OGRP - Open GNSS Receiver Protocol

Dirk KOWALEWSKI, Frank HEINEN, Germany,Bo GUSTAFSON, USA
Per GUSTAFSON, Sweden

Key words: GNSS, RTK, Boards, Receiver, OGRP, OSRP

SUMMARY

In this document we describe the idea and the development of a new type of GNSS receiver. First, we have a new open format for transporting GNSS created data and this developed oriented to an object format that greatly simplifies development and allows more effective programming. The open LINUX OS allows the developer in that it can be programmed directly on the and for GNSS board completely new perspectives. Nevertheless, this GNSS receiver is still small and easy to handle and with low power consumption and is otherwise in accordance with the major manufacturers in nothing. He can receive and process all GNSS signals and has a very fast and robust RTK algorithm. Furthermore, you also have the possibility of their own developments to install directly on the board. This makes this product unique and gives the customer and the system integrator unprecedented freedoms and opportunities for development.

ZUSAMMENFASSUNG

In diesem Dokument beschreiben wir die Idee und die Entwicklung eines neuartigen GNSS Receivers. Als erstes haben wir ein neues offenes Format zum Transport von GNSS Daten geschaffen und dieses weiterentwickelt zu einem objekt-orientierten Format, das Entwicklungen enorm vereinfacht und effektivere Programmierung zulässt. Das offene LINUX OS ermöglicht dem Entwickler, dadurch dass er direkt auf dem und für das GNSS Board programmieren kann, völlig neue Perspektiven. Trotzdem ist dieser GNSS Empfänger noch klein und handlich und mit geringem Stromverbrauch und steht auch sonst den großen Herstellern in nichts nach. Er kann alle GNSS Signale empfangen und verarbeiten und verfügt über einen sehr schnellen und robusten RTK Algorithmus. Des Weiteren haben Sie aber auch die Möglichkeit ihre eigenen Entwicklungen direkt auf dem Board zu installieren. Das macht dieses Produkt einzigartig und gibt dem Endkunden und dem Systemintegrator noch nie dagewesene Freiheiten und Entfaltungsmöglichkeiten.

OGRP - Open GNSS Receiver Protocol

Dirk KOWALEWSKI, Frank HEINEN, Germany,Bo GUSTAFSON, USA
Per GUSTAFSON, Sweden

1. MOTIVATION AND TARGETS

1.1 Motivation

If you take a look at the manufactures of precise and Multi-frequency receivers you can find not more than one dozen producers. You can buy complete integrated systems for surveying or machine control, ready to use and packed in boxes or you can buy a discrete receiver, i.e. only the board (card). Most of the manufactures give you the option to buy only GPS or pay more for GPS and GLONASS. You can pay for single, double or triple frequency receivers. It is possible to spend money on an RTK engine and on many options. Nobody gives you the chance to work with the original raw data or to implement you own RTK engine or other software, directly on the board and much less to implement your own tracking loops or search routines. Presently it is impossible to fully comprehend the results of the RTK engine. The argument of the big manufactures is that you do not need to understand, we do everything fine for you. That is only a part of the truth. If we stay with the example RTK engine, many applications require customized algorithms. In the agricultural sector there are vehicles moving at less than 5 km / h. If you try this with a standard RTK engine and map the movements of the vehicle exactly one will notice that it does not work. There are many good examples where the commercial RTK engines fail. This was our motivation to develop an open GNSS receiver with an open protocol. We call the open format OGRP (Open GNSS Receiver Protocol) respectively OSRP (Open Source Receiver Protocol). The receiver named precisely what it is: an OSR – Open Source Receiver.

1.2 Targets

The advantages and the scope are obvious. We want to allow integrators and users to access all raw data, develop and install their own applications directly on the board. Thus, the developer can write his/her own RTK engine tailored to his individual needs. Nevertheless, it is not only the RTK software; there are many other applications, which standard receiver and standard software does not supported. Maybe your application is timing or a new use that now becomes feasible. These freedoms and these opportunities are not only science and the developers to Good. Finally, you can benefit to a considerable extent of the end customer that he gets a cut for him GNSS receiver.

