LoRaTracker / **GPSTutorial**

A simple tutorial for troubleshooting GPSs on Arduino

38 commits **1** branch **0** releases **2** contributors

Branch: master New pull request Create new file Upload files Find File Clone or download

LoRaSAT Changes to 8M fix graph Latest commit 760e513 on 18 Apr

GPS performance comparisons	Changes to 8M fix graph	2 months ago
GPS_Sleep_Tester_UBLOX	Add GPS performance report	5 months ago
GPS_to_UKGRID	Add L86 results	4 months ago
Simple_HardwareSerial_GPS_Echo	Update instructions	7 months ago
Simple_SoftwareSerial_GPS_Echo	Change to readme	7 months ago
Readme.md	Updates to L76	4 months ago

Readme.md

GPS Tutorial

On public forums, such as Arduino, I see a great many posts along the lines of 'My GPS does not work'. Most often constructors connect a GPS to an Arduino, load up a complete application and find it does not work, there is no GPS location reported. Sometimes they do the sensible thing and load one of the simple examples that comes with an Arduino library, but still they find their GPS 'does not work'

In approximate order this would be my summary of the reasons behind a GPS that 'does not work'

1. GPS is indoors
2. GPS is connected incorrectly
3. Arduino program uses the wrong GPS baud rate
4. Arduino program is not correct
5. GPS is faulty
6. Arduino is faulty

So the first troubleshooting step is to realise that the GPS will probably only work when it's outside with a good view of the sky. It matters little how much you think you want or need the GPS to work indoors, that it worked before indoors, or that someone else says it should work indoors; **take it outside !**

If that does not solve your 'not working GPS' try a simple GPS echo program such as listed below. This reads character from the GPS and sends them to the Arduino IDE serial monitor so you can see what the GPS is up to. **Remember to take your GPS outdoors**.

Arduinos such as the Pro Mini and UNO only have one hardware serial port so you will need to use software serial to read the GPS, using software serial is described first. If you have an Arduino IDE supported device such as Atmega1284P or ATmega2560 then these have two or more hardware serial ports and these should be used instead of software serial, see 'Using Hardware Serial Ports' toward the end of this guide.

3.3V versus 5V logic GPSs and Arduinos

Some Arduinos, such as the UNO and Mega2560 use 5V logic. A lot of the actual GPS devices are 3.3V logic but are supplied on boards (modules) that have circuitry to convert the GPSs 3.3V logic to 5V. There are too many different GPS modules out there to list and identify which are for 3.3V connection only and which are for 5V connection, make sure you check before buying or using a GPS.

A GPS that is using 3.3V logic can be destroyed if you connect it to a 5V logic sytem such as UNO or Mega

Using Software Serial

This test program uses software to emulate a hardware serial port. It is not as reliable as using a hardware serial port and at speeds of 38400 and upwards you can miss characters from the serial device (GPS). A lot of GPSs use 9600 baud and this is normally reliable under softwareserial. The program is below, it continuously reads each character from the GPS and prints it to the serial monitor.

Note that software serial does not work on some Arduino pins. The full details of supported pins are found at this Internet link;

Software Serial Supported Pins

On a UNO, ProMicro or ATMega1284P you can use any available pins apart from 0 and 1, these are used for the serial monitor output. On the ATMega2560 you can only use these pins for software serial RX pin;

10,11,12,13,14,15,50,51,52,53,A8,A9,A10,A11,A12,A13,A14,A15.

You will need to tell the program what pin numbers you have the GPS TX and GPS RX pins connected to on the Arduino.

```
//Simple_SoftwareSerial_GPS_Echo

//Reads characters from a GPS using Software Serial and echoes them to the Arduino Serial Monitor at 115200 baud.

//Important Note
//--------------
//Make sure the #defines below for the GPS TX and GPS RX pin numbers below match what you have connected and that
the GPS
//baud rate used by the GPS is correct, its often 9600 baud but not always.

#define GPSTX A2 //pin number for GPS TX output - data from Arduino into GPS
#define GPSRX A3 //pin number for GPS RX input - to Arduino from GPS
#define GPSBaud 9600 //GPS Baud rate
#define Serial_Monitor_Baud 115200   //this is baud rate used for the Arduino IDE Serial Monitor

#include <Arduino.h>
#include <SoftwareSerial.h>
SoftwareSerial GPSserial(GPSRX, GPSTX);

void loop()
{
  while (GPSserial.available() > 0)
  Serial.write(GPSserial.read());
}

void setup()
{
 Serial.begin(Serial_Monitor_Baud);   //start Serial console ouput
 GPSserial.begin(GPSBaud);//start softserial for GPS at defined baud rate
}
```

