NMEA GPS Data UDP Forwarder

https://www.digi.com/resources/documentation/digidocs/90001537/references/r_gps_data_udp_forwarder.htm

Summary

This demo application runs forever, reading in NMEA sentences (output data from a GPS device) and forwards a subset of the data to one or more remote hosts via UDP/IP. The hypothetical system used for the demo is a Digi CP device with GPS within the vehicle, which is to forward ALL GPS data via Ethernet to a computer mounted within the vehicle, plus only send the RMC (Recommended Minimum Navigation Information) via cellular UDP/IP to a central Vehicle-Location-System (VLS).

To save data costs, the RMC message for the central VLS is only sent once per minute when the vehicle is moving, and once per 15 minutes if the vehicle is idle or the GPS signal is in error. This is important since the GPS device used in this demo creates about 200MB of data per month - and this model is a fairly QUIET one! Other GPS devices with more features can create 5 or 10 times more traffic.

Tested with the following equipment

- The USGlobalStat BU-353 USB GPS Receiver ($35 to $40 online - available everywhere; search for the model and you'll find offers galore)
- The scripts running on a Windows XP computer, with the GPS auto installed as COM6 (your's might vary)
- The scripts running on a Digi ConnectPort X4, with the GPS auto installed as /gps/0

Note that the method used for serial data allows this script to run on EITHER a Windows computer with the GPS installed as a USB-based serial port, or on Digi ConnectPort products supporting the GPS service (X4,X8,CP-WAN and so on) Check Virtual GPS NMEA Access to make sure your model and firmware includes this service.

Files required to run

Upload all of these to your Digi product's Python directory:

- digi_gps.py – main file, run this file
- digi_gps_config.py – edit this file to set the DESTINATION info, such as IP address, forward rates etc.

- digi_nmea.py – parses the GPS information stream
- digi_serial.py – the hardware-independent “serial port” object; works on the CPX4 and also under Windows
- Python_ext.zip - several standard Python modules needed – including the copy module

Other files

- pyserial-2.2.win32.exe pyserial module from sourceforge is required to run this under Windows. The Linux version of pyserial would allow this script to run under LINUX, however the serial port notation may change since I don't know where the GPS module will install itself.
- udp_sink_2101.py – simple dummy UDP host/server; listens on UDP port 2101 and prints (with timestamp) any ASCII messages seen
- udp_sink_2102.py – same as udp_sink_2101.py, but listens on UDP port 2102

When it runs, you will see SOMETHING like this (results may vary :-)

```
central does not want <GSA> sentences
local sending <$GPGSA,A,3,10,24,30,21,,,,,,,,5.0,2.8,4.2*3C
>
>>SPEED: 0.98 Kts - GPS says we are IDLE - not moving
local sending <$GPRMC,183037.000,A,4453.9342,N,09324.9895,W,0.98,30.37,090209,,*20
>
central does not want <GGA> sentences
local sending <$GPGGA,183038.000,4453.9342,N,09324.9898,W,1,04,2.8,307.4,M,-31.7,M,,0000*6F
>
central does not want <GSA> sentences
local sending <$GPGSA,A,3,10,24,30,21,,,,,,,,5.0,2.8,4.2*3C
>
>>SPEED: 1.19 Kts - GPS says we are IDLE - not moving
local sending <$GPRMC,183038.000,A,4453.9342,N,09324.9898,W,1.19,27.91,090209,,*20
>
```

Hopefully the comments in the file digi_gps_config.py explain what is required. Just note that you CANNOT use Windows NotePad.exe to edit these files because it mixes DOS and UNIX formats. Instead, WordPad.exe (also included in Windows) does NOT cause this problem

ZIP of the files

Digi_gps.zip

Extensions

This design could be extended (by you) to do the following:

- Use TCP/IP instead of UDP/IP. Since each GPS destination maintains its own socket, that socket could be changed to TCP or even SSL. However, TCP will increase your cost by 60% to 95%, and my own tests show UDP/IP is very reliable over cellular - a few dropped packets per 10,000.

- This demo has only 2 speeds for sending packets: 1 for idle and 1 for moving, yet a truck traveling at 60 miles-per-hour might benefit from 3 times more GPS data forwards than one traveling at 20 miles-per-hour. Thus make the rate at which packets are sent vary based on the change in position would help maximize central accuracy while minimizing data costs.

- Encrypt the payload with a simple AES algorithm - of course your host would need to use the same keys to decrypt.

- Manually monitor the time in the Digi product, and update if it drifts. Just be aware that GPS time IS NOT true time. Unlike UTC and time you get from a cell tower or via the Internet, GPS time is not adjusted for the wobble in the earth's path and so on. Thus there is a fudge adjustment you need to add, which today is about 15 seconds ... read online about GPS to learn how.

- Support a simple web page showing GPS info.

GPS Data UDP Forwarder updated on 25 Sep 2017 08:55 AM