



Tecnológico de Monterrey

Entrega final de proyecto Semana Tec TC1001S

Fernando Gutiérrez - A01424790

Alexia López García -A01639419

Luis Yair Ruiz Rodríguez - A01639766

Oliver Josel Hernández Rebollar - A01641922

Gustavo José Ortiz Zepeda - A01637220

Fecha de entrega:
22 de Marzo del 2023

Objetivo del prototipo

El prototipo está concebido para ser una herramienta multifacética en la recolección y manejo de información ambiental. A través de la interacción de sensores específicos con un módulo NodeMCU, se facilita el monitoreo de variables como la temperatura, la humedad y la distancia en tiempo real. Este conjunto de tecnología no solo permite la adquisición de datos sino también su transmisión a una base de datos en la nube, permitiendo así el acceso y la gestión de la información desde cualquier lugar a través de Internet. Este sistema integrado podría tener aplicaciones prácticas en el monitoreo de espacios controlados, tales como invernaderos o almacenes, donde mantener ciertas condiciones ambientales es crítico. Al mismo tiempo, gracias a su conexión a la red, ofrece la posibilidad de interactuar y posiblemente controlar otros dispositivos conectados, lo que abre un abanico de posibilidades en la automatización del hogar o en sistemas de alerta para la seguridad. En el ámbito doméstico, los datos recabados por el prototipo podrían automatizar ciertas funciones de la casa, como regular la humedad o la temperatura. También podría, si se configura para tal fin, ofrecer un sistema de notificaciones y alertas, por ejemplo, avisando a los usuarios de una posible entrada no autorizada si se detecta un cambio en la medición de distancia que supere un umbral preestablecido.

Por último, el registro continuo de información permite crear una base de datos histórica, la cual puede ser valiosa para identificar patrones y tendencias o para informar decisiones basadas en el análisis de datos acumulados a lo largo del tiempo. Este prototipo, por lo tanto, no solo captura instantáneas del entorno sino que también proporciona una perspectiva de evolución en las condiciones monitoreadas.

