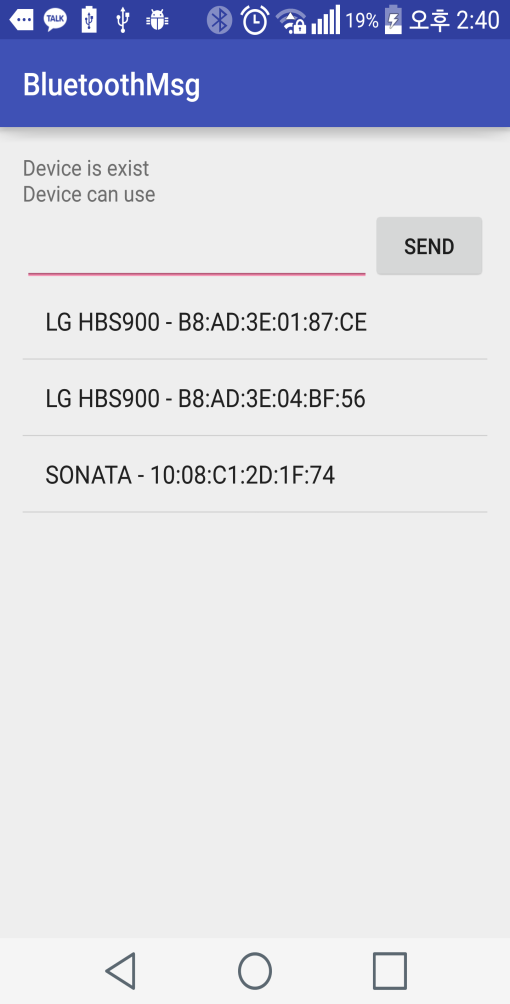
1. 권한설정  
<**uses-permission android:name="android.permission.BLUETOOTH"**/>  
<**uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"**/>

2. Layout



3. JavaCode

**import** android.bluetooth.BluetoothAdapter;  
**import** android.bluetooth.BluetoothDevice;  
**import** android.bluetooth.BluetoothServerSocket;  
**import** android.bluetooth.BluetoothSocket;  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.IntentFilter;  
**import** android.os.Bundle;  
**import** android.os.Message;  
**import** android.os.SystemClock;  
**import** android.os.Handler;  
  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.OutputStream;  
**import** java.util.ArrayList;  
**import** java.util.Set;  
**import** java.util.UUID;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **static final int *ACTION\_ENABLE\_BT*** = 101;  
 TextView **mTextMsg**;  
 EditText **mEditData**;  
 BluetoothAdapter **mBA**;  
 ListView **mListDevice**;  
 ArrayList<String> **mArDevice**; *// 원격 디바이스 목록* **static final** String ***BLUE\_NAME*** = **"BluetoothEx"**; *// 접속시 사용하는 이름  
 // 접속시 사용하는 고유 ID* **static final** UUID ***BLUE\_UUID*** = UUID.*fromString*(**"00001101-0000-1000-8000-00805F9B34FB"**);  
 ClientThread **mCThread** = **null**; *// 클라이언트 소켓 접속 스레드* ServerThread **mSThread** = **null**; *// 서버 소켓 접속 스레드* SocketThread **mSocketThread** = **null**; *// 데이터 송수신 스레드* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **mTextMsg** = (TextView)findViewById(R.id.***textMessage***);  
 **mEditData** = (EditText)findViewById(R.id.***editData***);  
 *// ListView 초기화* initListView();  
  
 *// 블루투스 사용 가능상태 판단* **boolean** isBlue = canUseBluetooth();  
 **if**( isBlue )  
 *// 페어링된 원격 디바이스 목록 구하기* getParedDevice();  
 }  
  
 *// 블루투스 사용 가능상태 판단* **public boolean** canUseBluetooth() {  
 *// 블루투스 어댑터를 구한다* **mBA** = BluetoothAdapter.*getDefaultAdapter*();  
 *// 블루투스 어댑터가 null 이면 블루투스 장비가 존재하지 않는다.* **if**( **mBA** == **null** ) {  
 **mTextMsg**.setText(**"Device not found"**);  
 **return false**;  
 }  
  
 **mTextMsg**.setText(**"Device is exist"**);  
 *// 블루투스 활성화 상태라면 함수 탈출* **if**( **mBA**.isEnabled() ) {  
 **mTextMsg**.append(**"\nDevice can use"**);  
 **return true**;  
 }  
  
