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

Received: 16 March 2023 | Revised: 28 July 2023 | Accepted: 15 September 2023

DOI: 10.33012/navi.628

ORIGINAL ARTICLE

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

Thomas Pany[1] | **Dennis Akos**[2] | **Javier Arribas**[3] | **M. Zahidul H. Bhuiyan**[4] | **Pau Closas**[5] | **Fabio Dovis**[6] | **Ignacio Fernandez-Hernandez**[7] | **Carles Fernández–Prades**[3] | **Sanjeev Gunawardena**[8] | **Todd Humphreys**[9] | **Zaher M. Kassas**[10] | **José A. López Salcedo**[11] | **Mario Nicola**[12] | **Mark L. Psiaki**[13] | **Alexander Rügamer**[14] | **Young-Jin Song**[15] | **Jong-Hoon Won**[15]

[1] University of the Bundeswehr Munich, Neubiberg, Germany

[2] University of Colorado, Boulder, USA

[3] Centre Tecnològic de Telecomunicacions de Catalunya, Barcelona, Spain

[4] Finnish Geospatial Research Institute, Kirkkonummi, Finland

[5] Northeastern University, Boston, USA

[6] Politecnico di Torino, Turin, Italy

[7] European Commission, Brussels, Belgium

[8] Air Force Institute of Technology, Wright-Patterson Air Force Base, USA

[9] The University of Texas at Austin, Austin, USA

[10] The Ohio State University, Columbus, USA

[11] Universitat Autònoma de Barcelona, Cerdanyola del Vallès, Spain

[12] LINKS Foundation, Turin, Italy

[13] Virginia Tech, Blacksburg, USA

[14] Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany

[15] Inha University, Incheon, South Korea

Correspondence
Thomas Pany
University of the Bundeswehr Munich, Neubiberg/Germany.
Email: thomas.pany@unibw.de

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. This report highlights selected SDR implementations and achievements that are available to the public or that influenced the general development of SDR. Aspects related to the standardization process of intermediate-frequency sample data and metadata are discussed, and an update of the Institute of Navigation SDR Standard is proposed. This work focuses on GNSS SDR implementations in general-purpose processors and leaves aside developments conducted on field programmable gate array and application-specific integrated circuit 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. This report represents the knowledge of the authors but is not meant as a complete description of SDR history.

Keywords

GNSS, software-defined radio

1 | INTRODUCTION

Receiver development has always been an integral part of satellite navigation, ever since early studies were conducted for the U.S. Global Positioning System (GPS). The very first receivers were huge devices, enabling a correlation of received satellite signals with internally generated code and carrier replicas by a mixture of digital and analog electronics (Eissfeller & Won, 2017). Advances in semiconductor technology soon enabled signal processing on dedicated chips. This technology was complex to handle and was primarily located within the U.S. industry. Despite the success of GPS and its Russian counterpart globalnaya navigazionnaya sputnikovaya sistema (GLONASS), internal receiver technology was barely

accessible to the broader research community for a long time, as it seemed to be impossible to realize global navigation satellite system (GNSS) signal processing on low-cost computers. Even in 1996, a key receiver design pioneer expressed skepticism that general-purpose microprocessors were, or would ever be, a suitable platform for implementing a 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) resources required to run the algorithms in real time were encouraging (Akos & 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 application-specific integrated circuits (ASICs), but complemented these receivers, 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 GNSS signals. Whereas the data format of 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). The result was the so-called Institute of Navigation (ION) software-defined radio (SDR) Standard (ION SDR Working Group, 2020).

As technology evolved further, new GNSS software radios emerged, and some deficiencies of the ION SDR Standard became apparent (Clements et al., 2021). These conditions prompted the present paper, whose contributions are four-fold. First, it presents the first history of GNSS SDR development (Section 2). Second, it offers a detailed description of select GNSS SDRs (Section 3). Third, it overviews recent front-end developments (Section 4). Finally, it summarizes the history of the ION SDR Standard and proposes an update thereto (Section 5).

2 | GNSS SDR HISTORY

The history of GNSS SDR requires more than a bit of recollection, which can be fraught with inaccuracies, none of which are intentional in the present work. Corrections would always be welcome.

The roots of GNSS SDR can be traced 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 a high-fidelity simulation of the internal signal processing within GPS and GLONASS receivers. Dennis Akos, a Ph.D. student in the department, 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 Federal Aviation Administration (FAA)/National Aeronautics and Space Administration (NASA) Joint University Program. Meanwhile, publication of "The Software Radio Architecture" in 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 & Braasch, 1996).

Development of this initial simulation/implementation was significantly furthered through cooperation with Dr. James B. Y. Tsui of the 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 intermediate-frequency (IF) data that were critical for developing and debugging those early algorithms. Akos was responsible for the lower-level signal processing (acquisition and 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 s 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 and pushed advances in analog-to-digital converter (ADC) 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. In 1999, the first "real-time" operation was achieved, processing 60 s of IF data in 55 s. This was a notable achievement at the time, given that renowned GPS expert Philip Ward, who was responsible for some of the first GPS receivers, had recently expressed skepticism about the prospect of a fully software-defined real-time GPS SDR, writing "The integrate-and-dump accumulators provide filtering and resampling at the processor baseband input rate, which is around 200 Hz [... and] well within the interrupt servicing rate of modern high-speed microprocessors. But the 5- to 50-MHz rates [of intermediate frequency samples] would not be manageable" (Kaplan, 1996). This real-time implementation effort was headed by student Per-Ludvig Normark and led to the results published by Akos et al. (2001).

In the meantime, Kai Borre, a geodesy professor at Aalborg University, had also developed MATLAB code for GPS receivers in the mid–late 1990s. Borre's code focused on the navigation block and included functions for the conversion of coordinates and time references, satellite position determination, and atmospheric corrections. The joint efforts of Akos, Borre, and others would later lead to a well-known book (Borre et al., 2007), a primary 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, together with Cambridge Silicon Radio (CSR), to push GNSS to a price point acceptable for mobile phone adoption. CSR, a dominant supplier of Bluetooth hardware to the mobile phone market at the time, acquired NordNav in 2006. NordNav and CSR jointly redesigned the CSR 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 have been numerous contributions to GNSS SDR development since these early years, many of which are from the co-authors of this paper. Selected developments by the authors are outlined in Section 3, including a survey of achievements by other researchers in Section 3.11. The authors are aware that many other important contributions are missing and make no claims of establishing a comprehensive description. To provide the reader with a better orientation about the chronological order of all developments, we present Table 1, reiterating that the selection of references is partly subjective and that similar developments have often been carried out by several research groups. The timeline demonstrates the flexibility of SDR technology, i.e., the same code base is used for GPS L1 C/A code signals and for signals of opportunitys (SOPs) from cellular terrestrial transmitters or from communication satellites in low Earth orbits (LEOs).

TABLE 1
Timeline of GNSS SDR Developments

Year	Milestone with comment	Reference
1995	Emergence of software radio approach	(Mitola, 1995)
1996	First publication of a GPS SDR development	(Akos & Braasch, 1996)
1999	First real-time software receiver with GPS L1 C/A code	(Akos et al., 2001)
2000	First textbook on GPS SDR published	(Tsui, 2000)
2002+	Use of bit-wise correlation and SIMD instructions	(Ledvina et al., 2003; Pany et al., 2003)
2002+	GNSS SDRs as commercial products	NordNav, IFEN, Trimble, Locus Lock, etc.
2004	First multi-GNSS/multi-frequency GNSS SDRs	(Ledvina, Psiaki, Sheinfeld, et al., 2004; Pany, Eissfeller, et al., 2004)
2004	First real-time GNSS/INS integration with SDR	(Gunawardena et al., 2004)
2005	GNSS SDR consolidation at Politecnico di Torino and LINKS Foundation	Section 3.9
2005	Demonstration of vector tracking with a GNSS SDR	(Pany et al., 2005)
2006	First real-time all-in-view embeddable GNSS SDR	(Humphreys et al., 2006)
2006	First use of SDR technology for AM SOP	(McEllroy, 2006; McEllroy et al., 2006)
2007	Start of widespread adoption of SDR technology in GNSS research	(Borre et al., 2007)
2007	First development of a snapshot receiver	Section 3.8
2009	First multicore GNSS SDR	(Humphreys et al., 2009)
2010	Adoption of a computer science best-practice collaborative framework	Section 3.5
2010	First use of GPUs for correlation	(Hobiger et al., 2010)
2011+	Use of GNSS SDR for ionospheric research	(O'Hanlon et al., 2011; Peng & Morton, 2011)
2012+	SDR developments at the Finnish Geospatial Research Institute	(Borre, Fernández-Hernández et al., 2022; Söderholm et al., 2016)

(*Continued*)

TABLE 1 (*Continued*)

Year	Milestone with comment	Reference
2012	Use of a DVB-T ultra-low-cost front-end for GNSS SDR	Section 3.5
2012+	Use of SDR technology for LTE SOP	(del Peral-Rosado et al., 2013; Driusso et al., 2017; Shamaei et al., 2018)
2014+	Use of GNSS SDRs	(Lightsey et al., 2014; Murrian et al., 2021)
2014	Use of SDRs for mixed cellular 3G GSM/CDMA and digital television SOP	(Yang et al., 2014)
2015+	Abundance of processing power for GNSS SDR available	(Dampf et al., 2015; Nichols et al., 2022)
2017+	Use of SDRs for 3G CDMA and 4G LTE SOP	(Kassas et al., 2017)
2018	First use of Python for dedicated teaching of GNSS SDR	Section 3.7
2018	First SDR enabling sub-meter-level carrier-phase-based uncrewed aerial vehicle navigation with 3G CDMA and 4G LTE SOP	(Khalife & Kassas, 2018, 2022)
2020	Formal adoption of ION SDR Standard	Section 5
2020	Use of SDR for stationary positioning with multi-constellation Orbcomm and Iridium LEO SOP	(Farhangian & Landry, 2020; Orabi et al., 2021)
2021	First SDR for 5G SOP	(Shamaei & Kassas, 2021b)
2021+	Use of GNSS SDR to support development of new navigation satellite systems	(Miller et al., 2023; Song et al., 2021)
2021	First SDR enabling vehicle navigation with multi-constellation LEO SOP	(Kassas et al., 2023, 2021)
2022	First SDR enabling aircraft navigation with cellular SOP	(Kassas, Abdallah, et al., 2022; Kassas, Khalife, Abdallah, Lee, Jurado, et al., 2022)

3 | CURRENT STATUS OF GNSS SDRS

In June 2023, a quick internet search did not reveal any comprehensive listing of all GNSS SDRs. Wikipedia (2023) lists seven 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 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 (such as C or C++), typically optimized for run-time efficiency and stability

Teaching/research tools: Software packages written in a high-level programming language (such as 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. Each subsection is represented by one entry in Table 2 to give the reader a quick overview of the main characteristics of each development. Section 3.1 describes the work of Psiaki, Ledvina, and Humphreys and their efforts in real-time processing on DSPs, with the bit-wise-parallel approach proving to be highly successful, even for space applications. Section 3.2 covers the work of Pany and others in their efforts with multi-constellation/multi-frequency GNSS. Section 3.3 and Section 3.4 cover the efforts of Borre and others in a readable open-source MATLAB GPS SDR starting with Borre et al. (2007), with the most recent GNSS update reported by Borre, Fernández-Hernández et al. (2022). Akos has also continued this academic development of a suite of open-source GNSS SDRs (Bernabeu et al., 2022). The widely used open-source receiver GNSS-SDR is described in Section 3.5. The AutoNav receiver used to support the development of the Korean Positioning System (KPS) is discussed in Section 3.6, and Section 3.7 provides a discussion of PyChips, the basis for tutorial classes of the ION. The Universitat Autònoma de Barcelona (UAB) snapshot GNSS software receiver is described in Section 3.8, while Section 3.9 discusses an SDR used, e.g., in

TABLE 2
Overview of GNSS SDRs Discussed in Section 3

Name	Main language	Open source	Main focus
GRID	C++	No	Real-time operation of advanced algorithms on embedded devices
MuSNAT	C++	No	Analysis of navigation signal processing and algorithm prototyping
SoftGPS	MATLAB	Yes	Suite of GNSS SDRs with widespread use and accompanying book
FGI-GSRx	MATLAB	Yes	Multi-GNSS SDR with accompanying book
GNSS-SDR	C++	Yes	Real-time SDR with modular structure and widespread use
AutoNav SDR	MATLAB	No	Support for KPS development, API, and GPU
PyChips	Python	No	Multi-GNSS and optimized for use in teaching classes
UAB Snapshot GNSS Receiver	MATLAB	No	Snapshot receiver that can be operated in the cloud
NGene	ANSI C	No	Efficient GNSS SDR used in numerous Galileo-related projects
MATRIX	MATLAB, C++	No	Combined processing of GNSS with cellular 3G/4G/5G and LEO (Starlink, OneWeb, Orbcomm, Iridium, and Globalstar) signals

authentication schemes or reflectometry or to assess the influence of non-standard GNSS transmissions. Section 3.10 extends the scope of SDR to non-GNSS signals.

At the beginning of GNSS SDR development, different receivers were linked to specific persons or research institutes; in contrast, today different receivers, tools, or code bases are often used at the same institute. Moreover, code bases first developed by a single institute have spread into different institutes. For example, the developments of Borre et al. (2007) forked into several branches (see, e.g., the work by FGI (2022), Bernabeu et al. (2022), and Zhang (2022)), as discussed in Section 3.3 and Section 3.4.

3.1 | Bit-Wise Parallelism and the Emergence of GRID

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 two years when he started to wonder whether the slow MATLAB operations could be translated to run in real time on a general desktop workstation. A bottleneck in GNSS digital signal processing occurs during the performance of 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 to achieve 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 of effort (Ledvina et al., 2003).

