POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA CIVILE,
AMBIENTALE E TERRITORIALE

ADDITION OF METEOROLOGICAL INFORMATION TO GNSS DATA PROCESSING SOFTWARE

(Master of Science Degree Thesis in Geoinformatics Engineering)

BY

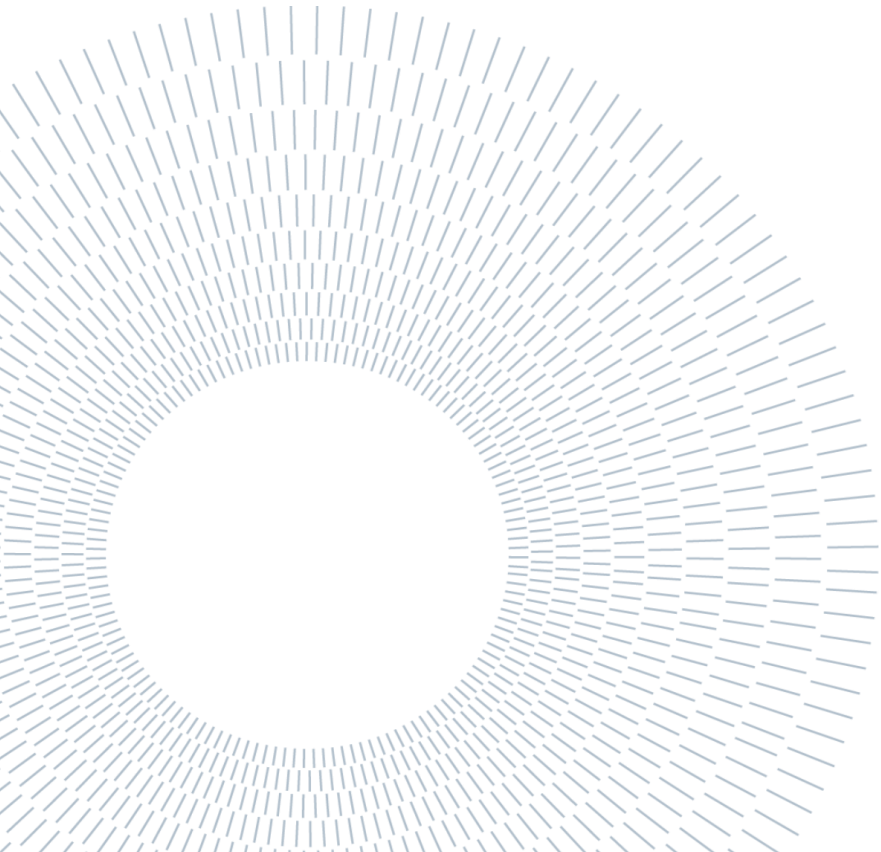
FELIX ENYIMAH TOFFAH

Student Id: **936545**

Advisor: **Professor Giovanna Venuti**

Co-Advisor: **Andrea Gatti**

Academic Year: **2020/2021**

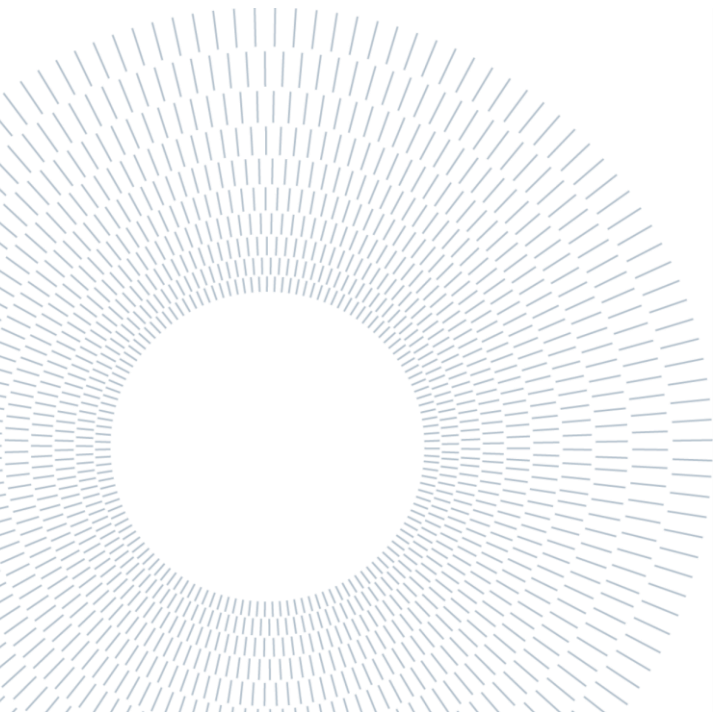


DEDICATION

I dedicate this work to my parents Mr Kobina Ebo Toffah and Miss Aba Abokomah.

"Have I not commanded you? Be strong and courageous. Do not be afraid; do not be discouraged, for the Lord your God will be with you wherever you go."

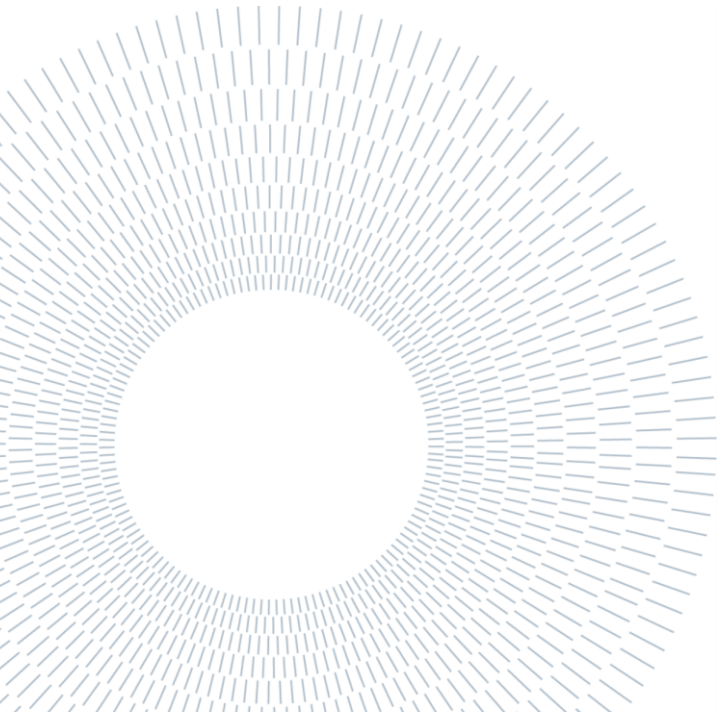
(Joshua 1:9, NIV)



ABSTRACT

The use of GNSS for positioning purposes, with the technology undergoing rapid advancement, has been widely explored. The interaction of the contents of the atmosphere, such as water vapour, with GNSS satellite signals makes the technology find application in meteorology. This is due to the possibility to estimate atmospheric water vapour content using the signal propagation delay caused by this interaction. One software for processing GNSS data is goGPS. In addition to providing estimates of a station coordinates, it is also able to estimate and produce plot of atmospheric water vapour contents. This is very useful in meteorology, for instance, to monitor the trend of the atmospheric water vapour content over an area. While making this analysis, it will be helpful to know the trend of other meteorological parameters such as temperature, pressure and wind direction. Also, having one plot containing the time series of the atmospheric water vapour content and the meteorological parameters will help interpret the water vapour trend. This has been the objective of this work, displaying the meteorological data obtained from a selected data source on the plots produced by goGPS software. This has been done using the graphical objects of Matlab, as goGPS is dependent on Matlab. In addition, RINEX meteorological files containing the data displayed on the plots are produced for future purposes. As an illustration of the usage of the visualisation tool, a case study was considered for a July 13, 2021 event at Malpensa Airport.

Key-words: goGPS, weather information, GNSS, atmosphere



SOMMARIO

L'uso del GNSS per scopi di posizionamento, con la tecnologia in rapido avanzamento, è stato ampiamente esplorato. L'interazione dei contenuti dell'atmosfera, come il vapore acqueo, con i segnali satellitari GNSS fa sì che la tecnologia trovi applicazione nell'area della meteorologia. Ciò è dovuto alla possibilità di stimare il contenuto di vapore acqueo atmosferico utilizzando il ritardo di propagazione del segnale causato da questa interazione. Un software per l'elaborazione dei dati GNSS è goGPS. Oltre a fornire stime delle coordinate di una stazione, è anche in grado di stimare e produrre grafici del contenuto di vapore acqueo atmosferico. Ciò è molto utile in meteorologia, ad esempio, per monitorare l'andamento del contenuto di vapore acqueo atmosferico su un'area. Durante l'analisi, sarà utile conoscere l'andamento di altri parametri meteorologici come temperatura, pressione e direzione del vento. Inoltre, avere un grafico contenente le serie temporali del contenuto di vapore acqueo atmosferico ei parametri meteorologici aiuterà a interpretare l'andamento del vapore acqueo. Questo è stato l'obiettivo di questo lavoro, visualizzare i dati meteorologici ottenuti da una fonte di dati selezionata sui grafici prodotti dal software goGPS. Questo è stato fatto utilizzando gli oggetti grafici di Matlab, poiché goGPS dipende da Matlab. Inoltre, per scopi futuri, vengono prodotti i file meteorologici RINEX contenenti i dati visualizzati sui lotti. A titolo illustrativo dell'utilizzo dello strumento di visualizzazione, è stato considerato un caso di studio per un evento del 13 luglio 2021 all'aeroporto di Malpensa.

Parole chiave: goGPS, informazioni meteo, GNSS, atmosfera

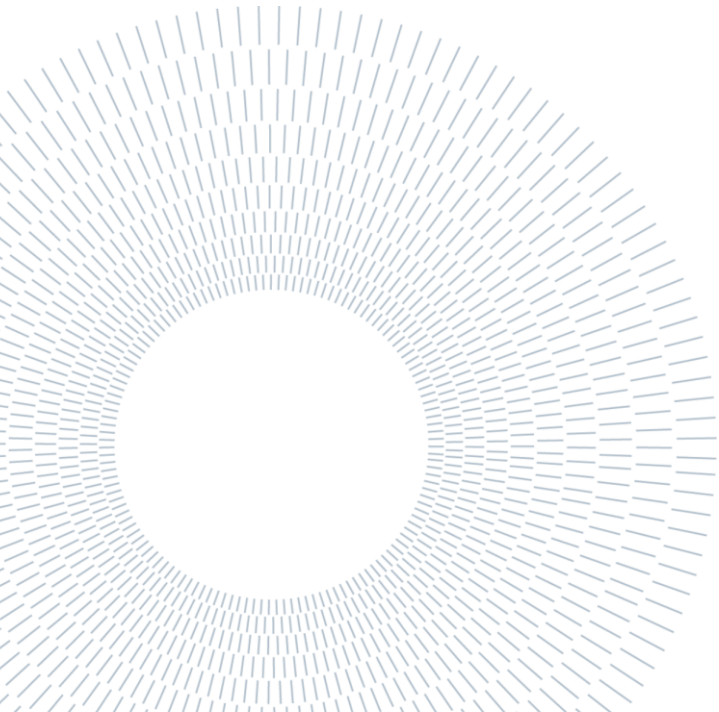
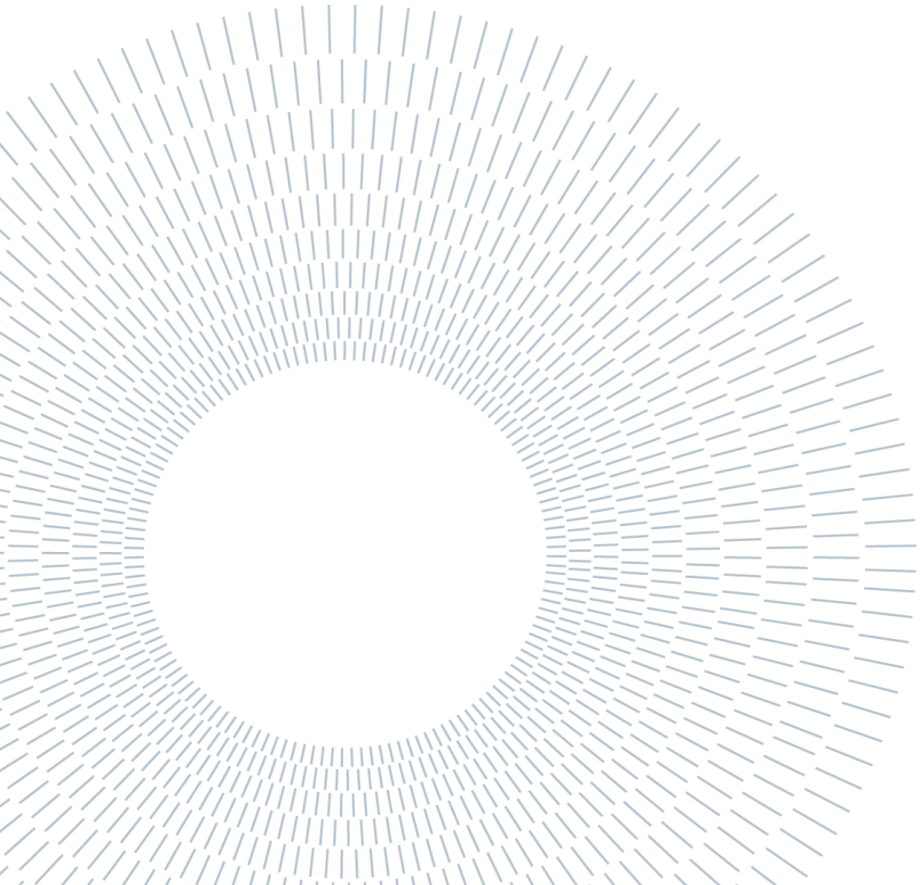


TABLE OF CONTENTS

x



CHAPTER 1

INTRODUCTION

1.1 Background information and statement of problem

The amount of water vapour in the atmosphere varies considerably over an area and across the globe. The water vapour and the interaction of the various elements of the atmosphere with electromagnetic radiations, such as solar radiation, play a major role in the changing state of the atmosphere and its dependent parameters such as pressure, temperature and humidity.

Several instruments are available for measuring atmospheric parameters. These instruments operate on land (such as thermometer, wind vane and barometer at Meteorological stations), on sea (such as acoustic depth sounder and marine sounding probes) and in the air (such as artificial satellites, Meteorological aircrafts, aerosondes and radiosondes). Data collected by instruments that operate in the air medium are very helpful as most parameters that influence the changing state of the atmosphere are found there. Among these, radiosondes (Figure 1) are commonly used to measure various atmospheric parameters including water vapour, pressure, and temperature. However, the cost of acquiring a radiosonde and establishing a radiosonde ground monitoring station makes it difficult to increase the spatial extent of a radiosonde network (Flores et al., 2013).

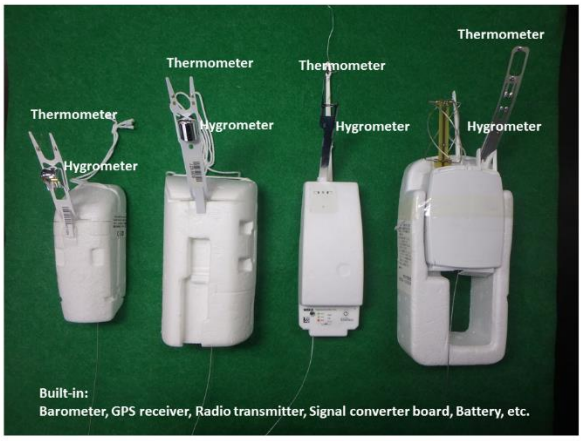


Figure 1 ***Radiosondes (source: Japan Meteorological Agency)***

Although not physically operating in the air, Global Navigation Satellite System (GNSS) measures space vehicle emitted signals that travel through and interact with the components of the atmosphere. The GNSS system is composed of space vehicles that operate in the skies, a control segment that monitors the space vehicles and user receivers that observe the signals emitted by the space vehicles for various purposes and applications.

The use of GNSS for positioning and other scientific purposes has been widely explored and has undergone impressive technological advancements. Accurate positioning has been possible through appropriate modelling of atmospheric interaction of the satellite signals which is subdivided into ionospheric and tropospheric components.

The ionospheric delay is caused by the interaction of free electrons in the ionosphere (about 100 to 1000 km altitude) with the satellite signal. This delay can be removed by a proper combination of observations on L1 (1575.42 MHz) and L2 (1227.60 MHz) frequencies made by double frequency receivers. For single frequency receivers, standard models (such as Klobuchar ionospheric model) exist for evaluating and removing this effect from the signal observations.

The tropospheric delay is partly caused by the presence of water vapour in the troposphere (0 to 40 km above the Earth Surface). This delay affects all the signals received by the GNSS station from all the satellites simultaneously in view at the time of observation. In standard processing of GNSS observations, the delays on the signal path from all the satellites in view are mapped in a common delay in the zenith direction above the receiver. This common delay known as the Zenith Tropospheric (or Total) Delay (ZTD) can be modelled using for instance the Saastamoinen model. The ZTD is the sum of the delay caused by the water vapour (Zenith Wet Delay, ZWD) and the delay caused by the dry gases of the troposphere in hydrostatic equilibrium (Zenith Hydrostatic Delay, ZHD). If temperature and pressure values are known at the time of observation, it is possible to exploit them to compute the ZHD component of the delay and to derive the wet component. Starting from the estimated ZTD, ZWD can be used to monitor the presence of water vapour in the atmosphere. Both ZTDs and ZWDs can then be assimilated into

numerical weather prediction models to enhance the prediction of heavy rains. The varying state of the atmospheric water vapour and its impact on the space vehicle signals make GNSS an innovative approach for measuring atmospheric water vapour contents.

Due to the rapid temporal and spatial variability of the water vapour content of the atmosphere, localised and dense observations of GNSS derived ZTDs are required for instance in the processing of convective storms. Just like radiosondes, a densified network of geodetic GNSS receivers for monitoring purposes has high costs of setup. Moreover, the risk of damages and losses can be discouraging. A monitoring system based on low-cost GNSS receivers represents a possible solution. As a justification for using these receivers, Biagi *et al*. (2016), achieved sub-cm accuracy levels when they used low-cost GNSS receivers for monitoring displacements and deformations. Also, Fermi *et*. *al* (2018) found that data from single frequency GNSS receivers (of which class belongs low-cost receivers) could provide more accurate results in estimating ZWD parameter for Meteorological purposes. In this light, many observation stations using GNSS satellite signals as an aid for meteorological purposes also use low-cost receivers.

