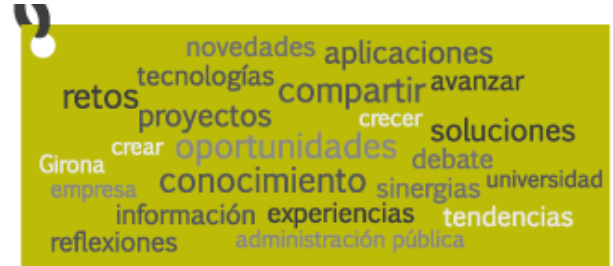


8as Jornadas SIG Libre

26, 27 y 28 marzo 2014

Girona



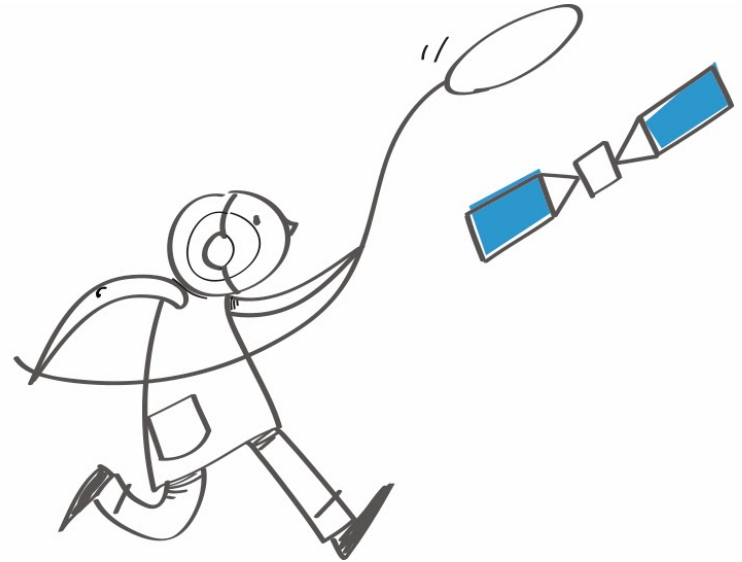
GNSS-SDR, un receptor definido por software abierto

Dr. Carles Fernández Prades

Centre Tecnològic de Telecomunicacions de Catalunya



Motivation



- Location has become an **embedded feature** in electronic devices (phones, digital cameras, portable gaming consoles).
- This massive deployment of GNSS receivers requires a high level of integration, low cost, small size and low power consumption. GPS integrated circuit (IC) manufacturers offer **single-chip solutions** easy to integrate in multi-function devices.
- This approach is very convenient for location based services and applications, since users and developers are interested in *using* the location information but not in *how* the position has been obtained.
- Example: Android's API. It provides a location package that contains classes with methods such as `getLatitude()`, `getLongitude()`, `getAltitude()`, `getSpeed()`, `getAccuracy()`, and so on.

- Location has become an **embedded feature** in electronic devices (phones, digital cameras, portable gaming consoles).
- This massive deployment of GNSS receivers requires a high level of integration, low cost, small size and low power consumption. GPS integrated circuit (IC) manufacturers offer **single-chip solutions** that integrate multi-function devices.
- This approach is very convenient for location-based services and applications, since users and developers are interested in *using* the location information but not in *how* the position has been obtained.
- Example: Android's API. It provides a location package that contains classes with methods such as `getLatitude()`, `getLongitude()`, `getAltitude()`, `getSpeed()`, `getAccuracy()` and so on.

What if you are interested in how the position is computed?

- Location has become an **embedded feature** in electronic devices (phones, digital cameras, portable gaming consoles).
- This massive deployment of GNSS receivers requires a high level of integration, low cost, small size and low power consumption. GPS integrated circuit (IC) manufacturers offer **single-chip solutions** easy to integrate in multi-function devices.
- This approach is very convenient for location-based services and applications, since users and developers are interested in *using* the location information but not in *how* the position has been obtained.
- Example: Android's API. It provides a location package that contains classes with methods such as `getLatitude()`, `getLongitude()`, `getAltitude()`, `getSpeed()`, `getAccuracy()` and so on.

What if you want to test your own algorithm?

What about the pros?

- Professional GNSS receivers feature dual-frequency reception and wireless connection for retrieving differential data (i.e., **corrections**) from reference stations in order to provide high-accuracy (cm-level) positioning.
- Receivers are **expensive** (5-10 k€), correction messages are **proprietary** (and with 1.5-5 k€ annual fees).
- You still do **not know** what your receiver is actually doing.

