S.E.P. S.E.S. TecNM

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

Instituto Tecnológico de Aguascalientes

ACTIVIDAD

Practica Bluetooth (BLE)

Materia

Tecnologías Inalámbricas

Maestro: Ricardo Alejandro Rodriguez Jimenez

Alumno:

Jorge Alberto Casas Demetrio

Jason Alexander Zacarias Garcia

Ingeniería en Tecnologías de la Información y la Comunicación

Contenido

Introducción

La comunicación inalámbrica eficiente es fundamental para la interacción entre dispositivos móviles y sistemas embebidos. En este documento se realiza una práctica donde dejamos de lado el bluetooth clásico y nos vamos por el lado de la tecnología Bluetooth Low Energy (BLE) que se ha consolidado como un estándar de la industria debido a su bajo consumo de energía y su arquitectura basada en servicios y características, superando al Bluetooth clásico en aplicaciones que requieren intercambios de datos puntuales y optimización de recursos.

El presente documento detalla el desarrollo e implementación de un sistema de control inalámbrico utilizando un microcontrolador ESP32 y una aplicación móvil personalizada, desarrollada en la plataforma MIT App Inventor. El objetivo principal de esta práctica es demostrar la capacidad del ESP32 para actuar como un servidor BLE, permitiendo el escaneo y la vinculación desde un dispositivo Android para el control remoto de periféricos.

Contenido

Link Video

https://youtu.be/lEh389Vwztw

Link GitHub

https://github.com/JasonMakana/Practica-Bluetooth-BLE-.git

Diferencia entre BLE y Bluetooth

Bluetooth clasico (BR/EDR): Es el estándar original diseñado para el intercambio continuo de datos a corto alcance. Está pensado para conexiones que requieren un flujo de datos constante y "pesado". Funciona mediante una conexión "siempre activa". Una vez que dos dispositivos se emparejan, mantienen un canal abierto para transmitir datos. Es excelente para aplicaciones que no pueden permitirse cortes, pero drena la batería rápidamente.

Bluetooth Low Energy (BLE): Introducido con la versión 4.0, no es una actualización del anterior, sino un diseño nuevo. Su objetivo principal es el **bajo consumo de energía**, permitiendo que dispositivos funcionen durante meses o años con una sola batería de botón. BLE funciona mediante un sistema de **"dormir y despertar"**. El dispositivo permanece en modo de reposo la mayor parte del tiempo y solo se "despierta" cuando necesita enviar un pequeño paquete de datos (por ejemplo, una lectura de temperatura o una notificación).

La principal diferencia radica en el **ciclo de trabajo** (Duty Cycle). Pero también tienen diferencias clave como lo es:

- **Bandas de frecuencia:** Ambos operan en la banda de **2.4 GHz ISM**, pero BLE utiliza 40 canales (3 de ellos para "anunciarse") mientras que el clásico utiliza 79 canales.
- **Latencia:** BLE tiene una latencia mucho menor (puede conectarse y enviar datos en milisegundos), mientras que el clásico tarda más en establecer la conexión tras un modo de ahorro.

