



Reto

Jesús Ramírez Delgado - A01274723

Arturo Sánchez Rodríguez - A01275427

Eliuth Balderas Neri - A01703315

Instituto Tecnológico de Monterrey

Grupo 570

Implementación de internet de las cosas

Profesores:

Alba Adriana Romero García

Emmanuel Torres Rios

Fecha de entrega: 14 de junio de 2023

**Enlace GitHub 1:** <https://github.com/A01274723/ProyectoIoT.git>

**Enlace GitHub 2 (extra):** [https://github.com/BalnerEli/IoT\\_TC1004B.git](https://github.com/BalnerEli/IoT_TC1004B.git)

## **Introducción. -**

En el ámbito de la domótica, la implementación de sistemas informáticos basados en microcontroladores conectados a Internet ha revolucionado la forma en que interactuamos con nuestro entorno. En este contexto, se plantea un desafío que combina los conocimientos y habilidades adquiridos a lo largo de nuestras interacciones con profesores y compañeros.

El objetivo principal de este bloque consiste en diseñar e implementar un sistema domótico para una "Smart City" o un "Smart Campus". Este sistema se basará en la utilización de microcontroladores interconectados en red, permitiendo la adquisición de datos mediante sensores y el control de actuadores en tiempo real.

El reto planteado incluye una serie de requerimientos que guiarán el desarrollo del proyecto. Entre ellos, se destaca la necesidad de implementar un conjunto de cinco microcontroladores que estarán conectados en red y programados para intercambiar información a través de diversos protocolos, como TCP, UDP, HTTP, MQTT, IFTTT, entre otros. La selección de estos protocolos estará basada en los requerimientos específicos de la aplicación.

Además, se deberá diseñar e implementar una arquitectura de información que permita recopilar los datos adquiridos, facilitar la comunicación entre los dispositivos de la red domótica y generar acciones de control en tiempo real. Esta arquitectura podrá ser alojada ya sea en un servicio "on premise" o en la nube, proporcionando flexibilidad y escalabilidad al sistema.

La programación, interconexión y puesta en operación de los dispositivos dentro de la red domótica será fundamental para implementar las funcionalidades específicas requeridas, tales como la adquisición de datos, el control de actuadores y la generación de información para los servicios en la red.

Además de los requisitos mencionados, se analizarán los requerimientos, se planificará y administrará el proceso de ejecución de la implementación del sistema domótico. Este enfoque garantizará una ejecución eficiente y exitosa del proyecto.

A lo largo de la implementación, se enfrentarán mini-retos que complementarán la especificación de referencia inicial. Estos mini-retos requerirán el diseño de algoritmos y servicios específicos para mejorar el sistema, adaptando los requerimientos a situaciones particulares. Algunos ejemplos de estos mini-retos podrían ser el control de encendido y apagado de LEDs y luminarias, la generación de señales de sincronización y reloj en la red domótica, la incorporación de sensores adicionales y la gestión avanzada de consumo eléctrico en los dispositivos de la red domótica mediante técnicas como el "deepSleep".

En resumen, este bloque representa una oportunidad para aplicar y poner en práctica los conocimientos adquiridos en un contexto real, desarrollando un sistema domótico

interconectado que cumpla con los requerimientos planteados y supere los desafíos adicionales que se presenten a lo largo del proceso de implementación.

En este repositorio se encuentra paso a paso los códigos empleados para la realización del reto IOT. Es preciso que para el desarrollo del reto, se tenga instalado el ide de Arduino, el cuál se encuentra en la siguiente liga: <https://www.arduino.cc/en/main/software>. La versión que se utiliza en este proyecto es la versión 1.8.19. Si se requiere mayor información para el desarrollo del proyecto, se puede consultar en [wiki](#).

Adicionalmente, las siguientes librerías son necesarias para la compilación del programa:

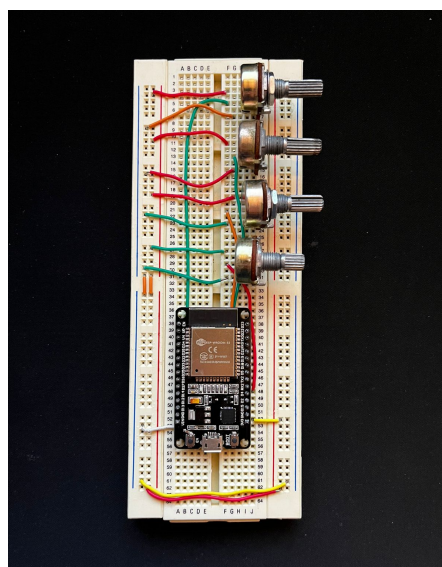
- DHT.h
- WiFi.h
- Firebase\_ESP\_Client.h
- Arduino.h

En cuanto a los materiales que se necesitan para el desarrollo del reto son:

- Protoboard
- ESP32
- 4 Potenciómetros
- Alambres de protoboard

Se busca diseñar e implementar un prototipo de un sistema digital capaz de obtener datos mediante el uso de sensores, procesarlos y depositarlos como información en una base de datos en una plataforma en internet (firebase), la cuál nos permitirá hacer un análisis integral de los datos.

