INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Remote GNSS Server Station

José Carlos Machado Libório

Master's degree in Telecommunications and Computer Engineering

Supervisor:
PhD Francisco António Bucho Cercas, Full Professor,
Iscte-IUL

September, 2024

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

Remote GNSS Server Station

José Carlos Machado Libório

Master's degree in Telecommunications and Computer Engineering

Supervisor:
PhD Francisco António Bucho Cercas, Full Professor,
Iscte-IUL

September, 2024

Acknowledgment

I would like to extend my heartfelt appreciation and profound gratitude to my supervisor, Professor Francisco António Bucho Cercas, for his unwavering support and guidance throughout this research journey. His expertise and insightful feedback have been invaluable, and his encouragement has been a constant source of motivation.

I am also deeply grateful to ISCTE - Instituto Universitário de Lisboa for providing an exceptional environment, equipped with the best resources and facilities, which has greatly contributed to my academic and personal growth.

Lastly, I dedicate this thesis to my family, whose unwavering support and love have been the foundation of my academic achievements. Their encouragement and belief in me have been instrumental at every step of this journey. I am forever indebted to them for providing the tools and inspiration necessary for my success

Resumo

O Sistema Global de Navegação por Satélite (GNSS) é um termo abrangente que engloba vários sistemas de navegação por satélite que fornecem posicionamento geoespacial com cobertura global. Esta tese apresenta o desenvolvimento de uma plataforma de monitorização remota de GNSS utilizando o recetor GNSS U-Blox EVK-M8T conectado a um Raspberry Pi. O Raspberry Pi, equipado com conectividade à internet, garante acesso remoto para utilizadores autorizados, ao mesmo tempo que fornece armazenamento local para monitorização e análise de dados previamente recebidos.

O software para esta plataforma foi construído de raiz, com especial atenção aos protocolos de recuperação e análise de dados. Isto garante que os dados são exibidos com precisão ao utilizador, seja através de ficheiros de texto ou de uma interface que acede à base de dados. A plataforma fornece uma base sólida para escalabilidade e adaptabilidade caso haja necessidade de adicionar mais funcionalidades no futuro.

A motivação por detrás deste desenvolvimento é fornecer um meio fiável e acessível de monitorização e análise de dados GNSS estacionado no campus do ISCTE, que pode ser aplicado em várias áreas, como posicionamento preciso, sincronização de tempo e monitorização de velocidade. Desafios significativos, como distinguir mensagens UBX de outros dados e garantir a análise precisa dos dados, foram abordados através de um estudo meticuloso e implementação do protocolo UBX. Trabalhos futuros podem envolver a expansão das capacidades da plataforma para lidar com tipos adicionais de mensagens GNSS e a integração de ferramentas de visualização de dados mais avançadas.

Palavras-chave: GNSS; Comunicação por Satélite; Análise de Dados; Monitorização Remota; U-Blox EVK-M8T; Raspberry Pi

Abstract

The Global Navigation Satellite System (GNSS) is an umbrella term encompassing various satellite navigation systems that provide geospatial positioning with global coverage. This thesis presents the development of a remote GNSS monitoring platform utilizing the U-Blox EVK-M8T GNSS receiver connected to a Raspberry Pi. The Raspberry Pi, equipped with internet connectivity, ensures remote access for authorized users while also providing local storage for monitoring and analyzing previously received data.

The software for this platform was built from scratch, with particular attention to data retrieval and parsing protocols. This ensures that the data is accurately displayed to the user, either through text files or an interface that accesses the database. The platform provides a solid baseline for scalability and adaptability if there is a need to add more features in the future.

The motivation behind this development is to provide a reliable and accessible means of GNSS data monitoring and analysis stationed at the ISCTE campus, which can be applied in various fields such as precise positioning, time synchronization, and velocity tracking. Significant challenges, such as distinguishing UBX messages from other data and ensuring accurate data parsing, were addressed through meticulous study and implementation of the UBX protocol. Future work could involve expanding the platform's capabilities to handle additional GNSS message types and integrating more advanced data visualization tools.

Keywords: GNSS; Satellite Communication; Data Parsing; Remote Monitoring; U-Blox EVK-M8T; Raspberry Pi

Contents

List of Figures

Glossary

3D: Three Dimensional. 5

ACID: Atomicity, Consistency, Isolation, Durability. 17
API: Application Programming Interface. 7
ASCII: American Standard Code for Information Interchange. 11

BDS: BeiDou Navigation Satellite System. 6

DDC: Direct Digital Control. 19

ECEF: Earth-centred-Earth-fixed. 16

FIFO: First-In-First-Out. 17

GEO: Geostacionary Earth Orbit. 6
GLONASS: Global'naya Navigazionnaya Sputnikovaya Sistema, or Global Navigation Satellite System. 5, 6
GND: Ground. 19
GNSS: Global Navigation Satellite System. ix, 1–9, 11, 18, 19, 27
GPS: Global Positioning System. 1, 5, 6, 11, 13

HTTP: Hypertext Transfer Protocol. 9

IGSO: Inclined Geo-Synchronous Orbit. 6
IoT: Internet of Things. 8
IRNSS: Indian Regional Navigation Satellite System. 5, 6
ISCTE: Instituto Superior de Ciências do Trabalho e da Empresa. 2, 3, 18

JAXA: Japan Aerospace Exploration Agency. 6

LLH: latitude, longitude, and height. 7, 16

MEO: Medium Earth Orbit. 6

NAVSTAR: NAVigation Satellite Timing And Ranging. 5
NEMA: National Marine Electronics Association. 11, 13, 27

PHP: PHP: Hypertext Preprocessor. 19

PNT: positioning, navigation and timing. 1

QoS: Quality of Service. 8
QZSS: Quasi-Zenith Satellite System (East Asia (Japan), Oceania). 6

RTK: Real-Time Kinematic. 1, 5

SV: satellite/space vehicle. 6

US: United States. 5
USA: United States of America. 5
USB: Universal Serial Bus. 9, 19

VPN: Virtual Private Network. 3, 18, 22

CHAPTER 1

Introduction

The technology associated with GNSS is widely available nowadays due to the vast number of artificial satellite constellations providing coverage almost worldwide. The main goal of GNSS is to provide positioning, navigation and timing (PNT) to GNSS receivers, whether they are for personal use, such as Global Positioning System (GPS) applications on our phones, or for public transportation and security.

This chapter will discuss, in the next section, what this thesis is about, providing context and background on the various applications GNSS offers. It will also cover the research questions, and goals that will aid in understanding the following chapters.

1.1. Context & Background

Navigation systems have evolved a great deal since hundreds of years ago when one of the most important instruments to indicate direction was the magnetic compass. In fact, it still is one of the most important, but nowadays, with the help of technologies like GPS, it is much easier for us to navigate. For a system like GPS to work, it needs a receiver, present in many consumer products like mobile phones, smartwatches, laptops, personal navigation devices, etc., to obtain navigation parameters (position, speed, and time) that are transmitted by satellite signals.

There are many applications for GNSS that go beyond personal use for navigation:

- Transportation (aerial, maritime and road);
- Agriculture that have the need to manage vast crop fields, yield monitoring, precise weed management systems and Real-Time Kinematic (RTK) GPS based plant mapping [**6**], and fisheries for dispatch and monitor fishing boat, acquire and analyze fishing record and safety [**7**];
- Emergency warning services that help find the localization where an accident took place and deploy the rescue teams as soon as possible;
- Surveying is also a very important application connected with GNSS technologies, which helps with mapping, environment changes monitoring, cartography, cadastral survey, hydrography, natural resources, geodesy and marine seismic exploration [**1**].
- Timing with the highest precision is possible due to each satellite containing multiple atomic clocks that contribute very precise time data to the GPS signals. "Precise time is crucial to a variety of economic activities around the world", like communication systems, electrical power grids and financial networks [**8**].

Since this thesis is about the implementation of a remote GNSS server station, other areas such as networking, computer architecture, programming, and web development will also be involved. Familiarization with the GNSS receiver (U-Blox EVK-M8T), the messages received, and the protocols being used for this project will be key factors for the success of this implementation.

1.2. Research Questions

In the development of a functional and permanent GNSS station with remote access, several research questions arise that guide the direction and focus of this thesis. The following research questions are central to this study:

- What are the applications that use GNSS and what are their advantages?
- How will this implementation differ from the already existing software?
- With this being a permanent GNSS station at Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), how can it be of use?
- What are the challenges associated with the retrieval and parsing of GNSS data, and how can they be addressed?
- How can the security of the GNSS station be ensured, particularly with remote access and data storage?
- What are the potential future enhancements that could be made to this GNSS station to improve its functionality and adaptability?
- How can the data collected from this GNSS station contribute to ongoing research and development in the field of satellite communications?

These questions aim to explore the applications, advantages, and unique contributions, as well as its potential and utility.

1.3. Goals

The goal of this project is to implement a functional and permanent GNSS station with remote access. For this, the GNSS receiver U-Blox EVK-M8T (Figure 1.1) will be connected to an antenna on the roof of ISCTE and to a Raspberry Pi computer, as this is a low-power platform that will be hosting a simple server.

FIGURE 1.1. An image of the EVK-M8T evaluation kit.

This server will handle the retrieval of the different messages provided by the receiver and store them in a local database. Filtering these messages will be vital for this implementation to ensure the functionality of this GNSS station, keeping only the most important ones. To ensure that all functionalities are working, testing will be done both locally and remotely to ensure the success of the implementation. Access will be made through a webpage which, for security reasons, will present a login screen and, for remote access, through a Virtual Private Network (VPN) connected to the ISCTE network to provide the necessary security measures for this server station.

