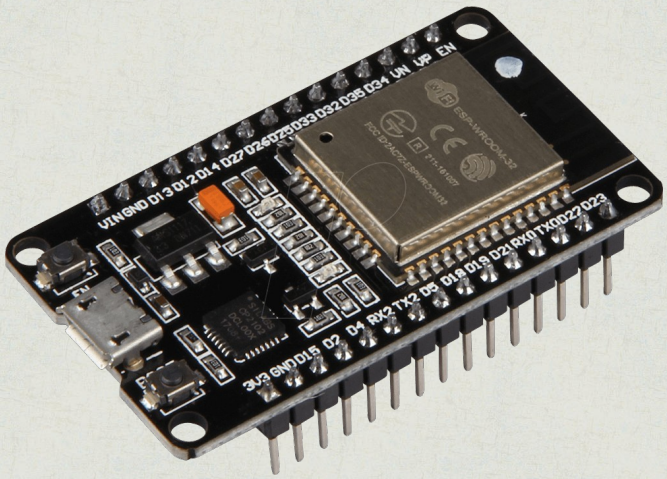
ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Tarea uh3tk4

PRUEBAS INICIALES DE COMUNICACIÓN BLUETOOTH



MATERIA: FP: Desarrollador de Dispositivos IoT

PROFESORES: Gonzalo Vera, Jorge Morales y Dante Violi

ALUMNO: Lisandro Juncos

Objetivos de la Tarea

- Establecer una conexión básica entre el ESP32 y un dispositivo Bluetooth.
- Validar la recepción de comandos vía Bluetooth y su ejecución en el ESP32.
- Probar la capacidad de emparejamiento del ESP32 con varios dispositivos.
- Medir la latencia en la comunicación entre dispositivos Bluetooth

Caso de uso:

Las pruebas fueron realizadas un dispositivo Android con la versión de **Bluetooth 4.2** utilizando la aplicación **Serial Bluetooth Terminal**.
Con respeto al modelo de ESP32, se utilizó una **NodeMCU WROOM ESP32s** de **38 pines**.

Configurar el emparejamiento

Se utilizó la librería **BluethootSerial** para el código de conexión Bluetooth, el cual es el siguiente:

```
#include <Arduino.h>
#include <BluetoothSerial.h>

BluetoothSerial SerialBT; // Crear un objeto BluetoothSerial

void setup() {
  Serial.begin(9600);
  SerialBT.begin("ESP32_NodeMCU"); // Iniciar Bluetooth con el nombre "ESP32_Bluetooth"
  Serial.println("El dispositivo está listo para emparejarse.");
}

void loop() {
  // Si hay datos disponibles desde el Monitor Serie, enviarlos vía Bluetooth
  if (Serial.available()) {
    String message = Serial.readString();  // Leer el mensaje del Monitor Serie
    SerialBT.print(message);               // Enviar el mensaje vía Bluetooth
    Serial.print("Enviando vía Bluetooth: ");
    Serial.println(message);
  }

  // Si hay datos recibidos vía Bluetooth, mostrarlos en el Monitor Serie
  if (SerialBT.available()) {
    String btMessage = SerialBT.readString();  // Leer el mensaje recibido vía Bluetooth
    Serial.print("Mensaje recibido vía Bluetooth: ");
    Serial.println(btMessage);                 // Mostrar el mensaje en el Monitor Serie
  }
}
```

Comandos de control de actuadores

En esta prueba se utilizó el LED integrado de la placa para verificar la recepción y manejo de comandos.
Se determinaron dos comandos "**Prender**" y "**Apagar**", ninguno tuvo errores al momento del control.
A las solicitudes enviadas que no fueran comandos se las toma como un mensaje cualquiera recibido.

Código:

```
void loop() {
 if (SerialBT.available()) {
    String command = SerialBT.readStringUntil('\n'); // Leer comando o mensaje

    command.trim(); // Remover espacios o saltos de línea innecesarios

    if (command == "Prender") {
      digitalWrite(LED_BUILTIN, HIGH); // Encender LED
      SerialBT.println("LED Encendido");
    } else if (command == "Apagar") {
      digitalWrite(LED_BUILTIN, LOW); // Apagar LED
      SerialBT.println("LED Apagado");
    } else {
      // Si no es un comando definido lo tratamos como mensaje
      Serial.println("Mensaje recibido via Bluetooth: " + command);
      SerialBT.println("Mensaje recibido via Bluetooth: " + command);
    }
  }
}
```

Demostración:

Des-conexión y re-conexión manual

Se implementó un código que te avisa cada vez que el dispositivo se conecta al ESP32 y los resultados fueron satisfactorios. De 100 pruebas (50 de conexión y 50 de des-conexión) ninguna presentó problemas.

```
void loop() {
  // Verifica si hay un cambio en el estado de conexión
  if (SerialBT.hasClient() && !connected) {
    // Se ha conectado un cliente
    connected = true;
    Serial.println("Un dispositivo se ha conectado.");
    SerialBT.println("Conexión con el ESP32 establecida.");

  } else if (!SerialBT.hasClient() && connected) {
    // El cliente se ha desconectado
    connected = false;
    Serial.println("El dispositivo se ha desconectado.");
  }
```

