

S.E.P.



S.E.S.



TecNM



Instituto Tecnológico de Aguascalientes

## ACTIVIDAD

### Evaluativo 1: Control de potencia Bluetooth

#### 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.....	3
Contenido .....	4
Código Arduino IDE .....	4
Código App Inventor .....	8
Materiales .....	10
Evidencias.....	11
Conclusión.....	11

## Introducción

Hoy en día, controlar cosas desde el celular ya es algo común y súper práctico, que gracias al IoT puede ser algo posible y fácil de implementar. En esta práctica, nos damos a la tarea de poder armar un sistema de control remoto usando Bluetooth Low Energy (BLE), que claro es muy bueno porque gasta poca batería y se conecta rápido con cualquier teléfono celular moderno.

Para esta práctica, se utilizó el ESP32-S3, un microcontrolador que tiene mucha potencia y ya trae el Bluetooth integrado. La idea fue sencilla pero muy completa: se crea una app en MIT App Inventor que busca y se conecta al ESP32. Una vez conectados, la app permite mandar comandos para encender o apagar un foco real a través de un relevador.

Para que el usuario sepa qué está pasando en todo momento, se configuro el LED RGB interno de la placa para que cambie de color según el estado: Verde si está libre, Azul si ya se conectó y Rojo si se pierde la conexión. Además, el monitor serial va avisando cada movimiento. Con esto, se logra integrar software móvil y hardware de potencia en un solo sistema funcional y fácil de usar.

## Contenido

### Código Arduino IDE

```
#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           // Pin del sensor led
```

```
#define PIN_FOCO 4          // Pin de Relevador
```

```
#define NUMPIXELS 1         // Cantidad de sensores led
```

```
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; // Conexion a BLE
```

```
bool conectado = false;           // Validacion para Leds
```

```
bool DatoRecibido = false;        // Valida que haya dato
```

```
String palabra = "";             // Guarda el dato resivido
```

```
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();  
    }  
};
```

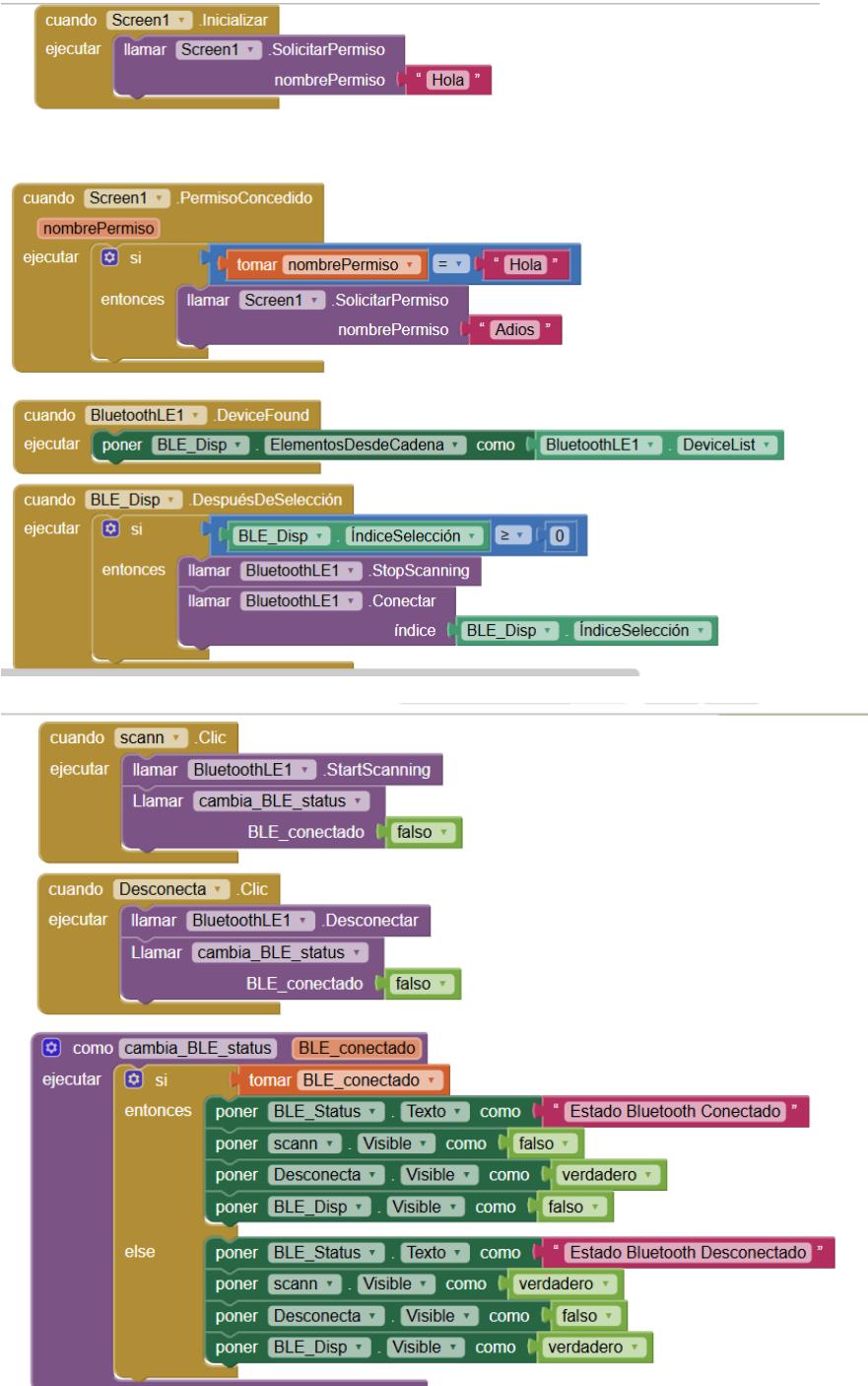
#### // Recepcion de datos que se activa cuando tú envías un texto desde el celular.

```
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(0, 255, 0); // Azul  
}
```

```
void loop() {  
  
    // ESTADO: CONEXIÓN  
    if (dispositivoConectado && !conectado) {  
        setColor(0, 0, 255); // Verde  
        conectado = true;  
    }  
  
    // ESTADO: DESCONEXIÓN  
    if (!dispositivoConectado && conectado) {  
        setColor(255, 0, 0); // Rojo  
        conectado = false;  
        delay(2000);  
        setColor(0, 255, 0); // Azul  
    }  
  
    // Acciones con los datos resvidos  
    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);  
        else if (palabra == "ON") digitalWrite(PIN_FOCO, HIGH);  
        else if (palabra == "OFF") digitalWrite(PIN_FOCO, LOW);;  
  
        DatoRecibido = false;      // Reinicia el dato  
    }  
}
```