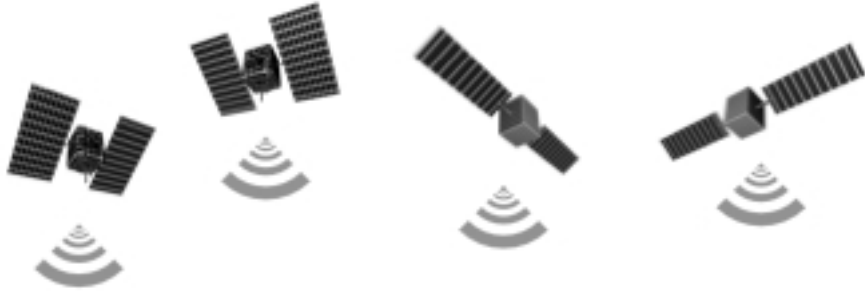
Wish list for a GNSS software receiver



Introducing GNSS SDR

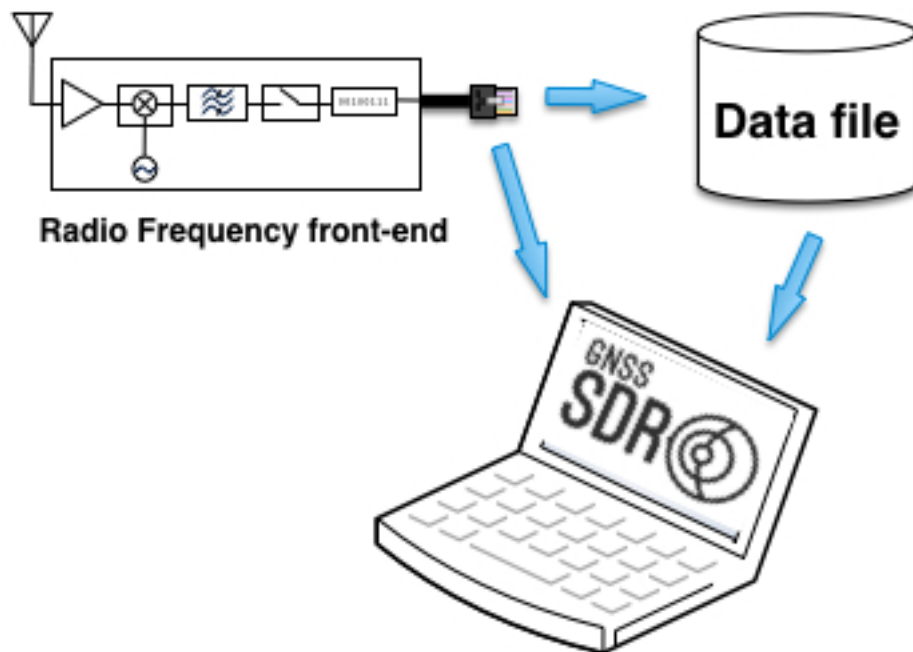
- The acronym stands for '**Global Navigation Satellite System – Software Defined Receiver**'.
- It is a tool for researchers, specially focused in signal processing.
- Flexible, fully configurable, easily extendible.
- Testbed for GNSS synchronization algorithms.
- Programmed in C++. Portable (currently builds in Linux and MacOS operating systems. The execution in embedded processors FPGA/ARM is ongoing work)
- **Open Source**: free as in *free beer and as in free speech*. Released under GPLv3.

The Software Defined Radio concept



Nice features w.r.t ASICs:

- Operational flexibility.
- Upgradability.
- Compatibility.
- Lower maintenance costs.
- Lower operational costs.



Unique features:

- Fair benchmarking of architectures and algorithms.
- Possibility of re-running the receiver over the same signal but with different architectures/algorithms/parameters.
- Allows a “GNSS receiver in the cloud”.
- **If open:** you know what your receiver is doing.

Current features

- Multiplatform (Linux and Mac OS X, 32 and 64 bit architectures).
- Multithreaded.
- Works with files and several RF front ends, including those compatible with the Universal Hardware Driver (UHD).
- Acquisition of **GPS L1 C/A**, **Galileo E1B** and **E1C** signals.
- SIMD-enabled for most popular processors.
- Implemented tracking loops: DLL + PLL, DLL + PLL/FLL, VEML.
- Connection to Matlab/Simulink via TCP for rapid prototyping and algorithm validation.
- Demodulation and decoding of the navigation message GPS NAV and Galileo INAV.
- Computation of PVT (Position - Velocity - Time) solution in real-time.
- Position solution exportable to KML files (can be opened by Google Earth and other similar tools) and to a serial port via NMEA.
- Generation of RINEX files (observables and navigation), v2.1 and v3. RTCM 3.2 is ongoing work.

Philosophy

Design principles

- Do not reinvent the wheel.
- Follow design patterns.
- Follow a clean coding style.
- Test-driven development approach.
- Follow standards.
- Try to produce *well-written* software.

Do not reinvent the wheel

- On the shoulder of giants:



GNU Radio is a software development toolkit that provides a framework to implement software radios. GPL v3.



FFTW is a C subroutine library for computing the discrete Fourier transform. GPL v2.



Boost is one of the most highly regarded and expertly designed set of libraries for the C++ programming language. Boost Software License.



Google C++ Testing Framework provides tools for writing C++ tests. BSD 3-Clause License.



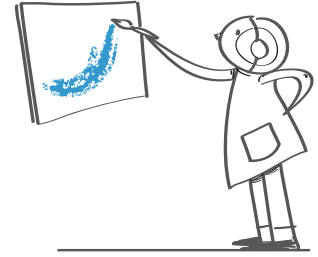
Armadillo is a C++ linear algebra library aiming towards a good balance between speed and ease of use. Mozilla Public License Version 2.0.

Design patterns

- Software design patterns are **descriptions of solutions to common software problems** arising in different contexts, capturing recurring structures and dynamics among software participants to facilitate reuse of successful, thoughtfully proven designs.
- They generally **codify expert knowledge of design strategies**, constraints and best practices. Following a pattern helps to resolve key design forces such as flexibility, extensibility, dependability, predictability, scalability, and efficiency.
- **They are not code recipes but generalized solutions** to commonly occurring problems.