Descripción del código – Cosas fundamentales

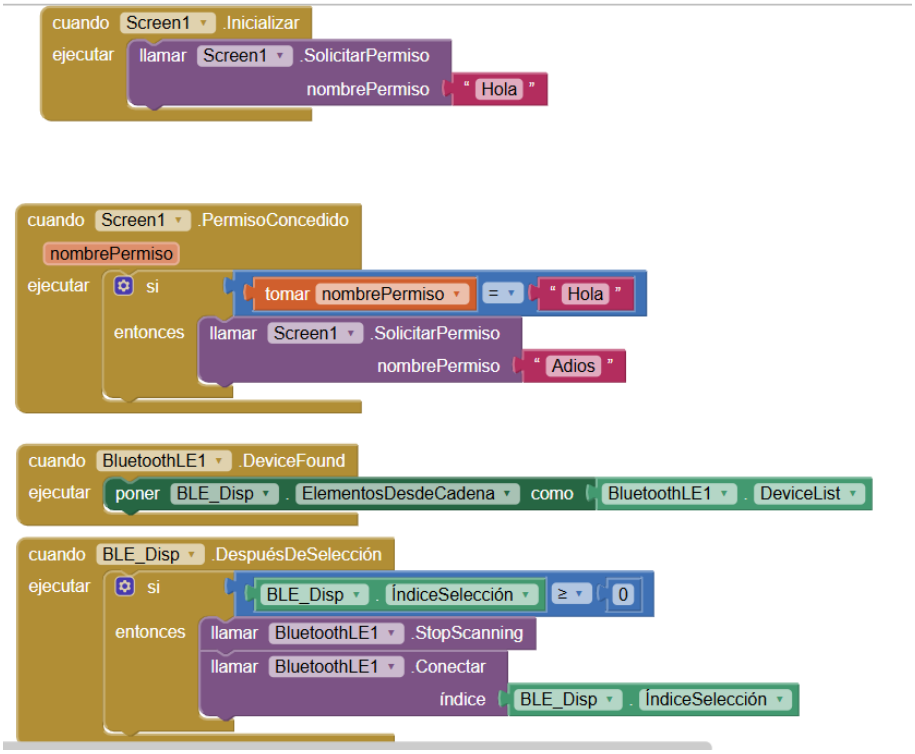
```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <Adafruit_NeoPixel.h>
//Se utilizan tres librerías principales:
  //BLEDevice/Server/Utils: Para que el ESP32 pueda hablar Bluetooth.
  //Adafruit_NeoPixel: Para controlar el LED de colores que trae el ESP32-S3 (usualmente en el pin 48).
 //PIN_FOCO (4): Un pin digital para encender o apagar algo externo.
#define PIN_RGB 48
#define PIN_FOCO 4
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN_RGB, NEO_GRB + NEO_KHZ800);
#define SERVICE_UUID "12345678-1234-1234-1234-1234567890ab"
#define CHARACTERISTIC_UUID "abcd1234-5678-1234-5678-abcdef123456"

bool dispositivoConectado = false;
bool conectado = false;
bool DatoRecibido = false;  // FLAG CRUCIAL
String palabra = "";

void setColor(uint8_t r, uint8_t g, uint8_t b) {