One software that can be used for post-processing GNSS observations is goGPS (see Chapter 4 for detailed description of the performance of goGPS). By making a plot of the processing results, goGPS produces, among others, plots of estimated amount of precipitable water vapour in the atmosphere (see Chapter 3 for further discussions) at an observation station (Figure 2).

However, while making estimates of the amount of precipitable water vapour and monitoring the position coordinates and ZTD data of a GNSS station, it might be useful to know the state of the atmosphere such as the variability of temperature and pressure to better understand the impact of the atmosphere on the time variation of the ZTD and position coordinates. This is very essential as the GNSS station is exposed to the atmosphere and the observed signals traverse the atmosphere from the space vehicle to the receivers. Some events such as earthquake, tsunami and flood can also significantly affect the GNSS observations.

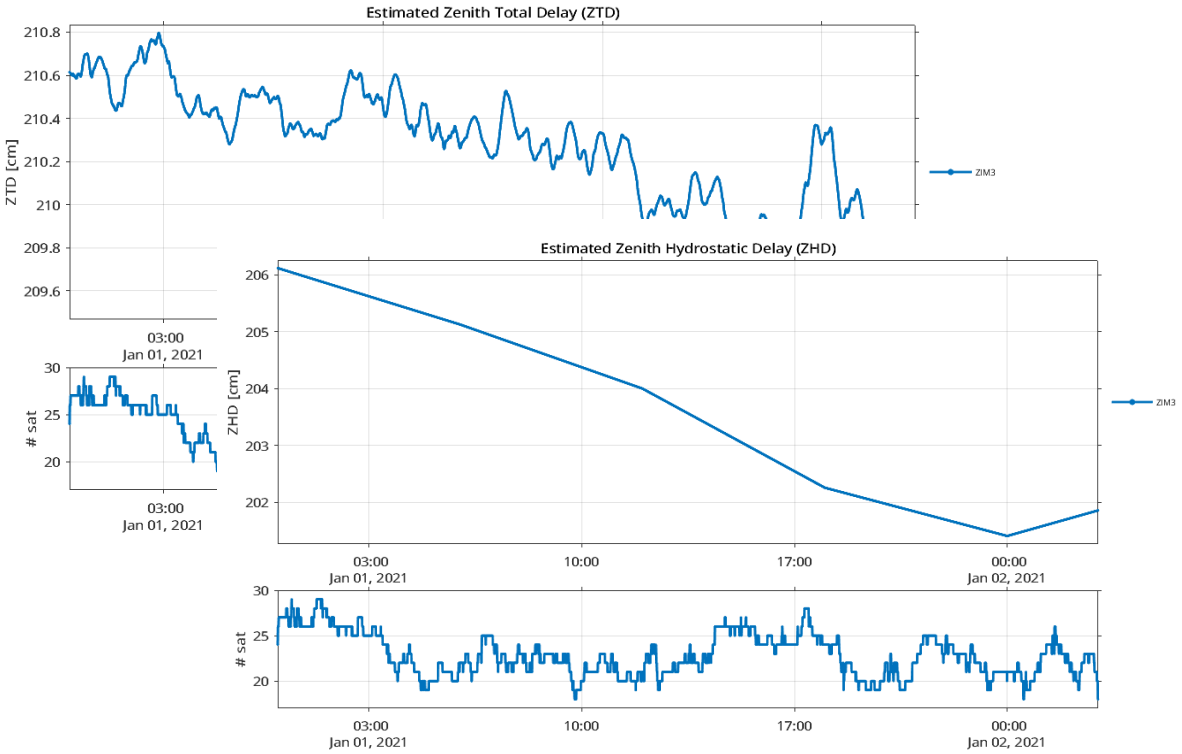


Figure 2 ***Current graphical outputs of goGPS***

Also, these events are useful for interpreting the GNSS position and ZTD estimates. Knowing these meteorological events and conditions will help better understand the trend of plots of the processing result.

In most cases, to analyse the trend of weather parameters such as temperature, pressure and water vapour content at a given station, independent plots are produced for each variable under consideration and the relation between the variables made by comparing the trends in the different plots. Producing a single plot that contains all those variables is time saving and makes the comparisons easier.

The current interface of plots produced by goGPS is as in Figure 3. It includes the standard menu items (File, Edit, View, etc.) found in a figure object in Matlab, additional Export menu item for exporting the plots for other purposes and the Aspect menu item for changing the background colour of the figure object. It also contains axes objects that show time series of the number of satellites and other parameters such as ZWD, ZHD and ZTD according to which variable is plotted.

Figure 3 ***goGPS plot of estimated ZWD***

1.2 Objective

The objective of this work is to develop a visualisation tool that displays weather parameters such as humidity, temperature and pressure as a component of the goGPS software. The displayed information will help better interpret the plots produced by goGPS. The tool will extract weather data from a given source and display them on goGPS plots. As an illustration of its usage, time series of GNSS position estimates will be analysed and interpreted using the added information. As an additional feature, a component will be developed for the export of the extracted weather data in a format that could be used for other purposes.

1.3 Expected Output

To achieve the set objectives, the visualistion tool (Figure 4) has to be displayed on a plot produced by goGPS software. From Figure 4, the line plots are the existing plots produced by goGPS software. The extracted data are displayed as rows containing the identifier and the values for each parameter. The icons symbolise the weather condition (such as rain, snow, clear weather, etc.) for time series of less than 30 hours. For time series of more than 30 hours, patches are instead shown coloured with different face colours and interpreted using the legend on the lower left. The axes labels and titles are the same as the ones shown in the goGPS plot. The tool has to be accessed using a menu item with submenu items as shown in Figure 5.

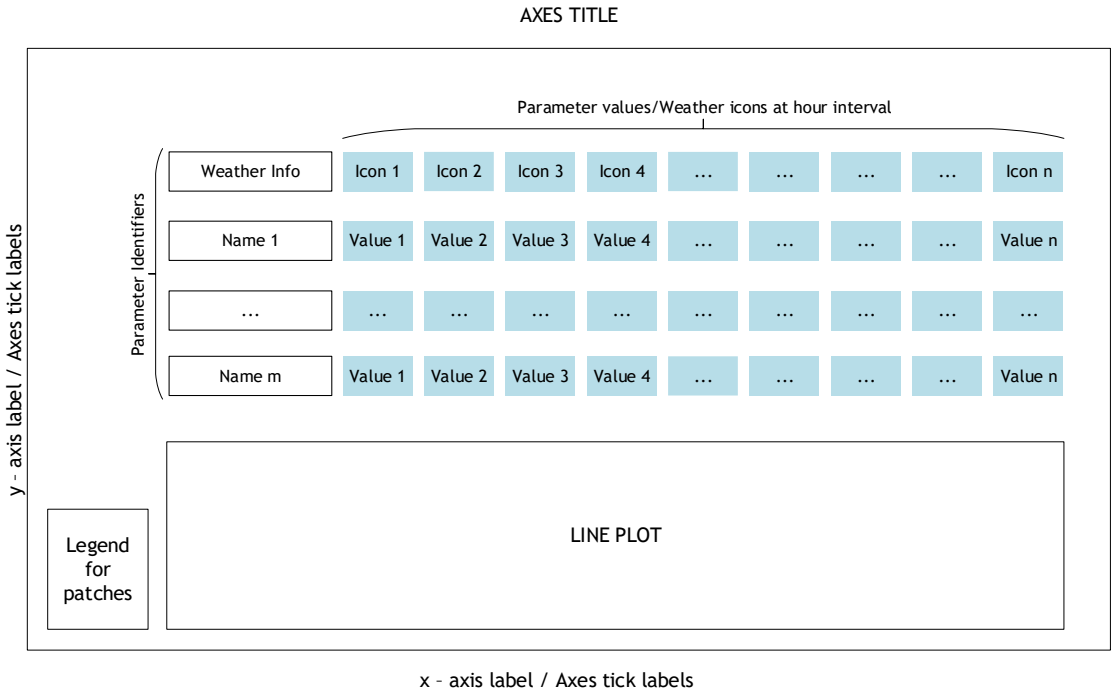


Figure 4 ***Expected output of the visualisation tool***

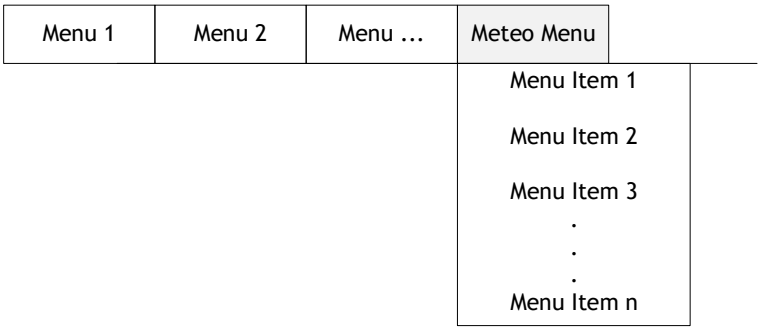


Figure 5 ***Expected menu item and its sub-items***

1.4 Technologies to be Used

- goGPS and MATLAB for the implementation of the tool
- Figma for getting the weather icons and Adobe Photoshop for their redesign
- Microsoft Office (Word, Excel, PowerPoint) for preparation of the reports
- GitHub for the deployment of the developed tool

1.5 Thesis Structure

The organisation of the thesis is carried out as follows.

The first chapter discusses the variability of atmospheric water vapour and the possibility to estimate its amount using GNSS techniques. It also introduces goGPS as a software that can be used to process observed GNSS data and produce plots for further analysis. Also, the objectives and expected results for this work are presented.

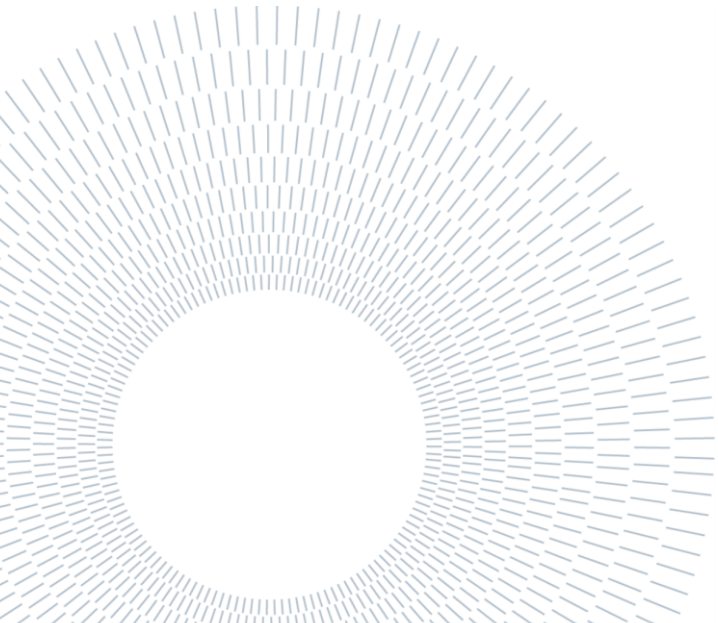
The second chapter gives an overview of atmospheric dynamics and phenomena. These are very useful to better understand the atmospheric dynamics and state that will be displayed on the goGPS plots.

The third chapter talks about how the atmosphere interacts with GNSS signals and the two delays affecting the signals. Also, models of ZWD, ZTD and precipitable water vapour are presented.

The fourth chapter gives a short introduction to goGPS software and Matlab tools that are used for this work.

The fifth chapter gives the procedures that were used for the achievement of the objectives while the sixth chapter gives the results of the work and additional information useful for interpreting the information added to the plots.

Finally, the seventh chapter concludes the work and gives recommendations that are useful for future developments.



CHAPTER 2

ATMOSPHERE

In this chapter, an overview of the atmosphere is given for knowledge purposes. This will be useful

- to describe the characterises of the atmospheric layers traversed by the satellite signals. A fair idea of the components of these layers and how they affect the traversing satellite signals is given,
- to describe different classes of clouds producing the various meteorological events shown on the plots.

2.1 Composition and structure of the atmosphere

The atmosphere is the layer of gases that surround the Earth, retained by the Earth's gravity. Up to about 80 km, the atmosphere consists chiefly of nitrogen and oxygen in almost constant proportions forming a layer known as the homosphere. There are also varying amounts of water vapour and carbon dioxide in the layer, which have a significant impact on weather and climate. In proportions, the atmosphere is composed of about 78 percent nitrogen, 21 percent oxygen, 0.9 percent argon, and 0.1 percent other gases which include water vapour, hydrogen, carbon dioxide, ozone, neon, etc.

The atmosphere is classified by composition of gases into the heterosphere and the homosphere, separated by the turbopause. In accordance with the change of temperature above the surface of the Earth, the atmosphere is divided into exosphere, thermosphere, mesosphere, stratosphere and the troposphere.

The heterosphere has a non-uniform composition of gases, with the heavier molecules lying beneath the lighter ones. It extends from about 80-100 km to the outermost part of the atmosphere. It contains the exosphere and the thermosphere. Beneath the heterosphere is the turbopause, which caps the homosphere beneath. The homosphere is characterised by a uniform mixture of gases.

The exosphere is the outermost layer of the atmosphere that fades into space. Beneath it is the thermosphere, separated by the thermopause.

The thermosphere lies between the thermopause on the upper part and the mesopause on the lower part. It extends from about 85 km to between 500 and 1000 km above the surface of the Earth. The solar radiation from the Sun heats up the air molecules in this layer and causes the dissociation of the molecules. This results in the freeing of electrons, creating many charged ions. Thus, the thermosphere contains most of the ionosphere. The layer is also characterised by low density of air molecules which are thoroughly mixed due to the turbulence in the layer.

Beneath the thermosphere is the mesosphere, bounded on the upper layer by the mesopause and on the lower part by the stratopause. The layer extends from about 50 to 80 km above the Earth's surface. In the mesosphere, the temperature decreases with increasing height as there is less absorption of solar radiation in the region and increasing cooling by CO_2 radiative emissions. The coldest regions of the Earth's atmosphere are found on the upper section of the mesosphere.

From the surface of the Earth, the stratosphere is the second lower layer, lying above the troposphere and separated from it by the tropopause. The stratopause separates the stratosphere from the mesosphere above. The layer extends from about 12 km to 50 km above the surface of the Earth. The stratosphere is the layer of the Earth's atmosphere that has a high concentration of ozone gas. These gases absorb much of the solar ultraviolet radiation coming from the Sun, causing an increase in the temperature of the layer with height. This temperature trend creates an atmosphere of stability such that the layers of air are stable, free from the weather phenomena that characterise the troposphere.

Finally, the troposphere is the last layer and the one lying on top of the Earth's surface. It extends to about 12 km above the surface of the Earth, bounded on the upper part by the tropopause. Radiations from the Sun that reach the surface of the Earth heats up the air molecules in the lower parts of the troposphere, causing a decrease in the temperature of the layer with increasing height. The water bodies on the Earth's surface (rivers, lakes, seas, etc.) humidify the troposphere through

evaporation and thus, much of the water vapour in the atmosphere is located in the troposphere, causing significant changes in the weather conditions in the layer. The troposphere impacts heavily on the GNSS system as the water vapour in this layer causes a significant delay on the travelling satellite signals.

Figure 6 gives a clear picture of the variation of temperature with height for the various layers of the atmosphere (Linacre and Geerts, 1997).

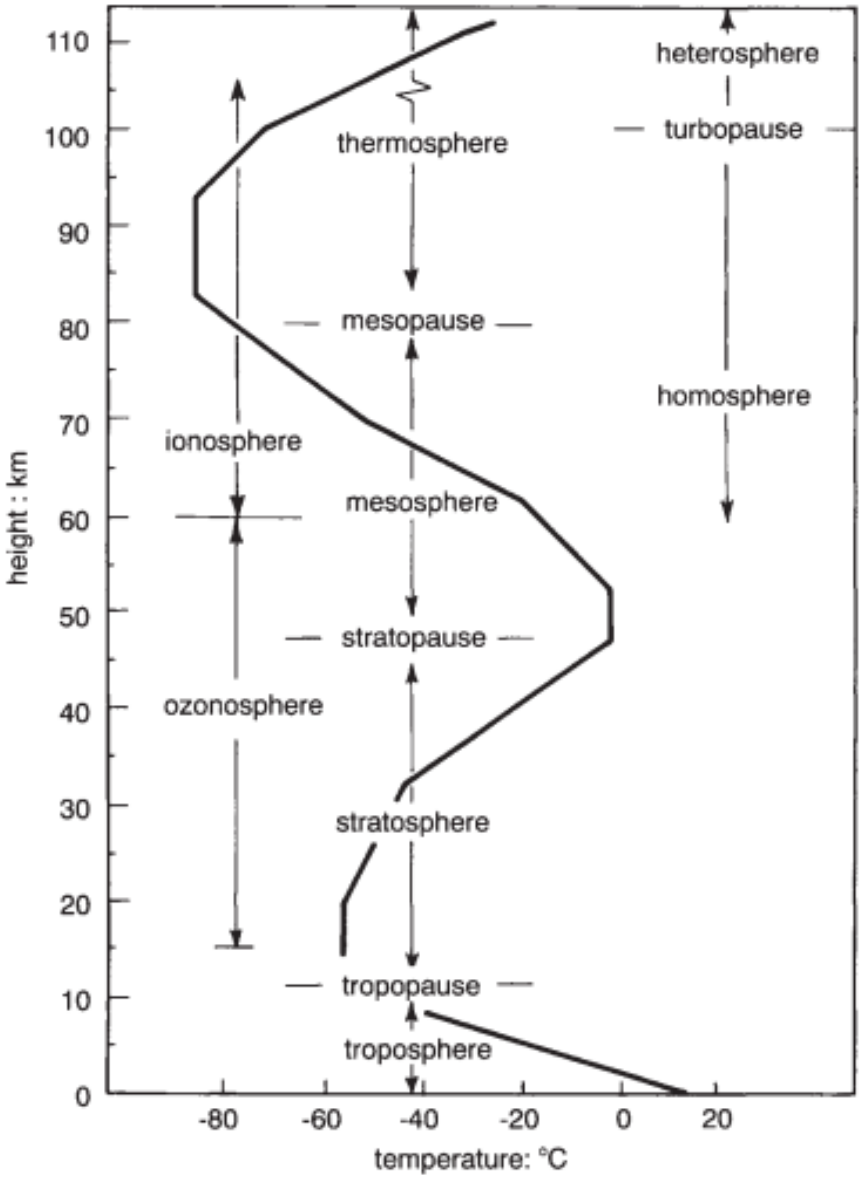


Figure 6 ***Variation of temperature with height of atmosphere***

The atmosphere is essential for life on Earth as it makes available water, CO_2, oxygen and other essential gases for living organisms. It also protects life on the Earth surface by absorbing harmful ultraviolet solar radiations. Conversely, the atmosphere serves as the medium in which most Meteorological phenomena take place. These Meteorological phenomena are usually interrelated and categorised into (Linacre and Geert, 1997)

- those concerning the air's composition and structure such as air mass,
- those involving the Sun's energy such as thunderstorm and lightning,
- processes related to the transformations of water from liquid to vapour, cloud, rain and snow, and
- winds such as dust storms.

