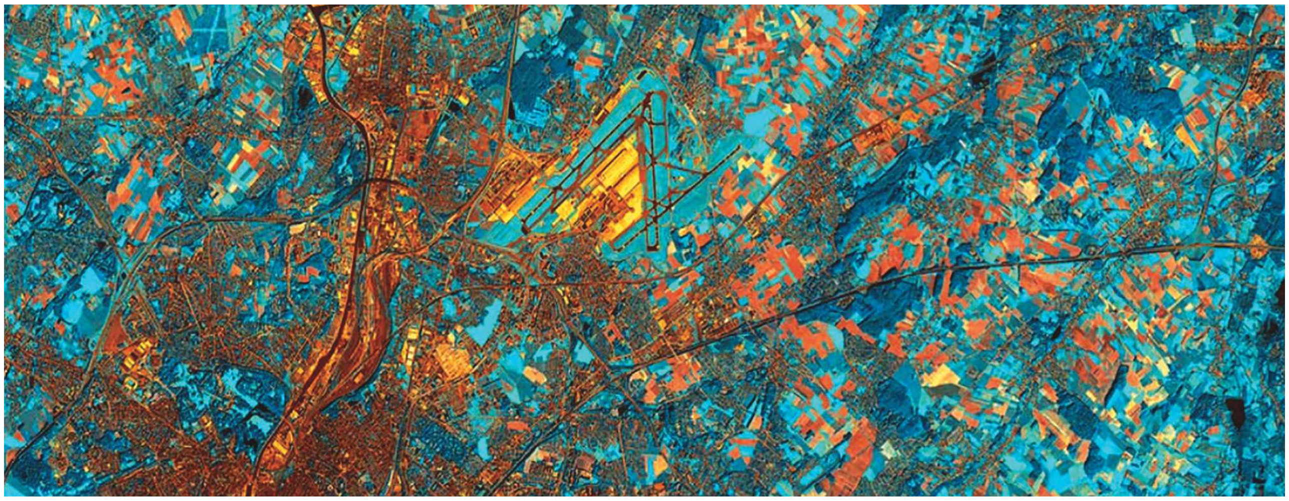
GNSS IoT Positioning
From Conventional Sensors to a Cloud-Based Solution



The advent of the Internet of Things (IoT) has considerably increased the number of services and applications that require positioning information. In this sense, IoT positioning sensors usually obtain and deliver their position to a central node where it is further managed and analyzed by a user or scheduler. Nonetheless, the stringent requirements of low-cost IoT sensors in terms of low power consumption to achieve larger battery lifetime are pushing current technologies to their limits. In this context, we propose a cloud-based Global Navigation Satellite System (GNSS) solution to deal with the typical constraints faced by IoT sensors by migrating the signal processing tasks from the sensor to cloud servers. Theoretical and experimental results demonstrate the feasibility of a cloud-based GNSS approach in energy efficiency, performance, and economic terms.

VICENTE LUCAS-SABOLA
IEEC-CERES, UNIVERSITAT AUTÒNOMA DE BARCELONA

GONZALO SECO-GRANADOS
IEEC-CERES, UNIVERSITAT AUTÒNOMA DE BARCELONA

JOSÉ A. LÓPEZ-SALCEDO
IEEC-CERES, UNIVERSITAT AUTÒNOMA DE BARCELONA

JOSÉ A. GARCÍA-MOLINA
EUROPEAN SPACE AGENCY, ESA/ESTEC

In recent years, modern society has moved towards the use of emerging technologies in order to facilitate day-to-day decisions while optimizing resources in an automatic way. This is the case of the Internet of Things (IoT), where physical objects such as bikes, wearables, urban furniture, etc., are connected within a network with the mission of providing some kind of information (e.g., temperature, humidity, lighting, etc.) that can later be used in different applications and services (see Additional Resources, D. Singh *et alia*). As an example, a smart city uses the information gathered by multiple IoT sensors distributed in an urban area to optimize the efficiency of city operations: waste management, smart lighting, traffic congestion, etc. The goal is to make cities more sustainable places and to manage them in a more effective, efficient, and social manner. In this context, positioning information remains a key component for a wide range of applications that use multi-sensor data for real time sensing or crowd-sourcing, among others (M. Batty *et alia*).

In general, IoT sensors must cope with many key challenges: identification, information privacy, security, interoperability, low-cost, etc. (R. Khan *et alia*). More importantly, even though semiconductor technologies are evolving by leaps and bounds, one of the main challenges IoT positioning sensors must face is power consumption. The battery life of a sensor is expected to last as long as possible (on the order of 10 years) in order to minimize human maintenance and hence reduce costs. To achieve a longer battery lifetime, IoT sensors usually work with short duty cycles: they remain in sleep mode, where the power consumption is significantly low (on the order of μA), and only swap to active mode (power consumption on the order of mA) when they sense data and communicate it to a central node. IoT positioning sensors often use the Global

Navigation Satellite System (GNSS) due to its coverage and its ease of use, i.e., the user is not required to install any kind of infrastructure. However, GNSS chipsets are power hungry devices, incurring a considerable decrease of the IoT sensor battery lifetime, an aspect that vendors are trying to tackle with the development of low-powered GNSS chipsets. Furthermore, as we will see in the next section, the performance of GNSS receivers is jeopardized in challenging environments such as indoor, light-indoor, or urban scenarios (G. Seco-Granados *et alia*).

Current GNSS IoT positioning solutions compute the so-called Position, Velocity, and Time (PVT) on the sensor itself, hence requiring a certain amount of computational resources (i.e., CPU and RAM) and thus consuming a certain amount of power. This is further aggravated by the increase of data (e.g., signals from multiple GNSS constellations) to be processed and the complexity of the GNSS signal processing techniques to be applied. This article sheds some light on the challenges of current IoT positioning sensors and proposes the use of a cloud-based GNSS positioning approach, in which the computational tasks typically carried out on-chip are migrated to a cloud server with the objective of enhancing the sensor's battery lifetime without compromising the performance (V. Lucas-Sabola *et alia*, 2016). The processing of GNSS signals in remote servers was initially proposed in the 1990s to reduce power consumption and economic cost of positioning sensors (A. Brown and R. Silva). Nowadays, the high-scalability and low-cost offered by cloud computing services make them the perfect choice for implementation of remote GNSS signal processing. These services require less energy than GNSS modules (J. Liu *et alia*; V. Lucas-Sabola *et alia*, 2017).

