Integration of GNSS RTK and PPP-RTK Capabilities into a Low-Power Node with Versatile Wireless Connectivity

Marc Subirà Pons

Resum– Aquest article presenta un projecte d'enginyeria que integra dues tècniques avançades de posicionament GNSS d'alta precisió—RTK i PPP-RTK—en un node integrat de baix consum dissenyat per a un desplegament fiable i a llarg termini en diversos escenaris del món real. RTK proporciona una precisió de posicionament a nivell centimètric aprofitant dades en temps real d'una estació base local, mentre que el PPP-RTK ofereix una major flexibilitat de desplegament mitjançant l'ús de serveis de correcció de tercers, encara que amb una lleugera pèrdua de precisió. El sistema utilitza una combinació de tecnologies de comunicació LTE, MQTT i LoRa, assegurant l'adaptabilitat a diferents entorns de connectivitat i complint amb els requisits específics de rendiment de xarxa de l'RTK i el PPP-RTK. El sistema ha estat rigorosament provat i validat, demostrant la seva idoneïtat per a aplicacions del món real que exigeixen alta precisió i fiabilitat.

Paraules clau– GNSS d'Alta Precisió, Real-Time Kinematic (RTK), Precise Point Positioning-RTK (PPP-RTK), Sistemes Integrats, Comunicació Sense Fils

Abstract– This paper presents an enginnering project that integrates two cutting-edge high-precision GNSS positioning techniques—RTK and PPP-RTK—into a low-power embedded node designed for reliable, long-term deployment in diverse real-world scenarios. RTK provides centimeter-level positioning accuracy by leveraging real-time data from a local base station, while PPP-RTK offers greater deployment flexibility by utilizing third-party correction services, albeit with a slight trade-off in precision. The system employs a combination of LTE, MQTT, and LoRa communication technologies, ensuring adaptability to varying connectivity environments and meeting the distinct network performance requirements of RTK and PPP-RTK. The system has been rigorously tested and validated, demonstrating its suitability for real-world applications that demand high precision and reliability.

Keywords– High-Precision GNSS, Real-Time Kinematic (RTK), Precise Point Positioning-RTK (PPP-RTK), Embedded Systems, Wireless Communication

1 Introduction

This project developed the necessary software and architectural components to integrate two state-of-the-art high-precision Global Navigation Satellite System (GNSS) positioning techniques into a compact, battery-powered embedded system. The primary objective was to create a reliable prototype capable of long-term deployment in real-world scenarios, catering to diverse connectivity requirements for high-precision geolocation applications.

The integrated positioning techniques are Real-Time Kinematic (RTK) and Precise Point Positioning (PPP)-RTK, both of which require a reliable real-time communication link for the exchange of differential GNSS information. Specifically, RTK operates by utilizing a nearby reference base station (hereafter referred to as the base) with a known position that transmits GNSS signal data to a positioning device (hereafter referred to as the rover). The rover processes this data to significantly enhance its positioning accuracy. Conversely, PPP-RTK eliminates the need for a local base station by connecting the rover to a third-party PPP service, albeit with some trade-offs in accuracy.

To accommodate these techniques, two operational

• E-mail de contacte: 1603424@uab.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Guillem Boquet Pujadas
• Curs 2024/2025

modes were implemented: one requiring the deployment of a dedicated base station and another operating without the need for a base station. The first mode supports deployment via Long Term Evolution (LTE) using a dedicated Message Queuing Telemetry Transport (MQTT) service or through direct communication leveraging Long Range (LoRa) modulation. This flexibility allows deployment in areas with or without existing communication infrastructure, and can reduce costs by avoiding the need for external communication services. For the second mode, custom connector software was developed to retrieve the necessary correction data from the external PPP-RTK service in real-time, trading off the cost of deploying a dedicated base station for the cost of subscribing to a third-party PPP-RTK correction service.

To address the challenges of the project, a rapid prototyping methodology was adopted, utilizing development platforms for testing prior to final integration. This approach greatly facilitated the development process and ensured effective functional testing of the software. The primary objectives included programming and configuring the device to operate as either a rover or a base station, ensuring that the RTK and PPP-RTK positioning methods remained independent of the wireless communication technologies employed. Communication channels were implemented across multiple protocols, including MQTT and Transmission Control Protocol (TCP), to ensure data integrity during transmission. Furthermore, the system incorporated Wi-Fi, LoRa, and LTE radio technologies, and a connector was developed to interface with external PPP-RTK services. To enhance deployment longevity, the node was optimized for low-power operation. Finally, several tests were conducted to validate the developed software and system's performance in real-world outdoor scenarios, ensuring its reliability and practical applicability for high-precision positioning applications.

The rest of the document is structured as follows: Section 2 explains the main technologies underlying the project; Section 3 outlines the project's objectives and the methodology employed; Section 4 provides a detailed description of the architecture, software, and communication channels necessary for RTK and PPP-RTK implementation; Section 5 details the system's software operation, including the tools and libraries developed and used; Section 6 presents the tests conducted to validate the system and their results; and Section 7 and 8 conclude the document discussing potential future work.

2 Background

This section provides an explanation of the hardware platform (Section 2.1) and the current state of high-precision GNSS positioning (Section 2.2) and communication technologies (Section 2.3) used in the development of the project. The explanation is general but emphasizes details relevant to the project's development. For a more detailed explanation, additional references are provided.

2.1 Hardware Platform

The targeted hardware is the Triumph node, a platform developed by the Wireless Networks Research Lab at the Universitat Oberta de Catalunya.

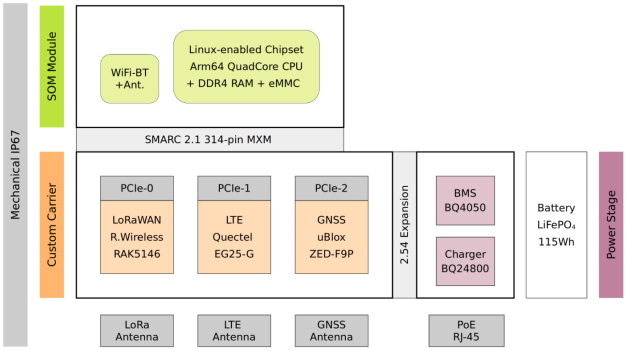

Fig. 1: Main hardware components of the Triumph node.