This first printout below is what the GPS output will typically look like for a GPS that has just been turned on, but see the heading 'No GPS Characters' below if you don't see this sort of printout;

```
$GPTXT,01,01,02,u-blox ag - www.u-blox.com*50
$GPTXT,01,01,02,HW  UBX-G60xx  00040007 FF7FFFFFp*53
$GPTXT,01,01,02,ROM CORE 7.03 (45969) Mar 17 2011 16:18:34*59
$GPTXT,01,01,02,ANTSUPERV=AC SD PDoS SR*20
$GPTXT,01,01,02,ANTSTATUS=DONTKNOW*33
$GPRMC,,V,,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
$GPGSV,1,1,00*79
$GPGLL,,,,,,V,N*64
$GPRMC,,V,,,,,,,,,,N*53
```

A few seconds later you should see something like this;

```
$GPGSV,1,1,01,03,,,19*73
$GPGLL,,,,,,V,N*64
$GPRMC,,V,,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
```

Note that the line reporting the satellites in view, the GPS sentence $GPGSV, has changed from $GPGSV,1,1,00*79 (meaning no satellites in view)\ to $GPGSV,1,1,01,03,,,19*73 which shows there is one satellite in view. The satellite is number 03 and the signal strength is 19, which is very weak. To eventually get a good GPS fix you will need several satellites in view with a signal strength of 25+.

If the $GPGSV sentence never moves from the no satellites in view output format (see above) then either the GPS is faulty or its antenna is.

The next thing that should happen is that the GPS starts reporting the time, 11:39:41.00 in this case;

```
$GPGSV,1,1,01,03,,,25*7C
$GPGLL,,,,,113940.00,V,N*44
$GPRMC,113941.00,V,,,,,,,,,,N*72
$GPVTG,,,,,,,,,N*30
$GPGGA,113941.00,,,,,0,00,99.99,,,,,,*69
```

Within a minute or so most GPS with decent antennas and a good view of the sky should then start reporting the position, the content of the position sentences $GPRMC and $GPGGA should look something like this;

```
$GPRMC,111218.00,A,5133.58788,N,00313.12853,W,0.205,,261018,,,A*61
$GNGGA,111218.00,5133.58788,N,00313.12853,W,1,06,1.36,241.1,M,49.4,M,,*55
```

Some GPS, particularly the types used in high altitude ballooning that have small ceramic stick antennas or a simple wire, can take 5 minutes or more getting a fix. I often see it claimed that (presumably a good) GPS can take many minutes, 15+, to acquire the first fix. I have seen this numerous times myself and its always been on a GPS with a poor antenna or in a poor location. It is rare in my experience for a modern GPS with a good working antenna to take more than one minute to get a fix under good conditions (outside with a clear view of the sky) and most often its around 45 seconds or less when started from cold, the best ones can get a fix in 30 seconds or so from cold. If your GPS takes longer than a minute to get a fix under good conditions then assume the GPS or its antenna is faulty.

If a GPS is reporting a number of satellites in view such as a $GPGSV sentence that looks like this (06 satellites in view) but does not get a fix;

```
$GPGSV,2,1,06,03,79,240,27,11,44,145,30,17,49,291,31,18,34,116,20*78
$GPGSV,2,2,06,23,26,180,30,32,,,29*4A
```

Then it's either got a poor view of the sky, has a poor antenna or is faulty. There is no magic bit of code that will fix this problem.

No GPS Characters

If the serial monitor is blank then either you have a faulty GPS or its connected wrong, its also possible the GPS baud rate is wrong. Reversing the GPSRX and GPSTX pins can be tried, I have not known this damage a GPS, but no guarantees.

If the characters you see on the serial monitor are garbage, then it likely you have the GPS baud rate wrong. Check the documentation for your GPS.