- ✓ E. Gamma, R. Helm, R. Johnson, and J. Vlissides, **Design Patterns: Elements of Reusable Object–Oriented Software**, Addison Wesley, Upper Saddle River, NJ, 1995.
- ✓ C. Fernández-Prades, C. Avilés, L. Esteve, J. Arribas, and P. Closas, "**Design patterns for GNSS software receivers**," in *Proc. of the 5th ESA Workshop on Satellite Navigation Technologies (NAVITEC'2010)*, ESTEC, Noordwijk, The Netherlands, Dec. 2010.

Coding style

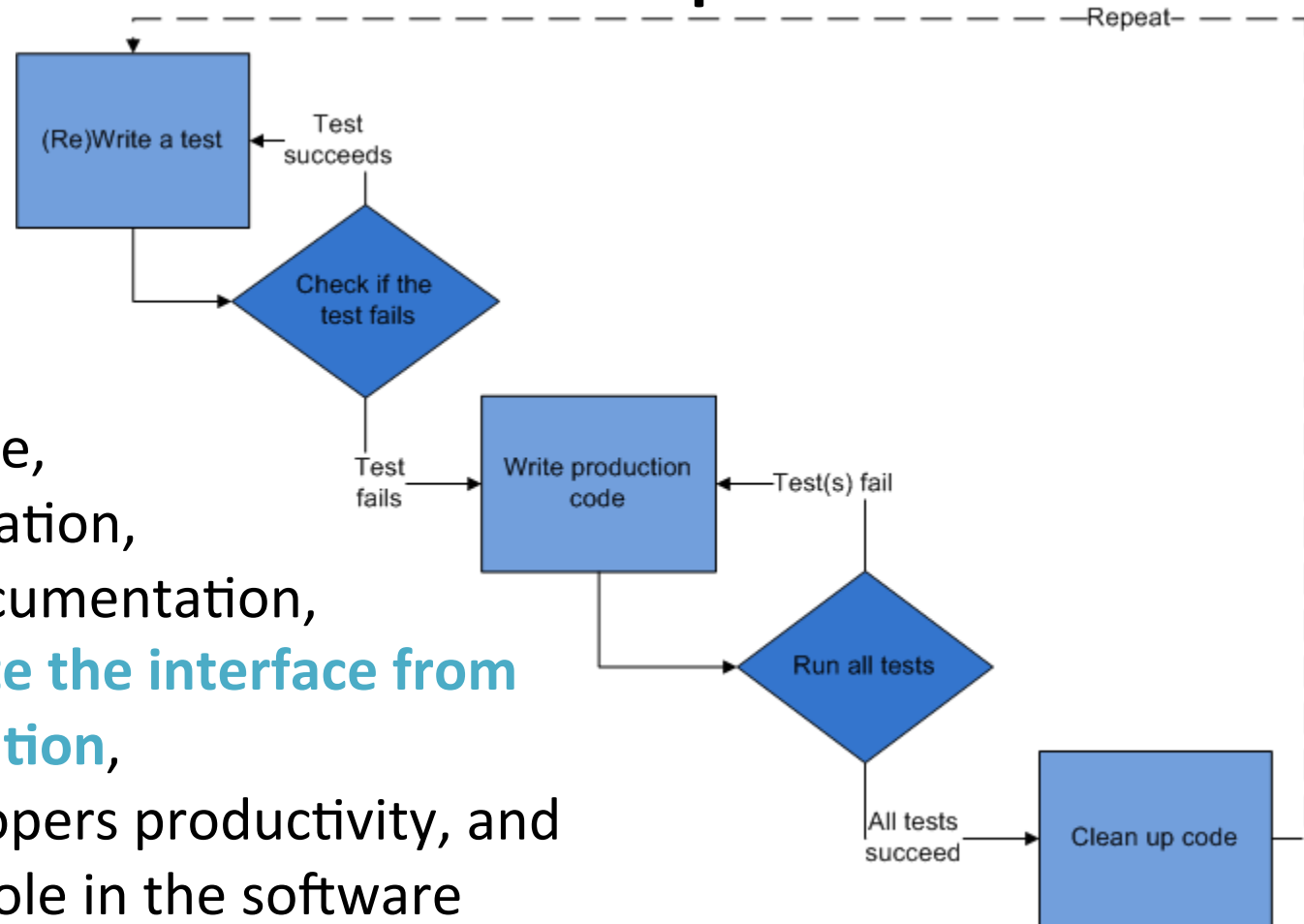


- Following programming guidelines and code conventions not only helps to avoid introducing errors, but cuts maintenance costs and favors effective code reuse.
- The following rules capture the most important aspects of coding style:
 - All should be as **understandable** as possible.
 - All should be as **readable** as possible, except when it would conflict with the previous rule.
 - All should be as **simple** as possible, except when it would conflict with the previous rules.
- Any violation to the guide is allowed if it enhances readability.