The Triumph node is a compact, adaptable embedded system featuring a System on Module (SOM) with an Arm64 QuadCore CPU running Linux, DDR4 RAM, and eMMC storage module. It supports wireless communication technologies such as Wi-Fi, Long Range Wide Area Network (LoRaWAN), and LTE and includes a high-precision GNSS module, a block diagram of its components is depicted in Figure 1. Additionally, the node is equipped with a robust battery management system (BMS) and a LiFePO4 battery, ensuring reliability and autonomous operation. Designed with an IP67 rating, it is suitable for deployment in harsh environmental conditions.

The specific details of the relevant components for the integration are:

- **SOM Module**. It consists of a QuadCore Arm64 CPU with a chipset capable of running a Linux-based operating system, 1GB of LPDDR4 RAM, and up to 64GB of eMMc storage. Additionally, it features a PoE port for power and SSH access to the system.
- **GNSS Module**. This module is composed of the u-blox ZED-F9P [1], a multi-band GNSS receiver compatible with RTK and PPP-RTK. It can process signals from up to 4 GNSS satellite constellations concurrently. It has two Universal Asynchronous Receiver-Transmitter (UART) and a USB interfaces. It uses a proprietary protocol named UBX on its serial interfaces for configuration writing and transmission of data generated by the receiver [2].
- **LoRaWAN Module**. This module is composed of the RAK5146 radio interface, an SPI module equipped with the LoRa SX1303 baseband processor. The SX1303 [3] processor includes all necessary hardware components to modulate/demodulate LoRa signals and extract the payload data from the received messages.
- **LTE Module**. This module is composed of the Quectel EG25-G [4], an LTE Cat 4 module compliant with the 3GPP Release 11 standard. The module offers maximum download speeds of 150 Mbps and upload speeds of 50 Mbps under LTE, with compatibility for 2G networks.

2.2 High-Precision Positioning

RTK and PPP-RTK are the two high-precision differential GNSS techniques integrated into the node, with distinct ap-

TABLE 1: COMPARISON OF HIGH-PRECISION GNSS [6]

	RTK	PPP-RTK	PPP
Init. Time[1] [s]	1	60	1800
Horiz. Acc.[2] [cm]	1	2–8	3–10
Coverage[3]	Local	Regional	Global
Bandwidth[4]	High	Moderate	Low
Infra. Density[5] [km]	10	100	1000

[1] Time required for initialization.
[2] Horizontal positioning accuracy.
[3] Geographical coverage of the technique.
[4] Required communication bandwidth.
[5] Approximate infrastructure density needed.

proaches and applications.

RTK relies on a observation-space representation (OSR) model, offering rapid and highly accurate positioning when a low-latency communication link is available. Conversely, PPP-RTK uses a state-space representation (SSR) model with global or regional coverage, enabling single-receiver solutions and reducing communication bandwidth requirements, making it more scalable. While RTK currently provides higher precision, as summarized concisely across different scales in Table 1, ongoing advancements in PPP-RTK aim to close this gap by improving convergence times.

Relevant details for the integration of each method are discussed in Section 2.2.1 and 2.2.2. The reader is referred to [5] for a complete and rigorous overview of the fundamentals, methods and applications of RTK and PPP-RTK.

2.2.1 RTK

RTK is a technique used to improve the accuracy of a GNSS receiver. Traditional GNSS systems, such as those used in mobile phones, typically have an error on the order of meters. The error in traditional systems mainly arises from atmospheric effects, clock deviations of the satellites, among others. Many of these effects are common in receivers located relatively close to each other.

RTK rectifies the measurements taken by the rover GNSS receiver using corrections transmitted from the base station, thereby eliminating environmental errors that simultaneously affect both receivers, such as atmospheric delays, satellite clock errors, and ephemeris errors. In addition, RTK calculates the relative position of the rover with respect to the base position; therefore, to obtain the geographic coordinates of the rover, it is also necessary to know by calibration the global position of the base.

RTK systems typically use the standardized RTCM format proposed by the Radio Technical Commission for Maritime Services. This format groups information into packets usually generated at a rate of 1 Hz at the base, and it is necessary for these packets to reach the rover before the messages from the subsequent second are generated. For this reason, it is important that the communication channel delay is less than one second.

2.2.2 PPP-RTK

PPP-RTK can be considered as a mixture of PPP and RTK. Conceptually, PPP-RTK is PPP but based on resolving integer phase ambiguities as the RTK technique [5]. PPP is also based, like RTK, on using corrections to improve the receiver's accuracy, but it achieves significantly worse accuracy and convergence time.

This technique provides an absolute position instead of a relative one to the base. Additionally, PPP-RTK uses fewer corrections, which are valid in much larger geographical areas compared to the coverage of an RTK base station. On the other hand, the main disadvantage of PPP-RTK is that it can take a minute or more to achieve centimeter-level accuracy, whereas RTK typically requires no more than a few seconds (see Table 1).

PPP-RTK services, also known as SSR services, aim to provide the best of both techniques. By sacrificing accuracy and convergence time, SSR services provide corrections valid for much wider regions compared to RTK. These services are usually provided by third parties, and a subscription is required, typically in the form of consumption plans that limit service use to a set number of hours per month.

PointPerfect [7], provided by the Swiss company u-blox, is an example of this type of GNSS correction service. It is compatible with any u-blox GNSS receiver and offers broad, uniform coverage across entire continents, while using a data rate of only 0.5-0.7 kbps. The service employs the SPARTN data format [8] to transmit correction data.

2.2.3 Identified Requirements

The specifications identified during the review of the state of the art required to successfully integrate the positioning techniques into the system are as follows:

- A precisely located and configured base station that continuously monitors GNSS signals and transmits corrections to the rover.
- A configured rover, implementing the advanced processing algorithm for double-differencing, carrier-phase observables, and integer ambiguity resolution, while applying real-time corrections from the base station.
- A reliable and low-latency data link to enable the real-time transfer of correction data.
- Access to multiple GNSS constellations and frequencies for robust positioning, and a clear view of the sky to maximize satellite visibility.
- For optimal accuracy, the distance between the base station and the rover should generally remain under 20 km.
- Rover compatibility with the correction data format of the PPP-RTK service.

2.3 Wireless Communication

This section briefly describes the wireless communication interfaces and technologies used in the integration.