IoT Positioning Sensors

IoT positioning solutions can be divided into three main groups: those based on GNSS, those based on non-GNSS, and those combining both GNSS and non-GNSS technologies in a hybrid manner. Outdoor IoT positioning typically relies on the use of GNSS modules that are in charge of capturing the GNSS signal transmitted by the satellites from one or multiple constellations such as Global Positioning System (GPS), Galileo, GLONASS, or BeiDou and applying the necessary signal processing techniques in order to obtain the PVT. Even though GNSS were originally designed for outdoor environments, novel techniques are designed to boost the performance in indoor scenarios, including the exploitation of distributed Receivers of Opportunity (RoO) located in close-by locations in a cloud-based GNSS framework (J. A. García-Molina *et alia*) and the implementation of advanced GNSS signal processing techniques. On the other hand, indoor IoT positioning usually relies on non-GNSS technologies namely 4G/Long Term Evolution (LTE), Wireless Local Area Network (WLAN), Low-Power Wide-Area Network (LPWAN), Ultra-Wide-Band (UWF), or Inertial Navigation Systems (INS). Eventually, IoT positioning sensors may perform on-chip hybrid positioning using a combination of GNSS and non-GNSS technologies at the expense of higher power consumption and cost (G. De Angelis *et alia*).

An IoT positioning sensor is typically composed of a MicroController Unit (MCU), a reception/transmission Radio-Frequency (RF) front-end (so-called communication module), an antenna, the respective positioning module, memory and the power supply (i.e., battery). The MCU is the brain of the sensor, an integrated circuit that includes one or more CPUs and a low capacity RAM able to perform basic computational tasks. The communication module (includes reception and transmission front-end) receives the requests and transfers the data or information to a central node. The positioning module varies depending on the technology used (e.g., GNSS, INS, LTE). In this article we only focus on GNSS-based solutions.

The positioning module of a GNSS-based IoT positioning sensor is a GNSS module, as depicted in Figure 1(a), which is in charge of capturing and conditioning the GNSS signals of interest with its own RF front-end (usually included in the GNSS module) and processing them to compute the position. Positioning data is often delivered by means of the National Marine Electronics Association

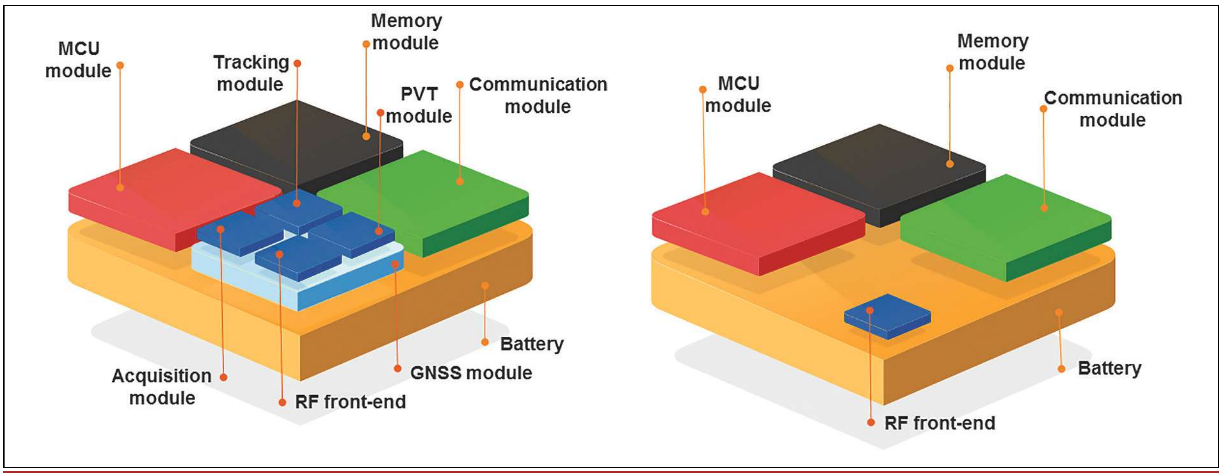
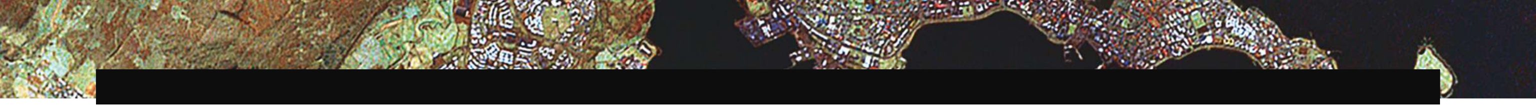


FIGURE 1 (a) Conventional GNSS IoT positioning sensor; (b) Cloud GNSS IoT sensor



(NMEA) protocol, producing a file that contains information regarding the position of the sensor, visible satellites, measurements, pseudoranges, etc., whose size is in the range from 1 to 5 kilobytes (kB). In order to reduce the amount of data to be transferred from the IoT sensor to the central node (uplink transmission), the NMEA is processed on-board and just the location of the sensor is delivered in the end, hence reducing the output to a few bytes. During the time the GNSS module is in active mode, it essentially switches between acquisition and tracking state. In the acquisition state, the GNSS module is searching and acquiring GNSS signals until it is capable of providing a position fix. This is the maximum power-consuming state of the GNSS module. Afterwards, it switches to a tracking state that provides position fixes with the already acquired satellite signals and searches for signals of new visible satellites. The tracking state is considerably less power-consuming than the acquisition state. Nevertheless, it is difficult to measure the power consumption of a GNSS module as it varies depending on the working conditions. For instance, a satellite with low Carrier-to-Noise ratio (C/N0) would require a longer coherent or non-coherent integration at the acquisition stage, thus needing a larger amount of computational resources which implies an increase in the power consumption.

Hence, one of the goals most Mass-Market (MM) GNSS chipset vendors have (in addition to achieving lower power consumptions) is to reduce the active time until providing a reliable position fix, also known as Time-To-First-Fix (TTFF). The TTFF depends on the starting mode at which the GNSS module initiates when it is switched from sleep to active mode. There are four different starting modes: cold, warm, assisted, and hot (F. Van Diggelen). In a cold start, all the possible frequency and code delays are searched and the ephemeris and broadcast time are decoded. In a warm start, only the broadcast time and ephemeris are decoded, as the frequency and code delays are held as prior information. In a hot start, frequency and code delays, broadcast time, and ephemeris data are already known. Novel GNSS modules include assisted start, which allows downloading ephemeris and broadcast time information from private servers or GNSS Data Centers (GDC) in order to achieve a faster TTFF, but requires a downlink internet connection. After downloading the assistance data, the GNSS module is able to perform an assisted start. The TTFF estimates of a GNSS module for cold, warm, assisted, and hot starts (GPS L1 C/A only) are approximately 44.34, 20.52, 2 and 0.51 seconds, respectively (M. Anghileri *et alia*). However, larger TTFF may be achieved in harsh environments due to the difficulties in decoding the broadcast message or ephemeris, larger acquisition times, etc.

