Got it! Since you're using the **HC-05 Bluetooth module**, which uses **Bluetooth SPP (Serial Port Profile)**, you don't need to worry about specific **Bluetooth Low Energy (BLE) UUIDs**. The HC-05 communicates over **Bluetooth Classic** (not BLE), so it operates with serial communication instead of BLE's characteristics and services.

Here’s an updated version of the app where we **use serial communication** to send data to the HC-05 module. This will work directly with the serial data protocol for Bluetooth Classic (SPP), which your Arduino code listens for ('1' or '0' to control the LED).

We will use the **react-native-bluetooth-serial** library instead of react-native-ble-plx because react-native-ble-plx is for **Bluetooth Low Energy (BLE)**, while the HC-05 uses **Bluetooth Classic**. This library will allow us to connect to the HC-05 via serial communication and send the 1 or 0 to control the LED.

**Steps:**

1. Install the react-native-bluetooth-serial package:
2. npm install react-native-bluetooth-serial --save
3. Link the library (for older versions of React Native, or if auto-linking doesn’t work):
4. react-native link react-native-bluetooth-serial
5. Update your BLEScreen.tsx component to use the Bluetooth Classic communication.

**BLEScreen.tsx (Updated with Bluetooth Classic Communication):**

import React, { useState, useEffect } from 'react';

import { ScrollView, Button, Text, View, Platform, PermissionsAndroid } from 'react-native';

import BluetoothSerial from 'react-native-bluetooth-serial';

const BLEScreen = () => {

const [devices, setDevices] = useState<any[]>([]); // Stores paired devices

const [connectedDevice, setConnectedDevice] = useState<any | null>(null);

const [isScanning, setIsScanning] = useState<boolean>(false);

const [isConnected, setIsConnected] = useState<boolean>(false);

// Request Android permissions for Bluetooth

const requestAndroidPermissions = async () => {

if (Platform.OS === 'android') {

const bluetoothPermission = await PermissionsAndroid.request(

PermissionsAndroid.PERMISSIONS.BLUETOOTH,

);

const bluetoothAdminPermission = await PermissionsAndroid.request(

PermissionsAndroid.PERMISSIONS.BLUETOOTH\_ADMIN,

);

const locationPermission = await PermissionsAndroid.request(

PermissionsAndroid.PERMISSIONS.ACCESS\_FINE\_LOCATION,

);

return (

bluetoothPermission === 'granted' &&

bluetoothAdminPermission === 'granted' &&

locationPermission === 'granted'

);

}

return true; // Automatically handled on iOS

};

useEffect(() => {

const initPermissions = async () => {

const hasPermission = await requestAndroidPermissions();

if (!hasPermission) {

alert('Permissions are required to use Bluetooth');

}

};

initPermissions();

}, []);

// Scan for Bluetooth devices

const scanForDevices = async () => {

setIsScanning(true);

try {

await BluetoothSerial.discoverUnpairedDevices();

const pairedDevices = await BluetoothSerial.list();

setDevices(pairedDevices); // Show paired devices

} catch (error) {

console.error('Failed to scan devices', error);

}

};

// Connect to the selected device (HC-05)

const connectToDevice = async (device: any) => {

try {

await BluetoothSerial.connect(device.id);

setConnectedDevice(device);

setIsConnected(true);

setIsScanning(false); // Stop scanning after connecting

} catch (e) {

console.log('Failed to connect to device', e);

}

};

// Send data to Arduino via HC-05 (Control the LED)

const sendData = async (data: string) => {

if (connectedDevice) {

try {

await BluetoothSerial.write(data);

console.log(`Sent data: ${data}`);

} catch (e) {

console.log('Error sending data', e);

}

} else {

alert('No device connected');

}

};

// Disconnect from the device

const disconnectDevice = async () => {

if (connectedDevice) {

try {

await BluetoothSerial.disconnect();

setIsConnected(false);

setConnectedDevice(null);

} catch (e) {

console.log('Error disconnecting', e);

}

}

};

return (

<ScrollView contentContainerStyle={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

<View style={{ padding: 20 }}>

<Text style={{ fontSize: 24, marginBottom: 20 }}>Control LED via Bluetooth</Text>

{/\* Start/Stop Scanning \*/}

<Button title={isScanning ? 'Stop Scanning' : 'Scan for Devices'} onPress={scanForDevices} />

{/\* List of paired devices \*/}

<View style={{ marginTop: 20 }}>

<Text style={{ fontSize: 18 }}>Paired Devices:</Text>

{devices.length === 0 ? (

<Text>No devices found</Text>

) : (

devices.map((device) => (

<Button

key={device.id}

title={`Connect to ${device.name || device.id}`}

onPress={() => connectToDevice(device)}

/>

))

)}

</View>

{/\* Connect/Disconnect Buttons \*/}

{isConnected && connectedDevice && (

<View style={{ marginTop: 20 }}>

<Text>Connected to: {connectedDevice.name || 'Unknown Device'}</Text>

<Button title="Disconnect" onPress={disconnectDevice} />

</View>

)}

{/\* LED control buttons \*/}

{isConnected && (

<View style={{ marginTop: 20 }}>

<Button title="Turn LED ON" onPress={() => sendData('1')} />

<Button title="Turn LED OFF" onPress={() => sendData('0')} />

</View>

)}

</View>

</ScrollView>

);

};

export default BLEScreen;

**Key Changes:**

1. **Using react-native-bluetooth-serial**: We replaced react-native-ble-plx with react-native-bluetooth-serial to support Bluetooth Classic (SPP) communication with the HC-05.
2. **Scanning for Devices**: Instead of using BLE scanning methods, we use BluetoothSerial.discoverUnpairedDevices() to scan for devices and BluetoothSerial.list() to list paired devices.
3. **Sending Data**: We use BluetoothSerial.write(data) to send data ('1' to turn on the LED and '0' to turn it off) to the Arduino through the HC-05.
4. **No UUIDs**: Since the HC-05 uses Bluetooth Classic, we no longer need to deal with UUIDs, unlike with Bluetooth Low Energy (BLE).

**Testing the App:**

1. **Hardware Requirements**:
   * Your Arduino should be connected to the HC-05 Bluetooth module.
   * Make sure the Arduino code is running, and it's expecting serial commands ('1' to turn the LED on and '0' to turn it off).
2. **Run the App**:
   * Build and run the app on a physical device (not a simulator) to test Bluetooth functionality.
   * Scan for the HC-05 device, connect to it, and use the buttons to control the LED on the Arduino.

This approach should now work with your **HC-05 Bluetooth module** using **Bluetooth Classic** and serial communication. Let me know if you need further assistance!