2.3.1 LTE

LTE is a mobile telecommunications standard designed to provide high-speed wireless communication for mobile devices and data terminals [9]. LTE meets all the necessary requirements to serve as the communication channel for RTK and PPP-RTK applications, as it offers more than sufficient bandwidth and low latency. Additionally, mobile networks are widespread across much of the territory, making them accessible in a variety of environments.

2.3.2 LoRa

LoRa is a wireless modulation technique patented by Semtech. It is widely used in Internet of Things (IoT) systems due to its long range and low power consumption [10]. LoRa enables devices to communicate over several kilometers, making it ideal for applications such as remote sensing and environmental monitoring. LoRa sacrifices data rate for longer range. As a result, it provides a slower data rate compared to other technologies like LTE or Wi-Fi, typically in orders of tens of kilobits per seconds [11]. In this project, LoRa facilitates the establishment of a point-to-point communication channel between the base and the rover in areas where there is no access to cellular networks.

2.3.3 MQTT

MQTT is a standardized protocol developed by the OASIS organization for message transmission in IoT systems [12]. It operates on a client-server model with the unique feature of two distinct client types. The first type, called Publishers, transmits messages containing data to the server (MQTT Broker), which stores and classifies these messages based on the topic specified by the client. The second type, called Subscribers, connects to the server and listens to specific topics. The server then forwards all messages published under a given topic to all clients subscribed to it.

3 Objective and Methodology

The present project targets a practical solution for precise positioning applications, using low power consumption technologies and wireless communication. The solution must be designed to be applicable in environments with access to external communication infrastructures as well as in more challenging environments, where only a point-to-point connection is possible.

To address these challenges, the proposed system integrates RTK and PPP-RTK techniques into the Triumph node to achieve centimeter-level precision. To ensure communication between the rover and the base or the PPP-RTK correction service, the project integrates various communication technologies, adapting to the conditions of the environment where the rover is located.

The methodology used in this project has been rapid prototyping. This methodology is based on the agile creation of prototypes, allowing for the validation of design ideas in the early stages of the project.

The problem has been divided into several tasks and independent prototypes. This approach significantly accelerates software development by using prototyping platforms. These platforms also help validate the proper functioning of each technology and study the feasibility of their use in this project. In this way, design errors can be detected and corrected before carrying out a complex process like integration into an embedded device.

For concept testing and functionality tests prior to integration into the Triumph node, the following prototyping platforms were used:

- **Ardusimple**. The model used for testing the RTK and PPP-RTK functionalities was the Ardusimple simpleRTK2B. This model was chosen because it has the same GNSS module as the Triumph node.
- **Raspberry Pi + LoRa HAT**. Two Raspberry Pi devices equipped with a LoRa HAT (Hardware Attached on Top) were used for functional tests of communication via LoRa.

Specifically, the following setups were used in the development:

- RTK system prototype using two Ardusimple devices.
- PPP-RTK prototype using a single Ardusimple device.
- LoRa-based communication channel prototype using two Raspberry Pi units equipped with LoRa HATs.

Finally, based on the knowledge and validations obtained from prototyping, the full software integration was successfully carried out on the Triumph node.

4 Functionality and Architectures

This section describes the new positioning operational modes added to the prototype and the system architectures required to support them.

Two new positioning modes have been introduced: one requiring the deployment of a dedicated base station and another operating without the need for a base station (baseless). Additionally, two methods have been developed for wirelessly exchanging correction information: via LTE and a MQTT service for both positioning modes, and through direct communication using LoRa modulation for only the dedicated base positioning mode.

The added functionality requires the three precise positioning system architectures presented in subsections 4.1, 4.2, and 4.3. Depending on the environment in which the devices are deployed, they will be programmed to operate in a specific mode aligned with one of the proposed architectures. The developed software is described in Section 5.

4.1 Dedicated Base Internet Mode

The diagram presented in Figure 2 illustrates the initial architecture proposed for the project. This architecture is based on RTK technique. The architecture includes the three fundamental components of the RTK system: the base station, the rover, and the communication channel.

The communication channel is deployed through an MQTT server hosted on a virtual machine in Amazon Web Services (AWS), accessible via the Internet. Both the base station and the rover use the LTE cellular network to access the Internet via the TCP/IP stack. In this system, the MQTT client of the rover and the base station connect to the MQTT server. The base station acts as a Publisher and the rover as a Subscriber. Finally, the MQTT broker is responsible for

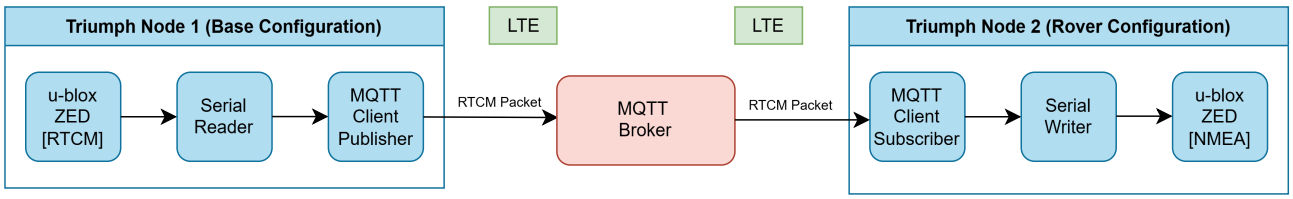

Fig. 2: Dedicated Base Internet Mode scheme.

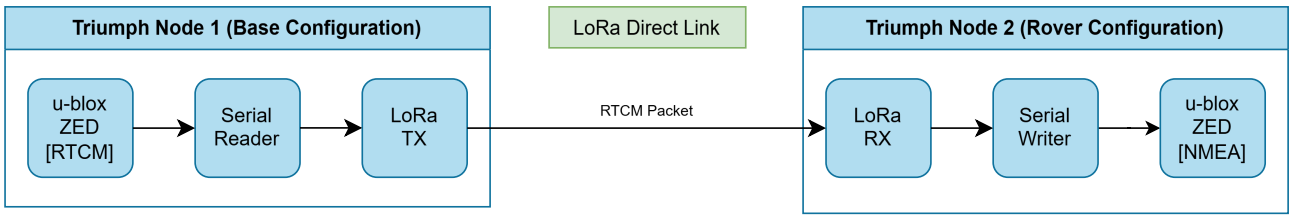

Fig. 3: Dedicated Base Direct Mode scheme.

routing all the RTCM packets published by the base in real-time.

