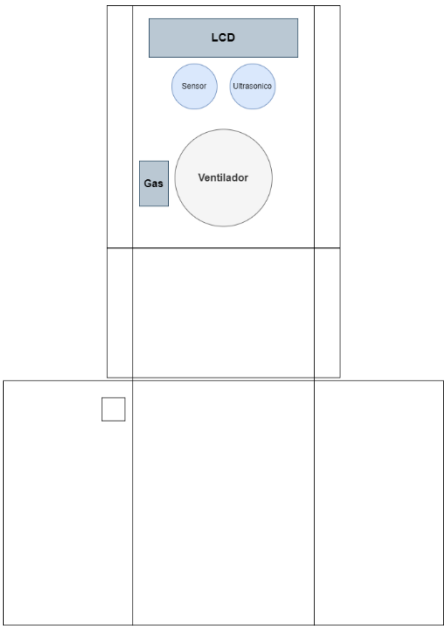


Figura 1: Boceto parte delantera

Adentro

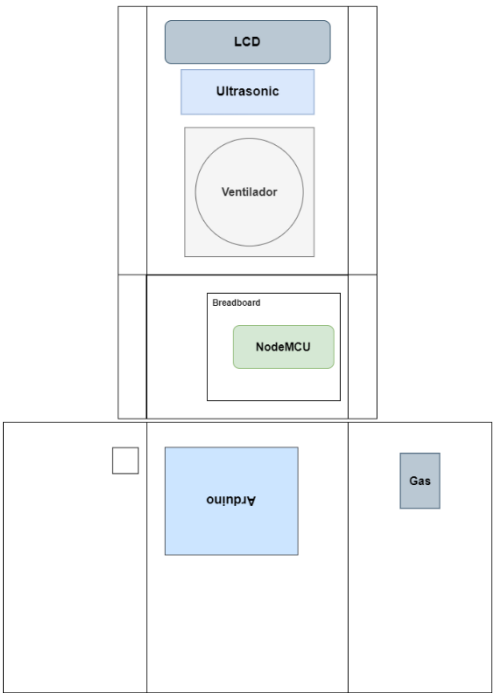
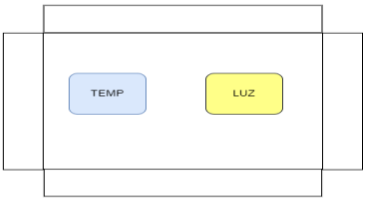