The emergence of IoT positioning applications has sparked the interest of several MM GNSS vendors. Indeed, one manufacturer provides Assisted GNSS (A-GNSS) services to their GNSS modules at system start-up to minimize the TTFF. Similarly, another operates a worldwide reference network to provide A-GNSS data to its users, thus boosting the TTFF speed and accuracy. Furthermore, novel GNSS modules from another vendor includes two Power Save Modes (PSMs) to reduce the average power consumption: Cyclic Tracking (PSMCT) for short update periods (1-10 seconds) and On/Off (PSMOO) for long update periods (larger than 10 seconds). Of particular interest is the PSMOO which is suitable for IoT applications with long update periods (typically from hours to days). Nonetheless, the use of the PSMOO may provide significant error in the position fix. Similar power modes are available in one company's GNSS modules: the GNSS Low Power (GLP) and the periodic mode, analogue to the PSMCT and PSMOO modes provided by another MM GNSS vendor, respectively, with the latter being the most suitable for IoT applications. See Additional Resources for more information on all of these vendors.

To sum up, MM GNSS chipsets offer power saving configurations oriented to IoT applications. However, a tradeoff is faced between accuracy and power consumption, which varies depending on the application or use. Additionally, the use of power save modes leads to a degradation in the accuracy performance of GNSS chipsets. Together with a reduction in power consumption, diminishing the TTFF becomes mandatory in order to minimize the amount of time the GNSS chipset is in active mode. To do so, vendors provide A-GNSS services so the GNSS module can implement an assisted start. Therefore, the IoT positioning sensor would require a downlink channel for downloading the GNSS assistance data, whose size is in the range of 1 to 3 kB per constellation. Note that IoT sensors do not typically use the downlink channel as they are already pre-configured to switch between states beforehand, and hence this feature is only occasionally used.

Cloud GNSS Receiver

In the previous section we discussed power-hungry GNSS modules and how their accuracy performance is jeopardized as power consumption is reduced. We now propose a cloud GNSS receiver that performs the GNSS signal processing tasks in a cloud server instead of on the sensor itself, thus facilitating the computational resources required by the IoT positioning sensor and hence reducing its power consumption without compromising the accuracy. In addition, the cloud GNSS receiver paves the way for innovative and more advanced applications due to the amount of available computational resources in the cloud servers: secure and authenticated GNSS positioning, crowdsourcing GNSS signal processing, pay-per-use insurance, etc.

The cloud GNSS receiver is considered a Software as a Service (SaaS), a remote application that can be used by any user or machine while it is completely transparent to him. That is to say, its services can be used without having any kind of knowledge of the software, algorithms, and computing resources used in the *back-end*. In this work, the cloud computing resources and services used are provided by Amazon Web Services (AWS) due to their wide range of cloud solutions, functionalities, and configuration options, along with their openness, flexibility, and low cost.

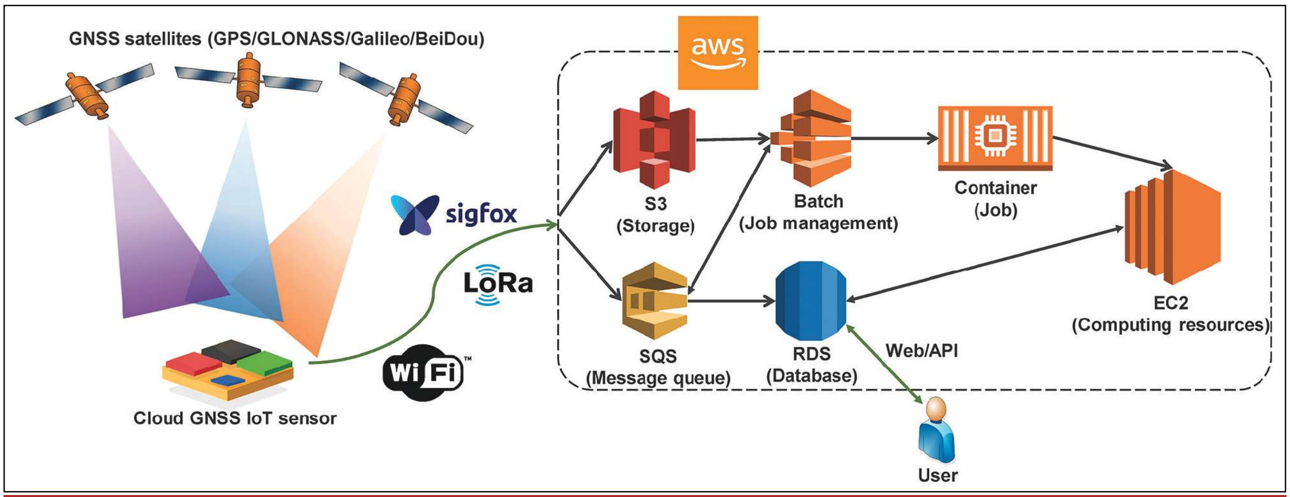


FIGURE 2 Cloud GNSS receiver architecture

Architecture

The architecture of the cloud GNSS receiver is composed of three main elements (Figure 2): the *cloud GNSS sensor*, the *cloud front-end* in charge of interacting with the user, and the *back-end* module where a High-Sensitivity (HS) GNSS Software Receiver (SwRx) is running and where all the computational tasks, reporting, and delivery of results are carried out. The *cloud GNSS sensor* is the IoT hardware element in charge of gathering the raw GNSS samples at the user side, and sending them to the cloud GNSS receiver for subsequent processing. It is composed of an RF front-end tuned to the GNSS band of interest that includes an Analog-to-Digital Converter (ADC) for digitizing the GNSS signals, memory, and a communication module for interacting with the cloud GNSS receiver.

