A u-blox GNSS receiver for orienteering mapping

Libor Pecháček
January 2019

Abstract

This document describes a GNSS receiver designed for orienteering mapping and experience gained from this endeavor. The text is intended as a view into consumer grade electronics commonly used in orienteering and as a hint about future development directions.

Table of Contents

Introduction

My use of GNSS receiver for map making started inconspicuously during a map revision for a club training. I was mapping a newly harvested area with tracks and fences with the traditional take-azimuth-pace-draw-on-paper methodology. However, I had a GPS data logger in my pocket. Later at home I put my scanned forest original together the GPS trace and saw that many of the lines overlap. That led me to trying out a tablet together with the GPS logger on a 4 km^2 map. The results were satisfactory in terms of position accuracy. However, and that was an important for the future experiments, the tablet occasionally failed to establish connection with the logger. I opened the logger case to see if there is something obvious to fix inside. Needless to say that I didn't see anything but I learned about how the device design. Later I tried out a GPS+GLONASS receiver and saw that it does not bring additional accuracy in my terrain. At least that drew my attention towards multi-constellation GNSS receivers. In summer 2018 I heard the news that the European Galileo navigation system is nearing completion with four additional satellites being launched into their orbits. That sparked my interest in how would a GPS+GLONASS+Galileo receiver behave.

Hardware design

The data logger case

As said earlier, I was using Canmore GT-750FL[1] data logger for my field work and that my tablet occasionally failed to establish Bluetooth connection to the receiver. I was interested to see if there is perhaps some trivial problem in it that I can fix and get the data logger work reliably.

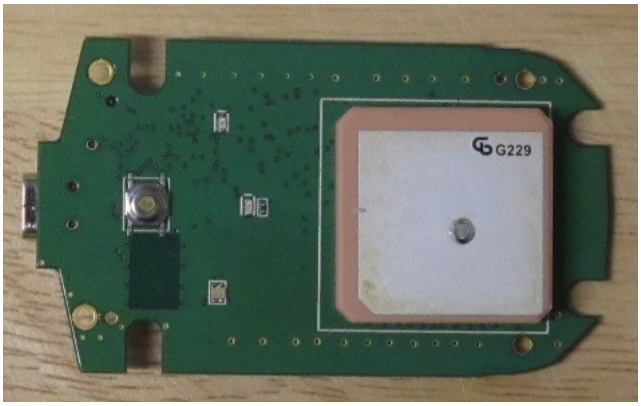

Figure 1: Canmore GT-750FL board – top side

Upon removing the fours screws which hold the device together the electronic board appeared. On one side there is receiver antenna, the common ceramic patch type, indicator LED's and power button. On the other side there is all the magic I was looking for.

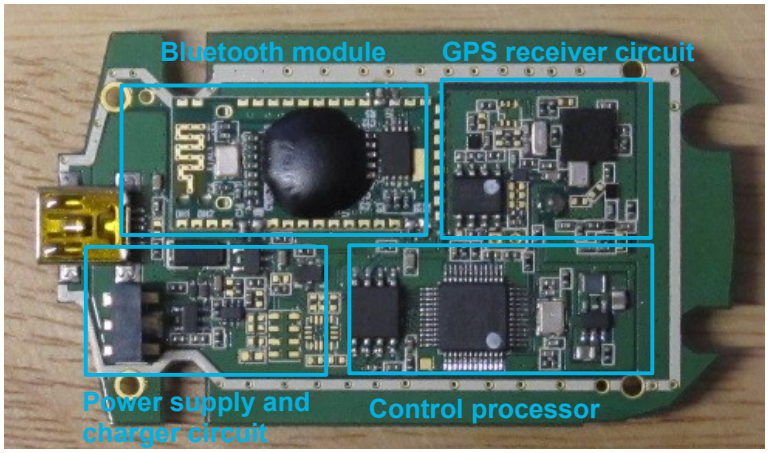

Figure 2: Canmore GT-750FL board – bottom side

At first glance it looked like complex device but after a few tens of minutes of staring at the board, and looking up the chip specifications, the design stood out. On he left there are battery charger and voltage stabilizer circuits and Bluetooth

communication module. On the right there are GPS receiver circuit and processor taking care of the data logging function.

The GPS chip is SiRFstar IV, contrary to product documentation. The processor is an ST Microelectronics STM32 micro controller with additional flash memory for data log.