2. STATE OT THE SCIENCE

A good starting point to evaluate the prior art is the website of the "open source" GNSS receiver project GPL GPS. The hardware platforms used there is based on the Zarlink GP4020. The activities seem unfortunately to end with a tutorial on GNSS Solutions at the ION2006. A similar approach taken by the "University of New South Wales" utilized a Namuru V1 receiver used in the course of a project of DLR. In addition, the Namuru-V1 receiver set up the Zarlink GP4020. Disadvantage of this approach was that the hardware built on an ASIC chipset, which could not be sell in large quantities and its technology was obsolete after some time in terms of hardware channels and processor used. Zarlink, now named Microsemi, refer on their website to a new development environment. Sigtec navigation PTY Ltd. this split in 2004 and became SigNav which is independent of Sigtec. SigNav was purchased in September 2011 by u-blox and the open development platform is not available for purchase from u-blox. All development kits of u-blox are indeed configurable, but only provide a ready PVT solution. The advancements of the Namuru receiver V2 and V3 are developments with FPGA solution designed specifically for LEO-Input use, special methods used to get a space-qualified, radiation-hardened solution rather than a highly accurate solution with the necessary computing power. The Aquarius Firmware can control the new versions of the Namuru receiver. In addition a wide range of approaches to Software Defined Radio (SDR) are found, but all have the disadvantage that although they show near real-time performance under laboratory-like circumstances, they are unsuitable for realistic real life scenarios, due to their size, weight, energy consumption etc. are.

3. OGRP AND OSRP

3.1 OGRP

Open GNSS Receiver Protocol for Open Source Receiver

Background

During the attempt to develop an open source receiver we needed a protocol for exchanging data between the components of a GNSS Receiver or in connection with external Hard or Software. The protocols which were available when we start were optimized for special cases or products, they were not open nor complete. So we decided to create a new one that may solve in almost any situation we/you need to transport data.

Preconditions

OGRP should be easy to learn, easy to read and easy to handle. It should be extendible by own needs and it should cover standard cases unambiguous.

early decisions

We decided to use JSON as underlying format. To get rid of ambiguous interpretation we decided to use JSON Schema (http://json-schema.org/documentation.html) as definition format. The transport mechanism of data is not part of the OGRP definition. But the definition allows transportation with all common mechanisms as TCP/IP, Socket or File.

Details of the format

JSON just define single Objects. For a continuous stream of data, for example via plain TCP socket, a file, a serial line with a wrapping wire protocol or similar ways you just send the complete objects separated by a new line.

The main messages are defined and some of them can be extended by own properties. For the predefined message types and message parts we use JSON-Schema draft v4 as a format to describe the properties. With such definitions it is easy to check own or given data on OGRP conformance. Messages which are not defined can easily defined on your own fitting into the common scheme. Please create your own JSON-Schema file to make the extension usable by other groups.

Create own extensions

Please always extend the given schema file or create an own schema file. JSON Schema.

Create a test set of data and control the schema file with a validator-tool.

When creating or extending follow these rules:

Each message type which is used as a single OGRP object has at least 3 fields which are always needed.

"id" containing the unique messagetypename

"version" in Version 1 this is always OGRP1

"timestamp" that contains the time of sending the message

For building names and declarations try to use common phrases, try to reuse phrases.

Try to get short names and declarations but they has to be clear and readable. Details should be described in the "description" part. You must use the "description" fields in the schema file for all documentation. Please think about all possible properties and define theme. "type" and "description" is a must have. Always think about "minimum", "required" and "additional Properties" and declare them.

Usage

To build or parse an OGRP Message you depend on the message type which ist declared in the "id". Depending on that you has to fill or read the rest of the object.

Optional fields are allowed and has to be marked in the schema. Writer don’t must fill optional fields.

The general 'timestamp' holds the system time of the sending tool. It is in seconds since 1.1.1970 0 O’clock. It can be a float value. Please have a look at the Schemafiles.

Times based on GNSS Measurement are placed within the special properties of the message.