4.2 Dedicated Base Direct Mode

The diagram in Figure 3 also uses the RTK technique. However, this architecture is designed for remote environments where access to the cellular network is not possible or not wanted.

This mode uses the wireless communication protocol LoRa to establish a direct channel between the base station and the rover, replacing the MQTT server from the previous scheme. Specific software was developed, detailed in Subsection 5.3, to handle the LoRa module for transmitting and receiving data over the LoRa protocol.

4.3 Baseless Internet Mode

The diagram in Figure 4 uses the PPP-RTK technique with the PointPerfect correction service from u-blox. This service has been chosen because it is provided by the manufacturer of the GNSS module used.

This architecture is designed for systems that can only rely on the rover device. As in the first scheme, Figure 2, access to the MQTT server is done through the Internet and the cellular network. Specific software was developed, detailed in Subsection 5.4 to connect the rover device to the PointPerfect service.

5 Software Development

This section describes the software components developed to integrate the positioning operation modes presented in Section 4 into the Triumph node. The explanation focuses on the key and specific aspects of this integration, discussing the libraries and software employed, while omitting technical implementation details that do not offer conceptual interest in this project, such as launching a virtual machine on AWS or log in to a third-party service.

Following the methodology outlined in Section 3, the software components were independently developed, tested with specific development boards, and finally integrated into the Triumph node. The steps followed are summarized below and detailed in the same order in the subsequent subsections:

1. Configuration of u-blox ZED-F9P GNSS modules to act as a base or rover.
2. Deployment of the MQTT server, corresponding to the MQTT Broker block shown in Figure 2.
3. Development of MQTT clients: MQTT Client Publisher and Subscriber blocks shown in Figures 2 and 4.
4. Development of LoRa transmission and reception programs: LoRa TX and LoRa RX blocks in Figure 3.
5. Development of the connector to the PointPerfect service: MQTT Client Subscriber shown in Figure 4.
6. Integration of all components into the Triumph node.

5.1 Base and Rover Configuration

The configuration of ZED-F9P devices is a critical step in implementing RTK and PPP-RTK techniques. This process requires two distinct but interrelated setups: one for the rover and another for the base station.

According to the ZED-F9P interface manual, the GNSS module offers a wide range of configuration options [2]. Key considerations for configuring the rover and base include:

- **Operating Mode**: Configure the module as *mobile* for the rover and as *fixed* for the base.
- **Output Messages**: Configuring the rover to generate NMEA messages and the base to generate RTCM messages.
- **Input Messages**: Defining the input correction message type for the rover (RTCM or SPARTN) while no input messages are required for the base.
- **GNSS Constellations**: Selecting the GNSS constellations and signals to be used. In this project, the chosen configurations include GPS (L1 and L2 bands), Galileo (E1 and E5b bands), and GLONASS (L1 and L2 bands) signals. Beidou was not used in order to reduce the required data rate for RTCM transmission.

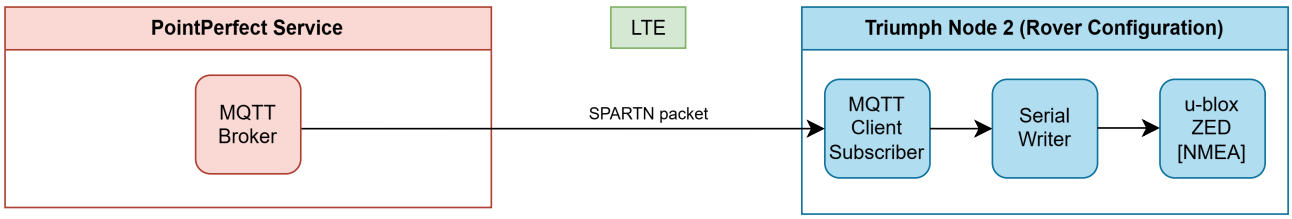

Fig. 4: Baseless Internet Mode scheme.

This decision was made because the measured position did not improve significantly when Beidou was used alongside the other GNSS constellations.

- **Base Station Position**: For an RTK base, specifying its precise global coordinates. The rover will obtain the base's position through RTCM 1005 message type.

There are numerous other configurations available for various purposes. For instance, there is an extensive list of messages dedicated to debugging GNSS devices and monitoring their state in specific aspects. However, for the specific needs of the project, NMEA messages are sufficient, as they provide essential information such as the number of visible satellites, measured position in geographic coordinates, RTK status for each GNSS constellation, and GNSS time.

Since multiple signals are utilized for each constellation, the Multiple Signal Message (MSM) format of RTCM messages is required for each GNSS system. In this project, MSM type 4 (MSM4) has been selected, as it is one of the most commonly used message types when employing the MSM format. The RTCM message IDs for multiple signals in the GPS, GLONASS, and GALILEO constellations are 1074, 1084, and 1094, respectively.

All communication with GNSS devices is conducted via serial ports. Consequently, as depicted in Figures 2, 3, and 4, the ZED-F9P modules are always accompanied by a program designed to read from or write to these ports. These programs incorporate mechanisms to ensure reliability and error handling during communication. Given that the messages exchanged over this channel adhere to a format specified by u-blox, the *pyublox2* library was employed for the development of these programs [13].

pyublox2 is a Python library designed for reading and writing messages to u-blox GNSS receivers. This software facilitated the construction and serialization of messages to be transmitted to the receiver. Additionally, it provides functionality to interpret the messages generated by the GNSS module and extract relevant information like global coordinates from NMEA messages or IDs from RTCM messages.

5.2 Dedicated MQTT Service

The first approach proposed in this project, illustrated in Figure 2, involves utilizing an MQTT service capable of retransmitting real-time data between connected devices. To provide this service, an MQTT Broker was deployed on an AWS-hosted machine accessible via the Internet. The base stations connected to this server publish their generated RTCM messages to a specific MQTT topic, and the server retransmits these messages to all connected rovers.

5.2.1 Server Deployment

For the installation of the MQTT server, the Eclipse Mosquitto broker was used. Eclipse Mosquitto is an open-source software that implements an MQTT broker developed by the Eclipse foundation [14].

In this project, real-time data transmission is a key feature. To adapt the MQTT protocol to this requirement, the broker's Quality of Service (QoS) level has been set to 0. At this level, the broker does not store messages published to topics, meaning that subscribers only receive messages published while they are actively connected, not any previously sent messages. This configuration, combined with low latency, ensures efficient real-time communication between the base station and the rover.