The main aim 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 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, the channel generates a 1-bit representation of 32 successive samples of its prompt pseudorandom noise (PRN) code replica and stores them in parallel in a single 32-bit unsigned integer word. The channel also generates an early-minus-late PRN code replica that requires 1.5 bits per sample, which utilizes 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 the accumulation of results in the 32-bit words. This operation involves sets of bit-wise Boolean operations, as per Ledvina et al. (2003), followed by a summation of the number of 1-bits in the resulting 32-bit unsigned integer words. Minimizing the execution time of the bit summation operations

proved to be a challenge. 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. To ensure a reasonable table size, only the bits in a 16-bit unsigned integer word were counted. The original receiver's 32-bit words were split in half, two table look-ups were performed, and the results were summed in order to count all of the 1-bits. The original algorithms were defined by Ledvina et al. (2003), Ledvina, Psiaki, Powell et al. (2004), and Ledvina, Psiaki, Powell et al. (2006).

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

A processor that can operate on wider segments of data, up to 512 bits for current single instruction multiple data (SIMD) instructions, gains substantial additional increases in signal processing speed (Nichols et al., 2022). However, 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 of the 1-bits in a word. This "popcount" intrinsic obviates the table look-ups that counted 1-bits in the original bit-wise 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.

After successfully running the basic algorithms in real time using 32-bit words, the Cornell group showcased the efficacy of real-time GNSS software radio by using the same techniques to develop a dual-frequency L1 C/A and L2C receiver (Ledvina, Psiaki, Sheinfeld, et al., 2004) and a GPS/Galileo L1 civilian receiver (Ledvina, Psiaki, Humphreys, et al., 2006). These real-time software GNSS receivers each required only several person-days to be developed 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. This requirement led to a supporting effort that successfully deduced the E1 PRN codes of the Galileo in-orbit validation (IOV) satellite GIOVE-A by recording their raw RF front-end samples and post-processing those samples using a suite of custom-designed SDR signal-processing algorithms in order to extract the chips from 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 (UT)

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, the C++-based UT Austin receiver is now a highly optimized science-grade multicore GNSS SDR (Humphreys et al., 2009; Nichols et al., 2022). Its main features are summarized in Table 3. This receiver 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 LEOs (Murrian et al., 2021), and the basis of the current state of the art in urban precise (decimeter-level) positioning (Humphreys et al., 2020; Yoder & Humphreys, 2023). As detailed by Nichols et al. (2022), GRID 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.

TABLE 3
Main Features of GRID

	GRID	
Feature	**Solution**	**Remark**
Operating system	GNU/Linux, macOS, Windows	
Programming environment	C++	
IF sample file input source	A wide array of formats	Will accommodate proposed ION SDR Standard
Real-time sample input	Yes	See Nichols et al. (2022)
Additional sensors	IMU, cameras, lidar	Requires PpEngine module
Supported GNSS	GPS, Galileo, BeiDou, SBAS, QZSS, CDMA	Nearly all open spreading codes and navigation message streams supported
Acquisition	Multi-threaded and FFT-optimized	
Tracking	Vectorized, multicore, Intel SIMD (SSE2 through AVX-512) and ARM NEON (64-bit and 128-bit) accelerations	Correlation no longer the primary bottleneck under some configurations; see Nichols et al. (2022)
Measurement output	All standard GNSS observables	Proprietary GBX format plus RINEX, NMEA, RTCM, MATLAB MAT-file, KML
Navigation	Extended Kalman filter based on pseudorange and Doppler measurements	Carrier-phase-based positioning available with PpEngine module
Further features	Vector tracking, multi-antenna, IMU integration, space-ready, interference mitigation and detection	
Availability	Source code available via commercial license from UT Austin	Turnkey solutions available via Locus Lock

Note: SBAS: satellite-based augmentation system

3.2 | Multi-Sensor Navigation Analysis Tool

The Multi Sensor Navigation Analysis Tool (MuSNAT) is an object-oriented but monolithic C++ software receiver maintained by the University of the Bundeswehr Munich (UniBwM) and was first mentioned in its present form by Pany et al. (2019). MuSNAT started as an operational real-time receiver development, but currently, it mostly serves to develop and demonstrate innovative signal-processing and navigation algorithms. MuSNAT is also used for teaching. It is freely available as executable for academic purposes from UniBwM (2023). Its main characteristics can be found in Table 4. In contrast to the bit-wise approach of Section 3.1 (which allows the design of 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/Advanced Micro Devices central processing units (CPUs). This allows samples to be represented as 8-bit or 16-bit values, and SIMD instructions such

TABLE 4
Main Features of MuSNAT

	MuSNAT	
Feature	**Solution**	**Remark**
Operating system	Windows 10/11	Compiles as GUI or as command-line version (port of command-line version to Linux under preparation)
Programming environment	Microsoft Visual Studio 2019 C++	CUDA, Intel OneAPI, vcpkg, and .net for GUI
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, etc.)	Nearly all open spreading codes available with at least one navigation message decoder for each system
Acquisition	Optimized FFT method	CPU and GPU supported
Tracking	Dot-product from Intel Performance Primitives (CUDA version for massive multi-correlator applications)	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, RTKLIB	Support of Galileo OSNMA/HAS and synthetic aperture processing via MATLAB interface
Availability	Executable plus data visualizer downloadable via UniBwM (2023)	Source code available for research projects with UniBwM

Note: OFDM: orthogonal frequency-division multiplexing

as AVX-512 currently allow processing of registers of up to 512 bits (i.e., 32 16-bit samples) in parallel.

GNSS software receiver developments were initiated at UniBwM in 2002, after it became clear that the software radio approach discovered by Akos would provide useful insights into GNSS receiver technology and would thus be indirectly helpful in designing and building the Galileo navigation satellite system. The first software receiver at UniBwM was designed for GPS L1 C/A only and was realized as a MATLAB/Simulink project for post-processing. To sample the GNSS signals, a commercial ADC with a peripheral component interconnect express (PCIe) connector from National Instruments (NI) was used (PXI 5112), which was connected either to a low-bandwidth GPS L1 C/A code front-end based on the Plessey GP 2010 RF chip set or later to one GPS L1/L2 high-bandwidth front-end, which was specifically developed by Fraunhofer IIS (Pany, Förster, et al., 2004). Soon after, the software for communicating with the ADC (written in C++, making use of the Microsoft Foundation classes) was upgraded to a full GPS L1 C/A plus L2CS (only medium-length L2 code was supported, not the long code) receiver. A detailed analysis published by Pany et al. (2003) revealed that both the SIMD instruction set and the size and structure of the CPU caches were important for real-time capabilities. Memory bandwidth is a key issue 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 these results, funding was secured to support a group of five researches over three years. This funding allowed the researchers to start a new software receiver project, this time making full use of C++ features for object-oriented development, and to develop a graphical user interface (GUI) connected to the processing core via a clearly defined interface that also allowed the core to run without a 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. A concise overview of the development during those years was written by Stöber et al. (2010), who described the improvements compared with the start of the project lain down by Pany, Eissfeller, et al. (2004).

A loose cooperation with IFEN GmbH was initiated, which eventually resulted in the SX3 receiver (IFEN GmbH, 2022). IFEN used the processing core as an 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 with the graphics processing unit (GPU) (Pany et al., 2012). Semi-codeless tracking of GPS L2P(Y) (i.e., P-code aided cross-correlation) was also implemented. The cooperation of UniBwM with IFEN lasted until 2013, when the development directions started to diverge. IFEN used the software primarily as a 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 from a commercial viewpoint.

The focus at UniBwM changed in 2017, as the old GUI could no longer be maintained. Furthermore, real-time operation became less important, as most scientific results were obtained in post-processing. Consequently, a new GUI was developed and attached to the proven processing core. Any run-time optimizations within the processing core that degraded the navigation performance (i.e., mostly causing additional noise in the code tracking loop) were removed. The core's logging output was directed to an SQL database to store all types of intermediate results in a

single file (in addition to the legacy ASCII logging into multiple files). A dedicated visualization tool was developed for this database.

The use of Windows and Visual Studio for developing a software radio is slightly unusual, but can be 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 performed for Windows only. In terms of numerical performance and code optimization, with the Intel C++ compiler and the Intel Performance Primitives, Intel provided and still provides the same quality on Windows as for Linux. Over the years, however, it became 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 already provided an impressive processing capability in 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 look-up tables were used, and a highly efficient correlator implementation with the Intel x86 pmaddubsw instructions was based on a signal sample representation as unsigned integers (including the necessary rewriting of the correlation formulae because of 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. With the use of FFT libraries provided by NVIDIA, porting the acquisition code porting from the CPU to GPU became comparably easy. The situation is different for signal tracking. The tracking code has been transferred to the GPU, and some optimization has been applied to minimize the amount of data transfer between the CPU and GPU. However, because 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 generates significant overhead. Thus, GPU-based tracking is 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, over the years, the use of dedicated assembler code required continuous adaptation to new CPU instruction sets (e.g., from SSE to AVX instructions). The performance gained by using hand-coded assembler routines instead of using the libraries provided by Intel (Intel Performance Primitives) 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 Intel Performance Primitives 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 (Dötterböck et al., 2023), vector tracking and deep GNSS/inertial navigation system (INS) coupling, support for long term evolution (LTE)/5G signals, and GNSS signal simulation. Notably, the maintenance of the huge C++ code base of MuSNAT at a university institute with a high fluctuation of researchers is demanding. The learning curve for good C++ development in this context is steep and is often inefficient for the purposes of obtaining a PhD degree. Therefore, interfaces from the C++ code to MATLAB

were established; for example, open service navigation message authentication (OSNMA) decoding, precise point positioning computation for high accuracy service (HAS), and lidar odometry have been 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). Bochkati et al. (2023) used this for ease in developing synthetic aperture algorithms.

UniBwM 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 required 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 types of development efforts, the decision was made to connect front-ends via TCP/IP. This approach is powerful in terms of bandwidth and is also generic; a first version of this approach has been described by 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) front-ends. At the time of writing this paper, a more efficient firmware for USRPs with direct field programmable gate array (FPGA) programming is being developed, which will allow data to be synchronously captured from an inertial measurement unit (IMU) together with GNSS signal samples.

3.3 | SoftGPS, SoftGNSSv3.0, and Derivatives

As mentioned above, the work by Borre et al. (2007) and the associated MATLAB receiver provided a cornerstone for GNSS SDR development. This receiver, initially called SoftGPS and then SoftGNSS (usually referred to as SoftGNSSv3.0), included the basic processing functions for GPS L1 C/A in a readable format and was useful for educational purposes. These functions included signal FFT-based acquisition, frequency, carrier phase, and code phase tracking, data synchronization and demodulation, pseudorange generation, and eventually PVT estimation. 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. In addition to Borre and Akos, SoftGNSS included relevant contributions by 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, as described below.

- A new SDR GNSS book (Borre, Fernández-Hernández 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 the GNSS software receiver (GSRx) developed at the Finnish Geospatial Research Institute (FGI) (FGI-GSRx), described in the following section, but the book also includes other MATLAB receivers. In particular, the dual-frequency GSRx (DF-GSRx), developed by Borre's PhD student P. Bolla, is a dual-frequency GPS L1/L5 receiver that includes dual-frequency acquisition techniques, measurement combination (including ionosphere-free measurements), and positioning. The book also includes a GPS L1 C/A snapshot receiver developed by Borre's former PhD student I. Fernandez-Hernandez, which is more modest than that described in Section 3.8, 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 for diving into the basic functions of GNSS receivers, such as calculating satellite positions from the ephemerides, performing 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. (2022), as above mentioned, have provided a collection of open-source SDRs developed at University of Colorado Boulder based on SoftGNSS.
- Zhang (2022) has provided a repository with adaptations of SoftGNSS for different front-ends.

3.4 | Finnish Geospatial Research Institute's Multi-GNSS Software Receiver

The software receiver developed by the FGI is known as the FGI-GSRx. The development of the FGI-GSRx started in 2012 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 satellites (GIOVE A and GIOVE B) from the European GNSS system, Galileo. Since then, the researchers at FGI have been continuously developing new capabilities for 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., 2015; Bhuiyan et al., 2014), 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 over the last decade as a research platform in different national and international research and development projects to develop, test, and validate novel receiver processing algorithms for robust, resilient, and precise positioning, navigation, and timing. 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 of the navigation message, pseudorange estimation, and PVT estimation. The software architecture is built such 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 those interested in learning about GNSS receivers. Some of the main features of FGI-GSRx are listed in Table 5.

The software receiver was released as open source in February 2022 (FGI, 2022). FGI-GSRx was also accompanied by the book *GNSS Software Receivers*, a next edition of one of the fundamental GNSS textbooks, published in 2022 by Cambridge University Press (Borre, Fernández-Hernández et al., 2022). This book systematically introduces software receiver processing functionalities, with experimental results for the GPS L1 C/A signal in Section 2 (Borre, Bhuiyan et al., 2022), GLONASS L1OF signal in Section 3 (Bhuiyan, Honkala et al., 2022), Galileo E1 OS signal in Section 4 (Bhuiyan, Söderholm, Ferrara et al., 2022), BeiDou B1I signal in Section 5 (Bhuiyan, Söderholm, Thombre et al., 2022), NavIC L5 signal in Section 6 (Thombre et al., 2022), and a single-frequency multi-constellation solution with three GNSS signals in Section 7 (Söderholm et al., 2022). The readers can easily follow the fundamental receiver processing chain for each individual GNSS signal,

TABLE 5
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 another operating system that can host MATLAB or OCTAVE.
Programming environment	MATLAB	Executes in MATLAB 2019 or any later version. The software receiver can be also executed in OCTAVE.
IF sample file input source	ION SDR Standard	Reads input data files following the ION SDR Standard.
Processing mode	Only operates as a post-processing GNSS receiver	Can read raw IF data for complete receiver processing or can load previously saved acquisition and/or tracking data in order to skip acquisition and/or tracking operations to be able to process the navigation solution depending on the 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-sensitivity 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 squares	Users can select the signal-to-noise ratio or elevation cut-off mask in order to select the satellites that contribute to the position computation.

with the distinctive changes among those signals discussed and highlighted in figures. One noteworthy contribution of this book is a method for integrating several GNSS signals to form a single-frequency multi-GNSS PVT solution, presented in Section 7.

