GNSS Software Defined Radio: History, Current Developments, and Standardization Efforts

Dennis Akos, *University of Colorado, Boulder*
Javier Arribas, *Centre Tecnològic de Telecomunicacions de Catalunya*
M. Zahidul H. Bhuiyan, *Finnish Geospatial Research Institute*
Pau Closas, *Northeastern University*
Fabio Dovis, *Politecnico di Torino, Italy*
Ignacio Fernandez-Hernandez, *European Commission*
Carles Fernández–Prades, *Centre Tecnològic de Telecomunicacions de Catalunya*
Sanjeev Gunawardena, *Air Force Institute of Technology*
Todd Humphreys, *The University of Texas at Austin*
Zaher M. Kassas, *The Ohio State University*
José A. López Salcedo, *Universitat Autònoma de Barcelona*
Mario Nicola, *LINKS foundation, Italy*
Thomas Pany, *Universität der Bundeswehr München*
Mark L. Psiaki, *Virginia Tech*
Alexander Rügamer, *Fraunhofer Institute for Integrated Circuits IIS, Germany*
Young-Jin Song, *Inha University, Incheon, South Korea*
Jong-Hoon Won, *Inha University, Incheon, South Korea*

BIOGRAPHY

Dennis Akos completed the Ph.D. degree in Electrical Engineering at Ohio University within the Avionics Engineering Center and is currently a faculty member with the Aerospace Engineering Sciences Department at the University of Colorado, Boulder.

Javier Arribas is a Senior Researcher in the Navigation & Positioning Research Unit at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Spain.

M. Zahidul H. Bhuiyan is a Research Professor at Finnish Geospatial Research Institute. His main research interests include multi-GNSS receiver development, PNT robustness and resilience, seamless positioning, etc.

Pau Closas is Associate Professor in Electrical and Computer Engineering at Northeastern University, Boston MA. His research interest include robust estimation and filtering for resilient GNSS receivers.

Fabio Dovis is a Full professor and leader of the Navigation Signal Analysis and Simulation group at the Electronics and Telecommunications Department of Politecnico di Torino, Italy. His research interests include advanced signal processing techniques for GNSS

Ignacio Fernandez-Hernandez works at the European Commission, DG DEFIS, as responsible for Galileo high accuracy and authentication. He is also an adjunct professor at KU Leuven.

Carles Fernández-Prades is a Senior Researcher and serves as Head of the Navigation & Positioning Research Unit at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Spain.

Sanjeev Gunawardena is a Research Associate Professor of Electrical Engineering with the Autonomy & Navigation Technology (ANT) Center at the Air Force Institute of Technology (AFIT). His research interests include RF design, digital systems design, high performance computing, SDR, and all aspects of satnav and associated signal processing.

Todd Humphreys is Professor of Aerospace Engineering at The University of Texas at Austin, where he leads the Radionavigation Laboratory and the Wireless Networking and Communications Group.

Zaher (Zak) Kassas is Professor of Electrical and Computer Engineering at The Ohio State University and Director of the U.S. Department of Transportation Center CARMEN (Center for Automated Vehicle Research with Multimodal AssurEd Navigation).

José A. López Salcedo is professor at Universitat Autònoma de Barcelona. His research interests lie in the field of statistical signal processing with application to snapshot GNSS receivers and future GNSS/PNT evolutions.

Mario Nicola is a researcher in the Space and Navigation Technologies research area at LINKSFoundation, Italy. His main activity is the implementation of algorithms for software radio GPS/Galileo receivers.

Thomas Pany is professor for satellite navigation at the Universität der Bundeswehr München and highly interested in GNSS signal processing and software radio technology.

Mark L. Psiaki is Professor and Crofton Faculty Chair of Aerospace and Ocean Engineering at Virginia Tech. His research interests are navigation, attitude and orbit determination, remote sensing, and general estimation and filtering.

Alexander Rügamer works in GNSS receiver development at the Fraunhofer Institute for Integrated Circuits IIS. His research interests are GNSS multi-band reception, integrated circuits, and immunity to interference.

Young-Jin Song is a Ph. D. student in the Department of Electrical and Computer Engineering at Inha University, South Korea. His research interests are GNSS software receiver, FPGA-based simulator/receiver, and weak signal processing.

Jong-Hoon Won is a professor and leader of Autonomous Navigation Laboratory in the Department of Electrical Engineering at Inha University, Incheon, South Korea. His research interests include GNSS signal design, receivers, navigation, target tracking systems, and self-driving cars.

ABSTRACT

Taking the work conducted by the Global Navigation Satellite System (GNSS) Software Defined Radio (SDR) working group during the last decade as a seed, this contribution summarizes for the first time the history of GNSS SDR development. It highlights selected SDR implementations and achievements that are available to the public or influenced the general SDR development. The relation to the standardization process of Intermediate Frequency (IF) sample data and metadata is discussed, and a recent update of the Institute of Navigation (ION) SDR standard is recapitulated. The work focuses on GNSS SDR implementations on general purpose processors and leaves aside developments conducted on Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuits (ASICs) platforms. Data collection systems (*i.e.*, front-ends) have always been of paramount importance for GNSS SDRs and are thus partly covered in this work. The work represents the knowledge of the authors but is not meant as a complete description of SDR history. Part of the authors plan to coordinate a more extensive work on this topic in the near future.

I. INTRODUCTION

Receiver development has always been an integral part of satellite navigation, ever since the early studies conducted for the American Global Positioning System (GPS). The very first receivers were huge devices, realizing the correlation of the received satellite signal with internally generated code and carrier replicas by a mixture of digital and analog electronics (Teunissen and Montenbruck, 2017). The advance of semiconductor technology soon after enabled signal processing on dedicated chips. This technology was of course complex to handle and mostly located within the US industry. Despite the success of GPS and its Russian counterpart Globalnaya Navigazionnaya Sputnikovaya Sistema (GLONASS), receiver internal technology was barely accessible to the broader research community for a long time, as it seemed to be impossible to realize GNSS signal processing on low-cost computers. Even in the year 1996 a key receiver design pioneer expressed skepticism that general purpose microprocessors were, or would ever be, a suitable platform for implementing an GNSS receiver (Kaplan, 1996).

The situation radically changed when the algorithms of a GPS receiver were first implemented as Matlab software on a desktop Personal Computer (PC) and estimates of Digital Signal Processor (DSP) processing power to run the algorithms in real-time were encouraging (Akos and Braasch, 1996; Akos, 1997). Soon after, real-time processing was demonstrated even on conventional PCs and the widespread use of software radio technology took off with exponential growth. Interestingly, software radio technology did not replace existing hardware receivers usually realized as one or more ASICs, but complemented these, allowing researchers to easily implement and test new algorithms or to develop highly specialized receivers with reasonable effort. Today, this is a well-established approach for military, scientific, and even commercial applications as described by Curran et al. (2018).

As different research groups developed their own software radios, they used different data collection systems to sample the GNSS signals. Whereas the data format of the digital GNSS signal streams is comparably easy to describe, the widespread use of software radio technology made it necessary to introduce a certain level of standardization, which was finally achieved by a group of researchers as documented by Gunawardena et al. (2021).

Technology further evolved, and not only new GNSS software radios appeared, but also some deficiencies of the standard (Clements et al., 2021). This triggered this work to recap the roots of GNSS SDR development in sect. II with a more detailed

focus on some receivers in sect. III and on front-end developments in sect. IV. Section V summarizes the history and the proposed update of the GNSS SDR standard.

II. GNSS SOFTWARE DEFINED RADIO HISTORY

The history of the Global Positioning System (GPS), or now more broadly known as the Global Navigation Satellite System (GNSS) Software Defined Radio (SDR), requires more than a bit of recollection which always can be fraught with inaccuracies, none of which are intentional, and corrections would always be welcome.

GNSS SDR traces its roots to Ohio University's Avionics Engineering Center around 1994. Professor Michael Braasch, a newly-minted faculty member of the Electrical and Computer Engineering department and already recognized as an expert in GNSS multipath, was interested in creating in a high-fidelity simulation of the internal signal processing within GPS and GLONASS receivers. Dennis Akos, a Ph.D. student in the Ohio University Electrical and Computer Engineering department/Avionics Engineering Center, was intrigued by the idea. Already harboring a keen interest in computer science and programming, Akos took on the simulation project at Braasch's request under the FAA/NASA Joint University Program. Meanwhile, publication of "The Software Radio Architecture" within the 1995 IEEE Communication Magazine Mitola (1995) fueled Akos's and Braasch's thinking that this "simulation" could instead be targeted toward an actual software radio implementation. The result was the first publication on GNSS SDR, which appeared in the proceedings of the 1996 ION Annual Meeting Akos and Braasch (1996).

Development of this initial simulation/implementation was significantly furthered through cooperation with Dr. James B. Y. Tsui of Wright Patterson Air Force Base. Well recognized as an expert in digital receivers, Tsui had recently taken an interest in satellite navigation. In 1995, two summer interns, Dennis Akos from Ohio University and Michael Stockmaster from The Ohio State University, worked under Tsui's guidance to develop a Matlab implementation of the signal processing required for basic GPS receiver operation. A digital oscilloscope was used to capture the initial IF data that were critical to developing and debugging those early algorithms. Akos was responsible for the lower-level signal processing (acquisition as well as code/carrier tracking), while Stockmaster implemented the navigation solution. The cumulative result was the first ever GPS SDR implementation. Although fully operational, it was "slow as molasses": processing 30 seconds of IF data required hours of computation time. Tsui published the first textbook on GPS SDR in 2000 (Tsui, 2000). A parallel contribution of this initial effort was the direct Radio Frequency (RF) sampling front-end, which garnered significant interest as well as pushed advances in analog-to-digital converter development (Akos et al., 1999).

After receiving his Ph.D. in 1997, Akos started his academic career as an Assistant Professor in the Systemteknik Department of Luleå University of Technology in Sweden, where he taught a course on computer architecture. It was here that GPS SDR first achieved real time operation. For a class project, Akos provided a Matlab-based GPS SDR and challenged a group of students to "get it to run as fast as possible" subject to the requirement that the complex accumulation products for each channel were within 10% of those produced by the original Matlab-based GPS SDR. It was in 1999 that the first "real time" operation was possible, processing 60 seconds of IF data in 55 seconds. This was a notable achievement at the time as renowned GPS expert Philip Ward, who was responsible for some of the first GPS receivers, wrote: "the integrate and dump accumulators provide filtering and resampling at the processor baseband input rate, which is around 200 Hz (...). The 200 Hz is well within the interrupt servicing rate of modern high-speed microprocessors, but the 5- or 50 MHz rates would not be manageable" in (Kaplan, 1996). This real time implementation effort was led by student Per-Ludvig Normark and led to the results published by Akos et al. (2001).

In the meantime, Kai Borre, a geodesy professor from Aalborg University, had also developed in the mid-late 1990s Matlab code for GPS receivers. Borre's code focused on the navigation block and including functions for e.g. conversion of coordinates and time references, satellite position determination and atmospheric corrections. The joint efforts of Akos, Borre and others would later lead to the well-known book (Borre et al., 2007), a main reference for GNSS SDR over the next years, and the related SoftGPS Matlab receiver.

Upon graduation, Normark continued his GNSS receiver development with the GPS Laboratory at Stanford University and then returned home to Sweden where he co-founded NordNav Technologies, which developed the first Galileo SDR, and helped establish the architecture, with Cambridge Silicon Radio (CSR) out of Cambridge, UK, to push GNSS to a price point acceptable to the mobile phone adoption. CSR, at the time a dominant supplier of Bluetooth hardware to the mobile phone market, acquired NordNav in 2006, and redesigned the 2.4 GHz radio to multiplex to the 1575.42 MHz GPS L1 band, exploiting the fact that most Bluetooth applications have a relatively low duty cycle. This approach, coupled with the real time software GPS implementation, provided a near-zero-added-cost GPS receiver.

There has been numerous contributions to GNSS SDR development since these early years, many of which are from the co-authors of this paper. Some selected developments are outlined in the next section. The authors of this paper are aware that many other important contributions are missing herein and agreed at the ION GNSS+ 2022 conference to extend this endeavor to a larger format, like a special issue of NAVIGATION, thereby maximizing the inclusion of all relevant contributors.

III. CURRENT STATUS OF GNSS SOFTWARE DEFINED RADIOS

In June 2022, a quick internet search did not reveal a comprehensive listing of all GNSS SDRs and Wikipedia (2022) lists six entries, which is far below the number of receivers known by the authors, even if the following criterion is applied to limit the scope: *a GNSS SDR (or software receiver) is defined as a piece of software running on a general purpose computer converting samples of a received GNSS signal into a Position Velocity and Time (PVT) estimate.* It is clearly understood that a front-end including Analog-to-Digital Conversion (ADC) is required to sample the received signal, but other than that no further functionality is allowed to be realized via hardware. With this definition, three categories of software receivers can be introduced:

- **real-time receivers:** monolithic or modular software packages written in an efficient low-level programming language (like C or C++) typically optimized for run-time efficiency and stability
- **teaching/research tools:** software packages written in a high level programming language like Python or Matlab optimized for code readability and flexibility
- **snapshot receivers:** receivers optimized for very short batches of signal samples

Furthermore, the software package shall allow some configuration flexibility and (at least theoretically) support the ION SDR standard. The following subsections introduce a few selected developments, emphasizing the rationale behind design choices and current status. Section III.1 describes the work of Psiaki, Ledvina, and Humphreys and their efforts in real-time processing on DSPs with the bit-wise approach proving to be highly successful even for space applications. Sect. III.2 covers work of Pany/others in their efforts with multiconstellation/multifrequency GNSS. Sect. III.3 and Sect. III.4 cover the efforts of Borre and others in a readable open source Matlab GPS SDR started in (Borre et al., 2007), with the most recent GNSS update reported in Borre et al. (2022). Akos has also continued this academic development of a suite of open source GNSS SDRs (Bernabeu et al., 2021). The widely used open-source receiver GNSS-SDR is described in III.5. The AUTONAV receiver used to support the development of Korean Positioning System (KPS) is discussed in sect. III.6 and PyChips (cf. sect. III.7) is the basis for tutorial classes of the ION. The snapshot approach is outlined in sect. III.8 and sect. III.9 discusses a SDR used e.g. to the authentication schemes, reflectometry or to assess the influence of non-standard GNSS transmissions. Section III.10 extends the scope of SDR to non-GNSS signals.

Whereas at the beginning of the GNSS SDR development the different receivers were linked to specific persons or research institutes, today often different receivers, tools or code bases are used at the same institute. On the other hand, code bases first developed by a single institute spread into different institutes. For example, the developments of Borre et al. (2007) forked into several branches [e.g. (FGI, 2022; Bernabeu et al., 2021; Zhang, 2022)], as discussed in sect. III.3 and sect. III.4.

Many key contributions to GNSS in general have been achieved with SDRs. Other SDR key publications cover implementations or algorithms which were in principle already known, but have been implemented for the first time with SDR technology. Some of those contributions are listed in the following subsections. Apart from them it is worthwhile to mention that the first real-time GNSS/INS integration with an SDR was achieved by Gunawardena et al. (2004) and one of the first GNSS SDR implementation on a Graphics Processing Unit (GPU) was reported in Hobiger et al. (2010). Furthermore, GNSS SDRs are known to achieve the highest possible sensitivity as different integration schemes or data wipe-off procedures can be performed in post-processing. This enables very long coherent integration times which is beneficial for sensitivity or multipath mitigation as reported in sect. III.8. Characterization of the GPS transmit antenna pattern with a 30-second long coherent integration resulting in 0 dBHz sensitivity is discussed by Donaldson et al. (2020). The same sensitivity was achieved by 300 noncoherent integrations each 1 second long for the purpose of indoor timing by iPosi Inc. (2015). On the other hand, graphical programming languages, such as LabVIEW and Simulink, are attractive choices for implementing SDRs, due to their flexibility, modularity, and upgradability. Moreover, since SDRs are conceptualized as block diagrams, graphical programming languages enable a one-to-one correspondence between the architectural conceptualization and software implementation (Hamza et al., 2009; Kassas et al., 2013).

