ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

APLICACIONES MÓVILES Y SERVICIOS TELEMÁTICOS

GRUPO 6:

Luis Guzmán

Cristóbal Lara

Jefferson Vega

MANUAL TÉCNICO

"Aplicación móvil avanzada – Monitor de calidad de aire"

Profesora: Msig. Adriana Collaguazo

Enlace repositorio: https://github.com/amst-fiec/1T2022-Monitor-de-calidad-del-aire-v1.0.git

I PAO – 2022

Introducción

La importancia de calidad de aire interior es vital debido a que regularmente las personas se mantienen en lugares de trabajo o domicilios en un 90%, por lo cual no se piensa frecuentemente en la calidad de aire que se respira que, a diferencia del aire exterior, se tiende a reciclar continuamente, provocando que se atrape contaminantes y permite que se acumulen dentro de espacios confinados. La IAQ (Indoor Air Quality), se refiere a las características ambientales dentro de edificaciones que puedan afectar a la salud humana, la comodidad o el rendimiento laboral. IAQ considera las concentraciones de contaminantes en el aire interior, con mediciones de temperatura y humedad, siendo la medida efectiva de niveles de IAQ para reducir los riesgos de salud asociados a la calidad deficiente del aire interior en espacios definidos, generando entornos más seguros y con menor impacto a la salud de las personas.

La contaminación del aire interior conlleva importantes riesgos para la salud a corto y largo plazo de los habitantes. Los síntomas típicos asociados con la mala calidad del aire interior incluyen irritación de los ojos, la nariz y la garganta, dolor de cabeza, náuseas, mareos y fatiga. En algunos casos, la exposición a la contaminación del aire interior puede provocar enfermedades respiratorias agudas y crónicas, como asma, cáncer de pulmón, neumonía, hipertensión sistémica, enfermedad pulmonar obstructiva crónica (EPOC). Así, la mala calidad del aire interior en el lugar de trabajo puede contribuir a la disminución de la productividad, el ausentismo e incluso posibles litigios. Al monitorear de manera efectiva la calidad del aire interior mediante un dispositivo que censará en tiempo real las variables ambientales de temperatura y humedad, comunicando mediante alertas asociadas a un contacto por vía WhatsApp reportando niveles bajos de calidad de aire en cualquier área definida.

Por lo cuál se garantiza que los habitantes y trabajadores puedan disfrutar de entornos saludables debido a la post-pandemia; se incluye una aplicación móvil que demostrará los escenarios que los usuarios dispongan a censar las variables ambientales y monitorear los niveles de la calidad de aire interior incrementando la bioseguridad en prevención de virus y contaminantes.

Diseño

Diseño del Software

- **App móvil**

La aplicación fue desarrollada en Android Studio Code con Java como lenguaje de programación principal. Se realizó una “Activity” de Splash para ser mostrada al inicializar la App.

En la actividad principal se permite acceder con una cuenta verificada de Google utilizando la autenticación de la misma asociada a Firebase. Además, en versiones de desarrollo se incorporó una entrada externa llamada prueba para poder acceder de manera más rápida al contenido.

Una vez realizado un log in exitoso, se creó una función que permite cerrar sesión de la cuenta accedida, así como también se realizó una verificación para que el usuario no tenga que iniciar sesión nuevamente si cerró la aplicación sin haber cerrado la misma. En el menú principal se pueden observar los escenarios que el usuario ha registrado previamente, estos se extraen directamente de la base de datos de Firebase, por lo que, si no existe ningún escenario registrado, se dispone de un botón “Registrar Escenario” que nos llevará a la siguiente ventana correspondiente.