Figura 3: Boceto parte interna

Tapadera Delantera



Tapadera Adentro

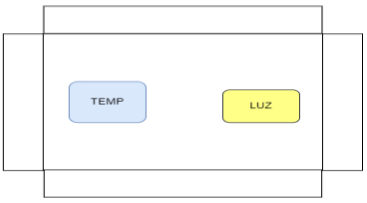


Figura 2: Boceto tapadera delantera y trasera

V. PROTOTIPO FINAL

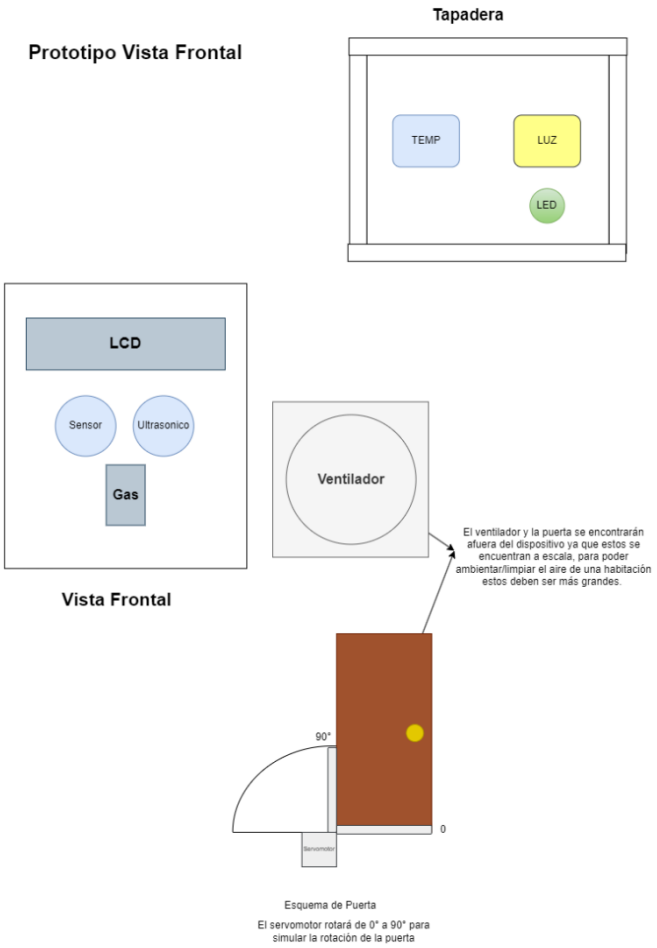


Figura 4. Prototipo vista frontal

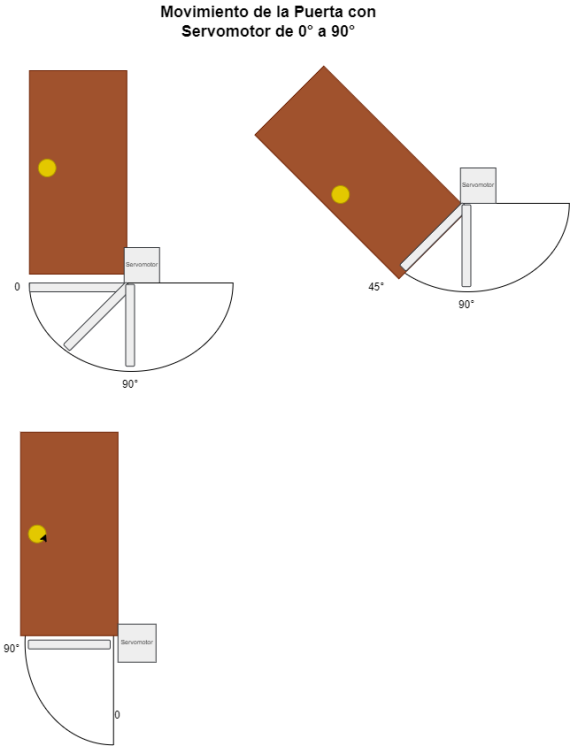


Figura 5: Movimiento de puerta con servomotor.

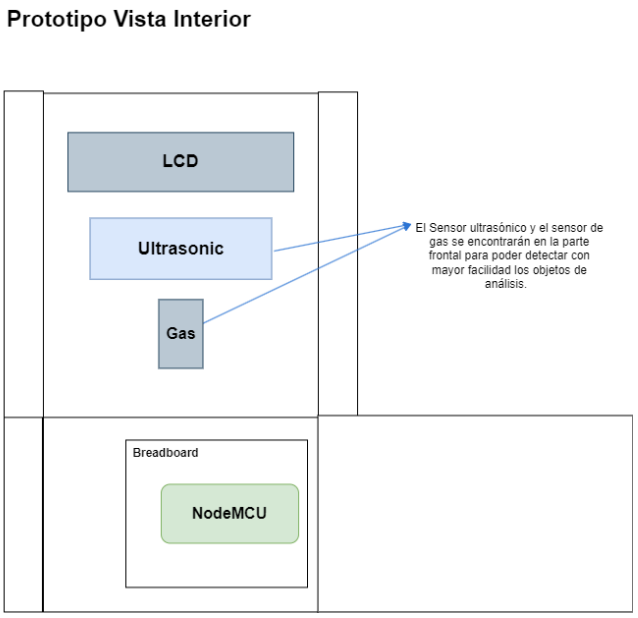


Figura 6: Protótipo vista interior

Componentes Usados

El proyecto fue desarrollado usando:

- Arduino UNO GENUINO
- Sensor de temperatura y Humedad DHT11
- Sensor de Gas MQ-135 (Solamente usando la parte digital)
- Servomotor
- Ventilador DC 12V
- Módulo de Fotorresistencia (Solamente usando la parte digital)
- Medidor de distancia HC-SR04
- LED de color verde
- 1 resistencia de 220 kΩ
- 1 Breadboard
- Jumpers Macho-Hembra
- Jumpers Macho-Macho