The Bluetooth module does not have any marking but the important point is that is it is soldered on top of the board and connects at only six points. The observation is that the Bluetooth function is isolated into a standard looking module which might be possible to get separately.

Figure 3: Bluetooth module sitting on the board.

Overall, the lesson learned was that a Bluetooth receiver usable for fieldwork consists of a GNSS receiver, Bluetooth communication module and a control processor which is somehow passing data between the two.

Galileo

With the news about four more Galileo satellites being guided to their orbits[2], I felt the craving for trying out the Galileo system. At the time I could not find a commercial receiver for a price I would pay for an experiment so I started looking for discrete components to combine into a usable device. I was looking for Bluetooth serial port module, a small and easily programmable micro controller and a Galileo capable GNSS receiver. Starting at AliExpress I gathered all the components after a few hours of research. I bought a "*Bluetooth 4.0 For Arduino Android IOS HM-10 BLE CC2540 CC2541 Serial Wireless Module*", "*STM32F103RCBT6 Maple Mini ARM Cortex-M3 leaflabs Leaf Controller Board Module 3.3V For Arduino STM32*" and "*UBLOX Micro GPSV5 NEO-M8N GPS Module GNSS HMC5983 SAW LNA Triple Band Antenna ANT*". As an extra I added "*0.96 inch IIC Serial Yellow Blue OLED Display Module 128X64 I2C SSD1306 12864 LCD Screen Board GND VCC SCL SDA 0.96" for Arduino*" for the possibility of displaying data from the micro controller.

All the components arrived quickly into my home and I started experimenting. I knew that the GNSS module has serial line output so I focused on the Bluetooth part to get the data into my tablet. I took the "Android Bluetooth 4.0 serial wireless module" and paired it with the tablet. Pairing worked but data didn't flow into Bluetooth GPS application. Experts likely have smile on their face now. The point is that Bluetooth 4.0 Low Energy (BLE) standard does not specify the serial port service which all the data loggers use for moving bits into Android devices. The correct service is Bluetooth 2.0 RFCOMM, governed by Serial Port Profile (SPP) specification. Back to web browser and search for Bluetooth RFCOMM module. HC-06 seems to be an established product for this purpose.

2 https://www.gsa.europa.eu/sites/default/files/content/press_releases/gsa_pr_18_02_galileo_quartet_successfully_launched.pdf

The HC-06 module did far much better. However, still I spent a few hours making it work. The product documentation is somewhat superficial and the product behavior diverges at a few points. The main hurdle was making the clone listen to configuration commands sent over the serial line. Typically such devices listen to so called AT-commands. After sending “AT” followed by enter key they respond with “OK”. This one did not. The trick was to send “AT” as one string, i.e. not two consecutive characters, *without* the carriage return character generated by enter key. After figuring this out it was easy to program the unit to my needs.

Then there was the GNSS receiver. I was careful to buy a genuine u-blox receiver and not one of the clones. That move paid off in availability of documentation and the device behaving to the specs. I was even surprised that there is u-Center software for configuring and testing u-blox chips. It took a while to wrap my head around the configuration mechanism and find all the tunables that enable Galileo system reception and reporting but in the end I succeeded to receive GPS+GLONASS+Galileo at my kitchen table! At this point I had NMEA data from the receiver and could move them to the Android Bluetooth GPS application for further processing.

Next I looked into the power system to make the receiver portable. All the components operate at 3.3 V. I also had some flat Lithium rechargeable battery cells from a laptop. Thankfully, battery protector and charger modules are easy to get at the local electronics shop so I bought them, together with a voltage stabilizer for the Bluetooth module and battery status display. The design shaped up. Connect the battery cell, whose voltage slides from 4.2 V down to 2.8 V as it discharges, to the battery side of the charger module. From the device side of charger module current goes via power switch and battery monitor to the GNSS module and via voltage stabilizer to the Bluetooth module. As the last step, connect the GNSS module serial line directly to the Bluetooth module serial port. The micro controller I bought is not necessary as both the modules retain their configuration and there is no need for power-up initialization.