The scope of SDRs was first extended to non-GNSS signals by McEllroy et al. (2006). SDRs became the implementation of choice in numerous studies aimed at exploiting signals of opportunity (SOPs) for navigation purposes (Kassas et al., 2017; Diouf et al., 2019, 2021), such as (i) cellular 3G code-division multiple-access (CDMA) (Pesyna et al., 2011; Yang and Soloviev, 2018; Khalife et al., 2018), 4G long-term evolution (LTE) (del Peral-Rosado et al., 2017; Shamaei et al., 2018; Shamaei and Kassas, 2018; Ikhtiari, 2019; Kang et al., 2019; Wang et al., 2022; Yang et al., 2022), and 5G new radio (NR) (Shamaei and Kassas, 2021b; Santana et al., 2021; Fokin and Volgushev, 2022; Abdallah and Kassas, 2022; Lapin et al., 2022; Tang and Peng, 2022; Del Peral-Rosado et al., 2022); (ii) AM/FM radio (McEllroy, 2006; Chen et al., 2020; Souli et al., 2021; Psiaki and Slosman, 2022); (iii) digital television (Souli et al., 2020; Yang and Soloviev, 2020; Souli et al., 2022); and (iv) Low Earth Orbit (LEO) satellites (Farhangian and Landry, 2020; Orabi et al., 2021; Farhangian et al., 2021; Pinell, 2021; Nardin et al., 2021; Zhao et al., 2022) .

Due to the enhanced analysis possibilities of GNSS SDR they proved to be very useful to understand ionospheric scintillation and

the first dedicated SDRs are described by Peng and Morton (2011); O'Hanlon et al. (2011). The authors used a general purpose front-end being reconfigurable for multi-GNSS multi-band signals, and a custom dual-frequency front-end, respectively. The first system further evolved into an intelligent, scintillation event-driven data collection as reported by Morton et al. (2015).

Commercialization of academic SDR developments is partly discussed in the following sections. Also a major receiver manufacturer provides GNSS SDRs, first starting with a timing receiver (Trimble Inc., 2005) and then moving to a flexible narrow-band receiver (Trimble Inc., 2017). Wide-band signals were later added with some signal processing now done on an FPGA as reported in PR Newswire (2021). The most recent commercial activity can be found in LocusLock (2022) and builds upon the software described in the following section.

1. Bit-Wise Parallelism for the High-Bandwidth Digital Signal Processing Receiver Operations

The original real-time GNSS software radio work by Akos (1997) inspired an effort within the Cornell GPS group. Psiaki had been working with non-real-time software GNSS signal processing in MATLAB for about 2 years when he started to wonder whether the slow MATLAB operations could be translated to run in real-time on a general desktop workstation. The bottleneck in GNSS digital signal processing occurs when doing the operations that initially process the high-frequency RF front-end samples. RF front-ends typically sample at 4 MHz or faster. A 12 channel receiver would have to perform on the order of 400 million operations per second or more in order do all of the needed signal processing. Psiaki conceived the concept of bit-wise parallel processing as a means of addressing this challenge. He recruited then-Ph.D. candidate Brent Ledvina to make an attempt at implementing these ideas in the C programming language on a Real-Time Linux desktop workstation. Ledvina succeeded in developing a 12-channel real-time L1 C/A-code receiver after about 6 months effort. The first publication about this receiver was (Ledvina et al., 2003).

The main concept of bit-wise parallelism is to work efficiently with RF front-end data that have a low number of quantization bits. If an RF front-end produces a 1-bit digital output stream, then 32 successive sign-bit samples can be stored in a single 32-bit unsigned integer word on a general-purpose processor. Thirty-two successive output samples of a 2-bit RF front-end can be stored in two 32-bit words, one containing the successive sign bits and the other containing the successive magnitude bits. Each channel of the software receiver generates a 1-bit or a 2-bit representation of 32 successive samples of its IF carrier replica, both in-phase and quadrature, and the successive samples are stored in parallel in 32-bit unsigned integer words. Similarly, it generates a 1-bit representation of 32 successive samples of its prompt Pseudo-Random Noise (PRN) code replica and stores them in parallel in a single 32-bit unsigned integer word. It also generates an early-minus-late PRN code replica that requires 1.5 bits per sample, which takes up two 32-bit unsigned integer words to store 32 samples. These replica signals can be generated very efficiently by using pre-tabulated 32-bit words. The software receiver then performs a series of bit-wise AND, OR, XOR, and similar operations that have the effect of performing PRN code mixing and IF-to-baseband carrier mixing. The outputs of the mixing operations are contained in a small number of 32-bit words, the number of which depends on the number of bits in each RF front-end output sample and the number of bits in the IF carrier replicas.

The final operation is accumulation of the results in the 32-bit words. This involves sets of bit-wise Boolean operations, as per Ledvina et al. (2003), followed by summation of the number of 1-bits in the resulting 32-bit unsigned integer words. The bit summation operations proved to be a challenge in terms of minimizing execution time. Ledvina solved this problem by using a pre-computed 1-dimensional data table whose input was the unsigned integer and whose output was the number of 1-bits. In order to keep the table size reasonable, it only counted the bits in a 16-bit unsigned integer word. The original receiver's 32-bit words were split in half, 2 table look-ups were performed, and the results summed in order to count all the 1-bits. The original algorithms are defined by Ledvina et al. (2003), Ledvina et al. (2004a) and Ledvina et al. (2006b).

When using very long PRN codes, such as the L2C CL code, the original method's whole-period PRN code tables of the proper 32-bit words at various code phases became impractically large. Therefore, a new method was developed for long PRN codes. It tabulates 32-bit words of short generic PRN code chip sequences, with all possible combinations of a short sequences of chips considered at various PRN code offsets relative to the start of the samples of the 32-bit word. Those methods are described by Psiaki (2006) and by Ledvina et al. (2007). This technique proved invaluable for dealing with long codes.

After getting the basic algorithms working in real-time, the Cornell group show-cased the efficacy of real-time GNSS software radio by using the techniques to develop a dual-frequency L1 C/A and L2C receiver (Ledvina et al., 2004b) and a GPS/Galileo L1 civilian receiver (Ledvina et al., 2006a). These real-time software GNSS receivers each required only several person-days to develop them from the original L1 C/A code receiver. Of course, the L1/L2 receiver required a new dual-frequency RF front-end. The GPS/Galileo receiver required knowledge of the civilian Galileo E1 PRN codes, which had not been published at that time. That led to a supporting effort which successfully deduced the Galileo GIOVE-A E1 PRN codes by listening to them and doing a lot of signal processing in order to pull the chips out of the noise (Psiaki et al., 2006).

The next development was to re-implement the bit-wise parallel code for embedded (low-power, low-cost) processing. Initially targeting a Texas Instruments DSP, this work was accomplished in 2006 by then-Ph.D. candidate Todd Humphreys (Humphreys et al., 2006). Later, as a professor at The University of Texas at Austin, Humphreys and his students—notably Jahshan Bhatti

and Matthew Murrian—undertook a sequence of significant expansions and improvements to this receiver. Called GRID/PpRx, the C++-based UT Austin receiver is by now a highly-optimized science-grade multicore GNSS SDR (Humphreys et al., 2009; Nichols et al., 2022). It was the first GNSS SDR to be adapted for spoofing (Humphreys et al., 2008), the first GNSS SDR to operate in space (Lightsey et al., 2014), the first receiver of any kind to show that centimeter-accurate GNSS positioning is possible with a smartphone antenna (Pesyna et al., 2014), the first receiver to be used to locate terrestrial sources of GNSS interference from low-Earth orbit (Murrian et al., 2021), and is the basis of the current state-of-the-art in urban RTK positioning (Humphreys et al., 2020; Yoder and Humphreys, 2022). As detailed in (Nichols et al., 2022), GRID/PpRx has also reaffirmed the commercial viability of GNSS SDR in widespread low-cost applications: it was recently licensed by a major aerospace company for use across all company operations, including in the thousands of satellites of the company's broadband Internet mega-constellation.

A processor that can operate on wider segments of data, up to 512 bits for current single instruction, multiple data (Single Instruction Multiple Data (SIMD)) instructions, gains substantial additional signal processing speed increases (Nichols et al., 2022). Note, however, that the speed increase factors over brute-force integer calculations are typically not as high as the number of bits per word. That is, the techniques do not speed up the operations by a factor of 32 when processing 32 samples in parallel by using 32-bit words to represent 32 samples. For a 2-bit RF front-end and a 32-bit processor, the speed-up factor might be only 4 because the bit-wise parallel approach requires multiple operations due to, say, a simple multiplication of one time series by another. If one doubles the number of bits per word, however, then the speed tends to double. A particularly helpful feature of some recent processor designs is their inclusion of a hardwired command to count all the 1 bits in a word. This "popcount" intrinsic obviates the table look-ups that counted 1-bits in the original bitwise parallel design. If the number of bits increases in the RF front-end samples and/or the IF carrier replicas, however, then the bit-wise parallel method of signal processing slows down. Signals represented by 3 or 4 bits might cause the processing speed gains of bit-wise parallel algorithms to be limited or even non-existent.

2. Multi Sensor Navigation Analysis Tool

The Multi Sensor Navigation Analysis Tool (MuSNAT) is an object-oriented but monolithic C++ software receiver maintained by the Universität der Bundeswehr München (UniBwM) and has been first mentioned in its present form by Pany et al. (2019). It started as an operational real-time receiver development, but currently it mostly serves to develop and demonstrate innovative signal processing and navigation algorithms. Furthermore, it is used for teaching. It is freely available as executable for academic purposes from (UniBwM, 2022). Its main characteristics can be found in Tab. 1. In contrast to the bit-wise approach of sect. III.1 (that allows to design very power-efficient implementations), the design idea of MuSNAT and its predecessors was to realize a high-end receiver running on powerful PCs or workstations. The bit-wise approach was replaced by using SIMD instructions of Intel/AMD Central Processing Units (CPUs). This allows to represent samples as 8-bit or 16-bit values and SIMD instructions like AVX-512 currently allow processing of registers of up to 512 bit (i.e. 32 16-bit samples) in parallel.

The GNSS software receiver developments started at UniBwM in 2002 after it became clear that the software radio approach discovered by D. Akos would provide useful insights into GNSS receiver technology and thus will be indirectly very helpful to design and build the Galileo navigation satellite system. The first software receiver at UniBwM was GPS L1 C/A only and was realized as a Matlab/Simulink project working in post-processing. To sample the GNSS signals a commercial ADC with a Peripheral Component Interconnect Express (PCIe) connector from NI was used (PXI 5112) that was connected either to a low-bandwidth GPS L1 C/A code front-end based on the Plessey GP 2010 RF chip set and later on to one GPS L1/L2 high-bandwidth front-end, which was specifically developed by Fraunhofer IIS (Pany et al., 2004b). Soon after, the software to communicate with the ADC (written in C++ making use of the Microsoft Foundation classes) was upgraded to a full GPS L1 C/A plus L2CS (L2 medium length code was supported only, not the long code) receiver. A detailed analysis published by Pany et al. (2003) revealed that not only the SIMD instruction set was important for the real-time capability but also the size and structure of the CPU caches. Memory bandwidth is one of the key issues when representing samples by multiple bits. One of the first achievements with this receiver was the demonstration of vector tracking (Pany et al., 2005).

Based on those results, funding to support a group of five researches over three years was secured. This allowed starting a new software receiver project, this time making full use of C++ features for object oriented development, and development of a Graphical User Interface (GUI) connected to the processing core via a clearly defined interface also allowing to run the core without GUI. The overarching development goal at that time was to realize a high-quality multi-GNSS multi-frequency receiver on a desktop PC or powerful laptop that could potentially be operated on a continuous basis to replace the (at that time) rather inflexible and expensive commercial GNSS receivers at Continuously Operating Reference Stations (CORSs). A concise overview of the development during those years was written by Stöber et al. (2010) and shows the improvements compared to the start of the project layed down by Pany et al. (2004a).

A loose cooperation with IFEN GmbH was initiated that eventually resulted in the SX3 receiver (IFEN GmbH, 2022). IFEN used the processing core as initial basis, improved the core, replaced the GUI, and developed new dedicated front-ends. The C++ code was further optimized to support more channels at higher bandwidth and almost instantaneous high-sensitivity acquisition

Table 1: Summary of MuSNAT

MuSNAT		
Feature	Solution	Remark
Operating System	Windows 10/11	Compiles as GUI or as command line version
Programming environment	C++ and CUDA	Microsoft Visual Studio and vcpkg
IF sample file input source	ION SDR standard and proprietary file readers	proprietary readers faster than ION SDR reader
real-time sample input	yes, via TCP/IP	server available via LabView for selected NI USRPs
additional sensors	LiDAR, IMU	LiDAR uses PCL format, IMU proprietary ASCII format, video formats supported but not yet used
Supported GNSS	GPS, Galileo, BeiDou, GLONASS, SBAS, OFDM (LTE, 5G)	nearly all open spreading codes available and at least for each system one navigation message decoder
acquisition	optimized Fast Fourier Transform (FFT) method	CPU and GPU supported
tracking run-time optimization	dot-product from Intel Performance Primitives	computational performance mostly limited by memory bus width
further features	multi-antenna, signal-generator, primary-secondary tracking, SQL database for logging, vector tracking, GNSS/INS integration, Matlab-interface, RTKLIB	

with the GPU (GPS World staff, 2012). Also semi-codeless tracking of GPS L2P(Y) (i.e. P-code aided cross-correlation) was implemented. The cooperation of UniBwM with IFEN lasted until 2013 when the development directions started to diverge. IFEN used the software mostly as base receiver platform with an Application Programming Interface (API) to support different applications, whereas UniBwM continued to modify the core, which was not always beneficial for software stability if seen from a commercial point of view.

The focus at UniBwM switched in 2017 as the old GUI could not be maintained anymore. Furthermore, real-time operation became less important as most scientific results were obtained in post-processing. The result was that a new GUI was developed and attached to the proven processing core. Any run-time optimizations within the processing core degrading navigation performance (i.e. mostly causing additional noise in the code tracking loop) were removed. The core's logging output was directed to a SQL database to store all different kind of intermediate results in a single file (additionally to the legacy ASCII logging into multiple files). A dedicated visualization tool for this database was developed.

The use of Windows and Visual Studio for developing a software radio is a little unusual, but is explained as follows. At UniBwM most researchers use Windows PCs to allow easy document exchange with each other and most importantly within the European Space industry. For this reason, all software receiver developments were done for Windows only. In terms of numerical performance and code optimization, Intel provided and still provides with the Intel C++ compiler and the Intel Performance Primitives the same quality on Windows as for Linux. Over the years it became, however, also clear that the potential use of the processing core on embedded devices and long-term stability might have been easier to achieve on the Linux operating system. IFEN ported part of the core to Linux, but not the full software receiver and showed that conventional desktop CPUs and embedded CPUs provided an impressive processing capability already in the year 2015 (Dampf et al., 2015).

As already mentioned, code optimization to achieve fast (and real-time) signal tracking was a main research focus in the first years. Different studies on CPU assembler instructions, CPU architecture and bottlenecks resulted in dedicated assembler implementations. Extensive lookup-tables were used and one very efficient correlator implementation with the Intel x86 pmaddubsw-instruction was based on a signal sample representation as unsigned integers (including the necessary rewriting of the correlation formulae due to the switch from the standard representation of samples as signed integers to unsigned integers). Fast Fourier Transform (FFT) based acquisition was already very efficient on the CPU and even more efficient on the GPU. The use of FFT libraries provided by NVIDIA made the acquisition code porting from CPU to GPU comparably easy. The situation is different for signal tracking. The tracking code has been transferred to the GPU and some optimization have been applied to minimize the amount of data transfer between CPU and GPU. However, since the correlation parameters are slightly different for each signal tracked, the correlation code is called multiple times and the latency to start one thread on the GPU

generated significant overhead. GPU-based tracking is thus currently only beneficial if a very large number (several hundreds) of correlators is configured per tracking channel, as pointed out by Pany et al. (2019). As modern desktop and laptop CPUs continue to improve and make use of a many-core structure, the need to port signal tracking to the GPU becomes less important. Furthermore, the use of dedicated assembler code required over the years continuous adaptation to new CPU instruction sets (e.g. from SSE to AVX instructions). The performance gained by using hand-coded assembler routines compared to the use of the libraries provided by Intel (IPP) is not always worth the effort and was not further actively pursued. Instead, dot-product routines (2 x 16-bit signed input to 64-bit output) from the IPP are employed for signal tracking.