The following diagram illustrates the architecture of the platform:

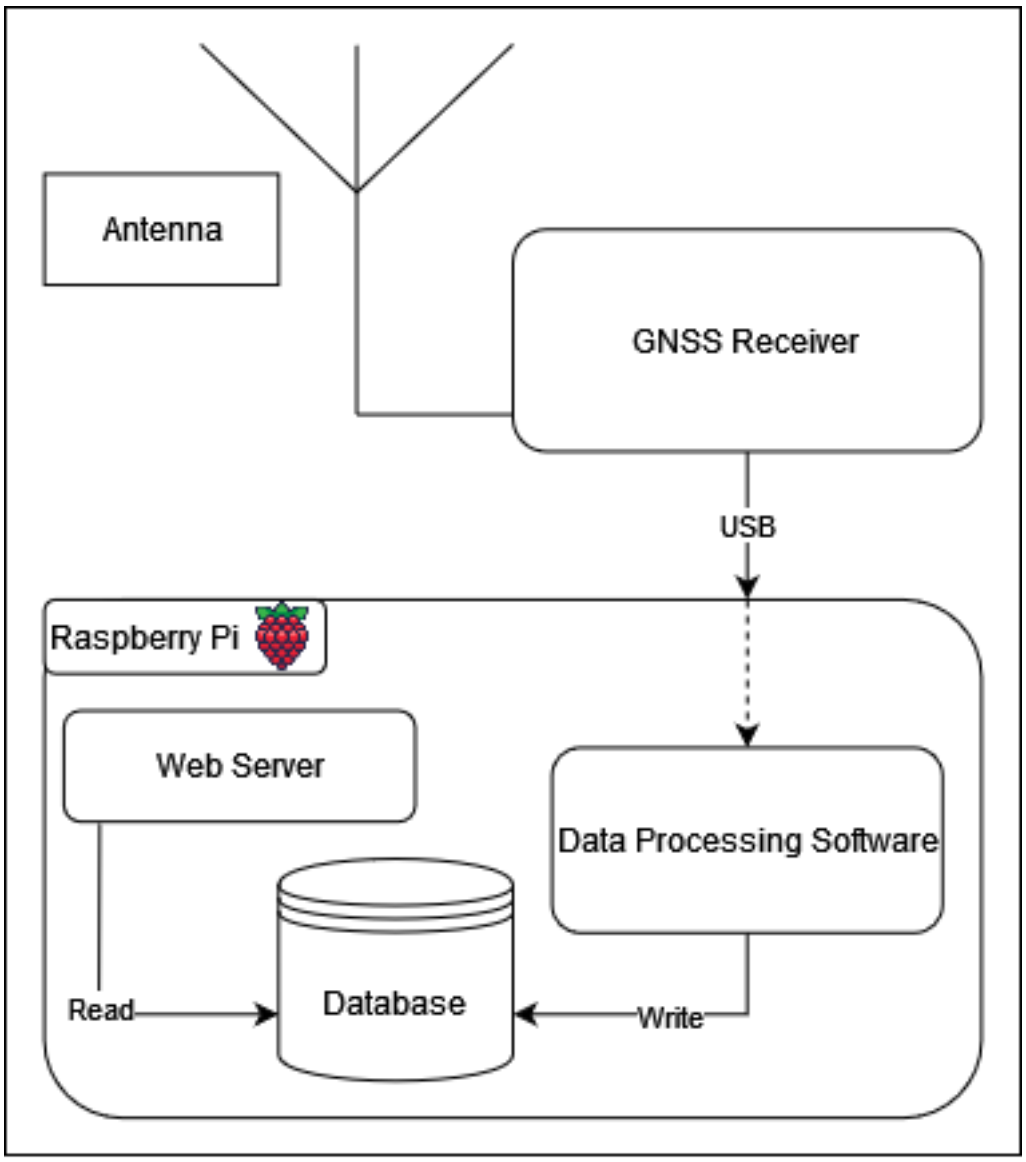

FIGURE 1.2. A superficial view of the platforms architecture diagram.

1.4. Contributions

This thesis makes several contributions to the field of GNSS satellite communications. Firstly, we provide a comprehensive state-of-the-art review of GNSS satellite communications and their applications (Chapter 2). This review serves as a foundational understanding for the subsequent development and implementation phases. Secondly, we conduct an in-depth study of the protocols used in satellite communications, focusing on the extraction and processing of GNSS data (Chapter 3). Thirdly, we develop a robust platform capable of receiving, parsing, and extracting GNSS messages. This platform includes a web interface that allows for the interpretation and visualization of GNSS data. The platform's design ensures efficient data handling and provides a secure means of remote access, thereby contributing to the practical implementation of a functional and permanent GNSS station with remote access capabilities (Chapter 4).

CHAPTER 2

Literature Review

GNSS refers to a network of satellites that provide location and time information to users globally. The most well-known GNSS system is the GPS developed by the United States (US) government. Currently, there are several GNSS systems in operation, including GPS, Global'naya Navigazionnaya Sputnikovaya Sistema, or Global Navigation Satellite System (GLONASS), BeiDou, Galileo, and Indian Regional Navigation Satellite System (IRNSS). These systems have improved significantly in terms of accuracy, availability, and coverage, with multi-constellation support and augmentations like RTK positioning. New GNSS technologies and services have been developed to meet the increasing demand for high-precision positioning, particularly in areas such as autonomous vehicles, drone navigation, and precision agriculture. Overall, GNSS technology continues to evolve and improve, with the aim of providing users with more accurate and reliable positioning information.

2.1. Background Concepts

This section will provide an introduction to GNSS, its constellations, the characteristics of its associated services, positioning, and other applications.

GNSS stands for Global Navigation Satellite System, and it's often referred to as an umbrella term because it includes all global satellite positioning systems. These systems can be global, providing global coverage, or regional, providing coverage in certain regions. Each system is a constellation of satellites that orbit at a distance of about 20,000 km, while the Earth's radius is about 6,000 km. For reference, geostationary orbit is about 35,785 km.

The first system that comes to mind is GPS, also known as NAVigation Satellite Timing And Ranging (NAVSTAR) GPS, which is a system owned by the United States of America (USA) government. It supports global coverage, radio signals positioning, synchronization system, navigation system, Three Dimensional (3D) positioning, and unique coordinates system [**1**]. This constellation consists of 31 satellites, of which 24 are operable 95% of the time. These 24 satellites are composed of 6 orbit planes with 4 satellites per orbit. "This 24-slot arrangement ensures users can view at least four satellites from virtually any point on the planet," quoted from the space segment of the Official U.S. government information about the Global Positioning System (GPS) and related topics [**9**]. While this last part was related to the space segment, there is also the control segment and user segment. The control segment consists of tracking, monitoring, and communicating with the constellation [**10**]. The user segment covers all the applications already

mentioned in the previous chapter, such as agriculture, safety, environment, surveying, timing, etc.

Galileo is another GNSS system launched by the European Global Navigation System in 2016 that provides better and improved positioning and timing information for European services. Unlike other GNSS, Galileo is the first to be specifically designed for civil purposes and is still under civilian control. It is designed to be compatible with other GNSS and interoperable with GPS and GLONASS. Since it is interoperable, it can use several constellations to increase positioning accuracy and consistency. This constellation consists of 24 satellites at about 23,000 km above the Earth. Galileo also provides services that are specific to this system, such as Open Service, High Accuracy Service, Public Regulated Service, Commercial Authentication Service, and Search and Rescue [**11**].

There are two other GNSS that have global coverage. One of them is GLONASS, a constellation that also consists of 24 satellites in orbit at 19,100 km above the Earth [**12**]. It is a system owned by the Russian Federation. The other GNSS system with global coverage is BeiDou (BeiDou Navigation Satellite System (BDS)), formerly known as Compass. BeiDou has a peculiar space segment that consists of 5 satellites located at Geostacionary Earth Orbit (GEO), 3 satellites located at Inclined Geo-Synchronous Orbit (IGSO), and 27 at Medium Earth Orbit (MEO) [**13**].

FIGURE 2.1. Reference to the GNSS with global coverage distances [**1**].

There are other systems that have regional coverage, such as IRNSS and Quasi-Zenith Satellite System (East Asia (Japan), Oceania) (QZSS), operated by Japan Aerospace Exploration Agency (JAXA).

Positioning is something that can be achieved with great precision thanks to the atomic clocks that satellites have. Unlike the commonly used quartz clocks that we use daily, atomic clocks don't lose accuracy over time (or it takes so long that it's negligible). The nominal frequency of these atomic clocks, observed from the Earth, is $f_0 = 10.23$ MHz. This high precision is crucial for accurate positioning. The satellite/space vehicle (SV) and clock rates are offset to compensate for relativistic effects. The clock rates are offset by $\Delta f/f = -4.4647 \cdot 10^{-10}$. The frequency, if hypothetically observed from the satellite, is equal to 10.22999999543 MHz. For our receivers to be able to calculate our position on the Earth, they need to receive signals from satellites. For this, two frequencies are utilized:

- L1 corresponds to the primary L-band carrier and is modulated by C/A (Coarse/Acquisition) code, P-code (precise code), and a 50 bit/second navigation message [**14**] [**15**].

$L1 = 154 \cdot f_0 = 1575.42$ MHz;

- L2 corresponds to the secondary L-band carrier and is modulated by P-code and a 50 bit/second navigation message [**14**] [**15**].

  $L2 = 120 \cdot f_0 = 1227.60$ MHz;

The concept of finding a position using satellite signals is called trilateration. First, the distance between the satellite and the receiver is calculated using the rate (speed) of travel (meters per second) [which is the speed of light $c = 299792458$ m/s] and time (seconds). The product of both will result in the distance (meters). However, getting the distance alone doesn't help since the position could be an infinite number of points on a sphere. Adding another satellite perspective provides a different set of results, which is an infinite number of points on a circumference. With a third satellite intersecting with the already existing set of solutions on the circumference, we get two possible positions for the receiver. Finally, with a fourth satellite, we can determine which of the two possible solutions is the actual position of the receiver.

2.2. Related Works

The author of this thesis has prior experience with a group project in a related area. This project involved developing an Android application that, given a fixed point P on Earth, P (latitude, longitude, and height (LLH)), and a specific date and time when the sun is visible, could calculate the exact length of the shadow projected by an object, building, etc.

The app has a simple interface (Figure 2.2), and the inputs are easy to understand. Nonetheless, there is an instruction screen that can be accessed by pressing the bottom right button. It uses the Google Maps Application Programming Interface (API) for easy understanding of the registered position. This app does not require an internet connection since it only uses the GNSS receiver built into the device. Once the permissions for the app to use the "Location" feature are given, it can automatically mark point P on the map, retrieving the latitude and longitude automatically. Between getting the coordinates given by the GNSS receiver and the date and time, calculations include geographic coordinates (LLH) conversion to Cartesian coordinates (XYZ), azimuth, elevation, and satellite coordinates using Ephemeris.