## Código App Inventor



```
inicializar global [SERVIDOR_UUID] como "12345678-1234-1234-1234-1234567890ab"
```

```
inicializar global [CHAR_UUID] como "abcd1234-5678-1234-5678-abcdef123456"
```

```
cuando [Verde] .Clic
ejecutar
    llamar [BluetoothLE1] .WriteStrings
        serviceUuid [tomar global SERVIDOR_UUID]
        characteristicUuid [tomar global CHAR_UUID]
        utf16 [falso]
        valores [construir una lista "VERDE"]
```

```
cuando [Rojo] .Clic
ejecutar
    llamar [BluetoothLE1] .WriteStrings
        serviceUuid [tomar global SERVIDOR_UUID]
        characteristicUuid [tomar global CHAR_UUID]
        utf16 [falso]
        valores [construir una lista "ROJO"]
```

```
cuando [Azul] .Clic
ejecutar
    llamar [BluetoothLE1] .WriteStrings
        serviceUuid [tomar global SERVIDOR_UUID]
        characteristicUuid [tomar global CHAR_UUID]
        utf16 [falso]
        valores [construir una lista "AZUL"]
```

```
cuando [Prender] .Clic
ejecutar
    llamar [BluetoothLE1] .WriteStrings
        serviceUuid [tomar global SERVIDOR_UUID]
        characteristicUuid [tomar global CHAR_UUID]
        utf16 [falso]
        valores [construir una lista "ON"]
```

```
cuando [Apagar] .Clic
ejecutar
    llamar [BluetoothLE1] .WriteStrings
        serviceUuid [tomar global SERVIDOR_UUID]
        characteristicUuid [tomar global CHAR_UUID]
        utf16 [falso]
        valores [construir una lista "OFF"]
```