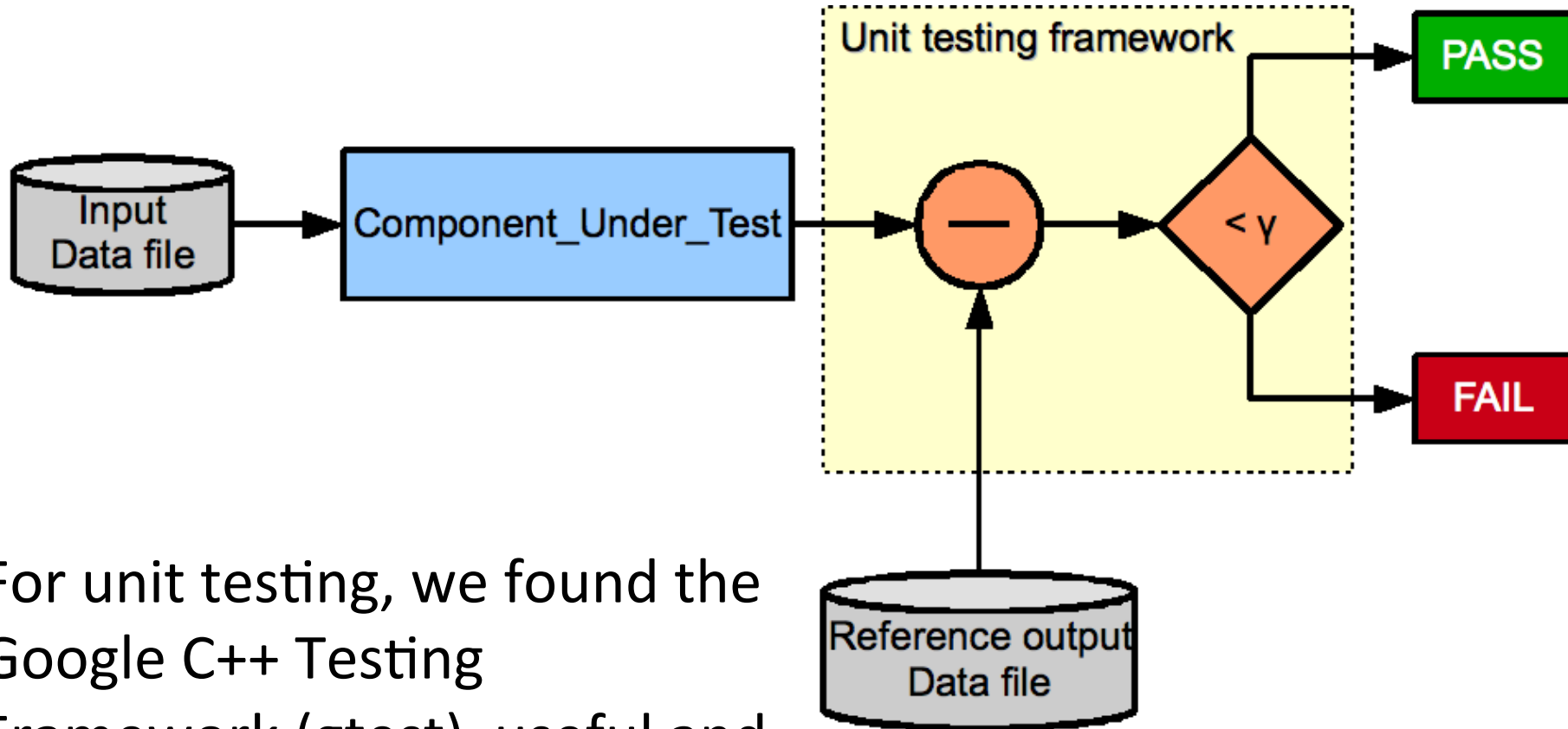
Test-driven development

Test-driven development:

- facilitates change,
- simplifies integration,
- automatizes documentation,
- helps to **separate the interface from the implementation**,
- increases developers productivity, and
- plays a central role in the software quality assurance process.