Para ello, en primer lugar, se debe realizar el circuito, el cuál nos permitirá medir las variables. En este caso, se utilizarán potenciómetros. Para la construcción del circuito se utilizó un protoboard, alambres de protoboard, ESP32, y cuatro potenciómetros, el cuál se puede observar a continuación en la Figura 1.



**Figura 1.** Circuito elaborado por el equipo

Por otro lado, se debe tener un conocimiento básico sobre el uso de bases de datos. En este caso se usará Firebase, la cuál se tiene que configurar adecuadamente como se muestra en [https://github.com/etorresr/TC1004B\\_IoT/wiki/Firebase-configuraci%C3%B3n](https://github.com/etorresr/TC1004B_IoT/wiki/Firebase-configuraci%C3%B3n)

Una vez obtenida nuestra base de datos, se debe configurar adecuadamente el código para poderlo comunicar el ESP32 con la base de datos (Firebase) y así mandar los datos en tiempo real. Estos pasos se encuentran en [https://github.com/etorresr/TC1004B\\_IoT/wiki/ESP32-y-Firebase](https://github.com/etorresr/TC1004B_IoT/wiki/ESP32-y-Firebase). Como se puede ver en el link adjunto, es necesario instalar las librerías Wifi.h y PubSubClient.h. Adicionalmente, se debe iniciar nuestra base de datos con los parámetros de configuración proporcionados en la configuración general del proyecto de nuestra base de datos.

```
// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBdNJ82PipRL0HfpVw5sqidBaZv65xWNOA",
  authDomain: "tc1004b-48578.firebaseio.com",
  databaseURL: "https://tc1004b-48578-default-rtdb.firebaseio.com",
  projectId: "tc1004b-48578",
  storageBucket: "tc1004b-48578.appspot.com",
  messagingSenderId: "510427288538",
  appId: "1:510427288538:web:f16e35a454faaef0c966e3",
  measurementId: "G-ZJGXX9HFJV"
};
```

**Figura 2.** Parámetros mostrados en la configuración general de una base de datos en Firebase

Una vez realizada la configuración entre nuestro ESP32 y Firebase, se deben hacer pruebas para ver que efectivamente se están enviando y recibiendo los datos en tiempo real.

Posteriormente, se debe realizar una lectura de datos de sensores usando los periféricos del ESP32. Para ello, se deben conectar los potenciómetros al ESP32, utilizando los pines GPIO. Posteriormente se deben configurar los potenciómetros para comenzar a hacer la lectura de los datos y luego almacenar y procesar los datos. El código completo se encuentra en este repositorio. (ESP32Reto.ino)

## Código Arduino (Potenciómetros). -

```
ESP32Reto | Arduino 1.8.19
File Edit Sketch Tools Help

ESP32Reto

#include <AzureIoTHub.h>
#include <Esp32MQTTClient.h>

#include <WiFi.h>
#include <PubSubClient.h>
#include <Firebase_ESP_Client.h>
#include <addons/TokenHelper.h>

// SSID y password de red abierta
const char* ssid = "INFINITUM8188";
const char* password = "Nm4Bz8Ec2t";

// api key del la base de datos en tiempo real
#define API_KEY "AIzaSyBdNJ82PipRL0HfpVw5sqidBaZv65xWNOA"

// URL del la base de datos en tiempo real
#define DATABASE_URL "https://tc1004b-48578-default-rtdb.firebaseio.com"

// Para definir el proyecto fbdo de firebase y darle autorizacion
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;

int potentiometerValues[4] = {0}; // Este arreglo sirve para guardar el valor de cada potenciometro.

void setup() {
  Serial.begin(115200);
  delay(10);
  setup_wifi();

  // Se definen los pines en los que se encuentran conectados los potenciometros al ESP32
  pinMode(A4, INPUT);
  pinMode(A5, INPUT);
  pinMode(A6, INPUT);
  pinMode(A7, INPUT);
}

// Se hace la configuracion y setup de la red
void setup_wifi() {
  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());
}
```



```
ESP32Reto | Arduino 1.8.19
File Edit Sketch Tools Help

ESP32Reto $

config.api_key = API_KEY;
config.database_url = DATABASE_URL;

// Se hace el signup en firebase
if (Firebase.signUp(&config, &auth, "", "")) {
  Serial.println("Signup successful");
  Firebase.reconnectWiFi(true); // Movido aquí
} else {
  Serial.printf("Signup failed. Reason: %s\n", config.signer.signupError.message.c_str());
}

// Assign the callback function for the long running token generation task
config.token_status_callback = tokenStatusCallback; // see addons/TokenHelper.h

Firebase.begin(&config, &auth);
}

void loop() {
  if (Firebase.ready() && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)) {
    sendDataPrevMillis = millis();

    // Leer valores analógicos de los potenciómetros
    int pot1Value = analogRead(A4);
    int pot2Value = analogRead(A5);
    int pot3Value = analogRead(A6);
    int pot4Value = analogRead(A7);

    potentiometerValues[0] = pot1Value;
    potentiometerValues[1] = pot2Value;
    potentiometerValues[2] = pot3Value;
    potentiometerValues[3] = pot4Value;

    // Imprimir los valores en el monitor serial
    Serial.print("Potentiometer 1: ");
    Serial.println(pot1Value);
    Serial.print("Potentiometer 2: ");
    Serial.println(pot2Value);
    Serial.print("Potentiometer 3: ");
    Serial.println(pot3Value);
    Serial.print("Potentiometer 4: ");
    Serial.println(pot4Value);

    // Autenticación con Firebase
    if (Firebase.ready()) {
      // En este bucle se van subiendo los datos de los potenciómetros a la base de datos en tiempo real de firebase
      for (int i = 0; i < 4; i++) {
        String path = "potentiometers/pot" + String(i + 1);
        if (Firebase.RTDB.setInt(&fbdo, path.c_str(), potentiometerValues[i])) {
          Serial.println("Potentiometer " + String(i + 1) + " value sent successfully");
          Serial.println("PATH: " + fbdo.dataPath());
          Serial.println("TYPE: " + fbdo.dataType());
        } else {
          Serial.println("Failed to send potentiometer " + String(i + 1) + " value");
          Serial.println("REASON: " + fbdo.errorReason());
        }
      }
    }
  }
}
```