In other types of projects, a QoS level 0 may be considered a bad practice. QoS is primarily designed to ensure that no message is lost by storing published messages in the MQTT Broker's memory and releasing them when a rover connects to the server, transmitting all saved messages at the time of connection. However, in this project, real-time communication with minimal latency is prioritized, which justifies the choice of QoS level 0, as the RTCM messages from previous seconds are not necessary.

Additionally, to establish a minimum level of cybersecurity on the MQTT server, it has also been configured to prevent anonymous access, requiring clients to authenticate with a username and password.

5.2.2 Client Script

The MQTT clients are responsible for connecting to the broker and either publishing or downloading RTCM messages, depending on whether the device is the base or the rover. For the development of these clients, the Python library paho-mqtt has been used.

The Python library paho-mqtt provides a client-class that facilitates connecting to an MQTT broker, enabling clients to either publish messages or subscribe to topics to receive messages [15]. The library uses a typical socket-based approach to access the TCP/IP stack through the internet interface provided by the LTE module. It offers a straightforward API for managing connections, handling message publication, and managing subscriptions in real time.

In this project, the base station MQTT client is configured as a Publisher, responsible for publishing RTCM messages to specific MQTT topics. Meanwhile, the rover device client acts as a Subscriber, receiving messages from the topics it subscribes to.

It is worth mentioning that MQTT packets, which encapsulate RTCM messages in their payload, introduce a small header of 2 to 5 bytes. This minimal overhead has a negligible impact on the required data rate and energy consump-

tion. As a result, the use of the MQTT protocol proves to be an effective solution for IoT applications, such as those employed in this project, where efficient data transmission and minimal energy usage are crucial.

5.3 LoRa Transmission and Reception

For environments without access to cellular networks, or in situations where the cost of contracting a service provider is high, or when there is a desire for the technology to be independent of third parties, the architecture with a direct communication channel based on LoRa modulation is ideal.

The developed LoRa TX and LoRa RX programs in the diagram of Figure 3 are responsible for establishing the communication channel. They access the LoRa RAK5146 module through the SPI port to receive and write RTCM messages in LoRa packets.

To develop these programs, the *lora_gateway* library has been used [16]. This library is an implementation of a gateway for the LoraWAN protocol made by Semtech for their hardware devices. Although this project does not utilize the full implementation of a gateway, the library provides a highly useful hardware abstraction layer for implementing transmitters and receivers for LoRa modulation.

LoRa is a modulation technique with many configurable parameters, such as transmission frequency, bandwidth, coding rate, and spreading factor. All of these parameters, except for the transmission frequency, have a direct impact on the data rate achievable with LoRa modulation. It has been ensured that both the transmitter and receiver are configured identically; otherwise, the receiver will not be able to demodulate the transmitted signals. Also, since the devices do not use the LoRaWAN network and their purpose is for testing and demonstration, all the parameters have been chosen in the most convenient way for implementation.

In this project, a frequency of 868.5 MHz, a bandwidth of 125 kHz, a coding rate of 1, and a spreading factor (SF) of 7 are used. With this configuration, a maximum data rate of 6.8 kbps (850 Bytes/s) is achieved. In Section 6.1, it is demonstrated that this data rate is sufficient to transmit all RTCM messages from each GNSS constellation within each second.

5.4 PointPerfect Service Connector

For the final architecture, presented in Figure 4, where a baseless system is used, only the rover device needs to be configured.

First, the rover has been set up to accept SPARTN messages as input packets to the GNSS module. Since the u-blox ZED-F9P modules are compatible with these types of messages, integrating the rover with the PointPerfect service is straightforward [7].

This service is based on an MQTT server with several topics. The corrections transmitted by the service are encrypted, and the GNSS module is responsible for decrypting them using the keys transmitted through the KEY topic. These keys remain valid for thirty days, after which they are replaced with a new key. The data is transmitted in the SPARTN format, a format created by several GNSS companies, including u-blox, Septentrio, and Geo++.

There are two main types of topics: continental and localized corrections [17]. The continental topics transmit valid corrections for an entire continent, such as Europe. In contrast, the localized topics transmit only those corrections that are valid for a specific region of 250 x 250 km. The localized approach reduces the data rate required for continental corrections. The PointPerfect connector software that was developed is compatible with both types of corrections.

To connect to the broker, an MQTT client is also used, based on the Python paho-mqtt library, with the distinction that authentication is performed using a digital certificate.

5.5 Integration into Triumph Node

Lastly, the most important task was the integration of all software into the Triumph node.

Before starting the integration, it was necessary to install all the libraries mentioned earlier onto the device. Since all the software was develop using Python, except the LoRa-related programs which were written in C, a connector needed to be created between the Python program that reads the RTCM messages from the ZED-F9P serial port and the C program that transmits messages using LoRa. In Figure 3, this program would be between the Serial Reader and the LoRa TX blocks.

There are many ways to solve this problem. In this project we used an internal Internet socket to transport data between the programs. Writing a program in C to read the RTCM messages was unnecessary. The structure of RTCM messages can vary depending on the message type and other parameters, and reading a full RTCM message would require parsing the packets. The pyublox2 library already handles this part of the work.

6 EXPERIMENTATION

This section presents a series of results obtained from the developed prototype to demonstrate the correct operation of the system and compare the performance of the RTK and PPP-RTK techniques.

To obtain these measurements, the architecture depicted in Figure 2 was used for RTK and the architecture in 4 for PPP-RTK. In case of RTK, the communication channel used does not affect the performance of the technique, as both communication technologies allow for a delay of less than one second and provide sufficient bandwidth to transmit the data.

Figures of the real setup are provided in Section A.1 of the Annex. Figure 9 illustrates the Triumph board equipped with all components required for outdoor placement. Figure 10 presents the PyGPSClient software user interface displaying GNSS data. Figure 11 depicts an outdoor test conducted using two ArduSimple modules, while Figure 12 demonstrates the complexity of a complete testing setup.

6.1 System Validation

In this subsection, results obtained with the Triumph node are presented to validate the correct operation of the developed software.