The FGI-GSRx can be utilized at universities and other research institutes as a tool for training graduate-level students and early-stage researchers and for providing hands-on experience in GNSS receiver development. This receiver 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.

3.5 | GNSS-SDR, an Open-Source Software-Defined GNSS Receiver

The software receiver developed by the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), named GNSS-SDR (but not related to the ION SDR Standard), is another example of a multi-band, multi-system receiver. This receiver has been constantly evolving since 2010, keeping pace with the newest GNSS algorithms and signals over more than a decade. The GNSS-SDR originated as a by-product of a CTTC research staff initiative, with the aim of providing a framework for collaborating with other researchers seeking to accelerate research and development of software-defined GNSS receiver technology. The receiver focuses on baseband signal processing, although it has the ability to run a navigation engine (refer to Table 6). The early stages of development progressed 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 were considered from the start.

The GNSS-SDR first became popular in August 2012, with reports of GNSS usage via extremely cheap (about $25) digital video broadcast-terrestrial (DVB-T) receivers based on Taiwan's Realtek RTL2832U chipset, sold in the form of USB

TABLE 6
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 that high-quality coding standards are met.
Processing mode	Real-time and post-processing	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 SDR 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, phase-locked loop (PLL), frequecy-locked loop (FLL). High-dynamics capabilities. SIMD-accelerated in both i686 and ARM CPUs (see Fernández–Prades et al. (2016a)).
Navigation	Traditional least squares, code- and carrier-based positioning modes	Positioning engine based on RTKLIB implementation (Takasu & Yasuda, 2009). All possible supported GNSS signal combinations are allowed.

dongles that allow users to watch over-the-air DVB-T European broadcast television on their PCs. These devices typically send partially decoded MPEG transport frames over the USB; however, by exploiting an undocumented mode of operation of the demodulator chip, the user was able to obtain raw in-phase and quadrature samples, stream them through the USB to a PC, and then apply the GNSS-SDR software processing, turning the DVB-T receiver into a GNSS receiver and delivering the position in real time (see Fernández–Prades et al. (2013)). In a parallel development, in November 2013, the European Space Agency (ESA) acknowledged GNSS-SDR as one of the first 50 receivers worldwide to achieve a successful Galileo position fix.

The project gained momentum and maturity over the years, and it currently has a solid and valuable user base that continuously provides 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 publicly and privately funded research projects (including the European Union Agency for the Space Programme [EUSPA], ESA, National Science Foundation [NSF], and NASA activities, as well as educational programs such as Google's 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) provided an 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).

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 available 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 (ranging from those released in 2014 to the most recent releases), 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), as well as position fixes in application-specific messages (*e.g.*, NMEA 0183), a variety of geographic information system-oriented file formats (KML, GeoJSON, GPX), and custom binary outputs that allow the observability of internal signal-processing subproducts.

3.6 | AutoNav SDR

The AutoNav SDR is a MATLAB-based multi-GNSS, 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 presented in Table 7. The critical point considered in the design phase of this SDR is the maximization of reconfigurability. Because 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 required. The AutoNav SDR is designed to provide full reconfigurability in terms of target signal combinations and signal characteristics, especially for the easy addition of new signal proposals. To achieve this, a basic

TABLE 7
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 the Parallel Computing Toolbox in MATLAB
Tracking	MEX correlator	18/8-bit code/carrier replica tables, 32-bit code/-carrier NCOs, bit shift operations
Further features	API, easy addition of new signals, RINEX observation logging, RF interference mitigation based on pulse blanking, direct state-tracking Kalman filter	

framework of software receiver was designed with an appropriate processing functional architecture and data structure in consideration of signal expandability. This framework was then applied to realize an SDR for GPS L1 C/A code signal as a first realization example by reconfiguring a configuration file via a 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 quickly added 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 MATLAB-based IF-level GNSS/KPS simulator can be ideally suited to test the navigation performance of any GNSS signal as well as new KPS signals by reconfiguring signal design parameters via a 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. Hence, 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 simple way using the Parallel Computing Toolbox of MATLAB. If the GPU is usable, local variables for the correlation (i.e., code and carrier replicas) are generated in the GPU memory using the gpuArray function. Then, FFT, inverse FFT, and correlations are automatically performed in the GPU. Finally, the correlation results are extracted via a gather function. With this simple approach, the execution time is reduced by a factor of approximately 2.12 compared with the general CPU-based acquisition, without the relatively complex development using CUDA.

Because the most time-consuming process of the receiver is the correlation 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 that 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 in 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; thus, the indices of the tables for current code and carrier replica generation are calculated via bit shift operations of 14 bits and 24 bits, respectively. With these implementations, the overall execution time became much faster (approximately five-fold faster) than the original double precision-based code, but it still cannot operate in real time. Currently, Inha University is developing an FPGA-based real-time GNSS receiver in which only the correlator is substituted by the FPGA board at the original AutoNav SDR.

To further enhance its flexibility, the AutoNav SDR also provides APIs at each part of the signal processing chain (such as the 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 in a similar manner with the dynamic link library (DLL). Because 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 code.

3.7 | 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 the Wideband Transform-domain Instrumentation GNSS Receiver (TRIGR) (see Section 5) and the ChameleonChips GNSS SDR Toolbox for MATLAB (Gunawardena, 2014).

One of the key promises of SDRs is their flexibility and, hence, their 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 a 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 assumption of programming language proficiency may not always hold true. Further, depending on the situation, it may be more efficient and beneficial for graduate students to make deeper progress on their research instead of spending time in becoming programming language experts. PyChips was developed from the ground up to support this notion. A more detailed introduction to PyChips has been provided by Gunawardena (2021). The main features of PyChips are summarized in Table 8. This receiver 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. The simulation portion of PyChips (comprising numerous source objects) synthesizes these signals at the sample level onto one or more sample streams that are grouped into objects called stream

TABLE 8
Main Features of PyChips

PyChips		
Feature	**Solution**	**Remark**
Operating system	Windows x64	Due to pre-compiled C/C++ bindings that currently use the Windows API for file reading and threading. Linux support under development.
Programming environment	SARDL, Python, C/C++	SDR entirely specified using JSON-based SARDL. Assembles pre-built configurable Python and C/C++ objects at run-time according to user SARDL specifications.
IF sample file input source	ION SDR 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. Code replicas specified as an assembly of sequence objects (static, subcarrier, overlay, mux, etc.; see Gunawardena (2021)).
Acquisition	FFT-based generic acquisition engine with configurable coherent and noncoherent integration settings	Automatically detects and implements circular vs. non-circular frequency-domain correlation based on code length.
Tracking	Generic tracking module assembled from configurable functional blocks (e.g., carrier wipe-off, code replica, correlator, gearbox, accumulator, etc.) and a generic controller object—all defined in SARDL	Employs split-sum correlation (Gunawardena & van Graas, 2006). Always operates on a 1-ms block of samples and retires the current block before operating on the next block (no sample shifting to align with the SV time of transmission). Direct initialization of tracking objects configured for other signals from the same SV (e.g., GPS L1 C/A to L1C, L2C, and L5).
Measurement output	Yes	CSV format.
Availability	Versions distributed at ION conference tutorials	Versions used at ION tutorials.

containers. A stream container is an abstraction of a satnav receiver's antenna(s) and RF front-end subsystem. The stream container can be multi-frequency or multi-element, can have different sample rates and bandwidths, and can have an IF or baseband sampling architecture and any and all combinations thereof. If the use case involves live-sky signal processing, 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 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.

A unique feature of PyChips is that all of the functionalities described above are defined/specified by a draft SDR language called Signal and ARchitecture Description Language (SARDL). SARDL is implemented as a grouping of JavaScript object notation (JSON) files. Current and next-generation advanced satnav signal structures and the receiver architectures used to process them are constructed by assembling pre-built low-level functional blocks. For example, as described by 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 SARDL with a vast number of diverse signal specifications, use cases, and applications in order to explore the concept of a "satnav signals and systems specification language." Today, the reference SDR that implements SARDL is written in Python and is therefore called PyChips. However, the ultimate goal of this effort is to contribute toward satnav SDR implementations that have the performance, power efficiency, and scalability of ASICs with the flexibility, reconfigurability, adaptability, and ease of use of software.

3.8 | UAB Snapshot GNSS Software Receiver

The UAB snapshot GNSS software receiver (cf. Table 9) was originally developed as part of the research activities on indoor GNSS positioning carried out by the Signal Processing for Communications and Navigation (SPCOMNAV) group at UAB in 2007. At that time, the group was involved in one of two parallel contracts awarded by the ESA 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 Wi-Fi, ultra wideband, 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 would be able to operate under the extremely weak signal conditions experienced indoors. This implementation involves working with attenuation losses of 10–40 dB, which drive the effective carrier power to noise power spectral density, i.e., C/N_0, down to values for which conventional GNSS receivers cannot operate.

The proposed HS-GNSS receiver implementation was based on a snapshot architecture in which a batch of input samples is processed at one time to provide the user's position. This approach is often referred to in the literature as "push-to-fix" or "acquisition-only" because no tracking stage is actually implemented at the receiver. Consequently, the receiver operates in open-loop mode by providing at its output the observables obtained directly 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 of the ESA in The Netherlands, where he was leading research activities concerning indoor GNSS and snapshot GNSS receivers. The core of the

TABLE 9
Main Features of the 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 noncoherently combining a set of coherent correlations. 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	Coarse-time navigation is implemented.
Further features	Implements near–far detection and interference detection	

Note: 3GPP: Third-Generation Partnership Project; RRLP: radio resource location services protocol

UAB snapshot GNSS receiver was inspired by the concept of double-FFT acquisition introduced by Jiménez-Baños et al. (2006). This algorithm uses two consecutive FFT operations to implement the correlation of the received signal with the local code replica and then the simultaneous estimation of the fine Doppler frequency and bit synchronization. Readers interested in the double-FFT algorithm and a detailed description of the UAB snapshot GNSS receiver implementation are referred to the 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 indoor working conditions. Nevertheless, the implementation is flexible and does not prevent the receiver from operating efficiently in other scenarios, such as outdoors. For an indoor environment, the most important impairment to be counteracted is the severe attenuation due to propagation through building materials and other obstacles. Attenuation of up to 40 dB can easily be experienced, thus requiring specific action to recover as much of the lost power in order to still be able to detect GNSS satellites. Because the received energy is the most important parameter from a signal detection and estimation viewpoint and because energy is simply the power multiplied by the observation time, the only way to compensate for an extremely weak received power is to increase the observation time. Thus, a longer piece of received signal must be processed, which requires 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 navigation message data symbols, residual Doppler errors, and clock instabilities. Consequently, 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 sufficiently long, 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 enabling the receiver to detect a few GNSS satellites indoors. An interesting discussion on the importance of sufficiently long coherent integrations has been provided by Pany et al. (2009).

The correlation between the received signal and the local replica is the most important operation of a snapshot GNSS receiver because, with such correlation, the most accurate code delay and Doppler observables must be estimated. Here, no tracking stage is implemented, and thus, there is no opportunity to further refine these observables in subsequent stages of the receiver. For this reason, the correlation must be implemented in the most optimal way, taking into account subtle details that might be ignored in conventional GNSS receiver implementations. Such optimality is achieved by the double-FFT algorithm implemented in the UAB snapshot GNSS receiver, which applies an optimal joint estimation of the code delay and fine Doppler over a long period of time, where potential sign transitions may occur because of 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., have been reported by Seco-Granados et al. (2012).

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

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 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 similar to the indoor case (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.

Migration of the UAB snapshot receiver into a cloud-based implementation was a major milestone that attracted the interest of the community and opened the door for new applications and use cases. The interest in cloud GNSS positioning was motivated by the fact that, at that time, GNSS software receivers were running

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 was well suited for a snapshot GNSS receiver implementation, in which a batch of samples could be sent to a remote server where the user's position would then be computed via the same tools as in any other snapshot GNSS receiver. These computations include using assisted GNSS (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 concept motivated the so-called "cloudGNSSrx," the cloud-based implementation of the UAB snapshot GNSS receiver, as described by 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 was built on AWS to automate and scale the remote execution of the receiver, and an API was developed for machine-to-machine communication, facilitating the provision of GNSS positioning to small IOT sensors (Lucas-Sabola et al., 2016). In this way, IOT sensors requiring GNSS positioning were able to offload most of the computational load to a remote server, thus significantly reducing the power consumption and extending their battery lifetime.

Low-power GNSS positioning is one of the main applications of cloud GNSS software receivers, because, for snapshots shorter than a few tens of milliseconds, the energy spent in sending the GNSS samples to the cloud is balanced by the significant energy saved at the user's terminal for not processing such samples and for performing such processing at the cloud instead (Lucas-Sabola et al., 2017). This feature was acknowledged by the former European GNSS Agency (GSA), now the EUSPA, who identified the UAB cloud GNSS receiver as promising technology 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 significantly improved the initial prototype and produced 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, such as advanced signal-monitoring techniques, spoofing detection, or authenticated/certified positioning, the latter being reported by Rügamer et al. (2016). Thus, there is a brilliant future ahead for cloud GNSS software receivers with many exciting new applications to come.