The C++ universe is huge, and it is easy to integrate external source code. For example, the famous RTKLIB and the ION SDR sample reader code have been integrated. The current research work with MuSNAT focuses on GNSS/INS/LiDAR integration, support of massive antenna arrays, vector tracking and deep GNSS/INS coupling, support for LTE/5G-signals and GNSS signal simulation. It has to be admitted that the maintenance of the huge C++ code-base of MuSNAT at a University institute with a high fluctuation of researchers is partly demanding. The learning curve for good C++ development in this context is steep and for the purposes of obtaining a PhD degree often an inefficient way. Therefore, interfaces from the C++ code to Matlab were established and for example Open Service Navigation Message Authentication (OSNMA) decoding, PPP-computation for HAS or LiDAR odometry are implemented in Matlab. Another development is to use MuSNAT to generate multi-correlator values that are then used within a full Matlab based receiver to emulate signal correlation via interpolation (Bochkati et al., 2022).

UniBwM has initially used front-ends from Fraunhofer IIS and the software receiver included low-level Universal Serial Bus (USB) drivers for real-time data transfer. The same approach was used to connect the front-ends from IFEN GmbH to the processing core. The effort to write stable high data-rate low-level drivers is significant and introduces a dependency on libraries and support from the USB chip manufacturers. To reduce these kinds of development efforts, the decision to connect front-ends via TCP/IP was felt. This approach is powerful in terms of bandwidth and also generic and a first version of it is described in (Arizabaleta et al., 2021). Furthermore, with e.g. LabVIEW from NI it is comparably easy to develop a simple TCP/IP signal source for Universal Software Radio Peripheral (USRP) frontends. At the time of writing this paper, a more efficient firmware for USRPs with direct FPGA programming is being developed and shall allow to synchronously capture data from an Inertial Measurement Unit (IMU) together with the GNSS signal samples.

3. SoftGPS, SoftGNSSv3.0 and Derivatives

As abovementioned, (Borre et al., 2007) and the associated Matlab receiver was a cornerstone for GNSS SDR development. This receiver, initially called SoftGPS, then SoftGNSS (usually referred to as SoftGNSSv3.0), included the basic processing functions for GPS L1 C/A in a readable format, very useful for educational purposes. These included signal FFT-based acquisition, frequency, carrier phase and code phase tracking, data synchronization and demodulation, pseudorange generation, and eventually PVT. The Matlab code, together with some samples, was provided in a CD with the book, and was also available at Aalborg University's Danish GPS lab website. Apart from K. Borre and D. Akos, SoftGNSS included relevant contributions by D. Plausinaitis and others. Unfortunately, Kai Borre passed away in 2017 and the Danish GPS Lab was discontinued. However, SoftGNSS and its derivatives remain quite alive. Here are some examples:

- A new SDR GNSS book, (Borre et al., 2022), extending SoftGPS functionality to several frequencies, GNSS and architectures, can be considered as the successor of (Borre et al., 2007). A main building block of this book is FGI-GSRx, described in the following section, but the book includes also other Matlab receivers. In particular, DF-GSRx (Dual-Frequency GNSS Software Receiver), developed by Borre's PhD student P. Bolla, is a dual-frequency GPS L1/L5 receiver that includes dual-frequency acquisition techniques, measurements combination (iono-free in particular) and positioning. The book also includes a GPS L1 C/A snapshot receiver developed by Borre's former PhD student I. Fernandez-Hernandez, more modest than that described later in III.8, but simple and quick to execute and therefore possibly useful for educational purposes.
- The Easy Suite libraries (Borre, 2003, 2009), still publicly available and used, provide an excellent educational tool to dive into basic functions of GNSS receivers, such as calculating satellite positions from the ephemerides, datum conversions, or computing the receiver position and its accuracy in multiple ways (least squares, Kalman filter, carrier phase ambiguity resolution, etc.)
- (Bernabeu et al., 2021), as above mentioned, provides a collection of open source SDRs developed at University of Colorado Boulder and based on SoftGNSS.
- (Zhang, 2022) provides a respository with adaptations of SoftGNSS for different front-ends.

4. Finnish Geospatial Research Institute's Multi-GNSS Software Receiver

The software receiver developed by Finnish Geospatial Research Institute (FGI) is famously known as the FGI-GSRx (FGI's GNSS Software Receiver). The development of the FGI-GNSS Software Receiver (GSRx) software receiver started in 2012

Table 2: Main features of FGI-GSRx

FGI-GSRx		
Feature	Solution	Remark
Operating System	Windows 10	Compiles in Windows 10 environment. The software receiver should run in other OS which can host MATLAB or OCTAVE.
Programming environment	MATLAB	Executes in MATLAB 2019 or any other later version. The software receiver can be also executed in OCTAVE.
IF sample file input source	ION SDR standard	Read input data files following ION SDR standard.
Processing mode	Only operate as post-processing GNSS receiver	It can read raw IF data for a complete receiver processing, or it can load previously saved acquisition and/or tracking data in order to skip acquisition and/or tracking operation to be able to process navigation solution depending on parameters set in the user configuration file.
Supported GNSS	GPS L1, Galileo E1, BeiDou B1, GLONASS L1, NavIC L5	Open source FGI-GSRx only supports single frequency multi-GNSS processing.
Acquisition	FFT-based signal acquisition	Sophisticated research specific implementation for high sensitive acquisition is not published as open source.
Tracking	Table-based three-stage tracking	Based on the tracking status of each individual satellite, the software receiver switches among three stages: i) PULL IN, ii) COARSE TRACKING and iii) FINE TRACKING.
Navigation	Traditional Least Square (LS)	Users can select SNR or elevation cut-off mask in order to decide on the satellites that contribute to the position computation.

from the open source GNSS software receiver released in 2007 by Prof. Borre and his colleagues Borre et al. (2007). The software receiver was able to track two IOV (In-Orbit Validation) satellites called GIOVE A and GIOVE B from the European GNSS system Galileo. Since then, the researchers at FGI have been continuously developing new capabilities to the software receiver with the inclusion of Galileo in 2013 (Söderholm et al., 2016), the Chinese satellite navigation system BeiDou in early 2014 (Bhuiyan et al., 2014, 2015), the Indian regional satellite navigation System NavIC in late 2014 (Thombre et al., 2015) , and the Russian satellite navigation system GLONASS in 2015 (Honkala, 2016).

The FGI-GSRx software receiver has been extensively used as a research platform for the last one decade in different national and international research and development projects to develop, test and validate novel receiver processing algorithms for robust, resilient and precise Position Navigation and Timing (PNT). At present, the FGI-GSRx can process GNSS signals from multiple constellations, including GPS, Galileo, BeiDou, GLONASS, and NavIC. The software receiver is intended to process raw IF signals in post-processing. The processing chain of the software receiver consists of GNSS signal acquisition, code and carrier tracking, decoding the navigation message, pseudorange estimation, and PVT estimation. The software architecture is built in such a way that any new algorithm can be developed and tested at any stage in the receiver processing chain without requiring significant changes to the original codes. FGI-GSRx provides a unique and easy-to-use platform not only for research and development, but also for whoever is interested in learning about GNSS receivers. The software receiver was released as open source in February 2022 (FGI, 2022). FGI-GSRx receiver is also tied with the book 'GNSS Software Receivers' by Cambridge University Press, a next edition of one of the fundamental GNSS textbooks, which is now in press to be published in the second half of 2022. Some of the main features of FGI-GSRx is listed in Tab. 2.

The FGI-GSRx software receiver can be utilized in universities and other research institutes as a tool for training graduate level students and early-stage researchers for getting hands-on experience on GNSS receiver development. It can also be utilized in the vast GNSS industry as a benchmark software defined receiver implementation. The software receiver is already being used in the 'GNSS Technologies' course offered widely in Finland - at the University of Vaasa, Tampere University, Aalto University and the Finnish Institute of Technology.

Table 3: Main features of GNSS-SDR

GNSS-SDR		
Feature	Solution	Remark
Operating System	GNU/Linux, macOS, Windows OS through WSL.	Included as a software package in Debian and Ubuntu, and in Macports for macOS. Tested on ArchLinux, CentOS 7, Fedora, OpenSUSE, Rocky Linux.
Programming environment	C++	Software linters are automatically run at each code change to ensure meeting high-quality coding standards.
Processing mode	Real-Time and Post-Processing.	It can work in real-time using a wide assortment of commercial RF front-ends, and in post-processing mode with a number of file formats (including input files produced by the ION standard conversion tools).
Supported GNSS	GPS L1, L2C, L5; Galileo E1, E5a, E5b, E6; Glonass L1 CA, L2 CA; Beidou B1, B3.	The modular design allows for easy inclusion of new signals.
Acquisition	FFT-based signal acquisition.	A-GNSS capabilities to accelerate the Time To First Fix.
Tracking	Multicorrelator-based Data and Pilot signal tracking.	Customizable DLL, PLL, FLL. High-dynamics capabilities. SIMD-accelerated both in i686 and ARM CPUs (see Fernández–Prades et al. (2016a)).
Navigation	Traditional Least Square (LS), code and carrier-based positioning modes.	Positioning engine based on RTKLIB implementation (Takasu and Yasuda, 2009). All possible supported GNSS signals combinations are allowed.

5. GNSS-SDR, an Open-Source Software-Defined GNSS Receiver

The software receiver developed by the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), uncreatively named GNSS-SDR (but not related to the ION SDR standard), is another example of a multi-band, multi-system receiver. It has been constantly evolving since 2010, keeping pace with the newest GNSS algorithms and signals over more than a decade. It originated as a by-product of a CTTC research staff initiative, with the aim of providing a collaborative framework with other researchers seeking to accelerate research and development of software-defined GNSS receiver technology. The receiver particularly focuses on baseband signal processing, although it has the ability to run a navigation engine (refer to Table 3). The early stages of development baked slowly under a personal side-project scheme, with no funding, but with the purely exploratory objective of designing an optimal architecture specifically suitable for GNSS signal processing, where concepts such as testability, extensibility, reusability, scalability, maintainability, portability, adaptability to new non-standard requirements, and adoption of Computer Science best practices considered from scratch.

Its first popularity peak came on August 2012, with the reporting of the usage for GNSS of extremely cheap (about $25) DVB-T receivers based on the Taiwan's Realtek RTL2832U chipset, sold in form of USB dongles that allow users to watch over-the-air DVB-T European broadcast television on their personal computers. Normally, those devices send partially-decoded MPEG transport frames over the USB, but exploiting an undocumented mode of operation of the demodulator chip, the user was able to obtain raw I&Q samples, stream them through USB to a personal computer and then apply the GNSS-SDR software processing, turning the DVB-T receiver into a GNSS receiver and delivering position in real-time (see Fernández–Prades et al. (2013)). On a parallel development, in November 2013, the European Space Agency acknowledged GNSS-SDR as one of the first 50 users worldwide to achieve a successful Galileo position fix.

The project gained momentum and maturity over the years, and today it enjoys a solid and valuable user base continuously providing feedback, enhancements, and new features. Current versions are included in major GNU/Linux distributions, such as Debian and Ubuntu, and in Macports for Apple's macOS. The software package has been used in several public and private-funded research projects (including EUSPA, European Space Agency (ESA), NSF and NASA activities, as well as in educational programs such as Google Summer of Code), and it has been reportedly used for research purposes worldwide. The authors opened a discussion of quality metrics and key performance indicators for any generic software-defined receiver (Fernández–Prades et al. (2016b), extended online version available at `https://gnss-sdr.org/design-forces/`) and proposed the concept of *continuous reproducibility* in GNSS signal processing (Fernández–Prades et al. (2018)).

Table 4: Main features of AutoNav SDR

AutoNav SDR		
Feature	Solution	Remark
Operating system	Windows	
Programming environment	MATLAB and C	
Processing mode	Post-processing	
Supported GNSS	GPS (L1 C/A, L2C, L5), GLONASS L1, Galileo (E1, E5a, E5b), BDS (B1I, B1C, B2a), QZSS (L1 C/A, L1C, L2C, L5), NavIC L5	Free selection of signal combination
Acquisition	GPU-based acquisition	Simple implementation using Parallel Computing Toolbox of MATLAB
Tracking	MEX correlator	18/8 bits code/carrier replica tables, 32 bits code/carrier Numerically Controlled Oscillators (NCOs), bit shift operations
Further features	API, easy addition of new signals, RINEX observation logging, Radio Frequency Interference (RFI) mitigation based on pulse blanking, direct state tracking Kalman filter	

The full project and source code documentation can be found online at `https://gnss-sdr.org`, a website with over 5000 unique visitors per month, which contributes to raising awareness on GNSS technology. The website content is also on a GitHub repository at `https://github.com/gnss-sdr/geniuss-place`, hence undergoing public scrutiny. The project is also well-connected to its software ecosystem and existing SDR platforms. It builds on a wide range of GNU/Linux distributions and versions (from most recent releases to those released in 2014), and it provides a Yocto / Openembedded layer, which allows its portability to a wide range of embedded platforms (see Fernández–Prades (2022)).

The software produces standard outputs for observables and navigation data (RINEX files and RTCM-104 v3.2 messages as defined by the Networked Transport of RTCM via Internet Protocol, NTRIP), as well as position fixes in application-specific messages (*e.g.*, NMEA 0183), a variety of GIS-oriented file formats (KML, GeoJSON, GPX), and custom binary outputs that allow the observability of internal signal processing sub-products.

6. AutoNav SDR

The AutoNav SDR is a MATLAB-based multi-GNSS and multi-frequency software receiver that was developed by the Autonomous Navigation Laboratory of Inha University, South Korea (Song et al., 2021). Its main features are arranged in Tab. 4. The critical point considered in the design phase of this SDR is the maximization of reconfigurability. Since South Korea is developing its own satellite navigation system, KPS which is targeted to operate from 2035 as reported by Ministry of Science and ICT of Korea (2021), a flexible receiver that can process not yet existent signals is highly required. The AutoNav SDR is profoundly designed to provide full reconfigurability in terms of target signal combinations and signal characteristics, especially for easy addition of the new signal proposals. To do so, a basic framework of software receiver was designed with a well-designed processing functional architecture and data structure in consideration of the expandability of the signals and then applied to realize an SDR for GPS L1 C/A code signal as a first realization example by reconfiguring a configuration file via GUI. Then, different signals of the other constellations (GLONASS, Galileo, BeiDou Navigation Satellite System (BDS), Quasi-Zenith Satellite System (QZSS), NavIC) and frequencies (L1, L2, L5) were added quickly by utilizing this expandability. In this way, KPS signal candidates can be easily added to the SDR to evaluate and compare the performance of each candidate in the signal design phase. Similarly, a reconfigurable GNSS simulator was developed at the same time with the same idea. This is a MATLAB-based IF level GNSS/KPS simulator which can be ideally suited to test the navigation performance of any GNSS signals as well as new KPS signals by reconfiguring signal design parameters via GUI.