Example

```
{
   "channel_measurements": [
      {
         "carrier_phase": 0.5525,
         "channel_number": 3,
         "doppler": -2662.27,
         "gnss": "GPS",
         "locktime": 42.1,
         "pseudorange": 24985714.38,
         "satellite_id": 11,
         "signal_type": "L1CA",
         "snr": 46
      },
      {
         "carrier_phase": 0.783,
         "channel_number": 3,
         "doppler": -2662.245,
         "gnss": "GPS",
         "locktime": 16.8,
         "pseudorange": 24985714.38,
         "satellite_id": 11,
         "signal_type": "P1",
         "snr": 22.5
      }
  ],
   "id": "channel_measurements",
   "protocol": "OGRP1",
   "timestamp": 1420066248
}
```

3.2 OSRP

Why another Format

When working with OGRP we got problems with ambiguous data in OGRP objects, also we found a lot redundant information in it. We needed data formats for settings, configuration and commands. OGRP could only be used in linear systems. We wanted to build a complex system.

New preconditions

Instead of changing the “cld” format, we decided to start for a new format but used what we learned while working on and with OGRP.
So we did a new attempt in defining JSON-Schemas for a data structure. The new attempt we call OSRP.
The Preconditions were the same, but added:
- avoid ambigious data
- avoid redundant data
- Data should be mergable, to allow complex systems

Result

The result looks a lot alike, but in detail, there are heavy differences.

The OSRP format get more hierarchical and containes many details to manage all aspects of data around a GNSS Receiver. OSRP could be used for building complex data flows which OGRP couldn't. The OSR Board using OSRP for all datatransports excluding where explicit formats needed. OSRP is used to change settings. Even the configuration files for the components of the OSR board are handled via OSRP.

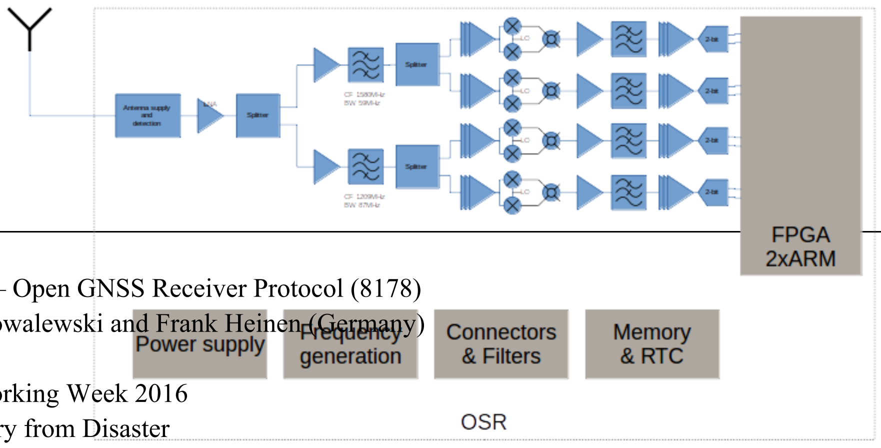
Example of an Configuration File:

```
{
"osrp":1,
"modul":"osrpsaver",
"timestamp":0,
"settings":[{
        "name":"savedir","value":"/data/test/","type":"text"
        },{
        "name":"Inputport","value":"5000",   "type":"integer","remark":"Portnumber to Stream"
        },{
        "name":"Prefix","value":"SAVED_","type":"text",   "remark":"Prefix for filenames"
        },{
        "name":"Savemodus",”value":"Record","type":"[Record,Dont record]" ,"remark":"Save
when possible or not"
        },{
        "name":"test", "value":"test", "type":"text"
        }
]}
```

4. OSR HARDWARE

The receiver is small with its 60x45mm printed circuit board. Yet OSR is a complete, programmable receiver. A highly adaptable analogue front end is teamed with a fully programmable system on a chip consisting of a large FPGA with in-package dual ARM processors.

Below we give an overview of the hardware design with information about the extents of programmability of the hardware.



4.1 Analogue Front End

An analogue front end consisting of signal conditioning and four super-heterodyne receivers is at the core of the Front End. Local Oscillator signals for down conversion are synthesized from a common 16.368MHz Temperature Compensated Crystal Oscillator (TCXO). Local Oscillator frequencies covering down conversion of all GNSS navigation signals can be synthesized with Phase Locked Loop (PLL) counter values under programmatic control. Each pair of mixers share one LO, so when down converting for example NavStar L1 and Glonass G1 in a signal pair sharing a common LO the G1 signal will be available in the upper Side Band and L1 in the lower Side Band. This scheme is possible thanks to image rejection mixers. Intermediary Frequency (IF) filtering is done in programmable, analogue Low Pass filters. The output of each signal path is available for further Digital Signal Processing as a 2-bit value. The Analogue to Digital Converters are sampled asynchronously.