The following image illustrates the interface of the Shadow Lenght Calculator:

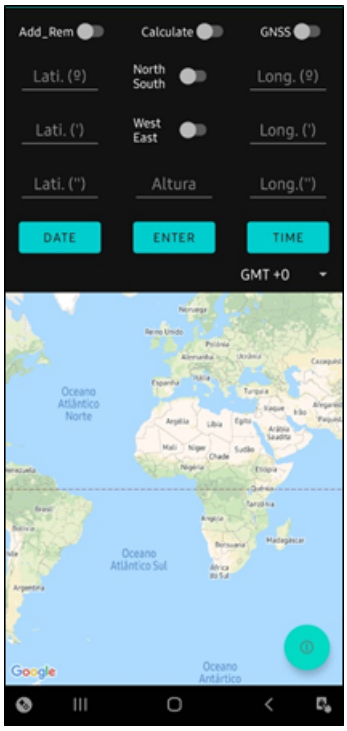

FIGURE 2.2. Shadow Length Calculator interface.

As illustrated in Figure 2.2, the app's interface is designed for simplicity and ease of use.

Another study, "Distributed GNSS-based Time Synchronization and Applications" [**16**] states that "in various applications, it is necessary to ensure a reliable and validated source(s) of synchronism." The article discusses the necessity of accurate synchronization for modern and future telecommunications applications. As mentioned before, GNSS systems have very accurate atomic clocks, and because several constellations provide worldwide coverage with enough visible satellites, "it is possible to synchronize to these atomic clocks wirelessly from practically any point on the surface of the Earth." Although there are some drawbacks, according to the authors, such as "a receiving antenna requires free line-of-sight to the satellites, complicating installation" or simply the unavailability of the GNSS system due to outage or extraordinary military activities. The authors describe several applications where synchronization is key:

- Power lines and Grids: time and frequency deviation control, multi rate billing, time tagging in measurement and fault detection systems, etc;
- Telecommunications: "time and frequency references to mobile base station and correct generation of the signals on the radio interface and handover procedures", "Quality of Service (QoS) over transport networks", etc;
- Automation of distributed systems: stocks, transportation, cloud computing, etc;
- Internet of Things (IoT) in consumer and industrial applications;
- Precise positioning;

- Legal time: "services that require legally confirmed time", like public tenders, auctions, etc.

Another work, developed by a group of former students from the master's degree in Telecommunications and Computer Engineering, in the subject of Digital Satellite Communications Systems, from 2016, is the "GNSS Tracker" [**17**]. This project involved the development of a web server for a GNSS station on a low-power consumption platform, similar to the one proposed in this thesis. The project features a modular architecture that enhances scalability and adaptability.

The platform consists of a Raspberry Pi running the data acquisition and treatment module, which communicates with an ISCTE server via Hypertext Transfer Protocol (HTTP), sending data to be stored in a database connected to the main website (a public web server hosted at ISCTE). In addition to running the data acquisition and treatment module, the Raspberry Pi hosts a "Data Website," an internal web server connected to the file storage, which also originates from the data acquisition and treatment module. This module also connects to a client, enabling remote monitoring of the received messages through a graphical interface, accessible within the network or via VPN. The Raspberry Pi receives data through Universal Serial Bus (USB) from a GNSS receiver.

Due to the similarities with GNSS Tracker, this project provided valuable insights and a solid foundation for the development of the platform proposed in this thesis.

CHAPTER 3

Development

3.1. Overview

The GNSS satellites send a set of messages that can be acquired by specific receivers. The U-Blox M8 is a high-precision receiver that can acquire these messages and estimate the position of the user. In this case, the receiver will only be used as an interface to obtain the messages, and its processing will be customized.

A platform will be developed for the autonomous analysis of messages from the receiver. In this context, a platform refers to a combination of hardware and software components designed to perform specific tasks or functions. Users can access a web interface to view information on the current position of the receiver and a history of messages. Key features of the web interface include periodic position updates, message history, data visualization, user authentication, and message type filtering. The platform will run on an always-on, low-power device, specifically a Raspberry Pi. The backend of the platform will include components such as 'Main', 'UBXParser', 'NMEAParser', 'UBXMessageExtractor', and 'DatabaseHandler', which handle the main logic, UBX message parsing, NMEA message parsing, extracting messages to text files, and database interactions, respectively.

3.1.1. Protocols

GPS receivers get messages with information that can come in binary or American Standard Code for Information Interchange (ASCII) characters. The received messages are defined by two main protocols: National Marine Electronics Association (NEMA) and UBX. NEMA is a standard that defines an electrical interface and data protocol for communications between marine electronics and instrumentation, and it may also have standards for other applications [**18**]. The UBX protocol, owned by U-Blox, allows us to receive information in raw format.

3.1.1.1. ***NMEA Protocol:*** The NMEA protocol transmits data in ASCII strings or "sentence" structures from one "talker" to multiple "listeners" at a time [**19**]. "Talkers" can include satellites, depth sounders (sonar), compasses, or anemometers, while "listeners" can include the GNSS receiver U-Blox EVK-M8T, chart-plotters, or radars.

The structure of NEMA sentences from the perspective of GPS receivers is as follows:

Each sentence always starts with the character '$', followed by a five-character address field consisting of uppercase letters. The first two characters are the Talker Identifier, and the next three are the Sentence Formatter. After the address field, there is a data field of variable length, followed by a checksum field that starts with a '*' and consists of

two characters representing a hexadecimal number. This checksum field represents the exclusive OR of all characters between '$' and '*'. Finally, after the checksum, there is a '<CR><LF>' sequence, which determines the end of the sentence [**3**] [**20**].

The following figure (3.1) illustrates the explained structure with an example:

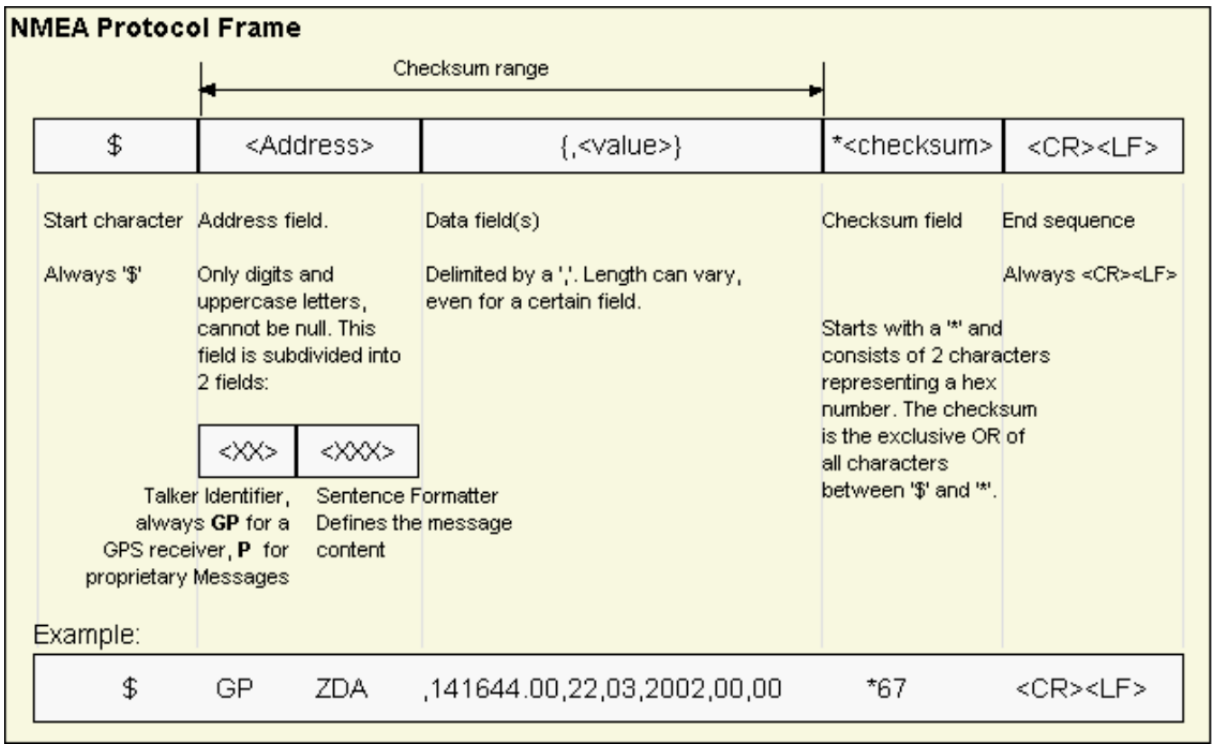

FIGURE 3.1. NMEA Protocol Frame [**2**].

The following figure (3.2) shows the description of each of the NEMA output messages:

Message	Description
GGA	Time, position and fix type data
GLL	Latitude, longitude, UTC time of position fix and status
GSA	GPS receiver operating mode, satellites used in the position solution, and DOP values
GSV	Number of GPS satellites in view satellite ID numbers, elevation, azimuth, & SNR values
MSS	Signal-to-noise ratio, signal strength, frequency, and bit rate from a radio-beacon receiver
RMC	Time, date, position, course and speed data
VTG	Course and speed information relative to the ground
ZDA	PPS timing message (synchronized to PPS)
150	OK to send message
151	GPS Data and Extended Ephemeris Mask
152	Extended Ephemeris Integrity
154	Extended Ephemeris ACK

FIGURE 3.2. NMEA Output Messages [**3**].

A full description of the listed NEMA output messages (3.2) can be found in [**3**].