Using Hardware Serial Ports

If you have an Arduino with additional hardware serial ports such as the ATmega1284P or ATmega 2560 you should use the hardware serial ports for the GPS, the echo code is here;

```
//Simple_HardwareSerial_GPS_Echo

//Reads characters from a GPS using one of the Arduinos Hardware Serial
```

```
//ports and echos them to the Arduino Serial Monitor at 115200 baud.

//Define the hardware serial port and the baud rate the GPS is using below, 9600 baud is common.

#include <Arduino.h>

#define GPSserial Serial1           //define hardware serial port the GPS is connected to, can be Serial1,
Serial2, Serial3 or Serial4
#define GPSBaud 9600                //GPS Baud rate
#define Serial_Monitor_Baud 115200  //this is baud rate used for the Arduino IDE Serial Monitor


void loop()
{
  while (GPSserial.available() > 0)
  Serial.write(GPSserial.read());
}


void setup()
{
 Serial.begin(Serial_Monitor_Baud);   //setup Serial monitor ouput
 GPSserial.begin(GPSBaud);            //start Hardware for GPS at defined baud rate
```

Note: You should change the **#define GPSserial Serial1** line to indicate the hardware serial port you want to use, Serial1, Serial2 or Serial3.

Note on Ublox GPSs

Some Ublox GPS will start-up with the position sentences $GPRMC and $GPGGA in $GNRMC and $GNGGA format like this;

```
$GNRMC,111218.00,A,5133.58788,N,00313.12853,W,0.205,,261018,,,A*7F
$GNGGA,111218.00,5133.58788,N,00313.12853,W,1,06,1.36,241.1,M,49.4,M,,*55
```

Do check that the Arduino GPS library you are using supports the $GNRMC and $GNGGA format, the latest copy of TinyGPSPlus on github does;

https://github.com/mikalhart/TinyGPSPlus

GPS Hot fix mode

Its a fact of life that GPSs consume a lot of power and this can be an issue if extended long term operation from small batteries is required, so for a portable GPS tracker. As an example I measured the current consumption of some different GPS as they are acquiring their first fix. When the GPS does get a fix the running current consumption can drop to about half the values shown (on average) but for the purposes of comparison the fist fix current is shown, If the software standby\backup current of the GPS is known it is in brackets.

UBLOX NEO 6M Bare module 60mA - 63mA (56uA) UBLOX NEO 6M Breakout board 57mA - 70mA (7mA) Beitian BN220 (UbloxM8) Breakout 52mA - 58mA (7.5mA) UBLOX 8 Breakout (?) 49mA - 58mA UBLOX 8N Breakout (?) 46mA - 52mA UBLOX MAX8Q Bare module 25mA - 27mA (19uA) GlobalTop PA6H 21mA - 23mA Quectel L80 18 - 20mA (1mA)

To use the UBLOX NEO 6M as an example, and they are popular and quite cheap on eBay, then at average 60mA if run continuously in a tracker powered by AA Alkalines, the batteries would only last about 48 hours, and that is just running the GPS.

When first turned on the GPS needs to download the GPS Almanac and ephemeris information so it knows the information on the GPS satellites in order to calculate a position. This download takes a while and is the reason the GPS can take a minute or so to get a first fix.

Download of the entire catalogue can take a bit longer than one minute, so when first turned on its a good idea to let the GPS run for a few minutes. After this period, provided the GPS has a backup facility you can turn off the main power to the GPS and it will retain the Almanac and ephemeris information. When the power to the GPS is turned back on, then it can quickly check the satellites and acquire a new fix in 5 seconds or so. This is obviously more power efficient than running the GPS all the time.

However the satellites the GPS receiver can see are moving across the sky and from time to time the GPS will need to download updated Almanac and ephemeris data which takes time. To see the effect I set up a tracker to read a GPS, wait for a new fix, then transmit the fix and the GPS on time to a remote receiver. I set to power on\off time of the GPS to 10 minutes and plotted the results over 24 hours. The results for a range of GPS are in the **GPS performance comparisons** folder in this repository.

By adding up all the fix times over 24 hours I was able to calculate how much power the one GPS was using in one day, this was around 17mAhr per day, quite an improvement over the 432mAhr that would be used if the GPS was powered on all the time.

Stuart Robinson

www.LoRaTracker.uk

October 2018