En la ventana de registro de escenarios se dispone del botón para registrar un nuevo escenario, el mismo que al ser pulsado abrirá un “AlertDialog” que se sobrepondrá a la pantalla y nos podrá registrar los datos necesarios para la creación del escenario, posterior a esto se podrá guardar el mismo o cancelar la acción. Estos escenarios serán creados y guardados en la base de datos de Firebase mediante un “Post”. Con la existencia de escenarios en la base de datos, el usuario en esta misma ventana podrá

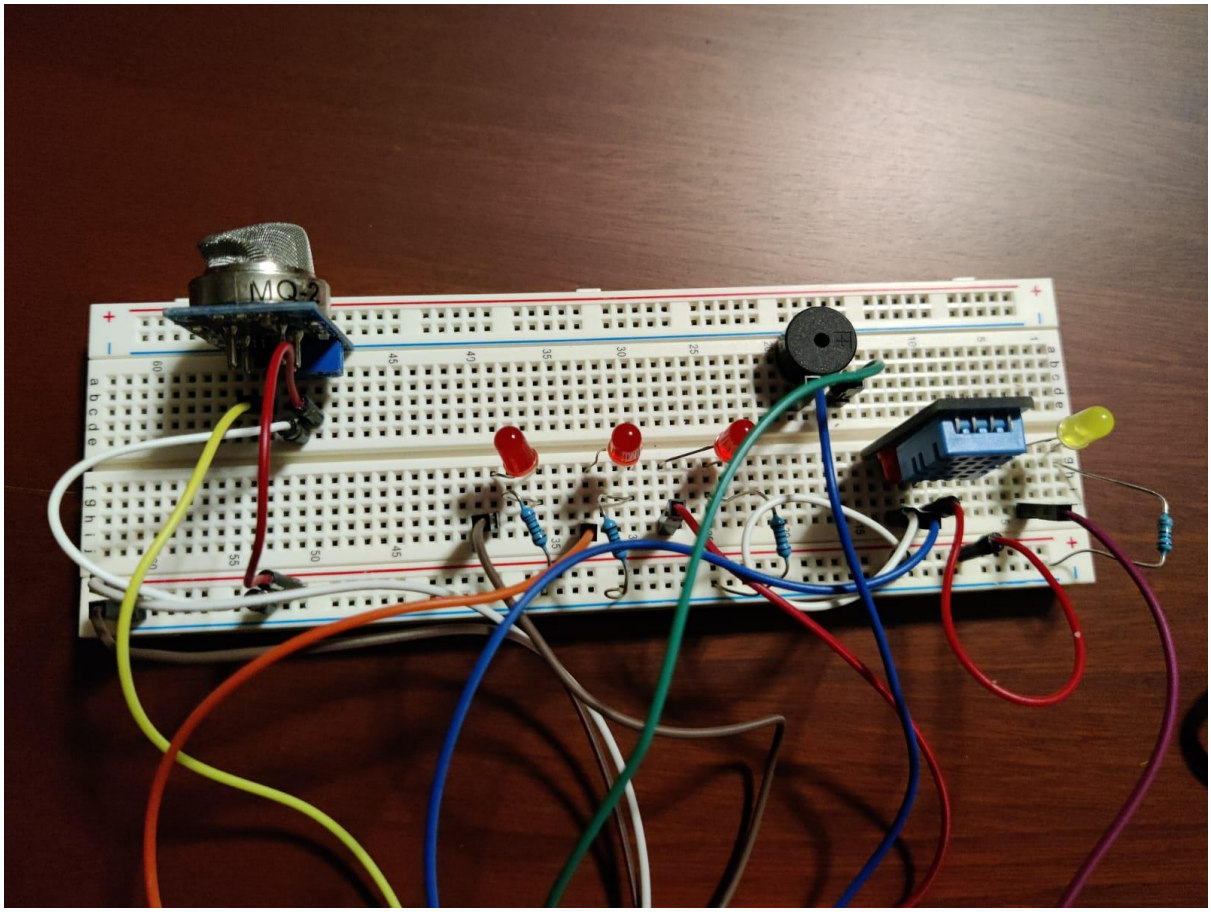
visualizarlos y será capaz de editarlos o eliminarlos con los respectivos botones que alteran directamente la base de datos. Estos escenarios son agregados al recuadro existente desde la base de datos por medio de un "inflate" del LinearLayout designado, utilizando un diseño preestablecido denominado "escenario.xml".