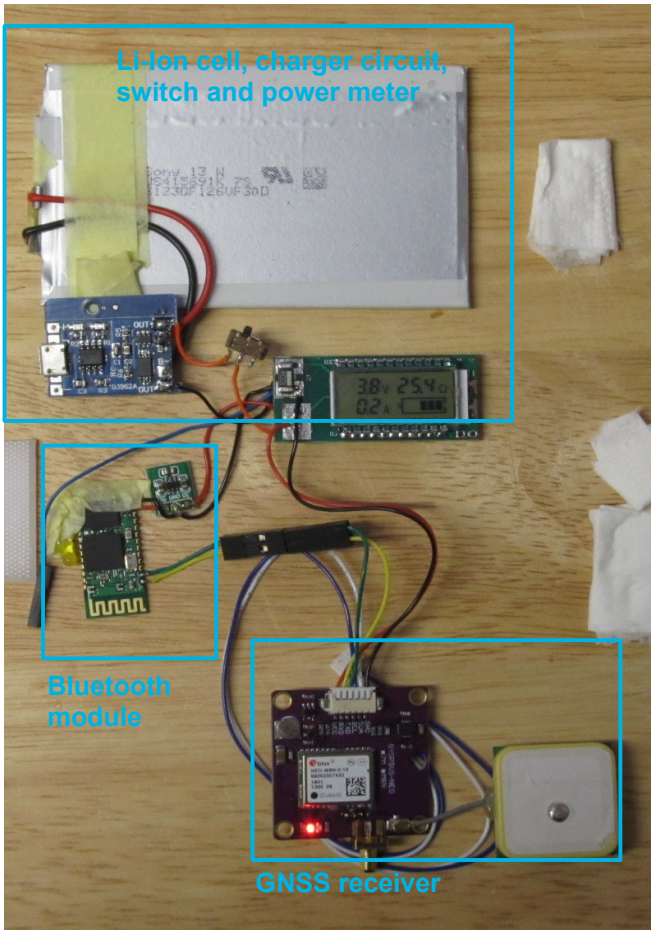

Figure 4: u-blox based Bluetooth GNSS receiver

The above is a working Bluetooth GNSS receiver. Given the choice of components, it can work even without additional component configuration. The u-blox GNSS module sends data at 9600 baud and the HC-06 Bluetooth module also operates at 9600

baud. However, to fully utilize the GNSS chip capabilities, it is advisable to raise the inter-component communication speed to 38400 baud. Higher speeds, however, make trouble in my circuit as the Bluetooth module seems incapable to reliably operate at fast pace.

After all the components were connected (soldering skills are required), the device needed to get some mechanical robustness. I placed everything atop the Li-Ion cell and separated the larger components from the cell with folded paper tissues.

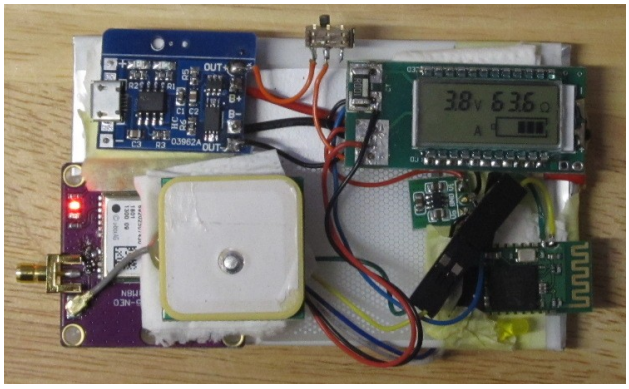

Figure 5: All components placed atop the Li-Ion cell

Especially the charger circuit emits a lot of heat during charging which should not transfer into the cell. Also the sharp module board edges should not touch the cell surface as Li-Ion cells are sensitive to damage and can even catch fire upon mechanical damage. When the components were in correct positions, I fixed them with painter's tape. The device was made waterproof by placing it in a plastic zip bag. Please note that the gold plated antenna connector on the GNSS module is not part of AliExpress delivery.

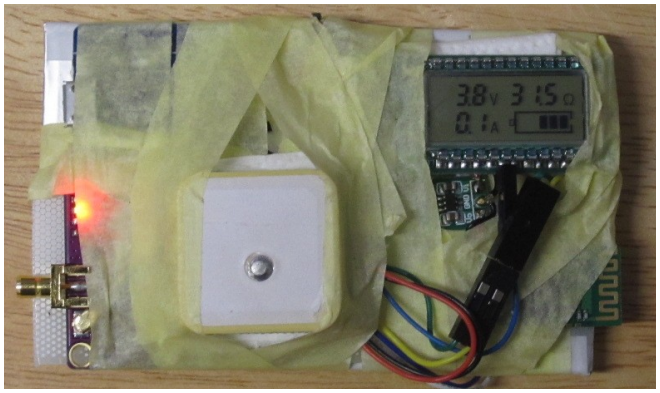

Figure 6: Complete Bluetooth GNSS receiver capable of receiving GPS-GLONASS-Galileo or GPS-BeiDou-Galileo

KISS at work

KISS is a popular engineering principle, often being transcribed as “Keep it simple stupid”. The common wisdom behind the words is that simpler systems tend to work better than complicated ones. What could be simplified about the above design?