Although the AutoNav SDR is targeted for post-processing only, the original correlation operation in MATLAB with variables of double precision was too slow at the beginning of its design phase. So, two simple accelerations were applied to the SDR: a GPU-based acquisition module and a MEX correlator for tracking. The GPU-based signal acquisition module was implemented in a very simple way using the Parallel Computing Toolbox of MATLAB. If the GPU is usable, local variables for the correlation

Table 5: Main features of PyChips

PyChips		
Feature	Solution	Remark
Operating System	Windows x64 (8, 10, 11)	Due to pre-compiled C/C++ bindings that currently use the Windows API for file reading and threading
Programming environment	Python 3.10	
IF sample file input source	ION SDR Metadata Standard	Parses ION metadata hierarchy to select the appropriate decoder kernel written in C++. Sample decoding is split across multiple threads using a data parallel architecture
Real-time sample input	Not currently supported	
Additional sensors	None	
Supported GNSS	Supports all civilian satnav signals (GPS, GLONASS, Galileo, BeiDou, QZSS, NavIC, SBAS)	Spreading codes defined as memory codes
Acquisition	FFT based generic acquisition engine with configurable coherent and non-coherent integration settings	Auto detects and implements circular vs. non-circular frequency-domain correlation based on code length
Tracking	User configurable generic tracking module object	Employs split-sum correlation Gunawardena and van Graas (2006). Always operates on 1 millisecond block of samples and retires current block before operating on next block (no sample shifting to align with SV time-of-transmission)
Measurement output	Yes	CSV format
Availability	Release to public GIT repository pending	Versions used at ION tutorials

(i.e., code and carrier replicas) are generated in the GPU memory using the gpuArray function. Then, FFT and Inverse Fast Fourier Transform (IFFT), and correlations are performed in the GPU automatically. Finally, the correlation results are extracted using a gather function. With this simple approach, it has approximately 2.12 times faster execution time compared to the general CPU-based acquisition, without the relatively complex development using CUDA.

Since the most time-consuming process of the receiver is the correlator in the signal tracking, a MEX function is employed to reduce the computational burden. The MEX function connects the MATLAB environment to the external function written in C/C++ language with an appropriate wrapper function, so the user can call it within MATLAB. The MEX correlator was written in the standard C language and uses integer-based variables. The SDR pre-generates the code and carrier replica tables at the initialization process with resolutions of 18 bits and 8 bits, respectively. The code and carrier Numerically Controlled Oscillators (NCOs) have a resolution of 32 bits, so the indices of the tables for current code and carrier replica generation are calculated using bit shift operations of 14 bits and 24 bits, respectively. With these implementations, the overall execution time became much faster (approximately 5 times) than the original double precision-based code, but it still cannot operate in real-time. Currently, Inha University is developing the FPGA-based real-time GNSS receiver that only the correlator would be substituted by the FPGA board at the original AutoNav SDR.

To further enhance the flexibility, the AutoNav SDR also provides APIs at each part of the signal processing chain (such as ring buffer, acquisition, tracking, navigation message extraction, position calculation, etc.). The API design was influenced by the ipexSR of Stöber et al. (2010) and was implemented similarly using the Dynamic Link Library (DLL). Since MATLAB can load a library from DLL and call a function within the library, the API concept of the C/C++-based software can also be used analogously in the MATLAB environment. If the SDR is converted to an executable file (.exe) and provided to a user, the user can freely modify functions or develop algorithms by generating the DLL, without the need for the whole source codes.

7. PyChips

Pychips is a relatively new object-oriented satnav SDR that has been developed from scratch since 2018. It is based on the experience gained from two previous implementations, namely the MATLAB SDR that was distributed with Wideband TRIGR (see section V) and the ChameleonChips GNSS SDR Toolbox for MATLAB (Gunawardena, 2014).

One of the key promises of SDRs is their flexibility and hence its utility as an education and research tool. In the satnav context, various publicly available SDRs can be used to teach basic courses on satnav systems, signal processing, and receiver design. However, there is an implicit assumption that students have the relevant programming language skills for that particular SDR. Students are expected to understand the inner workings of the SDR in detail, and, more importantly, to make modifications to the code to add advanced capabilities and/or revisions as part of their graduate research projects. While somewhat valid, this programming language proficiency assumption may not always hold true. Further, given the situation, it may be far more efficient and beneficial to have grad students make deeper progress on their research rather than spending time becoming programming language experts. PyChips was developed from the ground up to support this notion. A more detailed introduction to PyChips can be found in Gunawardena (2021). It is implemented in Python with C++ bindings where performance is absolutely essential for reasonably fast execution.

The current version of PyChips supports the creation and definition of entire constellations of satellites with advanced next-generation signal structures, along with interference sources and channel effects. This simulation portion of PyChips (comprising of numerous source objects) synthesizes these signals at the sample level on to one or more sample streams that are grouped into objects called stream containers. A stream container is an abstraction of a satnav receiver's antenna(s) and RF front-end subsystem. This could be multi-frequency, multi-element, with different sample rates and bandwidths, IF or baseband sampling architecture, and any and all combinations thereof. If live-sky signal processing is the use case, then one or more sampled SDR data files can be specified to instantiate a stream container object that is functionally identical and imperceptible from a simulated one. PyChips uses the ION SDR Metadata standard to determine the appropriate C/C++ decoder/unpacker/re-quantizer kernel to use for reading and parsing these SDR files.

The sample streams contained in a PyChips stream container are processed using numerous sink objects. Currently implemented examples include virtual oscilloscopes and spectrum analyzers, as well as acquisition engines and signal tracking modules.

The unique feature of PyChips is that, all of the functionality described above is defined/specified using a grouping of JavaScript Object Notation (JSON) files. Current and next-generation advanced satnav signal structures and the receiver architectures to process them are constructed by assembling together pre-built low-level functional blocks. For example, as described in Gunawardena (2021), the user can build receiver tracking modules to process GPS L1C TMBOC(6, 1, 4/33) and Galileo E1OS CBOC(6, 1, 1/11) MBOC signals as simple BOC(1, 1) signals to model a low-cost low-power mass market receiver, or a high-end survey-grade receiver taking full advantage of these 'dual personality' signals.

Indeed, at this stage, the goal of the PyChips project is to hone the JSON specification layer with a vast number of diverse signal specifications, use cases and applications, in order to have it become a 'satnav signals and systems specification language.' Today, the reference SDR that implements this language is written in Python and is therefore called PyChips. However, the ultimate goal of this effort is to contribute towards satnav SDR implementations that have the performance, power efficiency, and scalability of ASICs with the flexibility, reconfigurability, adaptability, and ease-of-use of software.

8. UAB Snapshot GNSS Software Receiver

The UAB snapshot GNSS software receiver (cf. Tab. 6) was originally developed as part of the research activities on indoor GNSS positioning that were carried out by the Signal Processing for Communications and Navigation (SPCOMNAV) group at Universitat Autònoma de Barcelona (UAB), back in 2007. At that time, the group was involved in one of the two parallel contracts that ESA awarded to assess the feasibility of indoor GNSS positioning, under the project named DINGPOS. The proposed strategy was to rely on a combination of technologies such as WiFi, Ultra Wideband (UWB), 2G/3G cellular networks and GNSS as discussed by López-Salcedo et al. (2008). As far as GNSS was concerned, UAB was in charge of developing the software implementation of a so-called high-sensitivity GNSS (HS-GNSS) receiver, which could be able to operate under the extremely-weak signal conditions experienced indoors. This involves working under 10 to 40 dB attenuation losses, which drive the effective C/N0 down to values where conventional GNSS receivers are not able to operate anymore.

The proposed HS-GNSS receiver implementation was based on a snapshot architecture where a batch of input samples were processed at a time to provide the user's position. This approach is often referred to in the literature as *push-to-fix* or *acquisition-only*, since no tracking stage is actually implemented at the receiver. This means that the receiver operates in open-loop mode by providing at its output the observables obtained straightaway from the acquisition stage. The implementation of the HS-GNSS software receiver was strongly influenced by the work already initiated by Gonzalo Seco-Granados before joining UAB, during his period from 2002 to 2005 as technical staff at the European Space Research and Technology Center (ESTEC) of ESA in The Netherlands, where he was leading the activities concerning indoor GNSS and snapshot GNSS receivers. Actually, the core of the UAB snapshot GNSS receiver was inspired on the same concept of double-FFT acquisition already introduced by Jiménez-Baños et al. (2006). This algorithm uses two consecutive FFT operations for implementing the correlation of the received signal with the local code replica, and then the simultaneous estimation of the fine Doppler and bit synchronization. Interested readers on the double-FFT algorithm and on a detailed description of the UAB snapshot GNSS receiver implementation will find a comprehensive description written by Seco-Granados et al. (2012).

From a general perspective, the UAB snapshot GNSS software receiver implements a set of specific signal processing techniques that are tailored to the particular working conditions indoors. Nevertheless, the implementation is flexible and it does not prevent the receiver to be operated efficiently in other scenarios, such as outdoors. Regarding the indoor environment, the most important impairment to be counteracted is certainly the severe attenuation due to the propagation through building materials and other obstacles. Attenuation up to 40 dB can easily be experienced, thus requiring a specific action to recover as much of the lost power in order to still be able to detect GNSS satellites. Since it is the received energy what matters from a signal detection and estimation point of view, and energy is nothing but power times the observation time, the only way to compensate for an extremely weak received power is by increasing the observation time. This means processing a longer piece of received signal, which means implementing very long correlation integration times at the GNSS receiver, on the order of hundreds of milliseconds or even a few seconds. Unfortunately, increasing the correlation time is hindered by the presence of the navigation message data symbols, residual Doppler errors and clock instabilities. So the approach adopted in practice by most snapshot GNSS receivers, particularly those intended for high-sensitivity applications, is to split a long correlation into pieces of shorter, but long-enough coherent correlations, whose outputs are then noncoherently accumulated. This combination of coherent and noncoherent correlation has proven to be successful in increasing the receiver sensitivity and thus still be able to detect a few GNSS satellites indoors. Actually, an interesting discussion on how important having long-enough coherent integrations was discussed by Pany et al. (2009).

The correlation between the received signal and the local replica is therefore the most important operation of a snapshot GNSS receiver. The reason is that with such correlation, the most accurate code delay and Doppler observables need to be estimated. This is because no tracking stage is implemented, and thus there will be no chance to further refine these observables in subsequent stages of the receiver. It is for this reason that the correlation must be implemented in the most optimal way, taking into account subtle details that might be ignored in conventional GNSS receiver implementations. This is one of the advantages of the double-FFT algorithm implemented in the UAB snapshot GNSS receiver, which implements the optimal joint estimation of the code-delay and fine Doppler over a long period of time, where potentially sign transitions may occur due to the presence of data modulating symbols. Additional considerations such as how to handle a non-integer number of samples when performing the FFT, the interpolation between consecutive correlation peaks, the code-Doppler effect over a long correlation period, etc. can be found in Seco-Granados et al. (2012).

The code delay and Doppler estimates provided by the acquisition stage are then directly used by the navigation module to compute the user's position. Such code-delay estimates are ambiguous at one code period because no absolute time reference is available, and therefore no other time-delay information can provided but that contained with a PRN code period. This is because just a batch of received samples is processed, and thus no access to the transmission time encoded onto the navigation message is available in general. As a result, the user's position needs to be computed without such time reference, which becomes a very specific feature of snapshot GNSS receivers. This problem can be solved thanks to what is known as coarse-time navigation, where the conventional navigation equations are augmented to include an additional unknown that represents the missing absolute time reference. The interested readers will find in (Van Diggelen, 2009, Ch.4) an excellent description of this method.

Since its development in 2008, the UAB snapshot GNSS receiver has been a key tool for many research activities at the SPCOMNAV group. This software has been used for instance, to characterize the multipath propagation indoors (López-Salcedo et al., 2009), to assess the feasibility of using GNSS receivers in missions to the Moon, where the weak-signal problem is very similar to the indoor one (Manzano-Jurado et al., 2014), to test near-far mitigation techniques that may appear in indoor/Space applications (Locubiche-Serra et al., 2016), to assess the impact of phase noise (Gómez-Casco et al., 2016), and to provide GNSS positioning to Internet of Things (IOT) sensors in smart cities (Minetto et al., 2020) by means of a cloud-based implementation of the UAB snapshot GNSS receiver that was developed from 2016 to 2018.

The migration of the UAB snapshot receiver into a cloud-based implementation was certainly a major milestone that attracted the interest of the community and opened the door for totally new applications and use cases. The interest in cloud GNSS positioning was motivated by the fact that GNSS software receivers were running at that time in local computers next to the user who collected the samples to be processed. However, with the advent and widespread deployment of cloud computing platforms such as Amazon Web Services (AWS), Microsoft Azure and Google Cloud, such local computers could actually be placed anywhere, and remote access could be granted to upload and process GNSS samples in a remote server in a scalable manner. Furthermore, this approach fitted pretty well with a snapshot GNSS receiver implementation, where a batch of samples could be sent to a remote server where the user's position would then be computed using the same tools as in any other snapshot GNSS receiver. That is, using A-GNSS for reducing the acquisition search space, making extensive use of FFT operations and computing the user's position by means of coarse-time navigation techniques.

This was the idea behind the so-called "cloudGNSSrx", the cloud-based implementation of the UAB snapshot GNSS receiver as described in SPCOMNAV (2019). The architecture was based on a dockerized compilation of the Matlab source code implementing the UAB snapshot GNSS receiver. Then a system of job queues, schedulers and load balancers were built on AWS to automate and scale the remote execution of the receiver, and an API was developed for machine-to-machine communication,

Table 6: Main features of UAB Snapshot GNSS Receiver

UAB Snapshot GNSS Receiver		
Feature	Solution	Remark
Operating system	Any supported by MATLAB	
Programming environment	MATLAB	MATLAB version 6.0 (R12, 2000) or higher.
Processing mode	Post-processing	
Supported GNSS	GPS (L1 C/A, L5), Galileo (E1C, E5a)	
Acquisition	FFT-based signal acquisition	Implementing the double-FFT algorithm for both code correlation and bit synchronization. Long correlations can be implemented by non-coherently combining a set of coherent correlations. Assisted-GNSS (A-GNSS) is used to narrow the acquisition search space. Compatible with 3GPP RRLP-compliant XML data.
Tracking	None	No tracking is implemented because the receiver architecture is based on snapshot mode (i.e. acquisition-only).
Navigation	Weighted Least Squares (WLS)	Coarse-time navigation is implemented.
Further features	Implements near-far detection and interference detection.	

facilitating the provision of GNSS positioning to small IOT sensors (Lucas-Sabola et al. (2016)). In this way, IOT sensors requiring GNSS positioning were be able to offload most of the computational load to a remote server, thus significantly reducing the power consumption and thus extending their battery lifetime.

Low-power GNSS positioning is actually one of the main applications of cloud GNSS software receivers, since for snapshots shorter than a few tens on milliseconds, the energy spent in sending the GNSS samples to the cloud pays off for the significant energy that is saved at the user's terminal for not processing such samples, and doing it at the cloud instead (Lucas-Sabola et al., 2017). This feature was actually acknowledged by the former GSA, now the European Union Agency for the Space Programme (EUSPA), who identified the UAB cloud GNSS receiver as one of the promising technologies for the future adoption of GNSS in the IOT domain (European Union Agency for the Space Programme, 2018). The cloud GNSS software receiver developed by UAB was then licensed in 2019 to the startup company Loctio, who enormously developed the initial prototype and made it a commercial product.

It is important to remark that apart from the low-power consumption use case, cloud GNSS software receivers can also be used to provide access to sophisticated signal processing techniques that cannot be implemented in conventional receivers. For instance, advanced signal monitoring techniques, spoofing detection or authenticated/certified positioning, the latter being reported by Rügamer et al. (2016). There is therefore a brilliant future ahead for cloud GNSS software receivers with many exciting new applications still to come.

9. The NGene Family of Receivers at Politecnico di Torino and LINKS

The development of the GNSS software receiver, noted as NGene, at Politecnico di Torino and LINKS foundation, roots back to the early years of 2000. At such time the Navigation Signal Analysis and Simulation Group (NavSAS) was already working on the software implementation of several sections of the GNSS baseband processing leveraging on the strong background of the group at Politecnico di Torino on digital signal processing and in particular in the digital simulation of complex communications system.

Such early work was addressing the optimized implementation of the acquisition and tracking stages, both as post processing tool, or as core processing units on programmable hardware. In 2005, under regional funding, the research team, at that time affiliated also partially to the Istituto Superiore Mario Boella (now part of the LINKS foundation) started to develop a fully software, real time GNSS receiver for GPS and for the upcoming Galileo signals.