  pixels.setPixelColor(0, pixels.Color(r, g, b));
  pixels.show();
}
```

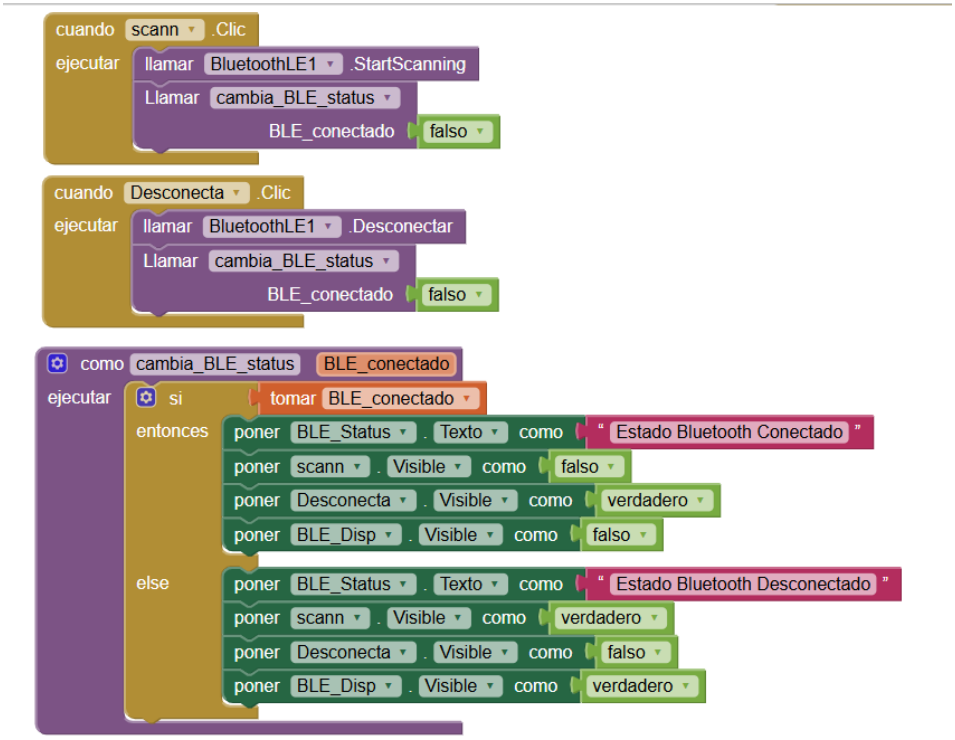
```
//Servidor que controla la conexión.
class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    dispositivoConectado = true;
    palabra = "";
    Serial.println("Conectado");
  }
  void onDisconnect(BLEServer* pServer) {
    dispositivoConectado = false;
    palabra = "";
    Serial.println("Desconectado");
    BLEDevice::startAdvertising();
  }
};

// Recepcion de datos que se activa cuando tú envías un texto desde el celular.
class MyCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic* pCharacteristic) {
    palabra = String(pCharacteristic->getValue().c_str());
    palabra.trim();
    palabra.toUpperCase();
    DatoRecibido = true;  // Avisamos al loop que hay algo nuevo
    Serial.println("Recibido: " + palabra);
  }
};
```

