Support Note

23 August, 2019

LOKUS GNSS RECEIVER: PAIRING AND CONNECTING THE RECEIVER TO A DEVICE

Supported platforms

You can connect a **LOKUS GNSS receiver** to a handheld device or tablet powered by the following operating systems:

- Android
- iOS
- Windows

Note: On Android devices, you must enable Mock Locations to allow the Android device to use GNSS positions from

If you have an existing application which supports Bluetooth GPS, then LOKUS will work seamlessly with that application after pairing.

If your existing application does not support Bluetooth GPS then we will share a file. After Integrating that file with your application you will get the Bluetooth GPS support.

Please follow the set up guide below for integration:

LOKUS Bluetooth GPS Setup

- **DOES YOUR APPLICATION HAVE EXTERNAL/ BLUETOOTH GPS SUPPORT? IF YES, JUST TAKE THE DEVICE AND IT IS READY TO USE WITH YOUR APPLICATION.**
- **Else you are just a few steps away from setting up T55 with your application (3 Tabs: Android, QT/ Dot Net, IOS) (lets start with android)**
  - **Step 1: Create a background service**
    - BluetoothGPSService.java (Service to parse Bluetooth Data)

```
package com.stesalitsystems.sxbluetooth.bluetooth;

//Created by STESALIT SYSTEMS LIMITED

import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Build;
import android.os.Bundle;
import android.os.IBinder;
import android.os.SystemClock;
import android.support.annotation.Nullable;
import android.support.v4.content.LocalBroadcastManager;
import android.text.TextUtils;
import android.util.Log;
```

```
import com.stesalitsystems.sxbluetooth.R;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.TimeZone;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;
import java.util.regex.Matcher;

import java.util.regex.Pattern;
/**
Created by Krishnendu Dasgupta on 01-04-2019.
*/
public class BluetoothGPSService extends Service {
private BluetoothAdapter btAdapter = null;
private BluetoothSocket btSocket = null;
private static final String LOG_TAG = "T55";
private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

// MAC-address of Bluetooth module
public String newAddress = null;
SharedPreferences pref;
private Timer timer;
private TimerTask timerTask;
InputStream tmpIn = null;
boolean enabled;
boolean ready=true;
TextUtils.SimpleStringSplitter splitter;
public static final String GPS_MOCK_PROVIDER = "gps";
String fixTime="";
private float precision = 10f;
boolean read_pubx=false;
@Nullable
@Override
public IBinder onBind(Intent intent) {
return null;
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
pref = getApplicationContext().getSharedPreferences(getResources().getString(R.string.pref_key), 0);
String service_status = pref.getString("address", "");
newAddress = pref.getString("address", "");
btAdapter = BluetoothAdapter.getDefaultAdapter();
BluetoothDevice device = btAdapter.getRemoteDevice(newAddress);

//Attempt to create a bluetooth socket for comms
try {
btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);

} catch (IOException e1) {
e1.printStackTrace();
}
// Establish the connection.
try {
btSocket.connect();
```

```
enabled=true;
} catch (IOException e) {
e.printStackTrace();
try {
enabled=false;
btSocket.close();//If IO exception occurs attempt to close socket
} catch (IOException e2) {
enabled=false;
e2.printStackTrace();
}
}
if(enabled){
startTimer();
}
return START_STICKY;
}
public void startTimer() {
//set a new Timer
timer = new Timer();
//initialize the TimerTask's job
initializeTimerTask();
//schedule the timer, to wake up every 1 second
timer.schedule(timerTask, 2000, 2000); //
}

public void initializeTimerTask() {
timerTask = new TimerTask() {
public void run() {
try {
if (pref.getBoolean("iserviceStarted", true)) {
//Service Should Start
tmpIn=btSocket.getInputStream();
try {
BufferedReader reader = new BufferedReader(new InputStreamReader(tmpIn,"US-ASCII"));
String s;
long now = SystemClock.uptimeMillis();
long lastRead = now;
while((enabled) && (now < lastRead+5000 )){
if (reader.ready()){
s = reader.readLine();
Log.v(LOG_TAG, "data: "+ System.currentTimeMillis()+" "+s);
parseNmeaSentence1(s+"\r\n");
if(read_pubx){
Location loc=parsePUBX(s+"\r\n");
if(loc==null){
sendMessageToActivity(loc,"false",getApplicationContext());
}else {
sendMessageToActivity(loc,"true",getApplicationContext());
}
}
```

```
ready = true;
lastRead = SystemClock.uptimeMillis();
} else {
Log.d(LOG_TAG, "data: not ready "+ System.currentTimeMillis());
SystemClock.sleep(500);
}