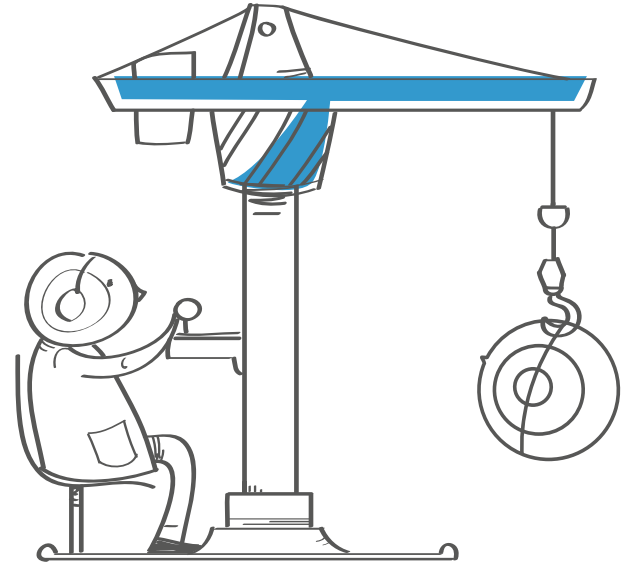


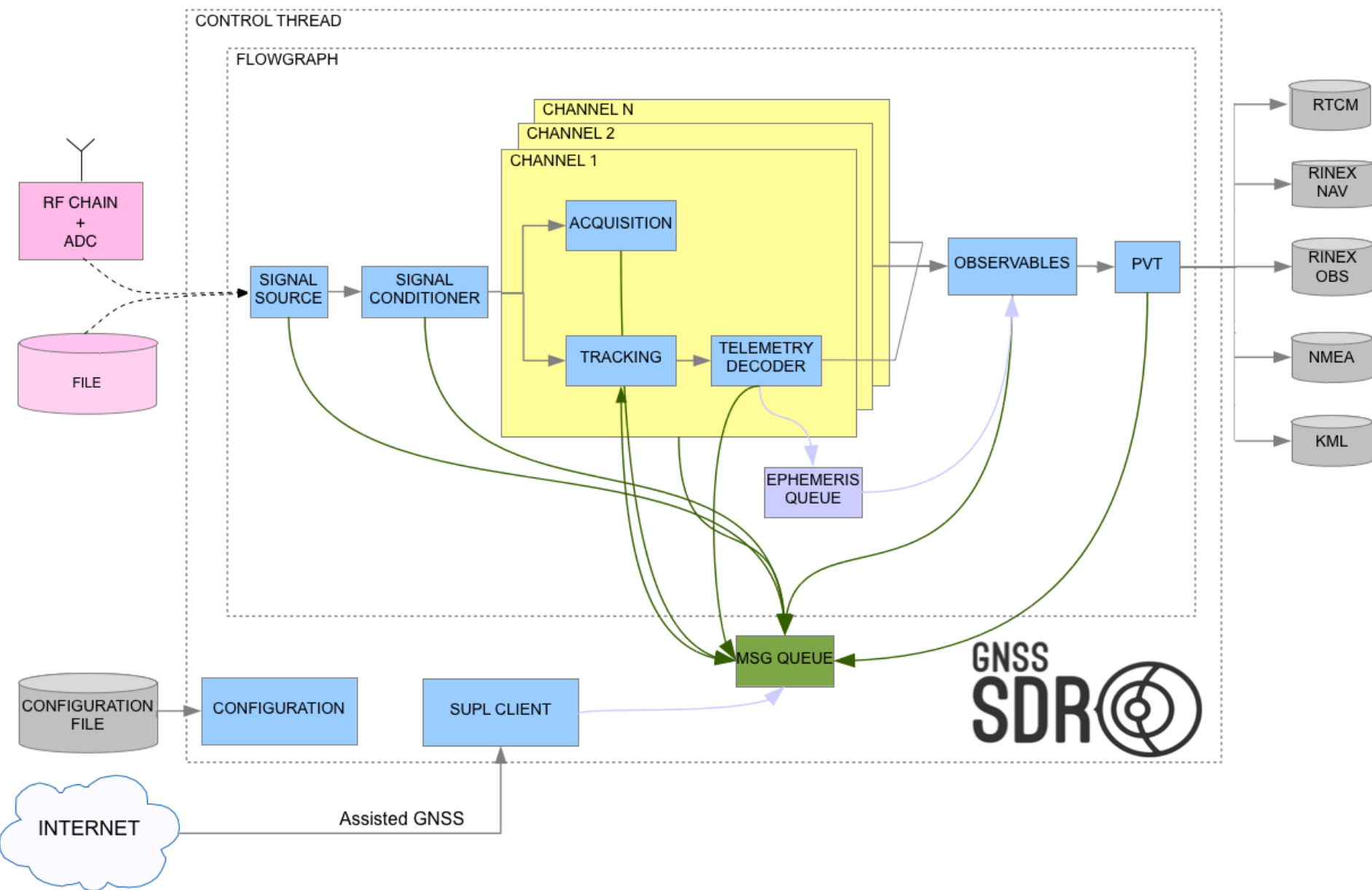
Automatic testing of new blocks



For unit testing, we found the Google C++ Testing Framework (gtest) useful and lightweight.