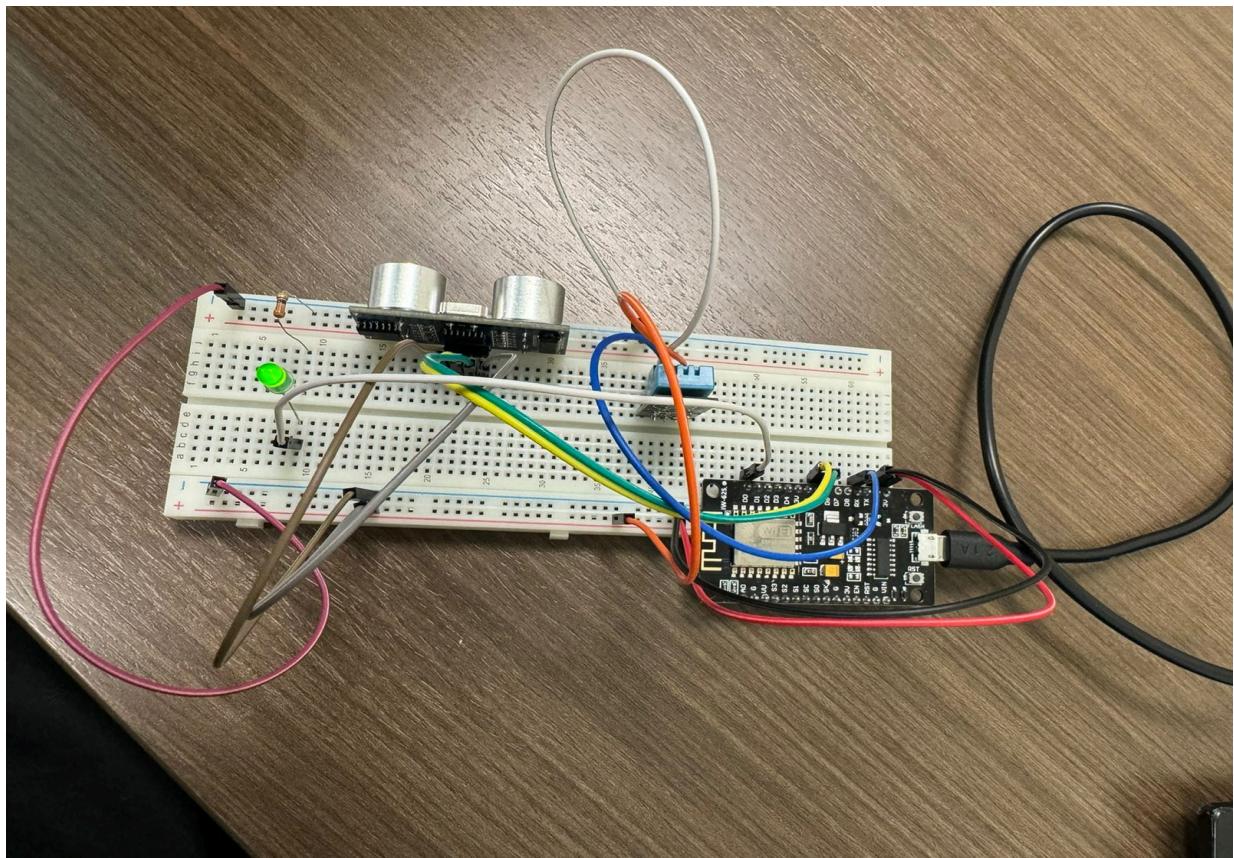
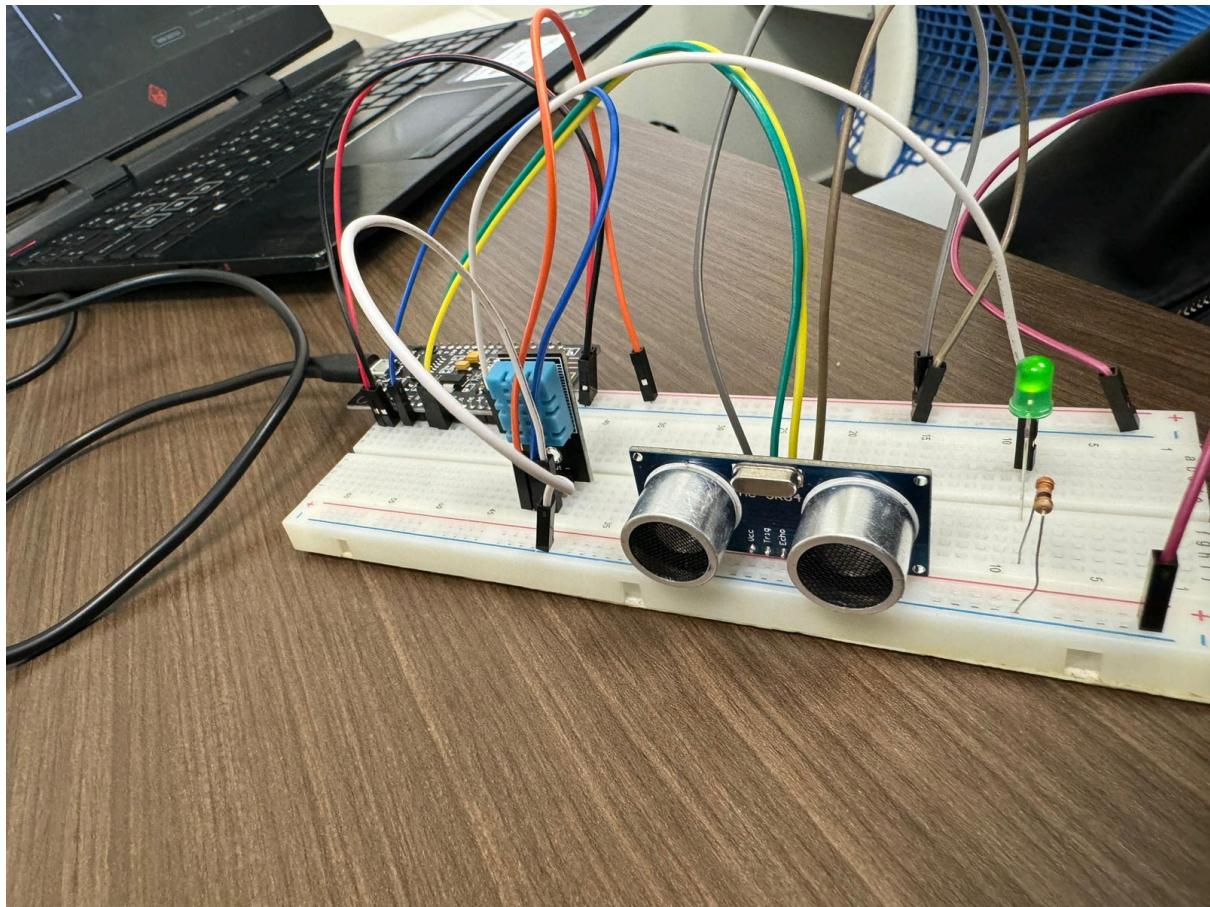
Lista de componentes utilizados

