Termómetro Inteligente

Asignatura: **Sistemas Digitales**
Curso: **2025/2026**

Nombre: **Alejandro Escalante Clemente**
DNI: **77030156M**

Repositorio GitHub del proyecto

Motivación

Este proyecto, viene motivado por la idea de obtener conocimientos de arduino y sensores, y sobretodo de comprender la generación de un canal bluetooth (en este caso de Bajas Emisiones o Low Emissions) para comunicar un microcontrolador ESP 32 con un dispositivo móvil en este caso. Se decidió centrarse en un termómetro puesto que es algo sencillo que hemos podido llegar a usar o ver, pero que no sabemos cómo de complicado puede llegar a ser.

La idea de usar bluetooth en vez de una pantalla viene dada porque con este proyecto se intenta no solo generar un termómetro a distancia común sino añadirle la idea de hacerlo lo más compacto posible, y a su vez servir como prototipo para mejoras futuras que se desarrollaran en el apartado Mejoras.

Aunque no era su objetivo principal, también ha derivado en el aprendizaje de creación de aplicaciones móviles sencillas con el programa MIT App Inventor, que es un software gratuito e intuitivo de programación por bloques.

Descripción:

Este proyecto se centra en el diseño e implementación de un termómetro digital sin contacto, preciso y fácil de usar, destinado a la medición de la temperatura corporal.

El funcionamiento del dispositivo se centra en un microcontrolador ESP 32 que controla dos sensores principales:

1. Sensor de Distancia, con el que se va a buscar la distancia óptima de medición, de tal manera que la medida sea lo más precisa posible para el sensor de temperatura utilizado.
2. Sensor de Temperatura, con el que se extrae el valor de temperatura y se imprime como valor resultante.

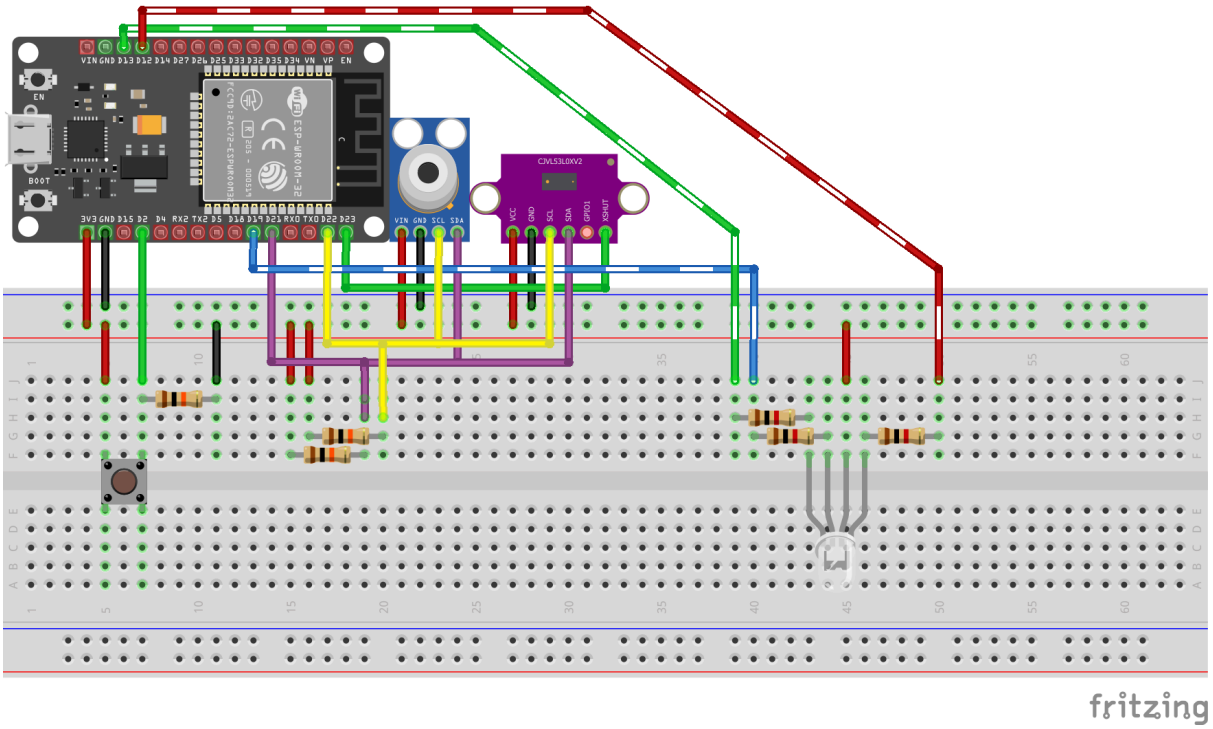
A partir de los datos obtenidos por los sensores descritos anteriormente, se mostrarán distintos colores a través de un led RGB para informar en todo momento al usuario que el dispositivo está en funcionamiento, un pulsador para que el usuario gestione el encendido o apagado del dispositivo, y una aplicación móvil que procesa los datos que posteriormente se comunican al usuario.

