Registro Diario de Avances – Sprint 6 Braille Express

**Fecha:** 15/06/2025

**Autor:** Ylia Jamile Ochoa Gutierrez

**Versión:** 1.0

# 1. Introducción

El presente documento describe las actividades ejecutadas durante el **Sprint 6** del proyecto **Braille Express**, centrado en la integración completa entre la aplicación móvil y el dispositivo físico mediante Bluetooth. Este Sprint tuvo como objetivo principal habilitar el envío de caracteres Braille desde la aplicación al dispositivo Arduino, asegurar la estabilidad de la conexión y brindar al usuario control sobre la reconexión manual en caso de fallos.

Se planificaron un total de **8 tareas**, correspondientes a tres historias de usuario clave:

* HI 3.1 (Conexión Bluetooth)
* HI 3.2 (Envío de caracteres a Arduino)
* HI 3.3 (Reinicio manual de Bluetooth)

El Sprint tuvo una duración estimada de **68 horas**, distribuidas entre desarrollo frontend, backend y pruebas de calidad.

# 2. Lineamientos de Diseño

* **Interfaz clara de Bluetooth**: Indicadores visuales de conexión y botón de reconexión accesible.
* **Compatibilidad con Arduino**: Protocolos simples para enviar caracteres sin errores.
* **Estabilidad**: Se priorizó una conexión fluida y control de fallos.
* **Consistencia visual**: Se respetó la estética general de la app, garantizando accesibilidad y claridad.

# 3. Historias de Usuario y Tareas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-3.1 | Conexión Bluetooth | 10 pts | 29/04/2025 | 19/05/2025 |
| HU-3.2 | Envío de Caracteres a Arduino | 10 pts | 29/04/2025 | 19/05/2025 |
| HU-3.3 | Reinicio manual de Bluetooth | 5 pts | 29/04/2025 | 19/05/2025 |

# 4. Evidencias de Código

## 4.1 Historia de Usuario 2.2: Ingreso de texto por voz

Archivo: fragment\_traductor.java

<TextView  
 android:id="@+id/txtEstadoBluetooth"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="Estado: Desconectado"  
 android:textColor="?android:textColorPrimary"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/swBluetooth"  
 app:layout\_constraintHorizontal\_bias="0"/>  
  
<Switch  
 android:id="@+id/swBluetooth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/txtEstadoBluetooth" />

Archivo: ConexionBluetooth.java

public class ConexionBluetooth {  
  
 private static final UUID *UUID\_SERIAL* = UUID.*fromString*("00001101-0000-1000-8000-00805F9B34FB");  
 private final BluetoothAdapter bluetoothAdapter;  
 private BluetoothSocket bluetoothSocket;  
 private OutputStream outputStream;  
  
 public ConexionBluetooth() {  
 bluetoothAdapter = BluetoothAdapter.*getDefaultAdapter*();  
 }  
  
 @RequiresPermission(allOf = {Manifest.permission.*BLUETOOTH\_CONNECT*, Manifest.permission.*BLUETOOTH\_SCAN*})  
 public boolean conectar(String macAddress, Context context) {  
 if (bluetoothAdapter == null || !bluetoothAdapter.isEnabled()) {  
 Log.*e*("Bluetooth", "Bluetooth no disponible o no habilitado.");  
 return false;  
 }  
  
 try {  
 BluetoothDevice dispositivo = bluetoothAdapter.getRemoteDevice(macAddress);  
 bluetoothSocket = dispositivo.createRfcommSocketToServiceRecord(*UUID\_SERIAL*);  
 bluetoothAdapter.cancelDiscovery();  
 bluetoothSocket.connect();  
 outputStream = bluetoothSocket.getOutputStream();  
  
 Log.*i*("Bluetooth", "Conexión establecida con: " + macAddress);  
 return true;  
  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al conectar: " + e.getMessage());  
 desconectar();  
 return false;  
 }  
 }  
  
 public void desconectar() {  
 try {  
 if (outputStream != null) outputStream.close();  
 if (bluetoothSocket != null) bluetoothSocket.close();  
 Log.*i*("Bluetooth", "Conexión Bluetooth cerrada.");  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al cerrar conexión: " + e.getMessage());  
 }  
 }  
  
 public boolean enviarDatos(String datos) {  
 if (outputStream != null) {  
 try {  
 outputStream.write(datos.trim().getBytes());  
 return true;  
 } catch (IOException e) {  
 Log.*e*("Bluetooth", "Error al enviar datos: " + e.getMessage());  
 }  
 }  
 return false;  
 }  
  
 public boolean estaConectado() {  
 return bluetoothSocket != null && bluetoothSocket.isConnected();  
 }  
}