 *// 사용자에게 블루투스 활성화를 요청한다* Intent intent = **new** Intent(BluetoothAdapter.***ACTION\_REQUEST\_ENABLE***);  
 startActivityForResult(intent, ***ACTION\_ENABLE\_BT***);  
 **return false**;  
 }  
  
 *// 블루투스 활성화 요청 결과 수신* **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **if**( requestCode == ***ACTION\_ENABLE\_BT*** ) {  
 *// 사용자가 블루투스 활성화 승인했을때* **if**( resultCode == ***RESULT\_OK*** ) {  
 **mTextMsg**.append(**"\nDevice can use"**);  
 *// 페어링된 원격 디바이스 목록 구하기* getParedDevice();  
 }  
 *// 사용자가 블루투스 활성화 취소했을때* **else** {  
 **mTextMsg**.append(**"\nDevice can not use"**);  
 }  
 }  
 }  
  
 *// 원격 디바이스 검색 시작* **public void** startFindDevice() {  
 *// 원격 디바이스 검색 중지* stopFindDevice();  
 *// 디바이스 검색 시작* **mBA**.startDiscovery();  
 *// 원격 디바이스 검색 이벤트 리시버 등록* registerReceiver(**mBlueRecv**, **new** IntentFilter( BluetoothDevice.***ACTION\_FOUND*** ));  
 }  
  
 *// 디바이스 검색 중지* **public void** stopFindDevice() {  
 *// 현재 디바이스 검색 중이라면 취소한다* **if**( **mBA**.isDiscovering() ) {  
 **mBA**.cancelDiscovery();  
 *// 브로드캐스트 리시버를 등록 해제한다* unregisterReceiver(**mBlueRecv**);  
 }  
 }  
  
 *// 원격 디바이스 검색 이벤트 수신* BroadcastReceiver **mBlueRecv** = **new** BroadcastReceiver() {  
 **public void** onReceive(Context context, Intent intent) {  
 **if**( intent.getAction() == BluetoothDevice.***ACTION\_FOUND*** ) {  
 *// 인텐트에서 디바이스 정보 추출* BluetoothDevice device = intent.getParcelableExtra( BluetoothDevice.***EXTRA\_DEVICE*** );  
 *// 페어링된 디바이스가 아니라면* **if**( device.getBondState() != BluetoothDevice.***BOND\_BONDED*** )  
 *// 디바이스를 목록에 추가* addDeviceToList(device.getName(), device.getAddress());  
 }  
 }  
 };  
  
 *// 디바이스를 ListView 에 추가* **public void** addDeviceToList(String name, String address) {  
 *// ListView 와 연결된 ArrayList 에 새로운 항목을 추가* String deviceInfo = name + **" - "** + address;  
 Log.*d*(**"tag1"**, **"Device Find: "** + deviceInfo);  
 **mArDevice**.add(deviceInfo);  
 *// 화면을 갱신한다* ArrayAdapter adapter = (ArrayAdapter)**mListDevice**.getAdapter();  
 adapter.notifyDataSetChanged();  
 }  
  
 *// ListView 초기화* **public void** initListView() {  
 *// 어댑터 생성* **mArDevice** = **new** ArrayList<String>();  
 ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**,  
 android.R.layout.***simple\_list\_item\_1***, **mArDevice**);  
 *// ListView 에 어댑터와 이벤트 리스너를 지정* **mListDevice** = (ListView)findViewById(R.id.***listDevice***);  
 **mListDevice**.setAdapter(adapter);  
 **mListDevice**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 *// 사용자가 선택한 항목의 내용을 구한다* String strItem = **mArDevice**.get(position);  
  
 *// 사용자가 선택한 디바이스의 주소를 구한다* **int** pos = strItem.indexOf(**" - "**);  
 **if**( pos <= 0 ) **return**;  
 String address = strItem.substring(pos + 3);  
 **mTextMsg**.setText(**"Sel Device: "** + address);  
  
 *// 디바이스 검색 중지* stopFindDevice();  
 *// 서버 소켓 스레드 중지* **mSThread**.cancel();  
 **mSThread** = **null**;  
  
 **if**( **mCThread** != **null** ) **return**;  
 *// 상대방 디바이스를 구한다* BluetoothDevice device = **mBA**.getRemoteDevice(address);  
 *// 클라이언트 소켓 스레드 생성 & 시작* **mCThread** = **new** ClientThread(device);  
 **mCThread**.start();  
  