Código de Arduino Explicado

1. Declaración de Librerías, Pines y Objetos

```
// importar estas librerías al IDE de Arduino
#include <LiquidCrystal_I2C.h> // Uso de LDC con el módulo I2C
#include <Wire.h> // I2C
#include <DHT.h> // Sensor de temperatura y Humedad
#include <MQ135.h>  // Librería para el sensor de gas
#include <Servo.h>    // Uso del Servomotor


// Objeto del LCD     Analog Fila Columna
LiquidCrystal_I2C  lcd(0x27,  16,   2);

// definiendo el tipo de dato según su medición
int SENSOR = 6;    // DHT11 a pin digital 6
float TEMPERATURA;
int HUMEDAD;

// Pines del sensor ultrasónico HC-SR04
int TRIG = 10;
int ECHO = 9;
int DURACION, DISTANCIA;

DHT dht(SENSOR, DHT11); // creación del objeto para el DHT11

// En #define se puede cambiar el tipo de sensor
// ya que existen varios de la serie MQ-XXX
#define PIN_MQ135 A1
MQ135 mq135_sensor(PIN_MQ135); //Creación del objeto MQ135

// Variables de control del Servomotor
Servo myservo;  // Crear un objeto para controlar el servomotor
int pos = 0;    // Variable que almacena la posición del servomotor

// Variable de control del Ventiladdor
int ventilador = 4;
```

2. Setup: inicializa los componentes

```
void setup()
{
  Serial.begin(9600); // Inicializacion del puerto serial 9600
  dht.begin();                          // inicializacion de sensor DHT11

  //LCD with I2C
  lcd.backlight(); // Activación luz trasera
  lcd.init();  // Inicio del LCD

  // HC-SR04 PinMode
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  // Conecta el servo en el pin 5 al objeto servo
  myservo.attach(5);

  // PinMode del Ventilador de 12V
  pinMode(ventilador, OUTPUT);

  // Se inicializa el pin digital LED_BUILTIN = 13 como salida.
  pinMode(LED_BUILTIN, OUTPUT);

}
```

3. Función seg_led: repite el ciclo después de indicar el estado de la luz y verificar la presencia humana

```
void seg_led(DISTANCIA, LUMEN){
 if(DISTANCIA <= 30 && lumen > 50 ){ //Presencia Humana, hay luz
    digitalWrite(LED_BUILTIN, HIGH);  // Manda un estado alto (Encendido)
    delay(1000);
 }else if (DISTANCIA > 30 && lumen > 50){ //No hay Presencia Humana, hay luz
    digitalWrite(LED_BUILTIN, LOW);  // Manda un estado bajo (Apagado)
    delay(1000);
 }

}
```

