Usando GPS Bluetooth

Como Usar um GPS Bluetooth

A classe **GPS** mostra como usar um dispositivo *GPS Bluetooth* e ler informações de localização deste. Ela mostra um dialogo de seleção para o usuário e inicia o recebimento de dados do dispositivo selecionado. Atualmente, ela apenas processa a mensagem RMC que é requerida em todos os dispositivos GPS. Isto dá a informação de latitude, longitude, direção e velocidade mas não altitude nem diluição.

Você precisa implementar mensagens para perda de signal, etc, se você deseja mostrá-las ao usuário.

Você também pode usar o método *Parse()* para converter os valores recebidos para graus, minutos e segundos, assim como um valor inteiro combinado multiplicado por 65536.

Este exemplo assume que o dispositivo GPS usa o canal RFCOMM 1 como canal de dados. Este tem sido o caso de todos os dispositivos que eu usei, então é uma suposição segura.

GPS.h

```
#ifndef GPS_H_
#define GPS_H_

#include <e32std.h>
#include <btmanclient.h>
#include <btextnotifiers.h>
#include <es_sock.h>
#include <in_sock.h>
#include <bt_sock.h>

class GPS : public CActive
{
public:
 GPS();
 virtual ~GPS();
 void ConstructL();

 void DoCancel();
 void RunL();

 TPtrC Lat();
 TPtrC Lon();

 TInt   speed, heading;

private:
 TBool Active;
 TBuf8<32> lat, lon;

 TBuf8<256> line;
 TBuf8<32> data;
 int state;
 RSocket iSendingSocket;
 RSocketServ iSocketServer;
};

#endif /*GPS_H_*/
```


(C) Copyright Nokia 2014. All rights reserved.

GPS.cpp

```
#include "GPS.h"
#include <e32math.h>

_LIT8(ZERO, "0");
_LIT(RFCOMM, "RFCOMM");
_LIT8(KGPS, "GPS");

void Parse(TDesC8 &data, TInt &deg, TInt &min, TInt &sec, TInt32 &whole)
{
 TLex8 lex(data);

 lex.Val(deg);
 lex.Inc();
 lex.Val(min);
 sec = min * 60;
 int l = (data.Length() - data.Find(_L8(".")));
 int x = 1;
 while (l-- > 0)
  x *= 10;
 sec /= x/100;

 min = deg % 100;
 deg /= 100;

 whole = deg * 65536 + (min * 65536 / 60) + (sec * 65536 / 36000);
}

GPS::GPS() : CActive(EPriorityLow)
{
 state = 0;
}

GPS::~GPS()
{
 TRAPD(err, Cancel());
 TRAP(err, Deque());
}

void GPS::RunL()
{
 int i, j;

 if (iStatus == KErrNone)
  {
  if (state == 1)
  {
   state = 2;
   line.Zero();
   iSendingSocket.Read(data, iStatus);
   SetActive();
  }
  else if (state == 2)
  {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    int rpos = 0;

    while ((rpos = data.Locate('\r')) != KErrNotFound)
    {
     line.Append(data.Left(rpos));
     if (data.Length() > rpos + 2)
     {
      if (data[rpos + 1] == '\n')
       data.Copy(data.Mid(rpos+2));
      else
       data.Copy(data.Mid(rpos+1));
     }
     else
     {
      data.Zero();
     }

     if (line.Length() > 10)
     {
      // RMC - lat, lon, speed
      if ((line[3] == 'R') && (line[4] == 'M') && (line[5] == 'C'))
      {
       j = i = 0;
       while ((i < line.Length()) && (line[i] != ','))
        i++;
       i++;

       j = i;
       while ((i < line.Length()) && (line[i] != ','))
        i++;
       i++;

       // If there is no signal, this is not A
       if (line[i] != 'A')
       {
        lat.Copy(ZERO);
        lon.Copy(ZERO);
        latdeg = londeg = latmin = lonmin = latsec = lonsec = 0;
        line.Zero();
        iSendingSocket.Read(data, iStatus);
        SetActive();
        return;
       }

       while ((i < line.Length()) && (line[i] != ','))
        i++;
       i++;
       j = i;
       while ((i < line.Length()) && (line[i] != ','))
        i++;

       // Copy latitude from the message
       lat.Copy(line.Mid(j, i-j));
       // If it's southern hemisphere, negate the value
       if (line[i+1] == 'S')
        lat.Insert(0, _L8("-"));
```