- NodeMCU v3 CH340:
- Sensor de Temperatura y Humedad (DHT)
- Sensor Ultrasónico
- LED
- Resistencias
- Protoboard
- Cables de Conexión

Video del Equipo:

<https://youtube.com/shorts/Pp9KpS5jroY?si=XGccdn4ZrCaIVFMM>
(Compañeros)

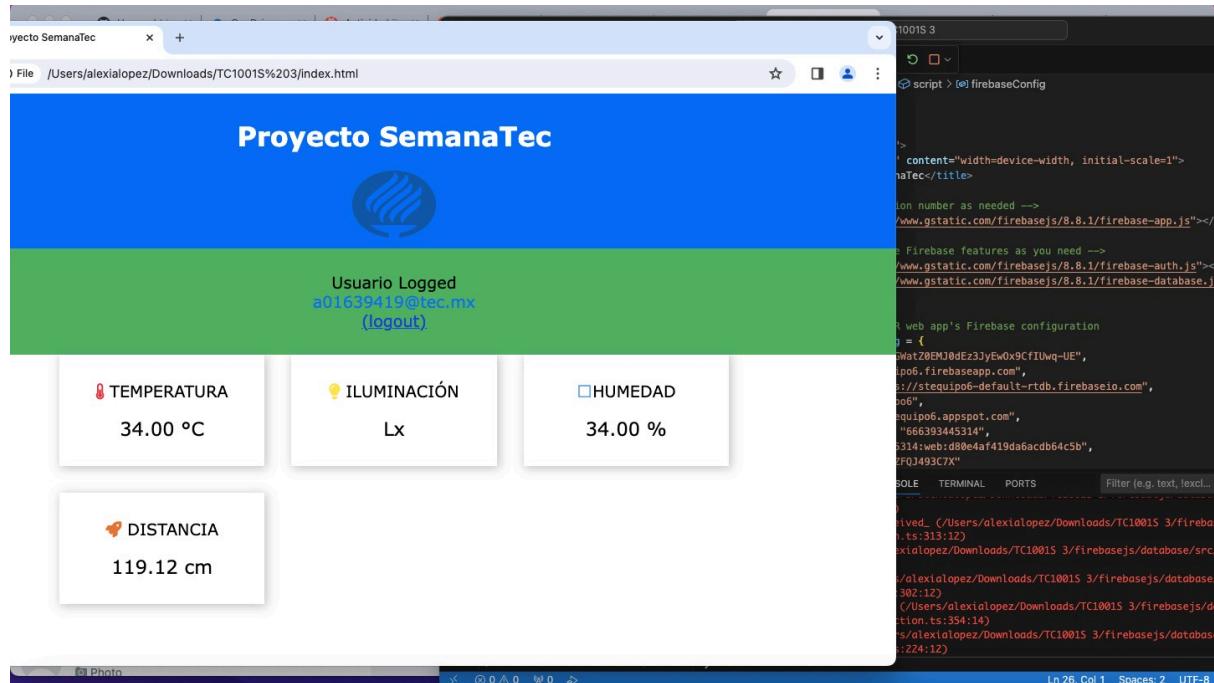
Círcuito en Físico



Diseño de la Aplicación.

Diseño Web.

(Conexión de el código)



Documentación:

- **Configuración de Firebase**

- firebaseConfig es un objeto que contiene la configuración necesaria para inicializar Firebase en la aplicación web. Incluye la apiKey, que es una clave de API pública utilizada para acceder a tu proyecto de Firebase.
- authDomain es el dominio de autenticación para tu proyecto Firebase.
- databaseURL es la URL de tu base de datos en tiempo real de Firebase, donde la aplicación puede almacenar y sincronizar datos.
- projectId es el ID de tu proyecto Firebase.
- storageBucket es la dirección del bucket de almacenamiento asociado con tu proyecto Firebase, utilizado para almacenar archivos como imágenes, videos, etc.
- messagingSenderId es el ID del remitente utilizado para enviar notificaciones push.
- appId es el ID de la aplicación, que es único para cada proyecto de Firebase.
- measurementId se relaciona con Google Analytics para Firebase, permitiendo recoger datos de uso de la aplicación.