The outcome of the work was the first release of the software receiver NGene as reported by Molino et al. (2009), which was able to process in real-time the GPS, Galileo and EGNOS signal components broadcast on the L1/E1 band, after Intermediate Frequency downconversion and digitalization of the signal ensemble reaching the antenna. IF downconversion and digitalization

were demanded to an external analog front-end device, which communicated via USB connection with the personal computer hosting the software receiver. The A/D converter with front-end filtering, along with the antenna and its Low-Noise Amplifier, were the only non-software elements of the receiving chain. Since then, many features were added on top of that fundamental architecture, which has remained since today the distinguishing feature of the NGene family of receivers. This reconfigurable software receiver has been since long time the principal development tool for in-lab analysis, development and prototyping of signal processing algorithms and architectures.

For example, thanks to the flexible implementation, NGene was adapted to process the Galileo In-Orbit validation signals (GIOVE-A) and later to excitingly process the first Galileo signals as soon as they were available as described in Margaria et al. (2012). Later it allowed the research team to be one of the first worldwide to obtain a position fix from the first four Galileo satellites.

The software receiver kept evolving and it has been adapted to address different applications, maximising the benefits of the software radio implementation. Today the NGene family is configurable to support many different RF-to-IF front-ends, USB connected with the software processor, responding to the needs of tens of activities and projects. A simplified, low-complexity version implements GNSS positioning capabilities in ARM-based embedded processors as described by Troglia Gamba et al. (2015b). Other branches of the software were adapted to fly a GNSS-R receiver for reflectometry tests (Troglia Gamba et al., 2015a) , to test antijamming algorithms, and in 2017 to support the detection of the transmission of a Non-standard code and the effects on Galileo positioning (Dovis et al., 2017) as well as during the Galileo outage event in 2019 (Dovis et al., 2019).

The implementation of the algorithms to authenticate the Galileo message via the OS Navigation Message Authentication mechanism (OSNMA) is one of the most recent branches of the NGene family (Nicola et al., 2022; Troglia Gamba et al., 2020a), together with the implementation of the set of functions to elaborate the future GPS Chimera authentication service (Troglia Gamba et al., 2020b).

10. The MATRIX SDR for Navigation with Signals of Opportunity

MATRIX (Multichannel Adaptive TRansceiver Information eXtractor) is a state-of-the-art cognitive SDR, developed at Kassas' Autonomous Systems Perception, Intelligence, and Navigation (ASPIN) Laboratory, for navigation with terrestrial and space-based SOPs (Kassas et al., 2020). MATRIX continuously searches for opportune signals from which it draws navigation and timing information, employing signal characterization on-the-fly as necessary. MATRIX could produce a navigation solution in a standalone fashion (Shamaei and Kassas, 2021a) or by fusing SOPs with sensors (e.g., IMU (Morales and Kassas, 2021), LiDAR (Maaref et al., 2019), etc.), digital maps (Maaref and Kassas, 2020), and/or other signals (e.g., GNSS) (Kassas et al., 2017). Fig. 1 shows MATRIX's architecture.

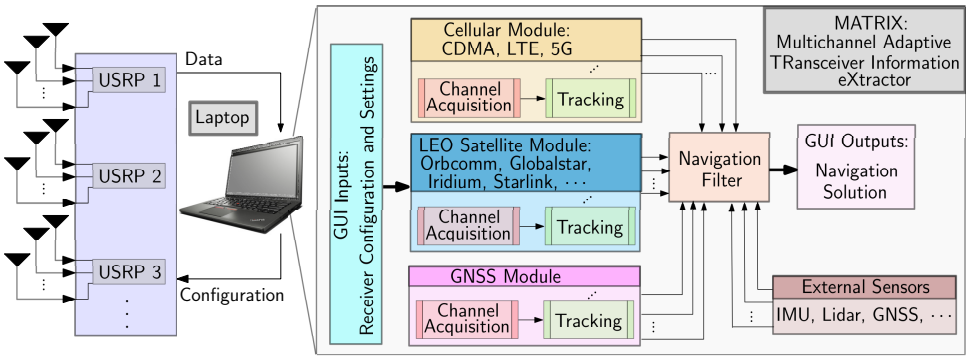


Figure 1: MATRIX cognitive SDR architecture. The SDR consists of: (i) USRPs to collect different radio signals, (ii) various modules to produce navigation observables from different types of signals (e.g., cellular, LEO satellites, etc.), (iii) external sensors (e.g., IMU, LiDAR, GNSS receivers, etc.), whose measurements can be fused with the navigation observables produced by the signal modules, and (iv) navigation filter that fuses all measurements to produce a navigation solution.

On one hand, MATRIX has achieved the most accurate navigation results to-date in the published literature with cellular SOPs (3G CDMA, 4G LTE, and 5G NR), achieving meter-level navigation indoors (Abdallah and Kassas, 2021) and on ground vehicles (Maaref and Kassas, 2022) and submeter-level navigation on unmanned aerial vehicles (Khalife and Kassas, 2022). In addition, MATRIX's efficacy has been demonstrated in a real-world GPS-denied environment (Kassas et al., 2022b), achieving a position root-mean squared error of 2.6 m with 7 cellular LTE eNodeBs over a trajectory of 5 km (one of which was more than 25 km away), during which GPS was intentionally jammed (Abdallah et al., 2022). MATRIX has also achieved remarkable results on high-altitude aircraft, where it was able to acquire and track cellular 3G CDMA and 4G LTE signals at altitudes as high as 23,000 ft above ground level and from cellular towers more than 100 km away (Kassas et al., 2022c). What is more,

meter-level high-altitude aircraft navigation was demonstrated over aircraft trajectories exceeding 50 km, by fusing MATRIX's cellular navigation observables with an altimeter (Kassas et al., 2022a).

On the other hand, MATRIX has achieved the first published results in the literature for exploiting unknown SpaceX's Starlink LEO satellite signals for positioning, achieving a horizontal positioning error of 10 m with Doppler observables (Neinavaie et al., 2021) and 7.7 m with carrier phase observables (Khalife et al., 2022). In addition, the first ground vehicle navigation results with multi-constellation LEO (Orbcomm, Iridium NEXT, and Starlink satellites) were achieved with MATRIX (Kassas et al., 2021), upon coupling its LEO navigation observables with an inertial navigation system (INS) in a tightly-coupled fashion through the simultaneous tracking and navigation (STAN) framework (Kassas et al., 2019).

IV. SDR FRONT-ENDS

A front-end is required to obtain digital samples for the SDR processing. The front-end's tasks are to receive, filter, amplify, down-convert, and further digitize and quantize the analog RF signal entering the GNSS antenna. Many different types of front-ends were used for GNSS SDRs. Roughly, five different categories can be identified:

- **discrete components:** Using RF-connectable components like Low Noise Amplifiers (LNAs), filters or ADCs it is comparable easy to realize the function of a front-end and log IF or baseband samples. Those setups are easy to realize but often bulky and sometimes prone to interference.
- **commercial signal recorders:** several companies offer GNSS signal recorders to allow to record (and often to replay) one or more GNSS frequency band. Usually they do not implement a real-time connection to an SDR.
- **generic non-GNSS front-ends:** SDR technology is used in many different fields of electrical engineering and front-ends covering a wide frequency are available. Their price ranges from a few Dollars (Fernández–Prades et al., 2013) to highly sophisticated multi-channel front-ends costing several ten-thousands of Dollars. The oscillator quality, bit-width or RF-filter characteristics is not always optimal for GNSS signal processing.
- **dedicated GNSS real-time front-ends**: Built for the purpose to realize a real-time GNSS SDR. A good example is described in sect. IV.1. They are compact and build with discrete components.
- **ASICs**: Some RF-ASICs seem to target GNSS SDR use and evaluation kits allow streaming of IF samples, e.g. RF Micro Devices, Inc., Greensboro (2006); NTLAB, UAB (2022).

GNSS signals need a comparable high sampling rate of the front-end and when connected to a PC via a USB cable the transfer was not always reliable in the early years. Various optimizations and workarounds have been implemented like watermarking the IF sample stream and skipping lost sample packets as invented by Foerster and Pany (2013). With the advent of USB 3.0 or PCIe those solutions became obsolete.

In the following section, we describe Fraunhofer USB front-ends as an example of user needs, main features and general architectures of GNSS SDR front-ends. For a broader perspective of GNSS-compatible front-ends in the market, the interested reader can refer to (Borre et al., 2022, Ch.12).

1. Fraunhofer USB Front-ends

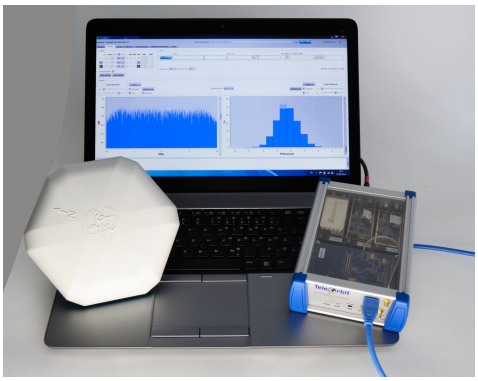
With the upcoming civil multi-band signals in GPS, and Galileo planning progressing, there was a need for the scientific community but also with some industrial partners, to have a multi-band SDR front-end solution, to enable also multi-band SDR development.

In 2006, Fraunhofer IIS developed a front-end, called the L125 Triband USB (see Fig. 2a), which allows a fixed frequency recording of L1/E1, L2, and L5/E5a via two USB 2.0 data streams with up to 40 MSPS sampling rate, a 2 or 4 bit ADC resolution, and one antenna input.

However, increasing customer requests for portable and flexible solutions concerning frequency band selection, adjustable sampling rates, intermediate frequencies, and multi-antenna support led to completely redesigning this USB front-end concept. One major request was reconfigurability on the SDR front-end side. While for specific projects, a single band receiver with a low-sampling rate is desired (i.e. to realize a real-time SDR), in other projects, a wideband and multi-frequency front-end may be needed. To cope with these different requirements in one SDR front-end hardware, a new version of the USB front-end was developed that realizes the signal conditioning to an on-board FPGA enabling the desired reconfigurability on the fly.

In 2012 Fraunhofer IIS (Rügamer et al., 2012) introduced the Flexiband multi-system, multi-band USB front-end depicted in Fig. 2b. Within the last ten years, this front-end has been used and validated in numerous scientific and industrial projects. Furthermore, it has been commercialized and is distributed as the "MGSE REC" product of TeleOrbit GmbH (2022).

(a) TriBand USB2.0 Front-end from 2006



(b) Flexiband USB3.0 Front-end from 2012 onwards

Figure 2: Two examplery USB Front-end from Fraunhofer IIS

A regular Flexiband unit consists of up to three analog reception boards, a carrier board with ADCs and FPGA, and a USB 3.0 interface board. A common antenna input port is supported, and separate front-end input signals for up to three antenna inputs. Three dual-channel ADCs sample the incoming signal with 81 MSPS and 8 bits I/Q. The raw data stream is received by an FPGA in which different digital operations like filtering, mixing, data rate, and bit-width reduction, as well as a digital Automatic Gain Control (AGC) are applied. Finally, a single multiplexed data stream is formed together with a checksum. This multiplexed stream is sent via an USB 3.0 interface to the PC. Data rates of up to 1296 MBit/s or 162 MByte/s raw data stream are supported. The Flexiband GUI software receives the raw multiplexed stream, checks its integrity, and demultiplexes it. The data streamed can be either written to hard disk or sent to a customer application (e.g., a software receiver). The raw samples can be stored as a multiplexed data stream, in an 8 bit/sample format, or directly as a .mat file for MATLAB. In parallel, the ION Metadata *.sdrx is provided.

Due to its bandwidth, sampling rates, quantization, and multiplexing schema flexibility, the ION metadata standard was a perfect fit to clearly and unambiguously define the configuration for the user. Therefore, right after the first conclusion of the ION metadata standard, each binary raw-data output file of the Flexiband front-end was equipped with an "sdrx" metadata file specifying the raw data format.

Finally, a replay variant of this Flexiband exists that reads in the raw IF samples on hard disk using the ION metadata standard specification and replays the digital data as an analog RF output stream supporting multiple GNSS bands at the same time.

V. ION SDR METADATA STANDARD

The events that led up to the suggestion to develop what became the ION SDR Metadata Standard can be traced back to circa 1999. Building upon the successful contributions made by Akos, the Ohio University Avionics Engineering Center undertook several research projects leveraging GPS SDRs. One such project was called the GPS Anomalous Event Monitor (GAEM) (Snyder et al., 1999). This was sponsored by the FAA Technical Center and led by Prof. Frank van Graas. Commercial GPS receivers within prototype LAAS ground facilities were experiencing brief unexplained outages. GAEM kept a continuous 10-second history of IF samples in a circular memory buffer. When an outage occurred, GAEM was triggered to dump this buffer to disk and collect for a further 10 seconds. These sample files were then post-processed in MATLAB to determine the cause of the anomaly. Early versions of GAEM used commercial data collection cards and had numerous issues related to their proprietary drivers. Around 2001, Gunawardena developed a refined version of GAEM that was based on one of the earliest PCI-based dual-ADC-plus-FPGA development cards commercially available. It collected two GPS L1 data streams at 5 MSPS and 2 MHz bandwidth. This version of GAEM was fielded at three airports and operated continuously for over 3 years and helped to characterize numerous anomalous events (Gunawardena et al., 2009). This GAEM also supported a continuous collection mode, and was used for several research projects including the characterization of GPS multipath over water (Zhu and Van Graas, 2009) and GPS/IMU deep integration demonstrations in flight (Soloviev et al., 2004). For the latter, the 2 kHz raw data from a MEMS IMU were interleaved with the SDR samples thanks to the FPGA-based architecture that allowed for such custom capabilities.

Circa 2002, as these research projects progressed, the 2 MHz bandwidth limitation of GAEM became apparent. There was a pressing need to support emerging research opportunities related to GPS L5, as well as high fidelity GPS signal quality monitoring. A multi-band and higher-bandwidth (24 MHz) front-end and SDR data collection system was needed. There were only a handful of vendors selling such systems at the time, and it wasn't clear if these would serve the purpose for satnav SDR

application (sampling coherency concerns, etc.). However, by far, the >$350k price tag of these systems precluded any hope of purchasing them for university research. It was decided to develop this capability in-house. In 2003, a 2-channel L1/L5 front-end with 24 MHz bandwidth and 56.32 MSPS was developed (Gunawardena et al., 2008). It was based on connectorized RF components. The sampling and collection subsystems were carried over from GAEM.

The capabilities of the dual-frequency high-bandwidth system attracted interest from several universities, government research groups, as well as a defense contractor. To support these opportunities, the development of a new system known as Wideband Transform-domain Instrumentation GNSS Receiver (TRIGR) was completed in 2008 (Gunawardena and Van Graas, 2011). The front-end was miniaturized to a single-frequency custom PCB module. Up to 8 such modules (with the required frequency options) were combined with an 8-channel 12-bit ADC to create modular systems for various sponsors. The raw samples from the ADC are transferred to a PCIe FPGA card where the 8 streams are packed in various formats according to the user's selection in a GUI. Supported formats range from any one stream at 1-bit sample depth, any 2 streams at 12 bits (sign extended to 16), to all 8 streams at 4 bits. The sustained data transfer rate from the PCIe FPGA card to the RAID storage array was limited to 240 MB/sec. As such, the appropriate format had to be selected to balance between the required capability and transfer rate. The generated file names embed a UTC timestamp as well as the packed stream order and sample depth.

The event-based data collection feature of GAEM needed to be incorporated into Wideband TRIGR. However, the >10× data rate meant that a 10-second circular buffer could not be easily implemented in RAM using 32-bit systems of the day. This issue was addressed by writing data as a sequence of smaller files, where a new file was spawned before the current file was closed – with some sample overlap for data integrity – a technique known as temporal splitting. A separate process was used to delete older files from the RAID array to make room for new ones – unless an event was received – in which case the files surrounding the event were moved to a folder for post processing.

With the myriad of sample packing formats available with Wideband TRIGR, along with the temporal splitting-based file generation scheme, it became clear that a machine-readable metadata file needed to be included with every collection. An XML schema was designed for this purpose.