2.2 Atmospheric Humidity

2.2.1 Hydrologic cycle

Water vapour contributes greatly to the changing dynamics of the atmosphere. It is a fundamental variable of study in the hydrologic cycle and in meteorology. The hydrologic cycle describes the continuous circulation of water between the Earth and the atmosphere.

Solar radiations cause evaporation of water from the surface of the Earth to the atmosphere. This water vapour then condenses into clouds, fog etc. and falls as precipitates or rain to the Earth (land or ocean). The water that falls on the surface of the Earth is transported to the oceans through surface runoff. Some of the rain percolates into the soil and is discharged into the ocean and other water bodies as groundwater. Figure 7 shows the phenomena of the hydrologic cycle. In this cycle, the atmosphere holds less amount of water vapour, which also has a smaller molecular density compared to oxygen (32 g/mol) and nitrogen (28 g/mol). Thus,

the water vapour exerts less vapour pressure in the air. The atmosphere also holds very little of the world's water.

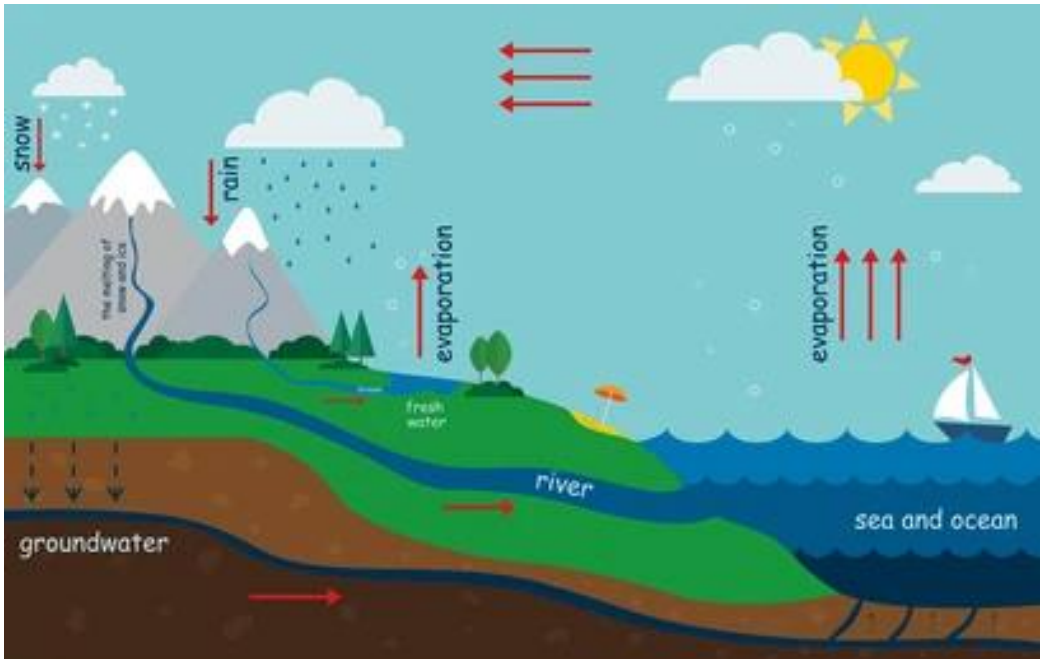


Figure 7 ***Hydrologic cycle***

2.2.2 Humidity of the air

The amount of water vapour present in the atmosphere is defined as humidity. Four indices usually used for determining humidity are absolute humidity, specific humidity, mixing ratio and relative humidity.

Absolute humidity refers to the mass of water vapour per unit volume of moist air without taking into consideration temperature. The higher the amount of water vapour present, the higher the absolute humidity. It is expressed as grams of water vapour per cubic meter of air. **Specific humidity** refers to the mass of water vapour per unit mass of dry air and is measured as grams of water vapour per kilogram of dry air. It represents the actual amount of water vapour present in a unit mass of air. **Mixing ratio** is similar to the specific humidity, the only difference being that it considers mass of moist air instead of dry air. **Relative humidity** is defined as the ratio of the partial vapour pressure of air to the saturated vapour pressure of the air

at a specified temperature. In its simplest terms, it specifies how much of the water holding capacity of the air has been occupied by the water vapour at the specified temperature. At low temperature, the water molecules tend to condense and the water holding capacity of the air decreases. Contrarily, warming the air will cause the water molecules to evaporate and the water holding capacity of the air to increase. Relative humidity is expressed as a percentage and is the variable most commonly used in literature for expressing humidity of the air. It is also the variable being used to describe the humidity of the atmosphere in this work.

As always said, humidity is dependent on temperature. For the same amount of water vapour, the humidity of cold air will be higher than that of warm air. A related parameter for describing the temperature dependence is the dew point. It is the point on a temperature scale to which an air parcel must be cooled to cause saturation. Below the dew point, the air deposits on surfaces as dew or cloud droplets.

2.2.3 Variation of humidity on the surface of the Earth

The amount of water vapour in the atmosphere at any point in time varies from place to place depending on factors such as elevation of the area with respect to the sea level, closeness to a water body, season of year, changes in weather phenomena during the day, latitude of the point of consideration etc.

Across the surface of the Earth, air masses originating from the continental landmasses are usually dry and have low humidity values as compared to those from the oceans, especially during the winter season in which the reduced temperature reduces the saturation vapour pressure of the air. Also, regions of higher latitudes have lower vapour pressure and lower humidity.

In addition, humidity decreases with increasing elevation and height of the atmosphere above the Earth surface. This is because moisture is added to the air mass nearer to the surface, which by convection is lifted upwards. At higher elevation, the temperature of the air reduces and the capacity of the air to hold moisture reduces, decreasing the humidity.

In addition, water vapour content of the atmosphere varies considerably during the day taking into perspective an air layer closer to the land surface. Lower temperatures at night cause condensation of water on cold surfaces and some also absorbed by colder soils. During clear or sunny days, water vapour evaporates from warm surfaces and increases the moisture content of the air. Sea breeze during the day also adds moisture to the atmosphere. Thus, humidity is higher during the day and lower at night hours. Similarly, the moisture content of land areas closer to water bodies are higher than those of inland.

Generally, wind moves air masses from one place to another. In windy hours, surrounding air is mixed with water vapour evaporating from surfaces, thus reducing the amount of water vapour in the atmosphere.

2.2.4 Measuring air humidity

Humidity can be measured using several devices such as psychrometers, electrical sensors and leaf wet sensors. Psychrometers are most commonly used in meteorological stations. The device consists of wet and dry bulb thermometers. The wet bulb is covered with a moist wick. When air is passed over the wick, it evaporates the moisture, causing the bulb to register a lower temperature than the dry bulb.

2.3 Atmospheric pressure

The gravitational force of the Earth pulls the gaseous molecules in the air towards the centre of the Earth. As a result, the gaseous molecules exert an amount of pressure. By measuring the pressure of the atmosphere, it is possible to know how much air there is above the measuring station. A reduction in atmospheric pressure value indicates a loss of air above the column. Atmospheric pressure is usually measured using a barometer made up of a column of mercury in a glass tube.

2.4 Atmospheric temperature

The temperature of the atmosphere is one variable whose variation is significantly felt by living beings. It also has a predominant impact on the changing scenes of the atmosphere and weather phenomena though pressure, water vapour and wind also have impacts. Solar radiation from the Sun is divided into visible, infra-red, ultraviolet and shorter wavelength radiations. Passing through the atmospheric layers, some of the radiations are absorbed by the molecules contained in the atmosphere. For instance, ozone molecules in the stratosphere absorb the ultraviolet radiations. A minimal amount of the solar radiations reaches and warms the Earth surface. Insolation defines the amount of solar radiation that reaches the surface of the Earth. Air masses in the lower troposphere closer to the Earth surface get warmed-up by the radiations reflected from the surface, rise to higher heights in the atmosphere, and undergo several transformations resulting in the changing conditions of weather. Approximately all weather phenomena occur in the troposphere due to the heating of the air mass near the surface of the Earth.

Temperature varies at different heights relative to the Earth's surface as discussed in Section 2.1 and the corresponding profile shown in Figure 6. In the troposphere, where most atmospheric phenomena occur, temperature decreases with increasing height above the Earth surface. This is due to the warming of the air mass near the Earth. Same also applies to higher elevations and mountains. Average temperatures tend to decrease by about 4.2 K per kilometre extra elevation (Linacre and Geerts, 1997).

Water has an especially high heat capacity at 4.18 $Jg^{-1}°C^{-1}$, while for land, it is usually less than 1 $Jg^{-1}°C^{-1}$ (Anon., 2022). This means that it takes more heat to warm a gram of water than land. As a result, the ocean responds very slowly to temperature changes than the land. Hence, the land has a higher temperature than the ocean. Thus, air masses over the land surface will be much warmer than air masses over the oceans.

Latitude of a station on the Earth's surface is also another factor that causes variation in temperature. Around the equator (low latitudes), the solar insolation is higher than the thermal radiation resulting in considerable net heat gain. The polar

regions (higher latitude), are characterised by a higher rate of thermal radiation than insolation, hence, more heat losses. These impacts may be nullified by the wind and ocean currents that carry air masses from the areas of higher temperatures to the areas of lower temperatures.

The variation of temperature during the 24-hour period of a day is known as diurnal temperature variation. The maximum temperatures usually occur after noon as the air still keeps hold of some of the solar radiations absorbed at noon. The lowest temperatures occur at night, usually after dawn. This occurs as the Earth surface undergoes radiative cooling processes, especially when the night is clear and heat can escape through the atmosphere. The air above however gets warmer than the surface resulting in an inversion of temperature.

In the same way, there are seasonal contrasts in temperature. During summer, the air above land has a higher temperature than the oceans. In winter, the air above oceans gets higher temperature than landmasses.

2.4.1 Measurement of atmospheric temperature

Temperature is usually measured using a thermometer. At a meteorological station, the standard height for the thermometer above the ground has to be between 1.25 and 2 m according to the World Meteorological Organisation and protected with the Stevenson screen. There is no single thermometer measuring the global temperature, however, individual measurements taken every day at several thousand stations over the land areas of the world are combined with thousands more measurements of sea surface temperature taken from ships moving over the oceans to produce an estimate of global average temperature every month (Trenberth et al., 2007). Such is the data source of DarkSKY (see Section 5.3) upon which this work is being carried out.

2.5 Clouds, categories, formation and processes

The data source being used for this work, DarSKY, classifies the state of the atmosphere in five major groupings: clear, cloudy, rain, sleet and snow. These are formed from clouds as a result of the changing conditions of the atmosphere due to the heating of the Earth surface by the Sun. The ultraviolet radiation of the Sun reaches the Earth's surface unevenly creating variations in air pressure. Low air pressure causes rising air that is lighter than the surrounding air masses. Rising air makes the water vapour in the air condense and form clouds, leading to rain, thunderstorms and hurricanes. On the other hand, high air pressure causes heavy and sinking air that makes the environment unstable. High air pressure is usually related to clear skies and sunshine. The condition of the sky at any point in time is determined by the predominance of a state (such as mostly cloudy, overcast, and cloudy) over other conditions of the sky.

Clouds are made up of tiny droplets of water and ice crystals suspended in the atmosphere. Clouds play an essential role in the hydrologic cycle of the planet Earth. They reflect some of the incident solar radiation into space and also absorb infrared radiations reflected from the Earth surface. On the other hand, clouds have a significant impact on the changing weather conditions.

Cloud droplets have a diameter of about 10 – 15 microns (1 micron = 1/1000 mm), each cubic metre of air will contain about 100 million droplets (Anon., 2022) and can remain in liquid form in temperatures of -30°C.

Clouds are generally formed when molecules of water vapour in the air condense into water droplets or ice crystals. It is essential that the air be saturated to a point that it is unable to hold any more water vapour. Also, there has to be availability of cloud condensation nuclei such as dust, clay and soot, to provide a surface on which the water molecules will condense. There are two basic ways in which saturation can be reached (Anon., 2021)

- By increasing the water vapour content of the air to a point such that the air is unable to hold any more water. This can be realised when the Sun heats up the

ground, which in turn heats the air in contact with it, causing the air to rise to higher atmospheres.

- By cooling the air to its dew point so that the air is unable to hold any more water vapour, making it favourable for condensation to occur. This cooling is realised
  - when a mass of warm air moves over a mass of cold air creating a frontal boundary
  - when the air mixes with colder air
  - through interaction of the air with mountains or undulated topography of an area
  - by nocturnal radiation loss

2.5.1 Categories of Clouds

There are various clouds identified by their shape and other attributes, however only ten are recognised according to the World Meteorological Organisation standards. These 10 basic classes of clouds, also referred to as genera, describe the height where they are formed and their appearances. These classes are listed in Table 1.

Table 1 ***Classification of Clouds***

Cloud Level	**Altitude (Km)**	**Class According to Shape**
High	6-10	Cirrus, Cirrocumulus, Cirrostratus

Medium	3-6	Altocumulus, Altostratus, Nimbostratus
Low	Less than 3	Stratocumulus, Stratus, Cumulus
towering vertical	0.6-6	Cumulonimbus

These genera are further divided into secondary classes of 14 species, which are further divided into 9 varieties of tertiary classes. The species describe the internal structure of the cloud while the varieties describe the transparency and arrangement of the clouds. These categories are as shown in Table 2.

Table 2 ***Categories of Clouds***

Classification	**Categories**
Species	calvus, capillatus, castellanus, congestus, fibratus, floccus, fractus, humilis, lenticularis, mediocris, nebulosus, spissatus, stratiformis, uncinus
variety	cumulogenitus, duplicatus, intortus, lacunosus, opacus, perlucidus, radiatus, translucidus, undulates, vertebratus,

In the following sections, the primary classification of clouds based on height of formation and its appearance will be discussed.

Cirrus

This is the main type of high cloud occurring in the upper troposphere. They appear as tufts and whitish and sometimes greyish in colour (Figure 8). It is made up chiefly of ice crystals that form because of the low temperatures at the top of the troposphere. On falling, these ice crystals evaporate on warmer layers and usually do not fall to the ground. They also give precipitation when they join together and form thicker components. They can be an indication of rain.

Figure 8 ***Cirrus***

Cirrocumulus

Cirrocumulus (Figure 9) is a layer of cloud made up of small elements in the form of ripples that form as a result of the transformation of cirrus and cirrostratus. They are whitish but sometimes appear greyish. They are smaller than the width of the littlest finger when one holds up the hand at arm's length. It consists, chiefly, of water vapour. Cirrocumulus clouds indicate an unstable atmosphere and lead to heavy showers.

Figure 9 ***Cirrocumulus***

Cirrostratus

Cirrostratus clouds (Figure 10) are high layer clouds with no variation in tone from one part to the other. They are formed at the forefront of the frontal weather system and also through the trails left by an air plane as it flies through the atmosphere. Cirrostratus clouds, which arise after cirrus and spread from one location over the sky, can occasionally indicate the arrival of a warm front, and so may be signals of precipitation in the next 12 to 24 hours.

Figure 10 ***Cirrostratus***

Altocumulus

This is a mid-level layer cloud that looks like it is made of regular cotton-wool balls. Altocumulus clouds (Figure 11) are comparatively larger in size than the cirrocumulus clouds and they give an indication of storm or development of thunderstorms later in the day.

Figure 11 ***Altocumulus***

Altostratus

Altostratus (Figure12) appears as a greyish or bluish layer of fibrous or striated cloud that often blurs the Sun or makes it appear as if one is looking through a frosted glass. They can be formed when a layer of cirrostratus descends from the higher atmospheres. Altostratus clouds do not usually produce rain but may thicken with progressive lowering of the base to form nimbostratus which give an indication of rain.

Figure 12 ***Altocumulus***

Nimbostratus

Nimbostratus (Figure 13) is a layer of low-level greyish cloud that precipitates rain reaching the ground. They form from the thickening of altostratus and are also thick enough to obscure the Sun from view.

Figure 13 ***Nimbostratus***

Stratocumulus

Stratocumulus (Figure 14) are low-level patches of clouds that vary in colour between white and grey. They might have gaps between them or joined together.

They are formed from the spreading out of cumulus clouds. Stratocumulus clouds do not often produce precipitation and when they do, they give out light rain or snow.

Figure 14 ***Stratocumulus***

Stratus

These are low-level clouds with greyish or whitish colour (Figure 15). The clouds form under stable atmospheric conditions when gentle breezes raise moist air over a cold surface. They can sometimes be very close to the ground in the form of fog or mist. Stratus cloud produces little to no rain and if it is thick enough, it can drizzle.