3.9 | The NGene Family of Receivers at Politecnico di Torino and LINKS

The development of NGene, a GNSS software receiver, originated at Politecnico di Torino and the LINKS Foundation in the early 2000s. At that time, the Navigation Signal Analysis and Simulation Group was already engaged in software implementation of various sections of GNSS baseband processing. This endeavor capitalized on the group's extensive expertise in digital signal processing and specifically in simulating complex communication systems.

Efforts initially focused on optimizing the acquisition and tracking stages, both as post-processing tools and as core processing units on programmable hardware. In 2005, under regional funding, the research team, partially affiliated with the Istituto Superiore Mario Boella (now part of the LINKS Foundation), commenced the development of a fully software-based, real-time GNSS receiver for GPS and forthcoming Galileo signals.

The outcome of the work was the software receiver NGene, as documented by Molino et al. (2009). NGene demonstrated real-time processing capabilities for GPS, Galileo, and European Geostationary Navigation Overlay Service signal components transmitted on the L1/E1 band. Prior to processing, the signals were subjected to IF downconversion and digitalization by an external analog front-end device. Communication between the front-end device and the software receiver occurred via a USB connection. The hardware component of the receiving chain consisted solely of the antenna, its low noise amplifier (LNA), and the analog-to-digital converter with front-end filtering, with all other components being software-based. This fundamental architecture laid the groundwork for subsequent enhancements and has been the essential building block of the NGene receiver family.

Thanks to its reconfigurable and modular structure, NGene has long served as the primary tool for in-lab analysis, development, and prototyping of signal-processing algorithms and architectures. Because of its flexible implementation, NGene was adapted to process the Galileo IOV signals (GIOVE-A) and subsequently to process the first Galileo signals immediately upon their availability, as detailed by Margaria et al. (2012). As a result, the research team was among the first worldwide to achieve a position fix using the initial four Galileo satellites. Over time, the software receiver continued to evolve and was tailored to address diverse applications, leveraging the advantages of software radio implementation (see Table 10).

Today, the NGene receiver family offers configurable support for various RF-to-IF front-ends, which connect to the software processor via USB, meeting the requirements of numerous activities and projects. A simplified, low-complexity version was developed to enable GNSS positioning capabilities on ARM-based embedded processors (Gamba, Nicola, et al., 2015). Additional branches of the software were adapted for GNSS reflectometry receiver deployment in reflectometry tests (Gamba, Marucco, et al., 2015), evaluation of anti-jamming algorithms, and detection of non-standard code transmission and its effects on Galileo positioning (Dovis et al., 2017), while also serving as a tool for studying the 2019 Galileo outage event (Dovis et al., 2019). One of the latest branches of the NGene family encompasses algorithms for authenticating Galileo messages using the OSNMA, as described by Nicola et al. (2022); Gamba et al. (2020a). Furthermore, a set of functions is being developed to support future GPS Chimera authentication service processing (Gamba et al., 2020b).

3.10 | The MATRIX SDR for Navigation with SOPs

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 can produce a navigation solution in a standalone fashion (Shamaei & Kassas, 2021a) or by fusing SOPs with sensors

(e.g., IMU (Morales & Kassas, 2021), lidar (Maaref et al., 2019), etc.), digital maps (Maaref & Kassas, 2020), and/or other signals (e.g., GNSS) (Kassas et al., 2017). Figure 1 shows MATRIX's architecture, and Table 11 lists its main features.

TABLE 10
Main Features of the NGene Receiver Family

NGene Receiver Family		
Feature	**Solution**	**Remark**
Operating system	GNU/Linux-based	Because it is based on standard libraries, it can also run on Windows.
Programming environment	ANSI C and assembly (Intel x86 and ARM SIMD instructions)	Eclipse IDE and GNU Compiler Collection compiler.
IF sample file input source	Binary file	
Processing mode	Real time and post-processing	Can work in real time from USB-based RF front-ends and in post-processing mode with binary file formats.
Additional sensors		
Supported GNSS	GPS L1 C/A, Galileo E1	
Acquisition	FFT-based algorithm	
Tracking	Multi-correlator-based data tracking loop	
Measurement output	Yes	Acquisition, tracking, and PVT results available as binary/text log files.
Availability	Restricted	Licensing of a public release currently under discussion.
Further features	Scintillation monitoring, interference detection, Galileo OSNMA authentication	Specific modified versions for research purposes.

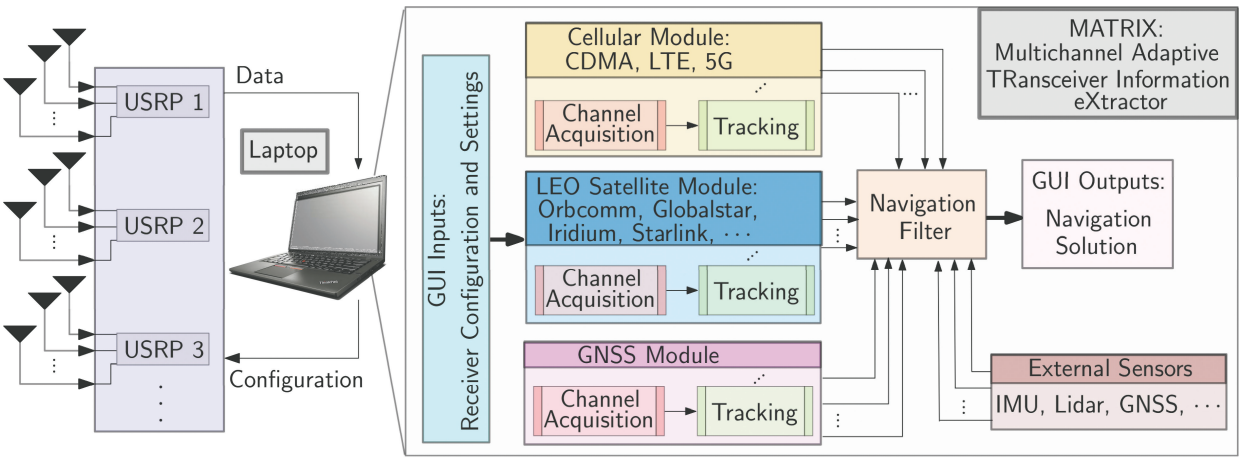


FIGURE 1 MATRIX cognitive SDR architecture
The SDR consists of (i) USRPs to collect different radio signals, (ii) various modules to produce navigation observables from different types of signals (e.g., cellular, LEO satellites, etc.), (iii) external sensors (e.g., IMU, lidar, GNSS receivers, etc.),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

TABLE 11
Main Features of MATRIX

MATRIX		
Feature	**Solution**	**Remark**
Operating system	Linux, Windows	
Programming environment	C++, MATLAB, LabVIEW	
IF sample file input source	Binary file	
Real-time sample input	Yes	Some SOP modules support real-time processing.
Additional sensors	IMU, GNSS	
Supported GNSS	GPS L1 C/A	
Acquisition	FFT-based signal acquisition	
Tracking	DLL, PLL, FLL, Kalman filter	Different tracking loops per SOP module.
Measurement output	Yes	Acquisition, tracking, and PVT results available as text/CSV files and via GUI.
Navigation	Weighted least squares, Kalman filter, Doppler, code- and carrier-based positioning modes	
Availability	Restricted	Licensing options available via The Ohio State University.

On one hand, MATRIX has achieved the most accurate navigation results to date in the published literature with cellular SOPs (3G code division multiple access (CDMA), 4G LTE, and 5G new radio (NR)), achieving meter-level navigation indoors (Abdallah & Kassas, 2021) and on ground vehicles (Maaref & Kassas, 2022) and sub-meter-level navigation on uncrewed aerial vehicles (Khalife & Kassas, 2022). In addition, MATRIX's efficacy has been demonstrated in a real-world GPS-denied environment (Kassas, Khalife, Abdallah, & Lee, 2022), 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, Khalife, Abdallah, Lee, Jurado, et al., 2022). Furthermore, 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, Abdallah, et al., 2022).

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, obtaining 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 satellites (Orbcomm, Iridium NEXT, and Starlink) were achieved with MATRIX (Kassas et al., 2021), upon coupling its LEO navigation observables with an INS in

a tightly coupled fashion through a simultaneous tracking and navigation framework (Kassas et al., 2019).

3.11 | Other Achievements with GNSS SDRs

Apart from the success stories described in the previous subsections, a number of other achievements have been accomplished with GNSS SDRs, as listed in this subsection.

The first real-time GNSS/INS integration with an SDR was achieved by Gunawardena et al. (2004), and one of the first GNSS SDR implementations on a GPU was reported by Hobiger et al. (2010).

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 are beneficial for sensitivity or multipath mitigation, as reported in Section 3.8. Characterization of the GPS transmit antenna pattern with a 30-s-long coherent integration resulting in 0-dBHz sensitivity has been discussed by Donaldson et al. (2020). The same sensitivity was achieved by 300 noncoherent integrations, each 1 s long, by iPosi Inc. (2015) for the purpose of indoor timing.

Graphical programming languages, such as LabVIEW and Simulink, are attractive choices for implementing SDRs because of their flexibility, modularity, and upgradability. Moreover, because 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 SOPs for navigation purposes (Diouf et al., 2021, 2019; Kassas et al., 2017), such as (i) cellular 3G CDMA (Khalife et al., 2018; Pesyna et al., 2011; Yang & Soloviev, 2018), 4G LTE (del Peral-Rosado et al., 2017; Ikhtiari, 2019; Kang et al., 2019; Shamaei & Kassas, 2018; Shamaei et al., 2018; Wang et al., 2022; Yang et al., 2022), and 5G NR (Abdallah & Kassas, 2022; del Peral-Rosado et al., 2022; Fokin & Volgushev, 2022; Lapin et al., 2022; Santana et al., 2021; Shamaei & Kassas, 2021b; Tang & Peng, 2022); (ii) AM/FM radio (Chen et al., 2020; McEllroy, 2006; Psiaki & Slosman, 2022; Souli et al., 2021); (iii) digital television (Souli et al., 2020, 2022; Yang & Soloviev, 2020); and (iv) LEO satellites (Farhangian et al., 2021; Farhangian & Landry, 2020; Nardin et al., 2021; Orabi et al., 2021; Pinell, 2021; Zhao et al., 2022).

Because of their enhanced analysis possibilities, GNSS SDRs proved to be very useful for elucidating ionospheric scintillation. The first dedicated SDRs were described by O'Hanlon et al. (2011); Peng & Morton (2011). The authors used a general-purpose front-end that was reconfigurable for multi-GNSS multi-band signals and a custom dual-frequency front-end. 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 was partly discussed in the previous sections. In addition, a major receiver manufacturer has provided GNSS SDRs, starting with a timing receiver (Trimble Inc., 2005) and then moving to a flexible narrowband receiver (Trimble Inc., 2017). Wideband signals were later added, with some signal processing performed on an FPGA, as reported by PR Newswire (2021). The most recent commercial activity has been reported by LocusLock (2022) and builds upon the software described in Section 3.1.

4 | SDR FRONT-ENDS

As outlined above, a front-end is required to obtain digital samples for SDR processing. The front-end tasks are to receive, filter, amplify, downconvert, and further digitize and quantize the analog RF signal entering the GNSS antenna. Many different types of front-ends have been used for GNSS SDRs. Roughly, five different categories can be identified:

Discrete components: Using RF-connectable components such as LNAs, filters, or ADCs, it is comparable easy to realize the function of a front-end and log IF or baseband samples. These setups are easy to realize, but are often bulky and sometimes prone to interference.

Commercial signal recorders: Several companies offer GNSS signal recorders that allow one to record (and often to replay) one or more GNSS frequency bands. These recorders usually 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 range are available. Their prices range from a few dollars (Fernández–Prades et al., 2013) to highly sophisticated multi-channel front-ends costing several tens of thousands of dollars. The oscillator quality, bit width, and RF-filter characteristics are not always optimal for GNSS signal processing.

Dedicated GNSS real-time front-ends: Built for the purpose of realizing a real-time GNSS SDR, these front-ends are compact and built with discrete components. A good example is described in Section 4.1.

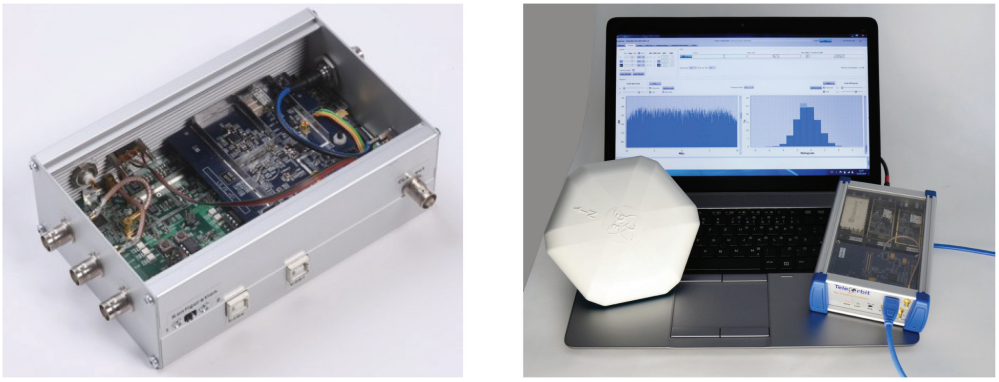
ASICs: Some RF ASICs seem to target GNSS SDR use, and evaluation kits allow streaming of IF samples, e.g., as reported by NTLAB, UAB (2022); RF Micro Devices, Inc., Greensboro (2006).

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

In the following section, we describe Fraunhofer USB front-ends as an example of user needs as well as the 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 the work by (Borre, Fernández-Hernández et al., 2022, Ch. 12).