Up until this time, apart from the FPGA-based real-time GPS receiver that was developed and used for certain projects, all SDR files generated by GAEM and Wideband TRIGR were post processed in MATLAB. As others have mentioned, this was excruciatingly slow – especially for Wideband TRIGR data. To address this issue, as well as to support the rapid emergence of multi-band and multi-constellation satnav signals, Gunawardena wrote and distributed a MATLAB SDR toolbox where correlation was performed in optimized C code and also leveraged multi-threading in a data parallel architecture. This toolbox, known as ChameleonChips, also read the XML metadata files produced by Wideband TRIGR to determine the appropriate sample unpacking kernel to use. This work was presented at ION GNSS+ 2013 in Nashville, TN (Gunawardena, 2013). During this presentation, it was suggested that the satnav SDR community should adopt a metadata standard – similar to the one developed for Wideband TRIGR – in order to alleviate the numerous headaches associated with sharing such files. This was met with widespread support and enthusiasm. Longstanding ION members Phillip Ward, Jade Morton, and Michael Braasch helped to pitch this idea to the ION Executive Committee.

During the January 2014 Council Meeting in San Diego, ION approved the process for establishing a formal standard (Gunawardena et al., 2021). The ION GNSS SDR Metadata Working Group (WG) was formed in April 2014 with Thomas Pany and Gunawardena as co-chairs (James Curran was added later as a third co-chair). Membership represented academia, industry (including satnav SDR product vendors as well as traditional satnav equipment manufacturers), non-profit research entities, and government agencies spanning countries in Europe, America, Asia, and Australia. The working group developed the standard as well as associated normative software over a course of six years. With regards to the normative software, while many individuals contributed, initial development of the C++ object model was performed by Michael Mathews of Loctronix while James Curran wrote much of the functionality to decode packed samples based on the metadata specification. The draft standard was adopted as a formal ION standard in January 2020.

1. Use of GNSS SDR Standard

Today the GNSS SDR standard serves as a reference to describe IF formats and is for example useful for public tenders or if for some means an established format is needed. A number of SDRs do include the C++ libraries to read meta-data and IF samples.

The level of exchange of IF samples between research groups is to some extent limited and much less executed compared to e.g. exchange of RINEX files. This is of course related to the huge size of IF sample files and to the fact that for the majority of GNSS use cases, RINEX observation data or PVT exchange is sufficient. Furthermore, GNSS SDRs still tend to use mostly the same front-end and once the respective data format is known, there is obviously no need to describe it via the XML format. A disadvantage of the C++ routines is their generic design, which renders sample reading quite slow, as each sample is isolated via a number of for-loops from the input files. Clements et al. (2021) did propose an algorithm to automatically generate optimized code for sample reading for a given IF format, but this proposal did not yet manifest into a usable implementation.

2. Standard Extension

Already during the standardization process a number of features for the standard were identified, that appear to be useful but lack of resources did not allow including them in the formal standardization procedure. Those features are described in the App. II of (ION SDR Working Group, 2020). Within the ION-GNSS+ 2022 meeting in September, the following points have been discussed and will be included in App. II of the next - draft-version V1.1 of the standard:

a) Flexible bit layout

The SDR metadata standard defines a "Lump" as the ordered containment of all samples occurring within an interval. The ordered containment is understood in a regular way holding the samples of the individual streams together. The authors of (Clements et al., 2021) see this as a limitation, as highly efficient SDRs may use efficient bit-packing schemes to optimize data transfer over communication lines that need buffering. They identify a need to distribute the samples of different Streams in interleaved ways over the Lump. This interleaving cannot be described by the V1.0 of the SDR metadata standard. To overcome this limitation, the authors propose a new but optional attribute for the Lump object, called "Layout". In case Layout is present, further information on the bit packing scheme needs to be provided, describing in an explicit way the type of each bit of a Lump. The authors propose a detailed proposal following the structure of the existing standard can be found for this new Lump layout. The proposal even includes more advanced bit use cases, like puncturing (e.g. explicit omitting of bits) and overwriting of bits by time markers.

b) Refined sample rate/epoch definitions

In the work (Clements et al., 2021), the authors note that the V1.0 of the SDR metadata standard makes implicit assumptions about the timing of the sampling process and staggered sampling cannot be described by it. Staggered sampling occurs if the sampling instants of different GNSS signals are delayed with respect to each other, and might be of use to increase observability of GNSS interference in a multi-antenna system. To overcome this limitation, the authors propose to add two new attributes for stream objects to shift the sampling epochs of different GNSS stream with respect to each other.

c) JSON format for metadata files

Comment ID 22 of the initial Request for Comments (RFC1) makes a suggestion that the WG considers markup languages other than XML for metadata files, specifically JSON, YAML and TOML (Anon, 2017). In 2017, this comment was addressed by asserting that the XML format will be maintained for the time being since normative software that parses XML had already been developed. However, the WG responded with the assurance that "other markup languages will be considered in the future based on community need and interest."

As of the time of this writing, and with the experience gained from developing PyChips (which is a satnav SDR that is completely described using a draft signal/system specification language based on JSON, as described in III.7), it is this author's opinion that JSON may have some distinct advantages over XML for future applications and use cases. For example, JSON streaming is a methodology for transferring object-oriented data over communications protocols (Wikipedia, 2022) and is widely used in well-known applications such as Plotly (2022). Hence, streaming JSON could be one way to parse SDR sample streams whose formats are changing dynamically.

Figure 3 shows a notional listing for a JSON formatted metadata description for the Flexiband front-end XML metadata listing found in Gunawardena et al. (2021).

To maintain compatibility with the existing and formally adopted XML-based metadata specification, it is understood that any adoption of another markup language such as JSON must include open source normative software and tools to convert between these formats. Adoption of JSON based metadata is currently being considered for future versions of PyChips. If and when a successful implementation has been achieved, consideration for adopting JSON as another valid option for representing ION Standard-compliant metadata in a future version of the standard will be requested.

SUMMARY AND CONCLUSION

Since GPS SDR developments started in the mid 90's, together with the operational declaration of GPS, its feasibility has been widely proven by several platforms and their derivatives. We define GNSS SDR platforms as those implementing the receiver functions in general purpose software and processors, and divide them in real-time receivers, teaching/research tools, and snapshot receivers. We then describe some of them, with focus on those related to the authors but also including other developments. In particular, and based on the pioneering work by D. Akos, we describe the Bit-Wise Parallelism platform by the Cornell GPS group, which led to GRID/PpRx by UT Austin; the MuSNAT receiver by UniBwM, which also led to IFEN GmbH's SX3 commercial receiver; The SoftGPS Matlab receiver and associated book, widely used for GNSS teaching and also influencing other platforms, such as FGI-GSRx; the popular C++ open source GNSS-SDR by CTTC; AutoNav SDR by Inha University; PyChips by S. Gunawardena and based on Python; the snapshot GNSS receiver by UAB, leading to cloudGNSSrx;

```
{"system": {
    "id": "Flexiband", "comment": "Front-end variant:II-4e PRS (E6abc/E1abc/uBlox)",
    "freqbase": {"format": "MHz", "value": 8.10e+01 },
    "equipment": "Flexiband Multi-band receiver", "types": "Flexiband Multi-band receiver",
    "sessions": {
      "0": {
        "comment": "flexiband", "toa": "2022-09-30T00:00:00.00Z",
        "contact": "J. Doe", "campaign": "Example Campaign", "scenario": "Example Scenario",
        "position": {"lat": "0.0", "lon": "0.0", "height": "0.0"},
        "files": {
          "0": {
            "url": "flexiband.usb", "timestamp": "2022-09-30T00:00:00.00Z", "owner": "Example Owner",
            "lanes": {
              "0": {
                "id": "Data",
                "block": {
                  "cycles": 506, "sizeheader": 6, "sizefooter": 6,
                  "chunk": {
                    "sizeword": 1, "countwords": 2, "endian": "undefined", "padding": "none", "wordshift": "left",
                    "lump": {
                      "streams": {
                        "E6abc_B3": {
                          "other_specs": {"Amp": "162", "Antenna power": "false", "AGC": "false"},
                          "ratefactor": 1, "quantization": 4, "packedbits": 4, "alignment": "undefined",
                          "shift": "undefined", "format": "IQ", "encoding": "INT",
                          "bands": {
                            "E6abc_B3": {
                              "centerfreq": {"format": "MHz", "value": 1.27e+03},
                              "translatedfreq": { "format": "kHz", "value": 8.75e+03},
                              "delaybias": { "format": "sec", "value": 0.0e+00},
                              "bandwidth": { "format": "Hz", "value": 0.0e+00}
                            }
                          }
                        },
                        "L1_E1abc_B1_G1": {
                          "other_specs": { "Amp": "120", "Antenna power": "false", "AGC": "false"},
                          "ratefactor": 1, "quantization": 4, "packedbits": 4, "alignment": "undefined",
                          "shift": "undefined", "format": "IQ", "encoding": "INT",
                          "bands": {
                            "L1_E1abc_B1_G1": {
                              "centerfreq": {"format": "MHz", "value": 1.58e+03},
                              "translatedfreq": {"format": "kHz", "value": -4.58e+03},
                              "delaybias": {"format": "sec", "value": 0.0e+00},
                              "bandwidth": {"format": "Hz", "value": 0.0e+00}
                            }}}}}}}}}}}}}}}
```


Figure 3: Notional JSON Representation of Flexiband Front-End Metadata from Gunawardena et al. (2021)

the real-time N-GENE receiver by LINKS, used for early testing of Galileo first signals and OSNMA and the MATRIX receiver by ASPIN for navigation with terrestrial and space-based SOP among others. We provide an overview of the tasks and components of SDR front-ends, and for this purpose we describe Fraunhofer developments from the last years as a reference. Finally, we discuss the SDR Metadata Standard, officially approved by ION in 2020, and its current extensions.

In view of the impact in the GNSS community and the progress in the last decades, we conclude that GNSS SDR has a promising future and will continue coexisting with FPGA and ASIC receivers for the decades to come.

ACKNOWLEDGEMENTS AND REMARKS

The authors are listed in alphabetical order to reflect the variety and importance of all contributions. The contributions have been partly edited by Thomas Pany, who also takes the responsibility for the structure of the work, the abstract and the introduction.

The GNSS SDR developments at the Universität der Bundeswehr München of sect. III.2 were supported by via numerous research projects administered by the German DLR and were financed by the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

The UAB snapshot GNSS software receiver was supported in part by numerous ESA-funded research projects and by the Spanish State Research Agency (AEI) project PID2020-118984GB-I00.

The MATRIX SDR development at the ASPIN Laboratory was supported by the Office of Naval Research (ONR) under Grants N00014-16-1-2305, N00014-19-1-2613, and N00014-19-1-2511; the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-22-1-0476; the U.S. Department of Transportation (USDOT) under Grant 69A3552047138 for the CARMEN University Transportation Center (UTC); and the National Institute of Standards and Technology (NIST) under Grant 70NANB17H192.

The authors would like to acknowledge the work of Prof. Kai Borre, who passed away in 2017, for his influential contributions to the GNSS SDR field over the last decades.

REFERENCES

Abdallah, A. and Kassas, Z. (2021). Multipath mitigation via synthetic aperture beamforming for indoor and deep urban navigation. *IEEE Transactions on Vehicular Technology*, 70(9):8838–8853.

Abdallah, A. and Kassas, Z. (2022). Opportunistic navigation using sub-6 GHz 5G downlink signals: A case study on a ground vehicle. In *Proceedings of European Conference on Antennas and Propagation*, pages 1–5.

Abdallah, A., Kassas, Z., and Lee, C. (2022). Demo: I am not afraid of the GPS jammer: exploiting cellular signals for accurate ground vehicle navigation in a GPS-denied environment. In *Proceedings of Workshop on Automotive and Autonomous Vehicle Security*, pages 1–1.

Akos, D. and Braasch, M. (1996). A Software Radio Approach to Global Navigation Satellite System Receiver Design. In *Proceedings of the 52nd Annual Meeting of The Institute of Navigation*, pages 455–463.

Akos, D., S andockmaster, M., J.B.Y., T., and Caschera, J. (1999). Direct bandpass sampling of multiple distinct RF signals. *IEEE Transactions on Communications*, 47(7):983–988.

Akos, D. M. (1997). *A Software Radio Approach to Global Navigation Satellite System Receiver Design*. PhD thesis, Fritz J. and Dolores H. Russ College of Engineering and Technology Ohio University.

Akos, D. M., Normark, P.-L., Enge, P., Hansson, A., and Rosenlind, A. (2001). Real-time gps software radio receiver. In *Proceedings of the 2001 National Technical Meeting of the Institute of Navigation*, pages 809–816.

Anon (2017). RFC1 Comments and Responses. `https://sdr.ion.org/RFC1_Comments_Responses.html`. [Accessed on 09/30/2022].

Arizabaleta, M., Ernest, H., Dampf, J., Kraus, T., Sanchez-Morales, D., Dötterböck, D., Schütz, A., and Pany, T. (2021). Recent enhancements of the multi-sensor navigation analysis tool (musnat). In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pages 2733–2753.

Bernabeu, J., Palafox, F., Li, Y., and Akos, D. (2021). A collection of sdrs for global navigation satellite systems (gnss). In *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, pages 906–919.

Bhuiyan, M., Söderholm, S., Thombre, S., Ruotsalainen, L., and Kuusniemi, H. (2015). Performance Analysis of a Dual-

Frequency Software-Defined BeiDou Receiver with B1 and B2 Signals. In *Proc. of the China Satellite Navigation Conference (CSNC)*.

Bhuiyan, M. Z., Söderholm, S., Thombre, S., Ruotsalainen, L., and Kuusniemi, H. (2014). Overcoming the Challenges of BeiDou Receiver Implementation. *Sensors 2014*, 14(11):22082–22098.

Bochkati, M., Dampf, J., and Pany, T. (2022). On the use of multi-correlator values as sufficient statistics as basis for flexible ultra-tight gnss/ins integration developments. In *2022 25th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE.

Borre, K. (2003). The GPS Easy Suite–Matlab code for the GPS newcomer. *GPS Solutions*, 7(1):47–51.

Borre, K. (2009). GPS Easy suite II. *InsideGNSS*, 2:48–51.

Borre, K., Akos, D., Bertelsen, N., Rinder, P., and Jensen, S. H. (2007). *A Software-Defined GPS and Galileo Receiver, Single Frequency Approach*. Birkhäuser, Boston.

Borre, K., Fernandez-Hernandez, I., Lopez-Salcedo, J. A., and Bhuiyan, M. Z. H. (2022). *GNSS Software Receivers*. Cambridge University Press, Cambridge.

Chen, X., Wei, Q., Wang, F., Jun, Z., Wu, S., and Men, A. (2020). Super-resolution time of arrival estimation for a symbiotic FM radio data system. *IEEE Transactions on Broadcasting*, 66(4):847–856.

Clements, Z., Iannucci, P. A., Humphreys, T. E., and Pany, T. (2021). Optimized bit-packing for bit-wise software-defined GNSS radio. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION-GNSS+ 2021)*. Institute of Navigation.

Curran, J. T., Fernández–Prades, C., Morrison, A., and Bavaro, M. (2018). Innovation: The continued evolution of the GNSS software-defined radio. *GPS World*, 29(1):43–49.

Dampf, J., Pany, T., Bär, W., Winkel, J., Stöber, C., Fürlinger, K., Closas, P., and Garcia-Molina, J. A. (2015). More than we ever dreamed possible: Processor technology for gnss software receivers in the year 2015. *Inside GNSS*, 10(4):62–72.

del Peral-Rosado, J., Estatuet-Castillo, R., Lopez-Salcedo, J., Seco-Granados, G., Chaloupka, Z., Ries, L., and Garcoa-Molina, J. (2017). Evaluation of hybrid positioning scenarios for autonomous vehicle applications. In *Proceedings of ION International Technical Meeting Conference*, pages 2541–2553.