Figure 15 ***Stratus***

Cumulus

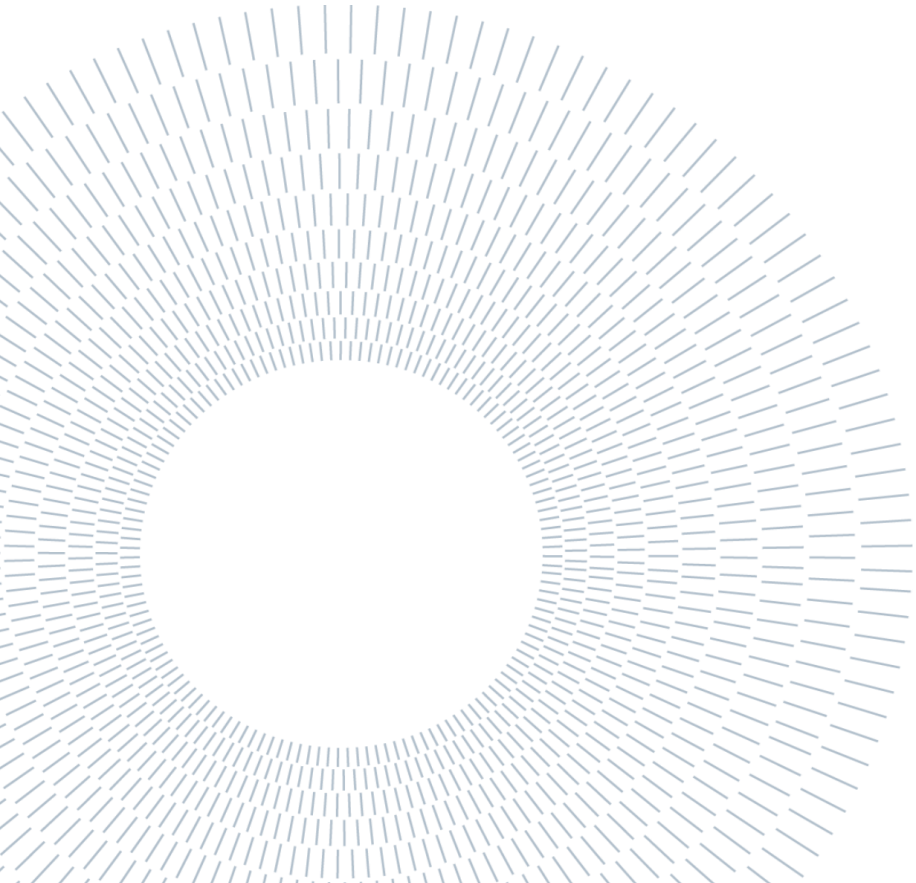
Cumulus clouds (Figure 16) are detached, cauliflower-shaped clouds that usually appear in fair weather conditions during the day. They usually form a few hours after daybreak and scatter before the Sun goes down. Cumulus clouds produce no rain or snow and lazily drift across the sky on a sunny day.

Figure 16 ***Cumulus***

Cumulonimbus

Cumulonimbus clouds (Figure 17) are dense and heavy clouds that extend into the atmosphere in plumes. The upper part of it is fibrous and spreads out in the shape of anvil. The base however is very dark with small clouds hanging with it or attached to it. They are formed through convection from small cumulus clouds. Cumulonimbus clouds are often associated with heavy downpours, hailstorms, lighting, thunder and even tornadoes.

Figure 17 ***Cumulonimbus***



CHAPTER 3

INTERACTION OF ATMOSPHERE WITH GNSS SIGNALS

The plots produced by goGPS software are based on the results of processing GNSS data. A summary of the techniques used by the goGPS software for processing data is given in Section 4.1. It will be thus essential to give a background to the plots in this section.

GNSS, abbreviation for Global Navigation Satellite System, allows for determination of the position of a point or station on the surface of the Earth by observing the electromagnetic signals emitted by space orbiting satellites. The signals are transmitted in the microwave band of the electromagnetic spectrum to the receivers. GNSS finds application in positioning, location-based services, transportation, scientific research activities, etc.

GNSS involves a number of satellite constellations that communicate with devices on the surface of the Earth for positioning purposes. Among these constellations are the NAVSTAR Global Positioning System (GPS) operated by the US government, the Russian GLONASS, the European Galileo and the China's Beidou that operate on a global scale (Figure 18) and the Japanese QZSS and the Indian IRNSS that offer services only in the Asia-Pacific region. Other systems such as the European Geostationary Navigation Overlay Service (EGNOS) offer similar services but on a local scale using geostationary satellites, improving upon the performance of GNSS. Though these constellations work independently, with the introduction of multi-constellation GNSS by the International GNSS Service, in the future, users will be able to use these constellations as one single system of GNSS to benefit from the enhancements contributed by the individual systems.

3.1 Delay on GNSS Signals

GNSS point positioning involves measurement of the signals emitted by GNSS satellites for the determination of the position of a receiver on the surface of the

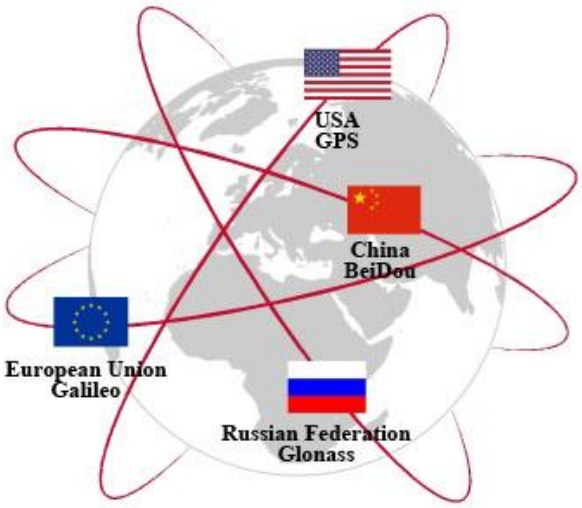


Figure 18 ***GNSS Systems that Operating on a Global Scale***

Earth. The emitted signals contain information about the clock, ephemerides and integrity of the satellite. For the purpose of understanding GNSS point positioning services, the fundamentals of GPS (Biagi, 2009) book is recommended. In this section, attention will be paid to the interaction of the satellite signals with the atmosphere on its travel from space to the Earth surface as this is the main factor used for GNSS meteorology and this work seeks to relate atmospheric parameters with GNSS data processed and plotted using the goGPS software.

In the absence of the atmosphere, the satellite signals would travel to the Earth surface at the constant speed of light of 299792458 m/s. However, the presence of charged electrons in the ionosphere and water vapour in the troposphere affects the signals by reducing their propagation speed and by bending their path. This results in propagation delays, which are respectively known as ionospheric and tropospheric delays.

3.1.1 Ionospheric Delay

The ionospheric delay is caused by the presence of free and charged electrons in the ionosphere (about 100 to 1000 km above the surface of the Earth). For a double frequency receiver, a proper combination of observations on both L1 and L2

frequencies of the signals can be used to get rid of the delay. For a single frequency receiver, there exist models, such as the Klobuchar model, for evaluating and removing the ionospheric effect. These models exploit specific correction parameters contained in the signal emitted by the satellite. Figure 19 shows a sample variation of ionospheric delay across the globe, for midday-GMT and 90° elevation of receiver position with respect to the satellite vehicle, with regions of higher ionospheric error shaded red and regions of lower ionospheric effects shaded blue. Figure 19 was produced using the standard ionospheric error correction model for a single frequency receiver specified by the IGS ionosphere working group. For details on the algorithms used to correct the delay due to ionospheric effects, for both single and dual frequency receivers, it is recommended to read IS-GPS-200L sections 20.3.3.5.2.5 for the single frequency receivers and 20.3.3.3.3.3 and 30.3.3.3.1.1.2 for the dual frequency receivers.

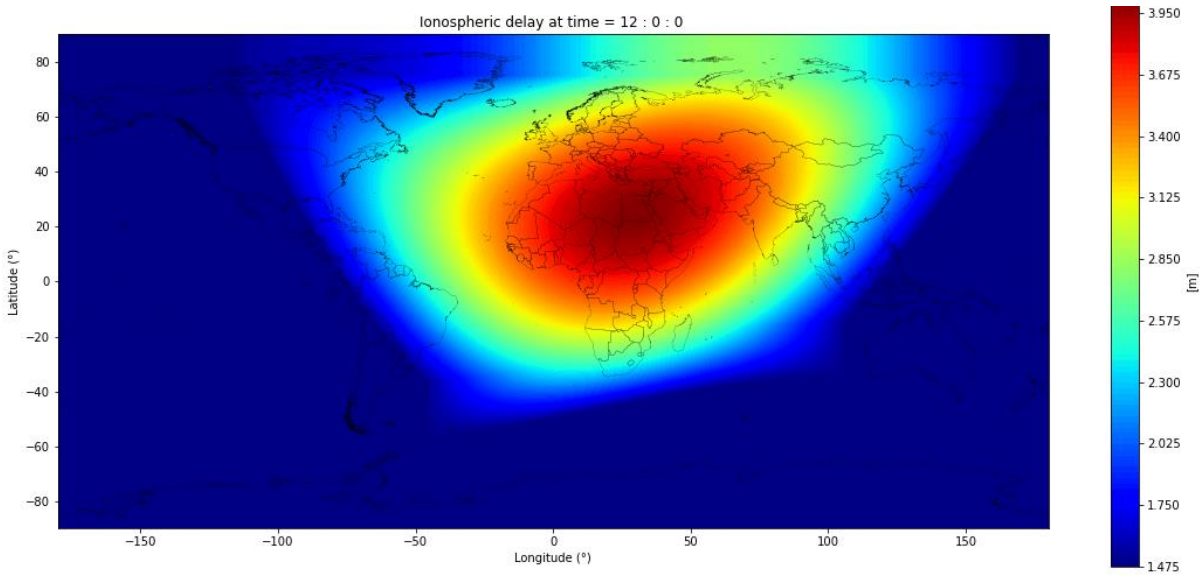


Figure 19 ***Variation of Ionospheric Effect (Global Map)***

3.1.2 Tropospheric Delay

This delay is caused by the presence of water vapour and other constituents in the atmosphere. The total tropospheric delay on the satellite signal is the sum of a wet delay caused by the dipole components of water vapour and a hydrostatic delay caused by dry air constituents and non-dipole components of water vapour. Water

vapour is a key element in the hydrologic cycle and its variation in the atmosphere plays a significant role in climate change and change of weather patterns. Also, water vapour is unique in the atmospheric components because it is the only constituent which possesses a dipole moment contribution to its refractivity (Bevis et al., 1992). The water vapour content of the atmosphere can be expressed as the height of an equivalent column of liquid water, known as the precipitable water vapour (PWV). By using a combination of ancillary measurement of surface pressure and temperature and the tropospheric delay exerted on the GNSS satellite signals, it is possible to derive the amount of precipitable water vapour in the atmosphere from the estimated tropospheric delay.

Conventional techniques for measuring atmospheric water vapour such as radiosondes are of less advantage with respect to GNSS due to low frequency of observation periods. Radiosondes for instance make observations once or twice a day (Smith et al., 2008). Ground-based GNSS stations are capable of providing continuous estimates of vertically integrated water vapour content over a global distribution of land-based locations, limited only by the oceans. On the other hand, though space-borne GNSS receivers can provide observations that are discontinuous with respect to time and geographical location, it can also provide a good coverage of the oceans as well as land.

3.2 Modelling tropospheric delay

The interaction of the satellite signals with the atmosphere is determined by the refractive index of the atmosphere, (Warren, 2019). Most space-based geodetic systems operate in one of two distinct frequency bands, either the microwave band (e.g. GPS) or in the optical band (e.g. Satellite laser ranging) in which the refractive index is dispersive in the ionosphere and non-dispersive in the other constituents of the atmosphere (Herring, 1992). GNSS signals are also able to efficiently excite the dipole component of the water vapour refractivity and therefore there is a large contribution to the refractive index from water vapour in the atmosphere. This causes a satellite signal passing through the atmosphere to encounter variations on

its path, causing the signal to become curved and travelling a longer distance with time than it would travel in a vacuum. The delay on the signal (Equation 1) is thus given by the difference in paths between travel in the atmosphere and travel in the vacuum, (Herring, 1992).

$$L_a = \int_{atm} n(s) \times ds - \int_{vac} ds \qquad 1$$

where L_a is the atmospheric delay correction,

$n(s)$ is the refractive index along the path followed by the ray as a function of position *s*,

the first integral is by definition the electromagnetic distance covered by the signal in the atmosphere, and

the second integral is the length of the path covered by the signal in vacuum.

Let S, defined in Equation 2, be the geometric length of the path followed by the signal.

$$S = \int_{atm} ds \qquad 2$$

By introducing this geometric length into Equation 1, we get the atmospheric delay to be, (Mendes, 1999):

$$L_a = \int_{atm} n(s) \times ds - \int_{atm} ds + \int_{atm} ds - \int_{vac} ds$$

By regrouping, the delay becomes

$$L_a = \left(\int_{atm} (n(s) - 1) \times ds \right) + \left(\int_{atm} ds - \int_{vac} ds \right) \qquad 3$$

The first term on the right of Equation 3 is known as the excess path delay (due to the signal travelling at a lower speed with respect to that of light in vacuum) and the second term is known as the geometric delay due to the traverse along the curved path. Considering a signal that travels in the zenith direction above a

receiver, the curvature path becomes negligible and the atmospheric delay thus becomes:

$$L_a = \int_{atm} \left(n(s) - 1\right) \times ds \qquad 4$$

The tropospheric delay in the zenith direction (90° elevation with respect to the space vehicle) above the receiver station is known as the Zenith Tropospheric or Total Delay (ZTD). The ZTD is nearly proportional to the amount of precipitable water vapour above a GNSS receiver station (Bevis et al., 1992). The precipitable water vapour gives the height of an equivalent column of atmospheric water vapour overlying a given point on the Earth's surface, making GNSS estimate of ZTD a tool for the remote sensing of the atmospheric water vapour. The delay for the other signals which travel in the slant directions different from the zenith direction and excluding potential multipath sources of signals is known as the slant total delay (STD).

Without considering variations in the composition of dry air and with the assumption of laboratory conditions of CO_2 free air, the refractive index, *n*, is related to the surface pressure, temperature and humidity as in Equation 5, (Thayer, 1974). The first two terms in the expressions on the right are the results of induced molecular polarisation of air and water vapour molecules respectively and the third term represents the effects of permanent dipole moment of water vapour molecules.

$$\left(n-1\right)\times 10^6 = N = K_1\left(\frac{P_a}{T}\right)\times Z_a^{-1} + K_2\left(\frac{e}{T}\right)\times Z_w^{-1} + K_3\left(\frac{e}{T^2}\right)\times Z_w^{-1} \qquad 5$$

where N is the refractivity of the atmosphere, P_a and e are the partial pressures of dry air and water vapour respectively in *mbar*, T is the absolute temperature in *Kelvin*, and Z_a^{-1} and Z_w^{-1} are the inverse compressibility factors (corrections for non-ideal gas behaviour) for dry air and water vapour respectively. Using the expressions for inverse compressibility given by Owens (1967), Thayer reorganises the inverse compressibilities as

$$Z_a^{-1} = 1 + P_d\left[57.9\times 10^{-8}\left(1 + \frac{0.52}{T}\right) - 9.4611\times 10^{-4}\times\frac{t}{T^2}\right] \qquad 6$$

$$Z_w^{-1} = 1 + 1650\left(\frac{e}{T^3}\right)\left(1 - 0.01317t + 1.75\times10^{-4}t^2 + 1.44\times10^{-6}t^3\right) \quad 7$$

Where P_a and e are the partial pressures of dry air and water vapour respectively in *mbar*, T is the absolute temperature in *Kelvin*, and t is the temperature of the air in degrees Celsius. There have been several determinations of the refractivity constants (K_1, K_2 and K_3) in Equation 5 in literature. However, the one empirically determined by Thayer is suggested for use by Davis et al. (1985). The values of the constants and standard deviations are given below.

$$K_1 = 77.6036 \pm 0.014 K/mbar$$
$$K_2 = 64.79 \pm 0.08 K/mbar$$
$$K_3 = (3.776 \pm 0.004)\times105 K^2/mbar$$

The first term on the right of Equation 5 does not depend on the water content of the atmosphere and is known as the hydrostatic component (N_d) of the atmospheric refractivity. The second and third terms are water dependent and known as the wet component (N_w) of the refractivity. The refractivity of the atmosphere is then expressed as

$$N = N_d + N_w \quad 8$$

By substituting Equation 8 into Equation 4 and simplifying, the zenith delay becomes

$$L_a = \int_{atm}\left(\frac{N}{10^6}\right)\times ds = 10^{-6}\times\int_{atm}(N)\times ds \quad 9$$

Consequently, by rewriting Equation 9 in terms of the hydrostatic and wet part of the atmospheric refractivity, the zenith delay becomes

$$L_a = 10^{-6}\int_{atm}(N_d)ds + 10^{-6}\int_{atm}(N_w)ds \quad 10$$

where the first term on the right is the zenith hydrostatic delay (ZHD) and the second term is the zenith wet delay (ZWD). Thus,

$$ZTD = ZHD + ZWD \quad (11)$$

Given measured values of meteorological parameters of pressure, humidity and temperature, the zenith hydrostatic delay can be modelled using for instance the Saastamoinen model (Teke *et al*. (2011), Bevis *et al*. (1992), Rocken *et al*. (1993), Saastamoinen (1973)). Although some effort has been made to develop models that can be used to predict the zenith wet delay from surface measurements, their predictive value is very poor compared to that of the zenith hydrostatic delay. This is because the zenith wet delay relies on the amount of water vapour in the atmosphere which varies rapidly in time and space. Two approaches can be used in high accuracy GPS analysis to estimate the zenith wet delay; a least squares estimation of one parameter per station per specified time interval and a stochastic estimation process using a Kalman filter (Rocken et al., 1993). In both methods, the estimation is based on the assumption that the atmosphere above a GPS antenna is azimuthally isotropic and that the slant wet delay is azimuthally related to the zenith wet delay by a mapping function. The second method involving the use of the Kalman filter process was initially used in goGPS software. However, recent developments see the use of the least squares approach. A fast technique for deriving the ZWD, also used in goGPS, is by estimating the ZTD from the observation data, the ZHD using the Saastamoinen model and deriving the ZWD from the difference between the two:

$$ZWD = ZTD - ZHD \quad (12)$$

Also, it is possible to derive an approximate value of the integrated water vapour from the zenith wet delay using the expression

$$IWV = \int k \times \Delta L \quad (13)$$

Where ΔL is the zenith wet delay and k is a constant (see Bevis et al. (1992) for the definition of k). The precipitable water vapour of the atmosphere is then derived from the product of the density of water and the integrated water vapour.

3.3 Atmospheric delay mapping functions

In most cases, GNSS observation signals from a satellite rarely come from the zenith direction above a receiver, with the space vehicle mostly being positioned at an angle different from the zenith one. The tropospheric delay then becomes a slant delay instead of zenith delay. The slant delay can be expressed as the product of the corresponding delay in the zenith direction and a mapping function which models the dependence of the tropospheric delay on the elevation angle of the satellite with respect to the user. The tropospheric delay can also be written as

$$La = \Delta L_{ZHD} m_{ZHD}(\theta) + \Delta L_{ZWD} m_{ZWD}(\theta) \quad 14$$

Where La is the tropospheric delay of the atmosphere,

ΔL_{ZHD} and ΔL_{ZWD} are the zenith hydrostatic and wet delays

$m_{ZHD}(\theta)$ and $m_{ZWD}(\theta)$ are the hydrostatic and wet mapping functions respectively, and

θ is the elevation angle of the satellite with respect to the receiver station.