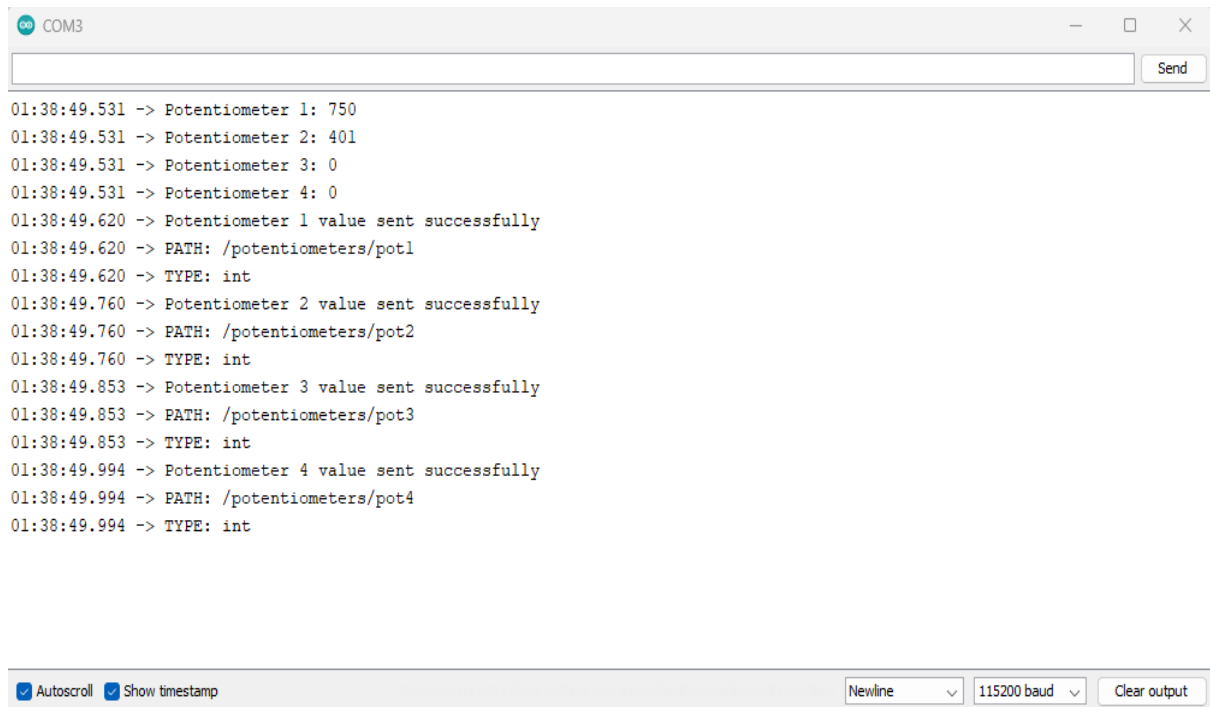
**Figura 4, 5.** La implementación de código arduino tiene como objetivo leer los valores de cuatro potenciómetros conectados a un Esp32 y enviar los valores a una base de datos en tiempo real, utilizando FireBase.

```
ESP32Reto | Arduino 1.8.19
File Edit Sketch Tools Help

Done uploading.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 5041.2 kbit/s)...
Hash of data verified.
Compressed 18656 bytes to 12053...
Writing at 0x00001000... (100 %)
Wrote 18656 bytes (12053 compressed) at 0x00001000 in 0.1 seconds (effective 1008.4 kbit/s)...
Hash of data verified.
Compressed 1074080 bytes to 625534...
Writing at 0x00010000... (2 %)
Writing at 0x00014000... (5 %)
Writing at 0x00018000... (7 %)
Writing at 0x0001c000... (10 %)
Writing at 0x00020000... (12 %)
Writing at 0x00024000... (15 %)
Writing at 0x00028000... (17 %)
Writing at 0x0002c000... (20 %)
Writing at 0x00030000... (23 %)
Writing at 0x00034000... (25 %)
Writing at 0x00038000... (28 %)
Writing at 0x0003c000... (30 %)
Writing at 0x00040000... (33 %)
Writing at 0x00044000... (35 %)
Writing at 0x00048000... (38 %)
Writing at 0x0004c000... (41 %)
Writing at 0x00050000... (43 %)
Writing at 0x00054000... (46 %)
Writing at 0x00058000... (48 %)
Writing at 0x0005c000... (51 %)
Writing at 0x00060000... (53 %)
Writing at 0x00064000... (56 %)
Writing at 0x00068000... (58 %)
Writing at 0x0006c000... (61 %)
Writing at 0x00070000... (64 %)
Writing at 0x00074000... (66 %)
Writing at 0x00078000... (69 %)
Writing at 0x0007c000... (71 %)
Writing at 0x00080000... (74 %)
Writing at 0x00084000... (76 %)
Writing at 0x00088000... (79 %)
Writing at 0x0008c000... (82 %)
Writing at 0x00090000... (84 %)
Writing at 0x00094000... (87 %)
Writing at 0x00098000... (89 %)
Writing at 0x0009c000... (92 %)
Writing at 0x000a0000... (94 %)
Writing at 0x000a4000... (97 %)
Writing at 0x000a8000... (100 %)
Wrote 1074080 bytes (625534 compressed) at 0x00010000 in 9.1 seconds (effective 943.9 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 2048.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

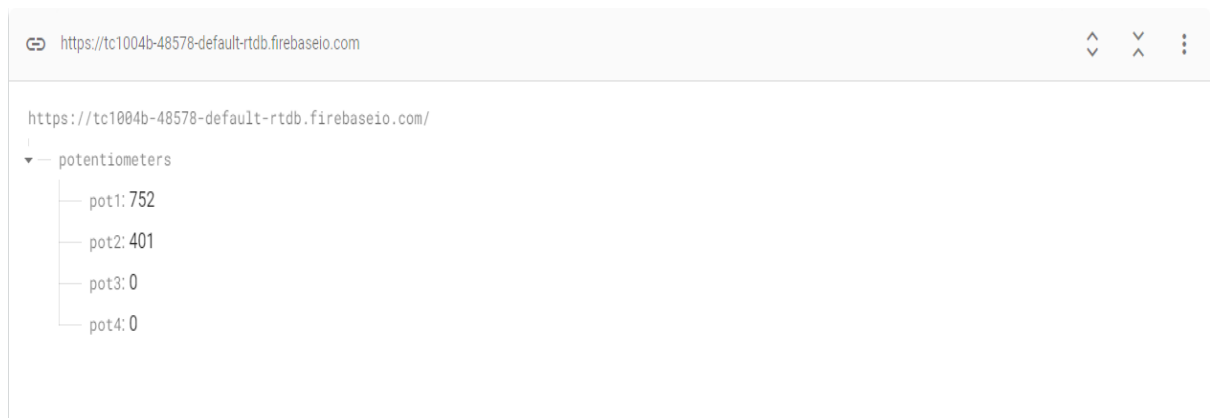
**Figura 6.** Al implementar el código en el Esp32 no solo podemos verificar el funcionamiento a través de FireBase, si no igual podemos utilizar el Monitor Serie, así vemos que los datos son correctos y también podemos verificar que manda los mismos datos a FireBase.