Finalmente, al regresar al menú principal se pueden visualizar los escenarios existentes dentro de la base de datos, los mismos a los cuales se puede acceder haciendo un clic. Luego de realizar dicha acción, se redirigirá a la pantalla que contiene la información del escenario seleccionado. Esta información está siendo capturada en tiempo real por el dispositivo IoT y mostrada por su aplicación, entre las variables observables se encuentra la temperatura, humedad y concentración de Co2, además, también se pueden observar los datos característicos del escenario cono número de emergencia, ubicación y aforo. Para terminar, esta pantalla incorpora un TextView que muestra si alguna de los parámetros censados por el dispositivo IoT se encuentra en estado de alerta.

- **Dispositivo IoT**

Metodología

Aquí se muestran las conexiones del circuito



Como se puede observar se tienen conectados varios elementos electrónicos los cuales incluyen tres leds rojos, uno amarillo, un buzzer pasivo, un DHT11 un sensor de Monóxido de Carbono 4 resistencias de 330 ohmios y un ESP8266.

El encargado del procesamiento es el ESP8266 el cual va a recolectar la información de los sensores y envía la señal a los indicadores que en este caso son los LEDs y el buzzer.

Los 3 Leds rojos con para indicar alerta en los casos de temperaturas, humedades y concentraciones de monóxido muy elevadas. También se tiene el Led amarillo el cual indica si encendido que hay una carencia den internet. Finalmente, el Buzzer emite sonido cuando el ESP8266 ya no puede mandar datos a la nube y hay un estado de alerta.

Aquí se puede mostrar partes esenciales del código, primero tenemos las condiciones de los parámetros definiendo rangos de temperatura, humedad y calidad de aire, y configurando las respectivas alertas.

```
float tempe = dht2.readTemperature( );

if (tempe < 22.0){
  alerta2 = "muy frio";

}
if (tempe >= 22 && tempe<24){
  alerta2 = "frio";

}
if (tempe >= 24 && tempe<25){
  alerta2 = "confort";

}
if (tempe >= 25 && tempe<27){
  alerta2 = "caliente";
  digitalWrite(LED1, LOW);
}
if (tempe >= 27){
  alerta2 = "peligro";
  digitalWrite(LED1, HIGH);


  WiFiClient client;
  HTTPClient http; //Declare an object of class HTTPClient
    //Specify request destination
  tobesend = "http://api.callmebot.com/whatsapp.php?";
  tobesend = tobesend + "phone=+593939793519";
  tobesend = tobesend + "&text=La+temperatura+es+peligrosa";
  tobesend = tobesend + "&apikey=855410";
  http.begin(client,tobesend);
  int httpCode = http.GET(); //Send the request
  if (httpCode > 0)
      {
        //Check the returning coderr
       String payload = http.getString(); //Get the request response payload
       Serial.println(payload); //Print the response payload
       }
  http.end(); //Close connection

}
```

