Objetivo

Crear una **APK Android nativa** (sin apps de terceros) que se conecte por **Bluetooth Clásico (SPP)** al módulo **HC-06**, permita **emparejar / listar / conectar / enviar / recibir** datos y muestre un log en pantalla.

Requisitos

- **Android Studio** (Hedgehog o posterior).
- **Phone Android** con Bluetooth clásico.
- **HC-06** ya cableado al Arduino y configurado (baud típico **9600**).
- PIN de emparejamiento del HC-06: **1234** o **0000** (según el módulo).

> **Compatibilidad**: `minSdk 21`, `targetSdk 34`. Permisos manejados para Android 6–14.

Estructura del proyecto

```
app/
  src/
    main/
      AndroidManifest.xml
      java/com/example/hc06/
        MainActivity.kt
      res/layout/
        activity_main.xml
  build.gradle.kts
settings.gradle.kts
```

---

1) build.gradle.kts (Module: app)

```
plugins {
    id("com.android.application")
    kotlin("android")
}

android {
    namespace = "com.example.hc06"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.hc06"
        minSdk = 21
```

```
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }
    kotlinOptions { jvmTarget = "17" }
}

dependencies {
    implementation("androidx.core:core-ktx:1.13.1")
    implementation("androidx.appcompat:appcompat:1.7.0")
    implementation("com.google.android.material:material:1.12.0")
    implementation("androidx.activity:activity-ktx:1.9.3")
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.8.6")
}
```

---

2) AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Necesario para dispositivos sin BLE; declaramos el feature de
Bluetooth clásico -->
    <uses-feature android:name="android.hardware.bluetooth"
android:required="true" />

    <!-- Permisos pre-Android 12 (API <= 30). No causan error si están
presentes en APIs nuevas -->
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
android:maxSdkVersion="30" />

    <!-- Android 12+ permisos granulados -->
```

```
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />


<!-- Sugerimos marcar el escaneo como no relacionado a ubicación (Android
12+) -->
    <uses-permission
        android:name="android.permission.BLUETOOTH_SCAN"
        android:usesPermissionFlags="neverForLocation" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HC-06 SPP"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

---

3) activity_main.xml (UI simple)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Estado: Idle"
        android:textStyle="bold" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">

        <Button
```

```
            android:id="@+id/btnEnableBt"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Encender BT" />

        <Button
            android:id="@+id/btnPaired"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_marginStart="8dp"
            android:text="Vinculados" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">

        <Button
            android:id="@+id/btnScan"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Escanear" />

        <Button
            android:id="@+id/btnConnect"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_marginStart="8dp"
            android:text="Conectar HC-06" />
    </LinearLayout>

    <ListView
        android:id="@+id/lvDevices"
        android:layout_width="match_parent"
        android:layout_height="160dp"
        android:layout_marginTop="8dp"
        android:dividerHeight="1dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">

        <EditText
```

```
            android:id="@+id/etMessage"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Texto a enviar" />

        <Button
            android:id="@+id/btnSend"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Enviar" />
    </LinearLayout>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:layout_marginTop="8dp">
        <TextView
            android:id="@+id/tvLog"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textIsSelectable="true"
            android:textSize="12sp"
            android:typeface="monospace"
            android:text="" />
    </ScrollView>

</LinearLayout>
```

---

4) MainActivity.kt (Lógica de Bluetooth SPP)

```
package com.example.hc06

import android.Manifest
import android.app.Activity
import android.bluetooth.*
import android.content.*
import android.os.*
import android.widget.*
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat

import java.io.IOException
import java.util.UUID
import kotlin.concurrent.thread
```

```
class MainActivity : AppCompatActivity() {

    private lateinit var tvStatus: TextView
    private lateinit var tvLog: TextView
    private lateinit var lvDevices: ListView
    private lateinit var etMessage: EditText
    private lateinit var btnEnableBt: Button
    private lateinit var btnPaired: Button
    private lateinit var btnScan: Button
    private lateinit var btnConnect: Button
    private lateinit var btnSend: Button

    private val deviceList = mutableListOf<BluetoothDevice>()
    private lateinit var adapterList: ArrayAdapter<String>

    private val sppUUID: UUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")

    private var bluetoothAdapter: BluetoothAdapter? = null
    private var socket: BluetoothSocket? = null
    private var readThread: Thread? = null

    private val enableBtLauncher = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
    ) { /* no-op */ }

    private val receiver = object : BroadcastReceiver() {
        override fun onReceive(context: Context, intent: Intent) {
            when (intent.action) {
                BluetoothDevice.ACTION_FOUND -> {
                    val device: BluetoothDevice? =

intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE)
                    device?.let {
                        if (!deviceList.any { d -> d.address ==
it.address }) {
                            deviceList.add(it)
                            adapterList.add("${it.name ?: "(sin nombre)"} - $
{it.address}")
                        }
                    }
                }
                BluetoothAdapter.ACTION_DISCOVERY_FINISHED -> {
                    log("Escaneo finalizado.")
                }
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        tvStatus = findViewById(R.id.tvStatus)
        tvLog = findViewById(R.id.tvLog)
        lvDevices = findViewById(R.id.lvDevices)
        etMessage = findViewById(R.id.etMessage)
        btnEnableBt = findViewById(R.id.btnEnableBt)
        btnPaired = findViewById(R.id.btnPaired)
        btnScan = findViewById(R.id.btnScan)
        btnConnect = findViewById(R.id.btnConnect)
        btnSend = findViewById(R.id.btnSend)

        adapterList = ArrayAdapter(this,
android.R.layout.simple_list_item_1, mutableListOf())
        lvDevices.adapter = adapterList

        // Obtener el adaptador
        bluetoothAdapter = (getSystemService(BLUETOOTH_SERVICE) as
BluetoothManager).adapter

        // Registrar receiver para descubrimiento
        val filter = IntentFilter().apply {
            addAction(BluetoothDevice.ACTION_FOUND)
            addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED)
        }
        registerReceiver(receiver, filter)

        btnEnableBt.setOnClickListener { ensureBluetoothEnabled() }
        btnPaired.setOnClickListener { listBonded() }
        btnScan.setOnClickListener { scan() }
        btnConnect.setOnClickListener { connectToHc06() }
        btnSend.setOnClickListener { send(etMessage.text.toString()) }

        lvDevices.setOnItemClickListener { _, _, position, _ ->
            val dev = deviceList[position]
            connect(dev)
        }

        requestRuntimePermissions()
    }

    private fun requestRuntimePermissions() {
        val perms = mutableListOf<String>()
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            perms += Manifest.permission.BLUETOOTH_SCAN
            perms += Manifest.permission.BLUETOOTH_CONNECT
        } else {
            // Para descubrir dispositivos en Android 6-11 se requiere
ubicación
            perms += Manifest.permission.ACCESS_FINE_LOCATION
```

```
        }
        val notGranted = perms.filter {
            ContextCompat.checkSelfPermission(this, it) !=
android.content.pm.PackageManager.PERMISSION_GRANTED
        }
        if (notGranted.isNotEmpty()) {
            ActivityCompat.requestPermissions(this,
notGranted.toTypedArray(), 1001)
        }
    }

    private fun ensureBluetoothEnabled() {
        val adapter = bluetoothAdapter ?: return
        if (!adapter.isEnabled) {
            val enableBtIntent =
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
            enableBtLauncher.launch(enableBtIntent)
        } else {
            toast("Bluetooth ya está activado")
        }
    }

    private fun listBonded() {
        adapterList.clear()
        deviceList.clear()
        val bonded = bluetoothAdapter?.bondedDevices ?: emptySet()
        if (bonded.isEmpty()) {

log("No hay dispositivos vinculados. Vaya a Ajustes > Bluetooth y empareje
con el HC-06 (PIN 1234/0000)")
            return
        }
        bonded.forEach {
            deviceList.add(it)
            adapterList.add("${it.name ?: "(sin nombre)"} - ${it.address}")
        }
        log("Mostrando ${bonded.size} vinculados.")
    }

    private fun scan() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S &&
            ContextCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_SCAN) !=
android.content.pm.PackageManager.PERMISSION_GRANTED) {
            toast("Falta permiso BLUETOOTH_SCAN")
            return
        }
        val adapter = bluetoothAdapter ?: return
        if (adapter.isDiscovering) adapter.cancelDiscovery()
        adapterList.clear(); deviceList.clear()
        adapter.startDiscovery()
```

```
        log("Escaneando...")
    }

    private fun connectToHc06() {
        // Intentamos por nombre
        val bonded = bluetoothAdapter?.bondedDevices ?: emptySet()
        val dev = bonded.firstOrNull { (it.name ?: "").contains("HC-06",
ignoreCase = true) }
        if (dev == null) {
            log("HC-06 no vinculado. Use 'Vinculados' o pulse un dispositivo
de la lista tras 'Escanear'.")
        } else connect(dev)
    }

    private fun connect(device: BluetoothDevice) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S &&
            ContextCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) !=
android.content.pm.PackageManager.PERMISSION_GRANTED) {
            toast("Falta permiso BLUETOOTH_CONNECT")
            return
        }
        // Cancelar discovery para acelerar la conexión
        bluetoothAdapter?.cancelDiscovery()

        tvStatus.text = "Estado: Conectando a ${device.name ?:
device.address}"
        log("Conectando a ${device.name} (${device.address})...")