## 4.2 Historia de Usuario 3.2 — Envío de caracteres a Arduino

Archivo: TraductorFragment.java

// Botón enviar al Arduino  
 btnEnviar.setOnClickListener(v -> {  
 String textoOriginal = txtEntrada.getText().toString();  
 String traduccionBraille = txtSalida.getText().toString();  
  
 if (textoOriginal.isEmpty()) {  
 Toast.*makeText*(requireContext(), "No hay datos para enviar", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 if (conexionBluetooth.estaConectado()) {  
 if (conexionBluetooth.enviarDatos(textoOriginal)) {  
 Toast.*makeText*(requireContext(), "Datos enviados al Arduino", Toast.*LENGTH\_SHORT*).show();  
  
 // Obtener ID del usuario (DNI) desde SharedPreferences  
 String idDNI = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*)  
 .getString("dni\_usuario", "sin\_dni");  
  
  
 // Guardar en Firebase  
 historialRepositorio.guardarTraduccion(textoOriginal, traduccionBraille, idDNI);  
 Toast.*makeText*(requireContext(), "✅ Guardado en historial", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(requireContext(), "Error al enviar datos", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(requireContext(), "Bluetooth no conectado", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 SharedPreferences prefs = requireActivity().getSharedPreferences("user\_session", Context.*MODE\_PRIVATE*);  
 String colorBoton = prefs.getString("color\_boton", "#FF6200EE");  
 try {  
 btnEnviar.setBackgroundColor(Color.*parseColor*(colorBoton));  
 } catch (IllegalArgumentException e) {  
 Log.*e*("ColorError", "Color inválido en configuración: " + colorBoton);  
 }  
  
}

Archivo: BRAILLEEXPRESS.ino

< #include <Servo.h>

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(13, 12); // RX, TX (Conecta TX del HC-06 al pin 13, RX al 12)

Servo servo1;

Servo servo2;

Servo servo3;

Servo servo4;

Servo servo5;

Servo servo6;

const int pinServo1 = 3;

const int pinServo2 = 5;

const int pinServo3 = 6;

const int pinServo4 = 8;

const int pinServo5 = 9;

const int pinServo6 = 10;

const int anguloActivo = 20;

const int anguloInactivo = 0;

void setup() {

  Serial.begin(9600);         // Inicia el Monitor Serial para ver los datos

  bluetooth.begin(9600);      // Inicia el Bluetooth

  servo1.attach(pinServo1);

  servo2.attach(pinServo2);

  servo3.attach(pinServo3);

  servo4.attach(pinServo4);

  servo5.attach(pinServo5);

  servo6.attach(pinServo6);

  resetearServos();

  Serial.println("Sistema Braille listo. Esperando datos por Bluetooth...");

}

void loop() {

  if (bluetooth.available()) {

    char letra = bluetooth.read();

    letra = tolower(letra);

    Serial.print("Letra recibida: ");

    Serial.println(letra);  // Muestra la letra en el monitor serial

    mostrarLetraBraille(letra);

  }

}

void resetearServos() {

  servo1.write(anguloInactivo);

  servo2.write(anguloInactivo);

  servo3.write(anguloInactivo);

  servo4.write(anguloInactivo);

  servo5.write(anguloInactivo);

  servo6.write(anguloInactivo);

}

void mostrarLetraBraille(char letra) {

  resetearServos();

  switch (letra) {

    case 'a': servo1.write(anguloActivo); break;

    case 'b': servo1.write(anguloActivo); servo2.write(anguloActivo); break;

    case 'c': servo1.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'd': servo1.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'e': servo1.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'f': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'g': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'h': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'i': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case 'j': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case 'k': servo1.write(anguloActivo); servo3.write(anguloActivo); break;         // 1,3

    case 'l': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); break; // 1,2,3

    case 'm': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 1,3,4

    case 'n': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 1,3,4,5

    case 'o': servo1.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break; // 1,3,5

    case 'p': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 1,2,3,4

    case 'q': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 1,2,3,4,5

    case 'r': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break; // 1,2,3,5

    case 's': servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); break; // 2,3,4

    case 't': servo2.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break; // 2,3,4,5

    case 'u': servo1.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,6

    case 'v': servo1.write(anguloActivo); servo2.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break; // 1,2,3,6

    case 'w': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 2,4,5,6

    case 'x': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,4,6

    case 'y': servo1.write(anguloActivo); servo3.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,4,5,6

    case 'z': servo1.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break; // 1,3,5,6

    case ' ': break; // 1,3,5,6

    // Números 1–0 representados como letras a–j con el signo numeral (no incluido aquí)

    case '1': servo1.write(anguloActivo); break;

    case '2': servo1.write(anguloActivo); servo2.write(anguloActivo); break;

    case '3': servo1.write(anguloActivo); servo4.write(anguloActivo); break;

    case '4': servo1.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case '5': servo1.write(anguloActivo); servo5.write(anguloActivo); break;

    case '6': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '7': servo1.write(anguloActivo); servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

    case '8': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case '9': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '0': servo2.write(anguloActivo); servo4.write(anguloActivo); servo5.write(anguloActivo); break;

     // Letras acentuadas (adaptación común en español)

    case 'á': servo1.write(anguloActivo); servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'é': servo1.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'í': servo2.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ó': servo1.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ú': servo1.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case 'ñ': servo1.write(anguloActivo); servo2.write(anguloActivo); servo5.write(anguloActivo); servo6.write(anguloActivo); break;