Respecto a la alimentación del dispositivo, hemos optado por la alimentación por cable al propio dispositivo móvil para evitar el uso de baterías externas, pero cómo se desarrollará en el apartado de posibles mejoras, se podría añadir un módulo de alimentación externo.

Técnicas y Materiales Utilizados

Materiales

- Placa: ESP 32 con conexión Tipo C
- Sensor Temperatura: MLX90614ESF-BAA-000-TU.
- Sensor IR: Pololu VL53L0X.
- Switch Pulsador
- Led RGB Ánodo Común.
- 3 Resistencias de 1kΩ.
- 3 Resistencias de 10kΩ.



Técnicas Usadas

Cálculo de distancia óptima

Para determinar la distancia de medición óptima, debemos de delimitar una superficie de medición menor que la superficie que deseamos medir, en este caso la frente de un individuo. Para ello, tenemos los siguientes datos:

- Distancia al objeto(D): La proporciona el sensor VL53L0X
- Ángulo del campo de visión (FOV), que en este caso es α = 90º

Con estos dos datos y sabiendo que la geometría de medición es un cono, con base la Superficie de medición (S) y vértice el sensor de temperatura, podemos realizar el cálculo.

$$S = 2 \cdot D \cdot \tan\left(\frac{\alpha}{2}\right)$$

Puesto que la tan(45º) = 1, la ecuación resultante es:

$$\mathbf{S = 2D}$$

Teniendo en cuenta una frente de altura mínima de 5 cm, determinamos una superficie circular de diámetro igual a la altura mínima.

$$S = \boldsymbol{\pi} \cdot (h/2)^2$$

En nuestro caso, da como resultante una superficie $S = 19.635\ cm$, y por lo tanto tenemos una distancia umbral $D = 9.818\ cm \cong 10\ cm$.

Calibración sensor de temperatura

En este proyecto se ha preguntado a una IA sobre como aumentar la precisión y nos a mostrado unos valores de offset y sensibilidad genéricos que vamos a utilizar como base; sin embargo, para una mayor precisión se requiere de un estudio que trata de las siguientes fases:

1. *Medición de temperatura corporal con temperatura ambiente uniforme y distintos sujetos:*

Este primer estudio trata de obtener la desviación de la medición objeto del prototipo cuando la medición ambiente se mantiene aproximadamente intacta, y compararla con la medición obtenida por un termómetro clínico.

Preferentemente se va a buscar una temperatura ambiente de 25ºC que será en nuestro caso la de referencia.

A raíz de los resultados obtenidos, se va a calcular el promedio sobre las distintas diferencias entre el prototipo y el termómetro clínico, y este será nuestro valor Θ = *offset*

2. *Medición de temperatura corporal a distintas temperaturas ambiente:*

En el siguiente estudio, obtendremos distintas medidas de distintos sujetos a distintas temperaturas ambiente para poder calcular la sensibilidad del prototipo. Para estas mediciones, aplicaremos el offset previamente calculado.

A través de los resultados obtenidos, calculamos la diferencia entre el prototipo y el termómetro clínico $(\Delta Terror)$,y la referenciamos a la diferencia de temperatura $(\Delta Tamb)$:

$$\Delta Terror/\Delta Tamb$$

Por último, promediamos el cálculo anterior al número de muestras tomadas y el resultado será la sensibilidad del prototipo, □ = *coeficiente de sensibilidad*.