Most of the existing mapping functions used in modelling the elevation angle dependence or the neutral atmosphere describes the atmosphere by the surface pressure, temperature, relative humidity, temperature lapse rate in the troposphere and the height of the tropopause (Niell, 1996). However, these mapping functions differ in the number of meteorological parameters that are incorporated. For instance, Chao's mapping function makes no reference to meteorological conditions whereas the one derived by Davis et al. (1985) incorporates surface pressure, temperature, relative humidity, height of the tropopause and the temperature lapse rate of the troposphere.

It is advisable to set a threshold for the elevation angle that will be used with the mapping function to avoid the probable introduction of systematic errors. These errors are mostly larger for lower elevation angles. A useful method to detect the presence of systematic errors from the mapping function is the use of elevation cut-

off test described by Davis et al., (1985). In the test, observations of all elevation angles are used for a baseline. The baselines are then re-estimated using a threshold for the elevation angle. If the mean of the difference between corresponding baseline lengths does not tend to zero, then, there are errors coming from the mapping function.

CHAPTER 4

goGPS, SOFTWARE DEVELOPMENT

4.1 goGPS

goGPS (logo in Figure 20) is a software developed by the Geomatics Research and Development (GReD) s.r.l., a spin-off of Politecnico di Milano, using the Matlab programming language. It provides a graphical user interface for the processing of GNSS data.

Figure 20 ***goGPS logo***

Figure 21 ***MATLAB logo***

Its development started in 2007 as a set of routines for processing GPS data based on the Kalman Filter algorithm. Due to the limiting performance factor of the algorithm, especially when used in the MATLAB environment, recent developments have used two batch least squares undifferenced engines for processing. The first engine uses a combination of the observables (such as iono-free observations) for computing solutions for only Precise Point positioning (PPP). The second engine does not perform any combination of the observables and uses all the frequencies and tracks observed for computing PPP solutions, adjusting baselines of networks with multiple receivers.

In the initial developments, there was support for only GPS single frequency low cost receivers, but the current version provides support for all other GNSS observations (including single and double frequency), except kinematic observations.

The first release of the software, was in 2009 under the GPLv3 licence. The current version of the Open Source software is still under development, with new features and functionalities being added to improve the performance and appearance of the software. Also, the current version (1.0) is being managed and maintained by GReD s.r.l., providing support for the open source community.

An executable version of the software is yet to be released. However, performing any functionality with the software can be done by running it on the MATLAB platform.

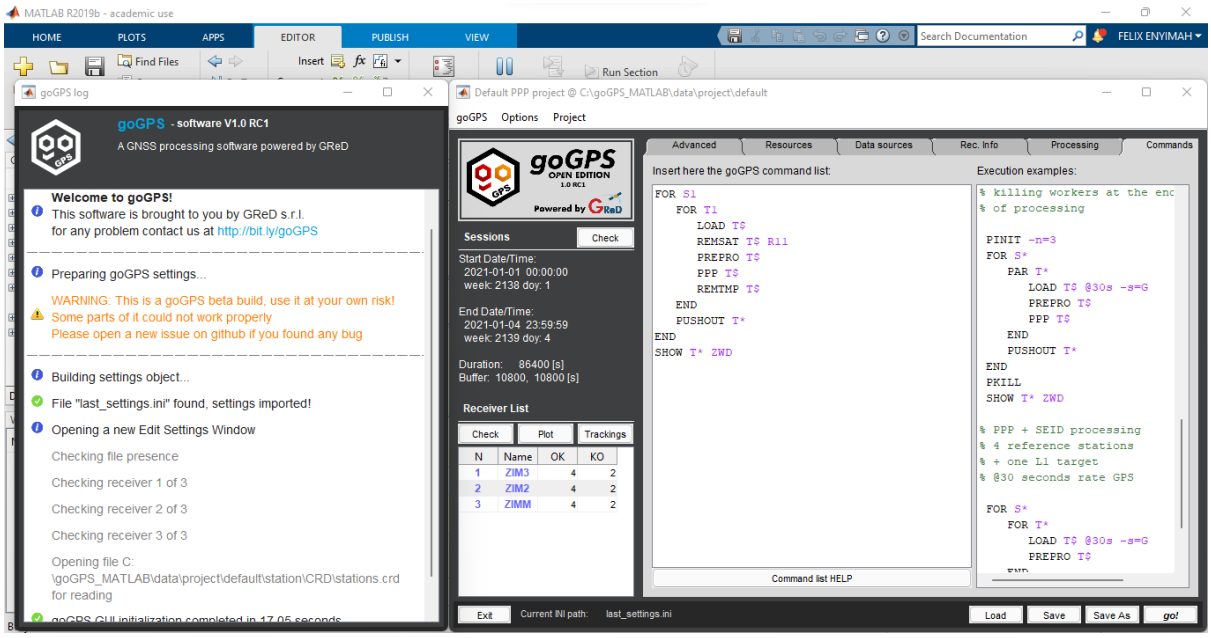


Figure 22 ***goGPS Interface***

4.2 Graphical Objects in MATLAB

4.2.1 Graphical Objects

In MATLAB, the outputs of a plot are directed to a window, known as a figure, separate from the command window. Default line styles and colours are then used to differentiate plots of data from other components of the figure such as menu bar and toolbar. However, it is possible to change the appearance of the plotted graph and add or remove annotations such as texts, lines, surfaces and patches. This

flexibility serves behind this work. Thus, this section discusses the graphical objects being used, their properties and accessibility. This is made possible by the available graphical objects and functions provided by MATLAB. These graphical objects are being exploited for this work to provide added information to the plots produced by goGPS.

They are organised in a hierarchical structure (Figure 23), each instance of the object being associated with a handle for its identification. There is a parent-child relationship existing between the objects, with those in the lower rank of the hierarchy being the descendants of those at the top of the hierarchy and it is essential for the existence of a parent object before a child object can be created. For instance, line objects are core graphical objects. To create a line, there is a need for an axes object to exist (because a line object is a descendant of an axes object), which also needs a figure. To create a child object, if the parent does not exist, MATLAB automatically creates the parent in order for the child to be created. Also, if parent(s) exist(s), MATLAB sets the most recent one as the parent object for the child. However, one can also specify the parent for a child object at creation time. The root object sits at the top of the hierarchy and is created at MATLAB startup by default. Thus, one does not need to create the root object. All other objects that are descendants of the root object however need to be instantiated before being used. For further description of these, it is recommended to read the documentation on ‘Using MATLAB Graphics’. However, brief descriptions of the relevant ones for this work are reported.

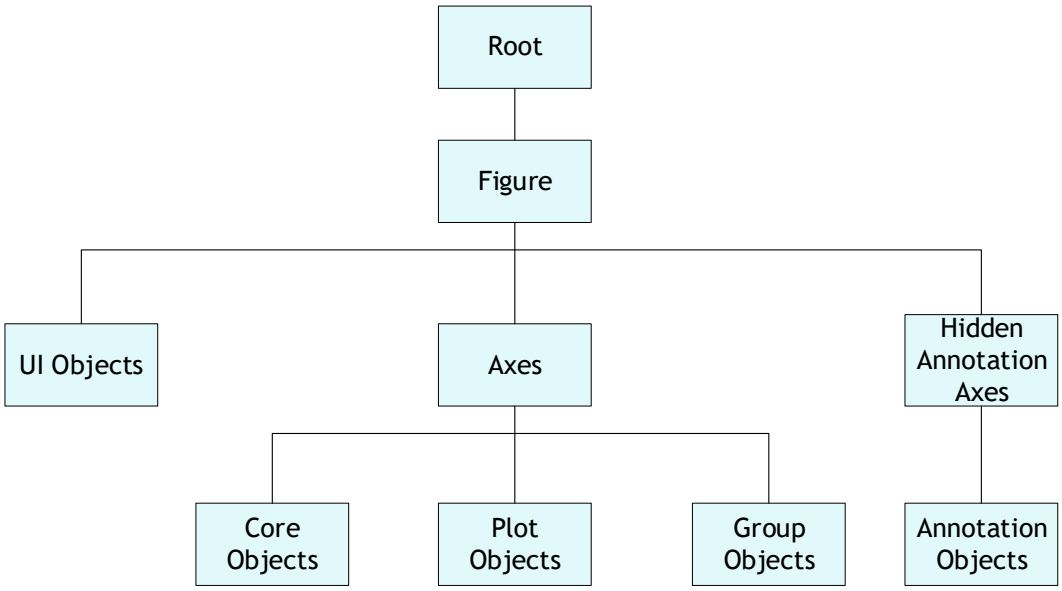


Figure 23 ***Hierarchy of graphical objects in MATLAB***

Graphical objects are displayed in a figure window containing menus, toolbars, user-interface objects, axes and its children, etc. (Figure 24).

Axes objects define the coordinate system for displaying graphs.

Core graphics objects include basic drawing primitives like line, text, patch, surface, images and light objects, which are not visible but affect the way some objects are coloured. These objects can be defined using the functions given in Table 3.

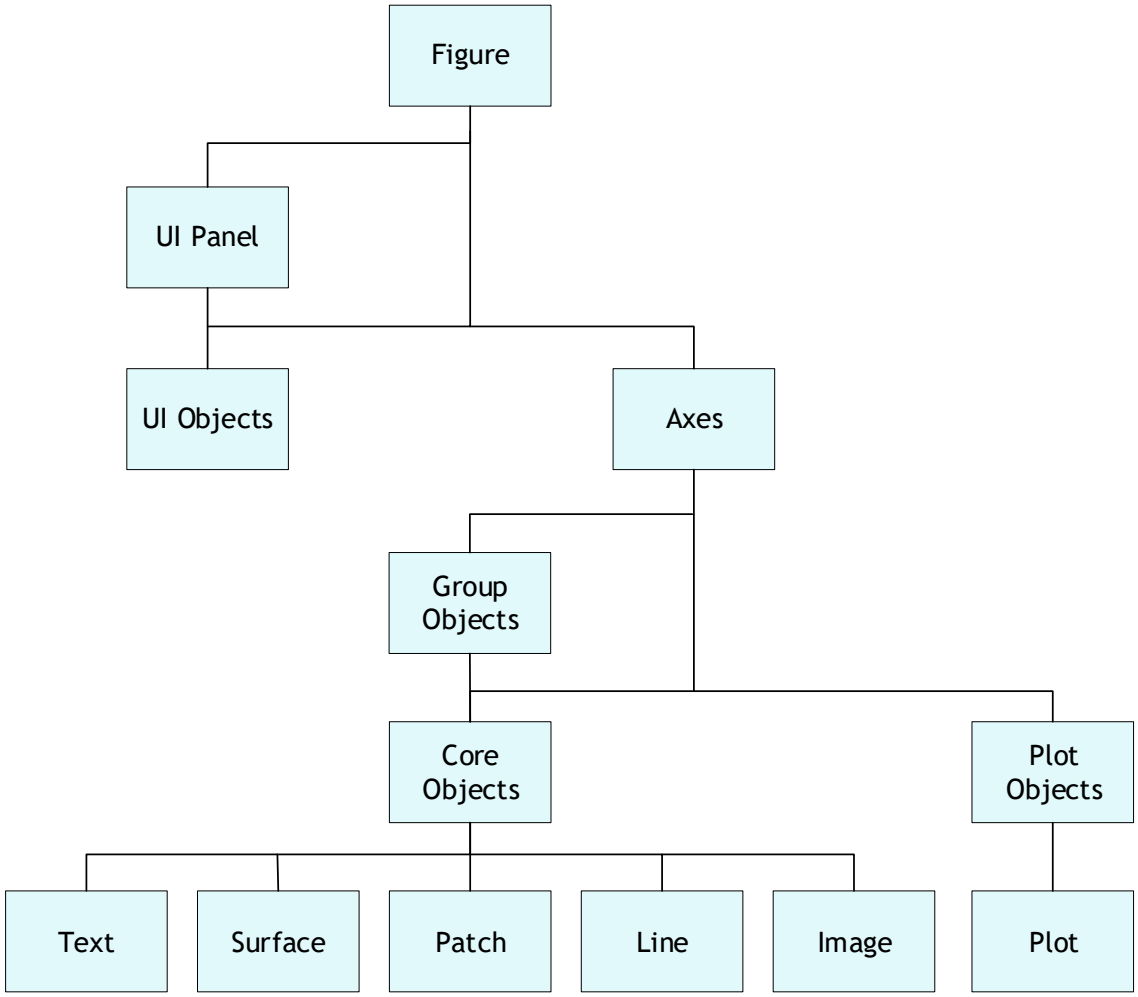


Figure 24 ***Descendants of a figure object***

Table 3 ***Graphical Object Functions***

Function	**Purpose**
surface	3-D grid of quadrilaterals created by plotting the value of each element in a matrix as a height above the *x*-*y* plane.
patch	Filled polygons with separate edge properties. A single patch can contain multiple faces, each coloured independently with solid or interpolated colours.

text	Character strings positioned in the axes.

4.2.2 Accessing Properties of MATLAB Graphical Objects

The properties of a graphical object control its appearance and behaviour after it has been instantiated. These properties include general information of the object such as the object's type, its parents and children (if available), whether it is visible and other relevant information peculiar to the particular object. The '*get*' function of MATLAB can be used to query the value(s) of a property of a graphical object. The basic syntax for setting the property of an object is given by:

returnedValue = get(objectHandle, '*PropertyName*')

where objectHandle refers to the identifier for the object being queried and PropertyName refers to the property of the object whose value is needed.

In addition, the '*set*' function available in MATLAB can also be used to change the value of the property of an object. The basic syntax for using the set method is:

set(objectHandle, '*PropertyName*', '*NewPropertyValue*')

The objectHandle and PropertyName arguments of the *'set'* function are as defined for the *'get'* function. The *'NewPropertyValue'* is the value being set for the property of the object. In an instance where the handle of an object has not been specified at creation time, it is possible to use the *'findobj'* method to query the particular object in the hierarchy having a particular property set to a particular value using the syntax:

findobj('*PropertyName*', '*PropertyValue*')

Alternatively, the *'findall'* method can be used to find an object in case its '*HandleVisibility*' property is set to off.

Also, an object and all its descendants can be removed from a graph using the '*delete*' method available in MATLAB.

4.2.3 Dealing With Images in MATLAB

In MATLAB, Images are represented as an array of data structure, thus working with images is comparable to working with any other data. Most images are stored as a two-dimensional array in which each element of the matrix corresponds to a single pixel in the image. On the other hand, RGB images require a three-dimensional array where the first, second and third planes represent the red, green and blue pixel intensities. The image is interpreted according to the numerical class in which the data is stored: double-precision floating-point (double), 16-bit unsigned integer (uint16) and 8-bit unsigned integer (uint8). Images stored in double-precision (64-bit) floating-point numbers require a very large memory and are not always ideal for storing images. Thus, it is of recommendation to store images as 8-bit or 16-bit unsigned integers that require one-eight or one-fourth, respectively, as much memory as double precision numbers.

There are three basic data types used in MATLAB in accordance with interpretation of the data matrix of the image: indexed, intensity and RGB image. An indexed image consists of a data matrix and a colormap. The colormap is automatically loaded with the image when the '*imread*' function is used. An intensity image is a data matrix whose values represent a range of intensities of the pixel values. An RGB image on the other hand is represented as an m-by-n-by-3 data array that defines the red, green and blue colour components of each pixel. Thus, the colour of each pixel is stored as a combination of the intensities of the red, green and blue components of colour planes at each pixel's location. The '*imread*' function is used to read an image, the '*imwrite*' function is used to write an image and the '*imfinfo*' function is used to obtain information about the image.

MATLAB supports the following graphics file formats:

- BMP (Microsoft Windows Bitmap)
- HDF (Hierarchical Data Format)
- JPEG (Joint Photographic Experts Group)
- PCX (Paintbrush)
- PNG (Portable Network Graphics)

- TIFF (Tagged Image File Format)
- XWD (X Window Dump)

The essential properties of an image in MATLAB are the CData, XData and YData. The CData property determines whether the image will be interpreted as an RGB image or as a colormap image. If the CData array is two-dimensional, the image is interpreted either as an indexed image or an intensity image and in each case, the colormap colours are used. If the CData has a dimension of m-by-n-by-3, the image is displayed as an RGB image.

CHAPTER 5

METHODOLOGY

The development of the component for the goGPS software was carried out using the well-known waterfall approach for software development. This was necessitated to measure progress and keep track of the development from the start to the end of the work. As the development proceeds, new and implicit goals emerge (Tang and Vliet, 2012) with sub goals that need to be solved to enhance the final output of the work. Changes to the output of previous stages on the waterfall line were made whenever a new requirement emerged. More so, results in each stage were communicated to GReD s.r.l. and based on the feedback, subsequent steps were taken or updates were made to the requirements.

After its completion, the visualisation tool was used to investigate an incident at Malpensa Airport in which Emirates' Boeing 777-300ER (A6-ECF) was damaged by hail storm and returned to Milan 90 minutes after taking off to John F. Kennedy (JFK) International Airport in New York City. The incident happened on July 13, 2021.

5.1 Feasibility study

At this stage in the development process, it was assessed whether the requirements for the project could be feasibly achieved. In order to achieve something comparable to what can be found on most weather prediction maps (example in Figure 25), the following questions had to be answered

- Is it possible to add an image or icon that represents different weather phenomena to a plot in MATLAB?
- Is it possible to add weather parameters such as humidity, temperature, pressure, etc. to a plot in MATLAB?
- Is it possible to display weather information for more than a day on a plot in MATLAB?
- Can the expected system be built at all?

- If it can be built, how long will it take to complete it?

Thanks to the guide provided by MATLAB on adding graphical objects to a plot, the project was found feasible. However, there had to be a trade-off of time for its realisation.

Figure 25 ***Weather prediction map interface of termostat app***

5.2 Requirements identification

During the description of the problem to be solved, that is developing an explanatory tool for goGPS, GReD s.r.l. provided initial 'MUST HAVE' requirements to be met. However, other 'SHOULD HAVE' requirements were also identified during the development stage and were agreed upon with GReD s.r.l. Below are listed the requirements the tool seeks to satisfy.