    // Signos de puntuación

    case '.': servo2.write(anguloActivo); servo5.write(anguloActivo); break;

    case ',': servo2.write(anguloActivo); break;

    case ';': servo2.write(anguloActivo); servo3.write(anguloActivo); break;

    case ':': servo2.write(anguloActivo); servo4.write(anguloActivo); break;

    case '?': servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case '!': servo2.write(anguloActivo); servo3.write(anguloActivo); servo5.write(anguloActivo); break;

    case '(': case ')': servo2.write(anguloActivo); servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case '"': servo2.write(anguloActivo); servo6.write(anguloActivo); break;

    case '-': servo3.write(anguloActivo); servo6.write(anguloActivo); break;

    case '\'': servo3.write(anguloActivo); break;

    case '/': servo2.write(anguloActivo); servo4.write(anguloActivo); servo6.write(anguloActivo); break;

    case '@': servo1.write(anguloActivo); servo6.write(anguloActivo); break;

    default:

      Serial.println("Letra no válida o sin asignación Braille.");

      resetearServos();

      return;

  }

  delay(1000);

  resetearServos();

}

## 4.3 Historia de Usuario 3.3 — Reinicio manual de Bluetooth

Archivo: TraductorFragment.java

swBluetooth.setOnCheckedChangeListener((buttonView, isChecked) -> {  
 if (isChecked) {  
 if (conexionBluetooth.conectar(*HC06\_MAC*, requireContext())) {  
 txtEstadoBluetooth.setText("Conectado");  
 Toast.*makeText*(requireContext(), "Bluetooth conectado", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 txtEstadoBluetooth.setText("Error de conexión");  
 Toast.*makeText*(requireContext(), "No se pudo conectar", Toast.*LENGTH\_SHORT*).show();  
 swBluetooth.setChecked(false);  
 }  
 } else {  
 conexionBluetooth.desconectar();  
 txtEstadoBluetooth.setText("Desconectado");  
 Toast.*makeText*(requireContext(), "Bluetooth desconectado", Toast.*LENGTH\_SHORT*).show();  
 }  
});

# 5. Burndown Chart

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **20/05/2025** | **21/05/2025** | **22/05/2025** | **23/05/2025** | **24/05/2025** | **25/05/2025** | **26/05/2025** | **27/05/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** | **Dia8** |
| **HU-3.1** | Conexión Bluetooth | 22 | 12 | 10 |  |  |  |  |  |  |
| **HU-3.2** | Envío de caracteres a Arduino | 47 |  | 2 | 10 | 10 | 8 | 7 | 5 | 5 |
| **HU-3.3** | Reinicio manual de Bluetooth | 6 |  |  |  |  |  |  | 3 | 3 |
| **Tiempo de trabajo** | | 72 | 12 | 12 | 10 | 10 | 8 | 7 | 5 | 8 |

# 6. Análisis del Sprint

* La integración de la conversión automática se realizó con éxito, reduciendo los pasos para el usuario y mejorando la experiencia general.
* Las pruebas iniciales revelaron algunas variaciones en la precisión del reconocimiento de voz, las cuales fueron corregidas mediante mejoras en la normalización del texto transcrito.
* La arquitectura modular del proyecto permitió realizar estas integraciones sin afectar las funciones ya existentes, manteniendo la estabilidad del sistema.
* Se ejecutaron las **8 tareas planificadas** dentro de las **72 horas estimadas.**
* Se completó la interfaz de conexión Bluetooth y se implementó la lógica de envío de caracteres a Arduino.
* Las pruebas confirmaron una **representación física correcta del Braille** y una **conexión estable.**
* La opción de **reinicio manual** funcionó correctamente ante desconexiones simuladas.
* Se resolvieron pequeñas demoras al enviar texto extenso, agregando pausas entre caracteres.

# 7. Conclusiones

El Sprint 6 representó un hito técnico en el desarrollo del proyecto *Braille Express*, al consolidar la conexión entre la aplicación y el hardware, permitiendo la traducción real y tangible del texto a Braille mediante el Arduino.

Las funcionalidades desarrolladas permiten al usuario:

* Visualizar y gestionar la conexión Bluetooth desde la interfaz.
* Enviar texto desde la app y verlo representado físicamente en Braille.
* Reiniciar manualmente la conexión en caso de errores, garantizando continuidad en el uso.

El equipo demostró un alto nivel de coordinación entre desarrollo y pruebas, cumpliendo con los objetivos planteados en tiempo y forma. La integración del hardware con el software fue exitosa y sienta las bases para las últimas fases del proyecto, que buscarán optimizar el rendimiento general y cerrar el ciclo completo de traducción, almacenamiento y representación física.

Este Sprint reafirma el compromiso del equipo con la inclusión digital y tecnológica de personas con discapacidad visual, acercando cada vez más el producto a su versión final funcional.