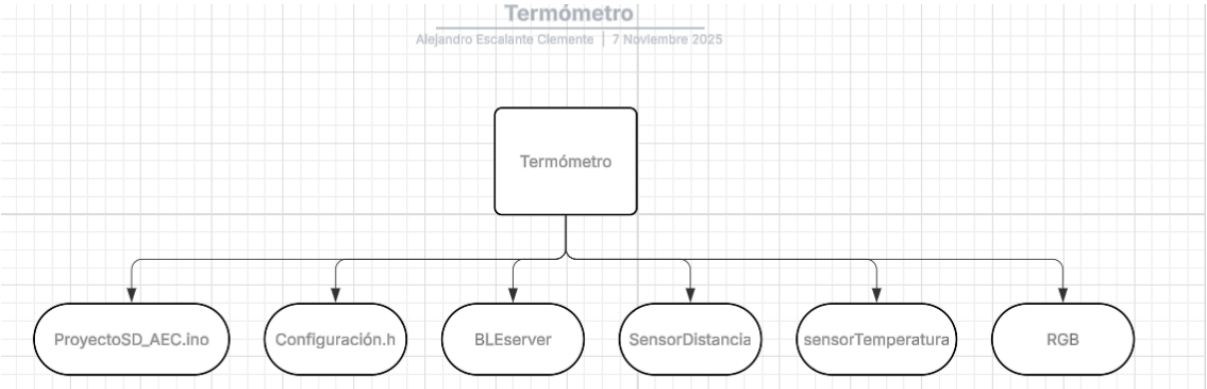
3. *Fórmula resultante:*

$$Medición = Tobj + (Tamb - 25ºC)\ x\ □ + \Theta$$

Se debe de avisar que esta calibración es lineal y aumenta la precisión del sistema, pero está limitado a un rango de temperatura ambiente de 15 a 35ºC y va perdiendo precisión conforme te alejas de este rango, además el modelo aplicado no es válido para temperaturas ambiente inferiores a 0ºC, por lo que se recomienda usar el prototipo en interiores.

Estructura de Archivos y Análisis del Código Fuente

Estructura de Archivos



Análisis del Código Fuente - Arduino

Configuración

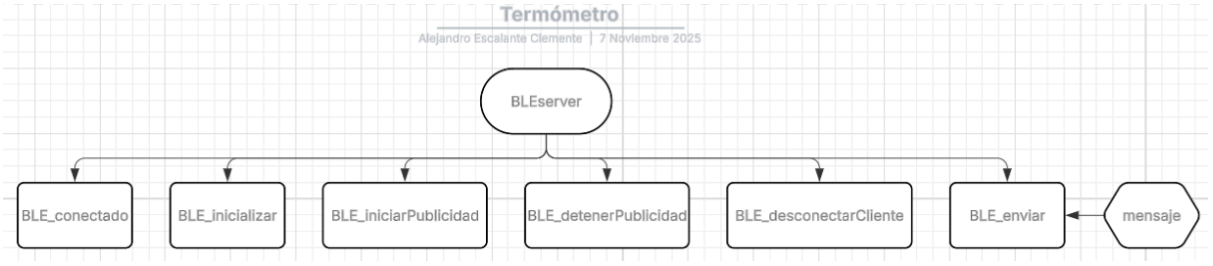
En este archivo se define la configuración del sistema, por lo tanto, se definen las siguientes variables:

1. Pines de los componentes
2. Constantes (PE. Umbrales, tiempos de control…)
3. UUIDs y nombre del servidor Bluetooth LE

Además se define el sistema de corrección de errores (debug).

Servidor Bluetooth LE

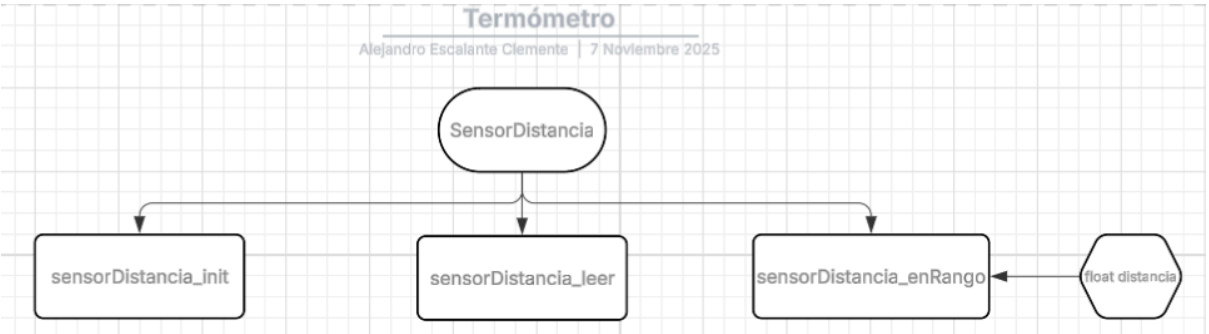
En este archivo buscamos crear el servidor BLE, iniciar/detener la publicidad, desconectar al cliente, informar cuando se conecte un cliente y enviar mensajes.