        thread(start = true) {
            try {
                val tmp = device.createRfcommSocketToServiceRecord(sppUUID)
                tmp.connect()
                socket = tmp
                runOnUiThread {
                    tvStatus.text = "Estado: Conectado a ${device.name ?:
device.address}"
                    toast("Conectado")
                }
                startReading()
            } catch (e: IOException) {
                runOnUiThread {
                    tvStatus.text = "Estado: Error de conexión"
                    log("Error conectando: ${e.message}")
                }
                closeSocket()
            }
        }
    }

    private fun startReading() {
```

```
        val s = socket ?: return
        readThread?.interrupt()
        readThread = thread(start = true) {
            val buffer = ByteArray(1024)
            try {
                val input = s.inputStream
                while (!Thread.currentThread().isInterrupted) {
                    val n = input.read(buffer)
                    if (n > 0) {
                        val text = String(buffer, 0, n)
                        runOnUiThread { log("<- $text") }
                    }
                }
            } catch (e: IOException) {
                runOnUiThread { log("Lectura finalizada: ${e.message}") }
            }
        }
    }

    private fun send(msg: String) {
        if (msg.isEmpty()) return
        val s = socket
        if (s == null || !s.isConnected) {
            toast("No conectado")
            return
        }
        thread(start = true) {
            try {
                s.outputStream.write(msg.toByteArray())
                s.outputStream.flush()
                runOnUiThread { log("-> $msg") }
            } catch (e: IOException) {
                runOnUiThread { log("Error enviando: ${e.message}") }
            }
        }
    }

    private fun log(line: String) {
        tvLog.append(line + "\n")
    }

    private fun toast(s: String) = Toast.makeText(this, s,
Toast.LENGTH_SHORT).show()

    private fun closeSocket() {
        try { socket?.close() } catch (_: Exception) {}
        socket = null
        readThread?.interrupt()
        readThread = null
    }
```

```
    override fun onDestroy() {
        super.onDestroy()
        unregisterReceiver(receiver)
        closeSocket()
    }
}
```

---

5) Pasos para compilar y generar la APK

1. Crear proyecto **Empty Views Activity** en Android Studio con *Package name* `com.example.hc06` (o el que gustes).
2. Sustituir/añadir los archivos anteriores en sus rutas.
3. Conectar un **dispositivo físico** (recomendado) y habilitar *USB debugging*.
4. `Build > Build Bundle(s) / APK(s) > Build APK(s)` → obtener APK en `app/build/outputs/apk/`.

---

6) Uso

1. En el teléfono, ir a **Ajustes > Bluetooth** y **emparejar** con *HC-06* (PIN **1234**/**0000**).
2. Abrir la app.
3. **Encender BT** si está apagado.
4. **Vinculados**: lista los emparejados (toca uno para conectar) o **Escanear** para descubrir.
5. **Conectar HC-06**: intenta conectar automáticamente al emparejado llamado “HC-06”.
6. Escribe un texto y pulsa **Enviar**. Los datos recibidos aparecen en el log.

---

7) Troubleshooting (rápido)

- **No encuentra HC-06**: verifica que el módulo esté energizado y visible. Si otro teléfono ya está conectado, el HC-06 no aceptará otra sesión.
- **Falla al conectar**: asegúrate de haber **emparejado antes** en el sistema. Cancela el *scan* antes de conectar (la app lo hace automáticamente).
- **Basura en el monitor serie del Arduino**: revisa el **baud** del `Serial` del Arduino y del módulo HC-06 (habitual 9600).
- **Android 12+**: concede **BLUETOOTH_SCAN/CONNECT**. En Android 6–11, concede **ubicación** para escanear.
- **Reintentos**: si queda en “Estado: Error de conexión”, pulsa nuevamente **Conectar** o elige el dispositivo en la lista.

---

8) Extensiones útiles (opcionales)

- Persistir el **último dispositivo** conectado en `SharedPreferences` y reconectar al abrir.
- Agregar terminador **"\n"** al enviar (`s.outputStream.write("$msg\n".toByteArray())`).
- Parseo de mensajes con **delimitadores** y **buffer**.
- Botones de comandos rápidos ("LED ON", "LED OFF").

- Migrar UI a **Jetpack Compose**.

---

9) Nota sobre seguridad

Esta app se conecta por **Bluetooth Clásico SPP** con el UUID estándar. El tráfico no está cifrado a nivel de aplicación. Para producción, considerar autenticación de comandos a nivel de protocolo.