Del Peral-Rosado, J., Nolle, P., Razavi, S., Lindmark, G., Shrestha, D., Gunnarsson, F., Kaltenberger, F., Sirola, N., Särkkä, O., Roström, J., Vaarala, K., Miettinen, P., Pojani, G., Canzian, L., Babaroglu, H., Rastorgueva-Foi, E., Talvitie, J., and Flachs, D. (2022). Design considerations of dedicated and aerial 5G networks for enhanced positioning services. In *Proceedings of Workshop on Satellite Navigation Technology (NAVITEC)*, pages 1–12.

Diouf, C., Janssen, G., Dun, H., Kazaz, T., and Tiberius, C. (2021). A USRP-based testbed for wideband ranging and positioning signal acquisition. *IEEE Transactions on Instrumentation and Measurement*, 70:1–15.

Diouf, C., Janssen, G., Kazaz, T., Dun, H., Chamanzadeh, F., and Tiberius, C. (2019). A 400 Msps SDR platform for prototyping accurate wideband ranging techniques. In *Proceedings of Workshop on Positioning, Navigation and Communications*, pages 1–6.

Donaldson, J. E., Parker, J. J., Moreau, M. C., Highsmith, D. E., and Martzen, P. D. (2020). Characterization of on-orbit gps transmit antenna patterns for space users. *NAVIGATION: Journal of the Institute of Navigation*, 67(2):411–438.

Dovis, F., Linty, N., Berardo, M., Cristodaro, C., Minetto, A., Nguyen Hong, L., Pini, M., Falco, G., Falletti, E., Margaria, D., Marucco, G., Motella, B., Nicola, M., and Troglia Gamba, M. (2017). Anomalous gps signals reported from svn49. *GPS World*.

Dovis, F., Minetto, A., Nardin, A., Falletti, E., Margaria, D., Nicola, M., and Vannucchi, M. (2019). What happened when Galileo experienced a week-long service outage. *GPS World*.

European Union Agency for the Space Programme (2018). GNSS User Technology Report. `https://www.gsa.europa.eu/system/files/reports/gnss_user_tech_report_2018.pdf`. [Accessed on 23-Sep-2022].

Farhangian, F., Benzerrouk, H., and Landry, R. (2021). Opportunistic in-flight INS alignment using LEO satellites and a rotatory IMU platform. *Aerospace*, 8(10):280–281.

Farhangian, F. and Landry, R. (2020). Multi-constellation software-defined receiver for Doppler positioning with LEO satellites. *Sensors*, 20(20):5866–5883.

Fernández–Prades, C. (2022). Yocto Geniux. Online: https://github.com/carlesfernandez/yocto-geniux. [Accessed: 12-Sep-2022]. doi: 10.5281/zenodo.4743667.

Fernández–Prades, C., Arribas, J., and Closas, P. (2016a). Accelerating GNSS software receivers. In *Proc. 29th Int. Tech. Meeting Sat. Div. Inst. Navig.*, pages 44–61, Portland, OR. doi: 10.33012/2016.14576.

Fernández–Prades, C., Arribas, J., and Closas, P. (2016b). Assessment of software-defined GNSS receivers. In *Proc. 8th ed. of NAVITEC Conf.*, ESA/ESTEC, Noordwijk, The Netherlands. doi: 10.1109/NAVITEC.2016.7931740.

Fernández–Prades, C., Closas, P., and Arribas, J. (2013). Turning a television into a GNSS receiver. In *Proc. 26th Int. Tech. Meeting Sat. Div. Inst. Navig.*, pages 1492–1507, Nashville, TN.

Fernández–Prades, C., Vilà–Valls, J., Arribas, J., and Ramos, A. (2018). Continuous reproducibility in GNSS signal processing. *IEEE Access*, 6(1):20451–20463. doi: 10.1109/ACCESS.2018.2822835.

FGI (2022). The FGI-GSRx Software Defined GNSS Receiver Goes Open Source. https://www.maanmittauslaitos.fi/en/topical_issues/fgi-gsrx-software-defined-gnss-receiver-goes-open-source. [Accessed 11-Aug-2022].

Foerster, F. and Pany, T. (2013). Device and method for producing a data stream on the basis of data packets provided with packet sequence marks, and satellite receivers for providing the data stream; us patent no. 8451170, 2011-03-03; german patent no. 102008014981, 2009-10-15. US Patent 8,451,170.

Fokin, G. and Volgushev, D. (2022). Software-defined radio network positioning technology design. problem statement. In *Proceedings of Systems of Signals Generating and Processing in the Field of on Board Communications*, pages 1–6.

Gómez-Casco, D., López-Salcedo, J. A., and Seco-Granados, G. (2016). Generalized integration techniques for high-sensitivity GNSS receivers affected by oscillator phase noise. In *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pages 1–5. IEEE.

GPS World staff (2012). Innovation: Software gnss receiver - gps world : Gps world. https://www.gpsworld.com/software-gnss-receiver-an-answer-for-precise-positioning-research/. (Accessed on 07/04/2022).

Gunawardena, S. (2013). A Universal GNSS Software Receiver MATLAB® Toolbox for Education and Research. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)*, pages 1560–1576. Institute of Navigation.

Gunawardena, S. (2014). A universal gnss software receiver toolbox. *Inside GNSS*, pages 58–67.

Gunawardena, S. (2021). A High Performance Easily Configurable Satnav SDR for Advanced Algorithm Development and Rapid Capability Deployment. In *Proceedings of the 2021 International Technical Meeting of The Institute of Navigation (ION ITM 2021)*, pages 539–554. Institute of Navigation.

Gunawardena, S., Pany, T., and Curran, J. (2021). ION GNSS software-defined radio metadata standard. *NAVIGATION: Journal of the Institute of Navigation*, 68(1):11–20.

Gunawardena, S., Soloviev, A., and van Graas, F. (2004). Real time implementation of deeply integrated software GPS receiver and low cost IMU for processing low-CNR GPS signals. In *Proceedings of the 60th Annual Meeting of The Institute of Navigation (2004)*, pages 108–114.

Gunawardena, S., Soloviev, A., and Van Graas, F. (2007-2008). Wideband Transform-Domain GPS Instrumentation Receiver for Signal Quality and Anomalous Event Monitoring. *NAVIGATION: Journal of the Institute of Navigation*, 54(4):317–331.

Gunawardena, S. and van Graas, F. (2006). Split-sum correlator simplifies range computations in gps receiver. *Electronics Letters*, 42:1469–1471.

Gunawardena, S. and Van Graas, F. (2011). Multi-Channel Wideband GPS Anomalous Event Monitor. In *Proceedings of the 24th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2011)*, pages 1957–1968. Institute of Navigation.

Gunawardena, S., Zhu, Z., Uijt de Haag, M., and Van Graas, F. (2009). Remote-Controlled, Continuously Operating GPS Anomalous Event Monitor. *NAVIGATION: Journal of the Institute of Navigation*, 56(2):97–113.

Hamza, G., Zekry, A., and Motawie, I. (2009). Implementation of a complete GPS receiver using Simulink. *IEEE Circuits and Systems Magazine*, 9(4):43–51.

Hobiger, T., Gotoh, T., Amagai, J., Koyama, Y., and Kondo, T. (2010). A GPU based real-time GPS software receiver. *GPS solutions*, 14(2):207–216.

Honkala, S. (2016). GLONASS Satellite Navigation Signal Implementation in a Software-defined Multi-constellation Satellite Navigation Receiver. Master's thesis, Aalto University.

Humphreys, T., Psiaki, M., Kintner, P., and Ledvina, B. (2006). GNSS Receiver Implementation on a DSP: Status, Challenges, and Prospects. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2370–2382. Institute of Navigation.

Humphreys, T. E., Bhatti, J., Pany, T., Ledvina, B., and O'Hanlon, B. (2009). Exploiting multicore technology in software-defined GNSS receivers. In *Proceedings of the ION GNSS Meeting*, pages 326–338, Savannah, GA. Institute of Navigation.

Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W., and Kintner, Jr., P. M. (2008). Assessing the spoofing threat: Development of a portable GPS civilian spoofer. In *Proceedings of the ION GNSS Meeting*, Savannah, GA. Institute of Navigation.

Humphreys, T. E., Murrian, M. J., and Narula, L. (2020). Deep-urban unaided precise global navigation satellite system vehicle positioning. *IEEE Intelligent Transportation Systems Magazine*, 12(3):109–122.

IFEN GmbH (2022). SX3 GNSS Software Receiver — ifen.com. `https://www.ifen.com/products/sx3-gnss-software-receiver/`. [Accessed 01-Sep-2022].

Ikhtiari, N. (2019). Navigation in GNSS denied environments using software defined radios and LTE signals of opportunities. Master's thesis, University of Canterbury, Christchurch, New Zealand.

ION SDR Working Group (2020). Global Navigation Satellite Systems Software Defined Radio Sampled Data Metadata Standard, Revision 1.0. `https://sdr.ion.org`.

iPosi Inc. (2015). iPosi GNSS Signal Processing and Assistance; Performance. `https://iposi.com/technology/Performance`. [Accessed on 2022-09-26].

Jiménez-Baños, D., Blanco-Delgado, N., López-Risueño, G., Seco-Granados, G., and Garcia-Rodriguez, A. (2006). Innovative techniques for GPS indoor positioning using a snapshot receiver. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2944–2955.

Kang, T., Lee, H., and Seo, J. (2019). Analysis of the maximum correlation peak value and RSRQ in LTE signals according to frequency bands and sampling frequencies. In *International Conference on Control, Automation and Systems*, pages 1182–1186.

Kaplan, E. (1996). *Understanding GPS Principles and Applications*. Artech House Publishers.

Kassas, Z., Abdallah, A., Lee, C., Jurado, J., Duede, J., Hoeffner, Z., Hulsey, T., Quirarte, R., Wachtel, S., and Tay, R. (2022a). Protecting the skies: GNSS-less accurate aircraft navigation with terrestrial cellular signals of opportunity. In *Proceedings of ION GNSS Conference*. accepted.

Kassas, Z., Bhatti, J., and Humphreys, T. (2013). A graphical approach to GPS software-defined receiver implementation. In *Proceedings of IEEE Global Conference on Signal and Information Processing*, pages 1226–1229.

Kassas, Z., Khalife, J., Abdallah, A., and Lee, C. (2020). I am not afraid of the jammer: navigating with signals of opportunity in GPS-denied environments. In *Proceedings of ION GNSS Conference*, pages 1566–1585.

Kassas, Z., Khalife, J., Abdallah, A., and Lee, C. (2022b). I am not afraid of the GPS jammer: resilient navigation via signals of opportunity in GPS-denied environments. *IEEE Aerospace and Electronic Systems Magazine*, 37(7):4–19.

Kassas, Z., Khalife, J., Abdallah, A., Lee, C., Jurado, J., Wachtel, S., Duede, J., Hoeffner, Z., Hulsey, T., Quirarte, R., and Tay, R. (2022c). Assessment of cellular signals of opportunity for high altitude aircraft navigation. *IEEE Aerospace and Electronic Systems Magazine*. accepted.

Kassas, Z., Khalife, J., Shamaei, K., and Morales, J. (2017). I hear, therefore I know where I am: Compensating for GNSS limitations with cellular signals. *IEEE Signal Processing Magazine*, pages 111–124.

Kassas, Z., Morales, J., and Khalife, J. (2019). New-age satellite-based navigation – STAN: simultaneous tracking and navigation with LEO satellite signals. *Inside GNSS Magazine*, 14(4):56–65.

Kassas, Z., Neinavaie, M., Khalife, J., Khairallah, N., Haidar-Ahmad, J., Kozhaya, S., and Shadram, Z. (2021). Enter LEO on the GNSS stage: Navigation with Starlink satellites. *Inside GNSS Magazine*, 16(6):42–51.

Khalife, J. and Kassas, Z. (2022). On the achievability of submeter-accurate UAV navigation with cellular signals exploiting loose network synchronization. *IEEE Transactions on Aerospace and Electronic Systems*. accepted.

Khalife, J., Neinavaie, M., and Kassas, Z. (2022). The first carrier phase tracking and positioning results with Starlink LEO satellite signals. *IEEE Transactions on Aerospace and Electronic Systems*, 56(2):1487–1491.

Khalife, J., Shamaei, K., and Kassas, Z. (2018). Navigation with cellular CDMA signals – part I: Signal modeling and software-defined receiver design. *IEEE Transactions on Signal Processing*, 66(8):2191–2203.

Lapin, I., Granados, G., Samson, J., Renaudin, O., Zanier, F., and Ries, L. (2022). STARE: Real-time software receiver for LTE and 5G NR positioning and signal monitoring. In *Proceedings of Workshop on Satellite Navigation Technology*, pages 1–11.

Ledvina, B., Powell, S., Kintner, P., and Psiaki, M. (2003). A 12-Channel Real-Time GPS L1 Software Receiver. In *Proceedings of the 2003 National Technical Meeting of The Institute of Navigation*, pages 767–782. Institute of Navigation.

Ledvina, B., Psiaki, M., Humphreys, T., Powell, S., and Kintner, P. (2006a). A Real-Time Software Receiver for the GPS and Galileo L1 Signals. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2321–2333. Institute of Navigation.

Ledvina, B., Psiaki, M., Powell, S., and Kintner, P. (2004a). Bit-Wise Parallel Algorithms for Efficient Software Correlation Applied to a GPS Software Receiver. *IEEE Transactions on Wireless Communications*, 3(5):1469–1473.

Ledvina, B., Psiaki, M., Powell, S., and Kintner, P. (2006b). Real-Time Software Receiver; US patent no. 7010060, 2006-03-07.

Ledvina, B., Psiaki, M., Powell, S., and Kintner, P. (2007). Real-Time Software Receiver; US patent no. 7305021; 2007-12-04.

Ledvina, B., Psiaki, M., Sheinfeld, D., Cerruti, A., Powell, S., and Kintner, P. (2004b). A Real-Time GPS Civilian L1/L2 Software Receiver. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, pages 986–1005. Institute of Navigation.

Lightsey, G., Humphreys, T., Bhatti, J., Joplin, A., O'Hanlon, B., and Powell, S. (2014). Demonstration of a Space Capable Miniature Dual Frequency GNSS Receiver. *NAVIGATION, Journal of the Institute of Navigation*, 61(1):53–64.

Locubiche-Serra, S., López-Salcedo, J. A., and Seco-Granados, G. (2016). Statistical near-far detection techniques for GNSS snapshot receivers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6570–6574. IEEE.

LocusLock (2022). Reliable, accurate, and intelligent GPS LOCK. `https://locuslock.com`. [Accessed on 2022-09-30].

López-Salcedo, J., Capelle, Y., Toledo, M., Seco, G., Vicario, J. L., Kubrak, D., Monnerat, M., Mark, A., and Jiménez, D. (2008). DINGPOS: a hybrid indoor navigation platform for GPS and Galileo. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pages 1780–1791.

López-Salcedo, J. A., Parro-Jiménez, J. M., and Seco-Granados, G. (2009). Multipath detection metrics and attenuation analysis using a GPS snapshot receiver in harsh environments. In *2009 3rd European Conference on Antennas and Propagation (EuCAP)*, pages 3692–3696.

Lucas-Sabola, V., Seco-Granados, G., López-Salcedo, J. A., García-Molina, J., and Crisci, M. (2017). Efficiency analysis of cloud GNSS signal processing for IoT applications. In *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, pages 3843–3852.

Lucas-Sabola, V., Seco-Granados, G., López-Salcedo, J. A., García-Molina, J. A., and Crisci, M. (2016). Cloud GNSS receivers: New advanced applications made possible. In *2016 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6. IEEE.

Maaref, M. and Kassas, Z. (2020). Ground vehicle navigation in GNSS-challenged environments using signals of opportunity and a closed-loop map-matching approach. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2723–2723.

Maaref, M. and Kassas, Z. (2022). Autonomous integrity monitoring for vehicular navigation with cellular signals of opportunity and an IMU. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5586–5601.

Maaref, M., Khalife, J., and Kassas, Z. (2019). Lane-level localization and mapping in GNSS-challenged environments by fusing lidar data and cellular pseudoranges. *IEEE Transactions on Intelligent Vehicles*, 4(1):73–89.