Previamente a la definición y desarrollo de las funciones en el archivo .cpp, definimos las librerías usadas, objetos globales (punteros) y una clase “MyServerCallbacks” para detectar los cambios de estado en el cliente y poder determinar si está o no conectado.

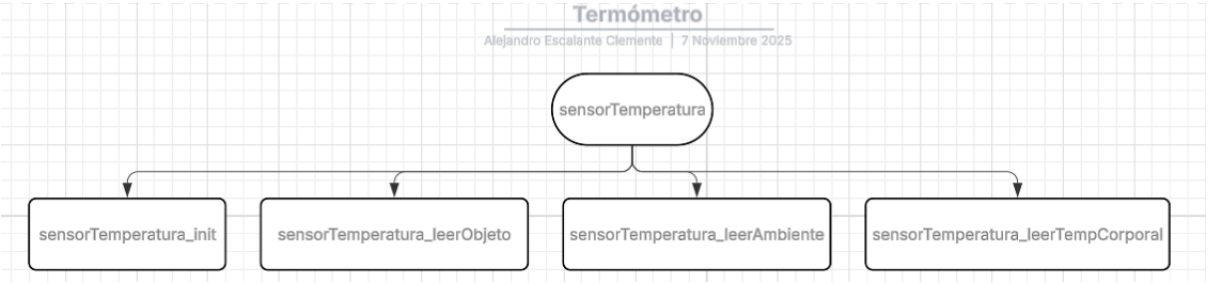
```
// ------------------------------
// Callbacks personalizados
// ------------------------------
//Definimos el siguiente callback para saber si hay un cliente conectado o no, para ello nos aprovechamos cuando se
//sobreescribe (override) el estado de conexión con las funciones onConnect y onDisconnect
class MyServerCallbacks : public NimBLEServerCallbacks {
  void onConnect(NimBLEServer* pServer) override {
    dispositivoConectado = true;
    debugln("📶 Cliente BLE conectado");
  }

  void onDisconnect(NimBLEServer* pServer) override {
    dispositivoConectado = false;
    debugln("❌ Cliente BLE desconectado");
    NimBLEDevice::startAdvertising();
  }
};
```

Sensor de Distancia



En este bloque se inicializa el sensor distancia VL53L0X con tecnología I2C y se definen sus funciones respectivas para leer muestras de distancia con la mejor relación precisión/tiempo posible, y para comprobar si estas mediciones están dentro del rango de medición de temperatura.

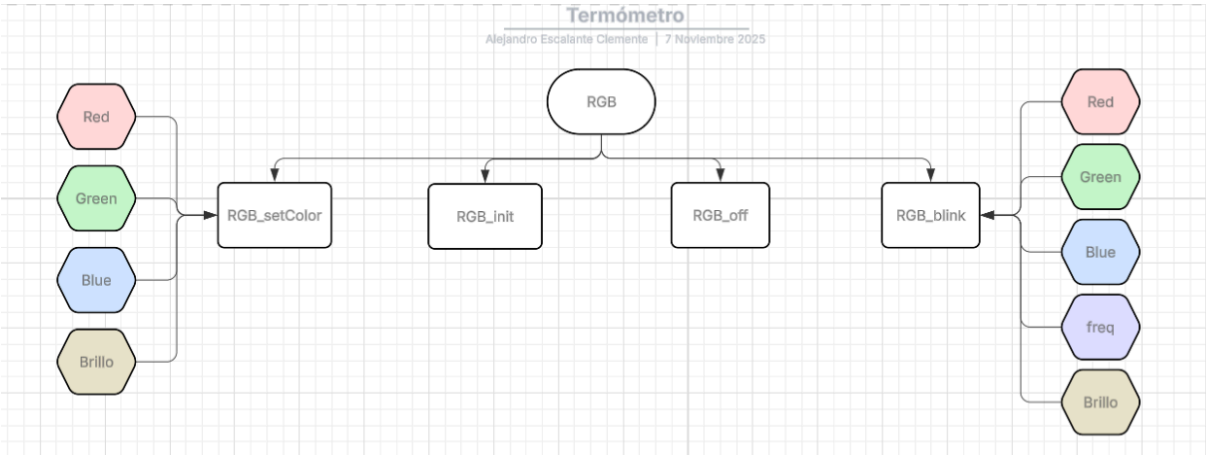
Sensor de Temperatura



Una vez conseguida la distancia óptima de medición se pasará a medir la temperatura, para ello hemos definido este archivo en el que se inicializa el sensor y se define el parámetro de emisividad.

Para la medición de temperatura, usamos la función de leer temperatura corporal que a su vez necesita de una medición IR de la radiación térmica del objeto, y una lectura de la temperatura ambiente.