1. Antenna
   The GYGPSV5-NEO module board gets delivered with active antenna. The point of amplifier in active antenna is covering signal losses on the cable from antenna to the receiver. With about 5 centimeters of cable between antenna and the board the amplifier just adds noise. Passive antenna will do.
2. Power system
   Few people have Li-Ion cells in their drawer. There is no point in assembling the power system from discrete parts. Use power bank as the source.
3. Bluetooth module
   Making the HC-06 clone work consumed an enormous amount of time. Later reconfiguration was equally painful. Use well documented components from reliable suppliers.

Bluetooth GNSS receiver – take 2

With the lessons learned earlier I bought second set of components. This time it was Drotek’s *Ublox NEO-M8N GPS module* (Drotek is u-blox’s technology partner) and SparkFun’s *SparkFun Bluetooth Modem - BlueSMiRF Silver*. BlueSMiRF Silver is a Class 2 Bluetooth device with 2.5 mW transmission power, which is enough for this device. Both devices can be powered from 5 V supply. The Drotek GNSS module has both serial and USB connection and comes with a cable that has Dupont connectors at one end.

Figure 7: Drotek GNSS module with cable

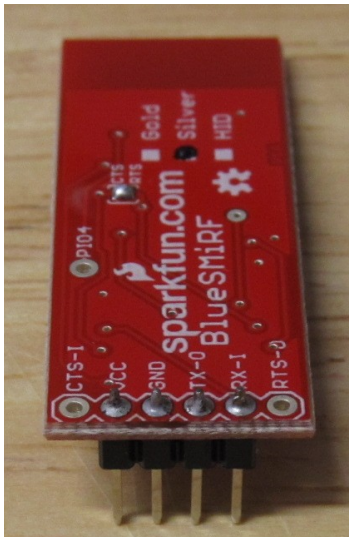

Figure 8: BlueSMiRF Silver module after soldering the pin header

The SparkFun Bluetooth module has holes ready for a pin header that can connect to the Drotek cable. This is the only soldering needed in this design. A local electronics repair shop can do the soldering for you if you don't have equipment for this task.

As part of the soldering RTS and CTS pads, clearly marked on the board, should be shorted. This is a recommendation from the module vendor for the type of connection we are going to use.

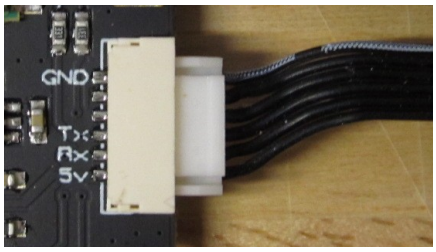

Figure 9: Connector markings on the GNSS module

Now that the pin header is in place, the devices can be connected. The six-wire cable from Drotek module uses four wires and their connections are marked on the board. The first wire in the flat cable (marked with color strip) is the ground. Connect it to the GND pin on the Bluetooth module. GNSS module 5V goes to VCC on Bluetooth module. The communication lines are crossed. Transmit pin of one module is connected to receive pin on the other module. Connect TX from one module to RX on the other one and vice versa.

This completes connection between the modules. Double check that GNSS module's GND and 5V go to GND and VCC on the Bluetooth module as well as TX on one side goes to RX on the other. After this the device is ready for programming.

The only essential step is the change of GNSS module serial line speed from the default 9600 baud to 115200 baud which is the Bluetooth module setting. Do it with the u-Center software after connecting the GNSS module with USB cable to your computer.