```
01:38:49.531 -> Potentiometer 1: 750
01:38:49.531 -> Potentiometer 2: 401
01:38:49.531 -> Potentiometer 3: 0
01:38:49.531 -> Potentiometer 4: 0
01:38:49.620 -> Potentiometer 1 value sent successfully
01:38:49.620 -> PATH: /potentiometers/pot1
01:38:49.620 -> TYPE: int
01:38:49.760 -> Potentiometer 2 value sent successfully
01:38:49.760 -> PATH: /potentiometers/pot2
01:38:49.760 -> TYPE: int
01:38:49.853 -> Potentiometer 3 value sent successfully
01:38:49.853 -> PATH: /potentiometers/pot3
01:38:49.853 -> TYPE: int
01:38:49.994 -> Potentiometer 4 value sent successfully
01:38:49.994 -> PATH: /potentiometers/pot4
01:38:49.994 -> TYPE: int
```

☒ Autoscroll ☒ Show timestamp Newline 115200 baud Clear output

El serial monitor en arduino es una herramienta de depuración y visualización que permite la comunicación bidireccional entre el arduino y el ordenador a través del puerto serie.



```
https://tc1004b-48578-default-rtdb.firebaseio.com/
├── potentiometers
│   ├── pot1: 752
│   ├── pot2: 401
│   ├── pot3: 0
│   └── pot4: 0
```

**Figura 7 y 8.** Podemos así verificar que los datos mandados son los mismos y así podemos ver que el código y el FireBase coinciden, mandando y recibiendo los mismos datos.



## FireBase. -

Firebase es una plataforma de desarrollo de aplicaciones móviles y web ofrecida por Google. Sus funciones principales incluyen una base de datos en tiempo real, autenticación de usuarios, almacenamiento en la nube, alojamiento web, mensajería en la nube, analítica, pruebas y calidad, y funciones en la nube. Firebase simplifica el desarrollo, implementación y crecimiento de aplicaciones al proporcionar una variedad de servicios y herramientas para facilitar estas tareas.