The *cloud front-end* is the interface through which a user or a machine, Human-to-Machine (H2M) and Machine-to-Machine (M2M), respectively, interacts with the cloud GNSS receiver. In the H2M approach, the user can access the cloud GNSS through an HTTP web service, where users can log in and enter into a private desktop. Then, new executions or jobs can be launched using an online graphic user interface that allows for configuration of the HS-GNSS SwRx. Notwithstanding, H2M interfacing is not a scalable approach and it is not suitable for large *cloud GNSS sensor* networks. It is for this reason that a M2M interface becomes mandatory. In this sense, an Application Programming Interface (API) has been built to allow sensors to automatically connect with the cloud GNSS receiver. Afterwards, the output results, PVT being an example, are stored in a database and can be retrieved at any time by the user. Both API and webpage are used to generate a new job with a raw GNSS sample file and a JavaScript Object Notation (JSON) file as inputs. The JSON is used to configure the HS-GNSS SwRx to the specific needs of the analysis to be done, and to the working conditions where the samples were gathered with parameters such as the number of snapshots, the GNSS band to be processed, the coherent and non-coherent integration time, etc. The flexibility offered by the cloud GNSS receiver allows the choice of some advanced features including using long integration times for processing weak GNSS signals, implementing signal-level analysis, etc. Metadata associated with the capture of raw GNSS samples including RF and Intermediate Frequency (IF), sampling rate, quantization, encoding, etc., can be included in the JSON or by using the Institute of Navigation (ION) Software-Defined Radio (SDR) metadata standard (J. Curran *et alia*). Likewise, assistance information regarding the list of satellites to be searched together with their Doppler frequency can also be attached to decrease the execution time (like a hot start in an MM GNSS receiver). Assistance information can be automatically generated by the cloud GNSS receiver by attaching an approximate location and the timestamp. Finally, a Receiver Independent Exchange Format (RINEX) navigation file must be attached in order to calculate the PVT of the sensor. In this context, the cloud GNSS receiver also procures the capability of downloading RINEX navigation files from servers of the International GNSS Service (IGS). This is mainly to avoid the need for capturing GNSS signals during a long interval of time (i.e., at least 30 seconds in GPS L1 C/A are required to read and decode the ephemeris embedded into the navigation message) and hence reduce the amount of data to be captured and transferred to the cloud to a few milliseconds of signal. Lastly, the files and job instructions generated by the *cloud front-end* (i.e., raw GNSS sample file, JSON, API commands) are transferred and communicated to the cloud *back-end*.

Each new job generated by a user or IoT positioning sensor is received and managed by a resource manager in charge of allocating cloud computing resources to the job. The *back-end* of the cloud GNSS receiver is where the bulk of the processing tasks are carried out. This comprises reading, decoding, and processing the raw GNSS sample file given the configuration parameters included in the JSON file to finally obtain the desired output such as PVT. This process is implemented by means of an HS-GNSS software receiver, a snapshot-based GNSS receiver with a C/N_0 sensitivity down to 15 dBHz (J. Lopez-Salcedo *et alia*). The HS-GNSS SwRx core is based on the extensive use of FFT processors and implements long integration times up to several seconds using advanced non-coherent integrations.

In addition, different AWS solutions are used to build the *back-end*: Elastic Cloud Computing (EC2), Simple Queue Service (SQS), Batch, Relational Database Service (RDS), Elastic Container Service (ECS), and Simple Storage Service (S3). EC2 provides re-sizable and scalable computing resources (i.e., RAM, CPU, storage, etc.) in the cloud as instances (virtual machines) that act as a physical computing machine. There is a wide range of available virtual machine types, each of them suitable for different uses. SQS is a message queuing service to communicate different modules and systems within the cloud infrastructure. Batch enables us to optimally compute jobs on AWS resources (i.e., EC2) based on its computing requirements (e.g., CPU, RAM). RDS is used to store and manage the database of the cloud infrastructure which includes information such as users, PVT results, job configurations, etc. ECS allows for the launch and scale of *Docker* containers on AWS when a job is received. A *Docker* container includes the HS-GNSS SwRx and all the necessary software for the processing of the raw GNSS sample file. S3 provides secure, durable and highly-scalable object storage and is used to store the data uploaded from the sensor, which is then downloaded in an EC2 virtual machine in order to be processed.

Thus, when a cloud GNSS IoT sensor (Figure 1(b)) launches a request, it first has to upload the raw GNSS sample file and the configuration data (i.e., JSON) to the S3 repository and send a request by means of a message to the SQS queue. When the SQS message is read by the resource manager, a new job is generated, inserted into the database by its identification number, and then managed by Batch, which will start a given EC2 instance type depending on the computing resources required by the job, and launch a *Docker* container from the ECS service. Once the container is launched, it automatically downloads the data regarding to the identification number of the job (i.e., raw GNSS sample file, JSON) from S3 and the HS-GNSS SwRx is launched. Finally, the output results are stored in the database from which it can be retrieved through the API or web page. This workflow is depicted in Figure 2.

Energy Consumption

As we have previously seen, instead of locally processing the GNSS data, the cloud GNSS IoT sensor (Figure 1(b)) only has to transfer it to a cloud server. This simple process is expected to be more energy efficient than conventional IoT positioning approaches where the PVT is computed on-chip. Nevertheless, it is known that data transmission is one of the most energy consuming stages of an IoT sensor. Depending on the application, it is even higher than performing the computational tasks locally in the device itself. In the cloud GNSS receiver, the size of the data (i.e., raw GNSS sample file) to be transmitted from the sensor to the cloud is directly proportional to the signal length (in time), the sampling frequency of the reception RF front-end, and the quantization of its ADC. Therefore, a trade-off is met between the size of the data to be transmitted and the energy consumed by the cloud GNSS IoT sensor (Figure 1(b)). Furthermore, the signal length is also related to the sensitivity of the GNSS receiver, as increasing the coherent or non-coherent integration time allows for the detection and acquisition of weaker GNSS signals. Hence, another trade-off is met between the sensitivity of the GNSS receiver and the size of the data to be transferred.