now = SystemClock.uptimeMillis();
}
} catch (IOException e) {
e.printStackTrace();
Log.e(LOG_TAG, "error while getting data", e);
} finally {
// cleanly closing everything...
}
}
}catch (Exception e){
//Some exception
e.printStackTrace();
}
}
};
}
private static void sendMessageToActivity(Location l, String msg, Context context) {
Intent intent = new Intent("GPSLocationUpdates");
// You can also include some extra data.
intent.putExtra("Status", msg);
Bundle b = new Bundle();
b.putParcelable("Location", l);
intent.putExtra("Location", b);
LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
}
public Location parsePUBX(String pubx){
String nmeaSentence = null;
Pattern xx = Pattern.compile("\\$([^*$]*)\\*([0-9A-F][0-9A-F])?\r\n");
Matcher m = xx.matcher(pubx);

if (m.matches()) {
nmeaSentence = m.group(0);
String sentence = m.group(1);
String checkSum = m.group(2);
Log.v(LOG_TAG, "data: " + System.currentTimeMillis() + " " + sentence + " cheksum; " + checkSum + "
control: " + String.format("%X", computeChecksum(sentence)));
splitter = new TextUtils.SimpleStringSplitter(',');
splitter.setString(sentence);
String command = splitter.next();
if (command.equals("PUBX")){
String msgId=splitter.next();
String strTime=splitter.next();
String lat=splitter.next();  //ddmm.mmmmm
String ns=splitter.next();
String lon=splitter.next();  //ddmm.mmmmm
```

```
String ew=splitter.next();
String altRef=splitter.next();
String sat=splitter.next();
if(sat=="NF")
sat="No Fix";
if(sat=="DR")
sat="Dead reckoning only solution";
if(sat=="G2")
sat="Stand alone 2D solution";
if(sat=="G3")
sat="Stand alone 3D solution";
if(sat=="D2")
sat="Differential 2D solution";
if(sat=="D3")
sat="Differential 3D solution";
if(sat=="RK")
sat="Combined GPS + dead reckoning solution";
if(sat=="TT")
sat="Time only solution";

String hAcc=splitter.next(); //Horizontal accuracy estimate in meter
String vAcc=splitter.next();    //Vertical accuracy estimat in meter
String SOG=splitter.next();     //Speed over ground(Km/Hr)
String COG=splitter.next();     //Course over ground in degree
String vVel=splitter.next(); //Vertical velocity (positive downwards)
String diffAge=splitter.next(); //Age of differential corrections (blank when DGPS is not used)
String hdop=splitter.next();
String vdop=splitter.next();
String tdop=splitter.next();    //TDOP, Time Dilution of Precision
String numSvs=splitter.next();  //Number of satellites used in the navigation solution
String reserved=splitter.next();    //Reserved, always set to 0
String dr=splitter.next();  //DR used
Location fix = new Location(GPS_MOCK_PROVIDER);
if (! strTime.equals(fixTime)){
fix = new Location(GPS_MOCK_PROVIDER);
fixTime = strTime;
long fixTimestamp = parseNmeaTime(strTime);
fix.setTime(fixTimestamp);
}
if (lat != null && !lat.equals("")){
fix.setLatitude(parseNmeaLatitude(lat,ns));
}
if (lon != null && !lon.equals("")){
fix.setLongitude(parseNmeaLongitude(lon,ew));
}
if (hAcc != null && !hAcc.equals("")){
fix.setAccuracy(Float.parseFloat(hAcc));
}
if (altRef != null && !altRef.equals("")){
fix.setAltitude(Double.parseDouble(altRef));
```

```
}
if (SOG != null && !SOG.equals("")){
fix.setSpeed(Float.parseFloat(SOG));
}
long fixTimestamp = parseNmeaTime(strTime);
fix.setTime(fixTimestamp);
fix.setElapsedRealtimeNanos(fixTimestamp);
return fix;
}
}
return null;
}
public Location parseNmeaSentence1(String gpsSentence){
String nmeaSentence = null;
Pattern xx = Pattern.compile("\\$([^*$]*)\\*([0-9A-F][0-9A-F])?\r\n");
Matcher m = xx.matcher(gpsSentence);

if (m.matches()){
nmeaSentence = m.group(0);
String sentence = m.group(1);
String checkSum = m.group(2);
Log.v(LOG_TAG, "data: "+ System.currentTimeMillis()+" "+sentence+" cheksum; "+checkSum +" control:
"+ String.format("%X",computeChecksum(sentence)));
splitter = new TextUtils.SimpleStringSplitter(',');
splitter.setString(sentence);
String command = splitter.next();
if (command.equals("GNGGA")){
/* $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
GGA        Global Positioning System Fix Data
123519       Fix taken at 12:35:19 UTC
4807.038,N   Latitude 48 deg 07.038' N
01131.000,E  Longitude 11 deg 31.000' E
Fix quality: 0 = invalid
1 = GPS fix (SPS)
= DGPS fix
= PPS fix
= Real Time Kinematic
= Float RTK
= estimated (dead reckoning) (2.3 feature)
= Manual input mode
= Simulation mode
08         Number of satellites being tracked
0.9        Horizontal dilution of position
545.4,M      Altitude, Meters, above mean sea level
46.9,M       Height of geoid (mean sea level) above WGS84
ellipsoid