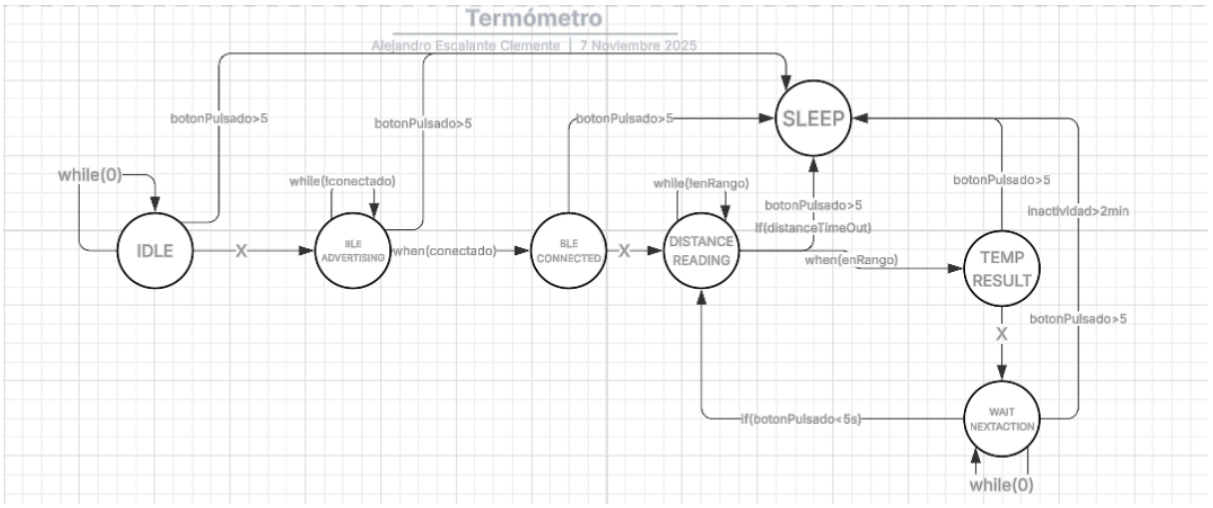
RGB



En este archivo se inicializa el led RGB junto a sus canales, frecuencia de PWM y resolución.

Además se definen funciones para establecer colores con formato RGB (RGB_setColor), para establecer un color y parpadeo (RGB_blink) o para apagar el led.