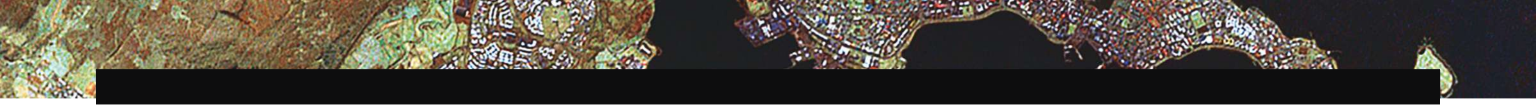
Component	Manufacturer	Model	Current consumption
GNSS module	u-blox	MAX-M8W	32 mA (Acquisition) [20]
GNSS module	u-blox	MAX-M8W	8.9 (Tracking) [20]
MCU module	Texas Instruments	CC1310	1.9-2.5 mA [18]
Communication module	Texas Instruments	CC1310	11.2 mA [18]
Memory module	Atmel	AT25M02	5 mA [2]

Table 1 Current concumption of the components of a conventional GNSS IoT sensor (Figure 1a)

In order to properly compare the expected energy consumption of both the conventional and cloud GNSS IoT sensors (Figure 1), we have to address the energy consumed by each of the components they comprise (V. Lucas-Sabola *et alia*, 2017). First, the energy consumption of a conventional GNSS IoT sensor (Figure 1(a)) performing a position fix is discussed. Basically, the required energy by a component can be obtained by its current consumption in active mode, the amount of time in active mode, and the supply voltage. For this study case, the supply voltage is set to 3.3 V. The current consumption has been obtained from datasheets of state-of-the-art components (**Table 1**). Regarding the current consumption of the components, the reader should note that the communication module also consumes an additional amount of energy due to the release of power from the antenna during transmission, and the GNSS module current consumption may vary depending on the working conditions of the sensor (e.g., open, mild, or obstructed environment).

The active time of the different components have been obtained as follows: the MCU is active from the time the sensor is switched on and begins capturing the GNSS signal until the PVT is sent; the active time of the memory and communication module is directly dependent on the size of the packet to be stored and transmitted, respectively (a few bytes in this study case); and the GNSS module active time is equal to the TTFF for each of the four different starts: 44.34, 20.52, 2, 0.5 seconds for cold, warm, assisted, and hot, respectively (TTFF for assisted depends on the downlink latency). Note that the acquisition time may vary depending on the working conditions, i.e., larger in harsh environments and hence increasing the energy consumption.

In **Figure 3** the expected energy consumption of the different components of a conventional GNSS IoT positioning



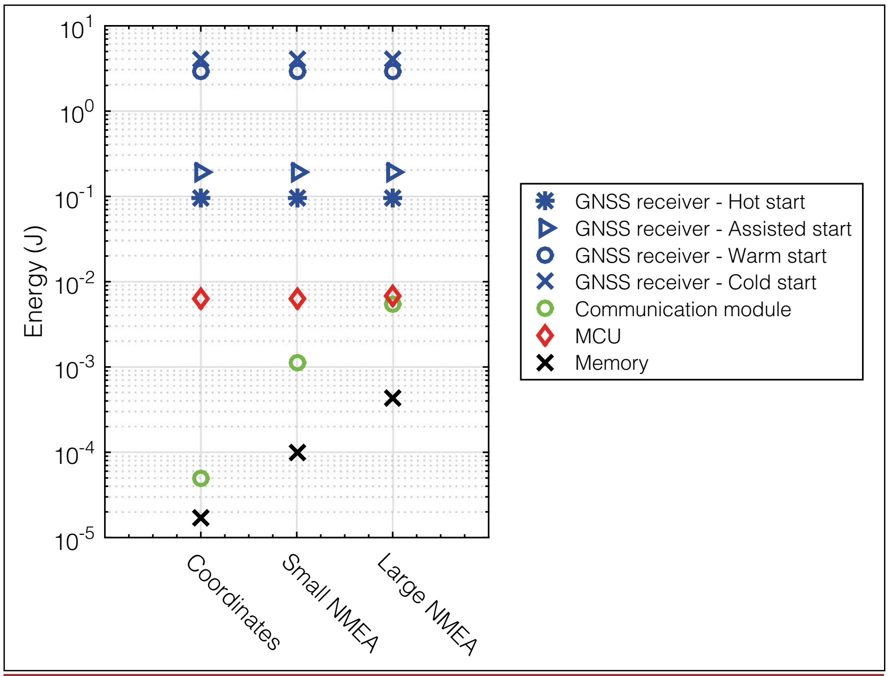


FIGURE 3 Expected energy consumption of each of the components of a conventional GNSS IoT sensor (see Figure 1(a)) with regard to the size of the data packet transmitted through the communication module; TTFF for hot, assisted, warm, and cold starts of 44.34, 20.52, 2, and 0.5 seconds, respectively

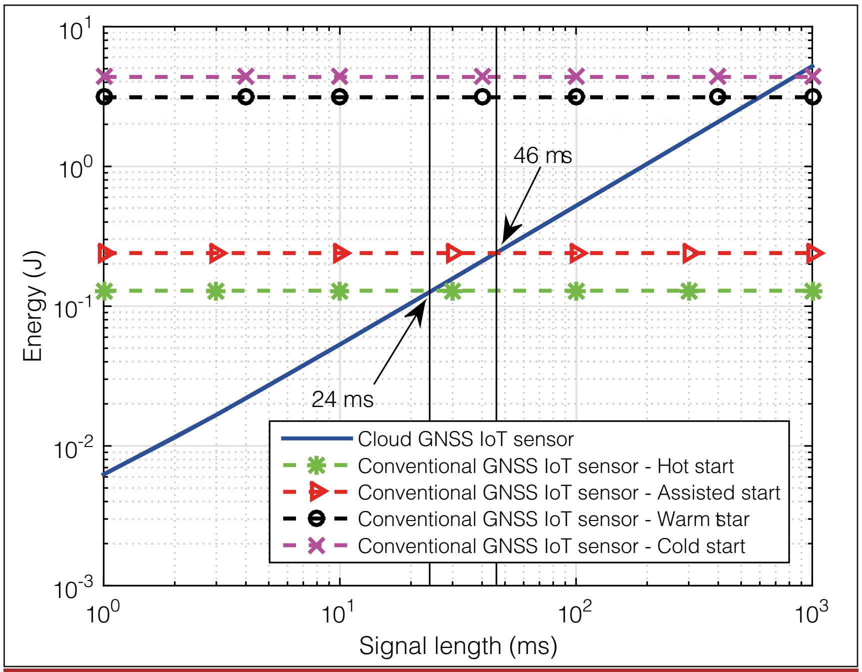


FIGURE 4 Expected energy consumption of a cloud and a conventional GNSS IoT sensor with respect to the signal length; TTFF for hot, assisted, warm, and cold start of 44.34, 20.52, 2, and 0.5 seconds, respectively

sensor (Figure 1(a)) is shown. It can be seen that the energy consumption of the GNSS module depends on the operation mode (i.e., hot, assisted, warm, cold), and hence depends on the TTFF. In contrast, the rest of the sensor components such as the MCU, memory, and communication modules have an energy consumption directly dependent on the packet to be handled and transmitted to the cloud. Indeed, the MCU must stay in active time until the data (i.e., PVT output) is sent to, for example, a central node. The size of the packet to be transmitted through the communication module when only the coordinates are sent is approximately 1-2 bytes, 1 kB for a small NMEA, and 5 kB for a large NMEA. We can clearly see that the GNSS module typically is the largest consumer of a conventional GNSS IoT sensor (Figure 1(a)), even for hot and assisted starts by an order of magnitude in comparison with the MCU and up to four orders of magnitude in contrast with the communication and memory modules.