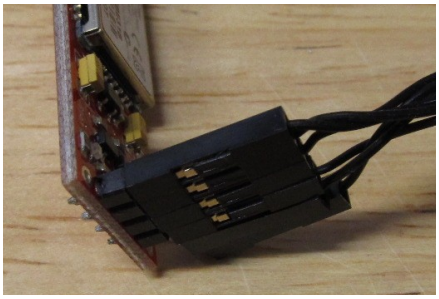

Figure 10: Cable connection to the Bluetooth module - two wires are unused

After plugging the USB cable into computer the blue power LED on GNSS module will light and red stat LED on the Bluetooth module will blink. The GNSS module will appear as a new COM port in Windows after installing the respective driver which comes with u-Center software. Refer to u-Center documentation[3] for instructions on how to connect the module and work with the software. The port settings can be found under UBX-CFG-PRT message. Change UART 1 baud rate to 115200 and save the configuration to non-volatile memory with UBX-CFG-CFG where you choose *Save current configuration* option.

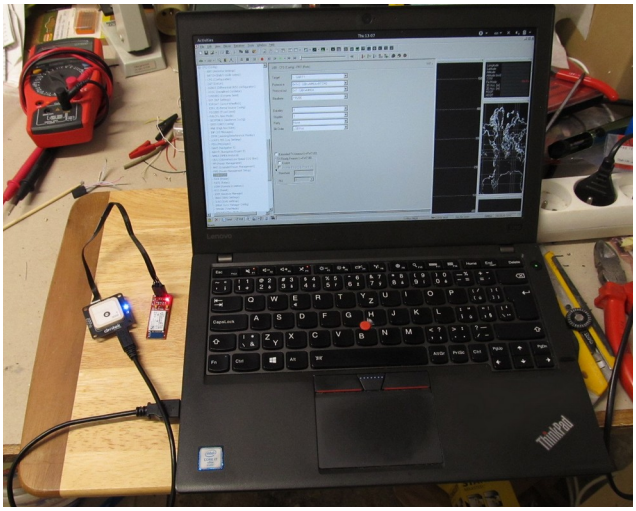

Figure 11: Programming the GNSS module

At this point you should be able to connect your Android device with the Bluetooth module and see NMEA messages in the Bluetooth GPS application log window. If it's not the case, check serial port in the configuration and re-check the cable connections.

As mentioned earlier, the device can be made portable by connecting it to a power bank. I had some small one in my drawer. You may opt for a flat model which can better fit into pocket. After applying a few paper tissues for insulation and a bit of painter's tape the device is ready for a trip.

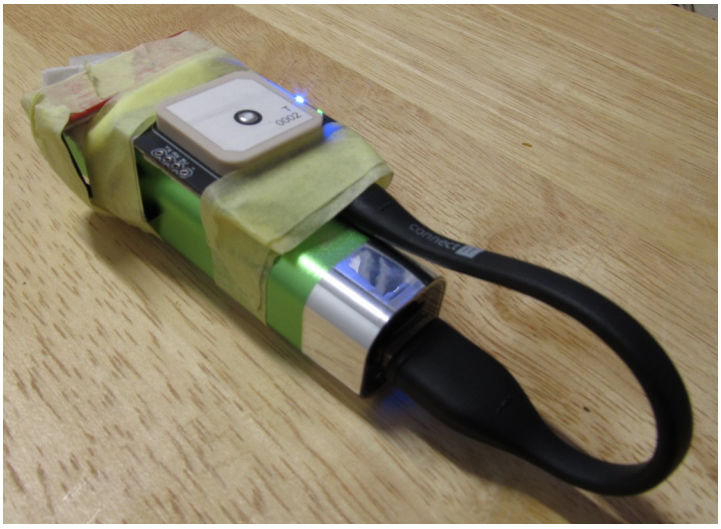

Figure 12: With connected power bank

Figure 13: Device test under open sky

In default configuration the receiver picks up GPS and GLONASS. See Software configuration section for details about tuning.

3 https://www.u-blox.com/sites/default/files/u-center_UserGuide_(UBX-13005250).pdf, How To section contains instructions on changing the port baud rate

Software configuration and usage

Connecting the receiver to computer

Depending on design there are two methods. The GYGPSV5-NEO board + HC-06 can be connected via Bluetooth. The Drotek board has a micro USB port which can be connected directly to a computer. In both cases the receiver port should appear as a new COM port in Windows.

It is possible to connect the module to a Linux based host and run u-center in Wine. If you are interested in more detailed instructions, contact the author of this document.

u-Center software

u-blox deliver evaluation software for their GNSS products. The software allows examining and adjusting every documented aspect of the GNSS receiver. Specifically, receiver status can be explored in great detail, including signal strengths, signal quality, signal jamming detection, multipath signal propagation and DPGS/SBAS status. Use of individual GNSS constellations and augmentation systems can be configured in the program, along with message update rate and message content. Refer to u-center user guide for instructions and u-blox Neo M8 protocol description for explanation of the individual parameters[4].