4. Loop

```
void loop() {
 //------------Medición de la distancia de objetos o personas----------
 //-------------- Codigo para HC-SR04----------------------------------
 digitalWrite(TRIG, HIGH);  // generación del pulso a enviar
 delay(1)                          // al pin conectado al trigger
 digitalWrite(TRIG, LOW); // del sensor

 DURACION = pulseIn(ECHO, HIGH); // con función pulseIn se espera un pulso
 DISTANCIA = DURACION / 58.2; // distancia medida en centímetros

 delay(1000);


 //---------- Medición de Temperatura Y Humedad
  // -----------       Sensor DHT11
  TEMPERATURA = dht.readTemperature(); // obtención de valor de temperatura
  HUMEDAD = dht.readHumidity(); // obtención de valor de humedad

          // x, y
   lcd.setCursor(0, 0);  //setCursor define la posición del cursor en el led
   lcd.print("Temperatura");
```

```
            // x, y
    lcd.setCursor(0, 1);
    lcd.print( String(TEMPERATURA) + " C");  // Imprime el valor de la temperatura
    delay(1000);
    lcd.clear(); // Limpia el display para ingresar nuevos datos

    // notificación de temperatura dentro de la habitación. (Móvil)

    /* clientMessage = instrucción que recibe de la app

     if(clientMessage = "G13_1"){      // Enciende el ventilador
       digitalWrite(ventilador, HIGH);
      }else if(clientMessage = "G13_0"){  // Apaga el ventilador
       digitalWrite(ventilador,LOW);
      }else{
       System.println("No se encontró el ventilador");
      }
    */


  //---------- Cantidad de Luz en el ambiente
   //----------   Módulo de Fotoresistencia
   int Value_D8 = analogRead(A0);   //Leyendo el pin analógico A0 en Arduino

   // Conversion Lux to Lumen
    // Donde 0 representa claridad y 100 oscuridad, ascendente
    int lux  = map(Value_D8, 0, 1024, 0, 100); //  0, 1024

   // Lumen = Lux x Area m2
     // NOTA 9m2 es 3 ancho 3 largo del cuarto
    lcd.setCursor(0, 0);
    lcd.print("Niveles de Luz");
    int lumen = lux*9;
     if (lumen  <= 50) {
      lcd.setCursor(0, 1);
      lcd.print("Normal = " + String(lumen));
      delay(1000);
      lcd.clear(); // Limpia el display para ingresar nuevos datos
     } else if (lumen  > 50) {
      lcd.setCursor(0, 1);
      lcd.print("High = " + String(lumen) );
      delay(1000);
      lcd.clear(); // Limpia el display para ingresar nuevos datos
     }

    //------------------Código para el LED----------------------------
    // DISTANCIA > 30  NO hay presencia humana
    // DISTANCIA <= 30  hay presencia humana

    // Primer Ciclo del LED
    if(DISTANCIA > 30 && lumen > 50){  // No hay presencia, luz encendida
       /*
          La App debe notificar que la habitación esta iluminada
          Pero no hay nadie en ella
           * Se puede enviar a través del serial de arduino
          algun indicador
       */


      // Segundo Ciclo 20 a 30 seg
      dseg_led(DISTANCIA, LUMEN);

    }else if(DISTANCIA <= 30 && lumen > 50 ){ //Presencia Humana, hay luz
      digitalWrite(LED_BUILTIN, HIGH); // Manda un estado alto (Endendico)
      delay(1000);                  // wait for a second
    }

    else{
     System.println("No se encontraron datos");
    }



   //---------- Medición de Calidad del Aire
   //----------   Sensor de Gas MQ-135
   float ppm = mq135_sensor.getPPM(); // Obtener el valor de CO2 en el aire
   float correctedPPM = mq135_sensor.getCorrectedPPM(TEMPERATURA, HUMEDAD);
    // clientMessage = instrucción que recibe de la app
     lcd.setCursor(0, 0);
     lcd.print("Niveles de Gas");

     //Inicio del temporizador DE 20 A 30 SEG
      if (ppm  <= 450) {
       lcd.setCursor(0, 1);
       lcd.print("Normal = " + String(correctedPPM));
       delay(1000);
       lcd.clear(); // Limpia el display para ingresar nuevos datos
        /*
        if(clientMessage = "G13_E"){      // Enciende el ventilador Web
          digitalWrite(ventilador, HIGH);
```

```
          /*
            La App debe notificar que el nivel de
            aire es correcto.
             * Se puede enviar a través del serial de
arduino
              algun indicador
              * O se hace desde la Api, indicando los
parámetros
                <= 450 niveles normales
                >= 451 niveles altos
           */
         }
         */

       } else if (ppm  > 451) {
         lcd.setCursor(0, 1);
         lcd.print("High = " + String(correctedPPM));
         delay(1000);
         lcd.clear(); // Limpia el display para ingresar
nuevos datos

         //Segundo Ciclo 20 a 30 seg
         /*
         if(clientMessage = "G13_F"){  // Apaga el
ventilador Web
           digitalWrite(ventilador,LOW);
         }
         */
       }
```

```
  //---------- Sistema de Seguridad - Móvil
  //----------    Servomotor gira de 0 a 180
  /* clientMessage = instrucción que recibe de la app

  if(clientMessage = "G13_A"){      // Abre la puerta
    for (pos = 0; pos <= 110; pos += 1) { // De 0 grados a
90 grados
      myservo.write(pos);           // Decirle al servo que
vaya a la posición en la variable 'pos'
      delay(15);                    // Espera 15 ms para que el
servo alcance la posición
    }
  }else if(clientMessage = "G13_B"){  // Cierra la puerta
    for (pos = 110; pos >= 0; pos -= 1) { // De 0 grados a
90 grados
      myservo.write(pos);           // Decirle al servo que
vaya a la posición en la variable 'pos'
      delay(15);                    // Espera 15 ms para que el
servo alcance la posición
    }
  }else{
    System.println("No hay movimiento");
  }
  */
```

```
  //----------Envío de Resultados a través del monitor Serial
9600
  Serial.println("hum "+ String(HUMEDAD));
  Serial.println("temp "+ String(TEMPERATURA));      //
escritura en monitor serial de los valores
  Serial.println("light "+ String(lumen));
  Serial.println("gas " + String(ppm));
  Serial.println("Distancia: " String(DISTANCIA));

}
```