(empty field) time in seconds since last DGPS update
(empty field) DGPS station ID number
*47         the checksum data, always begins with *
```

```
*/
// UTC time of fix HHmmss.S
String time = splitter.next();
// latitude ddmm.M
String lat = splitter.next();
// direction (N/S)
String latDir = splitter.next();
// longitude dddmm.M
String lon = splitter.next();
// direction (E/W)
String lonDir = splitter.next();
/* fix quality:
0= invalid
1 = GPS fix (SPS)
2 = DGPS fix
3 = PPS fix
4 = Real Time Kinematic
5 = Float RTK
6 = estimated (dead reckoning) (2.3 feature)
7 = Manual input mode
= Simulation mode
*/
String quality = splitter.next();
// Number of satellites being tracked
String nbSat = splitter.next();
// Horizontal dilution of position (float)
String hdop = splitter.next();
// Altitude, Meters, above mean sea level
String alt = splitter.next();
// Height of geoid (mean sea level) above WGS84 ellipsoid
String geoAlt = splitter.next();
// time in seconds since last DGPS update
// DGPS station ID number
if (quality != null && !quality.equals("") && !quality.equals("0") ){
Location fix = new Location(GPS_MOCK_PROVIDER);
if (! time.equals(fixTime)){
fixTime = time;
long fixTimestamp = parseNmeaTime(time);
fix.setTime(fixTimestamp);
Log.v(LOG_TAG, "Fix: "+fix);
}
if (lat != null && !lat.equals("")){
fix.setLatitude(parseNmeaLatitude(lat,latDir));
}
if (lon != null && !lon.equals("")){
fix.setLongitude(parseNmeaLongitude(lon,lonDir));
}
if (hdop != null && !hdop.equals("")){
fix.setAccuracy(Float.parseFloat(hdop)*precision);
}
if (alt != null && !alt.equals("")){
```

```
fix.setAltitude(Double.parseDouble(alt));
}
long fixTimestamp = parseNmeaTime(time);
fix.setTime(fixTimestamp);
if (nbSat != null && !nbSat.equals("")){
Bundle extras = new Bundle();
extras.putInt("satellites", Integer.parseInt(nbSat));
fix.setExtras(extras);
}
fix.setElapsedRealtimeNanos(fixTimestamp);

return fix;
}
else if(quality.equals("0")){
return null;
}
} else if (command.equals("GNRMC")){

/* $GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

Where:
RMC         Recommended Minimum sentence C
123519      Fix taken at 12:35:19 UTC
A           Status A=active or V=Void.
4807.038,N   Latitude 48 deg 07.038' N
01131.000,E  Longitude 11 deg 31.000' E
022.4       Speed over the ground in knots
084.4       Track angle in degrees True
230394      Date - 23rd of March 1994
003.1,W     Magnetic Variation
*6A         The checksum data, always begins with *
*/
// UTC time of fix HHmmss.S
String time = splitter.next();
// fix status (A/V)
String status = splitter.next();
// latitude ddmm.M
String lat = splitter.next();
// direction (N/S)
String latDir = splitter.next();
// longitude dddmm.M
String lon = splitter.next();
// direction (E/W)
String lonDir = splitter.next();
// Speed over the ground in knots
String speed = splitter.next();
// Track angle in degrees True
String bearing = splitter.next();
// UTC date of fix DDMMYY
String date = splitter.next();
// Magnetic Variation ddd.D
```

```
String magn = splitter.next();
// Magnetic variation direction (E/W)
String magnDir = splitter.next();
// for NMEA 0183 version 3.00 active the Mode indicator field is added
// Mode indicator, (A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator )
if (status != null && !status.equals("") && status.equals("A") ){
read_pubx=true;
Location fix=null;
if (! time.equals(fixTime)){
fix = new Location(GPS_MOCK_PROVIDER);
fixTime = time;
long fixTimestamp = parseNmeaTime(time);
fix.setTime(fixTimestamp);
}
if (lat != null && !lat.equals("")){
fix.setLatitude(parseNmeaLatitude(lat,latDir));
}
if (lon != null && !lon.equals("")){
fix.setLongitude(parseNmeaLongitude(lon,lonDir));
}
if (speed != null && !speed.equals("")){
fix.setSpeed(parseNmeaSpeed(speed, "N"));
}
if (bearing != null && !bearing.equals("")){
fix.setBearing(Float.parseFloat(bearing));
}
fix.setAccuracy(5f);
long fixTimestamp = parseNmeaTime(time);
fix.setElapsedRealtimeNanos(fixTimestamp);

return fix;
} else if(status.equals("V")){
read_pubx=false;
return null;
}
} else if (command.equals("GPGSA")){


/*  $GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39

Where:
GSA      Satellite status
A        Auto selection of 2D or 3D fix (M = manual)
3        3D fix - values include: 1 = no fix
2 = 2D fix
3 = 3D fix
04,05... PRNs of satellites used for fix (space for 12)
2.5      PDOP (Position dilution of precision)
1.3      Horizontal dilution of precision (HDOP)
2.1      Vertical dilution of precision (VDOP)
```

```