For the cloud GNSS IoT sensor (Figure 1(b)), the same procedure has been applied, but the GNSS module has been substituted with only a GNSS RF front-end, also including an ADC. In this sense, the GNSS RF front-end must be carefully chosen to avoid compromising the energy-efficiency of the cloud GNSS IoT sensor (Figure 1(b)), as it will not only contribute to its own energy consumption, but also in the energy consumed by the other components as they are directly dependent on the packet size. Indeed, the size of the data to be transferred to the cloud can be expressed as $L = T_{sl} b Fs_{rx}$, where L is the packet size in bits, T_{sl} is the captured GNSS signal length in seconds, b is the quantization bits of the ADC, and Fs_{rx} is the sampling frequency of the RF front-end. Therefore, there are three alternatives in order to work with small-sized packets: reduce the sampling frequency, work with a low quantization level, or capture a signal of short length. In this study case, with the objective of achieving low energy consumption, the RF bandwidth is set to 2 megahertz, enough to capture the main lobe of the GPS L1 C/A signal, and the sampling frequency is set to 4 megahertz. On the other hand, the ADC uses 1 bit to digitize the GNSS signal. According to state-of-the-art components, the current consumption of a GNSS RF front-end with such characteristics is approximately 5 mA.

A comparison between the energy consumed by the cloud and the conventional GNSS IoT sensor (Figure 1) with different starts is addressed in **Figure 4**. In order to implement a fair comparison, the energy consumption of the conventional GNSS IoT sensor (Figure 1(a)) for different starting modes has been obtained for a packet with size 2

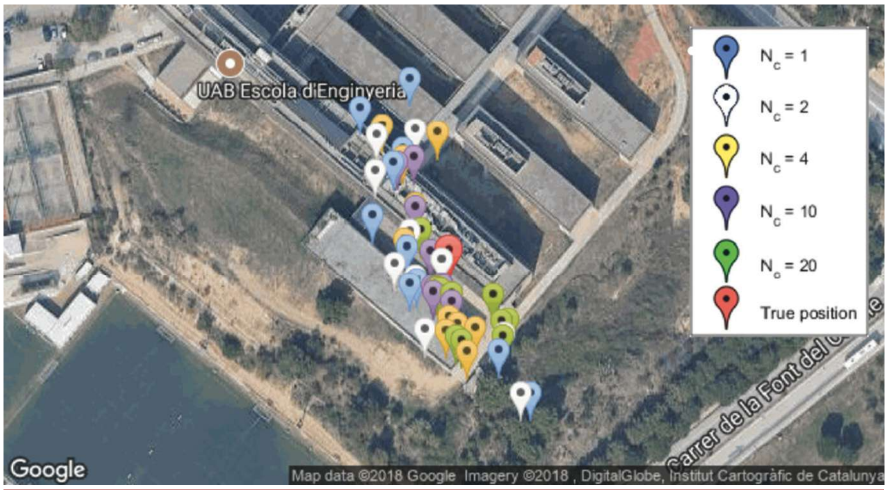


FIGURE 5 Accuracy performance of the cloud GNSS receiver in a static outdoor clear-sky scenario with different coherent integration times

bytes (see Figure 3). It can be seen how as the signal length to be captured by the cloud GNSS IoT sensor (Figure 1(b)) is moderately small, the cloud GNSS receiver offers a more energy efficient positioning solution than conventional approaches. For instance, compared to a conventional sensor with hot start, the cloud sensor provides energy savings up to one order of magnitude for small signal lengths (i.e., a few ms) and is more energy efficient by using a signal length up to 24 ms (bear in mind that a PVT can be obtained with just 1 or 2 ms of signal). Notwithstanding, the GNSS module cannot implement a hot start every time it provides a position fix, as the stored information expires (range from 30 minutes to 4 hours). MM GNSS chipsets usually download or try to decode ephemeris data (as in cold start) every 30 minutes with the objective of having recent ephemeris and navigation data and then provide a higher position accuracy. Likewise, in contrast with an assisted start, the cloud GNSS IoT sensor (Figure 1(b)) can capture and transfer up to 46 ms of GNSS data and still continue to be more energetically efficient. Finally, in comparison with the warm and cold starts, whose TTFF and hence the amount of time in active mode is significantly high (i.e., 30-40 seconds), the cloud GNSS IoT sensor (Figure 1(b)) remains active for some milliseconds (up to 600 and 800 ms, respectively), and thus energy efficiency can reach up to 2.5 orders of magnitude.

Performance of the Cloud GNSS Receiver

In this section we will briefly discuss the accuracy performance of the cloud GNSS receiver by means of an experimental test. To do so, a synthetic signal was generated with a GPS/Galileo signal generator simulating a static outdoors open-sky scenario at the School of Engineering of Universitat Autònoma de Barcelona (UAB). A low-cost RTL-SDR V3 USB front-end was used to capture the GPS L1 C/A signal with a sampling frequency of 2.048 MHz and generate the raw GNSS sample file, which is then transferred to the cloud GNSS receiver together with the required JSON configuration file. The visible satellites have been configured with a C/N_0 of roughly 46 dBHz.

The obtained results for GPS L1 C/A with different coherent integration times are provided in **Figure 5**. For a small signal length (i.e., 1 to 4 ms) a position error of tens of meters is obtained. However, as the coherent integration time is increased (i.e., 10 and 20 ms), and hence the amount of signal captured and sent to the cloud GNSS receiver is also larger, the positioning accuracy is enhanced to a few meters. In contrast to conventional GNSS IoT positioning approaches where the sensor must be in active mode for a long period of time to calculate the PVT (from a few seconds up to minutes, depending on the working conditions), in a cloud-based approach the sensor must be in active mode for just a few milliseconds, enough time to capture the desired GNSS signal and forward it to the cloud servers. Therefore, it is shown that the cloud GNSS receiver becomes an energy-efficient IoT positioning solution without compromising the obtained accuracy, particularly for those IoT applications that do not require precise positioning.