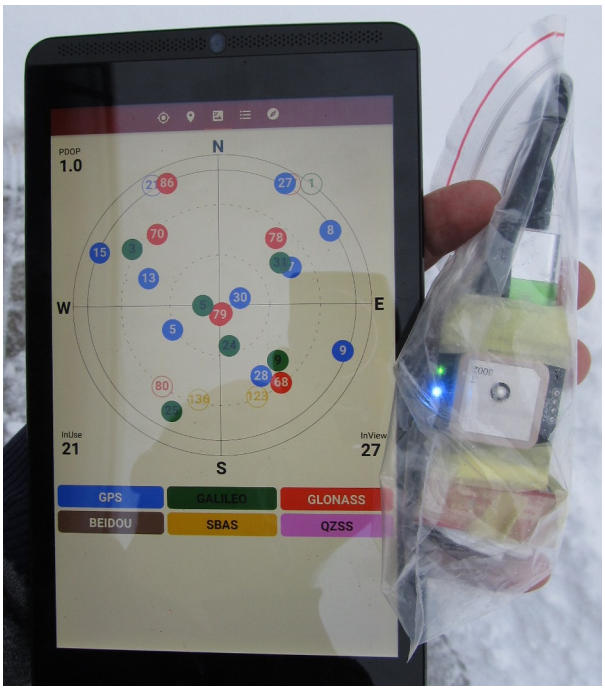

Figure 14: Sky map as presented in GNSS Commander - 27 satellites in view, 21 used for position fix. The three colors represent GPS, Galileo and GLONASS.

In the above described design the following parameters need to be changed to make effective use of Galileo signals:

- Configure GNSS constellations
- Make the receiver report Galileo satellites in view (optional)
- Change serial port speed (if needed)

GNSS constelltions are selected using UBX-CFG-GNSS message[5]. Open u-center, connect it to your receiver and open Messages view. Navigate to UBX-CFG-GNSS in the messages tree. Default enabled

4 https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_(UBX-13003221)_Public.pdf

5 https://www.u-blox.com/sites/default/files/u-center_UserGuide_(UBX-13005250).pdf, How To section contains instructions on changing the constellations selection.

configurations wil appear. Check the Enabled box next to Galileo and click Send button at the bottom of the window. At this point the receiver processes Galileo signals. For making the change permanent, go to UBX-CFG-CFG, choose Save current configuration and click Send.

While Galileo reception is enabled, the satellites are not reported in NMEA messages. The thing is that NMEA by default supports only two-digit satellite numbers. Galileo uses three-digit numbers as two-digit ones have been consumed by GPS and GLONASS. You can make the receiver make sending the three-digit satellite numbers by choosing UBX-CFG-NMEA and switching the NMEA Version field to value 4.1. Now the receiver will start sending GSV NMEA messages on the serial line. Again, making this change permanent requires saving current configutation into flash memory.

In case of the Drotek board, serial port speed can be changed freely via the USB port. Refer to u-center User Guide How To section for instructions on port speed change. Choose 115200 baud, send it to the receiver and save the configuration change into flash memory.

The Bluetooth-only design requires separate configuration of the GNSS board and the Bluetooth module using a TTL-level serial line converter. For this reason there is the black Dupont connector on the serial line in the first design.

Position accuracy reporting

These days HDOP value only weakly correlates with position accuracy estimate. The receiver has more exact position accuracy estimates based on its own pseudorange error measurements and corrections in use. So while Dilution Of Precision (DOP) is a scalar number expressing how good is the satellite constellation geometry for calculations, there are two sources of more relevant information. UBX-NAV-PVT shows *receiver's idea of position accuracy* in metres[6] and NMEA GST message contains *standard deviation* of longitude and latitude errors in metres. The standard deviation figure is 1-σ accuracy estimate and with 68% probability the true location lies within the indicated distance[7] and with 95% probability the true location is within double of that distance, which is known as 2-σ estimate. Strictly speaking, even the the accuracy estimate itself is subject to probability too. In case a GNSS signal spoofing attack[8] is mounted on your receiver,