 *39     the checksum data, always begins with *
*/
// mode : A Auto selection of 2D or 3D fix / M = manual
String mode = splitter.next();
// fix type  : 1 - no fix / 2 - 2D / 3 - 3D
String fixType = splitter.next();
// discard PRNs of satellites used for fix (space for 12)
for (int i=0 ; ((i<12)&&(! "1".equals(fixType)))  ; i++){
splitter.next();
}
// Position dilution of precision (float)
String pdop = splitter.next();
// Horizontal dilution of precision (float)
String hdop = splitter.next();
// Vertical dilution of precision (float)
String vdop = splitter.next();
} else if (command.equals("GPVTG")){


/*  $GPVTG,054.7,T,034.4,M,005.5,N,010.2,K*48

where:
VTG         Track made good and ground speed
054.7,T     True track made good (degrees)
034.4,M     Magnetic track made good
005.5,N     Ground speed, knots
010.2,K     Ground speed, Kilometers per hour
*48         Checksum
*/
// Track angle in degrees True
String bearing = splitter.next();
// T
splitter.next();
// Magnetic track made good
String magn = splitter.next();
// M
splitter.next();
// Speed over the ground in knots
String speedKnots = splitter.next();
// N
splitter.next();
// Speed over the ground in Kilometers per hour
String speedKm = splitter.next();
// K
splitter.next();
// for NMEA 0183 version 3.00 active the Mode indicator field is added
// Mode indicator, (A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator )
} else if (command.equals("GPGLL")){
/*  $GPGLL,4916.45,N,12311.12,W,225444,A,*1D
```

```
Where:
GLL        Geographic position, Latitude and Longitude
4916.46,N    Latitude 49 deg. 16.45 min. North
12311.12,W   Longitude 123 deg. 11.12 min. West
225444       Fix taken at 22:54:44 UTC
A          Data Active or V (void)
*iD         checksum data
*/
// latitude ddmm.M
String lat = splitter.next();
// direction (N/S)
String latDir = splitter.next();
// longitude dddmm.M
String lon = splitter.next();
// direction (E/W)
String lonDir = splitter.next();
// UTC time of fix HHmmss.S
String time = splitter.next();
// fix status (A/V)
String status = splitter.next();
// for NMEA 0183 version 3.00 active the Mode indicator field is added
// Mode indicator, (A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator )
}else if (command.equals("GPGSV")){

}
}else{

}
return null;
}
public double parseNmeaLatitude(String lat, String orientation){
double latitude = 0.0;
if (lat != null && orientation != null && !lat.equals("") && !orientation.equals("")){
double temp1 = Double.parseDouble(lat);
double temp2 = Math.floor(temp1/100);
double temp3 = (temp1/100 - temp2)/0.6;
if (orientation.equals("S")){
latitude = -(temp2+temp3);
} else if (orientation.equals("N")){
latitude = (temp2+temp3);
}
Log.e("latitude=",latitude+"");
}
return latitude;
}
public double parseNmeaLongitude(String lon, String orientation){
double longitude = 0.0;
if (lon != null && orientation != null && !lon.equals("") && !orientation.equals("")){
double temp1 = Double.parseDouble(lon);
double temp2 = Math.floor(temp1/100);
```

```
double temp3 = (temp1/100 - temp2)/0.6;
if (orientation.equals("W")){
longitude = -(temp2+temp3);
} else if (orientation.equals("E")){
longitude = (temp2+temp3);
}
Log.e("longitude=",longitude+"");
}
return longitude;
}
public float parseNmeaSpeed(String speed, String metric){
float meterSpeed = 0.0f;
if (speed != null && metric != null && !speed.equals("") && !metric.equals("")){
float temp1 = Float.parseFloat(speed)/3.6f;
if (metric.equals("K")){
meterSpeed = temp1;
} else if (metric.equals("N")){
meterSpeed = temp1*1.852f;
}
}
return meterSpeed;
}
public long parseNmeaTime(String time){
long timestamp = 0;
SimpleDateFormat fmt = new SimpleDateFormat("HHmmss.SSS");
fmt.setTimeZone(TimeZone.getTimeZone("GMT"));
try {
if (time != null && time != null){
long now = System.currentTimeMillis();
long today = now - (now %86400000L);
long temp1;
// sometime we don't have millisecond in the time string, so we have to reformat it
temp1 = fmt.parse(String.format((Locale)null,"%010.3f", Double.parseDouble(time))).getTime();
long temp2 = today+temp1;
// if we're around midnight we could have a problem...
if (temp2 - now > 43200000L) {
timestamp  = temp2 - 86400000L;
} else if (now - temp2 > 43200000L){
timestamp  = temp2 + 86400000L;
} else {
timestamp  = temp2;
}
}
} catch (ParseException e) {
Log.e(LOG_TAG, "Error while parsing NMEA time", e);
}
return timestamp;
}
public byte computeChecksum(String s){
byte checksum = 0;

for (char c : s.toCharArray()){
checksum ^= (byte)c;
}
return checksum;
}
}
```