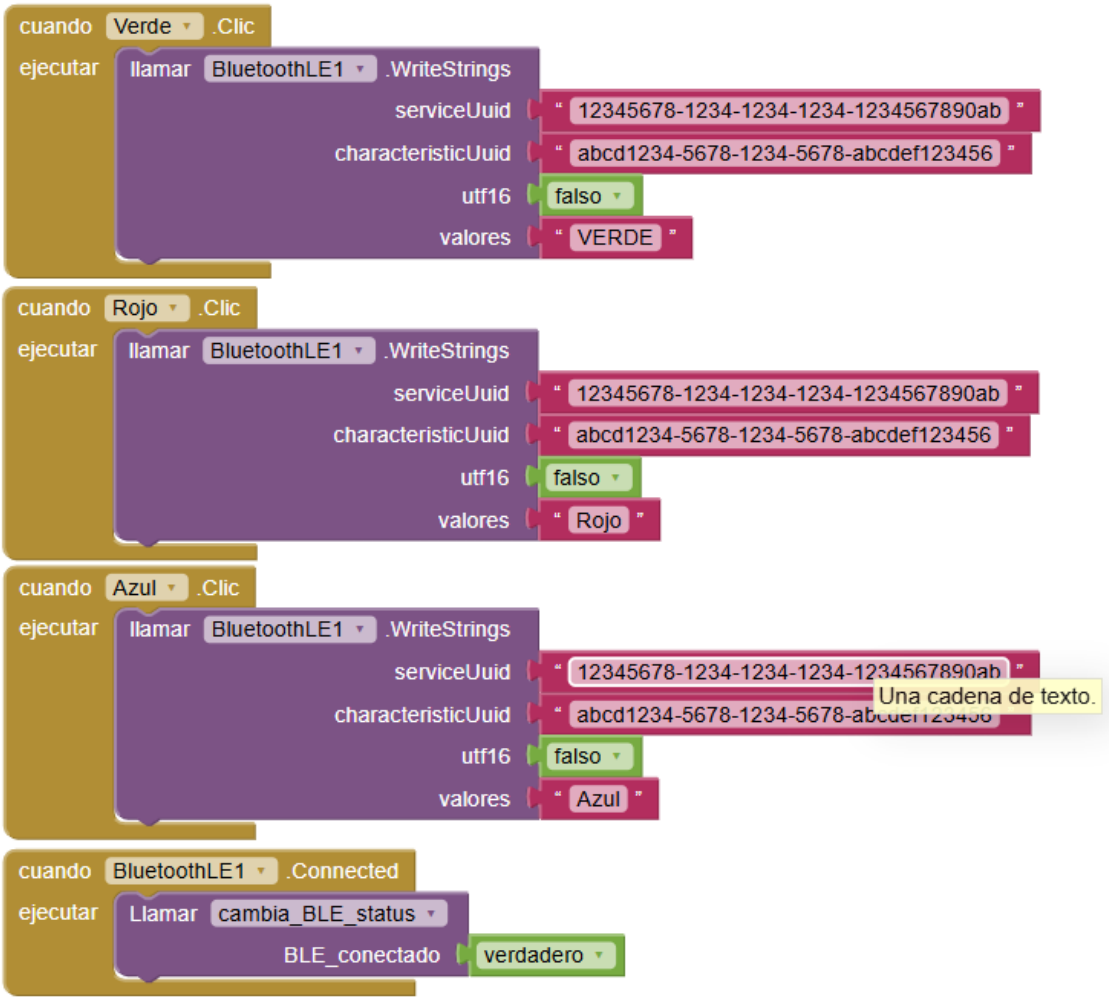
```
void setup() {
  Serial.begin(115200);
  pixels.begin();
  pixels.setBrightness(30);

  pinMode(PIN_FOCO, OUTPUT);
  digitalWrite(PIN_FOCO, LOW);

  BLEDevice::init("ESP32_S3_Team_mamalon");
  BLEServer* pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService* pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic* pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_WRITE |
BLECharacteristic::PROPERTY_WRITE_NR);

  pCharacteristic->setCallbacks(new MyCallbacks());
  pService->start();

  BLEDevice::startAdvertising();

  setColor(35, 132, 254);  // Azul cielo (Buscando)
}

void loop() {
```

```
  // 1. CAMBIO DE ESTADO: CONEXIÓN
  if (dispositivoConectado && !conectado) {
    setColor(160, 32, 240);  // Violeta (Conectado).
    conectado = true;
  }

  // 2. CAMBIO DE ESTADO: DESCONEXIÓN
  if (!dispositivoConectado && conectado) {
    setColor(255, 255, 0);
    conectado = false;       // Amarillo (Desconectado)
    delay(2000);             // Para que alcances a ver el amarillo antes de volver al azul
    setColor(35, 132, 254);  // Volver a Azul cielo
  }

  // 3. LÓGICA DE COMANDOS (Solo si está conectado y hay dato nuevo)
  if (dispositivoConectado && DatoRecibido) {
    if (palabra == "AZUL") setColor(0, 0, 255);
    else if (palabra == "ROJO") setColor(255, 0, 0);
    else if (palabra == "VERDE") setColor(0, 255, 0);
    DatoRecibido = false;          // IMPORTANTE: Bajamos la bandera
  }
}
```