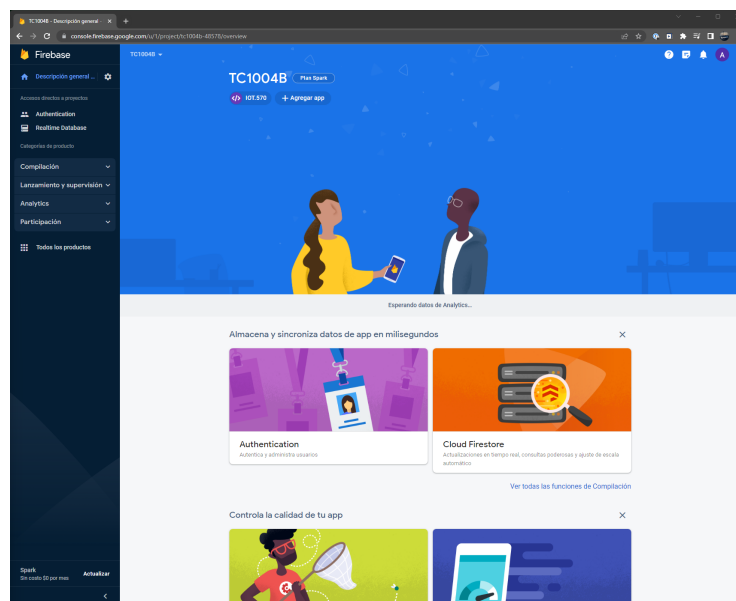


Figura 9. Página de inicio de FireBase.

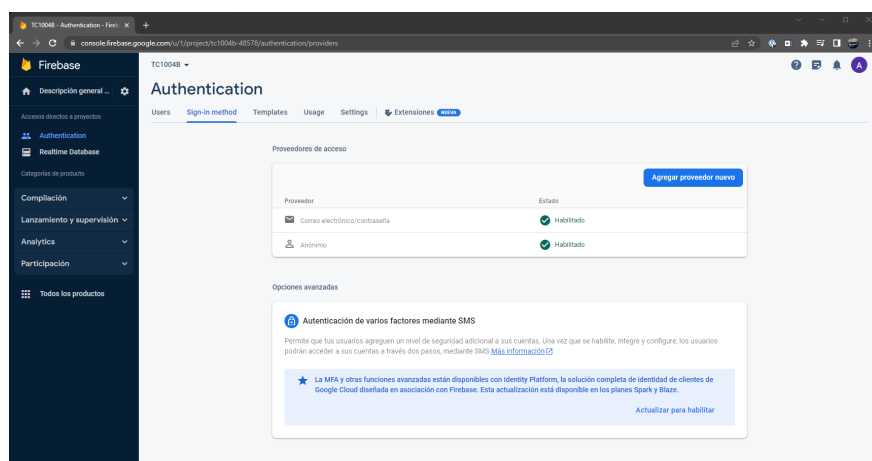
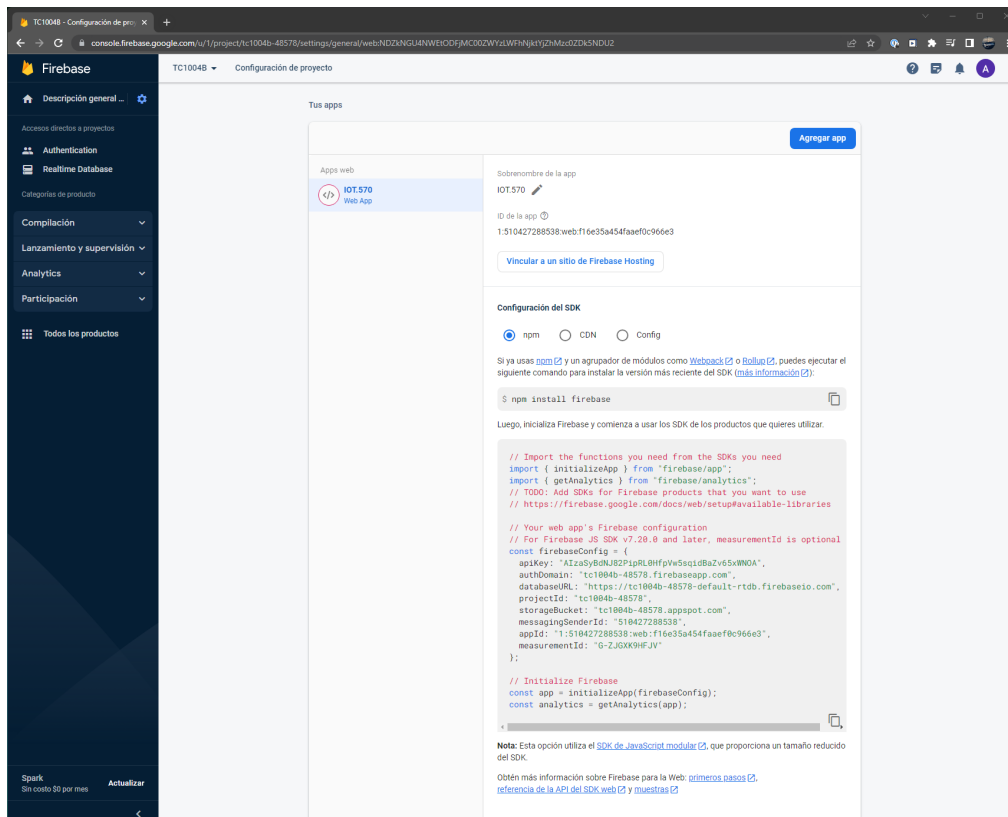
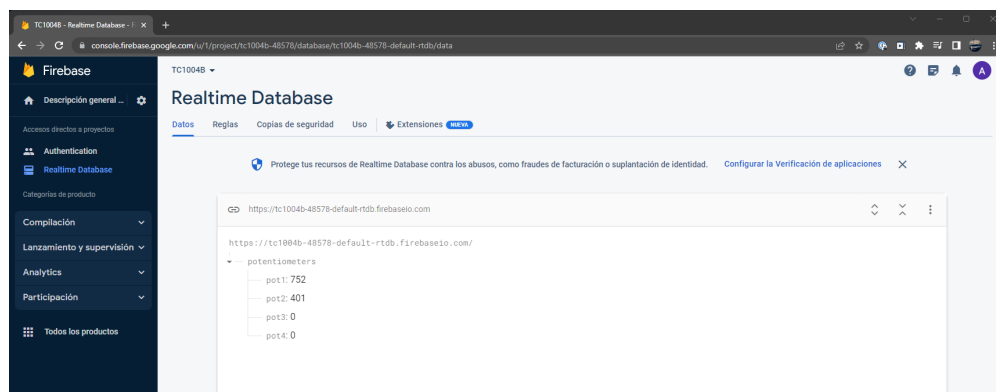


Figura 10. La autenticación en firebase sirve para identificar y autorizar a los usuarios que acceden a una aplicación o servicio, proporcionando un sistema seguro para controlar el acceso a los datos y funcionalidad de la aplicación.



**Figura 11.** La configuración sirve para utilizar los servicios de Firebase en una aplicación, se establecen las conexiones y credenciales necesarias para que la aplicación pueda interactuar correctamente con los servicios de Firebase, sirve para la autorización y seguridad, conexión con servicios específicos, personalización y ajustes de la aplicación, etc.



**Figura 12.** Firebase Realtime Database es un servicio de Firebase que proporciona una base de datos en tiempo real en la nube, permite almacenar y sincronizar datos en tiempo real entre diferentes clientes, como aplicaciones web, móviles o de escritorio.

**Configuración de página web. -**

Originalmente se iba a implementar App Inventor para crear una pequeña aplicación que se conecte a Firebase. Sin embargo para evitar problemas con App Inventor, relacionados a nuestro sistema operativo, se optó por desarrollar una aplicación web. La aplicación se desarrolló con HTML y CSS para la estructura y diseño de la página y Java Script para la lógica de la misma. El proyecto se desarrolló “por capas” es decir que hay un archivo correspondiente a cada elemento de la aplicación para poder tener un código mucho mas limpio, manipulable y sea mas sencillo hacer cambios y mantenimiento.

## Código