The following functions are under programmatic control in the Analogue Front End:

- LO frequency synthesizers
- RF and IF Gain
- IF Low Pass filter
- Either USB or LSB in each signal path
- power supply On/Off

4.2 Programmable hardware

The output of the Analogue Front End consists of four sets of 2-bit ADC signal pairs. These signals are sampled by logics in a FPGA. We have chosen Zynq7030 from Xilinx in order to keep choices for receiver implementation wide open. Zynq7030 consist of a Kintex 7 FPGA (125K Logic Cells, 400 DSP Slices) grouped with two ARM Cortex-A9 processors to form a SoC in a package.

The FPGA is used for digital signal processing such as further filtering, frequency conversion, I/Q-demodulation, PRN correlation and all the bits an pieces of observables measurement.

The ARM processors would ususally run a Linux Operating System and be host for RTK and general user applications. The SoC is complemented with Flash and RAM memory.

The FPGA and ARM processors are fully programmable.

4.3 Connector Interface

The receiver's connectivity to a system motherboard consists of a 80-pole connector. The interfaces include:

- Antenna supply
- SDIO interface (pending developer's implementation in FPGA)
- Ethernet (pending developer's implementation in FPGA , magnetics on mother board)
- Reference frequency signals (pending developer's implementation in FPGA)
- Shut Down
- Reset
- PPS (pending developer's implementation in FPGA)
- UART#1 & #2, 3.3V, 10 mA (pending developer's implementation in FPGA)
- JTAG
- USB#1 & #2 (pending developer's implementation in FPGA)
- VBAT (for on-board RTC)
- VDC

4.4 Design considerations for openness

An open source GNSS receiver must be highly configurable and adaptable. For openness, development tools must be easily accessible and have a long life expectancy. Else the receiver will fall in obsolescence as support tools disappear. There is currently no viable open source development environment we know of for the programmable hardware we have chosen. The manufacturer XILINX allows free access to a design environment to use with some components, including the Zynq7030. The dual core ARM processors on board the FPGA run Linux with open source options for program development.

5. OSR BOARDS CODES

The digitized signal from each analogue frequency path in the Open Source Receiver feeds into a Field Programmable Gate Array (FPGA) where the signal can be compared with reference signals, further segregated into components, interpreted and analyzed. The FPGA is equipped with a set of internal processors and is also able to pass on tasks to an external Real Time Computer (that can operate under Linux) and can access non-volatile Flash memory as well as VRAM that can be accessed by multiple processes simultaneously. The hardware can therefore be programmed and the tasks distributed with great flexibility.

Parts of the Flash memory may hold code to run the FPGA and the processors and can be made to load as the OSR powers up. Users can write their own codes or may use codes supplied either by the OSR team or by third parties. These codes typically find and separate out signals from the various satellites, decode navigation messages and correlate ranging signals as well as carrier signals to extract code and carrier raw GNSS data. Modules will be available for the GPS civilian signals L1 C/A and L2C, as well as a high performance semi-codeless module to use the restricted L2 signal. There will be SBAS modules for WAAS, EGNOS etc.. GLONASS G1 and G2 signal modules as well as Galileo E1 signals will be similarly available with more modules expected to be released over time. Users are encouraged to develop and market third party modules.

5.1 RTK engine

The raw GNSS data may be fed to a Real Time Kinematic (RTK-) engine to generate NMEA messages. Like with the modules mentioned above, the user may develop their own code, they may elect to use a solution provided by the OSR team or any third party that may offer solutions. Our RTK engine uses GPS L1 and L2 in its entry level configuration and can be upgraded to run in combination with any or all of the signals supported in our tracking modules. This RTK engine has many options, it can be configured to be tolerant to base data latency and can be equipped with a VRS-module to dial and use Virtual Reference Station networks. It can also use advanced features to handle difficult conditions with frequent tracking slips and has a slip tolerant on the fly ambiguity resolution mode.

5.2 Other raw GNSS processing engines

A large number of engines besides RTK are conceivable. We offer ready modules to generate base station reference data streams in RTCM or other formats. Other engines can output Receiver Independent Exchange Format (RINEX-) data streams or store the data onboard the OSR for post-processing. We anticipate the release of optimized timing and other specialty engines.