4.1 | Fraunhofer USB Front-Ends

The scientific community, along with some industrial partners, required a multi-band solution for the upcoming civil multi-band signals in GPS and Galileo planning. In 2006, Fraunhofer IIS developed a front-end called the L125 Triband USB (see Figure 2(a)), which allowed recording of fixed frequencies of L1/E1, L2, and L5/E5a. This front-end had one antenna input and, via two USB 2.0 connectors, could record data streams with a sampling rate of up to 40 megasamples per second (MSPS) and a 2- or 4-bit ADC resolution. However, increasing customer demands for portable and flexible solutions led to a complete redesign of the USB



(a) (b)

FIGURE 2 Two exemplary USB front-ends from Fraunhofer IIS (a) TriBand USB2.0 front-end from 2006 (b) Flexiband USB3.0 front-end from 2012 onwards

front-end concept. One major request was reconfigurability on the SDR front-end side. To meet these different requirements in one SDR front-end hardware component, a new version of the USB front-end was developed that realizes signal conditioning to an onboard FPGA, enabling desired reconfigurability on the fly. This feature was necessary to allow for a single-band receiver with a low sampling rate for specific real-time SDR projects, as well as a wideband and multi-frequency front-end for other projects.

In 2012, Fraunhofer IIS (Rügamer et al., 2012) introduced the Flexiband multi-system, multi-band USB front-end depicted in Figure 2(b). 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 regular Flexiband unit consists of up to three analog reception boards, a carrier board with ADCs and an FPGA, and a USB 3.0 interface board. A common antenna input port is supported, with 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 in-phase/quadrature. The raw data stream is received by an FPGA in which different digital operations such as filtering, mixing, data rate reduction, and bit-width reduction as well as digital automatic gain control are applied. Finally, a single multiplexed data stream is formed together with a checksum. This multiplexed stream is sent via a USB 3.0 interface to the PC. Data rates of up to 1296 MBit/s or 162 MByte/s for a raw data stream are supported. The Flexiband GUI software receives the raw multiplexed stream, verifies 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.

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

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

5 | ION SDR STANDARD

The previous sections have already clarified that data exchange between the various SDRs requires a certain level of standardization. The events that led up to the suggestion of developing what became the ION SDR Standard (also known as the ION GNSS Metadata Standard, ION SDR Metadata Standard, GNSS SDR Sampled Data Metadata Standard, or GNSS 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 project was sponsored by the FAA Technical Center and led by Prof. Frank van Graas. Commercial GPS receivers within prototype local-area augmentation system ground facilities were experiencing brief unexplained outages. GAEM maintained a continuous 10-s 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 an additional 10 s. 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 based on one of the earliest commercially available PCI-based dual-ADC-plus-FPGA development cards. This version collected two GPS L1 data streams at 5 MSPS and 2-MHz bandwidth. This version of GAEM was fielded at three airports, operated continuously for over three 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 & Van Graas, 2009) and GPS/IMU deep integration demonstrations in flight (Soloviev et al., 2004). For the latter project, the 2-kHz raw data from a micro-electromechanical system IMU were interleaved with 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 was not clear whether these systems would serve the purpose for satnav SDR applications (sampling coherency concerns, etc.). Moreover, the >$350k price tag of these systems far precluded any hope of their purchase for university research. Thus, researchers 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., 2007) 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, and a defense contractor. To support these opportunities, the development of a new system known as the Wideband TRIGR was completed in 2008 (Gunawardena & Van Graas, 2011). The front-end was miniaturized to a single-frequency custom printed circuit board module. Up to eight 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 eight 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 two streams at 12 bits (sign extended to 16), to all eight streams at 4 bits. The sustained data transfer rate from the PCIe FPGA card to storage via a redundant array of independent disks (RAID) was limited to 240 MB/s. Thus, the appropriate format had to be selected to balance between the required capability and transfer rate. The generated file names embed a coordinated universal time (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-s circular buffer could not be easily implemented in random access memory using the 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 files 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 for 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 step was excruciatingly slow, especially for Wideband TRIGR data. To address this issue and to support the rapid emergence of multi-band and multi-constellation satnav signals, Gunawardena wrote and distributed a MATLAB SDR toolbox in which correlation was performed in optimized C code and multi-threading was leveraged in a parallel data 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 adopt a metadata standard—similar to the standard developed for Wideband TRIGR—in order to alleviate the numerous difficulties associated with sharing such files. This suggestion 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 later added 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 WG developed the standard as well as associated normative software over a course of six years. With regard 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 the formal ION SDR Standard in January 2020.

5.1 | Use of the ION SDR Standard

Today, the ION SDR Standard serves as a reference for describing IF formats and is useful, for example, for public tenders or if an established format is needed. A number of SDRs include the C++ libraries to read metadata and IF samples.

The level of exchange of IF samples between research groups is limited to some extent, and such exchanges occur much less frequently than, e.g., the exchange of RINEX files. This trend is 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 primarily the same front-ends, 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) proposed an algorithm to automatically generate optimized code for sample reading for a given IF format, but this proposal has not yet been manifest into a usable implementation.

5.2 | ION SDR Standard Extension

During the standardization process, a number of features were identified that appeared to be useful for the standard; however, a lack of resources did not allow for the inclusion of all of these features in the formal standardization procedure. These features are described in Appendix II of ION SDR Working Group (2020). At the ION-GNSS+ 2022 meeting in September, the following points were discussed and will be included in Appendix II of the next version (V1.1) of the ION SDR Standard.

5.2.1 | Flexible Bit Layout

The ION SDR Standard defines a "lump" as the ordered containment of all samples occurring within an interval. The ordered containment is understood in a regular way, with the samples of individual streams held together. Clements et al. (2021) view this aspect as a limitation, as highly efficient SDRs may use efficient bit-packing schemes to optimize data transfer over communication lines that need buffering. They have identified a need to distribute the samples of different streams in interleaved ways over the lump. This interleaving cannot be described by V1.0 of the ION SDR Standard. To overcome this limitation, the authors propose a new but optional attribute for the lump object, called "layout." If the layout is present, further information on the bit packing scheme must be provided, explicitly describing the type of each bit of a lump. The authors presented a detailed proposal for this new lump layout following the structure of the existing standard. The proposal even includes more advanced bit use cases, such as puncturing (e.g., explicit omission of bits) and overwriting of bits by time markers.

5.2.2 | Refined Sample Rate/Epoch Definitions

Clements et al. (2021) noted that V1.0 of the ION SDR Standard makes implicit assumptions about the timing of the sampling process that are not suitable for

staggered sampling. Staggered sampling occurs if the sampling instants of different GNSS signals are delayed with respect to each other, which might be of use for increasing the observability of GNSS interference in a multi-antenna system. To overcome this limitation, the authors propose the addition of two new attributes for "stream" objects to shift the sampling epochs of different GNSS streams with respect to each other.

5.2.3 | *JSON Format for Metadata Files*

Comment ID 22 of the initial request for comments suggested that the WG should consider markup languages other than XML for metadata files, specifically JSON, YAML, and TOML (Anonymous, 2017). In 2017, this comment was addressed by asserting that the XML format would be maintained for the time being, as 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 (a satnav SDR that is completely described using a draft signal/system specification language based on JSON, as described in Section 3.7), 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 that described by Plotly (2022). Hence, streaming JSON could be one way to parse SDR sample streams whose formats are changing dynamically.

Figure 3 presents a notional listing for a JSON-formatted metadata description for the Flexiband front-end XML metadata listing reported by 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. The adoption of JSON-based metadata is currently being considered for future versions of PyChips. Once a successful implementation has been achieved, consideration for adopting JSON as another valid option for representing ION SDR Standard-compliant metadata in a future version of the standard will be requested.