Software architecture

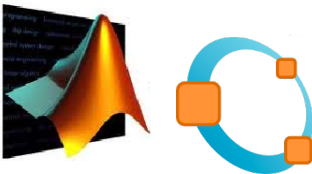




Instrumentation signal capture



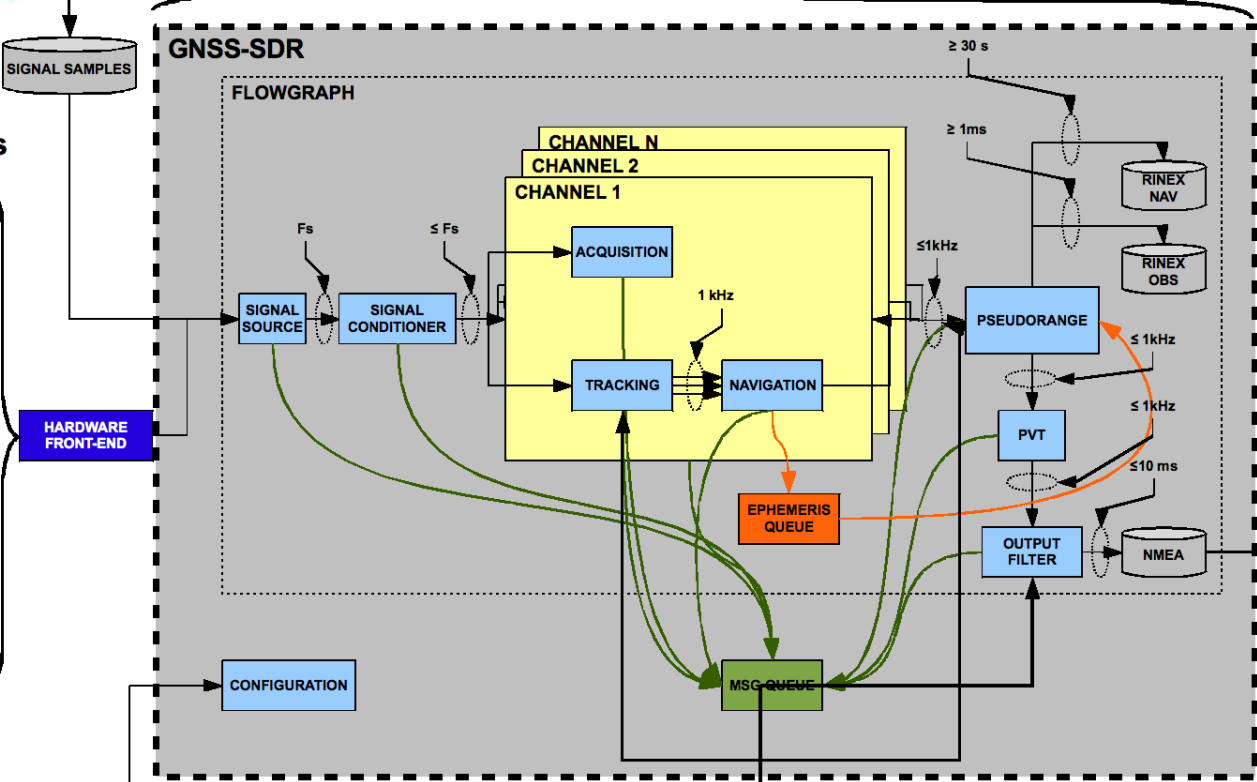
Intermediate signals analysis



GNSS external tools and scientific applications



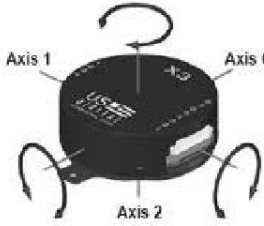
Real-time front-ends



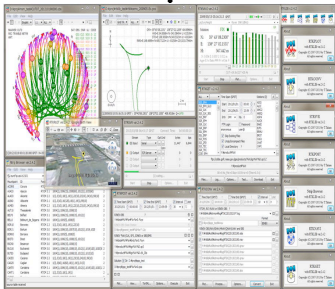
Detailed receiver configuration sets

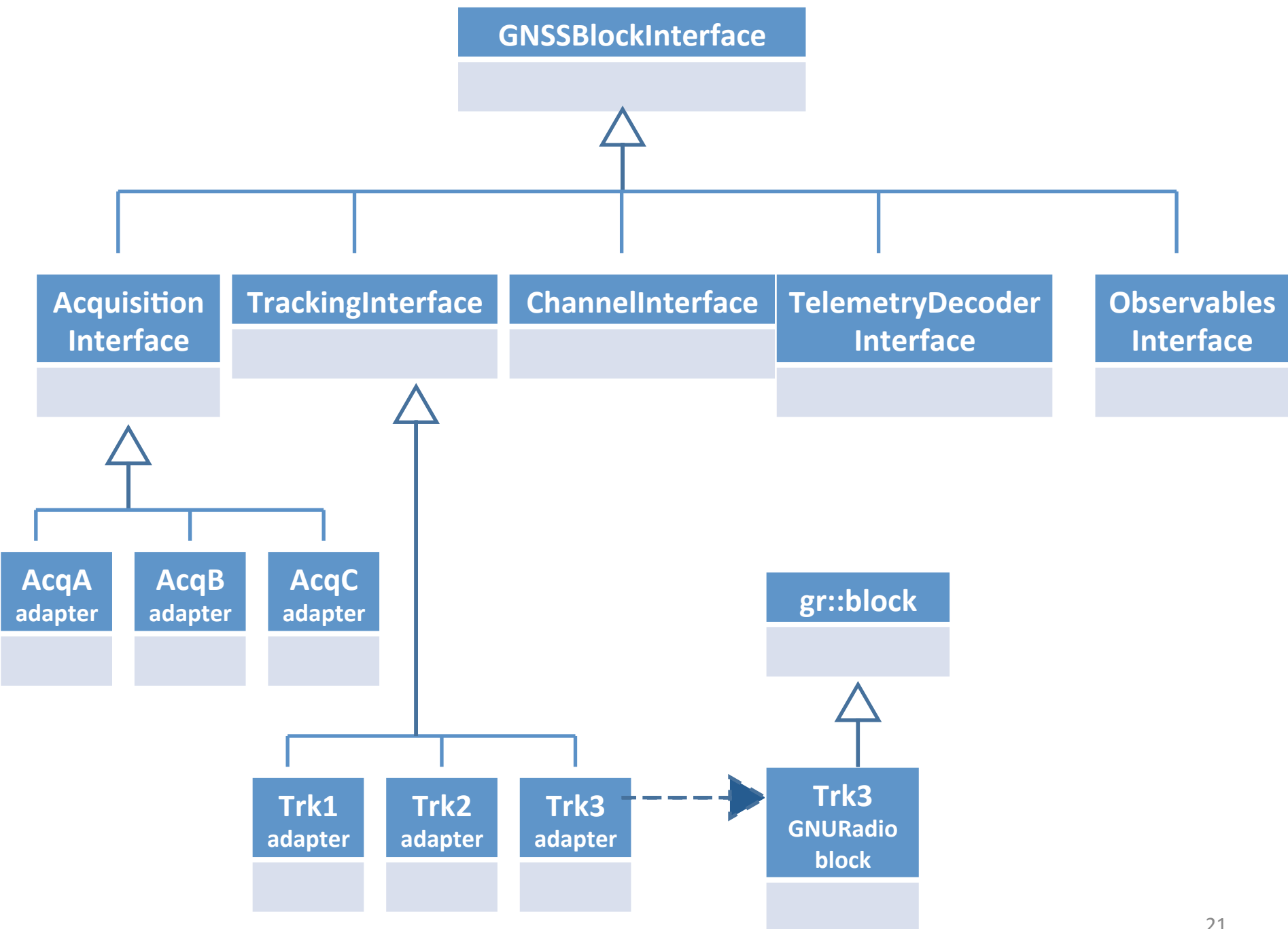


Inertial Measurement Unit



Positioning applications





Adding new algorithms

- I have designed an awesome acquisition algorithm and I want to test it in the framework of GNSS-SDR. What I have to do?
 - Create a GNU Radio block, derived from `gr::block`, implementing your algorithm (.cc and .h files) and putting it in [src/algorithms/acquisition/gnuradio_blocks/](#)
 - Create the adapter of such new block to `AcquisitionInterface` (.cc and .h files) and putting it in [src/algorithms/acquisition/adapters/](#)
 - Tell the system that a new block is available and include it in the 'production line': add the instantiation of your new block in [src/core/receiver/GNSSBlockFactory.cc](#)
 - Update the [CMakeLists.txt files](#) accordingly to tell the compiler that a new block exists.

Easy, examples are available

- Combined with certain DVB-T USB dongles, it constitutes the lowest cost solution available for GNSS software radio:



GPS antenna

+



Bias tee

+

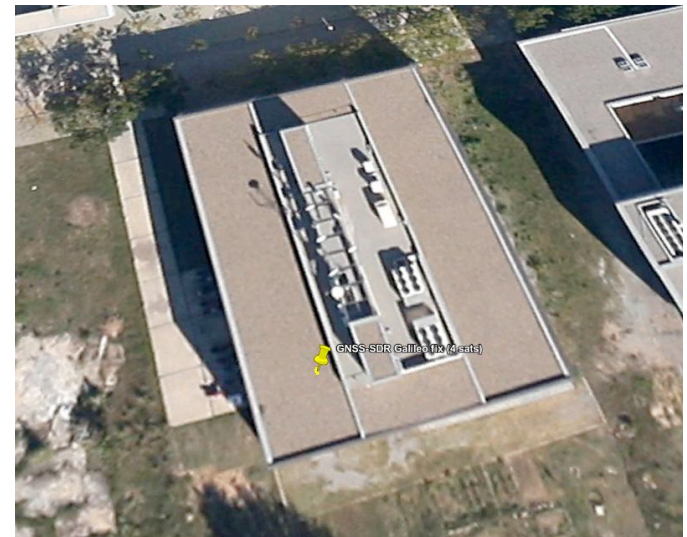