(C) Copyright Nokia 2014. All rights reserved.

```
      i += 3;
      j = i;
      while ((i < line.Length()) && (line[i] != ','))
       i++;

      // Copy longitude from the message
      lon.Copy(line.Mid(j, i-j));
      // If it's western hemisphere, negate the value
      if (line[i+1] == 'W')
       lon.Insert(0, _L8("-"));

      i += 3;
      j = i;
      while ((i < line.Length()) && (line[i] != ','))
       i++;

      // Read speed from the message
      TLex8 lex(line.Mid(j, i-j));
      lex.Val(speed);
      speed *= 10;
      if (lex.Peek() == '.')
      {
       lex.Inc();
       if (lex.Peek() != 0)
        speed += ((int)lex.Peek() - '0');
      }
      // Convert speed from knots to km/h
      speed *= 1852;
      speed /= 1000;

      i++;
      j = i;
      while ((i < line.Length()) && (line[i] != ','))
       i++;

      // Read heading from the message
      lex.Assign(line.Mid(j, i-j));
      lex.Val(heading);
      heading *= 10;
      // Check if it's not integer
      if (lex.Peek() == '.')
      {
       lex.Inc();
       if (lex.Peek() != 0)
        heading += ((int)lex.Peek() - '0');
      }
     }
    }

    line.Zero();
   }

   line.Append(data);

   iSendingSocket.Read(data, iStatus);
   SetActive();
```


(C) Copyright Nokia 2014. All rights reserved.

```
  }
  }
 // Error occurred...
 else
 {
  lat.Copy(ZERO);
  lon.Copy(ZERO);
 }
}

void GPS::DoCancel()
{
 if (state > 0)
  iSendingSocket.Close();
 iSocketServer.Close();
}

void GPS::ConstructL()
{
 TBTDeviceResponseParamsPckg resultPckg;

 CActiveScheduler::Add(this);

 // 1. Create a notifier
 RNotifier notif;
 User::LeaveIfError(notif.Connect());

 if (gpsid.Length() == 0)
 {
  state = 0;
  // 2. Start the device selection plug-in
  TBTDeviceSelectionParams selectionFilter;
  TUUID targetServiceClass(0x2345);
  selectionFilter.SetUUID(targetServiceClass);
  TBTDeviceSelectionParamsPckg pckg(selectionFilter);
  TRequestStatus status;
  notif.StartNotifierAndGetResponse(status,
KDeviceSelectionNotifierUid, pckg, resultPckg);
  User::After(2000000);

  // 3. Extract device name if it was returned
  User::WaitForRequest(status);

  User::LeaveIfError(iSocketServer.Connect());

  if (status.Int() == KErrNone)
  {
   User::LeaveIfError(iSendingSocket.Open(iSocketServer,
RFCOMM));
   TBTSockAddr address;
   gpsid.Copy(resultPckg().BDAddr().Des());
   address.SetBTAddr(resultPckg().BDAddr());
   // GPS devices usually use port 1 as data channel
   // so we don't have to query it
   address.SetPort(1);

   state = 1;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    iSendingSocket.Connect(address, iStatus);
    SetActive();
   }
   else
   {
    CHainMAppView::Static()->Notification(EGPSNotFound);
   }
  }
  else
  {
   User::LeaveIfError(iSocketServer.Connect());
   User::LeaveIfError(iSendingSocket.Open(iSocketServer, RFCOMM));

   TBTSockAddr address;
   TBTDevAddr a(gpsid);
   address.SetBTAddr(a);
   address.SetPort(1);

   state = 1;
   iSendingSocket.Connect(address, iStatus);
   SetActive();
  }
 }

 TPtrC GPS::Lat()
 {
  return lat;
 }

 TPtrC GPS::Lon()
 {
  return lon;
 }
```


(C) Copyright Nokia 2014. All rights reserved.