Código App Inventor






cuando Verde .Clic
ejecutar llamar BluetoothLE1 .WriteStrings
serviceUuid " 12345678-1234-1234-1234-1234567890ab "
characteristicUuid " abcd1234-5678-1234-5678-abcdef123456 "
utf16 falso
valores " VERDE "
cuando Rojo .Clic
ejecutar llamar BluetoothLE1 .WriteStrings
serviceUuid " 12345678-1234-1234-1234-1234567890ab "
characteristicUuid " abcd1234-5678-1234-5678-abcdef123456 "
utf16 falso
valores " Rojo "
cuando Azul .Clic
ejecutar llamar BluetoothLE1 .WriteStrings
serviceUuid " 12345678-1234-1234-1234-1234567890ab "
Una cadena de texto.
characteristicUuid " abcd1234-5678-1234-5678-abcdef123456
utf16 falso
valores " Azul "
cuando BluetoothLE1 .Connected
ejecutar Llamar cambia_BLE_status
BLE_conectado verdadero

Materiales y Herramientas

Hardware (Componentes Físicos)

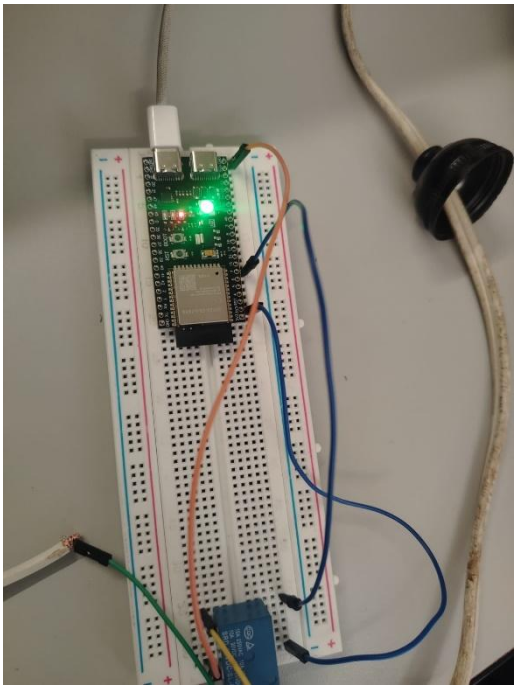
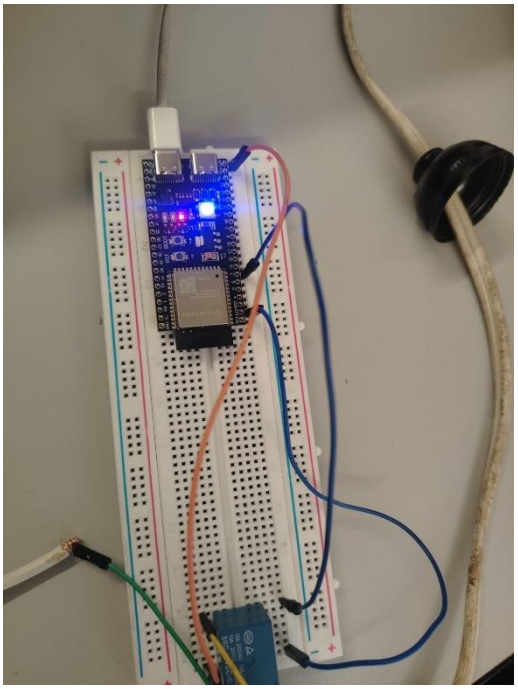
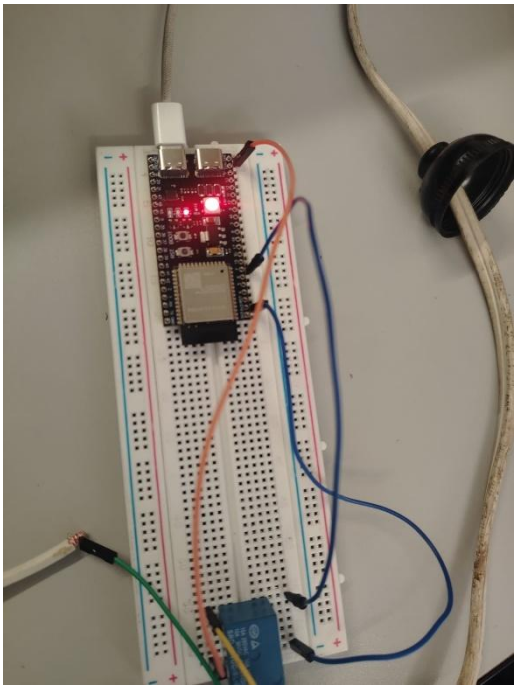
- **Microcontrolador ESP32:** ESP32-S3-WROOM-1-N16R8). Es el cerebro del proyecto, encargado de gestionar la pila de protocolos BLE y controlar las salidas PWM.
- **LED RGB:** (Integrado en el microcontrolador). El actuador que visualiza los estados de conexión y los colores seleccionados.
- **Cable USB a Micro-USB:** Para la alimentación del ESP32 y la carga del código desde la computadora.
- **Dispositivo Móvil (Android):** Smartphone o tablet para ejecutar la aplicación creada.