- Given any specified location in the world, it is required to obtain weather information to be shown on the plots of goGPS.
- Given any date of the past, retrieve the weather data.
- The data to be used has to be obtained from the DarkSKY API.
- It is required to show the type of weather (e.g. rainy, snowy, sunny, etc.).
- Other parameters useful to be shown on the plots include temperature, humidity, pressure, wind speed, wind direction etc.

- The user should be able to display the parameters using a menu item.
- The user should be able to hide or show desired parameters by accessing the menu items.
- The user should be able to hide or show all the parameters by accessing the 'Show ALL' item in the Meteo-menu.
- The system has to be developed as an explanatory tool for the plots produced by goGPS software.
- The system has to be adaptable for all estimates derived from GNSS observation data.
- The system has to be used by the users of goGPS software, thus, researchers, students, meteorologists, etc.

5.3 Design

To meet the requirements found, the components needed for the development of the explanatory tool for the plots were identified as follows:

- Meteo-Menu: This component provides a means of access to the various functions available to the user. The menu will contain the various items for displaying the necessary weather parameters on the plot. It depends on the availability of an existing figure (containing plot of goGPS).
- Figure: This component contains a plot on which the weather parameter needs to be shown. It is essential for the figure to contain a line plot, having a starting and ending point that define the period in which the weather parameters are needed. Also, the Figure component should contain location information (latitude and longitude) about the station being monitored or analysed. Upon this plot, the Data Manager component will either add or remove the weather parameters.
- Data Manager: This component contains the essential functions that, given the starting and ending time from the Figure component, draws the graphical objects for the various weather parameters on the Figure component. It also provides a means for removing the displayed weather parameters from the plot.

- Data Source: This component refers to the DarkSKY API and Open Topo Data API that provide the data used by the Data Manager component.
  The DarkSKY API has the advantage of providing data that covers the globe, it is historical and gives forecast and present weather conditions. Also, it displays its data on the web making it easier to be read using the webread function of Matlab. Alternatively, other APIs that offer weather data are the Meteoblue weather API and the Global Surface Summary of the Day (GSOD) of the National Oceanic and Atmospheric Administration. An API key is required to be purchased before the weather dataset of Meteoblue can be accessed. GSOD requires its dataset to be first downloaded in a CSV format. Thus, the DarkSKY API was the API of choice for the purpose of this work as no API key was required for access.
  The Open Topo Data API provides the geodetic height for a given station given its geodetic coordinates. It provides a global elevation dataset (including bathymetry and ice surface elevation near poles) developed by the National Oceanic and Atmospheric Administration (NOAA). It has an estimated accuracy of about 10 m, which vary spatially depending on the underlying data source. Other APIs that offer similar elevation data and could be relied upon in case of failure of Open Topo Data API are the National Geodetic Survey API, Open-Elevation API and Google Maps Elevation API. The Open Topo Data API is free and has public access without API key requirement and is considered for this work.

The component diagram (Figure 26) shows the various components, with the arrows indicating the dependencies between them. Figure 27 also identify the properties and methods available for the various identified components.

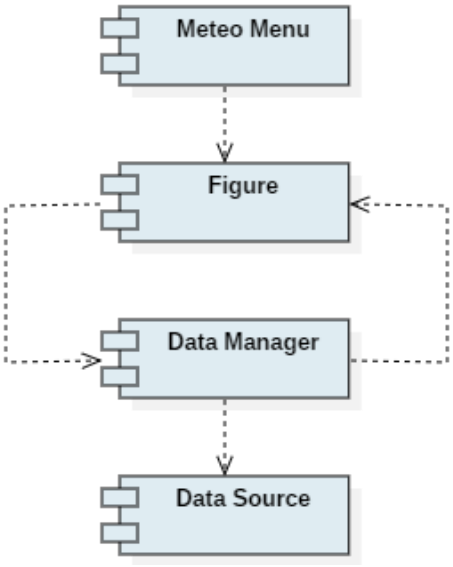


Figure 26 ***Component diagram***

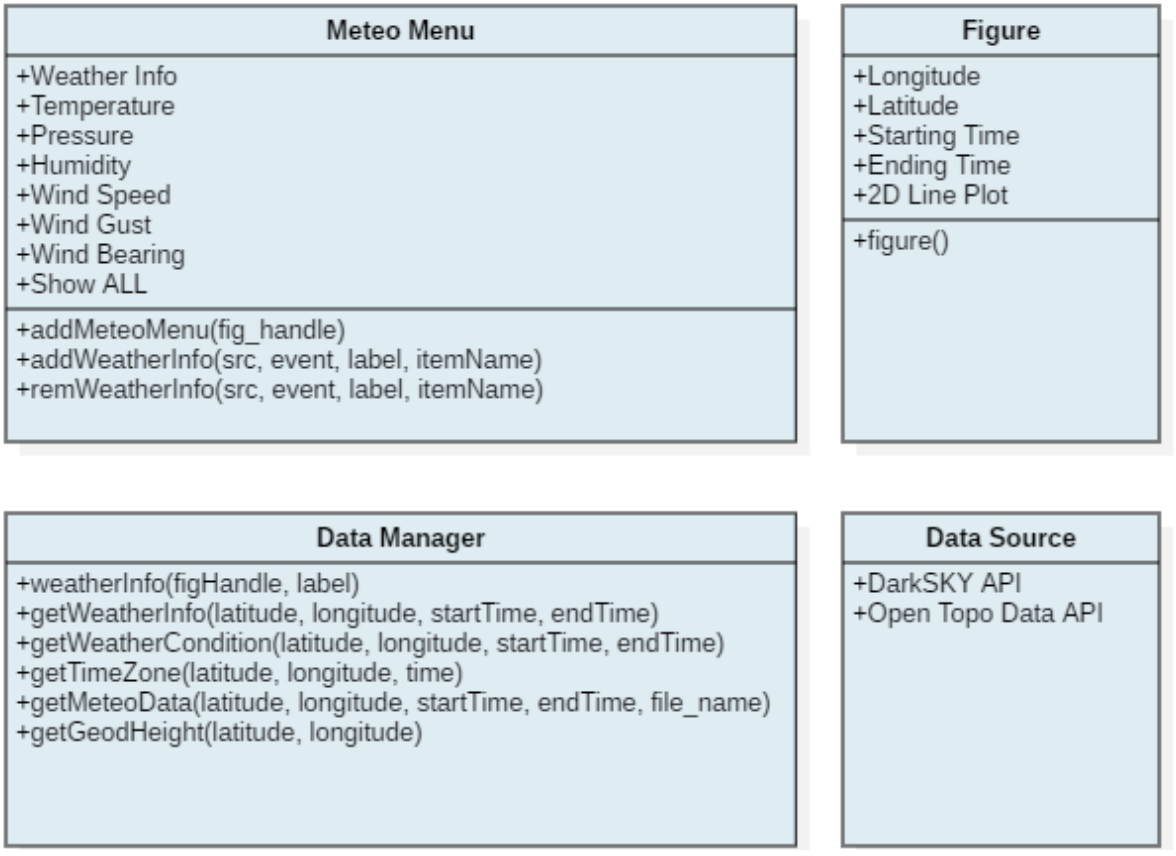


Figure 27 ***Component interfaces***

5.3.1 Sequence Diagram

To better understand the expected output and before commencing the implementation of the system, a sequence diagram (Figure 28) was created to show the interaction between the various components of the system for adding or removing the weather parameter information from the figure. The components are shown as vertical dashed lines. The horizontal direction shows the components, the vertical dimension shows with time sequence the order of message exchange between the components and the arrows indicate the message exchange between them. The rectangles indicate the period in which a component is active in a session.

The user instantiates the process by adding the Meteo menu to the figure by calling the addMeteoMenu() method of the Meteo menu component. The Meteo menu item contains the sub items labelled ‘Weather Info’, ‘Temperature’, ‘Pressure’, ‘Humidity’, ‘Wind Speed’, ‘Wind Gust’, ‘Wind Bearing’ and ‘Show ALL’.

By selecting any of the items, the addWeatherInfo() method of the Meteo menu component is activated, which gives the handle of the figure to the Data Manager component. At this point, the weatherInfo() method of the Data Manager retrieves the position (longitude and latitude) coordinates of the station being analysed as well as the starting and ending time of the analysis period from the plot on the figure. The position coordinates have to be in the WGS 84 reference system and the time values as string of numbers in the format ‘yyyy-MM-dd’. The acceptable range of position coordinates are as given in Table 4. It is thus expected that the 2D line plot will be produced at least from a GNSS related data. The weatherInfo() method then uses an HTTP Get request to read the Weather Info, Weather Condition and Time Zone data from the Data Source, (DarkSKY API), using respectively the getWeatherInfo(), getWeatherCondition() and getTimeZone() methods of the Data Manager. The data is read in an HTML format. At this point, the weatherInfo() method parses the raw data by extracting the relevant ones and performs data cleaning by removing the unwanted characters. However, the time values in the extracted data are in Unix Time format, which is inconsistent with the MATLAB time format used for the plots in goGPS. Hence, they are converted to MATLAB time format (the number of days since January 1, 0000) by calling the GPS_Time() method of goGPS. The updated time information is then used with the other data to create

the graphical objects. Depending on the label on the menu item which was selected from the Meteo menu, the graphical object is displayed on the plot. The displayed graphical object contains information about the corresponding weather parameter. Finally, the checked property of the selected menu item is set to 'ON' status.

In addition, the getMeteoData() method of the data manager accesses the weather data retrieved from the DarkSKY API and the getGeodHeight() gets the elevation data for the station from the Open Topo Data API. These data are then used as input for methods available in the Meteo_Data object of goGPS for the generation of the RINEX (i.e. Receiver INdependent Exchange (RINEX)) meteorological file.

Similarly, with the intention of removing a displayed graphical object from the plot, the user selects the corresponding checked menu item. The Figure component calls the remWeatherInfo() method of the Data Manager. The remWeatherIfno() method then gets the label of the menu item and the handle of the figure and removes the appropriate graphical object from the plot. Finally, the checked property of the respective menu item is set to 'OFF'.

Table 4 ***Acceptable Range of Position Coordinates***

Parameter	**Minimum**	**Maximum**
Latitude (Deg Decimal)	-90	90
Longitude (Deg Decimal)	-180	180

5.4 Development and testing

The implementation of the tool commenced using the bottom up approach. This was used, instead of the top-down approach, due to its flexibility for developing the units, ensuring that they produce the right results before integrating them with goGPS.

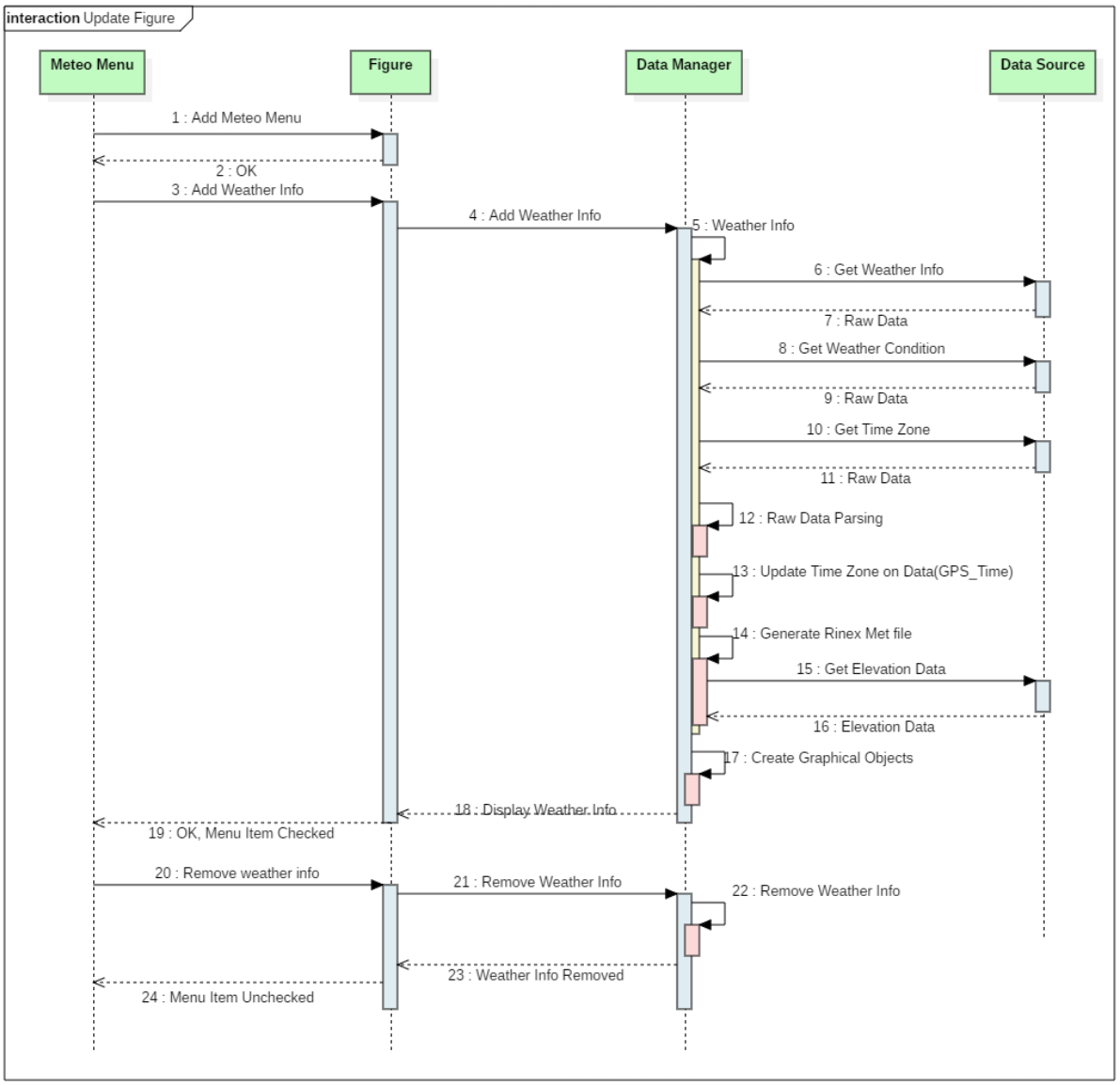


Figure 28 ***Sequence Diagram***

This section was divided into two major parts, implementation and testing of the various units and then integration and testing of the various units on the goGPS platform. The testing was carried out concurrently with the implementation phase due to the necessity of ensuring that the various units are error free and provide the right results in accordance with the specified requirements. This made it easy getting the units available and ensuring their smooth integration with the goGPS system.

5.4.1 Implementation of unit components

In this section, the components identified earlier were implemented in two steps:

i. Preparation of the data
ii. Visualisation of the data

A defensive approach was used at all stages of the implementation to handle cases where the end-user might attempt to use the program incorrectly.

Preparation of the data

At the data preparation stage, the Data Source component was developed. To start with, there was a need to create a temporary visualisation of the output of the webread() method of MATLAB to identify the necessary section where the needed data are (Figure 29). This was done using an arbitrary location and period.

Figure 29 ***Output of webread, needed data are highlighted red***

Some of the needed data were found to be in JSON format. There was a need to organise them in a structure that would be easier to use for creating the graphical objects. To create the structure out of the raw data, two major functions in MATLAB

were identified, jsondecode and cell2struct. As the name suggests, jsondecode decodes JSON-formatted text as MATLAB data type: empty double, scalar logical, scalar double, character vector, scalar structure, cell array, structure array, etc. On the other hand, cell2struct converts a cell array into a structure array. Through several trial and error, the cell2struct method was found very easy to be used for the purpose, thus, it is further discussed.

The cell2struct method accepts three input arguments: cellArray, field(s) and dim. The cellArray is the data that needs to be converted to a structure array. The field argument specifies the field names for the resulting structure array. This could be specified as a character array, a cell array of character vectors, or a string array.

The dim argument on the other hand indicates which axis (row or column) of the array is to be used for the structure array. If the rows of the cell array are to be used as field names for the output structure, then the field names are to be specified such that the number of fields is equivalent to the number of rows of the cell array. In this case, dim will have a numerical double value of 1. However, if the columns of the cell array are to be used as field names for the intended structure, then the field names are specified such that the number of columns of the cell array is equivalent to the number of fields of the expected output structure. In this case, dim will have a value of 2.

Data cleaning

Here, the implementation of the Data Manager component began. Before using the cell2struct method, the raw data obtained from DarkSKY API was an HTML file with embedded JavaScript section. This implies that it contains data that are not relevant for the purpose of displaying the graphical objects on the plots. The useful information was found in the JavaScript section by visual inspection of the output of webread on the command window, as highlighted in Figure 29. This was parsed by extracting the data from the JavaScript section and then removing unwanted characters such as ‘ ” ’, ‘:’ and ‘-’. The key names of the JSON formatted texts were used as field names for the expected structure. The values of the JSON formatted

texts were also supplied as values for the structure. Since in the cell array of the cleaned data the rows indicate the time sequence of observation data, a value of 2 was specified for the dim argument in order for the field names of the structure to be extracted from the columns of the cell array. The procedure was repeated in the getWeatherInfo(), getWeatherCondition() and getTimeZone() methods of the Data Manager components for retrieving the weather info, weather condition and time zone values. The outputs of the getWeatherInfo() and getWeatherCondition() methods are in a structure while that of getTimeZone() method is a numeric double data type.

The weather info refers to the values for the weather parameters such as temperature, pressure, humidity, wind speed, etc. These contain the needed information to be displayed on the plot. The weather condition refers to the condition of the weather for the epoch in which the station analysis is being made. Some of the possible values are overcast, clear, partially-clear-night, sunny etc. These are needed to select an icon for representing the weather during visualisation of the data on the plot. The time zone value represents the time zone in which the position coordinates of the station falls. This is very essential to appropriately convert the UNIX time of the cleaned data into MATLAB time in the weatherInfo() method, otherwise the position of the graphical objects will be shifted with respect to the true time-position on the plot.

At this stage, the weatherInfo() method of the Data Manager needs to be created. However, as the weatherInfo() method needs to display the graphical objects on a plot in a figure, a stub of the figure needed to be created. This was very essential as the figure on which the graphical objects are to be displayed are components of the goGPS system and only available at the integration stage. It is expected that the figure will contain a 2D line plot on which the meteorological information is to be displayed as an explanatory tool. Thus, the stub created depicts this feature.