  
 }  
 });  
 }  
  
 *// 다른 디바이스에게 자신을 검색 허용* **public void** setDiscoverable() {  
 *// 현재 검색 허용 상태라면 함수 탈출* **if**( **mBA**.getScanMode() == BluetoothAdapter.***SCAN\_MODE\_CONNECTABLE\_DISCOVERABLE*** )  
 **return**;  
 *// 다른 디바이스에게 자신을 검색 허용 지정* Intent intent = **new** Intent(BluetoothAdapter.***ACTION\_REQUEST\_DISCOVERABLE***);  
 intent.putExtra(BluetoothAdapter.***EXTRA\_DISCOVERABLE\_DURATION***, 0);  
 startActivity(intent);  
 }  
  
 *// 페어링된 원격 디바이스 목록 구하기* **public void** getParedDevice() {  
 **if**( **mSThread** != **null** ) **return**;  
 *// 서버 소켓 접속을 위한 스레드 생성 & 시작* **mSThread** = **new** ServerThread();  
 **mSThread**.start();  
  
 *// 블루투스 어댑터에서 페어링된 원격 디바이스 목록을 구한다* Set<BluetoothDevice> devices = **mBA**.getBondedDevices();  
 *// 디바이스 목록에서 하나씩 추출* **for**( BluetoothDevice device : devices ) {  
 *// 디바이스를 목록에 추가* addDeviceToList(device.getName(), device.getAddress());  
 }  
  
 *// 원격 디바이스 검색 시작* startFindDevice();  
  
 *// 다른 디바이스에 자신을 노출* setDiscoverable();  
 }  
  
  
 *// 클라이언트 소켓 생성을 위한 스레드* **private class** ClientThread **extends** Thread {  
 **private** BluetoothSocket **mmCSocket**;  
  
 *// 원격 디바이스와 접속을 위한 클라이언트 소켓 생성* **public** ClientThread(BluetoothDevice device) {  
 **try** {  
 **mmCSocket** = device.createInsecureRfcommSocketToServiceRecord(***BLUE\_UUID***);  
 } **catch**(IOException e) {  
 showMessage(**"Create Client Socket error"**);  
 **return**;  
 }  
 }  
  
 **public void** run() {  
 *// 원격 디바이스와 접속 시도* **try** {  
 **mmCSocket**.connect();  
 } **catch**(IOException e) {  
 showMessage(**"Connect to server error"**);  
 *// 접속이 실패했으면 소켓을 닫는다* **try** {  
 **mmCSocket**.close();  
 } **catch** (IOException e2) {  
 showMessage(**"Client Socket close error"**);  
 }  
 **return**;  
 }  
  
 *// 원격 디바이스와 접속되었으면 데이터 송수신 스레드를 시작* onConnected(**mmCSocket**);  
 }  
  
 *// 클라이언트 소켓 중지* **public void** cancel() {  
 **try** {  
 **mmCSocket**.close();  
 } **catch** (IOException e) {  
 showMessage(**"Client Socket close error"**);  
 }  
 }  
 }  
  
 *// 서버 소켓을 생성해서 접속이 들어오면 클라이언트 소켓을 생성하는 스레드* **private class** ServerThread **extends** Thread {  
 **private** BluetoothServerSocket **mmSSocket**;  
  
 *// 서버 소켓 생성* **public** ServerThread() {  
 **try** {  
 **mmSSocket** = **mBA**.listenUsingInsecureRfcommWithServiceRecord(***BLUE\_NAME***, ***BLUE\_UUID***);  
 } **catch**(IOException e) {  
 showMessage(**"Get Server Socket Error"**);  
 }  
 }  
  
 **public void** run() {  
 BluetoothSocket cSocket = **null**;  
  
 *// 원격 디바이스에서 접속을 요청할 때까지 기다린다* **try** {  
 cSocket = **mmSSocket**.accept();  
 } **catch**(IOException e) {  
 showMessage(**"Socket Accept Error"**);  
 **return**;  
 }  
  
 *// 원격 디바이스와 접속되었으면 데이터 송수신 스레드를 시작* onConnected(cSocket);  
 }  
  
 *// 서버 소켓 중지* **public void** cancel() {  
 **try** {  
 **mmSSocket**.close();  
 } **catch** (IOException e) {  
 showMessage(**"Server Socket close error"**);  
 }  
 }  
 }  
  