5.3 navXWeb

Motivation

To avoid the necceasarity of a buildin display we decided to implemten a WEB-Interface that should be able to handle all kind of user interaction. That contains:
- Install Plugins and Extensions
- configure the system with all its components
- handle software updates
- configure settings like tracking, loging etc.
- the classical outputs like Skyplot, filehandling etc.

We also wanted the navXweb able to upload thirdparty-plugins

Realisation
The Core WEB Server is a simple WEB Server that handles CGI calls.
For developers of plugins there are documentations, tutorials and libraries to generate all what is necessary to create a navXweb compliant plugin. Up to creating an archive that can be uploaded by using the navXweb on a concrete OSR board.
When such a plugin is uploaded the navXweb automaticaly generates sites for the settings.
The internal parts of the OSR-Board uses the same mechanism as the plugins.

NavXweb uses state of the art HTML5 and CSS3 technologie to give a good userexperience and to make it possible to run it on all devices from mobile to big-screen.

The navXweb don't use much resources of the OSR-Board. It just create the representaiton of data. To avoid heavy resource-using in the OSR it uses HTML5 technique to let the browser render complexer grafic data. For example when displaying the Skymap.

The navXweb has a build-in user management and a build in filexplorer for a defined part of the internal Filesystem.

6. CONCLUSION

There is a high risk for all parties involved to develop this new receiver platform. Is it from the market ever wanted an open interface on a GNSS receiver to have? Does the hardware and development platform really wish from the Users? These questions remain open and can be answer only by the users and our customers.
As already Alexander Graham Bell said, "Don't keep forever on the public road, going only where others have gone, and following one after the other like a flock of sheep. Leave the beaten track occasionally and dive into the woods. Every time you do so you will be certain to find something that you have never seen before. Of course, it will be a little thing, but do not ignore it. Follow it up, explore all around it; one discovery will lead to another, and before you know it, you will have something worth thinking about to occupy your mind. All really big discoveries are the results of thought."

Datagrid, Gutec and navXperience are sure they do the right thing and the market needs the OSR – Open Source Receiver and the new Protocol OSRP.

REFERENCES

[1] T. Pany, Navigation Signal Processing for GNSS Software Receivers, Artech House, 2010.

[2] D. Dötterböck, S. Ko and B. Eissfeller, "Multi-Constellation RTK with a Software Receiver," in *Proceedings of the ION GNSS 2011*, Portland, Oregon, USA, 2011.

[3] "Network Time Protocol," Wikipedia, 5 March 2014. [Online]. Available: http://en.wikipedia.org/wiki/Network_Time_Protocol.

[4] J. Hahn and E. Powers, "A Report on GPS and Galileo Time Offset Coordination Efforts," in *Proceedings of TimeNav'07*, Geneva, Switzerland, 2007.

[5] P. Henkel, K. Giger und C. Günther, „Multi-Carrier Vector Phase Locked Loop,“ IEEE Signal Processing Society, München, 2009.

[6] B. Eissfeller, C. Tiberius, T. Pany and G. Heinrichs, " Real-Time Kinematic in the Light of GPS Modernisation and Galileo," *Galileo's World,* pp. 28-34, Fall 2002.

[7] J. Avila-Rodriguez, G. W. Hein, S. Wallner, J.-L. Issler, L. Ries, L. Lestarquit and A. de Latour, "The MBOC Modulation: The Final Touch to the Galileo Frequency and Signal Plan," in *Proceedings of ION GNSS 2007*, Fort WOrth, Texas, USA, 2007.

.

BIOGRAPHICAL NOTES

Dirk Kowalewski
He received his Dipl.-Ing. of Geodesy from the TFH Berlin in April 1991. He started his career as a specialist in CAD Software, total station from Zeiss and Topcon and a GNSS specialist for Trimble, Ashtech and Topcon. He worked in these jobs for different companies about 10 years. In March 2001 he started with his own business and he was the founder and the director of Geo.IT Systeme GmbH. The next milestone was the foundation of navXperience together with Franz-Hubert Schmitz. With navXperience we started the developing of our 3G+C GNSS antenna technology in the beginning of 2010 and we started with the sales in the end of 2010. From 2009 to 2011 we worked together with Frank Heinen at the research project MoDeSh. The development was a six dots of freedom Software for the measurement of movements and deformation on vessels. In 2013 the idea of the OSR receiver was born and since 2015 navXperince, Gutec and Datagrid works together on this project. Since 2012 he is a member of the working group AK 3 “Measurement method and Systems” DVW Germany.