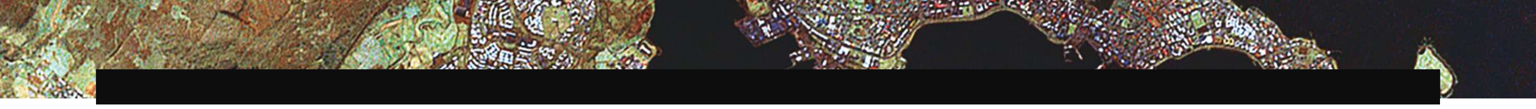
Economic Cost of GNSS Signal Processing in the Cloud

Processing the raw GNSS sample file in cloud servers instead of in the sensor itself as in conventional approaches implies the added cost of hiring cloud computing resources. Among the different AWS services used in the cloud GNSS receiver infrastructure, the service that implies a significant cost is the EC2 service. AWS offers three ways to pay for its services: on-demand, reserved, and spot. On-demand services are paid per hour use as they are opened and closed with a fixed cost. Reserved services, which are paid in advance, are suited for applications with predictable usage and offer discounts up to 75% in contrast with on-demand services. Moreover, we can bid for EC2 spot services whose cost may decrease up to 90% in comparison with on-demand (their fluctuating price varies depending on the supply and demand of EC2). Additionally, the price of different services may vary depending on the selected region of use, i.e., the geographic area where the EC2 servers are hosted.

For this test set-up, c3.xlarge (**Table 2**) have been selected to compose the cloud *back-end* for processing the raw GNSS sample files with signal length between

Instance type	c3.xlarge
vCPUs	4
RAM	7.5 GB
SSD	2x40 GB
Region	Frankfurt (EU)
On-demand price (taxes excluded)	$0.258 per hour
Reserved price (taxes excluded)	$0.11 per hour
Spot price (taxes excluded)	$0.0472 per hour

Table 2 Experimental set-up for the economic cost of cloud services: EC2 characteristics



1 and 5 ms. The allocation of the jobs generated by the requests of a network of cloud GNSS IoT sensors (Figure 1(b)) is assumed to be optimum in terms of usage time: the computational tasks of a given amount of sensors are sequentially performed in the same EC2 instance, hence reducing the cost per position fix. The monthly cost of the necessary cloud resources (i.e., EC2) for an IoT application that requires one position fix per hour is presented in Table 3: for $0.51, $0.23 and $0.1 per month, a cloud GNSS IoT sensor (Figure 1(b)) position can be calculated once per hour for on-demand, reserved and spot services, respectively. Notice that in typical IoT positioning applications, a position fix is usually requested from hours up to days. Hence, the cost of the cloud services used by a network of cloud GNSS IoT sensors (Figure 1(b)) should not be a showstopper as it has been demonstrated to be considerably low.

Payment type	Monthly cost
On-demand instances	$0.51
Reserved instances	$0.23
Spot instances	$0.10

Table 3 Monthly cost (taxes excluded) of cloud services (c3.xlarge EC2) employed by an IoT application that requires one position fix per hour; signal length set from 1 to 5 m

Conclusions

This article discusses the conventional solutions for IoT positioning, with GNSS-based solutions being the most widespread in positioning IoT sensor networks. The architecture of a conventional GNSS IoT positioning sensor (Figure 1(a)) has been addressed, together with the energy consumption of its different components, showing that the GNSS module is the largest consumer if the data to be transferred is not large. To tackle the dilemma of energy consumption in IoT positioning sensors, we propose the use of a cloud-based GNSS approach, in which the purpose of sensors is just to capture the GNSS signal and send it to a cloud server where it will then be processed.

The energy consumption of the proposed cloud GNSS IoT sensor (Figure 1(b)) has also been addressed and compared with state-of-the-art GNSS IoT positioning sensors. Under the constraint of working with a relatively small signal length, the use of the cloud GNSS receiver achieves a significant savings in the sensor's consumed energy, up to one order of magnitude compared with hot and assisted starts, and up to roughly 2.5 orders of magnitude in contrast with warm and cold starts.

Finally, the economic cost implied by the use of cloud services to process the GNSS data and obtain the position of the sensor has been shown to be low, thus ensuring the cloud GNSS receiver as a low-energy and low-cost solution for IoT positioning.

Acknowledgements

The views presented in this article represent solely the opinion of the authors and not necessarily the view of ESA. This work was partly supported by the European Space Agency (ESA) under contract No. 4000119070/16/NL/GLC and by the Spanish Government under grant TEC2017-89925-R.

Manufacturers

When the authors address the emergence of IoT positioning applications sparking the interest of MM GNSS vendors, they note **u-blox**, Thalwil, Switzerland, **Telit**, London, UK, and **Broadcom Corp.**, Irvine, CA, as examples.

In the section discussing the accuracy performance of the cloud GNSS receiver's experimental test, a synthetic signal was generated using a **Spirent** (West Sussex, UK) GPS/Galileo signal generator. Furthermore, the cloud GNSS receiver was using the baseline broadcast ephemeris from the **IGS** service.

Additional Resources

[1] Anghileri, M., M. Paonni, S. Wallner, J.-Á. Ávila-Rodríguez, and B. Eissfeller, "Ready to Navigate! A Methodology for the Estimation of the Time-to-First-Fix," *Inside GNSS*, Volume: 5, Issue: 2, 2010

[2] Atmel, "SPI Serial EEPROM - ATM25M02," *Data Sheet*, 2017

[3] Batty, M., K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart Cities of the Future," *European Physical Journal Special Topics*, Volume: 214, Issue: 1, 2012

[4] Bousquet, F. and u-blox, "Portables: The Challenge of Low Power and Good GNSS Performance," *White Paper*, 2017

[5] Broadcom Corporation, "Top Ten Advantages: AGPS Server and Worldwide Reference Network," 2007

[6] Brown, A. and R. Silva, "TIDGET Mayday System for Motorists," *IEEE Position Location and Navigation Symposium (PLANS)*, 1994