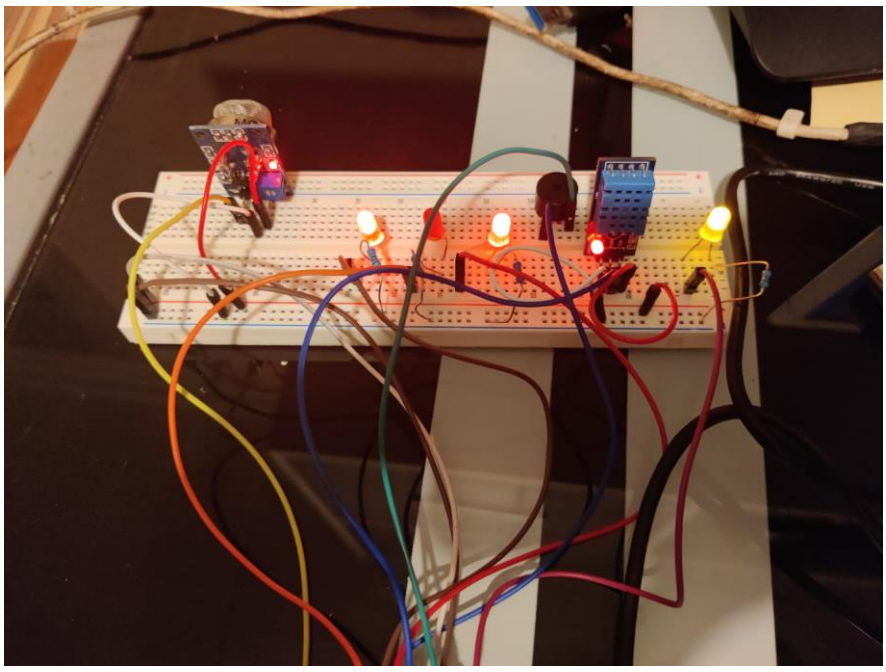
```
if (hume >= 33 && hume<40){
  alerta1 = "confortable";
  digitalWrite(LED2, LOW);
}
if (hume >= 50){
  alerta1 = "peligro electricidad";
  digitalWrite(LED2, HIGH);
  WiFiClient client;
  HTTPClient http; //Declare an object of class HTTPClient
    //Specify request destination
  tobesend = "http://api.callmebot.com/whatsapp.php?";
  tobesend = tobesend + "phone=+593939793519";
  tobesend = tobesend + "&text=La+humedad+es+alta+:+peligro+electricidad";
  tobesend = tobesend + "&apikey=855410";
  http.begin(client,tobesend);
  int httpCode = http.GET(); //Send the request
  if (httpCode > 0)
      {
        //Check the returning coderr
       String payload = http.getString(); //Get the request response payload
       Serial.println(payload); //Print the response payload
       }
  http.end(); //Close connection
}

if (analogSensor < 181 ){
  alerta3 = "pobre";
}
if (analogSensor > 181 && analogSensor< 225){
  alerta3 = "pobre";
}
if (analogSensor > 225 && analogSensor<300){
  alerta3 = "Muy mal";
  digitalWrite(LED3, LOW);
}
if (analogSensor > 300 && hume<350){
  alerta3 = "Peligro maximo";
  digitalWrite(LED3, HIGH);
  WiFiClient client;
  HTTPClient http; //Declare an object of class HTTPClient
    //Specify request destination
  tobesend = "http://api.callmebot.com/whatsapp.php?";
  tobesend = tobesend + "phone=+593939793519";
  tobesend = tobesend + "&text=La+concentracion+de+monoxido+es+peligrosa";
  tobesend = tobesend + "&apikey=855410";
  http.begin(client,tobesend);
  int httpCode = http.GET(); //Send the request
  if (httpCode > 0)
      {
```

Luego se muestra el envio de los datos a través de internet hacia la base de datos

