How to read Location from Bluetooth GPS

One of the most sold accessories for mobile phones and PDAs are Bluetooth GPS (BT-GPS). These simple devices connect to the GPS satellite system and allows to pinpoint your position with 5 meters precision. It's possible to access this information trough a JavaME application you just need to have Bluetooth phone with the Java APIs available on it.

JSR 179 - Location API

In devices where JSR 179 - Location API is implemented, there's no need to directly use JSR 82 (Bluetooth API) to read location data, since the Location API does all the necessary work for retrieval of the location data itself:

- Searching the GPS module in the BT local area
- Pairing with GPS device
- Retrieval of positioning data
- Parsing of low-level NMEA sentences, turning them into high-level Java objects.

To learn how to use GPS location with JSR 179, download the MIDP: Location API Developer's Guide v2.0 document, which also contains source code and a ready-to-test application.

Click here to perform a search for devices supporting JSR 179 - Location API.

JSR 82 - Bluetooth API

If your device does not implement JSR 179, you'll still be able to retrieve location data from a Bluetooth GPS module, but you'll have to do it manually. For that, you need to perform the following tasks:

To read location information from a bluetooth GPS, we need to implement the following tasks:

1. Search for the Bluetooth GPS device
2. Connect to the GPS device
3. Read and Parse NMEA sentences

For the first step we need to search for a Bluetooth device that implements the RFCOMM service, I include in the sample code a class, "BTManager.class", that does all this work. If you want to have more information about this, see this How to search for Bluetooth devices and services.

For the purpose of reading data from the BT-GPS lets create an GpsBt class. The first thing to do is to create some variable to store the URL address for your BT-GPS.

```
// current bluetooth device
public String btUrl = "";
public String btName = "";

public void setDevice(String btUrl, String btName) {
  this.btUrl = btUrl;
  this.btName = btName;
}
```

Now we need to connect to the device and start reading the data.

```
  public void start() {
  if (isActive) {
    stop();
  }
  connect();
  if (isConnected) {
    isActive = true;
    Thread t = new Thread(this);
    t.start();
  }
}

public void connect() {
  if (btUrl == null || (btUrl.trim().compareTo("") == 0)) {
    isConnected = false;
    return;
  }
  try {
    conn = (StreamConnection) Connector.open(btUrl, Connector.READ_WRITE);
    in = new DataInputStream(conn.openInputStream());
    isConnected = true;
    mode = 0;
  } catch (IOException e) {
    close();
  }
}

public void run() {
  isActive = true;
  while (isActive) {
      // check if connection is still open
      if (!isConnected && isActive) {
        // connect to gps device
        connect();
```


(C) Copyright Nokia 2013. All rights reserved.

```
        } else {
          // read NMEA Strings
          readNMEASentences();
        }
    }
    close();
    isActive = false;
  }
```

As you may have noticed I'm implementing the reading loop using a thread. The reason for this, is that the BT-GPS is always sending data so you need to keep read from it before the connection buffer overflows. The data the BT-GPS sends are NMEA sentences, these sentences gives several information about the GPS status. The one we are searching for is the GPGGA sentence that gives essential fix data and provides 3D location and accuracy data.

```
  public void readNMEASentences() {
     try {
       if (!isConnected) {
         return;
       }
       // check characters available
       int size = in.available();
       if (size <= 0) {
         return;
       }
       // read data
       for (int j = 0; j < size; j++) {
         int i = in.read();
         if (i != -1) {
           char l = (char) i;
           switch (mode) {
           case (STATE_SEARCH_SENTENCE_BEGIN): {
             // search for the sentence begin
             if (l == '$') {
               // found begin of sentence
               mode = 1;
               sb.setLength(0);
             }
           }
             break;
           case (STATE_READ_DATA_TYPE): {
             // check what kind of sentence we have
             sb.append(l);
             if (sb.length() == 6) {
               if (sb.toString().startsWith("GPGGA")) {
                 mode = STATE_READ_SENTENCE;
                 sb.setLength(0);
               } else {
                 mode = STATE_SEARCH_SENTENCE_BEGIN;
                 sb.setLength(0);
               }
             }
           }
             break;
           case (STATE_READ_SENTENCE): {
             // read data from sentence
             sb.append(l);
             if ((l == 13) || (l == 10) || (l == '$')) {
               mode = STATE_SEARCH_SENTENCE_BEGIN;
               currentInfo = new String(sb.toString());
             }
           }
             break;
           }

         } else {
           close();
         }
       }

     } catch (Exception e) {
       close();
     }
  }
```

After we have the correct sentence we just need to parse the information and retrieve it to the user

```
  public Location getLocation() {
     Location location = new Location();
     if (isConnected && isActive && currentInfo != null) {
         location.parseGPGGA(currentInfo);
     }
     return location;
  }
```

To help us to retrieve data we use a class, called Location, to parse the GGA sentence and wrap the position data. We are also using a class called StringTokenizer to split the tokens in the sentence.

```
 public class Location {

   // NMEA GPGGA Elements
   String utc;
   String latitude;
   String northHemi;
   String longitude;
   String eastHemi;
   String altitude;
   int quality;
   int nSat;
   String horDilution;
   String altitudeUnit;
   String geoidalHeight;
   String geoidalHeightUnit;
   String diffCorrection;
   String diffStationId;

   /**
    * Method that parses a NMEA string and returns Location. For more info check
    * this page: http://www.gpsinformation.org/dale/nmea.htm#GGA
    *
    * @param value -
    *           string that represent NMEA GGA string
```


(C) Copyright Nokia 2013. All rights reserved.

```
    */
    public void parseGPGGA(String value) {
      // Helper class to parse strings
      StringTokenizer tok = new StringTokenizer(value, ",");

      utc = tok.nextToken();
      latitude = tok.nextToken();
      northHemi = tok.nextToken();
      longitude = tok.nextToken();
      eastHemi = tok.nextToken();
      quality = Integer.parseInt(tok.nextToken());
      nSat = Integer.parseInt(tok.nextToken());
      horDilution = tok.nextToken();
      altitude = tok.nextToken();
      altitudeUnit = tok.nextToken();
      geoidalHeight = tok.nextToken();
      geoidalHeightUnit = tok.nextToken();
      diffCorrection = tok.nextToken();
      diffStationId = tok.nextToken();
    }
  }
```

And that's a wrap we now have a class capable of reading the NMEA sentences from the BT-GPS and retrieve location info. Check the full source code for an complete application that allows you to search for devices, connect to them and shows the location info on the screen.

Downloads

- Full Source Code
- Jad File
- Jar File

References

- Vietnamese GPS - Intelligent on the go
- NMEA Reference

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.