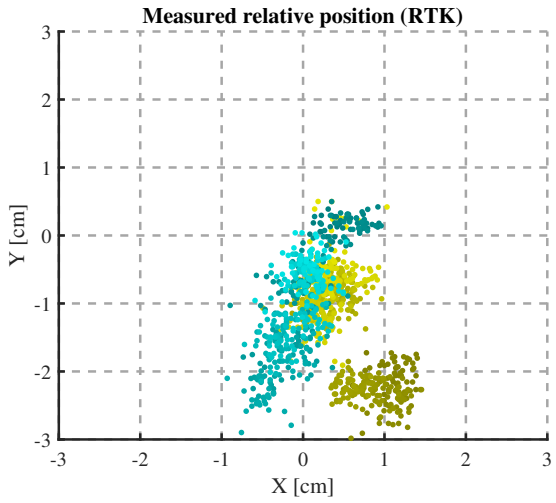

Fig. 5: Relative position measured with RTK from two different iterations. Each color represents a different iteration, and the brighter the point, the latest the measurement has been taken.

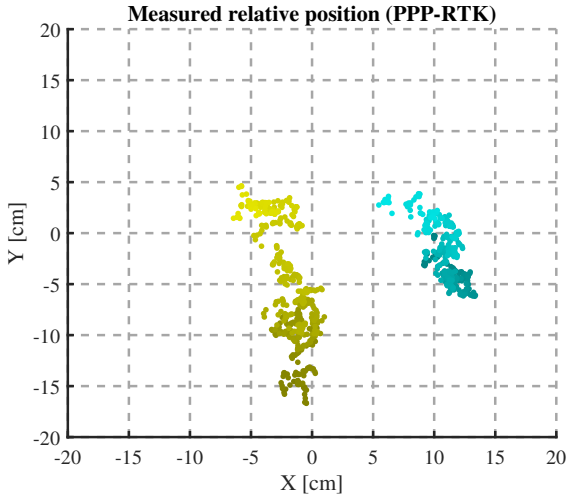

Fig. 6: Relative position measured with PPP-RTK from two different iterations. Each color represents a different iteration, and the brighter the point, the latest the measurement has been taken.

Figures 5 and 6 show two maps of relative position points to a reference measured by the developed prototype. These data are obtained from the NMEA messages provided by the GNSS receiver through a serial port. The NMEA messages contain information gathered by the receiver. Some of the fields in these messages include GNSS time, the number of satellites seen by the receiver, or the position in geographic coordinates (latitude and longitude). To display the values in Cartesian coordinates, the geographic coordinates were projected onto the Web Mercator system.

As observed, the accuracy of RTK surpasses that of PPP-RTK. This observation aligns with the values presented in Table 1. Specifically, the points measured using RTK remain within 3 cm of the reference, whereas the values obtained with PPP-RTK deviate by up to 5 cm, and in some instances, by as much as 15 cm from the reference.

In Table 2, the measured data rate required for the communication channel to transmit RTCM messages in real-time for the RTK technique is shown. It can be observed that the total data rate is lower than what could be provided by any LTE network or the LoRa signals with the previously mentioned configuration. It is also noted that all constellations require a similar number of bytes. However, the messages containing only the position of the base (necessary to perform RTK) are smaller.

RTK supports systems with mobile base stations; for this reason, an RTCM message is used to transmit the base position. In this project, only static bases have been used, so it would be possible to avoid continuously transmitting this message over the network. It would only be necessary the first time.

To test LoRa connectivity, multiple outdoor tests were performed. The base and rover were placed at two Cartographic and Geological Institute of Catalonia (ICGC) geodesic points, with a distance of 600 m between them and without a Line of Sight (LOS). In these tests, RTK fixed was achieved with an accuracy of under two centimeters relative to the location provided by the ICGC .

6.2 Performance Evaluation

In this section, the potential of this system is demonstrated using measurements obtained from the developed project. The objective is to demonstrate the system's performance using the results provided by the two techniques (RTK and PPP-RTK) in real-world environments and conditions. In Figure 7, the Empirical Cumulative Distribution Function (ECDF) of the distances relative to the reference point for both RTK and PPP-RTK techniques is shown. It can be observed that more than 50% of the RTK samples are within one centimeter of the reference. In contrast, 50% of the PPP-RTK samples are within 6 cm. Additionally, the PPP-RTK samples show greater dispersion compared to RTK. While 90% of the RTK samples are below 2 cm, 90% of the PPP-RTK samples are below 13 cm.

A second crucial metric for comparing advanced positioning techniques is the *Time to Fix*. In the GNSS ecosystem, there are multiple types of *fix*; however, this work focuses exclusively on the *RTK fixed* state. This state is achieved when the RTK or PPP-RTK algorithm resolves integer phase ambiguities, significantly enhancing accuracy. The *Time to Fix* in this work is measured from the moment the GNSS module receives the first correction until it attains the *RTK fixed* state. In each experiment, the GNSS module had already acquired and tracked the necessary GNSS signals, which does not require any exchange of information through the communication infrastructure.

In Figure 8, the ECDF functions of both techniques are

TABLE 2: AVERAGE MEASURED TRANSMISSION DATA RATE FROM BASE STATION TO ROVER

RTCM Message	Data Rate [Bytes/s]
GPS	158.9
Galileo	132.7
GLONASS	146.0
Base Position	25.0
Total	462.6

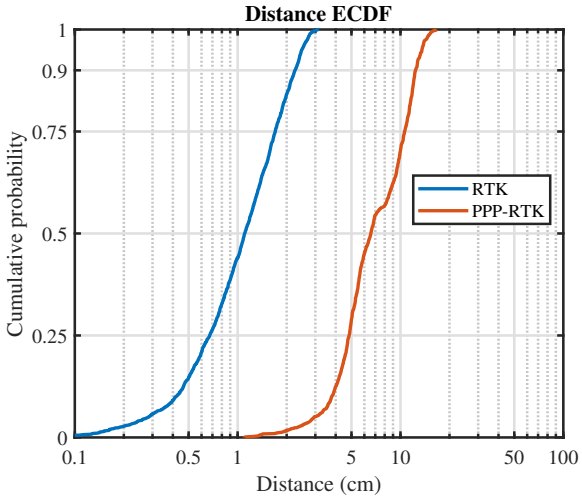

Fig. 7: ECDF of the distances between the sampled points in the *RTK fixed* state (i.e., the convergence of the algorithm to its best accuracy) and the reference point.