```
> index.html > html > body > header > h1
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta name="viewport" content="width=device-width, initial-scale=1.0">
5      <link rel="stylesheet" href="style.css">
6      <title> Internet de las Cosas</title>
7      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
8      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>
9      <script src="script.js"></script>
10 </head>
11 <body>
12     <header>
13         <br>
14         <br>
15         <br>
16         <div class="logoTec">
17             
18         </div>
19         <div class="espacio"></div>
20         <h1> Instituto Tecnológico y de Estudios Superiores de Monterrey </h1>
21         <div class="espacio"></div>
22         <h2> Reto </h2>
23         <div class="espacio"></div>
24         <div class="div1">
25             <p class="nombres">
26                 Arturo Sanchez Rodriguez
27                 <br>
28                 <br>
29                 Jesus Ramirez Delgado
30                 <br>
31                 <br>
32                 Eliuth Balderas Neri
33             </p>
34         </div>
35         <div class="espacio"></div>
36         <h3> Proyecto Final </h3>
37     </header>
38     <main>
39         <h4>Descripción Proyecto</h4>
40         <article class="articleStyle">
41             <p>Mediante una placa de Arduino ESP32 se conectaron varios modulos para poder realizar diferentes medidas.</p>
42             <p>Se conectaron 4 potenciómetros (por falta de material).</p>
43             <p>Una vez en funcionamiento, se conecto a una base de datos, en este caso se implementó Firebase, y se conecto a el IDE</p>
44             <p>de Arduino. Finalmente se programo esta App Web sencilla, que obtiene los datos de Firebase y tu puedes consultar en</p>
45             <p>tiempo real los datos de cada módulo conectado a la placa ESP32.</p>
46             <!-- Más contenido de texto -->
47         </article>
48         <div class="espacio"></div>
49         <div class="wrap">
50             <button class="button" onclick="window.location.href='botones.html'">Probar</button>
51         </div>
52     </main>
53     <footer>
54         <p> Gracias </p>
55     </footer>
56 </body>
57 </html>
```

**Figura 13.** Archivo **index.html** donde se desarrolla la estructura de la página principal. Esta misma se conecta a **botones.html**.