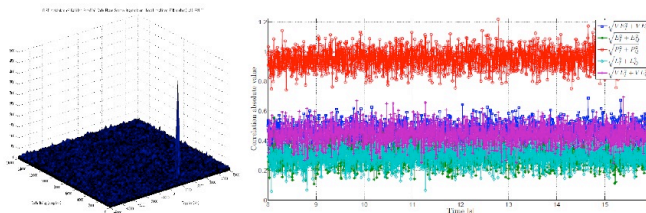


DVB-T USB dongle

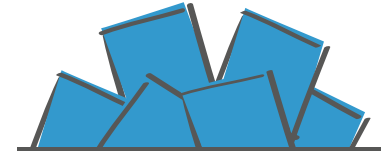
< 100 €

C. Fernández-Prades, J. Arribas, P. Closas, "**Turning a Television into a GNSS Receiver**", in Proc. of the ION GNSS+ 2013 Conference, Nashville, Tennessee (USA), 15-16 Sept. 2013.

- First open source solution to achieve Galileo-based position fixes (Nov. 2013).



Galileo Position Fix with Open Source Software Receiver Achieved, in GPS World, December 6, 2013. Available at: <http://gpsworld.com/galileo-position-fix-with-open-source-software-receiver-achieved/>







Development ecosystem

Infrastructure for project management, code development and efficient communication among users and developers is a key aspect in software projects.

- The **website** has been designed in terms of usability, functionality and extendability, ensuring an enjoyable and appealing user experience.
- **Public mailing list** for users and developers.
- For the source code, we used the service provided by SourceForge, which allows access to the code using a **revision control system**.
- **Documentation** is of paramount importance for users, developers, testers, software architects and students. We used Doxygen (HTML, LATEX, RTF or XML output formats).
- The project web page provides with detailed instructions about the installation, usage, **coding style** and general information about the software.