## Materiales

### Hardware (Componentes Físicos)

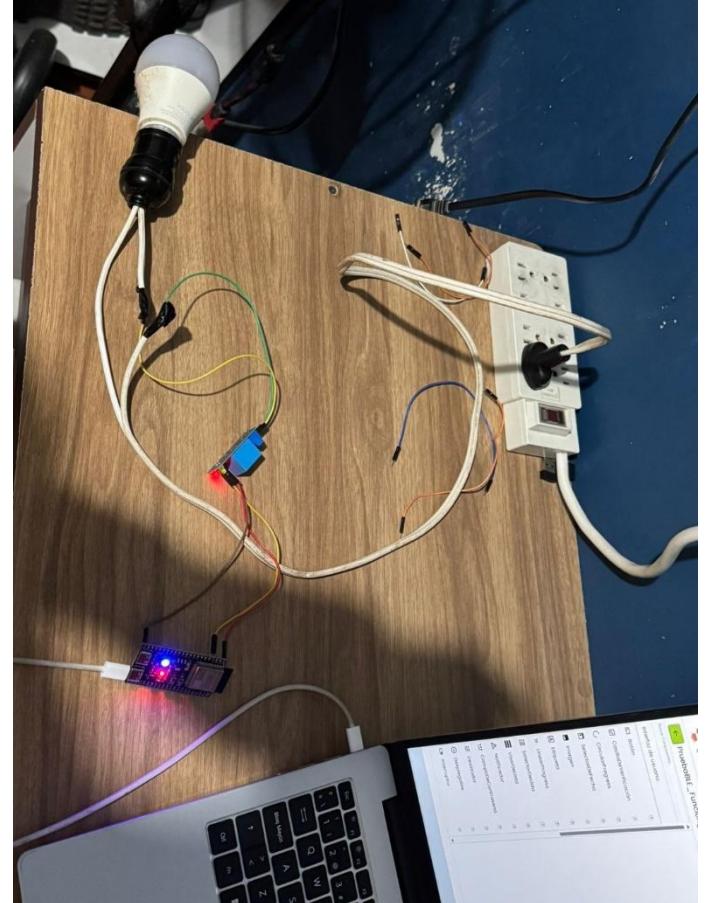
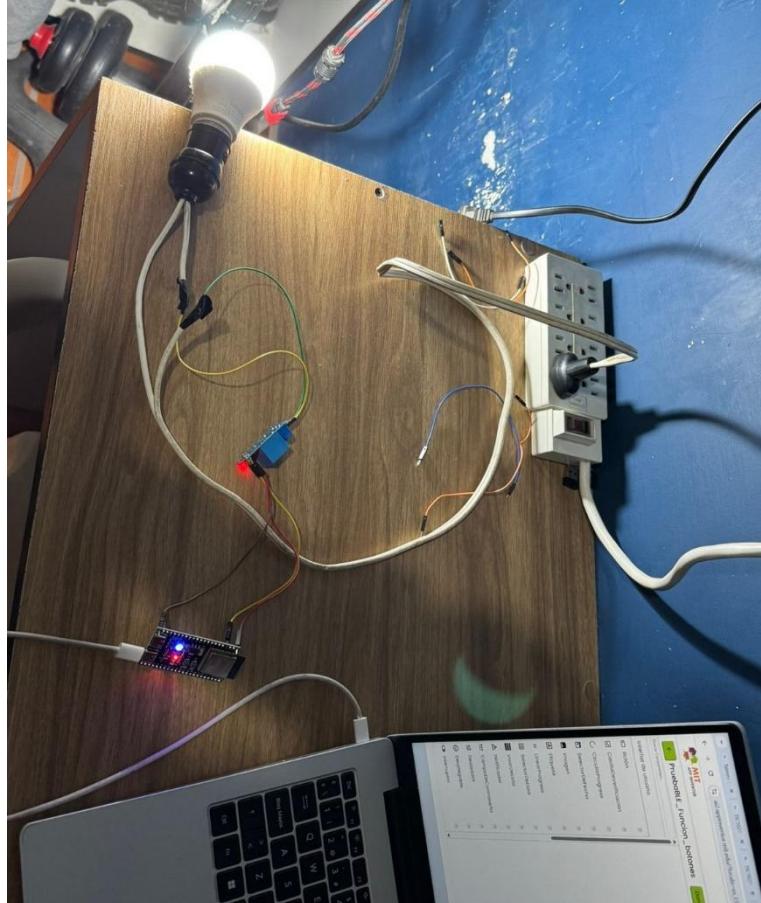
- **Microcontrolador ESP32:** ESP32-S3-WROOM-1-N16R8). Es el cerebro del proyecto, encargado de gestionar la pila de protocolos BLE y controlar las salidas PWM.
- **LED RGB:** (Integrado en el microcontrolador). El actuador que visualiza los estados de conexión y los colores seleccionados.
- **Cable 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.
- **Relevador con modulo:** Modulo con Relevador, este ya incorpora una conexión unificada para el uso con ESP32 y un par de leds indicadores
- **Extensión con socket:** Cable calibre 12 conectado a un socket de foco normal de casa

### 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

Jorge Alberto Casas Demetrio

En esta práctica logre entender que poder conectar el software móvil con el hardware físico no es complicado como parece cuando se logra entender la lógica que existe. Lo más interesante fue observar y comprender como es posible que un simple mensaje enviado desde App Inventor puede convertirse en una acción real, como lo fue activar un relevador y encender un foco. Además, con estas 2 prácticas ya realizadas, logro comprender el uso de las tecnologías inalámbricas e ir más allá del solo utilizarlas.

Jason Alexander Zacarias Garcìa

En esta práctica pude reforzar lo que ya habíamos hecho anteriormente, con esta se reforzo el tema del uso y funcionamiento de BLE, pero ahora implementando una conexión a alto voltaje; con esta práctica me llevo los conocimientos para poder hacer

una casa inteligente y viendo que en verdad no es muy difícil el tema de implementar estas conexiones.