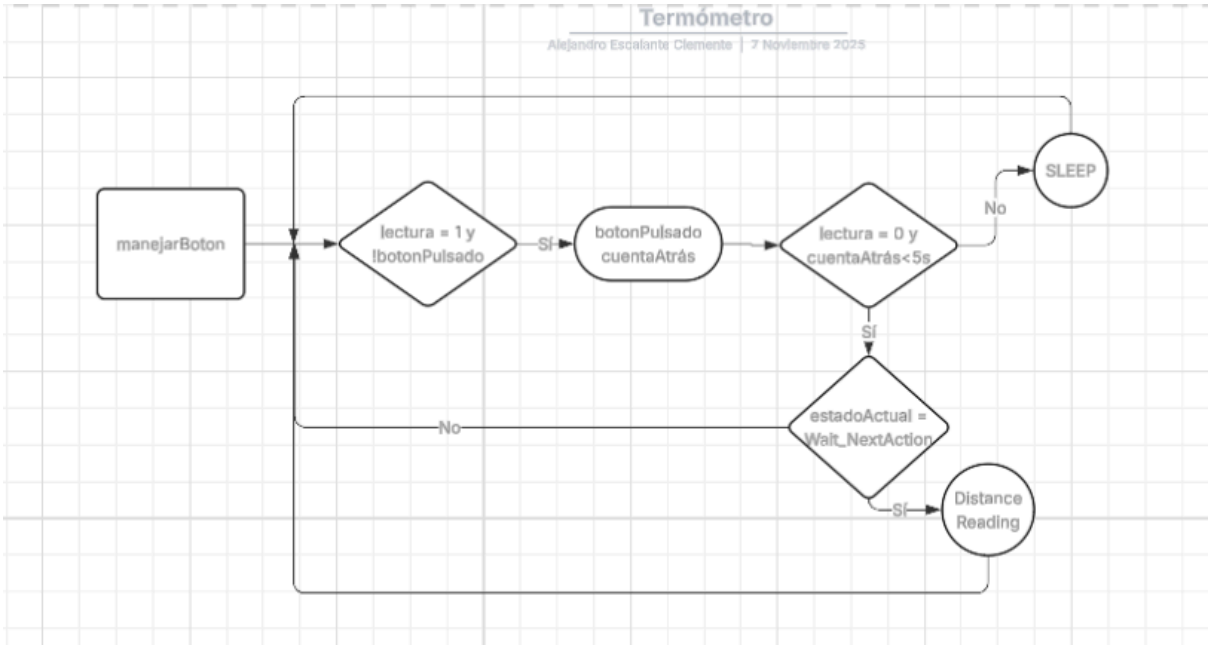
ProyectoSD_AEC.ino



Este es el archivo principal que define el comportamiento del termómetro digital a través de los archivos y funciones mencionadas anteriormente.

Está estructurado con una máquina de estados a la que se conecta el funcionamiento de un botón (función externa) y una gestión de tiempos con la función millis() para gestionar inactividad y la funcionalidad de apagado del sistema a través del botón.

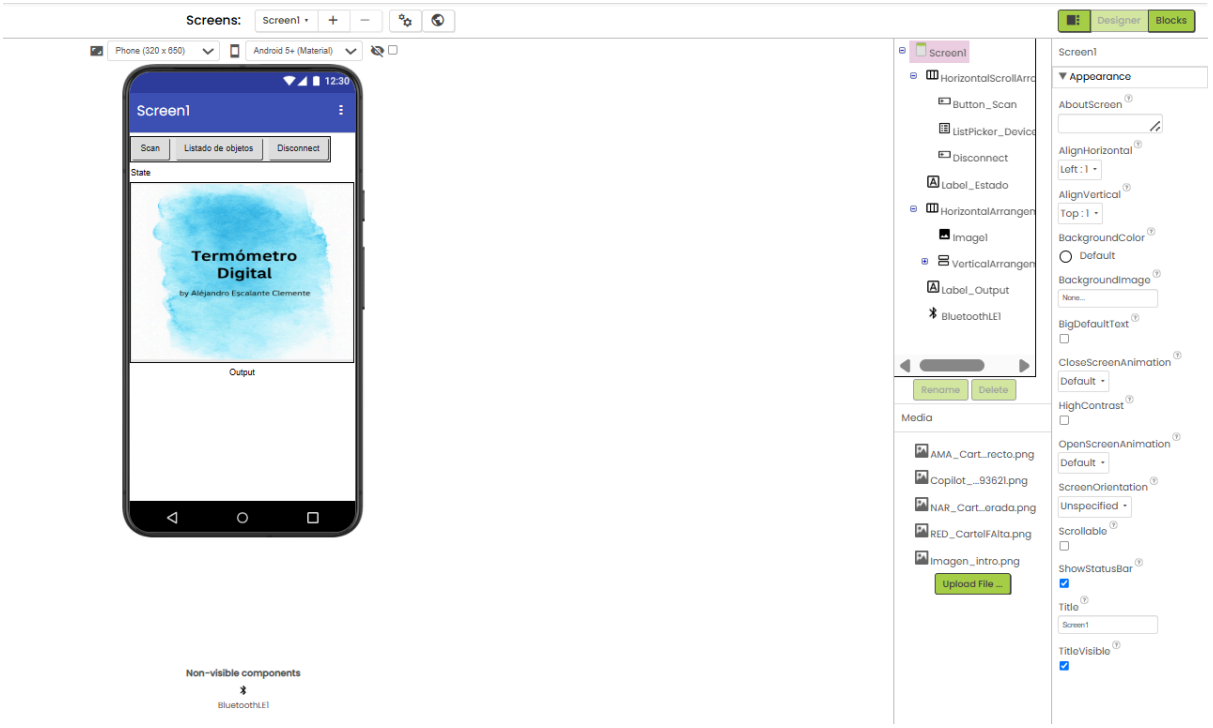
Función manejarBoton():



Análisis del Código Fuente - MIT App Inventor

En este apartado vamos a explicar como se ha creado la aplicación de móvil a través de MIT App Inventor, una aplicación que permite crear aplicaciones móvil con programación de bloques de una manera sencilla.

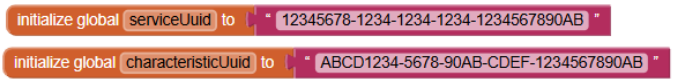
Cuenta con una ventana de diseño en la que se añaden botones, etiquetas, zonas de imágen y otros objetos que aparecerán posterior en la pantalla de los dispositivos, y una ventana de bloques en la que se programan los botones y otros objetos para conectarlos y añadir funcionalidades.