During the development of the platform, initial efforts were made to handle NEMA protocol messages received by the GNSS receiver. The code responsible for this task reads data from the serial port one byte at a time, identifying the start of an NEMA sentence by detecting the '$' character. Once an NEMA sentence is detected, the code continues to read the rest of the sentence until the end-of-sentence characters '<CR><LF>' are encountered. Although the 'NMEAParser.java' file contains some preliminary code for parsing NEMA sentences, it was primarily experimental and served as a foundation for future development. When the program starts, it receives initialization messages such as:

```
Received NMEA sentence: $GPTXT,01,01,02,u-blox AG - www.u-blox.com*50
Received NMEA sentence: $GPTXT,01,01,02,HW UBX-M80xx 00080000 *5D
Received NMEA sentence: $GPTXT,01,01,02,EXT CORE 2.30 (86283) Oct 20 2014
 13:51:49*40
```

These messages indicate that the receiver is initializing and provide information about the hardware and firmware versions. Although the initial focus shifted towards parsing UBX protocol message types, the groundwork laid for handling NEMA messages remains a valuable starting point for future enhancements.

3.1.1.2. ***UBX Protocol:*** The UBX protocol is a proprietary protocol developed by U-Blox, utilized by U-Blox GPS receivers to transmit GPS data to a host computer using asynchronous RS232 ports [**2**]. This protocol supports a wide range of data types and configurations, making it suitable for various applications.

The structure of a UBX packet is as follows: Each message starts with 2 Bytes, 0xB5 and 0x62, which serve as synchronization characters. Following these, there is a 1

Byte Class field that defines the basic subset of the message, and a 1 Byte ID field that specifies the particular message type. Next, a 2 Byte Length field indicates the length of the payload, excluding the synchronization characters, Class, ID, and checksum fields. The length is represented as an unsigned 16-Bit integer in Little Endian format. The payload itself is a variable-length field containing the actual data of the message. Finally, the packet concludes with a two-byte checksum field, comprising CK_A and CK_B, which is a 16-bit value calculated over the Class, ID, Length, and Payload fields [**2**].

The following figure (3.3) visualizes the structure of a UBX packet:

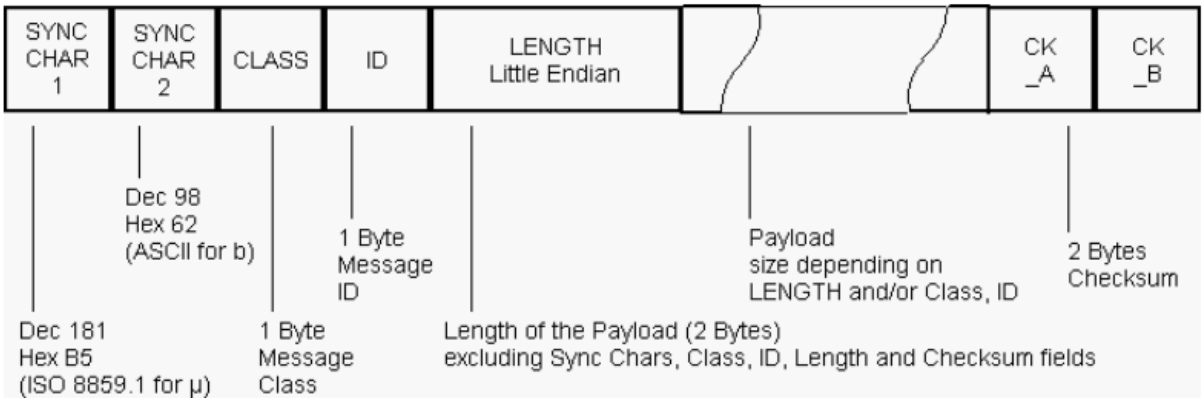

FIGURE 3.3. UBX Packet Structure [**2**].

Upon detecting the start of a UBX message, the system reads the entire message, including the class, ID, length, payload, and checksum. This ensures that the data is accurately captured and prepared for parsing. The UBX protocol's design allows for efficient handling of various message types, each identified by unique class and ID fields.

The parsing process involves interpreting the captured UBX messages by examining the class and ID fields to identify the message type and extracting the relevant fields from the payload. For instance, the `NAV-POSLLH` message, which provides position and altitude information, is parsed to extract the time of week, longitude, latitude, height, height above mean sea level, horizontal accuracy, and vertical accuracy. These fields are then converted into a readable format, such as degrees for longitude and latitude.

```
if (ubxMessage[2] == 0x01 && ubxMessage[3] == 0x02) {
    // Extract the fields from the payload
    int iTOW = getIntLittleEndian(ubxMessage, 6);
    int lon = getIntLittleEndian(ubxMessage, 10);
    int lat = getIntLittleEndian(ubxMessage, 14);
    int height = getIntLittleEndian(ubxMessage, 18);
    int hMSL = getIntLittleEndian(ubxMessage, 22);
    int hAcc = getIntLittleEndian(ubxMessage, 26);
    int vAcc = getIntLittleEndian(ubxMessage, 30);

    // Convert the longitude and latitude to degrees
```

```
        double lonDegrees = lon / 1e7;
        double latDegrees = lat / 1e7;
    }
```

These fields are extracted from the payload and converted to a readable format, such as degrees for longitude and latitude. The following figure (3.4) visualizes the structure and payload contents of the `NAV-POSLLH` message:

32.17.16 UBX-NAV-POSLLH (0x01 0x02)

32.17.16.1 Geodetic position solution

Message	**UBX-NAV-POSLLH**					
Description	**Geodetic position solution**					
Firmware	Supported on: • u-blox 8 / u-blox M8 protocol versions 15, 15.01, 16, 17, 18, 19, 19.1, 19.2, 20, 20.01, 20.1, 20.2, 20.3, 22, 22.01, 23 and 23.01					
Type	Periodic/Polled					
Comment	See important comments concerning validity of position given in section Navigation Output Filters. This message outputs the Geodetic position in the currently selected ellipsoid. The default is the WGS84 Ellipsoid, but can be changed with the message UBX-CFG-DAT.					
Message Structure	Header	Class	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x01	0x02	28	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	1e-7	lon	deg	Longitude
8	I4	1e-7	lat	deg	Latitude
12	I4	-	height	mm	Height above ellipsoid
16	I4	-	hMSL	mm	Height above mean sea level
20	U4	-	hAcc	mm	Horizontal accuracy estimate
24	U4	-	vAcc	mm	Vertical accuracy estimate

FIGURE 3.4. NAV-POSLLH Message Structure [**4**].

Since the header, class, ID, and length fields together are 6 bytes, the offsets in the code are 6 bytes greater than the offsets specified in the byte offset in figure 3.4 [**4**], which are relative to the start of the payload. For instance, the iTOW field, which has an offset of 0 in the manual, is accessed at offset 6 in the code. Similarly, the longitude field, with an offset of 4 in the manual, is accessed at offset 10 in the code. This adjustment ensures that the parser correctly interprets the payload data within the context of the entire UBX message structure.

Following the same implementation method, five other types of UBX messages were implemented, each with unique data that identifies them.

The example above, `NAV-POSLLH`, is described as "Geodetic Position Solution" and, as the name implies, it contains data regarding positioning (POS) and latitude, longitude, and height (LLH).

Another `NAV` type message is `NAV-TIMEUTC`, described as "UTC Time Solution," which includes data such as time accuracy estimate, the current date, and validity flags to validate the accompanying data.

The third type is `NAV-CLOCK`, described as "Clock Solution," which contains "clock bias," "clock drift," "time accuracy estimate," and "frequency accuracy estimate" used for timing accuracy and clock synchronization.

The `NAV-POSECEF` message is described as "Position Solution in ECEF," which means Earth-centred-Earth-fixed (ECEF) and includes the ECEF X, Y, and Z coordinates along with "Position Accuracy Estimate."

Next is `NAV-DOP`, described as "Dilution Of Precision," which contains dimensionless values representing the accuracy of the satellite in locating the receiver's antenna. Lower values indicate better accuracy, with 1 being a "good" indicator.

Lastly, `NAV-VELNED` is described as "Velocity Solution in NED Frame," which stands for VELocity in North, East, Down and contains data regarding the velocity in those directions.

[**4**]

This implementation demonstrates the application of knowledge from the "u-blox 8 / u-blox M8 Receiver description" [**4**]. By effectively handling and parsing UBX messages, the platform provides accurate and interpretable GPS data, which can be further utilized for various applications, including data export to text files and databases.

3.2. Data Extraction and Export

Although the messages are printed out in the terminal as they are received, it is crucial to store them for further use in other applications, such as being displayed in a user interface. Given that this platform is intended to run on a low-power device like a Raspberry Pi, storage management becomes a significant concern. To address this, the system is designed to delete text files or database entries older than 30 days to preserve storage space. These retention periods can be easily adjusted, allowing for flexibility during testing and deployment. For instance, during testing, shorter retention periods were used to ensure the logic was functioning as intended. This section will delve into the mechanisms for extracting and exporting data, detailing how messages are stored and managed to maintain system efficiency and reliability.

3.2.1. Text File Extraction and Management

The system is designed to extract UBX messages and store them in text files for further analysis. The extraction process is scheduled to run every 4 hours, during which 1 minute of continuous UBX messages is captured and saved. Each extracted file is named with the prefix `UBXExtraction_` followed by the current date and time in the

format `YYYY_MM_dd_HH_mm_ss`. This naming convention ensures that each file is uniquely identifiable based on the time of extraction.

To manage storage efficiently, the system includes a mechanism to delete older files. This cleanup process runs once a day and removes any files that are older than 30 days. Running the cleanup process daily ensures that if more frequent extractions are performed, such as every minute, the files are not deleted immediately upon reaching the age threshold. This approach prevents excessive deletion operations and maintains a balance between data retention and storage management.

The extraction and deletion processes are handled by scheduled tasks, ensuring that they run automatically at the specified intervals without requiring manual intervention. This automated approach helps maintain system efficiency and reliability, allowing the platform to operate continuously with minimal maintenance.

3.2.2. Database Storage and Management

The choice of database for this platform is SQLite, which is particularly well-suited for environments where messages are constantly being received and stored. SQLite is a lightweight, embedded database that runs within the application itself, eliminating the need for a separate database server. This makes it an ideal choice for low-power devices like the Raspberry Pi. SQLite requires minimal setup and configuration, offers efficient storage and fast read/write operations, and is Atomicity, Consistency, Isolation, Durability (ACID)-compliant, ensuring reliable transaction processing and data integrity. Additionally, its low memory footprint and CPU usage make it highly suitable for resource-constrained platforms.