```

# style.css > % header
1  html, body {
2      height: 100%;
3      margin: 0;
4      padding: 0;
5  }
6
7  header {
8      background-color: black;
9      width: 100%;
10     height: 100%;
11 }
12
13 .espacio {
14     margin-top: 50px;
15     margin-bottom: 50px;
16 }
17
18 .logoTec {
19     display: flex;
20     justify-content: center;
21     align-items: center;
22 }
23
24 .logoTec img{
25     display: block;
26     max-width: 100%;
27     max-height: 100%;
28 }
29
30 h1 {
31     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
32     color: white; /* Cambia el color a blanco */
33     font-size: 30px; /* Aumenta el tamaño de la fuente a 24 píxeles */
34     text-align: center; /* Centra el texto horizontalmente */
35 }
36
37 h2 {
38     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
39     color: white; /* Cambia el color a blanco */
40     font-size: 24px; /* Aumenta el tamaño de la fuente a 24 píxeles */
41     text-align: center; /* Centra el texto horizontalmente */
42 }
43
44 .nombres {
45     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
46     color: white; /* Cambia el color a blanco */
47     font-size: 18px; /* Aumenta el tamaño de la fuente a 24 píxeles */
48     text-align: center; /* Centra el texto horizontalmente */
49 }
50
51
52 h3 {
53     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
54     color: white; /* Cambia el color a blanco */
55     font-size: 18px; /* Aumenta el tamaño de la fuente a 24 píxeles */
56     text-align: center; /* Centra el texto horizontalmente */
57 }
58
59 h4 {
60     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
61     color: black; /* Cambia el color a blanco */
62     font-size: 24px; /* Aumenta el tamaño de la fuente a 24 píxeles */
63     text-align: center; /* Centra el texto horizontalmente */
64 }
65
66 .articleStyle {
67     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
68     color: black; /* Cambia el color a blanco */
69     font-size: 18px; /* Aumenta el tamaño de la fuente a 24 píxeles */
70     text-align:center; /* Centra el texto horizontalmente */
71 }
72
73 .boton {
74     font-family: 'Helvetica Neue', Arial, sans-serif; /* Cambia la fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
75     color: white; /* Cambia el color a blanco */
76     font-size: 30px; /* Aumenta el tamaño de la fuente a 24 píxeles */
77     text-align: center; /* Centra el texto horizontalmente */
78 }
79
80 html, body {
81     height: 100%;
82 }
83
84 .wrap {
85     height: 100%;
86     display: flex;
87     align-items: center;
88     justify-content: center;
89 }
90
91 .button {
92     width: 140px;
93     height: 45px;
94     font-family: 'Roboto', sans-serif;
95     font-size: 11px;
96     text-transform: uppercase;
97     letter-spacing: 2.5px;
98     font-weight: 500;
99     color: #0000;
100    background-color: #fff;
101    border: none;
102    border-radius: 45px;
103    box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.1);
104    transition: all 0.3s ease 0s;
105    cursor: pointer;
106    outline: none;
107 }
108
109 .button: hover {
110     background-color: rgb(33, 33, 186);
111     box-shadow: 0px 15px 20px rgba(46, 61, 229, 0.4);
112     color: #fff;
113     transform: translateY(-7px);
114 }
115

```

**Figura 14.** Archivo **style.css** encargada de asignarle un estilo/diseño a **index.html** y aspectos generales de la aplicación.

```

1  <!DOCTYPE html>
2  <head>
3      <meta name="viewport" content="width=device-width, initial-scale=1.0">
4      <title>Internet de las Cosas</title>
5      <link rel="stylesheet" href="styleBotones.css">
6      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
7      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>
8      <script src="script.js"></script>
9  </head>
10 <body>
11     <h1>Medidas</h1>
12     <div></div>
13     <a href="index.html"><span></span>Regresar</a>
14     <div class="wrap">
15         <button class="button" id="myButton">Mostrar datos</button>
16         <p class="escritura" id="myData"></p>
17     </div>i≠
18 </body>
19 </html>

```

**Figura 15.** Archivo **botones.html** al cual accedes por medio de **index.html** aquí sencillamente por medio de un botón se comunica a firebase y obtiene los últimos datos de la base de datos en tiempo real de cada potenciómetro.

