Android-er

For Android development, from beginner to beginner

Thursday, June 9, 2016

Scan specified BLE devices with ScanFilter

The former post show how to scan Bluetooth LE device with BluetoothLeScanner, accept all BLE devices. We can scan ScanFilter to scan specified BLE devices only.

Edit scanLeDevice() method of MainActivity.java in last post.

```java
package com.blogspot.android_er.androidbleget;

import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanCallback;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothLeScanner mBluetoothLeScanner;

    private boolean mScanning;

    private static final int POS_ENABLE_BLUETOOTH = 1;

    Button btnScan;
    ListView listViewLE;

    List<BluetoothDevice> listBluetoothDevice;
    ListAdapter adapterLeScanResult;

    private Handler mHandler;
    private static final long SCAN_PERIOD = 10000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Check if BLE is supported on the device.
        if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
            Toast.makeText(this,
                    "BLUETOOTH_LE not supported in this device!",
                    Toast.LENGTH_SHORT).show();
            finish();
        }

        getBluetoothAdapterAndLeScanner();

        // Checks if Bluetooth is supported on the device.
        if (mBluetoothAdapter == null) {
            Toast.makeText(this,
                    "bluetoothManager.getAdapter()==null",
                    Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        btnScan = (Button)findViewById(R.id.scan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                scanLeDevice(true);
            }
        });

        listViewLE = (ListView)findViewById(R.id.lelist);

        listBluetoothDevice = new ArrayList<>();
        adapterLeScanResult = new ArrayAdapter<BluetoothDevice>(
                this, android.R.layout.simple_list_item_1, listBluetoothDevice);
        listViewLE.setAdapter(adapterLeScanResult);
        listViewLE.setOnItemClickListener(scanResultOnItemClickListener);

        mHandler = new Handler();
    }

    AdapterView.OnItemClickListener scanResultOnItemClickListener =
            new AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                    final BluetoothDevice device =
                            (BluetoothDevice) parent.getItemAtPosition(position);

                    String msg = device.getAddress() + "\n"
                            + device.getBondState(null).toString() + "\n"
                            + getRSSerialType(device)();

                    new AlertDialog.Builder(MainActivity.this)
                            .setTitle(device.getName())
                            .setMessage(msg)
                            .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                }
                            })
                            .show();
                }
            };

    private String getBTDeviceType(BluetoothDevice d){
        String type = "";

        switch (d.getType()){
            case BluetoothDevice.DEVICE_TYPE_CLASSIC:
                type = "DEVICE_TYPE_CLASSIC";
                break;
            case BluetoothDevice.DEVICE_TYPE_DUAL:
                type = "DEVICE_TYPE_DUAL";
                break;
            case BluetoothDevice.DEVICE_TYPE_LE:
                type = "DEVICE_TYPE_LE";
                break;
            case BluetoothDevice.DEVICE_TYPE_UNKNOWN:
                type = "DEVICE_TYPE_UNKNOWN";
                break;
            default:
                type = "unknown...";
        }

        return type;
    }

    @Override
    protected void onResume() {
        super.onResume();

        if (!mBluetoothAdapter.isEnabled()) {
            if (!mBluetoothAdapter.isEnabled()) {
                Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, REQ_ENABLE_BLUETOOTH);
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        if (requestCode == REQ_ENABLE_BLUETOOTH && resultCode == Activity.RESULT_CANCELED) {
            finish();
            return;
        }

        getBluetoothAdapterAndLeScanner();

        // Checks if Bluetooth is supported on the device.
        if (mBluetoothAdapter == null) {
            Toast.makeText(this,
                    "bluetoothManager.getAdapter()==null",
                    Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        super.onActivityResult(requestCode, resultCode, data);
    }

    private void getBluetoothAdapterAndLeScanner(){
        // Get BluetoothAdapter and BluetoothLeScanner.
        final BluetoothManager bluetoothManager =
                (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
        mBluetoothAdapter = bluetoothManager.getAdapter();
        mBluetoothLeScanner = mBluetoothAdapter.getBluetoothLeScanner();

        mScanning = false;
    }

    /*
    to call startScan (ScanCallback callback),
    Requires BLUETOOTH_ADMIN permission.
    Must hold ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission to get results.
    */
    private void scanLeDevice(final boolean enable) {
        if (enable) {
            btnScan.invalidate(false, true);
            listViewLE.invalidateViews();

            // Stops scanning after a pre-defined scan period.
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    mBluetoothLeScanner.stopScan(scanCallback);
                    listViewLE.invalidateViews();

                    Toast.makeText(MainActivity.this,
                            "Scan timeout",
                            Toast.LENGTH_LONG).show();

                    mScanning = false;
                    btnScan.setEnabled(true);
                }
            }, SCAN_PERIOD);

            //mBluetoothLeScanner.startScan(scanCallback);

            //scan specified devices only with ScanFilter
            ScanFilter scanFilter = new ScanFilter.Builder()
                    .setDeviceName("LED0")
                    .setServiceUuid(ParcelUuid.fromString(BluetoothLeService.UUID_SERVICE_GENUINO101_ledService))
                    .build();
            List<ScanFilter> scanFilters = new ArrayList<ScanFilter>();
            scanFilters.add(scanFilter);

            ScanSettings scanSettings =
                    new ScanSettings.Builder().build();

            mBluetoothLeScanner.startScan(scanFilters, scanSettings, scanCallback);

            mScanning = true;
            btnScan.setEnabled(false);
        } else {
            mScanning = false;
            mBluetoothLeScanner.stopScan(scanCallback);
            mScanning = false;
            btnScan.setEnabled(true);
        }
    }

    private ScanCallback scanCallback = new ScanCallback() {
        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);
            addBluetoothDevice(result.getDevice());
        }

        @Override
        public void onBatchScanResults(List<ScanResult> results) {
            super.onBatchScanResults(results);
            for(ScanResult result : results){
                addBluetoothDevice(result.getDevice());
            }
        }

        @Override
        public void onScanFailed(int errorCode) {
            super.onScanFailed(errorCode);
            Toast.makeText(MainActivity.this,
                    "onScanFailed: " + String.valueOf(errorCode),
                    Toast.LENGTH_LONG).show();
        }

        private void addBluetoothDevice(BluetoothDevice device){
            if(!listBluetoothDevice.contains(device)){
                listBluetoothDevice.add(device);
                listViewLE.invalidateViews();
            }
        }
    };
}
```

Next:
- Bluetooth LE Gatt Example - connect to Bluetooth LE device and display GATT Services

Related:
- Android code sample: Bluetooth Low Energy, Bluetooth LE Gatt Example

by: Erik   [icons]

2 comments:

calaMaxi said...

...can you make this working with intent ACTION_SCREEN_OFF and put all into logcba)a ?

June 22, 2016 at 3:00 PM

Unknown said...

These code works perfect for FAT0; however, ScanFilter has failed for microchip BHT7 BLE module, without putting the scan filter. I can search both of my HM10 and BHT7, why these what could be cause?

December 17, 2016 at 10:50 AM

Post a Comment

Newer Post   Home   Older Post

Subscribe to: Post Comments (Atom)