[7] Curran, J., M. Arizabaleta, T. Pany, and S. Gunawardena, "The Institute of Navigation's GNSS SDR Metadata Standard," *Inside GNSS*, Volume: 12, Issue: 6, 2017

[8] De Angelis, G., A. De Angelis, V. Pasku, A. Moschitta, and P. Carbone, "A Hybrid Outdoor/Indoor Positioning System for IoT Applications," *Proceedings of the 1st IEEE International Symposium on Systems Engineering (ISSE 2015)*, 2015

[9] García-Molina, J. A. and J. M. Parro-Jiménez, "Cloud-based GNSS Processing of Distributed Receivers of Opportunity: Techniques, Applications and Data-Collection Strategies," *6th International Colloquium - Scientific Fundamental Aspects of GNSS/Galileo*, 2017

[10] Khan, R., S. U. Khan, R. Zaheer, and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," *Proceedings of the IEEE 10th International Conference on Frontiers of Information Technology (FIT 2012)*, 2012

[11] Liu, J., B. Priyantha, T. Hart, Y. Jin, W. Lee, V. Raghunathan, H. S. Ramos, and Q. Wang, "CO-GPS: Energy Efficient GPS Sensing with Cloud Offloading," *IEEE Transactions on Mobile Computing*, Volume: 15, Issue: 6, 2016

[12] López-Salcedo, J., Y. Capelle, M. Toledo, G. Seco, J. López Vicario, D. Kubrak, M. Monnerat, A. Mark, and D. Jiménez, "DINGPOS: A Hybrid Indoor Navigation Platform for GPS and GALILEO," *21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, 2008

[13] Lucas-Sabola, V., G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Cloud GNSS Receivers: New Advanced Applications Made Possible," *Proceedings of the International Conference on Localization and GNSS (ICL-GNSS)*, 2016

[14] Lucas-Sabola, V., G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Efficiency Analysis of Cloud GNSS Signal Processing for IoT Applications," *Proceedings of ION GNSS (ION GNSS 2017+)*, 2017

[15] Seco-Granados, G., J. A. López-Salcedo, D. Jiménez-Baños, and G. López-Risueño, "Challenges in Indoor Global Navigation Satellite Systems," *IEEE Signal Processing Magazine*, February 2012

[16] Singh, D., G. Tripathi, and A. J. Jara, "A Survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services," *2014 IEEE World Forum Internet of Things (WF-IoT 2014)*, 2014

[17] Telit, "K3 Series Power Modes," *Application Note*, 2017

[18] Texas Instruments, "CC1310 SimpleLink Ultra-Low-Power Sub-1 GHz Wireless MCU," *Data Sheet*, 2016

[19] u-blox, "Power Management Considerations for u-blox 7 and M8 GNSS Receivers," *Application Note*, 2014

2018
JOINT NAVIGATION CONFERENCE
Military Navigation Technology
The Foundation for Military Ops
July 9–12, 2018
Tutorials: July 9
Show Dates: July 10–11
Hyatt Regency
Long Beach, California
Classified Session: July 12
The Aerospace Corporation
Sponsored by the Military Division of the Institute of Navigation
ION
INSTITUTE OF NAVIGATION
www.ion.org/jnc

[20] u-blox, "u-blox M8 Concurrent GNSS Modules," *Data Sheet,* 2016

[21] Van Diggelen, F., *A-GPS: Assisted GPS, GNSS, and SBAS,* Artech House, 2009

Authors

Vicente Lucas-Sabola received a B.Sc. in telecommunication systems engineering in 2015 and a M.Sc. in telecommunication engineering in 2017, both from Universitat Autònoma de Barcelona (UAB). Since 2015 he has been involved in the development of a Cloud GNSS receiver. Since 2017 he has also been pursuing a PhD at the SPCOMNAV group at the Department of telecommunication and systems engineering, IEEC-CERES, UAB, dealing with topics related to Cloud GNSS signal processing for Internet of Things (IoT) applications.

Gonzalo Seco-Granados received a Ph.D. degree in telecommunications engineering from Universidad Politècnica de Catalunya and an MBA from IESE, the graduate business school of the University of Navarra. From 2002 to 2005, he was with the European Space Agency, Netherlands. Since 2006, he has been an associate professor at the Universidad Autònoma de Barcelona, where he coordinates the SPCOMNAV (Signal Processing for communications and Navigation) group, IEEC-CERES. His research interests include signal-processing techniques for advanced features of GNSS receivers and localization using next-generation wireless communications networks.

José A. López-Salcedo received his Ph.D. degree in telecommunications engineering from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 2007. He is Associate Professor at the Department of Telecommunications and Systems Engineering, IEEC-CERES, Universitat Autònoma de Barcelona (UAB). He has held several visiting appointments at the University of California Irvine, University of Illinois at Champaign-Urbana and the European Commission, Joint Research Centre in Ispra, Italy. His research interests lie in the field of signal processing for communications and navigation, with emphasis on cloud and IoT GNSS signal processing and the convergence of 5G/GNSS systems.

José A. García-Molina is a Radio Navigation engineer at ESA/ESTEC in Noordwijk, The Netherlands, where he leads several R&D projects and internal research activities on GNSS receiver technology and signal processing techniques for ground and space applications in the context of different ESA programs (including Galileo). His main research interests include signal processing and estimation theory, GNSS/Galileo receivers and signals, unambiguous estimation of high-order BOC signals, Cloud GNSS receivers, techniques and applications, collaborative positioning, and MIMO-GNSS signal processing.

Em. Univ.-Prof. Dr.-Ing. habil. Dr. h.c. Guenter W. Hein is Professor Emeritus of Excellence at the University FAF Munich. He was ESA Head of EGNOS & GNSS Evolution Programme Dept. between 2008 and 2014, in charge of development of the 2nd generation of EGNOS and Galileo. Prof. Hein is still organising the ESA/JRC International Summerschool on GNSS. He is the founder of the annual Munich Satellite Navigation Summit. Prof. Hein has more than 300 scientific and technical papers published, carried out more than 200 research projects and educated more than 70 Ph. D.´s. He received 2002 the prestigious Johannes Kepler Award for "sustained and significant contributions to satellite navigation" of the US Institute of Navigation, the highest worldwide award in navigation given only to one individual each year. G. Hein became 2011 a Fellow of the US ION. The Technical University of Prague honoured his achievements in satellite navigation with a Doctor honoris causa in Jan. 2013. He is a member of the Executive Board of Munich Aerospace since 2016. IG