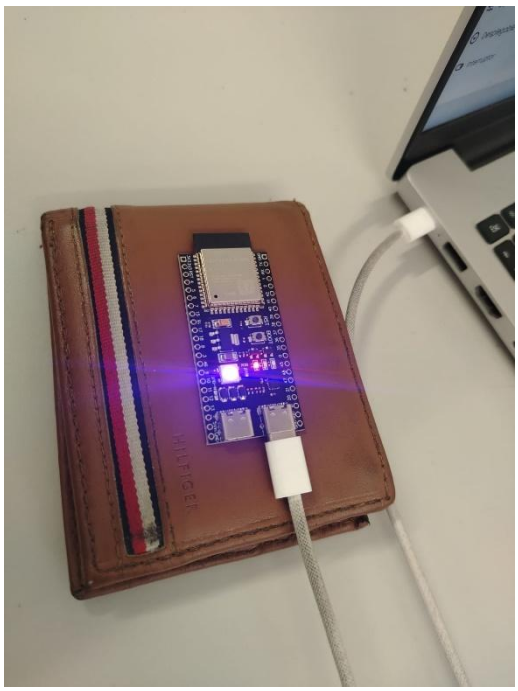
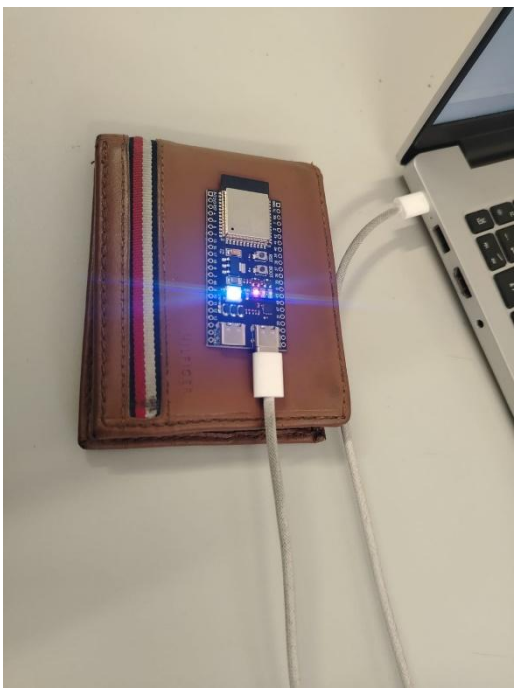
Software (Entornos de Desarrollo)

Las plataformas digitales que permitieron la programación y el diseño:

- **Arduino IDE:** Entorno utilizado para escribir, compilar y subir el código en C++ al ESP32.
- **MIT App Inventor:** Plataforma web basada en bloques para el desarrollo de la aplicación móvil.
- **Extensión BluetoothLE para App Inventor:** Complemento esencial (.aix) para habilitar las funciones de Bluetooth de Bajo Consumo, ya que no vienen por defecto.

Evidencias

Conclusión

Jason Alexander Zacarias Garcia

Con esta actividad pudimos aprender la forma en que funciona el ble, como es que el ESP32-s3 hace la comunicación y el cómo funciona app inventor.

Fue interesante ver el cómo es que esta tecnología bluetooth es tan fácil de usar y tan útil en tema de transmisión de datos de manera rápida y sin la necesidad de conexión por cable.

Otro punto importante para destacar es la forma de programarlo, ya que es un lenguaje más "antiguo" el cuales fue un poco más difícil de aplicar, ya que tenía menor noción de la sintaxis, y me fue muy interesante el investigar y ver la forma de programación, al igual, la forma en la que trabaja.

Jorge Alberto Casas Demetrio

Esta práctica me permitió entender cómo conectar un celular con un microcontrolador para manejarlo a distancia mediante el uso del bluetooth ble. Logramos que nuestra propia aplicación se comunicara con el ESP32 de forma inalámbrica, aprendiendo que para que esto funcione, ambos dispositivos deben hablar el mismo "idioma" y seguir las mismas reglas de conexión.

Lo más importante de este proyecto fue aprender a resolver problemas reales: desde configurar los permisos en el celular hasta programar el microcontrolador para que reaccione según su estado. Al final, logramos un sistema funcional donde, con solo presionar un botón en la pantalla, el hardware responde al instante, lo que nos da una base sólida para crear proyectos de tecnología más avanzados en el futuro.

Bibliografía

Giacinti, O., & Giacinti, O. (2025, 4 abril). *Bluetooth Low Energy vs. Bluetooth: Comprender las principales diferencias y aplicaciones*. Surgere. https://surgere.com/es/sin-categorizar/bluetooth-low-energy-vs-bluetooth-comprender-las-principales-diferencias-y-aplicaciones/