```

styleBotones.css ? :a:showBefore
1 {
2   font-family: "Helvetica Neue", Arial, sans-serif; /* Cambia la Fuente a 'Helvetica Neue', Arial o cualquier otra fuente sans-serif */
3   color: #blanchetdaband; /* Cambia el color a blanco */
4   font-size: 18px; /* Aumenta el tamaño de la fuente a 24 píxeles */
5   text-align: center; /* Centra el texto horizontalmente */
6 }
7
8 .espacio {
9   margin-top: 50px;
10  margin-bottom: 50px;
11 }
12
13 body {
14   background-color: #black;
15 }
16
17 html, body {
18   height: 100%;
19 }
20
21 .wrap {
22   height: 100%;
23   display: flex;
24   align-items: center;
25   justify-content: center;
26   margin-top: -150px;
27 }
28
29 .button {
30   width: 200px; /* Aumentar el ancho del botón */
31   height: 40px; /* Aumentar la altura del botón */
32   font-family: "Roboto", sans-serif;
33   font-size: 14px; /* Aumentar el tamaño de la fuente */
34   text-transform: uppercase;
35   letter-spacing: 2.5px;
36   font-weight: bold;
37   color: #fff;
38   background-color: #fff;
39   border: none;
40   border-radius: 40px;
41   box-shadow: 0px 0px 15px #rgb(0, 0, 0, 0.1);
42   transition: all 0.3s ease-in;
43   cursor: pointer;
44   outline: none;
45   margin-right: 10px; /* Espacio entre botones */
46 }
47
48 .button:last-child {
49   margin-right: 0; /* Eliminar margen derecho del último botón */
50 }
51
52 .button:hover {
53   background-color: #rgb(33, 33, 186);
54   box-shadow: 0px 15px 20px #rgb(46, 61, 229, 0.4);
55   color: #fff;
56   transform: translateX(-7px);
57 }
58
59 {
60   margin: 0;
61   padding: 0;
62   background: #f3f3f3;
63 }
64
65 a {
66   position: absolute;
67   top: 1%;
68   left: 2.5%;
69   transform: translate(0, 0);
70   width: 180px;
71   height: 50px;
72   background: #f3f3f3;
73   text-decoration: none;
74   text-transform: uppercase;
75   text-align: center;
76   line-height: 50px;
77   color: #blanchetdaband;
78   font-size: 20px;
79   font-family: verdana;
80   letter-spacing: 4px;
81 }
82
83 a:before,
84 a:after,
85 span:before,
86 span:after
87 {
88   content: "";
89   position: absolute;
90   width: 180px;
91   height: 180px;
92   background: #fff;
93   transition: 1s;
94   mix-blend-mode: hue;
95 }
96
97 a:before
98 {
99   top: -2px;
100  left: -2px;
101 }
102
103 a:after
104 {
105   top: -2px;
106   right: -2px;
107 }
108
109 span:before
110 {
111   bottom: -2px;
112   left: -2px;
113 }
114
115 span:after
116 {
117   bottom: -2px;
118   right: -2px;
119 }
120
121 a:ho:ver:be:fore,
122 a:ho:ver:a:f:ter,
123 a:ho:ver:s:p:a:n:b:e:fore,
124 a:ho:ver:s:p:a:n:a:f:ter
125 {
126   width: calc(180px/2);
127   height: calc(50px/2);
128 }
129
130 .escritura {
131   font-family: "Roboto", sans-serif;
132   color: #blanchetdaband; /* Cambia el color a blanco */
133   font-size: 18px; /* Aumenta el tamaño de la fuente a 24 píxeles */
134   text-align: center; /* Centra el texto horizontalmente */
135 }

```

**Figura 16.** Archivo **styleBotones.css** para asignar un estilo/diseño a **botones.html**.

```

1 window.addEventListener('load', (event) => {
2   // Your web app's Firebase configuration
3   var firebaseConfig = {
4     apiKey: "AIzaSyBdNJ82PipRL0HfpVw5sqidBaZv65xwN0A",
5     authDomain: "tc1004b-48578.firebaseio.com",
6     databaseURL: "https://tc1004b-48578-default-rtdb.firebaseio.com",
7     projectId: "tc1004b-48578",
8     storageBucket: "tc1004b-48578.appspot.com",
9     messagingSenderId: "510427288538",
10    appId: "1:510427288538:web:f16e35a454faef0c966e3",
11    measurementId: "G-ZJGXX9HFJV"
12  };
13
14  // Initialize Firebase
15  firebase.initializeApp(firebaseConfig);
16
17  //Funcion para el boton "Mostrar datos"
18
19  var database = firebase.database();
20
21  document.getElementById('myButton').addEventListener('click', function() {
22    var ref = database.ref('/');
23
24    ref.on('value', function(snapshot) {
25      document.getElementById('myData').textContent = JSON.stringify(snapshot.val(), null, 2);
26    });
27  });
28 });

```

**Figura 17.** Archivo **script.js** donde se conecta firebase por medio de las credenciales correspondientes y se crea una pequeña función para el correcto funcionamiento del botón “mostrar datos” en **botones.html** para que con la acción de presionar click, obtenga los datos de la base de datos en tiempo real de firebase.

Una vez corriendo nativamente, la página arrojó un correcto funcionamiento y obtiene exitosamente los datos requeridos.



**Figura 18.** “Header” de la página principal (index.html) esta contiene una presentación de los integrantes del equipo.



**Figura 19.** “Body” de la página principal (index.html) el cual contiene una descripción breve del proyecto y un botón que comunica la segunda página (botones.html).



**Figura 20.** En esta página secundaria (botones.html) tiene la capacidad de regresar a la pagina principal (index.html) por medio del botón “regresar” esta página es la encargada de mostrar los últimos datos de cada potenciómetro de nuestra base de datos en tiempo real de Firebase. Al presionar el botón “Mostrar Datos” se conecta a Firebase y muestra los datos deseados.

## Conclusiones. -

Utilizar Firebase para analizar datos recolectados por sensores como en este caso el uso de potenciómetros, ofrece varias ventajas. En primer lugar, nos permite un almacenamiento de datos en tiempo real, lo que facilita el análisis de datos y toma de decisiones basada en datos actualizados. Por otro lado, aun en este caso únicamente se utilizaron 5 variables, Firebase es una plataforma escalables, puesto a que permite manejar grandes volúmenes de datos, lo que también nos otorga un buen rendimiento. Por otro lado, Firebase es una plataforma bastante fácil de usar, ya que es intuitiva, lo que permite gestionar y analizar los datos de una manera sencilla. Adicionalmente, como Firebase guarda los datos en la nube, permite que se tenga un acceso fácil en cualquier momento desde cualquier lugar. Finalmente, como se vio en el reto, Firebase se puede integrar con otras herramientas, lo que hace que tenga muchas más posibilidades para analizar los datos recolectados.

## Referencias. -