SUMMARY AND CONCLUSION

Since the beginning of GPS SDR developments in the mid 1990s, together with the operational declaration of GPS, the feasibility of GPS SDR has been widely proven by several platforms and their derivatives. We defined GNSS SDR platforms as those implementing receiver functions in general-purpose software and processors and divided them in real-time receivers, teaching/research tools, and snapshot receivers. We described some of these platforms, with a focus on those related to the authors but also including other developments. In particular, and based on the pioneering work by Akos, we described the bit-wise parallelism platform developed by the Cornell GPS group, which led to GRID by UT Austin; the MuSNAT receiver by UniBwM, which led to IFEN GmbH's SX3 commercial receiver; the SoftGPS MATLAB receiver and associated book, which are widely used for GNSS

```
{"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)

teaching and are also influencing other platforms, such as FGI-GSRx; the popular C++ open-source GNSS-SDR by CTTC; AutoNav SDR by Inha University; PyChips by Gunawardena, based on Python; the snapshot GNSS receiver developed by UAB, leading to cloudGNSSrx; the real-time NGene receiver developed by LINKS, used for early testing of the first Galileo signals and OSNMA; and the MATRIX receiver by ASPIN for navigation with terrestrial and space-based SOPs, among others. We provided an overview of the tasks and components of SDR front-ends, and for this purpose, we described Fraunhofer developments from the last few years as a reference. Finally, we discussed the ION SDR Standard, officially approved by ION in 2020, and its current extensions.

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

ACKNOWLEDGMENTS AND REMARKS

The authors are mainly 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 responsibility for the structure of the work, the abstract, and the introduction and is thus listed as first author. This article is based on a contribution by the same authors to the ION-GNSS+ conference 2022 in Denver, CO.

The GNSS SDR developments at the University of the Bundeswehr Munich, described in Section 3.2, were supported by numerous research projects administered by the German Aerospace Center and were financed by the German Federal Ministry for Economic Affairs and Climate Action. The authors acknowledge financial support from the University of the Bundeswehr München for this publication.

At Northeastern University, support has been partially provided by the NSF under awards ECCS-1845833 and CCF-2326559.

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

The MATRIX SDR development at the ASPIN Laboratory was supported by the Office of Naval Research under grants N00014-16-1-2305, N00014-19-1-2613, and N00014-19-1-2511; the Air Force Office of Scientific Research under grant FA9550-22-1-0476; the U.S. Department of Transportation under grant 69A3552047138 for the CARMEN University Transportation Center; and the National Institute of Standards and Technology 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.

The views expressed in this article are those of the individual authors and do not reflect the official policy or position of any state or government entity.

REFERENCES

Abdallah, A., & Kassas, Z. (2021). Multipath mitigation via synthetic aperture beamforming for indoor and deep urban navigation. *IEEE Transactions on Vehicular Technology*, *70*(9), 8838–8853. https://doi.org/10.1109/TVT.2021.3094807

Abdallah, A., & Kassas, Z. (2022). Opportunistic navigation using sub-6 GHz 5G downlink signals: A case study on a ground vehicle. *Proc. of the 16th European Conference on Antennas and Propagation (EuCAP),* Madrid, Spain, 1–5. https://doi.org/10.23919/EuCAP53622.2022.9769276

Abdallah, A., Kassas, Z., & 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. *Proc. of the Workshop on Automotive and Autonomous Vehicle Security (AutoSec),* San Diego, CA, 1. https://dx.doi.org/10.14722/autosec.2022.23049

Akos, D., & Braasch, M. (1996). A software radio approach to global navigation satellite system receiver design. *Proc. of the 52nd Annual Meeting of the Institute of Navigation,* Cambridge, MA, 455–463. https://www.ion.org/publications/abstract.cfm?articleID=1104

Akos, D. M. (1997). *A software radio approach to global navigation satellite system receiver design* [Doctoral dissertation, Fritz J. and Dolores H. Russ College of Engineering and Technology, Ohio University]. https://web.stanford.edu/group/scpnt/gpslab/pubs/theses/DennisAkosThesis97.pdf

Akos, D. M., Normark, P.-L., Enge, P., Hansson, A., & Rosenlind, A. (2001). Real-time GPS software radio receiver. *Proc. of the 2001 National Technical Meeting of the Institute of Navigation,* Long Beach, CA, 809–816. https://www.ion.org/publications/abstract.cfm?articleID=194

Akos, D. M., Stockmaster, M., Tsui, J. B., & Caschera, J. (1999). Direct bandpass sampling of multiple distinct RF signals. *IEEE Transactions on Communications*, *47*(7), 983–988. https://doi.org/10.1109/26.774848

Anonymous. (2017). *RFC1 comments and responses.* https://sdr.ion.org/RFC1_Comments_Responses.html

Arizabaleta, M., Ernest, H., Dampf, J., Kraus, T., Sanchez-Morales, D., Dötterböck, D., Schütz, A., & Pany, T. (2021). Recent enhancements of the multi-sensor navigation analysis tool (MuSNAT). *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021),* St. Louis, MO, 2733–2753. https://doi.org/10.33012/2021.17960

Bernabeu, J., Palafox, F., Li, Y., & Akos, D. (2022). A collection of SDRs for global navigation satellite systems (GNSS). *Proc. of the 2022 International Technical Meeting of the Institute of Navigation,* Long Beach, CA, 906–919. https://doi.org/10.33012/2022.18230

Bhuiyan, M., Söderholm, S., Thombre, S., Ruotsalainen, L., & Kuusniemi, H. (2015). Performance analysis of a dual frequency software-defined BeiDou receiver with B1 and B2 signals. *Proc. of the China Satellite Navigation Conference (CSNC),* Xi'an, China, 827–839. Springer. http://dx.doi.org/10.1007/978-3-662-46638-4_72

Bhuiyan, M. Z., Söderholm, S., Thombre, S., Ruotsalainen, L., & Kuusniemi, H. (2014). Overcoming the challenges of BeiDou receiver implementation. *Sensors*, *14*(11), 22082–22098. https://doi.org/10.3390/s141122082

Bhuiyan, M. Z. H., Söderholm, S., Thombre, S., & Kuusniemi, H. (2022). BeiDou B1I receiver processing. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 153–163. Cambridge University Press. https://doi.org/10.1017/9781108934176.007

Bhuiyan, M. Z. H., Söderholm, S., Ferrara, G., Kirkko-Jaakkola, M., Kuusniemi, H., López-Salcedo, J. A., & Fernández-Hernández, I. (2022). Galileo E1 receiver processing. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 140–152. Cambridge University Press. https://doi.org/10.1017/9781108934176.006

Bhuiyan, M. Z. H., Honkala, S., Söderholm, S., & Kuusniemi, H. (2022). GLONASS L1OF receiver processing. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 126–139. Cambridge University Press. https://doi.org/10.1017/9781108934176.005

Bochkati, M., Dampf, J., & Pany, T. (2022). On the use of multi-correlator values as sufficient statistics as basis for flexible ultra-tight GNSS/INS integration developments. *Proc. of the 2022 25th International Conference on Information Fusion (FUSION),* Linköping, Sweden, 1–8. https://doi.org/10.23919/FUSION49751.2022.9841253

Bochkati, M., Dampf, J., & Pany, T. (2023). Receiver clock estimation for RTK-grade multi-GNSS multifrequency synthetic aperture processing. *Proc. of the 2023 IEEE/ION Position, Location, and Navigation Symposium (PLANS),* Monterey, CA, 968–976. https://doi.org/10.1109/PLANS53410.2023.10140032

Borre, K. (2003). The GPS Easy Suite—Matlab code for the GPS newcomer. *GPS Solutions*, *7*(1), 47–51. https://doi.org/10.1007/s10291-003-0049-3

Borre, K. (2009). GPS Easy Suite II. *Inside GNSS*, *2*, 48–51. https://www.insidegnss.com/pdf/EasySuite.pdf

Borre, K., Akos, D., Bertelsen, N., Rinder, P., & Jensen, S. H. (2007). *A software-defined GPS and Galileo receiver, single frequency approach*. Birkhäuser. https://doi.org/10.1007/978-0-8176-4540-3

Borre, K., Fernandez-Hernandez, I., Lopez-Salcedo, J. A., & Bhuiyan, M. Z. H. (Eds.). (2022). *GNSS software receivers*. Cambridge University Press. https://doi.org/10.1017/9781108934176

Borre, K., Bhuiyan, M. Z. H., Söderholm, S., Kuusniemi, H., Fernández-Hernández, I., & López-Salcedo, J. A. (2022). GPS L1 C/A receiver processing. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 108–125. Cambridge University Press. https://doi.org/10.1017/9781108934176.004

Chen, X., Wei, Q., Wang, F., Jun, Z., Wu, S., & Men, A. (2020). Super-resolution time of arrival estimation for a symbiotic FM radio data system. *IEEE Transactions on Broadcasting*, *66*(4), 847–856. https://doi.org/10.1109/TBC.2019.2957666

Clements, Z., Iannucci, P. A., Humphreys, T. E., & Pany, T. (2021). Optimized bit-packing for bit-wise software-defined GNSS radio. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021),* St. Louis, MO, 3749–3771. https://doi.org/10.33012/2021.18015

Curran, J. T., Fernández–Prades, C., Morrison, A., & Bavaro, M. (2018). Innovation: The continued evolution of the GNSS software-defined radio. *GPS World*, *29*(1), 43–49. https://www.gpsworld.com/innovation-the-continued-evolution-of-the-gnss-software-defined-radio/

Dampf, J., Pany, T., Bär, W., Winkel, J., Stöber, C., Fürlinger, K., Closas, P., & 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. https://www.researchgate.net/publication/289355114_More_Than_We_Ever_Dreamed_Possible_Processor_Technology_for_GNSS_Software_Receivers_in_the_Year_2015

del Peral-Rosado, J., Estatuet-Castillo, R., López-Salcedo, J., Seco-Granados, G., Chaloupka, Z., Ries, L., & Garcoa-Molina, J. (2017). Evaluation of hybrid positioning scenarios for autonomous vehicle applications. *Proc. of the 30th International Technical Meeting of the Satellite Division*

of the Institute of Navigation (ION GNSS+ 2017), Portland, OR, 2541–2553. https://doi.org/10.33012/2017.15220

del Peral-Rosado, J., Lopez-Salcedo, J., Seco-Granados, G., Zanier, F., Crosta, P., Ioannides, R., & Crisci, M. (2013). Software-defined radio LTE positioning receiver towards future hybrid localization systems. *Proc. of the 31st AIAA International Communication Satellite Systems Conference,* Florence, Italy, 1–11. https://doi.org/10.2514/6.2013-5610

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., Pjoani, G., Canzian, L., Babaroglu, H., Rastorgueva-Foi, E., Talvitie, J., & Flachs, D. (2022). Design considerations of dedicated and aerial 5G networks for enhanced positioning services. *Proc. of the 10th Workshop on Satellite Navigation Technology (NAVITEC),* Noordwijk, Netherlands, 1–12. https://doi.org/10.1109/NAVITEC53682.2022.9847557

Diouf, C., Janssen, G., Dun, H., Kazaz, T., & Tiberius, C. (2021). A USRP-based testbed for wideband ranging and positioning signal acquisition. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–15. https://doi.org/10.1109/TIM.2021.3065449

Diouf, C., Janssen, G., Kazaz, T., Dun, H., Chamanzadeh, F., & Tiberius, C. (2019). A 400 Msps SDR platform for prototyping accurate wideband ranging techniques. *Proc. of the 16th Workshop on Positioning, Navigation and Communications (WPNC),* Bremen, Germany, 1–6. https://doi.org/10.1109/WPNC47567.2019.8970251

Donaldson, J. E., Parker, J. J., Moreau, M. C., Highsmith, D. E., & Martzen, P. D. (2020). Characterization of on-orbit GPS transmit antenna patterns for space users. *NAVIGATION, 67*(2), 411–438. https://doi.org/10.1002/navi.361

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., & Gamba, M. T. (2017). Anomalous GPS signals reported from SVN49. *GPS World*. https://www.gpsworld.com/anomalous-gps-signals-reported-from-svn49/

Dovis, F., Minetto, A., Nardin, A., Falletti, E., Margaria, D., Nicola, M., & Vannucchi, M. (2019). What happened when Galileo experienced a week-long service outage. *GPS World*. https://www.gpsworld.com/why-galileo-experienced-a-week-long-service-outage

Dötterböck, D., Pany, T., Lesjak, R., Prechtl, T., & Tabatabaei, A. (2023). PRN sequence estimation with a self-calibrating 40-element antenna array. *NAVIGATION, 70*(4). https://doi.org/10.33012/navi.600

Driusso, M., Marshall, C., Sabathy, M., Knutti, F., Mathis, H., & Babich, F. (2017). Vehicular position tracking using LTE signals. *IEEE Transactions on Vehicular Technology*, *66*(4), 3376–3391. https://doi.org/10.1109/TVT.2016.2589463

Eissfeller, B., & Won, J.-H. (2017). Receiver architecture. In P. J. Teunissen & O. Montenbruck (Eds.), *Springer handbook of global navigation satellite systems*, 365–400. Springer. https://doi.org/10.1007/978-3-319-42928-1

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

Farhangian, F., Benzerrouk, H., & Landry, R. (2021). Opportunistic in-flight INS alignment using LEO satellites and a rotatory IMU platform. *Aerospace*, *8*(10), 280–281. https://doi.org/10.3390/aerospace8100280

Farhangian, F., & Landry, R. (2020). Multi-constellation software-defined receiver for Doppler positioning with LEO satellites. *Sensors*, *20*(20), 5866–5883. https://doi.org/10.3390/s20205866

Fernández–Prades, C. (2022). *Yocto geniux.* https://github.com/carlesfernandez/yocto-geniux. https://doi.org/10.5281/zenodo.7848142

Fernández–Prades, C., Arribas, J., & Closas, P. (2016a). Accelerating GNSS software receivers. *Proc. of the 29th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS + 2016),* Portland, OR, 44–61. https://doi.org/10.33012/2016.14576

Fernández–Prades, C., Arribas, J., & Closas, P. (2016b). Assessment of software-defined GNSS receivers. *Proc. of the 2016 8th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC),* Noordwijk, Netherlands, 1–9. https://doi.org/10.1109/NAVITEC.2016.7931740

Fernández–Prades, C., Closas, P., & Arribas, J. (2013). Turning a television into a GNSS receiver. *Proc. of the 26th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2013),* Nashville, TN, 1492–1507. https://www.ion.org/publications/abstract.cfm?articleID=11334

Fernández–Prades, C., Vilà-Valls, J., Arribas, J., & Ramos, A. (2018). Continuous reproducibility in GNSS signal processing. *IEEE Access*, *6*(1), 20451–20463. https://doi.org/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

Foerster, F., & 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 (U.S. Patent No. 8451170; German Patent No. 102008014981). https://patents.google.com/patent/US20110050491

Fokin, G., & Volgushev, D. (2022). Software-defined radio network positioning technology design. Problem statement. *Proc. of Systems of Signals Generating and Processing in the Field of on Board Communications,* Moscow, Russian Federation, 1–6. https://doi.org/10.1109/IEEECONF53456.2022.9744391

Gamba, M. T., Marucco, G., Pini, M., Ugazio, S., Falletti, E., & Lo Presti, L. (2015). Prototyping a GNSS-based passive radar for UAVs: An instrument to classify the water content feature of lands. *Sensors*, *15*(11), 28287–28313. https://doi.org/10.3390/s151128287