VII. SMART CONNECTED DESIGN FRAMEWORK

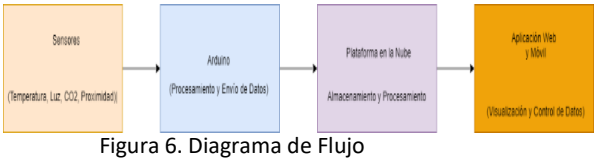


Figura 6. Diagrama de Flujo

VI. MUCKUPS MOVIL Y WEB

Figura 7. Muckup móvil

En la aplicación móvil se tendrá la opción de poder manejar la velocidad de la ventilación cuando la temperatura se encuentre elevada, esta tiene una lista de velocidades, de igual forma se puede habilitar o deshabilitar el acceso a la habitación.

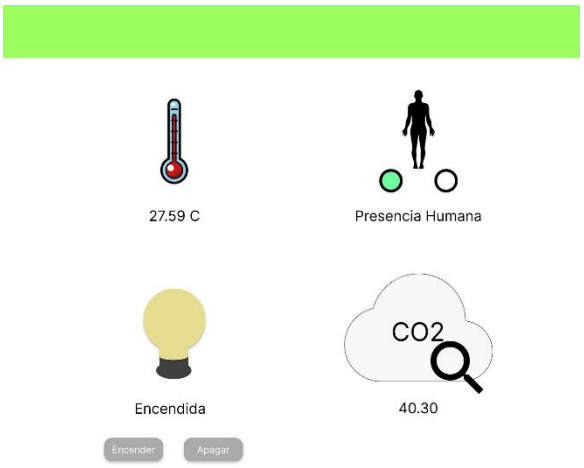


Figura 8. Muckup web

En el sitio web se tiene acceso al resto de opciones que se logran apreciar en la imagen como el encendido o apagado de la luz, la visualización de temperatura, si existe presencia humana se pondrá verde una luz en dado contrario se pondrá en rojo, y también la visualización del CO2.

VII. CAPAS DEL FRAMEWORK IOT

0. Things

Los objetos físicos que se encuentra en la habitación:

- Un ventilador que realiza la ventilación de la habitación.
- Una LED que simula la bombilla de la habitación.

1. Device Hardware

Los componentes físicos utilizados para el funcionamiento adecuado del dispositivo IoT:

- Un LCD con interfaz I2C para mostrar la información recopilada.
- Un sensor ultrasónico HC-SR04 para detectar la presencia humana.
- Un módulo de fotorresistencia para medir la luz ambiente.
- Un Sensor de temperatura DHT-11 para monitorear la temperatura de la habitación.
- Un sensor de gas MQ-135 para detectar el nivel de monóxido de carbono dentro de la habitación.
- Un servomotor para el movimiento de la puerta
- Un Arduino UNO para la comunicación con el backend de la API.

2. Device Software

Para llevar a cabo la implementación del dispositivo IoT, se emplearon diversas tecnologías y herramientas:

- Se utilizó el entorno de desarrollo de Arduino para administrar las funcionalidades de los componentes del dispositivo.
- Para establecer la comunicación entre el dispositivo IoT y otros elementos del sistema, se empleó el framework Flask de Python en el backend.
- La aplicación móvil se creó utilizando el entorno de desarrollo de App Inventor, con el objetivo de que fuera compatible con el sistema operativo Android.
- Para la implementación de la aplicación web, se utilizó React.

3. Communications

El Arduino UNO se encarga de transmitir los datos recopilados a través de la comunicación serial hacia una API a la que tanto la aplicación móvil como la aplicación web se conectan mediante el protocolo MQTT.

4. Cloud Platform

Para implementar nuestra API en la nube, utilizamos la plataforma fl0 debido a su facilidad de uso y escalabilidad. Además, para la gestión de la base de datos en la nube, optamos por Clever Cloud.

5. Cloud Applications

Hemos creado una aplicación tanto web como móvil que habilita a los usuarios para acceder a información meteorológica en tiempo real, obtenida de la ubicación de su habitación, al mismo tiempo que les proporciona el control de dispositivos en dicha área. Además, nuestra aplicación dispone de una sección dedicada al registro histórico de notificaciones recibidas.

IV.MQTT

MQTT es un protocolo de mensajería cliente-servidor público/de suscripción. Es fácil de implementar. Estas características son ideales para su uso en muchas situaciones, incluidos entornos restrictivos como entornos de máquina a máquina (M2M) o Internet de las cosas (IoT) que requieren pequeñas cantidades de código y/o versiones extendidas.