Bo Gustafson
Bo Gustafson earned a MSc in Physics and Astronomy from the Lund University in Sweden in 1977 and a Ph D. in Astronomy and Astrophysics in 1981 also from Lund Uni. Gustafson was a Humboldt fellow at the MPI Kernfysik in Heidelberg, an Acad. of Science fellow at Uppsala Univ,

Sweden, and a QMW fellow at London Univ center for Mathematics, a NASA fellow at the State Univ. of NY Albany (SUNYA) and a NASA, NSF, US Airforce and US Army PI at the University of Florida (UF) where he became a Professor and Director of the UF's Laboratory for Astrophysics and authored over a hundred peer reviewed papers. Gustafson developed the UF Microwave analogue EM scattering laboratory that became widely known as the leading laboratory of its type worldwide. He and his team developed space instrumentation for NASA and ESA. Gustafson served in a range of NASA functions including Adviser for the Scientific utilization of the International Space Station, Science adviser to NASA for its Star Dust mission and science- or instrument- team member on most NASA and ESA deep space missions over more than three decades including the current Rosetta mission. In 1999 Gustafson funded DataGrid Inc to apply space technology and develop innovative geodetic grade GNSS hardware, firmware and software. DataGrid introduced Li-based batteries in GNSS instruments as early as 1999, pioneered the use of FPGA based GNSS receivers in the early 2000s to name a few of its innovations. DataGrid delivers receivers and RTK engines for both keyed and unkeyed receivers to the US DoD.

Per Gustafson

He received his M.Sc. In Physics from the University of Lund, Sweden in 1987. From 1987 to 1999 he is employed by SAAB, Aircraft Division in Linköping as Systems Engineer. He works on computer simulations of electromagnetic effects, transient ionizing radiation effects in electronics and in systems. During this time he assesses electronic systems from sub-contractors for radiated transient effects and deepens interest and understanding in electronic systems development. In 1998 he founds Gutec AB with the aim to facilitate industry's adoption of modern electronics design tools such as VHDL. Soon he gets involved in GNSS development and devotes his time to GNSS antenna and receiver design, which is his main interest since 2000. He his a IEEE member since 2001.

Frank Heinen

Dipl. -Ing. of Electroetechnics, Nachrichtentechnik at the Univerity of Duisburg in 1989.
He developed operating systems for acoustic discharge measurement hardware till 1991 and then went for 11 years to Ziegler Informatics and get a specialist in CAD-Sotfware development.
After that he went to IVC-AG and TRIGIS as a GIS-Software specialist. In December 2004 he get freelancer in the Fields of GEO, GIS and CAD. Since 2005 he worked with GeoIT. Together with GeoIT he developed an OS that enhance an existing TOPCON device and give it a WEB-Interface that maps the whole settings area and give a lot add-ons. With GeoIT he worked at the research project MODESH. Since the foundation of navXperience he also works with and later for navXperience. There he worked on the GOOSE Project.

CONTACTS

Dipl.-Ing. Dirk Kowalewski
navXperience GmbH
Querweg 20
13591 Berlin

GERMANY
Tel. +49 30 375 896 7-0
Fax + 49 30 375 896 7-1
Mail: dirk.kowalewski@navxperience.com
Webpage: www.navxperience.com

Dipl.-Ing. Frank Heinen
navXperience GmbH
Querweg 20
13591 Berlin
GERMANY
Tel. +49 30 375 896 7-0
Fax + 49 30 375 896 7-1
Mail: frank. heinen@navxperience.com
Webpage: www.navxperience.com

Prof. Dr.-Ing. Bo Gustafson
DataDrid Inc.
1022 NW 2nd Street
Gainesville, Florida 32601
USA
Phone: +1 352 371 7608
Mail: bo_gustafson@datagris-international.com
Webpage : www.datagrid-international.com

M.Sc. Per Gustafson
Gutec AB
Industrigatan 21c
234 35 LOMMA
SWEDEN
Phone: +46 40 416681
Mail: per@gutec.se
Webpage: www.gutec.se