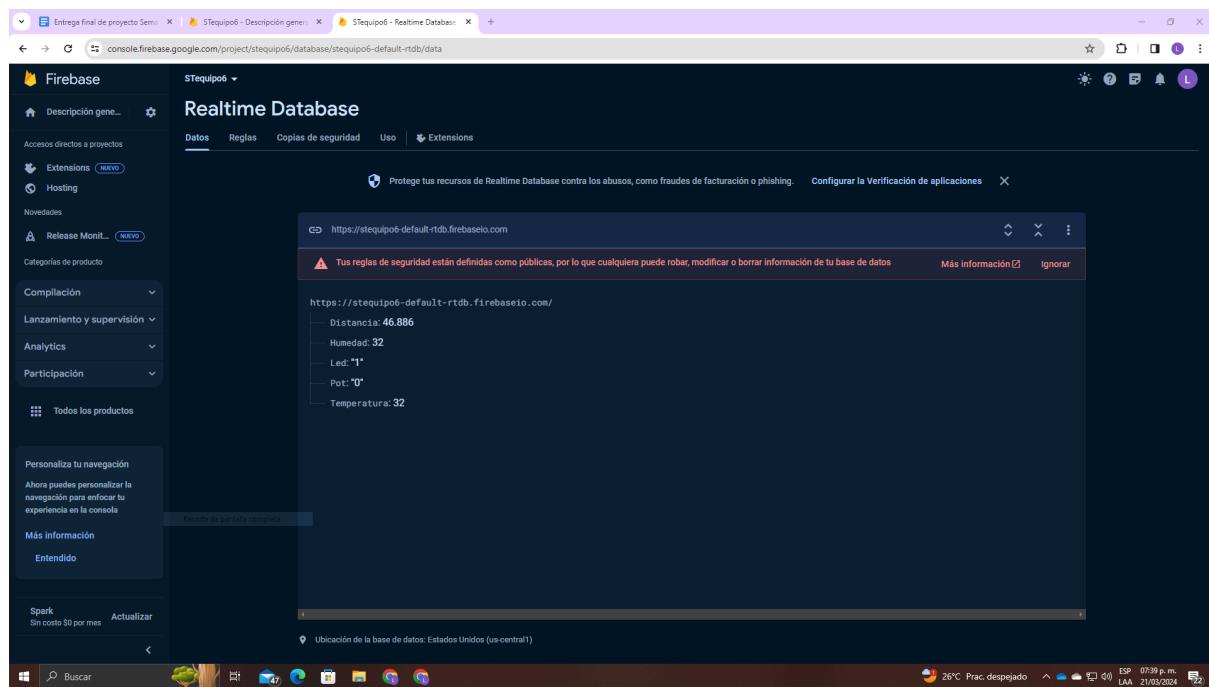
Scripts de Firebase:

- Se incluyen scripts externos que cargan la SDK de Firebase. Estos scripts son esenciales para que la aplicación web interactúe con los servicios de Firebase.

Debugging y Errores:

- En la consola de la herramienta de desarrollo (posiblemente Visual Studio Code), hay una indicación de que el usuario fue desconectado y luego reconectado, lo cual podría ser parte del sistema de autenticación de la aplicación.
- Además, hay un mensaje de error en la consola que indica un problema con la lectura de una propiedad. Este tipo de errores normalmente ocurre cuando se intenta acceder a una propiedad de un objeto que no existe o no está definido.

Screenshot de la base de datos.



Documentación:

- **Librerías:** Se incluyen dos bibliotecas esenciales para la operación del código:

- `ESP8266WiFi.h`: proporciona las funciones necesarias para conectar el ESP8266 a una red WiFi.
 - `FirebaseESP8266.h`: ofrece las funcionalidades para interactuar con la base de datos Firebase.
- **Credenciales WiFi y Firebase:** Se definen las constantes `ssid` y `password` con los datos de la red WiFi a la que se conectará el ESP8266. Se establece la `API_KEY` que autentica las peticiones a Firebase. Se asignan las cadenas `FIREBASE_HOST` y `FIREBASE_AUTH` que contienen la URL de la base de datos Firebase y el token de autenticación respectivamente.
 - **Variables y Configuración Inicial:**
 - Se crea un objeto `FirebaseData` llamado `firebaseData` que se usará para enviar y recibir datos de Firebase.
 - Se declara la variable `iterar` para controlar la ejecución del bucle `while` dentro de `loop()`.
 - Se configura un pin LED (pin 16) como salida, que se utilizará para mostrar el estado de un LED basado en los datos de Firebase.
 - **Función setup():** Se ejecuta una vez al iniciar el microcontrolador y se encarga de:
 - Iniciar la comunicación serial.
 - Configurar los pines del LED como salidas.
 - Conectar el ESP8266 a la red WiFi especificada y esperar hasta que la conexión sea exitosa.
 - Inicializar la conexión a Firebase y configurar la reconexión automática de WiFi.
 - **Función loop():** Se ejecuta repetidamente y contiene:
 - Una cadena `nodo` que define la ruta de la base de datos de Firebase donde se almacenarán o leerán los datos.
 - Un bucle `while` que, mientras `iterar` sea verdadero, repetidamente verifica el valor del nodo "Led" en Firebase.
 - Si el valor obtenido es "1", el pin LED se enciende; si es "0", se apaga.
 - El resto del código en `loop()` está comentado y muestra cómo escribir y leer varios tipos de datos en Firebase, así como enviar (push) datos a una lista.