Gamba, M. T., Nicola, M., & Falletti, E. (2015). eNGene: An ARM based embedded real-time software GNSS receiver. *Proc. of the 28th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2015),* Tampa, FL, 3178–3187. https://www.ion.org/publications/abstract.cfm?articleID=12891

Gamba, M. T., Nicola, M., & Motella, B. (2020a). Galileo OSNMA: An implementation for ARM-based embedded platforms. *Proc. of the 2020 International Conference on Localization and GNSS (ICL-GNSS),* Tampere, Finland, 1–6. https://doi.org/10.1109/ICL-GNSS49876.2020.9115539

Gamba, M. T., Nicola, M., & Motella, B. (2020b). GPS Chimera: A software profiling analysis. *Proc. of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2020),* 3781–3793. https://doi.org/10.33012/2020.17717

Gómez-Casco, D., López-Salcedo, J. A., & Seco-Granados, G. (2016). Generalized integration techniques for high-sensitivity GNSS receivers affected by oscillator phase noise. *Proc. of the 2016 IEEE Statistical Signal Processing Workshop (SSP),* Palma de Mallorca, Spain, 1–5. https://doi.org/10.1109/SSP.2016.7551809

Gunawardena, S. (2013). A universal GNSS software receiver MATLAB® toolbox for education and research. *Proc. of the 26th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2013),* Nashville, TN, 1560–1576. https://www.ion.org/publications/abstract.cfm?articleID=11230

Gunawardena, S. (2014). A universal GNSS software receiver toolbox. *Inside GNSS*, 58–67. https://insidegnss.com/a-universal-gnss-software-receiver-toolbox/

Gunawardena, S. (2021). A high performance easily configurable satnav SDR for advanced algorithm development and rapid capability deployment. *Proc. of the 2021 International Technical Meeting of the Institute of Navigation*, 539–554. https://doi.org/10.33012/2021.17808

Gunawardena, S., Pany, T., & Curran, J. (2021). ION GNSS software-defined radio metadata standard. *NAVIGATION, 68*(1), 11–20. https://doi.org/10.1002/navi.407

Gunawardena, S., Soloviev, A., & van Graas, F. (2004). Real time implementation of deeply integrated software GPS receiver and low cost IMU for processing low-CNR GPS signals. *Proc. of the 60th Annual Meeting of the Institute of Navigation,* Dayton, OH, 108–114. https://www.ion.org/publications/abstract.cfm?articleID=5602

Gunawardena, S., Soloviev, A., & van Graas, F. (2007). Wideband transform-domain GPS instrumentation receiver for signal quality and anomalous event monitoring. *NAVIGATION, 54*(4), 317–331. https://www.ion.org/publications/abstract.cfm?articleID=102456

Gunawardena, S., & van Graas, F. (2006). Split-sum correlator simplifies range computations in GPS receiver. *Electronics Letters*, *42*, 1469–1471. https://doi.org/10.1049/el:20062921

Gunawardena, S., & van Graas, F. (2011). Multi-channel wideband GPS anomalous event monitor. *Proc. of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2011),* Portland, OR, 1957–1968. https://www.ion.org/publications/abstract.cfm?articleID=9744

Gunawardena, S., Zhu, Z., Uijt de Haag, M., & van Graas, F. (2009). Remote-controlled, continuously operating GPS anomalous event monitor. *NAVIGATION, 56*(2), 97–113. https://www.ION.org/publications/abstract.cfm?articleID=102491

Hamza, G., Zekry, A., & Motawie, I. (2009). Implementation of a complete GPS receiver using Simulink. *IEEE Circuits and Systems Magazine*, *9*(4), 43–51. https://doi.org/10.1109/MCAS.2009.934706

Hobiger, T., Gotoh, T., Amagai, J., Koyama, Y., & Kondo, T. (2010). A GPU based real-time GPS software receiver. *GPS solutions*, *14*(2), 207–216. http://doi.org/10.1007/s10291-009-0135-2

Honkala, S. (2016). *GLONASS satellite navigation signal implementation in a software-defined multi-constellation satellite navigation receiver* [Unpublished Master's thesis, Aalto University]. https://aaltodoc.aalto.fi/bitstream/handle/123456789/20169/master_Honkala_Salomon_2016.pdf?sequence=1&isAllowed=y

Humphreys, T., Psiaki, M., Kintner, P., & Ledvina, B. (2006, September). GNSS receiver implementation on a DSP: Status, challenges, and prospects. *Proc. of the 19th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2006),* Fort Worth, TX, 2370–2382. https://www.ion.org/publications/abstract.cfm?articleID=7061

Humphreys, T. E., Bhatti, J., Pany, T., Ledvina, B., & O'Hanlon, B. (2009). Exploiting multicore technology in software defined GNSS receivers. *Proc. of the 22nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2009),* Savannah, GA, 326–338. https://www.ion.org/publications/abstract.cfm?articleID=8436

Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W., & Kintner, P. M., Jr. (2008). Assessing the spoofing threat: Development of a portable GPS civilian spoofer. *Proc. of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008),* Savannah, GA, 2314–2325. https://www.ion.org/publications/abstract.cfm?articleID=8132

Humphreys, T. E., Murrian, M. J., & Narula, L. (2020). Deep-urban unaided precise global navigation satellite system vehicle positioning. *IEEE Intelligent Transportation Systems Magazine*, *12*(3), 109–122. https://doi.org/10.1109/MITS.2020.2994121

IFEN GmbH. (2022). *SX3 GNSS Software Receiver — ifen.com.* https://www.ifen.com/receivers/sx3-gnss-software-receiver/

Ikhtiari, N. (2019). *Navigation in GNSS denied environments using software defined radios and LTE signals of opportunities* [Master's thesis, University of Canterbury]. UC Library Repository. http://dx.doi.org/10.26021/3061

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/

Jiménez-Baños, D., Blanco-Delgado, N., López-Risueño, G., Seco-Granados, G., & Garcia-Rodriguez, A. (2006). Innovative techniques for GPS indoor positioning using a snapshot receiver. *Proc. of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2006),* Fort Worth, TX, 2944–2955. https://www.ion.org/publications/abstract.cfm?articleID=6834

Kang, T., Lee, H., & Seo, J. (2019). Analysis of the maximum correlation peak value and RSRQ in LTE signals according to frequency bands and sampling frequencies. *Proc. of the 2019 19th International Conference on Control, Automation and Systems (ICCAS),* Jeju, Korea (South), 1182–1186. https://doi.org/10.23919/ICCAS47443.2019.8971462

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., & Tay, R. (2022). Protecting the skies: GNSS-less accurate aircraft navigation with terrestrial cellular signals of opportunity. *Proc. of the 35th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2022),* Denver, CO, 1014–1025. https://doi.org/10.33012/2022.18579

Kassas, Z., Bhatti, J., & Humphreys, T. (2013). A graphical approach to GPS software-defined receiver implementation. *Proc. of the 2013 IEEE Global Conference on Signal and Information Processing,* Austin, TX, 1226–1229. https://doi.org/10.1109/GlobalSIP.2013.6737129

Kassas, Z., Khalife, J., Abdallah, A., & Lee, C. (2020). I am not afraid of the jammer: Navigating with signals of opportunity in GPS-denied environments. *Proc. of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2020)*, 1566–1585. https://doi.org/10.33012/2020.17737

Kassas, Z., Khalife, J., Abdallah, A., & Lee, C. (2022). 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. https://doi.org/10.1109/MAES.2022.3154110

Kassas, Z., Khalife, J., Abdallah, A., Lee, C., Jurado, J., Wachtel, S., Duede, J., Hoeffner, Z., Hulsey, T., Quirarte, R., & Tay, R. (2022). Assessment of cellular signals of opportunity for high altitude aircraft navigation. *IEEE Aerospace and Electronic Systems Magazine*, *37*(10), 4–19. https://doi.org/10.1109/MAES.2022.3187142

Kassas, Z., Khalife, J., Shamaei, K., & Morales, J. (2017). I hear, therefore I know where I am: Compensating for GNSS limitations with cellular signals. *IEEE Signal Processing Magazine*, *34*(5), 111–124. https://doi.org/10.1109/MSP.2017.2715363

Kassas, Z., Kozhaya, S., Saroufim, J., Kanj, H., & Hayek, S. (2023). A look at the stars: Navigation with multi-constellation LEO satellite signals of opportunity. *Inside GNSS Magazine*, *18*(4), 38–47. https://insidegnss.com/a-look-at-the-stars-navigation-with-multi-constellation-leo-satellite-signals-of-opportunity/

Kassas, Z., Morales, J., & Khalife, J. (2019). New-age satellite-based navigation – STAN: Simultaneous tracking and navigation with LEO satellite signals. *Inside GNSS*, *14*(4), 56–65. https://insidegnss.com/new-age-satellite-based-navigation-stan-simultaneous-tracking-and-navigation-with-leo-satellite-signals/

Kassas, Z., Neinavaie, M., Khalife, J., Khairallah, N., Haidar-Ahmad, J., Kozhaya, S., & Shadram, Z. (2021). Enter LEO on the GNSS stage: Navigation with Starlink satellites. *Inside GNSS*, *16*(6), 42–51. https://insidegnss.com/enter-leo-on-the-gnss-stage-navigation-with-starlink-satellites/

Khalife, J., & Kassas, Z. (2018). Precise UAV navigation with cellular carrier phase measurements. *Proc. of the 2018 IEEE/ION Position, Location, and Navigation Symposium (PLANS),* Monterey, CA, 978–989. https://doi.org/10.1109/PLANS.2018.8373476

Khalife, J., & 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*, *58*(5), 4261–4278. https://doi.org/10.1109/TAES.2022.3162770

Khalife, J., Neinavaie, M., & Kassas, Z. (2022). The first carrier phase tracking and positioning results with Starlink LEO satellite signals. *IEEE Aerospace and Electronic Systems Magazine*, *56*(2), 1487–1491. https://doi.org/10.1109/TAES.2021.3113880

Khalife, J., Shamaei, K., & 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. https://doi.org/10.1109/TSP.2018.2799167

Lapin, I., Granados, G., Samson, J., Renaudin, O., Zanier, F., & Ries, L. (2022). STARE: Real-time software receiver for LTE and 5G NR positioning and signal monitoring. *Proc. of the Workshop on Satellite Navigation Technology (NAVITEC),* Noordwijk, Netherlands, 1–11. https://doi.org/10.1109/NAVITEC53682.2022.9847544

Ledvina, B., Powell, S., Kintner, P., & Psiaki, M. (2003). A 12-channel real-time GPS L1 software receiver1. *Proc. of the 2003 National Technical Meeting of the Institute of Navigation,* Anaheim, CA, 767–782. https://www.ion.org/publications/abstract.cfm?articleID=3823

Ledvina, B., Psiaki, M., Humphreys, T., Powell, S., & Kintner, P. (2006). A real-time software receiver for the GPS and Galileo L1 signals. *Proc. of the 19th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2006),* Fort Worth, TX, 2321–2333. https://www.ion.org/publications/abstract.cfm?articleID=6942

Ledvina, B., Psiaki, M., Powell, S., & Kintner, P. (2004). Bit-wise parallel algorithms for efficient software correlation applied to a GPS software receiver. *IEEE Transactions on Wireless Communications*, *3*(5), 1469–1473. https://doi.org/10.1109/TWC.2004.833467

Ledvina, B., Psiaki, M., Powell, S., & Kintner, P. (2006). *Real-time software receiver* (U.S. Patent No. 7010060). U.S. Patent and Trademark Office. https://patents.google.com/patent/US7010060/en

Ledvina, B., Psiaki, M., Powell, S., & Kintner, P. (2007). *Real-time software receiver* (U.S. Patent No. 7305021). U.S. Patent and Trademark Office. https://patents.google.com/patent/US7305021B2/en

Ledvina, B., Psiaki, M., Sheinfeld, D., Cerruti, A., Powell, S., & Kintner, P. (2004). A real-time GPS civilian L1/L2 software receiver. *Proc. of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004),* Long Beach, CA, 986–1005. https://www.ion.org/publications/abstract.cfm?articleID=5775

Lightsey, G., Humphreys, T., Bhatti, J., Joplin, A., O'Hanlon, B., & Powell, S. (2014). Demonstration of a space capable miniature dual frequency GNSS receiver. *NAVIGATION*, *61*(1), 53–64. https://doi.org/10.1002/navi.52

Locubiche-Serra, S., López-Salcedo, J. A., & Seco-Granados, G. (2016). Statistical near-far detection techniques for GNSS snapshot receivers. *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* Shanghai, China, 6570–6574. https://doi.org/10.1109/ICASSP.2016.7472943

Locus Lock. (2022). *Reliable, accurate, and intelligent GPS lock.* https://locuslock.com

López-Salcedo, J. A., Capelle, Y., Toledo, M., Seco, G., López-Vicario, J., Kubrak, D., Monnerat, M., Mark, A., & Jiménez, D. (2008). DINGPOS: A hybrid indoor navigation platform for GPS and Galileo. *Proc. of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008),* Savannah, GA, 1780–1791. https://www.ion.org/publications/abstract.cfm?articleID=8090

López-Salcedo, J. A., Parro-Jiménez, J. M., & Seco-Granados, G. (2009). Multipath detection metrics and attenuation analysis using a GPS snapshot receiver in harsh environments. *Proc. of the 2009 3rd European Conference on Antennas and Propagation (EuCAP),* Berlin, Germany, 3692–3696. https://ieeexplore.ieee.org/document/5068391

Lucas-Sabola, V., Seco-Granados, G., López-Salcedo, J. A., García-Molina, J., & Crisci, M. (2017). Efficiency analysis of cloud GNSS signal processing for IoT applications. *Proc. of the 30th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2017),* Portland, OR, 3843–3852. https://doi.org/10.33012/2017.15237

Lucas-Sabola, V., Seco-Granados, G., López-Salcedo, J. A., García-Molina, J. A., & Crisci, M. (2016, June). Cloud GNSS receivers: New advanced applications made possible. *Proc. of the 2016 International Conference on Localization and GNSS (ICL-GNSS),* Barcelona, Spain, 1–6. https://doi.org/10.1109/ICL-GNSS.2016.7533852

Maaref, M., & 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. https://doi.org/10.1109/TITS.2019.2907851

Maaref, M., & 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. https://doi.org/10.1109/TITS.2021.3055200

Maaref, M., Khalife, J., & 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. https://doi.org/10.1109/TIV.2018.2886688

Manzano-Jurado, M., Alegre-Rubio, J., Pellacani, A., Seco-Granados, G., López-Salcedo, J. A., Guerrero, E., & García-Rodríguez, A. (2014). Use of weak GNSS signals in a mission to the moon. *Proc. of the 2014 7th ESA Workshop on Satellite Navigation Technologies and European*

Workshop on GNSS Signals and Signal Processing (NAVITEC), Noordwijk, Netherlands, 1–8. http://dx.doi.org/10.1109/NAVITEC.2014.7045151

Margaria, D., Linty, N., Favenza, A., Nicola, M., Musumeci, L., Falco, G., Falletti, E., Pini, M., Fantino, M., & Dovis, F. (2012). Contact! – First acquisition and tracking of IOV Galileo signals. *Inside GNSS*, *7*, 45–55. https://insidegnss.com/wp-content/uploads/2018/01/IGM_janfeb12-NavSAS.pdf

McEllroy, J. (2006). *Navigation using signals of opportunity in the AM transmission band* [Master's thesis, Air Force Institute of Technology]. AFIT Scholar. https://scholar.afit.edu/cgi/viewcontent.cgi?article=4453&context=etd

McEllroy, J., Raquet, J., & Temple, M. (2006). Use of a software radio to evaluate signals of opportunity for navigation. *Proc. of the 19th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2006),* Fort Worth, TX, 126–133. https://www.ion.org/publications/abstract.cfm?articleID=6768

Miller, N. S., Koza, T. J., Morgan, S. C., Martin, S. M., Neish, A., Grayson, R., & Reid, T. (2023). SNAP: A Xona Space Systems and GPS software-defined receiver. *Proc. of the 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Monterey, CA, 897–904. https://doi.org/10.1109/PLANS53410.2023.10139956