- **Step 2: Create a background service**
  - DeviceListActivity.java (To list paired devices and connect)

```
package com.stesalitsystems.sxbluetooth.bluetooth;

/**
 * Created by STESALIT SYSTEMS LIMITED
 */

import android.app.AlertDialog;
import android.app.AppOpsManager;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.stesalitsystems.sxbluetooth.MainActivity;
import com.stesalitsystems.sxbluetooth.R;

import java.io.IOException;
import java.util.List;
import java.util.Set;
import java.util.UUID;

public class DeviceListActivity extends AppCompatActivity {

    // textview for connection status
    TextView textConnectionStatus;
    ListView pairedListView;
```

```
//An EXTRA to take the device MAC to the next activity
public static String EXTRA_DEVICE_ADDRESS;
private TextView tv_toolbar;
// Member fields
private BluetoothAdapter mBtAdapter;
private ArrayAdapter<String> mPairedDevicesArrayAdapter;
SharedPreferences pref;
private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
@Override
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.layout_device_list);
   pref = this.getSharedPreferences(getResources().getString(R.string.pref_key), 0);
   boolean loginStatus=pref.getBoolean("isOn",false);
   textConnectionStatus = (TextView) findViewById(R.id.connecting);
   textConnectionStatus.setTextSize(40);
   FloatingActionButton btn_refresh=(FloatingActionButton)findViewById(R.id.btn_refresh);
   btn_refresh.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         Intent i = new Intent(DeviceListActivity.this, DeviceListActivity.class);
         startActivity(i);
         finish();
      }
   });
   // Initialize array adapter for paired devices
   mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.row_device);

   // Find and set up the ListView for paired devices
   pairedListView = (ListView) findViewById(R.id.paired_devices);
   pairedListView.setOnItemClickListener(mDeviceClickListener);
   pairedListView.setAdapter(mPairedDevicesArrayAdapter);
   Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
   tv_toolbar=(TextView)findViewById(R.id.tv_toolbar);
   tv_toolbar.setText("Select Device");
   tv_toolbar.setTextSize(15);
   toolbar.setBackgroundColor(Color.parseColor("#0079bf"));
   setSupportActionBar(toolbar);
   getSupportActionBar().setDisplayShowTitleEnabled(true);
   getSupportActionBar().setDisplayHomeAsUpEnabled(true);
   toolbar.setNavigationOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
```

```
            //do something you want
            onBackPressed();
        }
    });

}
public void launchHomeScreen(){
    SharedPreferences pref = this.getSharedPreferences(getResources().getString(R.string.pref_key), 0);
    String service_status=pref.getString("address","");
    Intent i = new Intent(DeviceListActivity.this, MainActivity.class);
    i.putExtra("address",service_status);
    startActivity(i);
    finish();
}
@Override
public void onResume()
{
    super.onResume();
    //It is best to check BT status at onResume in case something has changed while app was paused etc
    checkBTState();

    mPairedDevicesArrayAdapter.clear();// clears the array so items aren't duplicated when resuming
from onPause

    textConnectionStatus.setText(" "); //makes the textview blank

    // Get the local Bluetooth adapter
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

    // Get a set of currently paired devices and append to pairedDevices list
    Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

    // Add previously paired devices to the array
    if (pairedDevices.size() > 0) {
        findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);//make title viewable
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        mPairedDevicesArrayAdapter.add("no devices paired");
    }
}

//method to check if the device has Bluetooth and if it is on.
//Prompts the user to turn it on if it is off
private void checkBTState()
{
```

```
    // Check device has Bluetooth and that it is turned on
    mBtAdapter= BluetoothAdapter.getDefaultAdapter(); // CHECK THIS OUT THAT IT WORKS!!!
    if(mBtAdapter==null) {
      Toast.makeText(getBaseContext(), "Device does not support Bluetooth",
Toast.LENGTH_SHORT).show();
      finish();
    } else {
      if (!mBtAdapter.isEnabled()) {
        //Prompt user to turn on Bluetooth
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, 1);
      }
    }
  }
  // Set up on-click listener for the listview