Descripción del código

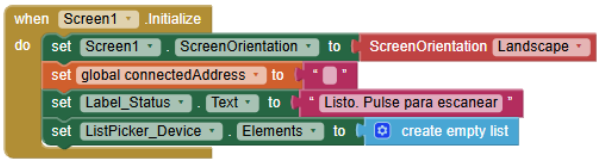
Variables globales

Definimos como variables globales, aquellas que vamos a usar en numerosas ocasiones en el código:



Inicialización de la pantalla

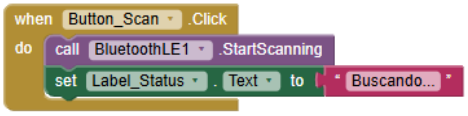
Se define la pantalla junto a su orientación (horizontal o landscape) y aquellos objetos que deban definirse inicialmente.



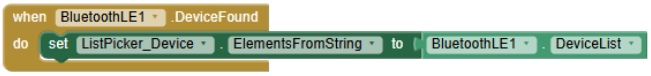
Gestión Bluetooth LE

El sistema de conexión bluetooth sigue los siguientes pasos:

1. Botón de Escanear: Una vez pulsado este botón, el dispositivo empieza a buscar dispositivos cercanos que tengan conexión bluetooth.

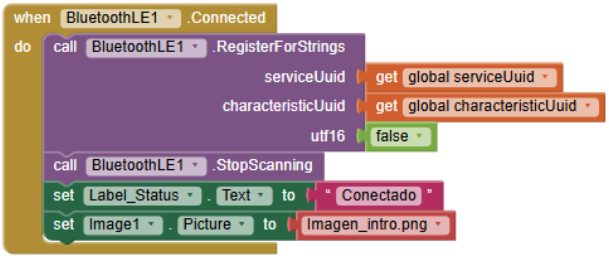


2. DeviceFound: Este bloque añade los dispositivos encontrados a una lista en común para la posterior selección del dispositivo deseado.

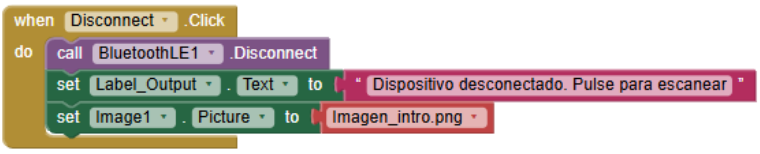


3. AfterPicking: Una vez hemos terminado de escanear, entramos en la lista “Listado de Objetos” y seleccionamos el dispositivo deseado para conectarnos

4. Connected: Si el dispositivo es el deseado, en este caso el termómetro, se conecta y establece en el registro las uuids de los punteros pServer y pCharacteristic para poder recibir información.



En caso de querer desconectarse del dispositivo, debemos pulsar el botón “Disconnect” y podremos buscar otro dispositivo.



Gestión de Información Recibida

Para la gestión de la información recibida tendremos en cuenta los códigos introducidos en el código de arduino y que los mensajes que se envían tienen el siguiente formato: [“mensaje”].

En cuanto a la recepción del mensaje, usamos el bloque principal de stringRecieved para detectar un nuevo dato, y posteriormente insertamos la uuid desde la que proviene la información con el bloque ReadString.

Una vez obtenido el mensaje, filtramos a través de los código DIST, RES, MSG, STATE o NEW, e imprimimos el contenido en el lugar correspondiente o realizamos los cambios que se hayan definido.

En la figura 1, se observan los bloques.

Resultado

Se ha obtenido un dispositivo de medición de temperatura con la búsqueda de la mayor precisión posible para un sensor térmico específico a través del cómputo de la distancia óptima de medición y la calibración lineal aproximada.

Se ha probado en protoboard y posteriormente se ha llevado a cabo la creación de un prototipo soldado en una placa y cableado por la parte inferior. Los sensores quedan sin soldar a espera de la creación de una carcasa para el prototipo con la intención de no usar cableado extra que luego pueda ser molesto en el montaje de la carcasa.

Mejoras

Se ha diseñado un prototipo funcional de termómetro digital a distancia; sin embargo, está abierto a mejoras futuras así como a otras modificaciones dependiendo del objetivo que se quiera obtener:

1. Modificación en los componentes: Este proyecto se ha llevado a cabo con un ESP32, aunque si queremos llevar más allá la idea de compacto, se recomendaría el uso de un ESP32 c3 supermini que cuenta con suficientes pines. También queda abierta la posibilidad de añadir un módulo de alimentación externo en vez de usar la propia batería del celular a través de un cable USB tipo C.