Broker:

MQTT es un protocolo ligero que admite Internet de las cosas (IoT). Este artículo explica la funcionalidad de su centro central conocido como el corredor MQTT, compara sus diversas implementaciones y revisa sus casos de uso, características y mejores prácticas.

Un corredor MQTT es una entidad intermedia que permite a Clientes MQTT comunicarse. Específicamente, un corredor MQTT recibe mensajes publicados por clientes, filtra los mensajes por tema y los distribuye a los suscriptores.

Estas son algunas de las principales razones por las que los corredores MQTT son importantes:

Enrutamiento de mensajes: El corredor MQTT recibe mensajes de los editores y los enruta a los suscriptores apropiados en función de sus suscripciones de temas.}

Escalabilidad: Los corredores MQTT pueden manejar una gran cantidad de conexiones simultáneas, lo cual es esencial para los escenarios de comunicación IoT y M2M, donde puede haber miles o incluso millones de dispositivos conectados.

Seguridad:
Los corredores MQTT pueden proporcionar medidas de seguridad como autenticación y cifrado para garantizar que los datos transmitidos entre dispositivos y aplicaciones IoT sean seguros.

Integración:
Los corredores MQTT pueden integrarse con otros protocolos de comunicación y plataformas en la nube para proporcionar una solución completa de IoT. Por ejemplo, los corredores MQTT pueden integrarse con AWS IoT, Google Cloud IoT o Microsoft Azure IoT Hub para proporcionar un ecosistema IoT sin interrupciones.

Gestión de la sesión:
El corredor MQTT es responsable de administrar las sesiones de los clientes, incluido el mantenimiento de información sobre las suscripciones del cliente y el manejo de los mensajes que se retienen para su entrega a los clientes cuando se conectan.

Brokers MQTT Gratuitos (de código abierto):

- Mosquitto: Mosquitto es uno de los brokers MQTT de código abierto más populares. Es ligero y fácil de configurar, lo que lo hace adecuado para proyectos de IoT y aplicaciones pequeñas. Mosquitto es mantenido por la Eclipse Foundation y está disponible de forma gratuita.

- HiveMQ: HiveMQ es otro broker MQTT de alto rendimiento que ofrece una versión gratuita llamada HiveMQ Community Edition. Es conocido por su escalabilidad y capacidades avanzadas, lo que lo hace adecuado para proyectos más grandes y críticos.

- EMQ X: EMQ X es un broker MQTT de código abierto que se destaca por su escalabilidad capacidad de manejar un gran número de conexiones simultáneas. Ofrece una versión gratuita con características básicas y una versión comercial con características adicionales.
- RabbitMQ: Aunque RabbitMQ es más conocido como un sistema de mensajería general, también puede ser configurado para admitir MQTT mediante un plugin. Es altamente personalizable y se utiliza en aplicaciones empresariales.

Brokers MQTT de Pago:

- AWS IoT Core: AWS IoT Core es un servicio de AWS que proporciona un broker MQTT administrado en la nube. Ofrece alta disponibilidad, escalabilidad y seguridad para proyectos de IoT a escala empresarial.

- Microsoft Azure IoT Hub: Azure IoT Hub es un servicio de Microsoft Azure que admite MQTT, entre otros protocolos. Ofrece características avanzadas de administración de dispositivos y escalabilidad. Los costos varían según el nivel de servicio.

- Google Cloud IoT Core: Google Cloud IoT Core es un servicio de Google Cloud que proporciona un broker MQTT administrado. Ofrece integración con otros servicios de Google Cloud y se factura según el uso.

- IBM Watson IoT Platform: IBM Watson IoT Platform ofrece un broker MQTT como parte de su suite de servicios de IoT. Proporciona análisis avanzados y herramientas de visualización. Los costos se basan en el consumo de recursos.

REPOSITORIO DE GITHUB

https://github.com/Angelgt3/ACE2_2S23_G13

REFERENCIAS

[1] G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529-551, April 1955.

[2] J. Clerk Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[4] K. Elissa, "Title of paper if known," no puplicado.

[5] R. Nicole, "Title of paper with only first word capitalized," *J. Name Stand. Abbrev.*, en impresión.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740-741, August 1987 [*Digests 9th Annual Conf. Magnetics Japan*, p. 301, 1982].