Manzano-Jurado, M., Alegre-Rubio, J., Pellacani, A., Seco-Granados, G., López-Salcedo, J. A., Guerrero, E., and García-Rodríguez, A. (2014). Use of weak GNSS signals in a mission to the moon. In *2014 7th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, pages 1–8. IEEE.

Margaria, D., Linty, N., Favenza, A., Nicola, M., Musumeci, L., Falco, G., Falletti, E., Pini, M., Fantino, M., and Dovis, F. (2012). Contact! - first acquisition and tracking of IOV galileo signals. *Inside GNSS*, 7:45–55.

McEllroy, J. (2006). Navigation using signals of opportunity in the AM transmission band. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, USA.

McEllroy, J., Raquet, J., and Temple, M. (2006). Use of a software radio to evaluate signals of opportunity for navigation. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 126–133.

Minetto, A., Dovis, F., Vesco, A., Garcia-Fernandez, M., López-Cruces, À., Trigo, J. L., Molina, M., Pérez-Conesa, A., Gáñez-Fernández, J., Seco-Granados, G., et al. (2020). A testbed for GNSS-based positioning and navigation technologies in smart cities: The HANSEL project. *Smart Cities*, 3(4):1219–1241.

Ministry of Science and ICT of Korea (2021). Launch to Become the World's Seventh 'Space Powerhouse'! `https://www.msit.go.kr/bbs/view.do?sCode=eng&mId=4&mPid=2&bbsSeqNo=42&nttSeqNo=568`. [Accessed on 14-Sep-2022].

Mitola, J. (1995). The software radio architecture. *IEEE Communications Magazine*, 33(5):26–38.

Molino, A., Nicola, M., Pini, M., and Fantino, M. (2009). N-GENE GNSS software receiver for acquisition and tracking algorithms validation. In *European Signal Processing Conference*, pages 2171–2175.

Morales, J. and Kassas, Z. (2021). Tightly-coupled inertial navigation system with signals of opportunity aiding. *IEEE Transactions on Aerospace and Electronic Systems*, 57(3):1930–1948.

Morton, Y., Jiao, Y., and Taylor, S. (2015). High-latitude and equatorial ionospheric scintillation based on an event-driven multi-gnss data collection system. *Proc. Ionospheric Effect Sym., Alexandria, Va*.

Murrian, M. J., Narula, L., Iannucci, P. A., Budzien, S., O'Hanlon, B. W., Powell, S. P., and Humphreys, T. E. (2021). First results from three years of GNSS interference monitoring from low Earth orbit. *Navigation, Journal of the Institute of Navigation*, 68(4):673–685.

Nardin, A., Dovis, F., and Fraire, J. (2021). Empowering the tracking performance of LEO-based positioning by means of meta-signals. *IEEE Journal of Radio Frequency Identification*, 5(3):244–253.

Neinavaie, M., Khalife, J., and Kassas, Z. (2021). Acquisition, Doppler tracking, and positioning with Starlink LEO satellites: First results. *IEEE Transactions on Aerospace and Electronic Systems*, 58(3):2606–2610.

Nichols, H. A., Murrian, M. J., and Humphreys, T. E. (2022). Software-defined GNSS is ready for launch. In *Proceedings of the ION GNSS+ Meeting*.

Nicola, M., Motella, B., Pini, M., and Falletti, E. (2022). Galileo osnma public observation phase: Signal testing and validation. *IEEE Access*, 10:27960–27969.

NTLAB, UAB (2022). Unique ICs for GNSS Receivers. `https://ntlab.lt/`. [Accessed on 2022-09-26].

O'Hanlon, B. W., Psiaki, M. L., Powell, S., Bhatti, J. A., Humphreys, T. E., Crowley, G., and Bust, G. S. (2011). Cases: A smart, compact gps software receiver for space weather monitoring. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 2745–2753.

Orabi, M., Khalife, J., and Kassas, Z. (2021). Opportunistic navigation with Doppler measurements from Iridium Next and Orbcomm LEO satellites. In *Proceedings of IEEE Aerospace Conference*, pages 1–9.

Pany, T., Dötterböck, D., Gomez-Martinez, H., Hammed, M. S., Hörkner, F., Kraus, T., Maier, D., Sanchez-Morales, D., Schütz, A., Klima, P., and Ebert, D. (2019). The multi-sensor navigation analysis tool (MuSNAT) – architecture, LiDAR, GPU/CPU GNSS signal processing. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages 4087–4115. Institute of Navigation.

Pany, T., Eissfeller, B., Hein, G., Moon, S., and Sanroma, D. (2004a). IPEXSR: A PC based software GNSS receiver completely developed in Europe. In *Proc. ENC-GNSS 2004*.

Pany, T., Förster, F., and Eissfeller, B. (2004b). Real-time processing and multipath mitigation of high-bandwidth l1/l2 gps signals with a pc-based software receiver. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, pages 971–985.

Pany, T., Kaniuth, R., and Eissfeller, B. (2005). Deep integration of navigation solution and signal processing. In *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, pages 1095–1102.

Pany, T., Moon, S. W., Irsigler, M., Eissfeller, B., and Fürlinger, K. (2003). Performance assessment of an under-sampling swc receiver for simulated high-bandwidth gps/galileo signals and real signals. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 103–116.

Pany, T., Riedl, B., Winkel, J., Wörz, T., Schweikert, R., Niedermeier, H., Lagrasta, S., López-Risueño, G., and Jiménez-Baños, D. (2009). Coherent integration time: The longer, the better. *Inside GNSS*, 4(6):52–61.

Peng, S. and Morton, Y. (2011). A usrp2-based multi-constellation and multi-frequency gnss software receiver for ionosphere scintillation studies. In *Proceedings of the 2011 International Technical Meeting of The Institute of Navigation*, pages 1033–1042.

Pesyna, K., Kassas, Z., Bhatti, J., and Humphreys, T. (2011). Tightly-coupled opportunistic navigation for deep urban and indoor positioning. In *Proceedings of ION GNSS Conference*, pages 3605–3617.

Pesyna, Jr., K. M., Heath, Jr., R. W., and Humphreys, T. E. (2014). Centimeter positioning with a smartphone-quality GNSS antenna. In *Proceedings of the ION GNSS+ Meeting*.

Pinell, C. (2021). Receiver architectures for positioning with low Earth orbit satellite signals. Master's thesis, Lulea University of Technology, School of Electrical Engineering, Sweden.

Plotly (2022). Plotly JSON chart schema. `https://plotly.com/chart-studio-help/json-chart-schema/`. [Accessed on 09/30/2022].

PR Newswire (2021). New Trimble DA2 Receiver Boosts Performance of Trimble Catalyst GNSS Positioning Service. `https://www.prnewswire.com/news-releases/new-trimble-da2-receiver-boosts-performance-of-trimble-catalyst-gnss-positioning-service-301381160.html`. [Accessed on 2022-09-29].

Psiaki, M. (2006). Real-Time Generation of Bit-Wise Parallel Representations of Over-Sampled PRN Codes. *IEEE Transactions on Wireless Communications*, 5(3):487–491.

Psiaki, M., Humphreys, T., Mohiuddin, S., Powell, S., Cerruti, A., and Kintner, P. (2006). Searching for Galileo. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 1567–1575. Institute of Navigation.

Psiaki, M. and Slosman, B. (2022). Tracking digital FM OFDM signals for the determination of navigation observables. *NAVIGATION, Journal of the Institute of Navigation*, 69(2).

RF Micro Devices, Inc., Greensboro (2006). RFMD Announces Availability of the RFMD(R) GPS RF8110 Scalable GPS Solution. `https://ir.qorvo.com/node/12736/pdf`. [Accessed on 2022-09-29].

Rügamer, A., Förster, F., Stahl, M., and Rohmer, G. (2012). A Flexible and Portable Multiband GNSS front-end System. In *Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012)*, pages 2378–2389. Institute of Navigation.

Rügamer, A., Rubino, D., Lukcin, I., Taschke, S., Stahl, M., and Felber, W. (2016). Secure position and time information by server side PRS snapshot processing. In *Proceedings of the 29th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2016)*, pages 3002–3017.

Santana, G., de Cristo, R., and Branco, K. (2021). Integrating cognitive radio with unmanned aerial vehicles: An overview. *Sensors*, 21(3):830–856.

Seco-Granados, G., López-Salcedo, J. A., Jiménez-Baños, D., and López-Risueño, G. (2012). Challenges in indoor global navigation satellite systems: Unveiling its core features in signal processing. *IEEE Signal Processing Magazine*, 29(2):108–131.

Shamaei, K. and Kassas, Z. (2018). LTE receiver design and multipath analysis for navigation in urban environments. *NAVIGATION, Journal of the Institute of Navigation*, 65(4):655–675.

Shamaei, K. and Kassas, Z. (2021a). A joint TOA and DOA acquisition and tracking approach for positioning with LTE signals. *IEEE Transactions on Signal Processing*, pages 2689–2705.

Shamaei, K. and Kassas, Z. (2021b). Receiver design and time of arrival estimation for opportunistic localization with 5G signals. *IEEE Transactions on Wireless Communications*, 20(7):4716–4731.

Shamaei, K., Khalife, J., and Kassas, Z. (2018). Exploiting LTE signals for navigation: Theory to implementation. *IEEE Transactions on Wireless Communications*, 17(4):2173–2189.

Snyder, C., Feng, G., and Van Graas, F. (1999). GPS Anomalous Event Monitor (GAEM). In *Proceedings of the 55th Annual Meeting of The Institute of Navigation (ION AM 1999)*, pages 185–189. Institute of Navigation.

Söderholm, S., Bhuiyan, M., Thombre, S., Ruotsalainen, L., and Kuusniemi, H. (2016). A Multi-GNSS Software-defined Receiver: Design, Implementation, and Performance Benefits. *Annals of Telecommunications*, 71:399–410.

Soloviev, A., Gunawardena, S., and Van Graas, F. (2004). Deeply Integrated GPS/Low-Cost IMU for Low CNR Signal Processing: Flight Test Results and Real Time Implementation. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, pages 1598–1608. Institute of Navigation.

Song, Y. J., Lee, H. B., and Won, J. H. (2021). Design of multi-constellation and multi-frequency gnss sdr with fully reconfigurable functionality. *Journal of Positioning, Navigation, and Timing*, 10(2):91–102.

Souli, N., Kolios, P., and Ellinas, G. (2020). Relative positioning of autonomous systems using signals of opportunity. In *Proceedings of IEEE Vehicular Technology Conference*, pages 1–6.

Souli, N., Kolios, P., and Ellinas, G. (2021). Online relative positioning of autonomous vehicles using signals of opportunity. *IEEE Transactions on Intelligent Vehicles*, pages 1–1.

Souli, N., Kolios, P., and Ellinas, G. (2022). Adaptive frequency band selection for accurate and fast positioning utilizing SOPs. In *Proceedings of International Conference on Unmanned Aircraft Systems*, pages 1309–1315.

SPCOMNAV (2019). Cloud GNSS Receiver. `http://cloudGNSSrx.com`. [Accessed on 23-Sep-2022].

Stöber, C., Anghileri, M., Ayaz, A. S., Dötterböck, D., Krämer, I., Kropp, V., Won, J.-H., Eissfeller, B., Güixens, D. S., and Pany, T. (2010). ipexsr: A real-time multi-frequency software gnss receiver. In *Proceedings ELMAR-2010*, pages 407–416. IEEE.

Takasu, T. and Yasuda, A. (2009). Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. In *International symposium on GPS/GNSS*, volume 1. International Convention Center Jeju Korea.

Tang, G. and Peng, A. (2022). 5G receiver design based on downlink intermittent signals tracking algorithm. In *Proceedings of China Satellite Navigation Conference*, pages 462–471.

TeleOrbit GmbH (2022). MGSE REC: GNSS Radio Frequency Front-End. `https://teleorbit.eu/en/satnav/mgse-rec/`. [Accessed on 09/06/2022].

Teunissen, P. J. and Montenbruck, O., editors (2017). *Receiver Architecture*, pages 365–400. Springer International Publishing, Cham.

Thombre, S., Bhuiyan, M., Söderholm, S., Kirkko-Jaakkola, M., Ruotsalainen, L., and Kuusniemi, H. (2015). A Software Multi-GNSS Receiver Implementation for the Indian Regional Navigation Satellite System. *IETE Journal of Research*, pages 246–256.

Trimble Inc. (2005). Trimble Introduces Future-Ready GNSS Positioning Technology. `https://investor.trimble.com/news-releases/news-release-details/trimble-introduces-future-ready-gnss-positioning-technology`. [Accessed on 2022-09-29].

Trimble Inc. (2017). Trimble DA1 Catalyst GNSS Systems . `https://geospatial.trimble.com/products-and-solutions/trimble-da1`. [Accessed on 2022-09-29].

Troglia Gamba, M., Marucco, G., Pini, M., Ugazio, S., Falletti, E., and Lo Presti, L. (2015a). Prototyping a gnss-based passive radar for uavs: An instrument to classify the water content feature of lands. *Sensors*, 15(11):28287–28313.

Troglia Gamba, M., Nicola, M., and Falletti, E. (2015b). eNGene: An ARM based embedded real-time software GNSS receiver. In *28th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2015*, volume 4, pages 3178–3187.

Troglia Gamba, M., Nicola, M., and Motella, B. (2020a). Galileo osnma: An implementation for arm-based embedded platforms. In *2020 International Conference on Localization and GNSS, ICL-GNSS 2020 - Proceedings*, page 1–6.

Troglia Gamba, M., Nicola, M., and Motella, B. (2020b). Gps chimera: A software profiling analysis. In *Proceedings of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ 2020*, pages 3781–3793.

Tsui, J. B.-Y. (2000). *Fundamentals of Global Positioning System Receivers: A Software Approach*. Wiley-Interscience.

UniBwM (2022). MuSNAT — LRT 9. `https://www.unibw.de/lrt9/lrt-9.2/software-packages/musnat`. [Accessed on 07/04/2022].

Van Diggelen, F. (2009). *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House.

Wang, P., Wang, Y., and Morton, J. (2022). Signal tracking algorithm with adaptive multipath mitigation and experimental results for LTE positioning receivers in urban environments. *IEEE Transactions on Aerospace and Electronic Systems*, 58(4):2779–2795.

Wikipedia (2022). GNSS software-defined receiver. `https://en.wikipedia.org/wiki/GNSS_software-defined_receiver`. [Accessed 27-Jun-2022].

Wikipedia (2022). JSON streaming. `https://en.wikipedia.org/wiki/JSON_streaming`. [Accessed on 09/30/2022].

Yang, C., Arizabaleta-Diez, M., Weitkemper, P., and Pany, T. (2022). An experimental analysis of cyclic and reference signals of 4g LTE for TOA estimation and positioning in mobile fading environments. *IEEE Aerospace and Electronic Systems Magazine*, 37(9):16–41.

Yang, C. and Soloviev, A. (2018). Positioning with mixed signals of opportunity subject to multipath and clock errors in urban mobile fading environments. In *Proceedings of ION GNSS Conference*, pages 223–243.

Yang, C. and Soloviev, A. (2020). Mobile positioning with signals of opportunity in urban and urban canyon environments. In *Proceedings of IEEE/ION Position, Location, and Navigation Symposium*, pages 1043–1059.

Yoder, J. E. and Humphreys, T. E. (2022). Low-cost inertial aiding for deep-urban tightly-coupled multi-antenna precise GNSS. *Navigation, Journal of the Institute of Navigation*. To be published.

Zhang, X. (2022). GitHub - TMBOC/SoftGNSS: Current working ver. of SoftGNSS v3.0 for GN3sV2, GN3sV3, NT1065EVK, and NUT4NT samplers. `https://github.com/TMBOC/SoftGNSS`. [Accessed on 2022-09-09].

Zhao, C., Qin, H., and Li, Z. (2022). Doppler measurements from multiconstellations in opportunistic navigation. *IEEE Transactions on Instrumentation and Measurement*, 71:1–9.

Zhu, Z. and Van Graas, F. (2009). Earth-Surface Multipath Detection and Error Modeling for Aircraft GPS Receivers. *NAVIGATION: Journal of the Institute of Navigation*, 56(1):45–56.