In this system, a batch processing approach is used for exporting messages to the database, rather than writing each message individually as it arrives. Messages are received and parsed every second, and using batch processing reduces the number of write operations to the database, which is beneficial for performance and longevity of the storage medium. Specifically, the system stores one message every 60 seconds, resulting in 43,200 messages for each type of message over a 30-day period. By batching these messages, the system only writes one out of every 60 messages to the database, significantly reducing the write frequency.

Similar to the file extraction process, a First-In-First-Out (FIFO) philosophy is applied to manage the database entries. The system includes a mechanism to delete database entries that are older than 30 days, running this cleanup process once a day. This retention policy helps to conserve storage space while ensuring that recent data is readily available for use. The retention period and batch size can be easily adjusted to meet different requirements during testing and deployment, providing flexibility and adaptability to the system.

Overall, the use of SQLite and batch processing for database storage and management ensures efficient, reliable, and scalable handling of the continuous stream of messages received by the platform.

The image below (3.5) showcases the current tables in this platform.

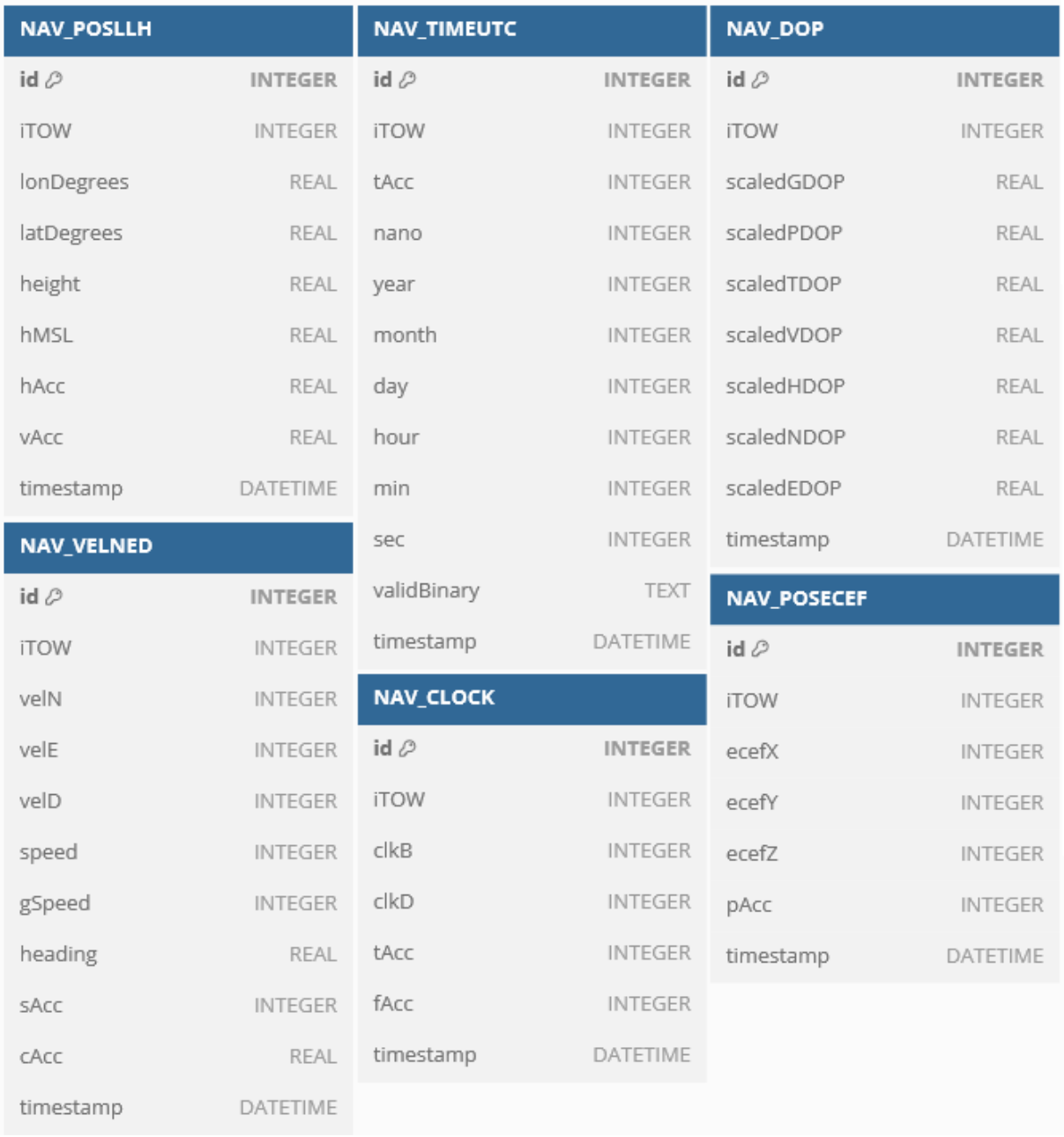

FIGURE 3.5. Database Tables

3.3. Interface

The interface plays a crucial role in this platform by providing users with access to the data received by the GNSS receiver and stored in the database. This interface is implemented as a webpage, which can be accessed both locally within ISCTE and remotely via a VPN. The use of a VPN ensures a secure connection to the campus network from outside locations, maintaining the integrity and confidentiality of the data. The webpage interface allows users to view specific GNSS data, perform queries to filter and retrieve particular messages from the database, and analyze the GNSS data effectively.

3.3.1. Webpage

The webpage uses as its foundation the simple PHP file browser, Encode Explorer, which "was designed to be used in safe mode and so it is kept simple and functional" [**21**]. This simplicity and functionality make it ideal for an interface hosted on a Raspberry Pi. Encode Explorer includes an option for password protection, which is utilized in this case. The rest of the interface, beyond the login page, can only be accessed if the login is successful. Upon logging in, users are presented with a list of the text files exported mentioned in subsection 3.2.1.

Additionally, two more webpages were created. The first is "Database Access," which allows users to view all messages of each type by selecting the desired type to be shown in a table. The second is "Receiver Location," which retrieves the most recent coordinates of the receiver from the database and displays its location on a map. This map supports four different map tile layers with different purposes, provided by Leaflet [**22**].

The programming languages used for the webpage include PHP: Hypertext Preprocessor (PHP) for handling database connections, queries, and logic; HTML for structuring content; CSS for styling; and JavaScript for handling map interactions.

3.4. Hardware

The platform consists of an antenna connected to the GNSS receiver, which in turn is connected to the Raspberry Pi. The Raspberry Pi is powered and connected to the network via Ethernet.

3.4.1. U-Blox EVK-M8T

Connected to the Raspberry Pi through USB since it provides power and data transfer. The following table [1] contains the receivers specifications:

Parameter	Specification
Serial Interfaces	1 USB V2.0
	1 RS232, max. baud rate 921,6 kBd DB9 ±12 V level 14 pin -3.3 V logic
	1 Direct Digital Control (DDC) (I2C compatible) max. 400kHz
	1 SPI - clock signal max. 5,5 MHz - SPI DATA max. 1 Mbit/s
Timing interfaces	2 Time-pulse outputs
	1 Time-mark input
Power Supply	5V via USB or external powered via extra power supply pin 14 (V5_IN) 13 (Ground (GND))
Normal Operation temperature	-40^{o}C to +65^{o}C

TABLE 1. EVK-M8T Technical Specifications [**23**]

3.4.2. Raspberry Pi 4 - Model B - 1GB RAM

The device is running the Ubuntu Server version 23.04 operating system. Since it is lighter in terms of resource usage compared to the desktop version, it is more suitable for this platform. An overview and specifications of this model can be found in [**24**].

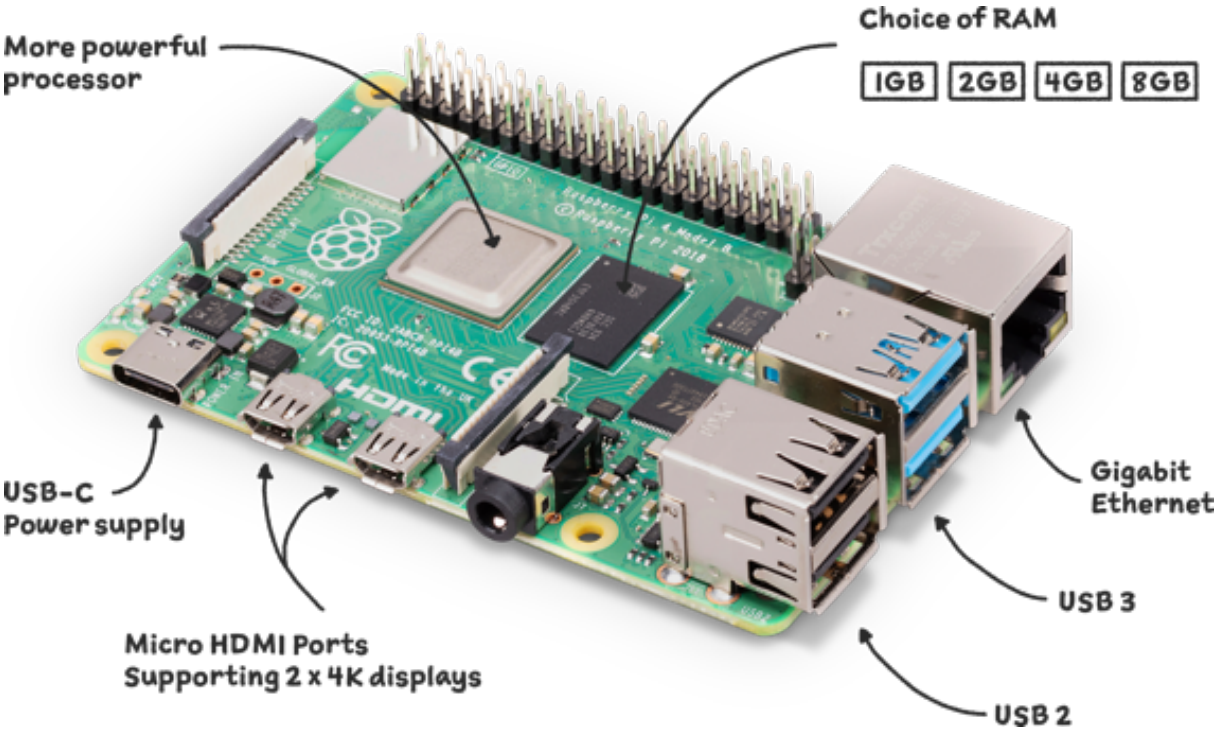

FIGURE 3.6. Raspberry Pi 4 - Model B - 1GB RAM [**5**].