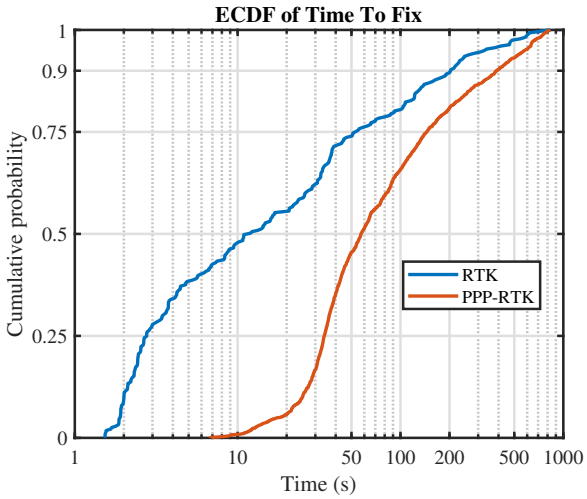

Fig. 8: ECDF of the time required to achieve the *RTK fixed* state from the moment of receiving the first correction.

compared once more, this time for the *Time To Fix*. It can be observed that RTK is also superior to PPP-RTK in this regard. In 50% of cases, RTK achieves a *Fix* in less than 10 seconds. In contrast, PPP-RTK takes up to one minute. By referring to Table 1, it can be seen that the values obtained are close to the state of the art of both techniques. However, one of the factors that worsen the convergence time of PPP-RTK has also been identified.

In PPP-RTK, the service PointPerfect periodically transmits state representation corrections and does not take into account the rover's connection time. For this reason, the rover may wait up to 30 seconds for the first data packet [17]. If the service were to recognize the moment a user connects and, without any waiting, transmit the correction packet, it could significantly improve the time to the *RTK fixed* state.

7 Conclusions

This paper presents the design and prototyping of a precise positioning system adaptable to various environments and conditions. The development is based on RTK and PPP-RTK techniques and addresses the communication requirements using different technologies and architectures.

The prototype is built around hardware specifically designed to enable high-precision GNSS applications, featuring PCIe modules to support all functionalities employed, and it operates on a Linux-based operating system. The implemented system relies on three different architectures to tackle challenges posed by various scenarios.

The first architecture is based on an RTK system with a dedicated base station. The rover device accesses RTCM messages from the base through an intermediary MQTT server. This server runs on an AWS machine accessible via the Internet.

Secondly, for environments where Internet access is not feasible, the project introduces a second architecture. This replaces the MQTT broker with a direct channel between the base and the rover using the LoRa modulation. However, this architecture has certain limitations, mainly available bandwidth on the channel.

Thirdly, for scenarios where the deployment of a base station wants to be avoided, the use of the u-blox PointPerfect SSR service accessible via the Internet is proposed. As a trade-off for not having a dedicated base station, the system experiences a reduction in accuracy and a delay in obtaining reliable measurements. This document also compares the loss of accuracy and the added latency between RTK and PPP-RTK techniques.

Since integrating functionalities into complex hardware devices can result in numerous unforeseen challenges, this work adopted a rapid prototyping methodology. Specifically, development platforms such as Raspberry Pi and ArduSimple have been used for software development and functional field-testing to validate effectiveness. Subsequently, the system was fully integrated into the target hardware.

This work, along with the developed software and utilized hardware, serves as a foundational functional system for high-precision positioning applications.

8 Future Work

One of the main objectives of this work was to establish a foundation for future projects focused on precise positioning in embedded devices. Consequently, the proposed solution was designed to be highly extensible and adaptable, allowing for customization to accommodate a wider variety of scenarios or to achieve general improvements.

A first proposal consists of adapting the system based on an MQTT server to support multiple base stations. The MQTT protocol, designed for IoT applications, significantly facilitates communication between numerous devices simultaneously. Within this architecture, a handover mechanism would need to be developed to select the nearest base station from all the topics managed by the MQTT broker. By deploying additional base stations, the system could significantly enhance its coverage while maintaining accuracy, as long as the selected base station remains sufficiently close.

A second proposal is to use the LoRaWAN network instead of only using LoRa modulation. LoRaWAN is a

MAC-layer communication protocol based on LoRa technology. This network allows the interconnection of low-power IoT devices, offering an efficient solution for this specific use case. LoRaWAN operates within unlicensed ISM bands (865-868 MHz), which are regulated by the European Telecommunications Standards Institute (ETSI) guidelines. Among the applicable restrictions, the most notable is the duty cycle limitation of 1%, which corresponds to a maximum of 36 seconds per hour dedicated to transmission. This limitation could directly affect RTK functionality, which requires continuous and low-latency data transmission. For this reason, a feasibility study should be conducted to evaluate the viability of this system, specifically the LoRa-based architecture shown in Figure 3, within a LoRaWAN network. In this approach, the base station would act as a LoRaWAN gateway, providing RTCM corrections via the downlink to all rovers within its coverage area. Additionally, to improve energy efficiency, the base station and rover devices should be programmed to activate their GNSS receiver and other components only when the connection between both sites is established, matching the duty cycle regulation.

To address the time-to-fix issue discussed in Section 6.2, it is proposed to utilize the PointPerfect service delivered via NTRIP instead of MQTT. NTRIP accounts for user connectivity, potentially reducing the time-to-fix by up to 30 seconds.

Lastly, it is suggested to evaluate alternative SSR services to assess whether they provide improvements over the PointPerfect service employed in this project.

ACKNOWLEDGMENTS

I would like to thank the Wireless Networks Research Lab at Universitat Oberta de Catalunya for providing the necessary equipment and the opportunity to carry out this project as part of their research. Special thanks to Borja Martínez Huerta, Xavier Vilajosana Guillén and Guillem Boquet Pujadas for their invaluable support in introducing me to the field of IoT development, wireless connectivity, and for helping me grow as an engineer.

REFERENCES

[1] u-blox. “Zed-f9p module.” (2023), [Online]. Available: https://www.u-blox.com/en/product/zed-f9p-module.

[2] u-blox. “U-blox f9 hpg 1.50 interface description.” (2025), [Online]. Available: https://content.u-blox.com/sites/default/files/documents/u-blox-F9-HPG-1.50_InterfaceDescription_UBXDOC-963802114-12815.pdf.

[3] SEMTECH. “Lora gateway baseband processor.” (2020), [Online]. Available: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000Hlli/Te0cB6.fNWAPfxRfoFz38R6LOTf3sLAJhD4CpS2RwFc.