   También se recomienda variar la apertura focal del sensor de temperatura MLX90614 con la idea de aumentar la precisión e incluso la distancia de medición (FOV90º → FOV5º), al hacer estos cambios se debe recalcular la distancia óptima de medición o incluso se pueden definir varios rangos.
2. Diseño y construcción de una carcasa: Dependiendo del apartado número 1, se pueden llegar a diversas ideas de carcasas, entre las que yo recomendaría se encuentra tratar de hacer una carcasa imantada que se pueda adherir a la carcasa del dispositivo móvil para añadir fijación. Si se decanta por añadir alimentación externa por pilas por ejemplo, me decantaría por un diseño algo más robusto e incluso en forma de pistola como los que se encuentran en mercado.
3. Mejoras en el código: entre mis ideas para mejorar el código o añadir más funcionalidades se encuentran las siguientes:
   - Creación de bibliotecas o funciones con las que modificar todos los colores de los leds con la intención de generar modos daltónicos que se puedan adaptar a todos los públicos.
   - Siguiendo la línea anterior, añadir un buzzer en la aplicación móvil para transmitir por vía oral los resultados de las mediciones y análisis.
4. Variar el objetivo a medir: Este prototipo mide distancia y temperatura, su cálculo está definido para medir temperatura corporal con la mayor precisión posible, sin embargo se le puede buscar otras aplicaciones con ligeras modificaciones, como sería el caso de un módulo para coches para detectar goteras puesto que se ha buscado un diseño compacto lo cual es ideal para esta aplicación en concreto.

A este apartado se le añade la mejora en la calibración descrita con un modelo de estudio y comparación con un dispositivo clínico ya calibrado. Además de la investigación de una metodología no lineal para poder abarcar un rango de temperaturas más amplio.

Conclusión

Se han cumplido con creces los objetivos principalmente planteados e incluso se han ido añadiendo distintas funcionalidades y modificaciones debido a la investigación de métodos óptimos de medición y de intentos de mejorar la precisión.

Además se han comprendido diversos conceptos y metodologías, se ha logrado el objetivo de establecer una conexión Bluetooth y se ha llegado a un prototipo funcional.

Bibliografía:

- Github ESP32 board manager by espressif
- MIT App Inventor Ejemplos

Herramientas de IA

1. Generación de imágenes: Microsoft Copilot
2. IA - Chatgpt

Librerías - Github:

1. MLX90614 by Adafruit
2. VL53L0X by Pololu
3. Nimble-Arduino by h2zero (version 1.4.2)

Boards Manager - Github:

1. Arduino AVR Boards by Arduino (version 1.8.6)

Referencias - Figuras

Se añade la figura que debido a su tamaño, no se pudo añadir anteriormente:

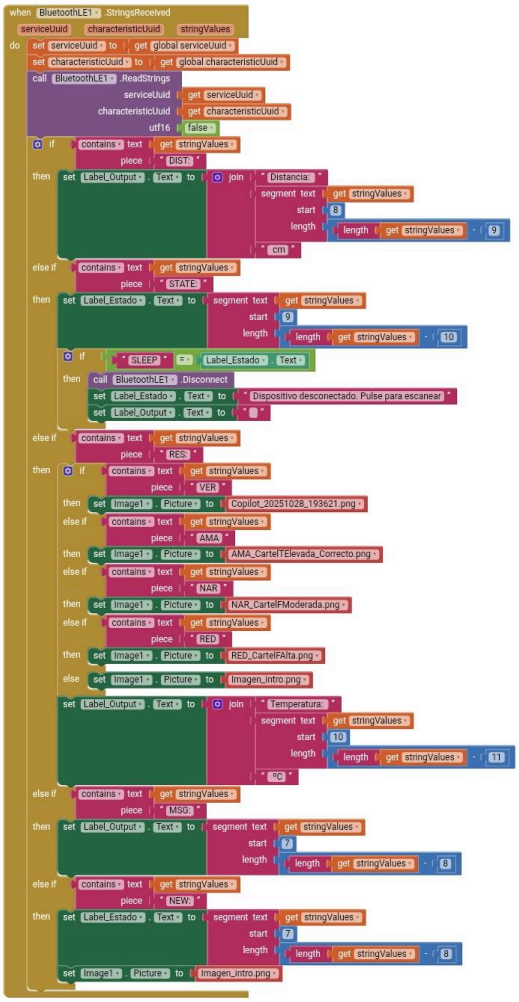


Figura 1. *Gestión de datos recibidos*