CHAPTER 4

Results

Even as these results were being gathered, some errors and inconsistencies were found, which led to improvements in the platform. Here are the results of the development of this platform.

The following image demonstrates direct access to the platform:

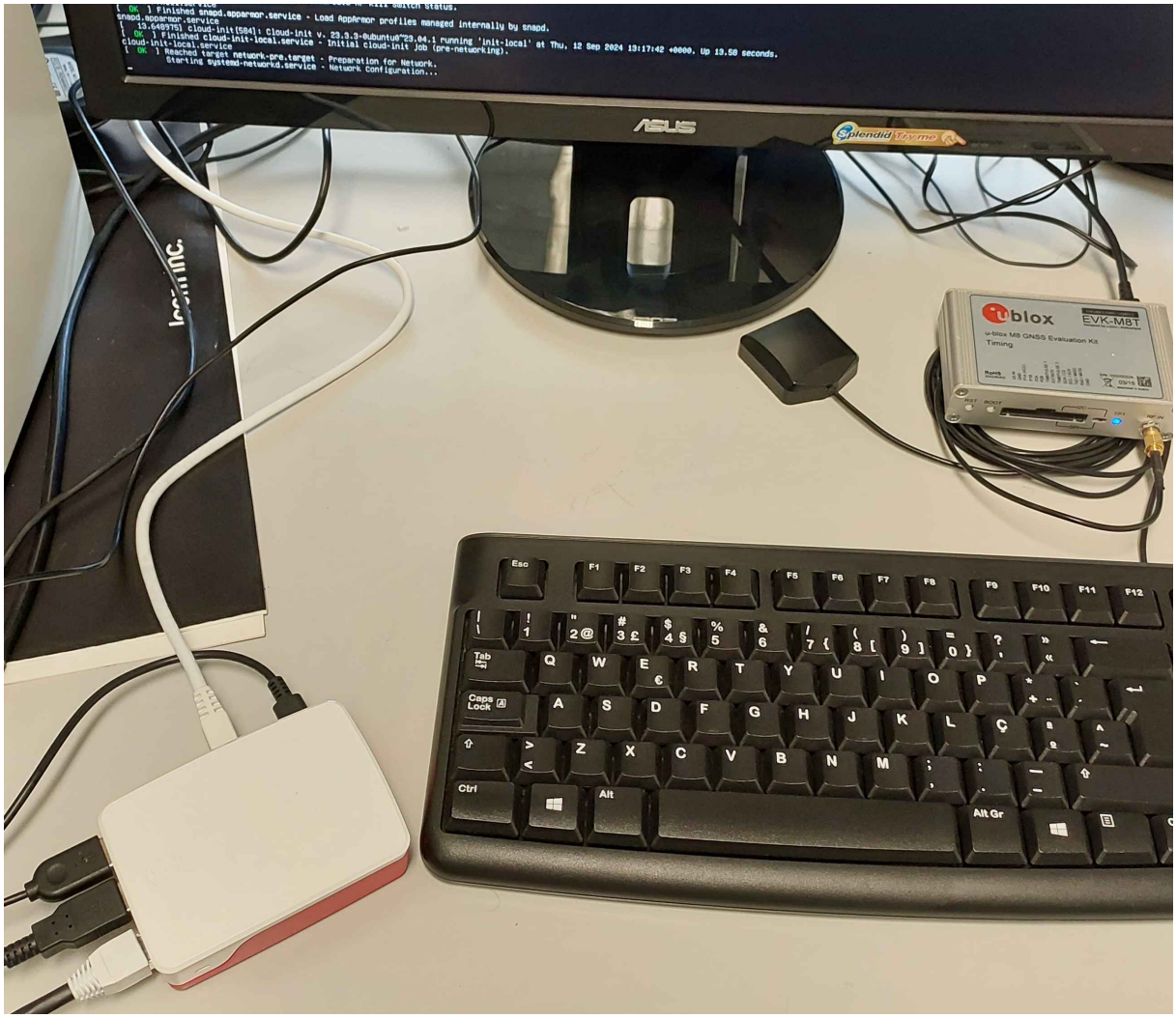

FIGURE 4.1. Raspberry Pi connected to a keyboard, GNSS receiver, and monitor.

When first turning on the Raspberry Pi, it is necessary to access it directly by connecting a monitor and keyboard to configure the network settings and set up its IP address. Once this is done, we can connect to it via SSH using the program PuTTY [**25**].

After starting the program, we can see the output of the messages in real time:

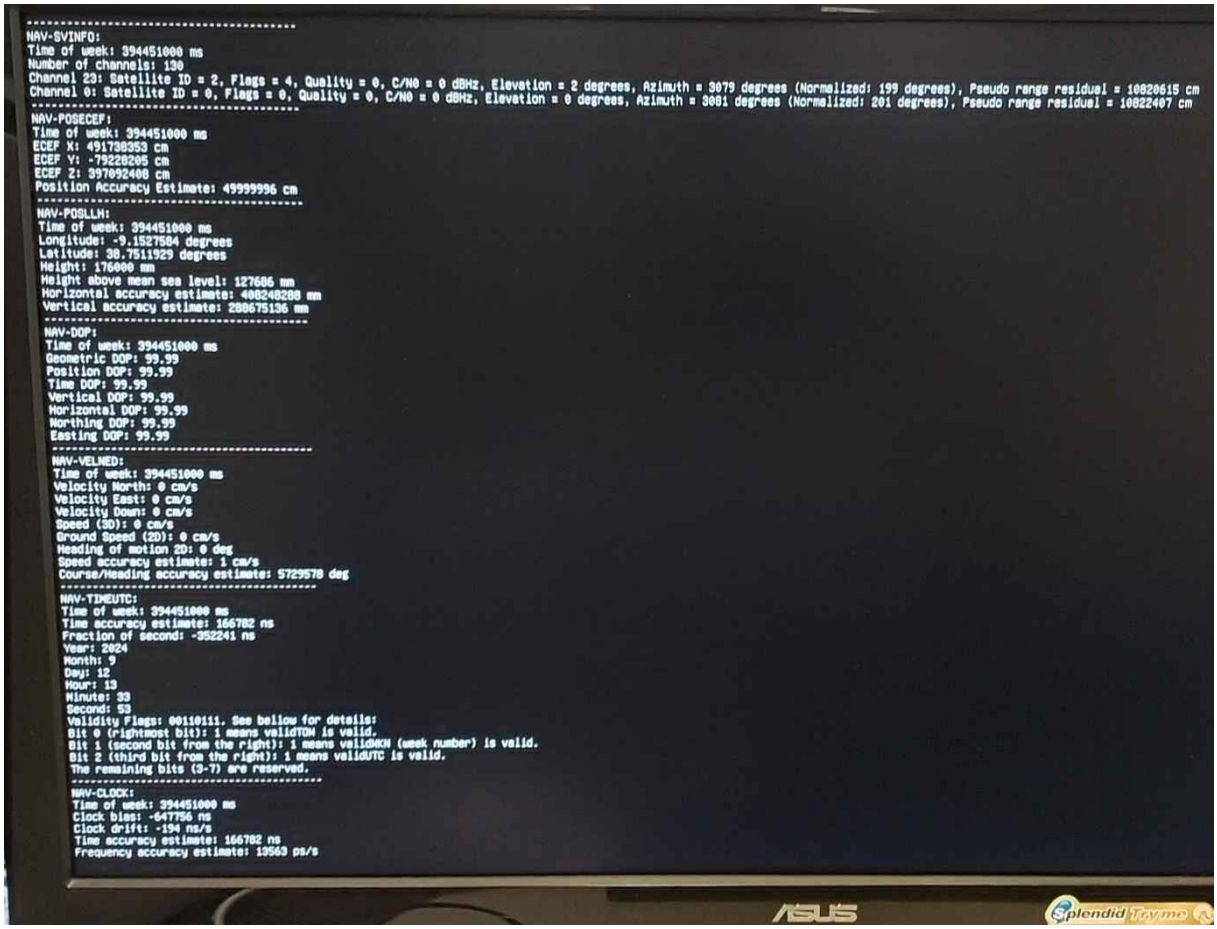

FIGURE 4.2. Output of the messages in real time.

These messages are received every second, making them unsuitable for detailed analysis. To address this, an interface for the platform was developed.

The following screenshots were all taken while remotely connected through VPN.