- The source code is freely accessible through Subversion, an open source version control system.
- Both the source code repository and the public mailing list are hosted at **SOURCEforge**
- The build process should be easily maintained and highly portable. We use  **CMake** Cross-platform Make, a tool that takes care about compiling the sources with the right options.
- Regarding the compiler, the GNU Compiler Collection and clang  (BSD license) can be used. 
- Logging is important for debugging and tracing purposes. In GNSS-SDR, logging is handled by the Logging Library for C++ (google-glog ) , a library that implements application-level logging.



Conclusions

- The proposed software receiver targets **multi-constellation / multi-frequency** architectures, and provides code and phase observables in standard formats, enabling RTK and Precise Point Positioning techniques.
- Goals: efficiency, modularity, interoperability and flexibility demanded by user domains that require **non-standard features**.
- The source code was released under the GNU General Public License (GPL), thus ensuring the **freedom of modifying, sharing, and using the code for any purpose**.
- This secures practical usability, inspection, and continuous improvement by the research community, allowing the discussion based on **tangible code** and the analysis of results obtained with **real signals**.
- It is also intended to be a framework for algorithm testing and an **educational tool**.

Publications

- C. Fernández-Prades, J. Arribas, P. Closas, "**Turning a Television into a GNSS Receiver**", in Proc. of the **ION GNSS+ 2013 Conference**, Nashville, Tennessee (USA), 15-16 Sept. 2013.
- L. Esteve, **Contribución al diseño de GNSS-SDR. Un Receptor GNSS de código abierto**, MS Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, July 2013.
- C. Fernández-Prades, J. Arribas, L. Esteve, D. Pubill, P. Closas, "**An Open Source Galileo E1 Software Receiver**", in Proc. of the **6th ESA Workshop on Satellite Navigation Technologies** (NAVITEC 2012), ESTEC, Noordwijk, The Netherlands, Dec. 2012.
- J. Arribas, **GNSS Array-based Acquisition: Theory and Implementation**, PhD Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, June 2012.
- C. Fernández-Prades, J. Arribas, P. Closas, C. Avilés, and L. Esteve, "**GNSS-SDR: an open source tool for researchers and developers**", in Proc. of the **ION GNSS 2011 Conference**, Portland, Oregon, Sept. 19-23, 2011.
- C. Fernández-Prades, C. Avilés, L. Esteve, J. Arribas, and P. Closas, "**Design patterns for GNSS software receivers**", in Proc. of the **5th ESA Workshop on Satellite Navigation Technologies** (NAVITEC'2010), ESTEC, Noordwijk, The Netherlands, Dec. 2010.

Available at <http://gnss-sdr.org/documentation/publications>

Thank you!



The screenshot shows the GNSS SDR website. At the top is the logo and a navigation bar with links: The Project, Documentation, Participate, Source Code, and Contact us. Below this is a large banner with the text "Global Navigation Satellite System" and "Software Defined Receiver" in blue, accompanied by an illustration of a telescope. The main content area is divided into three columns, each with an icon and text. The first column, titled "The process of developing a GNSS software receiver", features a crane icon and describes the open-source framework. The second column, titled "A research & educational tool", features a telescope icon and describes the source code's utility. The third column, titled "How to collaborate in the project", features a puzzle icon and describes the collaborative environment. At the bottom of these columns are links for "LEARN MORE" and "PARTICIPATE!". A dark footer at the very bottom contains three sections: "GET STARTED" (with a link to "How GNSS-SDR works"), "FEATURED ARTICLES" (with a link to "General overview"), and "GET IN TOUCH" (with links to "Distribution lists" and "Team").

GNSS SDR

The Project Documentation Participate Source Code Contact us

Global Navigation Satellite System

Software Defined Receiver

The process of developing a GNSS software receiver

GNSS-SDR is an open-source GNSS software receiver freely available to the research community. This project provides a common framework for GNSS signal processing which can operate in a variety of computer platforms. This tool is intended to foster collaboration, increase awareness, and reduce development costs in the field of GNSS receiver design and customized use of GNSS signals.

A research & educational tool

GNSS-SDR source code can be easily accessed and analyzed to understand GNSS receiver technology, being very useful in the context of education. It shows how signals are processed while passing through the different modules. For researchers, it is a perfect tool for testing, comparing and benchmarking new algorithms, easily integrating them in a complete receiver.

How to collaborate in the project

GNSS-SDR offers a set of professional, efficient and easy-to-use tools that allow collaborative research in an effective way. GNSS receiver design is a hot, active topic, where feedback among users and developers and infrastructure for code development are of paramount importance. Learn more about how YOU can collaborate in the project and benefit from the community work!

LEARN MORE

PARTICIPATE !

GET STARTED

FEATURED ARTICLES

GET IN TOUCH

How GNSS-SDR works

General overview

Distribution lists

Team

Please visit <http://gnss-sdr.org> and find:

- ✓ An overview about the project.
- ✓ Instructions for downloading and building the source code.
- ✓ Documentation and tutorials.
- ✓ Links to the mailing list.
- ✓ Suggestions for participation.