Recepción de datos mediante consola serial

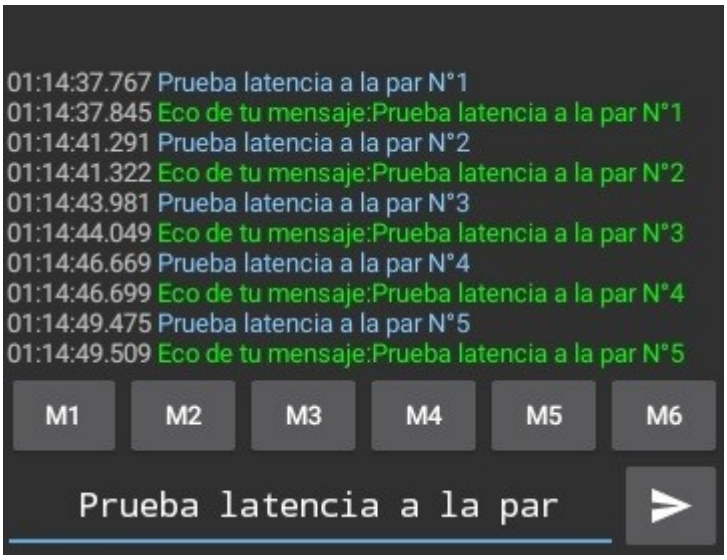
Se pueden recibir muchos tipos de datos mediante la consola serial, tales como: **Comandos de texto** (Como los utilizados en la verificación de actuadores), **Mensajes alfanuméricos**, **Datos numéricos** (enteros o flotantes), **Bytes sin formato** (raw bytes), **Datos Json** y **Comandos AT**.

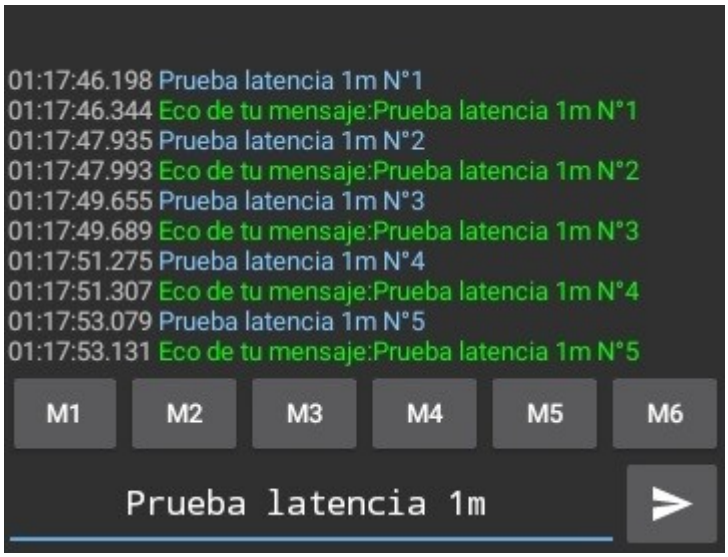
En mis pruebas solo verifiqué la recepción de **mensajes alfanuméricos**, **datos numéricos** y **comandos de texto** y, de nuevo, de las 50 pruebas, todas fueron exitosas.

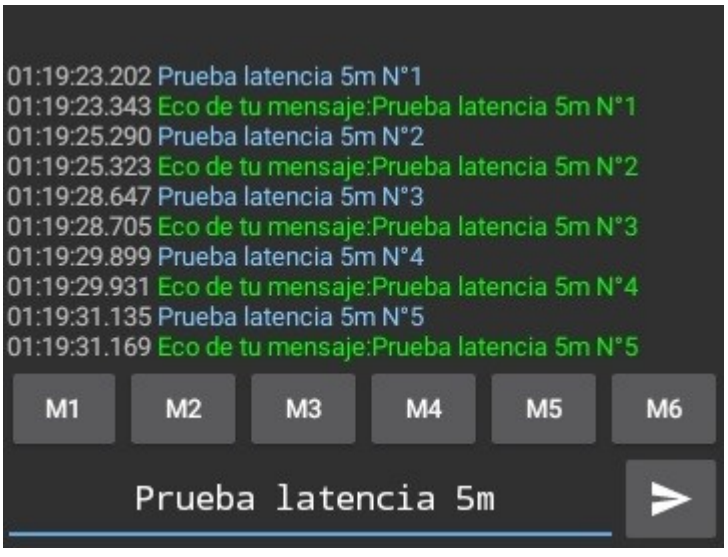
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   PUERTOS

--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
El dispositivo está listo para emparejarse.
Un dispositivo se ha conectado.
Mensaje recibido via Bluetooth: Mensaje N°1
Mensaje recibido via Bluetooth: Mensaje N°2
Mensaje recibido via Bluetooth: Mensaje N°3
Mensaje recibido via Bluetooth: Mensaje N°4
Mensaje recibido via Bluetooth: Mensaje N°5
```

Latencia de respuesta

Para esta prueba utilizamos un código que haga una función de "**eco**", es decir, enviar un mensaje desde el dispositivo conectado (teléfono), recibirlo en el ESP32 y luego devolverlo al dispositivo. El dispositivo mide el tiempo que tomó desde que se envió el mensaje hasta que recibió la respuesta.
Luego de 5 pruebas en diferentes distancias (sin ningún tipo de interferencia) sacamos el promedio y calculamos la latencia para cada una.







El promedio de latencia fue:
A la par: 46.8 ms | **1 metro:** 54.4 ms | **5 metros:** 84.4 ms | **10 metros:** 131.6 ms

Como se puede observar, fueron resultados bastante esperables.