  private OnItemClickListener mDeviceClickListener = new OnItemClickListener()
  {
    public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3)
    {
        textConnectionStatus.setText("Connecting...");
        // Get the device MAC address, which is the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);
        SharedPreferences.Editor editor = pref.edit();
        editor.putString("address",address);//Save the address for future reference
        editor.putBoolean("isOn",true);
        editor.commit();
        BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
        BluetoothDevice device = btAdapter.getRemoteDevice(address);
        boolean enabledBt=false;
        //Attempt to create a bluetooth socket for comms
        try {
          BluetoothSocket btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
          if(btSocket.isConnected()){
            try {
              btSocket.close();//If IO exception occurs attempt to close socket
            }catch (IOException e2) {
              e2.printStackTrace();
            }
            if(btSocket.isConnected()){
              Toast.makeText(getApplicationContext(),"Bluetooth already connected!! If required please
restart phone or device bluetooth!!",Toast.LENGTH_LONG).show();
              enabledBt=false;
              textConnectionStatus.setText("Connected.....");
            }else {
              try{
                btSocket.connect();
                try {
```

```
                        btSocket.close();//If IO exception occurs attempt to close socket
                        enabledBt=true;
                      } catch (IOException e2) {
                        e2.printStackTrace();
                      }
                    }catch (IOException e) {
                      enabledBt=false;
                      Toast.makeText(getApplicationContext(),"Cannot Connect to device!! Please check if it is
On!! Or restart the device",Toast.LENGTH_LONG).show();
                      textConnectionStatus.setText("");
                    }
                  }
                }else {
                  try{
                    btSocket.connect();
                    btSocket.close();//If IO exception occurs attempt to close socket
                    enabledBt=true;
                  }catch (IOException e) {
                    e.printStackTrace();
                    try {
                      btSocket.close();//If IO exception occurs attempt to close socket
                      btSocket.connect();
                      btSocket.close();//If IO exception occurs attempt to close socket
                      enabledBt=true;
                    } catch (IOException e2) {
                      enabledBt=false;
                      Toast.makeText(getApplicationContext(),"Cannot Connect to device!! Please check if it is
On!! Or restart the device",Toast.LENGTH_LONG).show();
                      e2.printStackTrace();
                      textConnectionStatus.setText("");
                    }
                  }
                }
              } catch (IOException e1) {
                enabledBt=false;
                Toast.makeText(getApplicationContext(),"Cannot Connect to device!! Please check if it is On!!
Or restart the device",Toast.LENGTH_LONG).show();
                e1.printStackTrace();
                textConnectionStatus.setText("");
              }
              if(enabledBt){
                startService(new Intent(DeviceListActivity.this, BluetoothGPSService.class));
              }
            }
          };
          @Override
          public void onBackPressed(){
            AlertDialog.Builder alertBuilder = new AlertDialog.Builder(DeviceListActivity.this);
            alertBuilder.setTitle("Confirm!!");
            alertBuilder.setIcon(android.R.drawable.ic_lock_power_off);
            alertBuilder.setMessage("Are you sure you want to exit?");
            alertBuilder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {
                finish();
              }
            });
            alertBuilder.setNegativeButton("No", new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {

              }
            });
            alertBuilder.setCancelable(false);
            AlertDialog dialog = alertBuilder.create();
            dialog.getWindow().getAttributes().windowAnimations = R.style.DialogTheme;
            dialog.show();
          }
        }
```