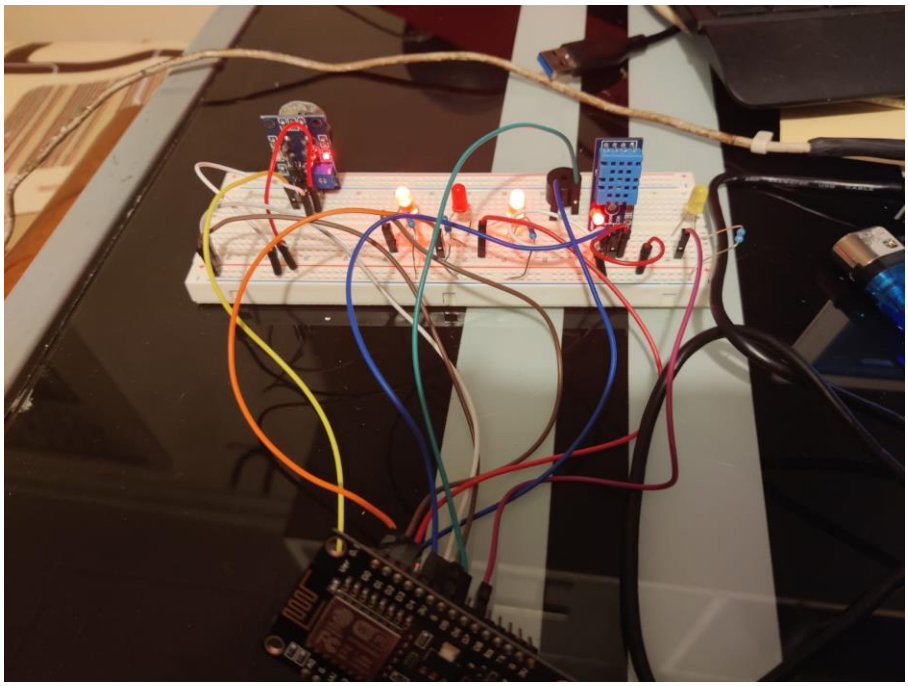
```
//Serial.println(hume);
if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 5000 || sendDataPrevMillis == 0)){
  sendDataPrevMillis = millis();
  // Write an Int number on the database path test/int
  if (Firebase.RTDB.setFloat(&fbdo, "Salón Eventos/temperatura", tempe)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
    tone(Passive_buzzer, LOW) ; //DO note 523 Hz
  }
  count++;

  // Write an Float number on the database path test/float
  if (Firebase.RTDB.setFloat(&fbdo, "Salón Eventos/calidadaire", analogSensor)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
    tone(Passive_buzzer, LOW) ; //DO note 523 Hz
  }
  if (Firebase.RTDB.setFloat(&fbdo, "Salón Eventos/humedad", hume)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
   if (Firebase.RTDB.setString(&fbdo, "Casa GYE/alertahumedad", alerta1)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
    tone(Passive_buzzer, LOW) ; //DO note 523 Hz
  }
  else{
    if(tempe >= 27){
      tone(Passive_buzzer, 523) ; //DO note 523 Hz
      delay (500);
      tone(Passive_buzzer, 587) ; //RE note ...
      delay (500);
    }

  }
  if (Firebase.RTDB.setString(&fbdo, "Casa GYE/alertatemperatura", alerta2)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
  if (Firebase.RTDB.setString(&fbdo, "Casa GYE/alertacalidad", alerta3)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
}
if(WiFi.status() != WL_CONNECTED){
```

Resultados

A continuación, se muestra uno de los casos en el cual no hay internet en el dispositivo por lo que se enciende el LED amarillo.



Este escenario muestra que hay 2 alertas ya que se han encendido 2 Leds rojos los cuales indican una alerta tanto en temperatura (Led izquierdo) y una alerta de humedad (sensor derecho)



Aquí se puede mostrar la base de datos la que constantemente se encuentra actualizándose.

Conclusiones

- Se realizó el despliegue de un dispositivo IoT con sensores de temperatura, humedad y CO2 para la monitorización de variables ambientales designadas en espacios determinados mediante una aplicación móvil que alerta en tiempo real la calidad de aire interior.
- La integración de las tecnologías embebidas como el módulo ESP8266 conectado vía Wifi permitió una comunicación efectiva en tiempo real mediante una base de datos Real Time como Firebase para controlar los niveles de una calidad de aire deficiente.
- Se focalizó la importancia de administrar de forma inteligente escenarios definidos para incrementar la calidad de aire que garanticen bioseguridad y un entorno saludable en el contexto laboral, doméstico y educativo.

Presupuesto

Cabe destacar que los elementos para la elaboración del hardware fueron brindados sin costo al equipo siendo propiedad de la Máster, sin embargo, se detalla una lista de los precios aproximados para cada componente.

Arduino Uno	$29.00
ESP8266	$8.00
DHT-11	$6.00
MQ-2	$8.50
Baterías de Litio	$13.00
Protoboard	$5.00
Cables Jumpers	$4.00
Total	$73.50

Apéndice

- **Código fuente del hardware:**

https://github.com/amst-fiec/1T2022-Monitor-de-calidad-del-aire-v1.0/tree/main/Codigo%20Arduino

- **Código fuente de la aplicación:**

https://github.com/amst-fiec/1T2022-Monitor-de-calidad-del-aire-v1.0

Referencias

https://www.aeroqual.com/blog/why-monitor-indoor-air-quality

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

https://firebase.google.com/docs