Once the data have been cleaned, prepared and made available in a structure, and the getWeatherInfo, getWeatherCondition and getTimeZone units are ready, the next step of the unit's implementation, i.e. visualisation of the data, begins.

Visualisation of the data

At this stage, the weatherInfo() method of the Data Manager plays a key role and its functionality is realised. It begins by obtaining the handle of the figure on which the graphical objects are to be plotted. Although the labels of the sub menu items of the Meteo menu are being used by the weatherInfo() method as input argument (as shown in the component diagram in Figure 27), a stub of the Meteo menu component is not necessarily needed. The Meteo menu component is only needed at the integration stage.

The method retrieves the coordinates of the station being analysed from the figure. If no such information is available, the program aborts, the method returns and an error message (Figure 30) is thrown to the user in modal mode. This implies that all other operations on the figure are paused until the user attends to the message.

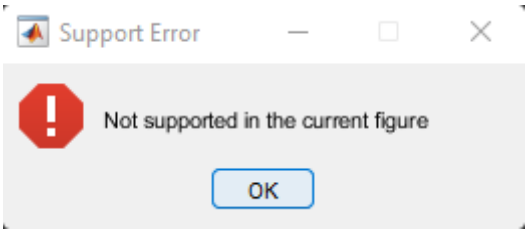


Figure 30 ***Support error message***

It then finds the starting and ending time information of the line plot from the axes on which the graphical objects are to be displayed. With the time and position information at hand, the getWeatherInfo(), getWeatherCondition() and getTimeZone() methods are called to retrieve the meteorological information from the DarkSKY API. It also calls the GPS_Time method of goGPS to update the time data with the time zone in which the station falls.

Using the weather conditions, the appropriate icons need to be selected and rightly set for display, thus the next step is to choose and set the size of the icons.

Choosing the Icons

The icons used for representing the various weather conditions are similar to that published by Yangtze (2021). These contain representations of most weather phenomena. These icons were too large in size for this purpose, requiring a large amount of memory size. Images in MATLAB have a CData property that contain the actual data for the pixels of the image. According to the guidelines for using graphical objects in MATLAB, to increase the rate at which MATLAB can update the CData property of an image object you can optimise CData using the guides below

i. Use the smallest data type possible. Using a uint8 data type for an image will be faster than using a double data type.

ii. Use the smallest acceptable matrix.

iii. If the speed at which the image is displayed is of highest priority, you may need to compromise on the size and quality of the image. Again, decreasing the size will reduce the time needed to copy the matrix.

Thus, to ensure faster loading of the images and reduce the size of memory required to store them, it was essential to redesign the icons using the symbols specified by Yangtze (2021). For each icon, a dimension of 120 pixel by 120 pixel units were specified at a resolution of 100 pixels per inch. This ensured a uniform dimension of the images with an average size of 3.76 KB, which was considered very useful to store all the images and for multiple images to be loaded at the same time without delay. Additionally, a transparent background was set for all the images so that only the required symbol for the weather phenomena will be on display without background effect.

However, when analyses are being made for station data that spans two days or more, the icons appear to be crowded on display, with same images being repeated multiple times and thus too much for the plot box region of the axes. Thus, a generalisation of similar and consistent icons had to be produced, using patches with unique solid face colours to represent each icon. The patches have the same height as the icons, with the width spanning across the number of consistent and similar icons in the data. This generalisation effect was thus considered effective for

stations that have more than 30 hours of observation data. Finally, the icons are stored as 8-bit RGB images for easier handling in MATLAB.

A major setback that was faced was that, the X-axis and Y-axis properties of the axes on which the images were to be displayed have different data units, with the X-axis having units of time and the Y-axis having data unit depending on the type of data being analysed over the station. Thus, it was essential to control the aspect ratio of the images so that they will be displayed correctly, with similar height and width irrespective of the unit of data being used for the Y-axis.

Controlling Aspect Ratio and Display Size of Icons

Because the axes on which the icons are to be shown have different units, there was a need to perform a transformation (Figure 31) in order to preserve the aspect ratio of the icons. This was done with respect to the constant time unit on the x-axis that is common for all other monitoring (observation) data and by exploiting the pixel units for the screen on which the figure will be shown. By this means, it will be possible to create equal width and height values for the icons, for them to be correctly displayed on the screen.

The width of the icon was determined in the original unit on the x-axis and then converted to pixel units. An equal dimension of width in pixel units was also set for the height in pixel units. Then the height was converted to the original unit on the y-axis. This helped keep the aspect ratio and sizes of the images the same on display, irrespective of the units on the y-axis.

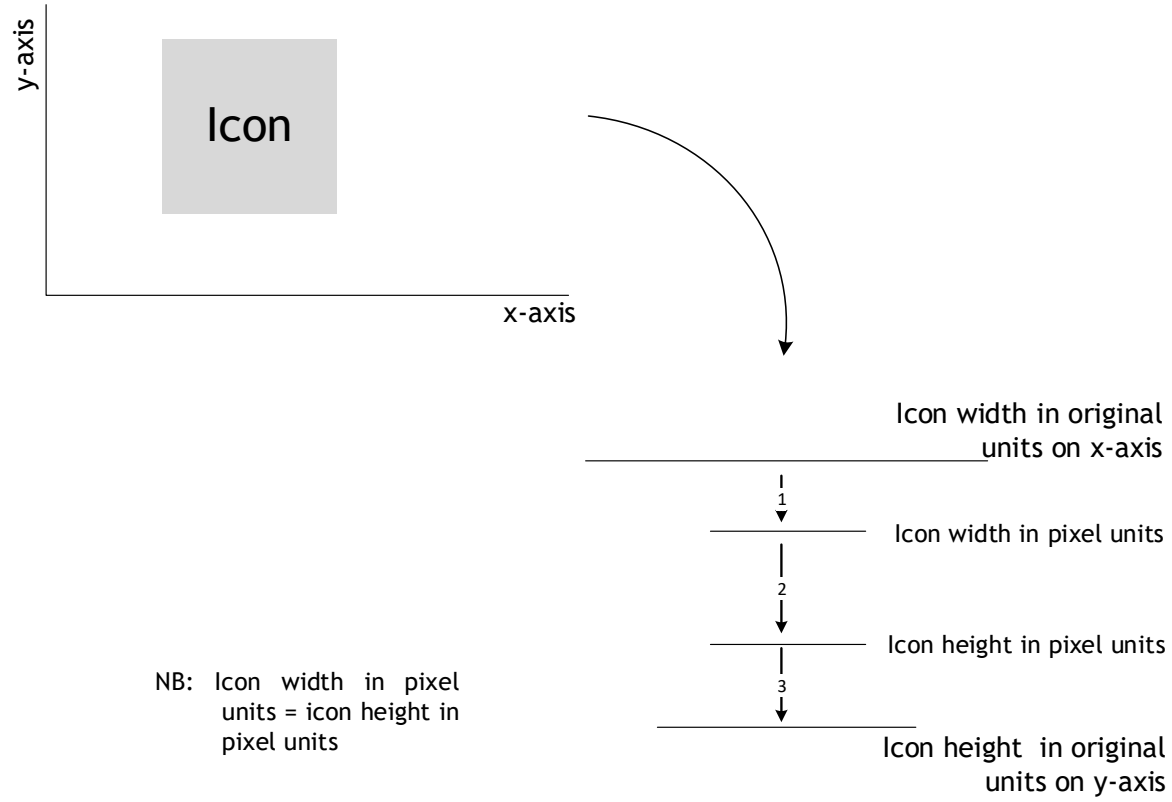


Figure 31 ***Transformation of Icon Dimensions***

Displaying the icons

The icons are then loaded into MATLAB using the syntax below where 'image name' is the name in which the image is saved.

```
imread('image name')
```

Afterwards, the image can be displayed. To properly display the images on the plots, it was essential to identify a container for them. Two graphical objects, Patch and Surface, were identified to be possible candidates to contain the images.

Patches are filled 2-D polygons that are positioned and oriented using the coordinate system of the axes object. A single patch can be coloured with multiple face colours. A patch can be created using the patch function. Surfaces are 3-D representations of matrix data created by plotting the value of each matrix element as a height

above the X-Y plane. They are composed of quadrilaterals whose vertices are specified by the matrix data. Also, the coordinate system of the axes positions and orientates the surface objects. A graphical object of the surface type can be created using the surface or surf functions. The surface function was used in this work as 3-D view was not needed for this work (required with the surf function).

Through several trial and errors, the surface object was found efficient to contain the images. Thus, the CData of the image was supplied as an input argument to the surface object and a height value of zero (0) above the XY plane. The syntax below was used to create the surface object.

```
surface(ax, [x1, x2; x1, x2], [y2, y2; y1, y1], zeros(2), ...
    'FaceColor', 'texturemap',  ...
    'EdgeColor','none', ...
    'CData', im, ...
    'CDataMapping', 'direct', ...
    'AlphaData', alpha, ...
    'FaceAlpha', 'texturemap');
```

where ax is the parent axes object in which the surface is to be displayed, CData and AlphaData are the corresponding properties of the object. The other arguments are corresponding Property-Value pairs of arguments needed for creating the surface object. Of worth noting among the arguments is the FaceAlpha argument. This property sets the transparency of the surface. In MATLAB, there are four possible values that could be set for this property, which are a scalar value, flat, interp and texturemap. The scalar values could be in the 0 and 1 range, which represent fully opaque and completely transparent respectively. The flat value uses a different transparency value based on a specified AlphaData property. The ‘interp’ value interpolates the transparency for the face also based on a specified AlphaData property. The ‘texturemap’ property transforms the data in AlphaData so that it conforms to the surface. Since it was expected that the textural appearance of the

image would be maintained on display, the texturemap value was chosen as a suitable value for the FaceAlpha property of the surface object. As it was necessary to generalise the similar and consistent weather phenomenon type, the patch object was used for representing the icons, as previously discussed, using the syntax below

```
patch(ax, posX, posY, col);
```

where ax is the parent axes in which the patch object is to be displayed, posX and posY specify the position coordinates of the object, and col define the face colour to be used to represent the particular weather phenomena. Whenever a change in the icon type occurs in the observation period, the patch colour is correspondingly changed. To be able to interpret the different colours in accordance with the type of weather phenomena, a legend object was created to give interpretation to the various colours used for the patches in relation to the weather conditions available for the period of observation. A complete list of the icons and their colours are presented in Appendix I. The patches and surface objects have been created specifically for the 'Weather Info' item of the Meteo menu. The patch objects are shown for station analysis containing more than 30 hours of observation data and surface objects are shown for observation containing less than 30 hours of data.

Displaying the text objects

The other weather parameters such as temperature, pressure and humidity have values, which are of numerical data type and need to be converted to strings for display. Afterwards, these data values are displayed on the plot using the text function, parenting the text object to the axes object containing the plot. Graphical objects of the text type are character strings that are also positioned according to the coordinate system of the axes object in which it is to be placed. The text objects (containing the parameter values) are displayed with one hour spacing, with the corresponding weather parameter label (as shown in Figure 4). Thus, their positions are specified with a constant value on the Y-axis and the corresponding time values on the X-axis. The text object creation syntax is given below:

```
text(ax, posTX, posTY, parameterValue, 'Clipping', 'on');
```

where ax is the parent axes object

posX and posY are the x and y position coordinates for the object, the posY value remains fixed while the posX values are updated every hour to display the new value of the weather parameter.

parameterValue, formally labelled as the String property of the text object, is the value of the weather parameter to be displayed as a string character.

To avoid displaying the parameter values beyond the boundaries of the axes object, the 'Clipping' property of the text object has been set to the 'ON' state. By default, the horizontal alignment property of the text objects is set to the left position. However, for the text objects used for the labels of the weather parameters, they are positioned with a right horizontal alignment to avoid overlap with the starting values of the weather parameters.

It is important to note that for all the graphical objects used to add weather parameters to a plot in goGPS, a tag was specified for identifying the object according to the label of the item selected from the Meteo menu. A tag is a user-defined string that can be associated with a graphical object for identification.

The order in which the parameters are displayed on the screen

To avoid overlapping of the graphical objects, the objects are positioned for the weather parameters on a basis of first selected, first displayed. Thus, the objects are displayed depending on which item is selected first from the Meteo menu. Also, it is possible to display all the parameters using the 'Show ALL' item of the Meteo menu. A constant value depending on the units of the Y-axis has been set to separate the parameters on display, with an hour spacing on the X-axis.

Setting the axis limits

Because MATLAB automatically scales an axes to encompass a line plot, in most instances, the graphical objects will be displayed outside the bounds of the axes object. Thus, it was necessary to update the axis limits of the plots in order to make room for the graphical objects to be visible when a menu item is selected. This was performed using the function call below:

```
xlim([xmin    xmax])

ylim([ymin    ymax])
```

where xmin and ymin are the new bottom left coordinates for the axes boundary and xmax and ymax are the top right coordinates of the axes bounds.

Creating legend for the patch objects

The plots in goGPS are produced with a default legend based on the graphical objects displayed in the parent axes. To be able to create a legend for the patch objects, another axes object needed to be created. This is because MATLAB has the limitation of creating only one legend object per axes object. Thus, a copy of the existing axes in which the patch object has been displayed was created, hiding all its contents. To define the data content of the legend, the following steps were followed and summarised in Figure 32.

i. Create a vector containing unique icons in the icon types returned by the data of DarkSKY API.
ii. Specify a colour for each icon type in the unique icon set
iii. Create an empty vector of patch colours, patch objects, and icon names to be used for the legend. Each of these vectors has to have a length equal to the length of the set of unique icons created in step I above.
iv. At an epoch in which each parameter is to be displayed, obtain the icon to be used.

v. Find the index of the identified icon in the unique list and obtain its corresponding colour from the unique set of icon colours.
vi. Create the patch object with the icon colour as face colour.
vii. If the icon colour is not a member of the vector of patch colours, add it to the patch colours vector at the index of the icon obtained in step v.
viii. Add the handle of the patch object, whose colour has just been added to the vector of patch colours, to the vector of patch objects also at the same index as the index of the icon colour obtained in step v.
ix. Add the name of the icon to the vector of legend icons, also at the same index as the index of the icon obtained in step v above.
x. Create the legend object using the legend function of MATLAB, specifying as input arguments the vectors of patch objects and legend icons.
xi. If the number of hours in the data set is more than 30 hours, show the legend, otherwise hide it. This is necessary to ensure that the legend is displayed only when generalisation with the patches is being used to represent the weather information.

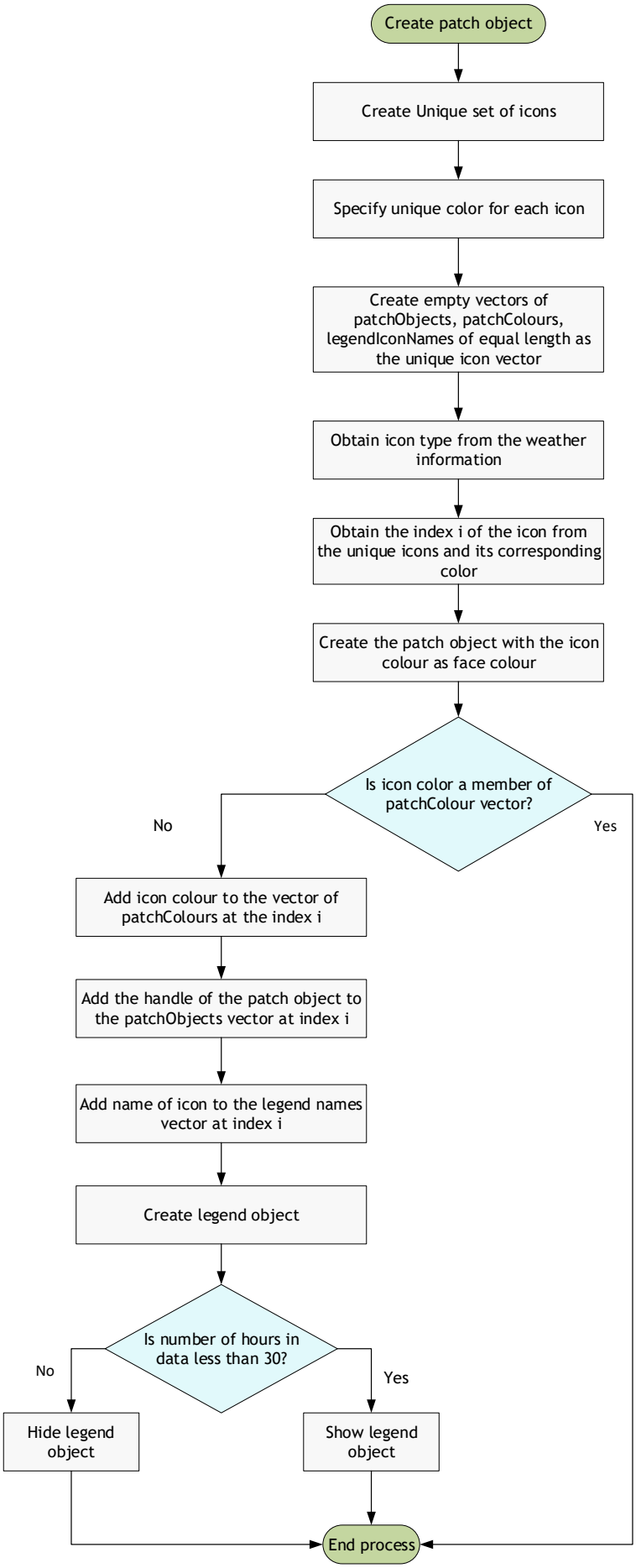


Figure 32 ***Creating legend object for the patch objects***

Removing a graphical object

By selecting an item from the Meteo menu list, its checked property is turned on and the corresponding graphical object is displayed on the plot. By selecting a checked Meteo menu item again, its checked status has to be turned off and this was implemented as follows. Using the label on the selected item, the tag on its corresponding graphical object is searched from the descendants of the parent figure. The tag identifier on the object is used as an input argument for the findobj function to find the handle of the graphical object. Then, the corresponding object is deleted passing the handle of the object as an input argument for the delete function. Then the checked status of the object is turned off.