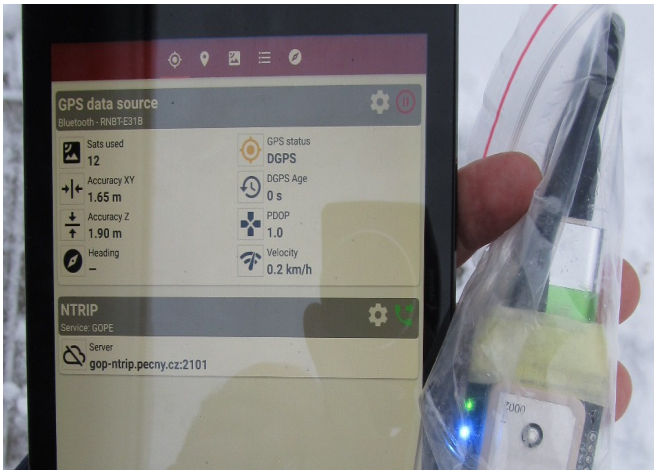

Figure 15: Position accuracy estimate as presented in GNSS Commander – PDOP 1.0, estimated 1-σ horizontal accuracy is 1.65 m

6 As of writing this text the author does know how is the UBX-NAV-PVT accuracy estimate calculated. Observations show that they are slightly higher than lon/lat standard deviation figures.

7 https://en.wikipedia.org/wiki/68-95-99.7_rule

8 https://en.wikipedia.org/wiki/Spoofing_attack#GPS_spoofing

the reported numbers lose any ties to the reality perceived by humans. No matter how reliable technology becomes, it’s an aid for work and use of common sense is still required.

Another important indicator is use of corrections. They can come as DGNSS corrections via RTCM stream on any port or, perhaps more often, as SBAS corrections from the geostationary service. Use of corrections is already reflected in multiple NMEA sentences issued by the receiver and it is beneficial to have an application that presents this data to the user. To date, GNSS Commander is one of these applications.

GST message type needs to be enabled in CFG-MSG. Choose F0-07 NMEA GxGST and check the check box next to UART1 label. Click Send button and store configuration into flash memory to make the change permanent.

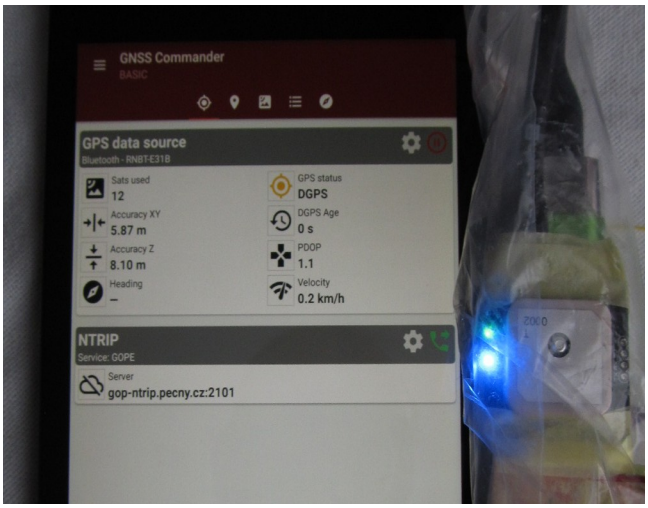

Figure 16: Position accuracy estimate in indoor environment – PDOP 1.1, estimated 1-σ horizontal accuracy is 5.87 m

Which constellations to choose

ublox NEO M8N can track all constellations but their combination is limited. The module features two radio front ends and therefore can receive signals on two distinct center frequencies. GPS, Galileo and the local systems broadcast at 1575 MHz. GLONASS and Beidou use two other frequencies. The chosen constellation must reflect this fact.

GPS+GLONASS+Galileo looks like a good choice for Europe as GLONASS corrections are broadcast in SBAS and DGNSS systems. East Asian locations may benefit from GPS+BeiDou+Galileo+QZSS due to higher number of BeiDou and QZSS satellites.

Conclusion

This exersise has demonstrated that it is possible to build a multi-constellation GNSS receiver for about the price of Garmin GLO with minimum knowledge of electrical engineering. The resulting device performs on par with commonly used consumer devices and has been successfully used to create a new map.

The need for manual assembly is offset by possibility to tune and debug the device. For example, the possibility of feeding DGNSS corrections[9] into the device in a controlled way is open. GYGPSV5-NEO board allows attaching external antenna after soldering a connector onto the board. The same design can be reused for future enhancements. When there are multi-frequency modules available in the future, their UART output can be wired to the Bluetooth module in the same way, long before similar commercial devices offering the same performance level hit the market.