Here is the login page:

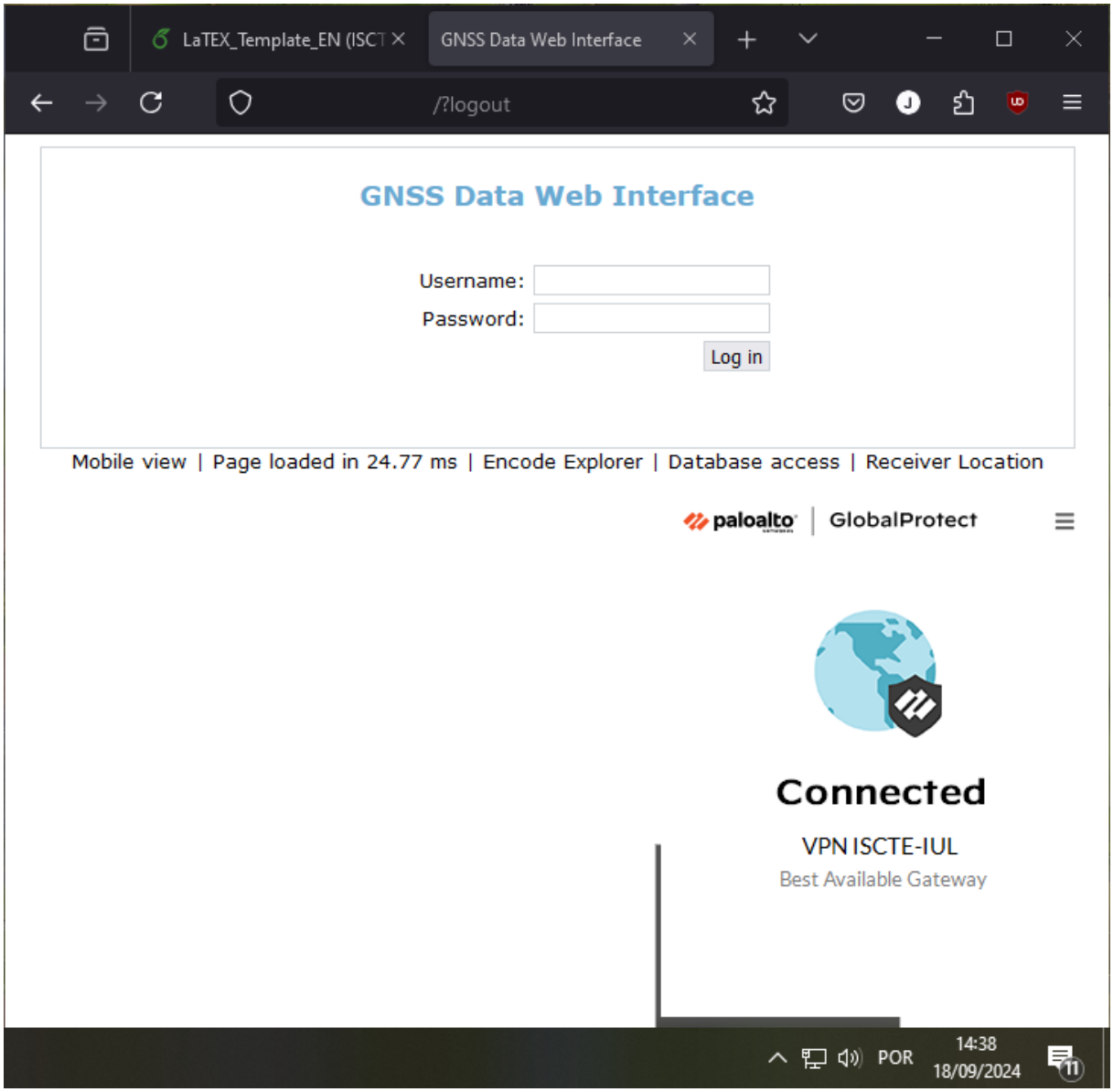

FIGURE 4.3. Login page.

To access any of the data, login credentials are required.

Here is the screen after logging in:

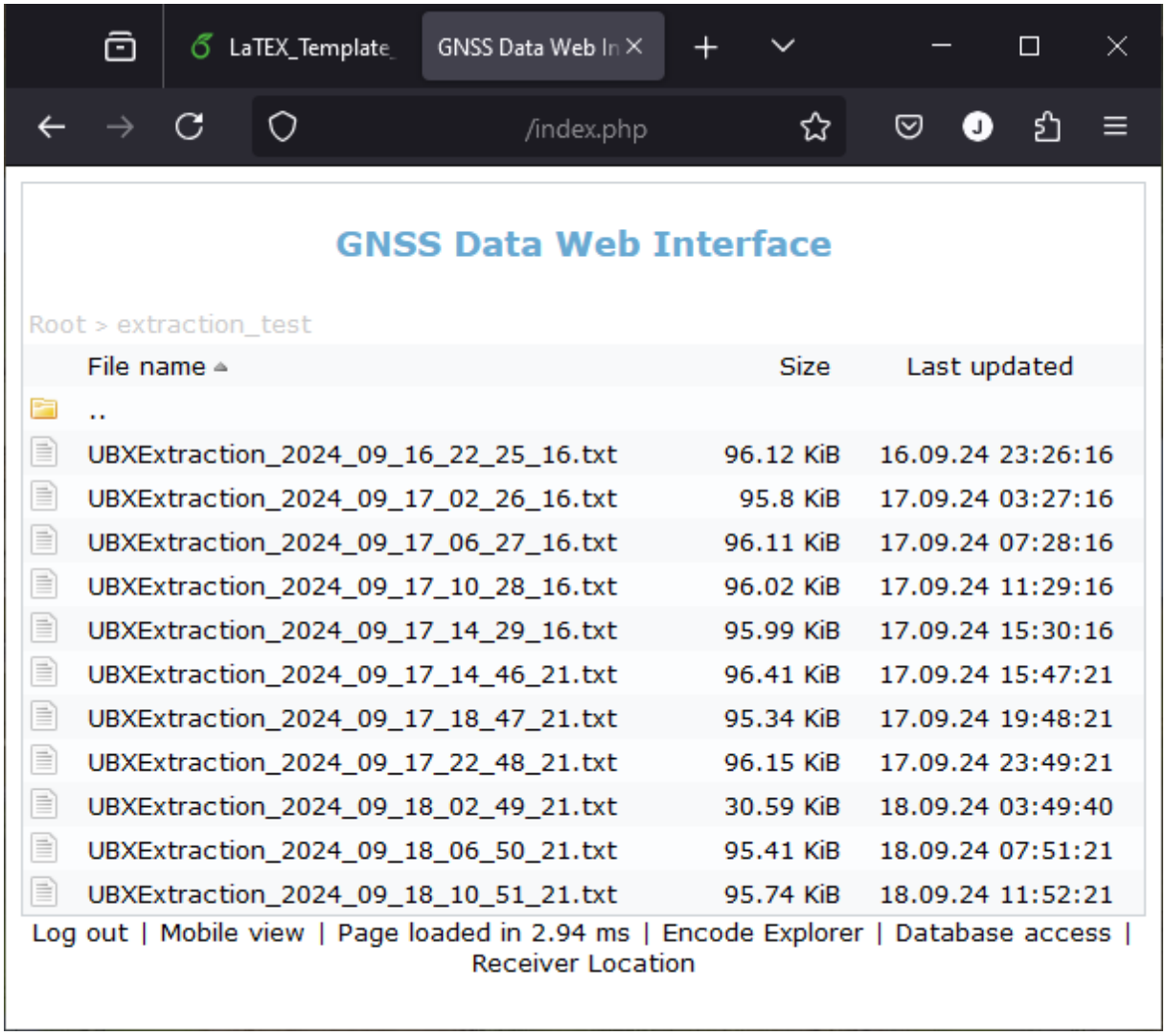

FIGURE 4.4. Screen after logging in.

After logging in, we have access to the text files, which can be opened in the browser. Additionally, at the bottom of the screen, there are added options for "Database access" and "Receiver Location". These options can only be accessed if logged in.

Here is the Database access page:

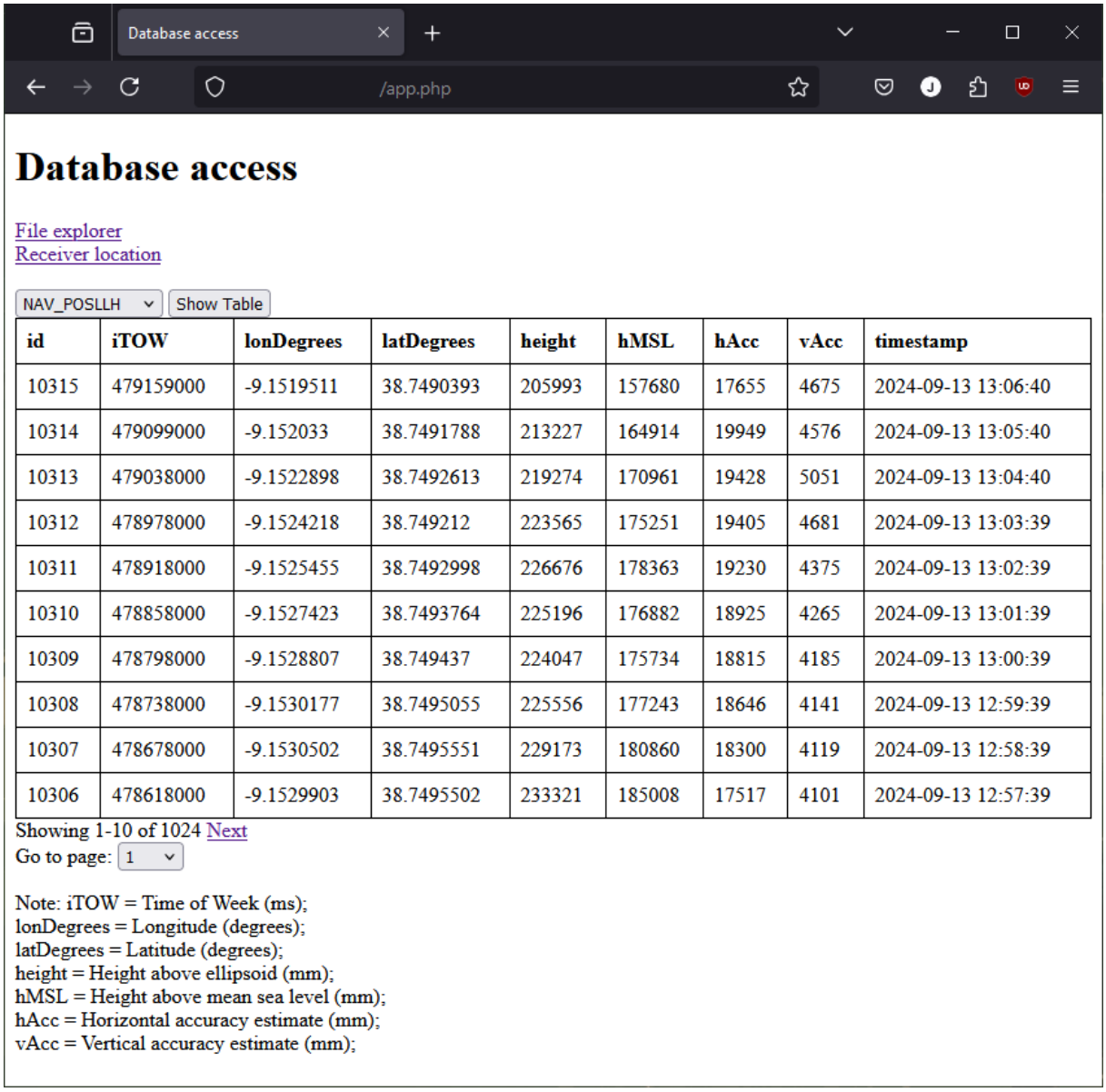

FIGURE 4.5. Database access page.