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., & López-Salcedo, J. A. (2020). A testbed for GNSS-based positioning and navigation technologies in smart cities: The HANSEL project. *Smart Cities*, *3*(4), 1219–1241. https://doi.org/10.3390/smartcities3040060

Ministry of Science and ICT of Korea. (2021). *Launch to become the world's seventh 'space powerhouse'!* https://www.msit.go.kr/eng/bbs/view.do?sCode=eng&mId=4&mPid=2&pageIndex=36&bbsSeqNo=42&nttSeqNo=568&searchOpt=ALL&searchTxt=

Mitola, J. (1995). The software radio architecture. *IEEE Communications Magazine*, *33*(5), 26–38. https://doi.org/10.1109/35.393001

Molino, A., Nicola, M., Pini, M., & Fantino, M. (2009). N-GENE GNSS software receiver for acquisition and tracking algorithms validation. *Proc. of the 17th European Signal Processing Conference,* Glasgow, UK, 2171–2175. https://ieeexplore.ieee.org/document/7077862

Morales, J., & Kassas, Z. (2021). Tightly-coupled inertial navigation system with signals of opportunity aiding. *IEEE Transactions on Aerospace and Electronic Systems*, *57*(3), 1930–1948. https://doi.org/10.1109/TAES.2021.3054067

Morton, Y., Jiao, Y., & Taylor, S. (2015). High-latitude and equatorial ionospheric scintillation based on an event-driven multi-GNSS data collection system. *Proc. of the 14th International Ionospheric Effects Symposium*, Alexandria, VA. https://api.semanticscholar.org/CorpusID:53550046

Murrian, M. J., Narula, L., Iannucci, P. A., Budzien, S., O'Hanlon, B.W., Powell, S. P., & Humphreys, T. E. (2021). First results from three years of GNSS interference monitoring from low Earth orbit. *NAVIGATION*, *68*(4), 673–685. https://doi.org/10.1002/navi.449

Nardin, A., Dovis, F., & 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. https://doi.org/10.1109/JRFID.2021.3077082

Neinavaie, M., Khalife, J., & 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. https://doi.org/10.1109/TAES.2021.3127488

Nichols, H. A., Murrian, M. J., & Humphreys, T. E. (2022). Software-defined GNSS is ready for launch. *Proc. of the 35th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2022),* Denver, CO, 996–1013. https://doi.org/10.33012/2022.18313

Nicola, M., Motella, B., Pini, M., & Falletti, E. (2022). Galileo OSNMA public observation phase: Signal testing and validation. *IEEE Access*, *10*, 27960–27969. https://doi.org/10.1109/ACCESS.2022.3157337

NTLAB, UAB. (2022). *Unique ICs for GNSS Receivers.* https://ntlab.lt/

O'Hanlon, B. W., Psiaki, M. L., Powell, S., Bhatti, J. A., Humphreys, T. E., Crowley, G., & Bust, G. S. (2011). Cases: A smart, compact GPS software receiver for space weather monitoring. *Proc. of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2011),* Portland, OR, 274–2753. https://doi.org/10.15781/T2N010949

Orabi, M., Khalife, J., & Kassas, Z. (2021). Opportunistic navigation with Doppler measurements from Iridium Next and Orbcomm LEO satellites. *Proc. of the IEEE Aerospace Conference,* Big Sky, MT, 1–9. https://doi.org/10.1109/AERO50100.2021.9438454

Pany, T., Dötterböck, D., Gomez-Martinez, H., Hammed, M. S., Hörkner, F., Kraus, T., Maier, D., Sanchez-Morales, D., Schütz, A., Klima, P., & Ebert, D. (2019). The multi-sensor navigation analysis tool (MuSNAT) – Architecture, LiDAR, GPU/CPU GNSS signal processing. *Proc. of the 32nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2019),* Miami, FL, 4087–4115. https://doi.org/10.33012/2019.17128

Pany, T., Eissfeller, B., Hein, G., Moon, S., & Sanroma, D. (2004). IPEXSR: A PC based software GNSS receiver completely developed in Europe. *Proc. of the ELMAR-2010*, Zadar, Croatia, 407–416. https://ieeexplore.ieee.org/document/5606128

Pany, T., Falk, N., Riedl, B., Hartmann, T., Stangl, G., & Stöber, C. (2012). Software GNSS receiver: An answer for precise positioning research. *GPS World*, *23*(9), 60–66. https://www.gpsworld.com/software-gnss-receiver-an-answer-for-precise-positioning-research/

Pany, T., Förster, F., & Eissfeller, B. (2004). Real-time processing and multipath mitigation of high-bandwidth L1/L2 GPS signals with a PC-based software receiver. *Proc. of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004),* Long Beach, CA, 971–985. https://www.ion.org/publications/abstract.cfm?articleID=5774

Pany, T., Kaniuth, R., & Eissfeller, B. (2005). Deep integration of navigation solution and signal processing. *Proc. of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005),* Long Beach, CA, 1095–1102. https://www.ion.org/publications/abstract.cfm?articleID=6305

Pany, T., Moon, S. W., Irsigler, M., Eissfeller, B., & Fürlinger, K. (2003). Performance assessment of an under-sampling SWC receiver for simulated high-bandwidth GPS/Galileo signals and real signals. *Proc. of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS/GNSS 2003),* Portland, OR, 103–116. https://www.ion.org/publications/abstract.cfm?articleID=5187

Pany, T., Riedl, B., Winkel, J., Wörz, T., Schweikert, R., Niedermeier, H., Lagrasta, S., López-Risueño, G., & Jiménez-Baños, D. (2009). Coherent integration time: The longer, the better. *Inside GNSS*, *4*(6), 52–61. https://www.insidegnss.com/auto/novdec09-wp.pdf

Peng, S., & Morton, Y. (2011). A USRP2-based multi-constellation and multi-frequency GNSS software receiver for ionosphere scintillation studies. *Proc. of the 2011 International Technical Meeting of the Institute of Navigation,* San Diego, CA, 1033–1042. https://www.ion.org/publications/abstract.cfm?articleID=9551

Pesyna, K., Kassas, Z., Bhatti, J., & Humphreys, T. (2011). Tightly-coupled opportunistic navigation for deep urban and indoor positioning. *Proc. of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2011),* Portland, OR, 3605–3616. https://www.ion.org/publications/abstract.cfm?articleID=9914

Pesyna, K. M., Jr., Heath, R. W., Jr., & Humphreys, T. E. (2014). Centimeter positioning with a smartphone-quality GNSS antenna. *Proc. of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2014),* Portland, OR, 1568–1577. https://www.ion.org/publications/abstract.cfm?articleID=12308

Pinell, C. (2021). *Receiver architectures for positioning with low Earth orbit satellite signals* [Unpublished Master's thesis, Lulea University of Technology, School of Electrical Engineering]. http://www.diva-portal.org/smash/get/diva2:1638231/FULLTEXT01.pdf

Plotly. (2022). *Plotly JSON chart schema.* https://plotly.com/chart-studio-help/json-chart-schema/

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

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. https://doi.org/10.1109/TWC.2006.1611075

Psiaki, M., Humphreys, T., Mohiuddin, S., Powell, S., Cerruti, A., & Kintner, P. (2006). Searching for Galileo. *Proc. of the 19th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2006),* Fort Worth, TX, 1567–1575. https://www.ion.org/publications/abstract.cfm?articleID=6973

Psiaki, M., & Slosman, B. (2022). Tracking digital FM OFDM signals for the determination of navigation observables. *NAVIGATION*, *69*(2). https://doi.org/10.33012/navi.521

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

Rügamer, A., Förster, F., Stahl, M., & Rohmer, G. (2012). A flexible and portable multiband GNSS front-end system. *Proc. of the 25th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2012),* Nashville, TN, 2378–2389. https://www.ion.org/publications/abstract.cfm?articleID=10432

Rügamer, A., Rubino, D., Lukcin, I., Taschke, S., Stahl, M., & Felber, W. (2016). Secure position and time information by server side PRS snapshot processing. *Proc. of the 29th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2016),* Portland, OR, 3002–3017. https://doi.org/10.33012/2016.14781

Santana, G., de Cristo, R., & Branco, K. (2021). Integrating cognitive radio with unmanned aerial vehicles: An overview. *Sensors*, *21*(3), 830–856. https://doi.org/10.3390/s21030830

Seco-Granados, G., López-Salcedo, J. A., Jiménez-Baños, D., & 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. https://doi.org/10.1109/MSP.2011.943410

Shamaei, K., & Kassas, Z. (2018). LTE receiver design and multipath analysis for navigation in urban environments. *NAVIGATION*, *65*(4), 655–675. https://doi.org/10.1002/navi.272

Shamaei, K., & Kassas, Z. (2021a). A joint TOA and DOA acquisition and tracking approach for positioning with LTE signals. *IEEE Transactions on Signal Processing, 69*, 2689–2705. https://doi.org/10.1109/TSP.2021.3068920

Shamaei, K., & 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. https://doi.org/10.1109/TWC.2021.3061985

Shamaei, K., Khalife, J., & Kassas, Z. (2018). Exploiting LTE signals for navigation: Theory to implementation. *IEEE Transactions on Wireless Communications*, *17*(4), 2173–2189. https://doi.org/10.1109/TWC.2018.2789882

Snyder, C., Feng, G., & Van Graas, F. (1999). GPS anomalous event monitor (GAEM). *Proc. of the 55th Annual Meeting of the Institute of Navigation,* Cambridge, MA, 185–189.

Söderholm, S., Bhuiyan, M., Thombre, S., Ruotsalainen, L., & Kuusniemi, H. (2016). A multi-GNSS software-defined receiver: Design, implementation, and performance benefits. *Annals of Telecommunications*, *71*, 399–410. https://doi.org/10.1007/s12243-016-0518-7

Söderholm, S., Bhuiyan, M. Z. H., Ferrara, G., Thombre, S., & Kuusniemi, H. (2022). A multi-GNSS software receiver. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 174–188. Cambridge University Press. https://doi.org/10.1017/9781108934176.009

Soloviev, A., Gunawardena, S., & Van Graas, F. (2004). Deeply integrated GPS/low-cost IMU for low CNR signal processing: Flight test results and real time implementation. *Proc. of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004),* Long Beach, CA, 1598–1608. https://www.ion.org/publications/abstract.cfm?articleID=5840

Song, Y. J., Lee, H. B., & 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. https://doi.org/10.11003/JPNT.2021.10.2.91

Souli, N., Kolios, P., & Ellinas, G. (2020). Relative positioning of autonomous systems using signals of opportunity. *Proc. of the IEEE 91st Vehicular Technology Conference (VTC2020),* Antwerp, Belgium, 1–6. https://doi.org/10.1109/VTC2020-Spring48590.2020.9128912

Souli, N., Kolios, P., & Ellinas, G. (2021). Online relative positioning of autonomous vehicles using signals of opportunity. *IEEE Transactions on Intelligent Vehicles*, *7*(4), 873–885. https://doi.org/10.1109/TIV.2021.3124727

Souli, N., Kolios, P., & Ellinas, G. (2022). Adaptive frequency band selection for accurate and fast positioning utilizing SOPs. *Proc. of the International Conference on Unmanned Aircraft Systems (ICUAS),* Dubrovnik, Croatia, 1309–1315. https://doi.org/10.1109/ICUAS54217.2022.9836189

SPCOMNAV. (2019). *Cloud GNSS Receiver.* http://cloudGNSSrx.com

Stöber, C., Anghileri, M., Ayaz, A. S., Dötterböck, D., Krämer, I., Kropp, V., Won, J.-H., Eissfeller, B., Güixens, D. S., & Pany, T. (2010). ipexSR: A real-time multi-frequency software GNSS receiver. *Proc. of the International Symposium on Electronics in Marine (ELMAR-2010),* Dubrovnik, Croatia, 407–416. https://ieeexplore.ieee.org/document/5606128

Takasu, T., & Yasuda, A. (2009). Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. *Proc. of the International symposium on GPS/GNSS,* Jeju, Korea, Vol. 1. https://www.researchgate.net/publication/228811569

Tang, G., & Peng, A. (2022). 5G receiver design based on downlink intermittent signals tracking algorithm. *Proc. of the China Satellite Navigation Conference (CSNC 2022),* Beijing, China, 462–471. https://doi.org/10.1007/978-981-19-2576-4_41

TeleOrbit GmbH. (2022). *MGSE REC: GNSS radio frequency front-end.* https://teleorbit.eu/en/satnav/mgse-rec/

Thombre, S., Bhuiyan, M., Söderholm, S., Kirkko-Jaakkola, M., Ruotsalainen, L., & Kuusniemi, H. (2015). A software multi-GNSS receiver implementation for the Indian regional navigation satellite system. *IETE Journal of Research*, *62*(2), 246–256. https://doi.org/10.1080/03772063.2015.1093968

Thombre, S., Bhuiyan, M. Z. H., Söderholm, S., & Kuusniemi, H. (2022). NavIC L5 receiver processing. In K. Borre., I. Fernández-Hernández, J. A. López-Salcedo & M. Z. H. Bhuiyan (Eds.), *GNSS software receivers*, 164–173. Cambridge University Press. https://doi.org/10.1017/9781108934176.008

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

Trimble Inc. (2017). *Trimble DA1 catalyst GNSS systems.* https://geospatial.trimble.com/en/links?dcs=Collection-133515

Tsui, J. B.-Y. (2000). *Fundamentals of Global Positioning System receivers: A software approach*. Wiley-Interscience. https://doi.org/10.1002/0471200549

UniBwM. (2023). *MuSNAT — LRT 9.* https://www.unibw.de/lrt9/lrt-9.2/software-packages/MuSNAT

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

Wang, P., Wang, Y., & 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. https://doi.org/10.1109/TAES.2021.3139569

Wikipedia. (2022). *JSON streaming.* https://en.wikipedia.org/wiki/JSON_streaming

Wikipedia. (2023). *GNSS software-defined receiver.* https://en.wikipedia.org/wiki/GNSS_software-defined_receiver

Yang, C., Arizabaleta-Diez, M., Weitkemper, P., & 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. https://doi.org/10.1109/MAES.2022.3186650

Yang, C., Nguyen, T., & Blasch, E. (2014). Mobile positioning via fusion of mixed signals of opportunity. *IEEE Aerospace and Electronic Systems Magazine*, *29*(4), 34–46. https://doi.org/10.1109/MAES.2013.130105

Yang, C., & Soloviev, A. (2018). Positioning with mixed signals of opportunity subject to multipath and clock errors in urban mobile fading environments. *Proc. of the 31st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2018),* Miami, FL, 223–243. https://doi.org/10.33012/2018.15833

Yang, C., & Soloviev, A. (2020). Mobile positioning with signals of opportunity in urban and urban canyon environments. *Proc. of the IEEE/ION Position, Location, and Navigation Symposium (PLANS),* Portland, OR, 1043–1059. https://doi.org/10.1109/PLANS46316.2020.9109876

Yoder, J. E., & Humphreys, T. E. (2023). Low-cost inertial aiding for deep-urban tightly-coupled multi-antenna precise GNSS. *NAVIGATION, 70*(1). https://doi.org/10.33012/navi.561

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

Zhao, C., Qin, H., & Li, Z. (2022). Doppler measurements from multiconstellations in opportunistic navigation. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1–9. https://doi.org/10.1109/TIM.2022.3147315

Zhu, Z., & Van Graas, F. (2009). Earth-surface multipath detection and error modeling for aircraft GPS receivers. *NAVIGATION*, *56*(1), 45–56. https://doi.org/10.1002/j.2161-4296.2009.tb00443.x

How to cite this article: Pany, T., Akos, D., Arribas, J., Bhuiyan, M. Z. H., Closas, P., Dovis, F., Fernandez-Hernandez, I., Fernández–Prades, C., Gunawardena, S., Humphreys, T., Kassas, Z., López-Salcedo, J. A., Nicola, M., Psiaki, M. L., Rügamer, A., Song, Y-J., & Won, J-H. (2024). GNSS software-defined radio: History, current developments, and standardization efforts. *NAVIGATION, 71*(1). https://doi.org/10.33012/navi.628