[4] Quectel. “Quectel eg25-g.” (2025), [Online]. Available: https://docs.rs-online.com/266f/A700000007115507.pdf.

[5] P. J. Teunissen and O. Montenbruck, *Springer handbook of global navigation satellite systems*. Springer, 2017, vol. 10.

[6] S. A. Yusuf, A. Khan, and R. Souissi, “Vehicle-to-everything (V2X) in the autonomous vehicles domain–A technical review of communication, sensor, and AI technologies for road user safety,” *Transportation Research Interdisciplinary Perspectives*, vol. 23, p. 100 980, 2024.

[7] u-blox. “PointPerfect — u-blox.com.” (2025), [Online]. Available: https://www.u-blox.com/en/product/pointperfect.

[8] “Spartn: Secure position augmentation for real time navigation.” (2025), [Online]. Available: https://www.spartnformat.org/.

[9] B. Furht and S. A. Ahson, *Long Term Evolution: 3GPP LTE radio and cellular technology*. Crc Press, 2016.

[10] Q. Zhou, K. Zheng, L. Hou, J. Xing, and R. Xu, “Design and implementation of open lora for iot,” *IEEE Access*, vol. 7, pp. 100 649–100 657, 2019. DOI: 10.1109/ACCESS.2019.2930243.

[11] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low power wide area networks: An overview,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017. DOI: 10.1109/COMST.2017.2652320.

[12] E. Foundation. “Mqtt: The standard for iot messaging.” (2025), [Online]. Available: https://mqtt.org/.

[13] semuconsulting. “Pyubx2: A python library for ubx protocol parsing and generation.” (2025), [Online]. Available: https://github.com/semuconsulting/pyubx2.

[14] E. Foundation. “Mosquitto: An open source mqtt broker.” (2025), [Online]. Available: https://mosquitto.org/.

[15] E. Foundation. “Paho-mqtt: Mqtt python client library.” (2025), [Online]. Available: https://pypi.org/project/paho-mqtt/.

[16] Semtech. “GitHub - Lora-net/lora_gateway: Driver/HAL to build a gateway using a concentrator board based on Semtech SX1301 multi-channel modem and SX1257/SX1255 RF transceivers. — github.com.” (2025), [Online]. Available: https://github.com/Lora-net/lora_gateway/tree/master.

[17] u-blox. “Pointperfect service description.” (2025), [Online]. Available: https://developer.thingstream.io/guides/location-services/pointperfect-service-description.

ANNEX

A.1 Additional Figures

This section includes some figures obtained during the project. Figure 9 illustrates a fully equipped Triumph node ready for outdoor deployment. Figure 10 depicts a typical system status using the PyGPSClient software, showing the satellites in view, the coordinates, and other relevant parameters. Furthermore, Figure 11 demonstrates an outdoor RTK system test using two Ardusimple devices along with their corresponding antennas. Finally, Figure 12 showcases an outdoor setup for testing an RTK system with LoRa as the communication channel. In this configuration, a Triumph board operates as the rover, while a Raspberry Pi with a LoRa HAT, connected to an Ardusimple, acts as the base station. The figure also highlights all external components required for the test, such as an Ethernet switch, a portable battery, and a power strip to power the devices.

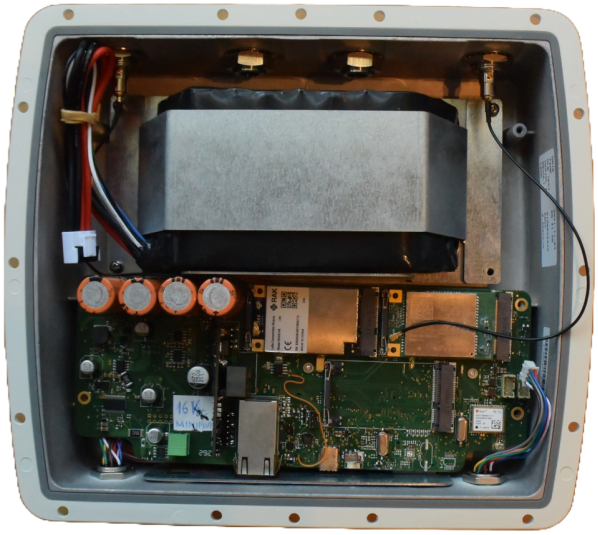

Fig. 9: Triumph node with all the equipment to be placed outdoors, including a battery and protection box.

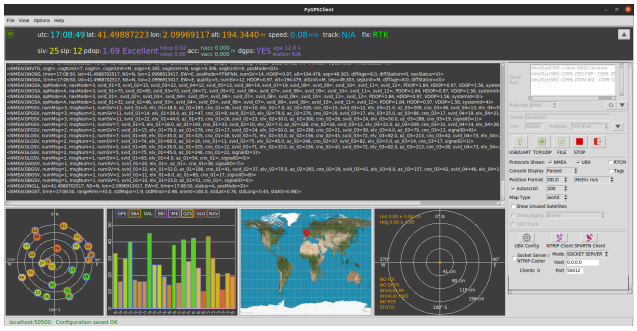

Fig. 10: PyGPSClient software UI displaying a typical GNSS scenario.

Fig. 11: Outdoor tests with two Ardusimple modules using an ICGC geodesic reference point.

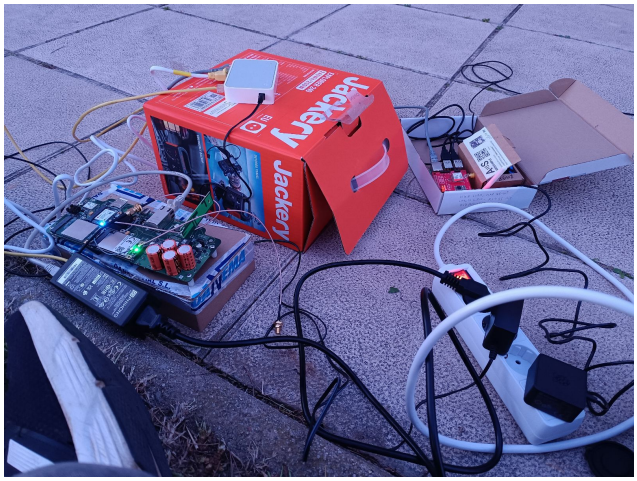

Fig. 12: Outdoor tests with the Triumph node as the rover and a Raspberry Pi connected to an Ardusimple module as the base station.