This webpage displays a simple table that shows entries from the database, including all types of messages, starting from the most recent entry, as described in Chapter 3. Each entry has a note below it clarifying what each of the columns represents and their respective measurements. At the top, there are two links: one to go back and another to go to the Receiver location page.

Here is the Receiver location page:

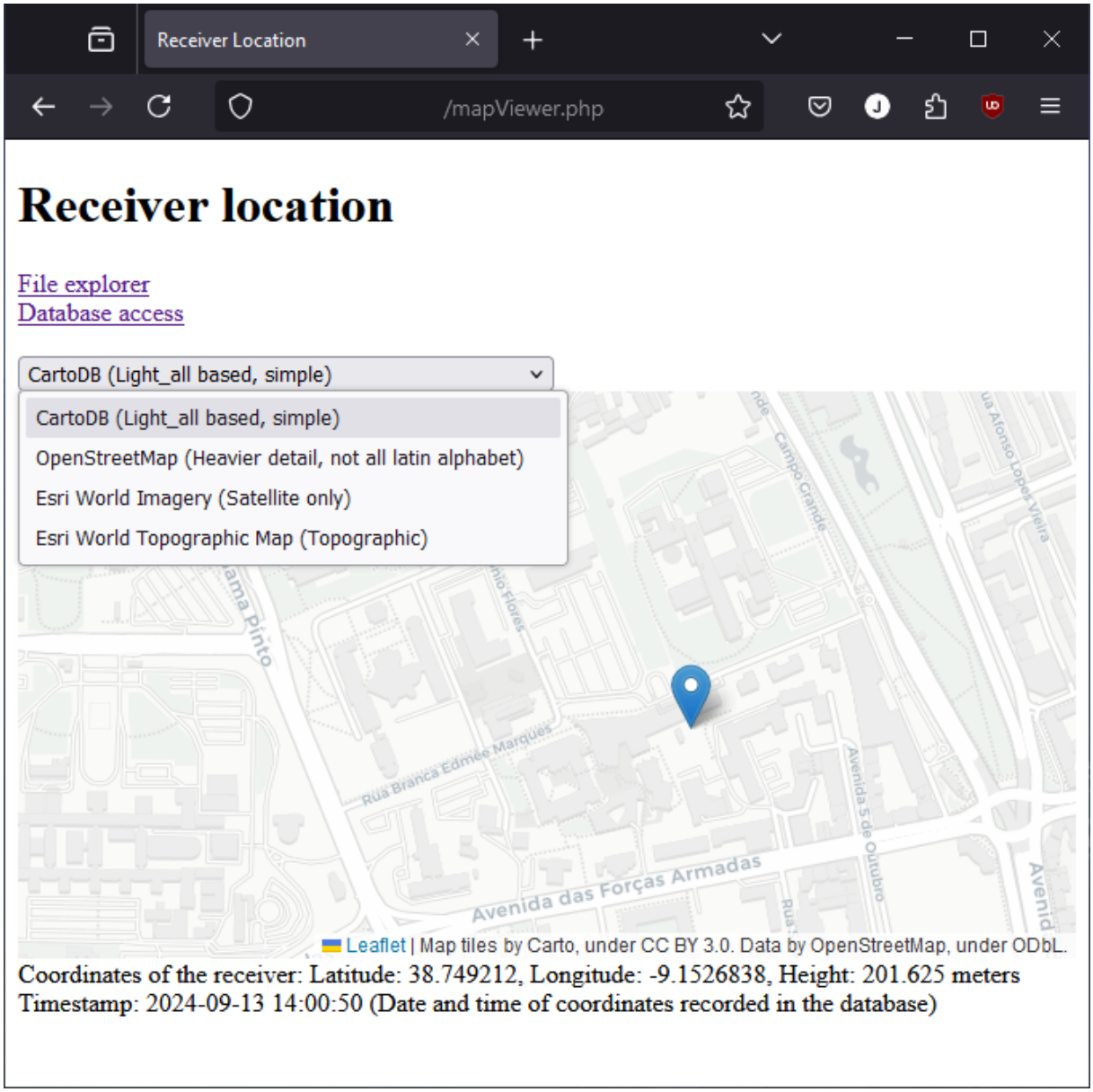

FIGURE 4.6. Receiver location page.

Finally, on this webpage, we have a map displaying the location of the receiver. By default, it starts with a simple light map, but there is an option to choose other types if needed. Below the map, there is information from the last entry in the database about the receiver's location.

CHAPTER 5

Conclusions

This thesis focused extensively on GNSS, its applications, the constellations it involves, and satellite communications. Although the primary focus was on GNSS, the project also integrated synergies from other academic areas to make the platform functional. These areas included programming for retrieving and parsing data from the GNSS receiver, web development to enable data analysis, and networking to allow remote access to the Raspberry Pi.

Problems were systematically addressed by analyzing the output and reviewing potential improvements. This approach was particularly crucial in the initial phase, where efforts were made to distinguish UBX messages from other data that appeared as ASCII characters in the output terminal. Further study of the manual by u-blox [**4**] was vital for successfully parsing UBX messages.

One of the mistakes made was not considering data export and extraction earlier in the project. This oversight resulted in additional time spent later on fixing the UBX message parser to ensure each message could be extracted to a text file or database.

Although this project resulted in a simple remote GNSS server station, it provides a solid foundation for future development. As mentioned in subsection 3.1.1.1, the NEMA message type was primarily studied for its protocol and structure, leaving room for further development in handling NEMA messages. Regarding the UBX protocol, which was the primary focus, more message types can be parsed following the examples of the existing ones. New messages, such as `UBX-MGA` (Multiple GNSS Assistance messages), could be parsed to provide additional satellite information. This information could then be used to calculate satellite positions and display them on the map implemented in the interface webpage.

In conclusion, this thesis has laid the groundwork for a functional GNSS platform, integrating various academic disciplines and providing a basis for future enhancements and applications.

References

[1] J. Sanguino and F. Cercas, "Global navigation satellite systems."

[2] EF, "Nmea, ubx protocol specification." https://www.sparkfun.com/datasheets/GPS/Modules/u-blox5_Protocol_Specifications(GPS.G5-X-07036).pdf, Last accessed 10 September 2023.

[3] SiRF, "Nmea reference manual." https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual-Rev2.1-Dec07.pdf, Last accessed 08 September 2023.

[4] ublox, "u-blox 8 / u-blox m8 receiver description." https://www.u-blox.com/docs/UBX-13003221, Last accessed 14 August 2024.

[5] R. P. T. Ltd., "Raspberry pi 4 computer, model b." https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, Last accessed 19 October 2023.

[6] M. Perez-Ruiz and S. K. Upadhyaya, "Gnss in precision agricultural operations," in *New Approach of Indoor and Outdoor Localization Systems* (F. B. Elbahhar and A. Rivenq, eds.), ch. 1, Rijeka: IntechOpen, 2012.

[7] X. Haizhong, "Gnss application & practice in fishery and timing in china."

[8] O. U. government information about the Global Positioning System (GPS) and related topics, "Timing." https://www.gps.gov/applications/timing/, Last accessed 02 February 2023.

[9] O. U. government information about the Global Positioning System (GPS) and related topics, "Space segment." https://www.gps.gov/systems/gps/space/, Last accessed 02 February 2023.

[10] O. U. government information about the Global Positioning System (GPS) and related topics, "Control segment." https://www.gps.gov/systems/gps/control/, Last accessed 02 February 2023.

[11] E. E. U. A. for the Space Programme, "What is galileo?." https://www.gsc-europa.eu/galileo/what-is-galileo, Last accessed 03 February 2023.

[12] "About glonass." https://glonass-iac.ru/en/about_glonass/, Last accessed 03 February 2023.

[13] B. N. S. System, "The main architecture." http://en.beidou.gov.cn/SYSTEMS/System/, Last accessed 03 February 2023.

[14] O. U. government information about the Global Positioning System (GPS) and related topics, "Navstar gps space segment/navigation user interfaces." https://www.gps.gov/technical/icwg/IS-GPS-200D.pdf, Last accessed 04 February 2023.

[15] N. O. of Cyber Security & Critical Infrastructure Coordination, "Glossary of gps terminology." https://gis.ny.gov/coordinationprogram/reports/presentations/gps/GPS_Glossary.pdf, Last accessed 04 February 2023.

[16] D. Petrov, S. Melnik, and T. Hämäläinen, "Distributed gnss-based time synchronization and applications," in *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 130–134, 2016.

[17] L. Silva, P. Romano, V. Francisco, and R. Ferreira, "Gnsstracker." Master's degree in Telecommunications and Computer Engineering, in the subject of Digital Satellite Communications Systems, from 2016.

[18] P. Bennett, "The nmea faq." https://web.archive.org/web/20140215150802/http://www.kh-gps.de/nmea.faq, Last accessed 19 August 2023.

[19] J. Bagur, "Gps nmea 0183 messaging protocol 101." https://docs.arduino.cc/learn/communication/gps-nmea-data-101, Last accessed 08 September 2023.

[20] E. S. Raymond, "Nmea revealed." https://gpsd.gitlab.io/gpsd/NMEA.html, Last accessed 08 September 2023.

[21] M. Rei, "Encode explorer." http://encode-explorer.siineiolekala.net/, Last accessed 23 October 2023.

[22] V. Agafonkin, "Leaflet." https://leafletjs.com/, Last accessed 03 September 2024.

[23] ublox, "Evk-m8t." https://content.u-blox.com/sites/default/files/products/documents/EVK-M8T_UserGuide_%28UBX-14041540%29.pdf, Last accessed 18 October 2023.

[24] R. P. T. Ltd., "Raspberry pi 4 computer, model b." https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf, Last accessed 19 October 2023.

[25] S. Tatham, "Putty." https://www.putty.org/, Last accessed 24 September 2024.