❒ layout_device_list.xml (Design for the listing activity)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="#fff"
  >
  <include
    android:id="@+id/tootlbar_head"
    layout="@layout/toolbar"/>
  <TextView android:id="@+id/title_paired_devices"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/tootlbar_head"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Select Your Bluetooth GPS from paired devices:"
    android:visibility="gone"
    android:textSize="15dp"
    android:textColor="#0079bf"
    android:layout_margin="5dp"
    />
  <ListView android:id="@+id/paired_devices"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/title_paired_devices"
    android:stackFromBottom="false"
    android:layout_weight="1"
    />

  <TextView
    android:id="@+id/connecting"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20dp"
    android:layout_below="@+id/paired_devices"
    android:textColor="#0079bf"/>

  <TextView
    android:id="@+id/infoText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:layout_alignParentBottom="true"
    android:textStyle="bold"
    android:textSize="15dp"
    android:text="If no devices are listed please pair your device in Android settings"
    android:textColor="#0079bf" />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center">

  </LinearLayout>
  <android.support.design.widget.FloatingActionButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@android:drawable/ic_popup_sync"
    android:layout_above="@+id/infoText"
    android:layout_marginBottom="5dp"
    android:id="@+id/btn_refresh"
    android:layout_alignParentRight="true"/>
</RelativeLayout>
```

- **Step 3: In the Activity to view the location information add the Broadcast Receiver to get the location data from Bluetooth GPS**
  - In Activity to show/ use location data add the following lines

```
private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent intent) {
         // Get extra data included in the Intent
         String message = intent.getStringExtra("Status");
         if(message.equalsIgnoreCase("true")){
            Bundle b = intent.getBundleExtra("Location");
            Location lastKnownLoc = (Location) b.getParcelable("Location");
            if (lastKnownLoc != null) {
               tvLatitude.setText(String.valueOf(lastKnownLoc.getLatitude()));
               tvLongitude.setText(String.valueOf(lastKnownLoc.getLongitude()));
               tvAccuracy.setText(String.valueOf(lastKnownLoc.getAccuracy()));
               tvTimestamp.setText((new Date(lastKnownLoc.getTime()).toString()));
               tvProvider.setText(String.valueOf(lastKnownLoc.getAltitude()));
            }
         }
         tvStatus.setText(message);

         // Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
      }
   };

** Also initialise the Broadcast receiver in onCreate():
LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver, new
IntentFilter("GPSLocationUpdates"));
o           Step 4: Finally add the service definition in Android Manifest
<service
android:name=".service.BluetoothGPSService"
          android:enabled="true" >
 </service>
```