On the software side, complete receiver documentation opens a way towards building mobile application that would report accuracy estimates and warn if they exceed acceptable limits, feed A-GNSS[10] data to the receiver and reporting .

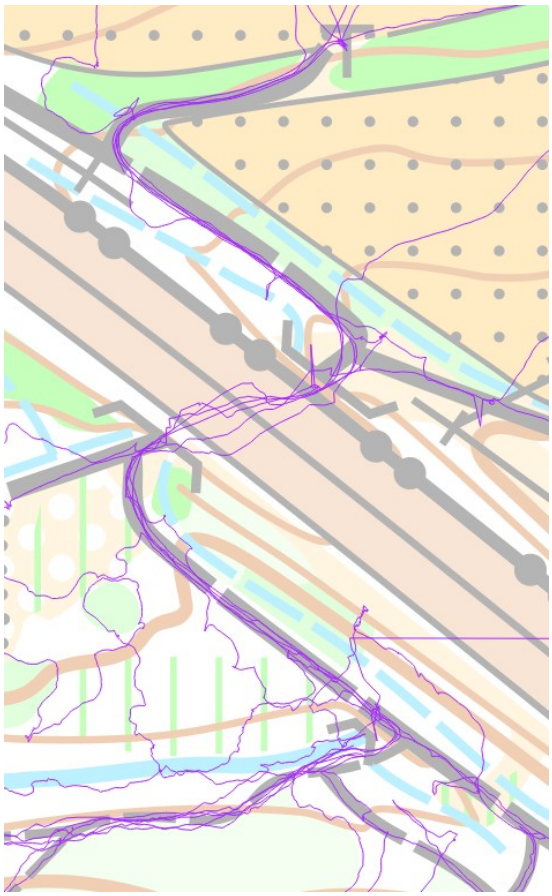

Figure 17: Receiver accuracy demonstration

9 Corrections from a national or experimental DGNSS network as a RTCM stream.

10 https://en.wikipedia.org/wiki/Assisted_GPS, u-blox offer access to their A-GNSS service called AssistNow via Internet.

Lessons learned

Choice of power bank for the KISS design

Not every power bank is suitable as a power source for the Drotek module. I've bought very cheap and lightweight Omega 2000mAh Credit Card Alu. The module failed to get proper fix when it was powered from the power bank output connector. In the end, I've crack-opened the power bank case and connected the module directly to the inside lithium cell via a switch. Obviously not every power bank is suitable for powering the receiver and some experimentation may be needed.

If you decide to follow my example and disassemble a power bank, be careful enough to keep the lithium cell intact and never bypass the cell protection circuit. The protection circuit is usually attached onto the cell itself so tap into the point where the wires from the battery connect to the power management board.

Trouble with the memory backup battery

In both the GYGPS-V5 board and the Drotek board there is a miniature accumulator that should help keeping the module memory and real time clock running. However, this component turned out to be troubled. It fails in the job of keeping memory contents in place as satellite ephemerides and almanacs get lost upon power down. The worse part, however, comes after a few months of sleep. The accumulator gets into a deep discharge and on power-on the module fails to restore configuration from flash memory. The likely reason is that the volatile memory fails to operate due to undervoltage caused by the deeply discharged accumulator. The visible symptom is that the modules does not communicate over Bluetooth. The non-intrusive solution is to let the module powered on until the accumulator charges at least a little bit (say, about ten minutes) and then quickly turn the power off and on. After that all is back to normal.

In my first receiver I disconnected the accumulator by removing the on-board 10 kΩ resistor and wired the module backup power pin to the main battery via a ~68 kΩ resistor and a 3.3 V Zener diode. The effect is that the module keeps ephemerides and almanacs, resulting in faster time to fix and more reliable satellite signal search on startup. At this point please note that ephemerides are valid for about 3-4 hours and almanacs expire in two weeks.

License note

This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). You are free to share and adapt the document for any purpose, even commercially, as long as you properly attribute the original work and refrain from applying legal terms or technological measures that legally restrict others from doing anything the license permits. The licensor cannot revoke these freedoms as long as you follow the license terms.

See https://creativecommons.org/licenses/by/4.0/legalcode for the full license text.

Revision history

Revision	*Date*	*Name*	*Comment*
R01	2019-01-16	lpechacek	Initial release
R02	2020-01-27	lpechacek	Add Lessons Learned