Código

```
#include <DHT.h>
#include <DHT_U.h>
#include <ESP8266WiFi.h>
#include "FirebaseESP8266.h"

// Sensor DHT
#define DHTPIN 3
#define DHTTYPE DHT11
DHT_Unified dht(DHTPIN, DHTTYPE);

// WiFi y Firebase
#define ssid "viva"
#define password "3123133772"
#define API_KEY "AlzaSyDRJGWatZ0EMJ0dEz3JyEwOx9CfIUwq-UE"
const char *FIREBASE_HOST = "https://stequipo6-default-rtdb.firebaseio.com";
const char *FIREBASE_AUTH =
"jplcQC9Ubdh7DlIBmEQ5Bx6O0x4mZunA3IWDLCLs";
FirebaseData firebaseData;
int ledPin = 16; // Define the LED pin

void setup() {
    Serial.begin(115200);
    Serial.println();

    // Initialize the DHT sensor
    dht.begin();

    // Set the LED pin as an output
    pinMode(ledPin, OUTPUT);

    // Connect to WiFi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(250);
    }
    Serial.println("\nConnected to WiFi");

    // Initialize Firebase
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
}
```

```

void loop() {
    // Read temperature and humidity from the DHT sensor
    sensors_event_t event;
    dht.temperature().getEvent(&event);
    if (!isnan(event.temperature)) {
        Serial.print("Temperature: ");
        Serial.print(event.temperature);
        Serial.println("°C");
        // Send the temperature to Firebase under the "Temperature" node
        Firebase.setFloat(firebaseData, "/Environment/Temperature", event.temperature);
    }

    dht.humidity().getEvent(&event);
    if (!isnan(event.relative_humidity)) {
        Serial.print("Humidity: ");
        Serial.print(event.relative_humidity);
        Serial.println("%");
        // Send the humidity to Firebase under the "Humidity" node
        Firebase.setFloat(firebaseData, "/Environment/Humidity", event.relative_humidity);
    }

    // Check the LED state from Firebase and control the physical LED accordingly
    Firebase.getString(firebaseData, "/Controls/Led");
    if (firebaseData.stringValue() == "1") {
        digitalWrite(ledPin, HIGH);
    } else if (firebaseData.stringValue() == "0") {
        digitalWrite(ledPin, LOW);
    }

    delay(2000); // Wait for 2 seconds before the next reading and update
}m

```

Conclusiones

La integración del hardware con el NodeMCU y los sensores de temperatura, humedad y distancia, junto con la correcta programación en el IDE de Arduino, nos permitió recopilar datos del entorno de manera precisa. La habilidad de transmitir estos datos a una base de datos en tiempo real, utilizando Firebase, no solo incrementó nuestra comprensión de las bases de datos noSQL y el manejo de la nube, sino que también aseguró que nuestra aplicación fuera robusta y escalable.

El desarrollo de la interfaz de usuario con App Inventor, a pesar de ser una plataforma de nivel de entrada, demostró ser una excelente elección para un prototipado rápido y efectivo. Nos permitió visualizar los datos en tiempo real y gestionar las interacciones con el usuario de manera intuitiva.

En conjunto, este proyecto fue una clara muestra de cómo la intersección de diferentes tecnologías puede ser utilizada para crear soluciones innovadoras y prácticas. Los resultados han demostrado que con la combinación adecuada de componentes de hardware y software, y con una buena dosis de creatividad y colaboración, se pueden alcanzar metas que tienen un impacto tangible en el mundo real. Ha sido un camino lleno de aprendizaje continuo y una gran satisfacción al ver nuestra solución en acción. A lo largo del desarrollo, enfrentamos y superamos desafíos técnicos, lo cual afianzó nuestro conocimiento en áreas como programación de microcontroladores, sensores, comunicación inalámbrica y desarrollo de aplicaciones.