 *// 메시지를 화면에 표시* **public void** showMessage(String strMsg) {  
 *// 메시지 텍스트를 핸들러에 전달* Message msg = Message.*obtain*(**mHandler**, 0, strMsg);  
 **mHandler**.sendMessage(msg);  
 Log.*d*(**"tag1"**, strMsg);  
 }  
  
 *// 메시지 화면 출력을 위한 핸들러* Handler **mHandler** = **new** Handler() {  
 **public void** handleMessage(Message msg) {  
 **if** (msg.**what** == 0) {  
 String strMsg = (String)msg.**obj**;  
 **mTextMsg**.setText(strMsg);  
 }  
 }  
 };  
  
 *// 원격 디바이스와 접속되었으면 데이터 송수신 스레드를 시작* **public void** onConnected(BluetoothSocket socket) {  
 showMessage(**"Socket connected"**);  
  
 *// 데이터 송수신 스레드가 생성되어 있다면 삭제한다* **if**( **mSocketThread** != **null** )  
 **mSocketThread** = **null**;  
 *// 데이터 송수신 스레드를 시작* **mSocketThread** = **new** SocketThread(socket);  
 **mSocketThread**.start();  
 }  
  
 *// 데이터 송수신 스레드* **private class** SocketThread **extends** Thread {  
 **private final** BluetoothSocket **mmSocket**; *// 클라이언트 소켓* **private** InputStream **mmInStream**; *// 입력 스트림* **private** OutputStream **mmOutStream**; *// 출력 스트림* **public** SocketThread(BluetoothSocket socket) {  
 **mmSocket** = socket;  
  
 *// 입력 스트림과 출력 스트림을 구한다* **try** {  
 **mmInStream** = socket.getInputStream();  
 **mmOutStream** = socket.getOutputStream();  
 } **catch** (IOException e) {  
 showMessage(**"Get Stream error"**);  
 }  
 }  
  
 *// 소켓에서 수신된 데이터를 화면에 표시한다* **public void** run() {  
 **byte**[] buffer = **new byte**[1024];  
 **int** bytes;  
  
 **while** (**true**) {  
 **try** {  
 *// 입력 스트림에서 데이터를 읽는다* bytes = **mmInStream**.read(buffer);  
 String strBuf = **new** String(buffer, 0, bytes);  
 showMessage(**"Receive: "** + strBuf);  
 SystemClock.*sleep*(1);  
 } **catch** (IOException e) {  
 showMessage(**"Socket disconneted"**);  
 **break**;  
 }  
 }  
 }  
  
 *// 데이터를 소켓으로 전송한다* **public void** write(String strBuf) {  
 **try** {  
 *// 출력 스트림에 데이터를 저장한다* **byte**[] buffer = strBuf.getBytes();  
 **mmOutStream**.write(buffer);  
 showMessage(**"Send: "** + strBuf);  
 } **catch** (IOException e) {  
 showMessage(**"Socket write error"**);  
 }  
 }  
 }  
  
 *// 버튼 클릭 이벤트 함수* **public void** onClick(View v) {  
 **switch**( v.getId() ) {  
 **case** R.id.***btnSend*** : {  
 *// 데이터 송수신 스레드가 생성되지 않았다면 함수 탈출* **if**( **mSocketThread** == **null** ) **return**;  
 *// 사용자가 입력한 텍스트를 소켓으로 전송한다* String strBuf = **mEditData**.getText().toString();  
 **if**( strBuf.length() < 1 ) **return**;  
 **mEditData**.setText(**""**);  
 **mSocketThread**.write(strBuf);  
 **break**;  
 }  
 }  
 }  
  
 *// 앱이 종료될 때 디바이스 검색 중지* **public void** onDestroy() {  
 **super**.onDestroy();  
 *// 디바이스 검색 중지* stopFindDevice();  
  
 *// 스레드를 종료* **if**( **mCThread** != **null** ) {  
 **mCThread**.cancel();  
 **mCThread** = **null**;  
 }  
 **if**( **mSThread** != **null** ) {  
 **mSThread**.cancel();  
 **mSThread** = **null**;  
 }  
 **if**( **mSocketThread** != **null** )  
 **mSocketThread** = **null**;  
 }  
  
}