5.4.2 Integration with the goGPS system

The developed units were integrated with the goGPS system using the bottom-up approach. This means that the components that do not depend on any other component start the integration, then successively, other components that rely on the independent or other components are integrated. The component relationship is shown as a dependency diagram in Figure 33. Here, the light blue coloured components are the newly developed components while the light green components are the components existing in the goGPS system. The methods and attributes of the newly developed components are as shown in the Component interface diagram in Figure 27. The methods and attributes of the existing goGPS components being accessed are shown in the dependency diagram in Figure 33. However, for the sake of space, only the components of goGPS that are being used in the integration are shown in Figure 33. The integration plan is executed using the following steps.

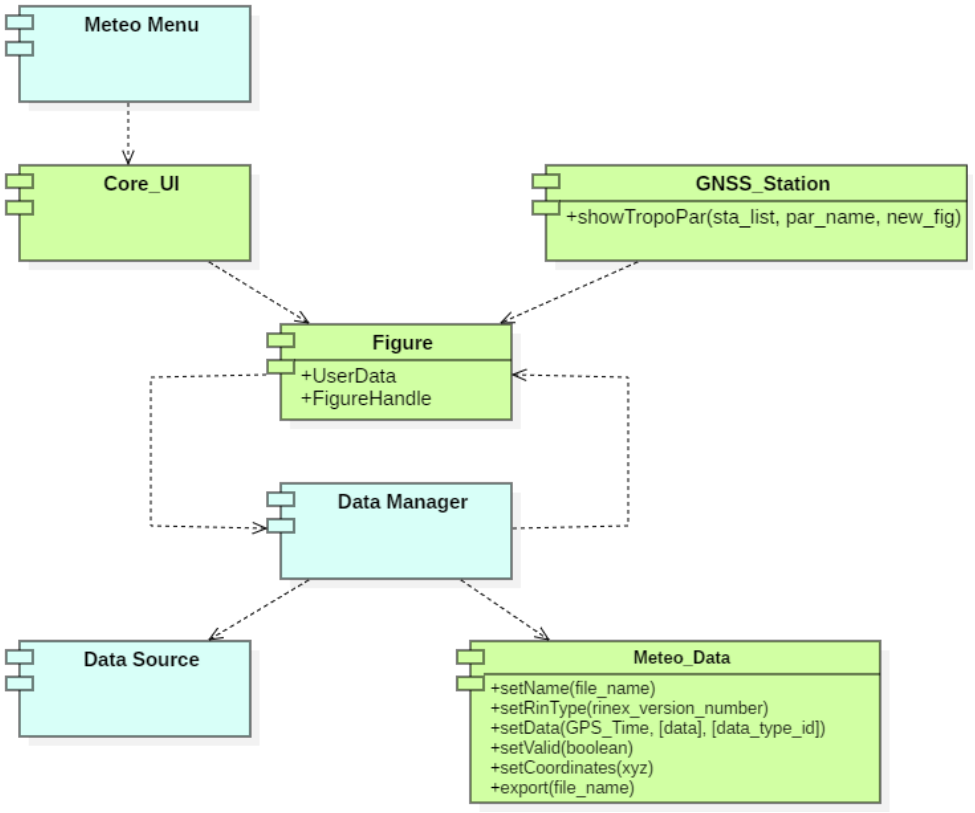


Figure 33 ***Dependency Diagram***

- Step 1

The Data Manager component is integrated with the Data source component using arbitrary latitude and longitude station coordinates and arbitrary starting and ending times (Figure 34). This is to ensure that the data structure is available for the figure later on.

Figure 34 ***Integration Step 1***

- Step 2

The figure component is made available by selecting any of the plot (Figure

35) from the goGPS after post processing.

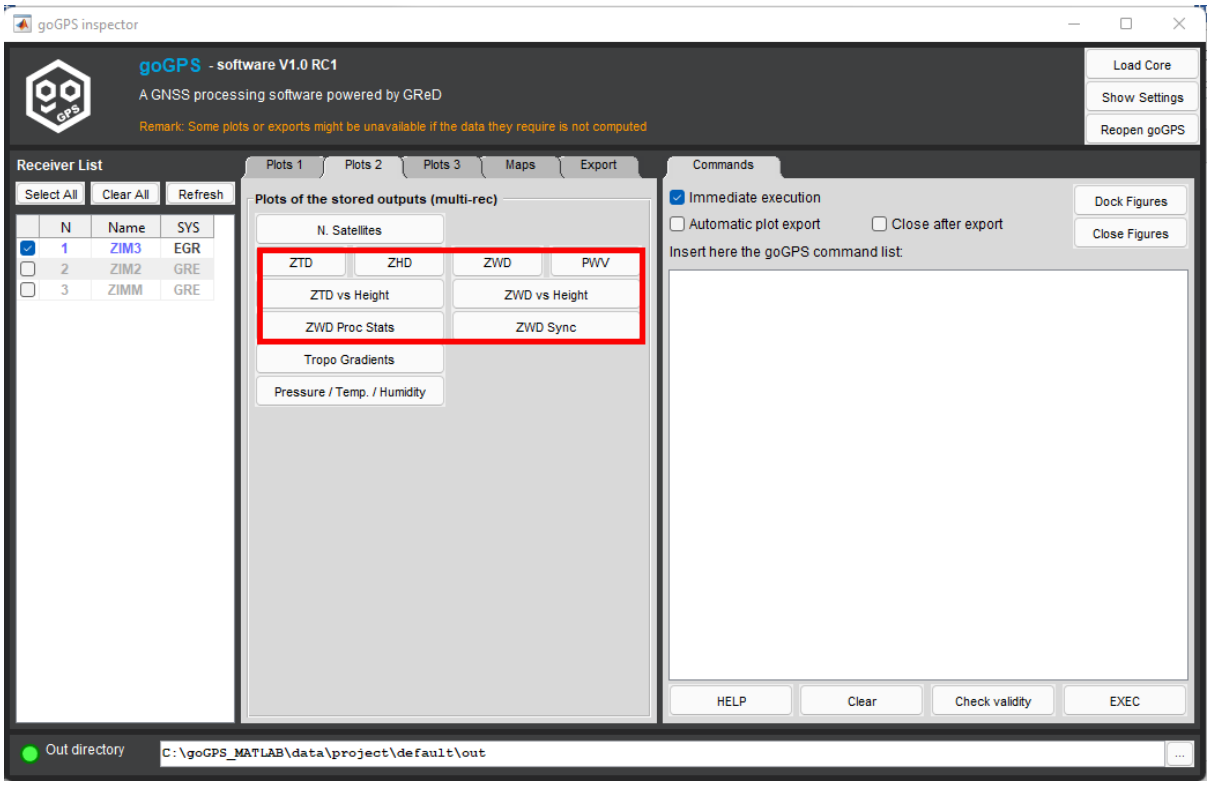


Figure 35 ***Available plots in goGPS***

- Step 3

The showTropoPar() method of the GNSS_Station component is modified using the expression below. It gets the mean of the geodetic coordinates of the station from the coordinates of the station made available after post processing. The geodetic coordinates are the longitude and latitude coordinates of the station obtained from the post-processing of the GNSS data. It then adds these coordinates to the UserData property of the Figure component using its handle (Figure 36).

```
[lat, lon] = sta_list(1).getCoo.getMedianPos.getGeodetic();

f.UserData = struct('fig_name', fig_name, 'coordinate', [lat, lon]);
```

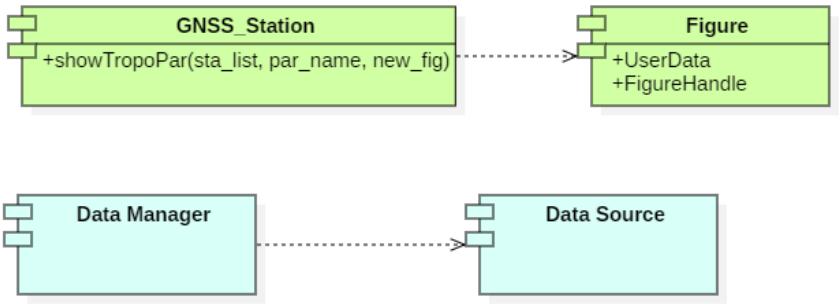


Figure 36 ***Integration Step 3***

- Step 4

At this stage, the Meteo menu component is integrated with the Core_UI component, which adds the menu item 'Meteo', and its sub-items to the figure (Figure 37).

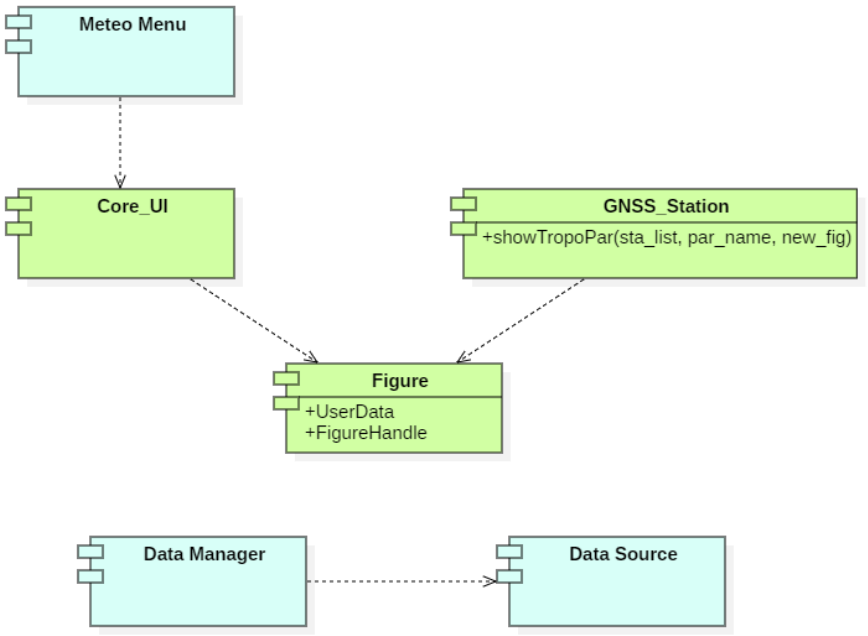


Figure 37 ***Integration Step 4***

- Step 5

At this step, by using the label(s) of the sub-items of the 'Meteo' menu item, the Figure component is integrated with the Data Manager component.

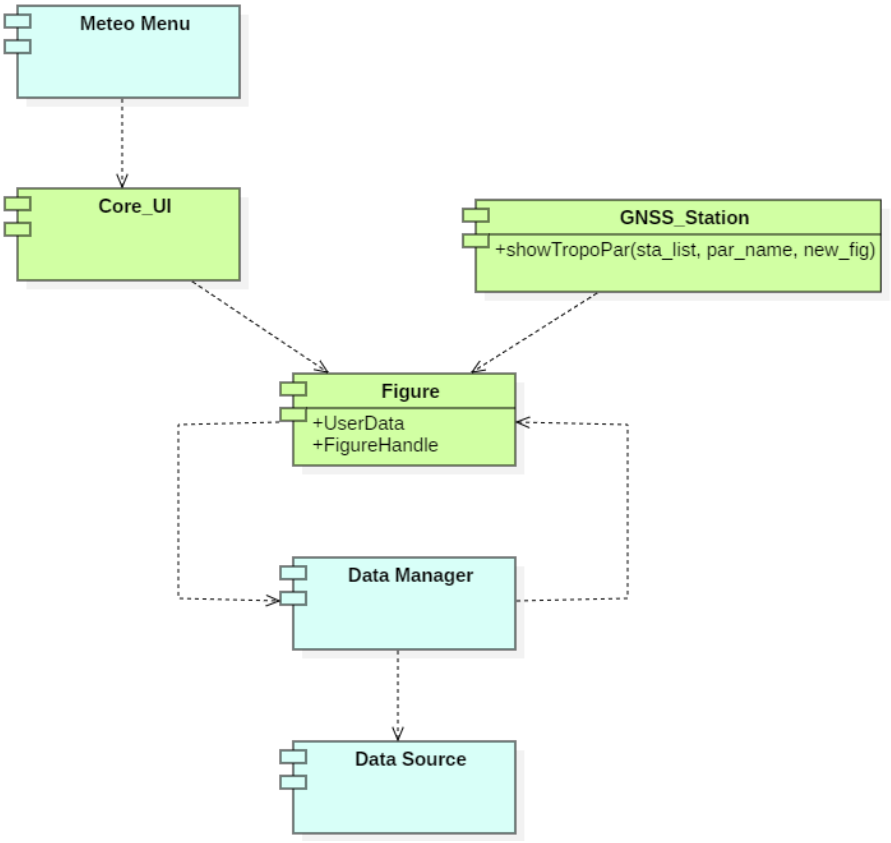


Figure 38 ***Integration Step 5***

- Step 6

Finally, the Data Manager component is then integrated with the Meteo_Data component of goGPS (Figure 35) to complete the system integration (Figure 38).

After the integration, the Meteo menu and its sub-items are available and the user can select any of the items to display the graphical objects (weather parameter) associated with the selected item.

To ensure confidence in the developed system, lots of tests were carried out and are described below.

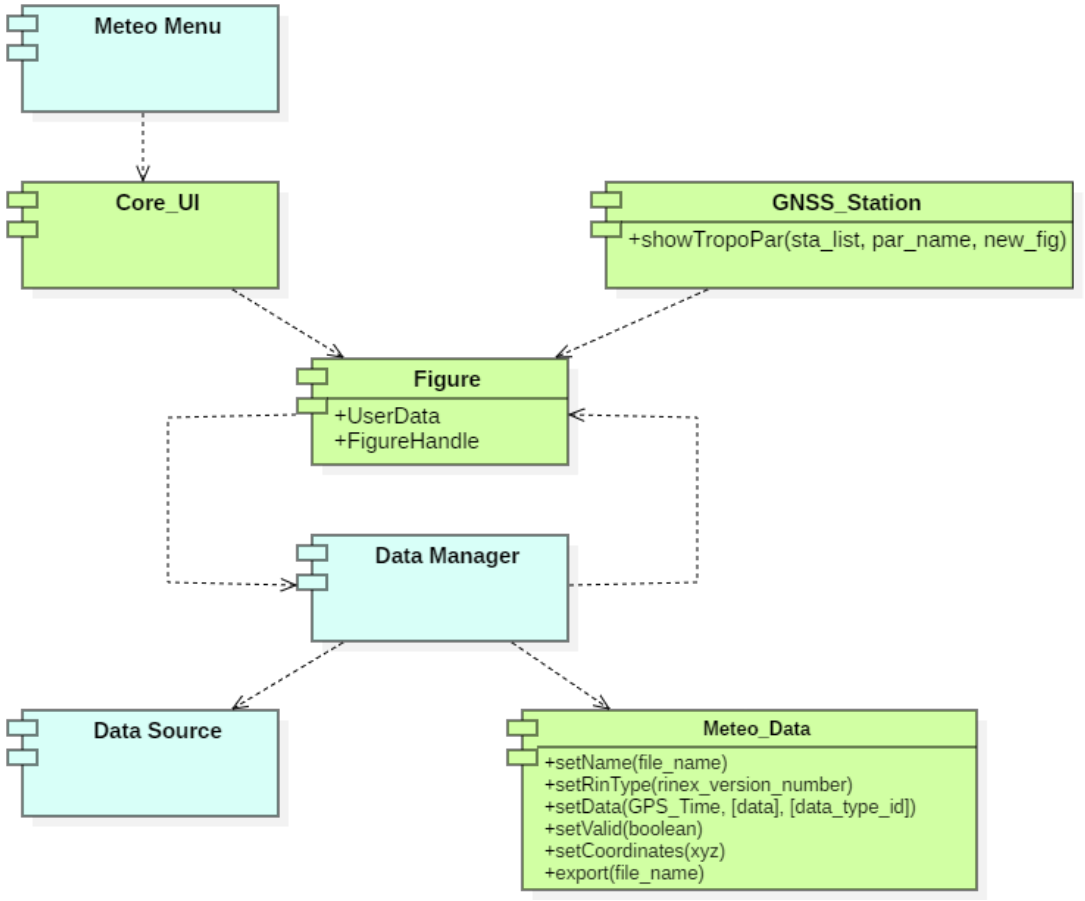


Figure 39 Integration Step 6

5.4.3 Testing

Of major importance in the execution of a software project is the satisfaction and reliability of the end-product in meeting the demands, requirements and specifications of the end-user. The end-product thus needs to be tested to ensure that it is free from errors, faults and failures before being released to the end user. Testing of a software means executing the end product of a software project to see whether it produces the correct output for a given input (Vliet, 2007). This involves the use of a set of test cases which according to the IEEE standard for software development is a set of test inputs, execution conditions and expected results developed for a particular objective, such as to verify compliance with a specific requirement. The set of test cases may be derived from the specifications of the software, in which case it is known as black-box testing, or derived from the

implementation of the software, in which case it is known as white-box testing. In similar vein, the technique to be used could be based on

i. detecting faults in the program also known as Fault-based testing,
ii. detecting errors in the program known as Error-based testing or
iii. checking that the program satisfies the requirements specified at the beginning of the project which is known as Coverage-based testing.

As stated in Section 5.4, testing of the developed program was carried out along the implementation. However, the details of it are here discussed.

The bottom-up approach was adopted for the testing, in the order of unit testing, integration testing, system testing and acceptance testing. The white-box test was used for the testing of the unit components and also at the integration stage. The main objective for carrying out the test in this instance was to find as many errors as possible in the codes. In most of the cases, the test of the unit components were carried out by printing variables, computation results, function return values etc. on the command window to ensure that there are no errors in the code. In this instance, anomalous input variables were used for the attributes of the various components. Also, the data manager was tested several times to ensure that after parsing the raw data from the data source (DarkSKY), it organises the data in a structure. Finally, the integration test was performed for the purpose of finding that the interaction between the components depicted in the dependency diagram in Figure 32 works perfectly.

The black-box testing approach was used for the system and acceptance testing, ensuring that the program developed meets the requirements specified in Section 5.2. This test was performed for the purpose of enhancing confidence in the developed system in appropriately displaying the graphical objects for the various weather parameters. In the system testing, it was found that the font colour of the graphical objects of the text type does not contrast the light background colour of goGPS. This was because the text objects are created with an explicit default font colour of white, which is similar to the light background of goGPS. This was reported to GReD s.r.l. and improvements are being made on it. After the system test, the files were given to GReD s.r.l. for performing code inspection and walkthrough. At

present, the developed system works on the Windows Operating System platform. It is anticipated that it will work on other operating system platforms as well, in accordance with goGPS requirements. The test is stopped after all the requirements specified in Section 5.2 have been met and the recommendations proposed by GReD s.r.l. are satisfied after code inspection.

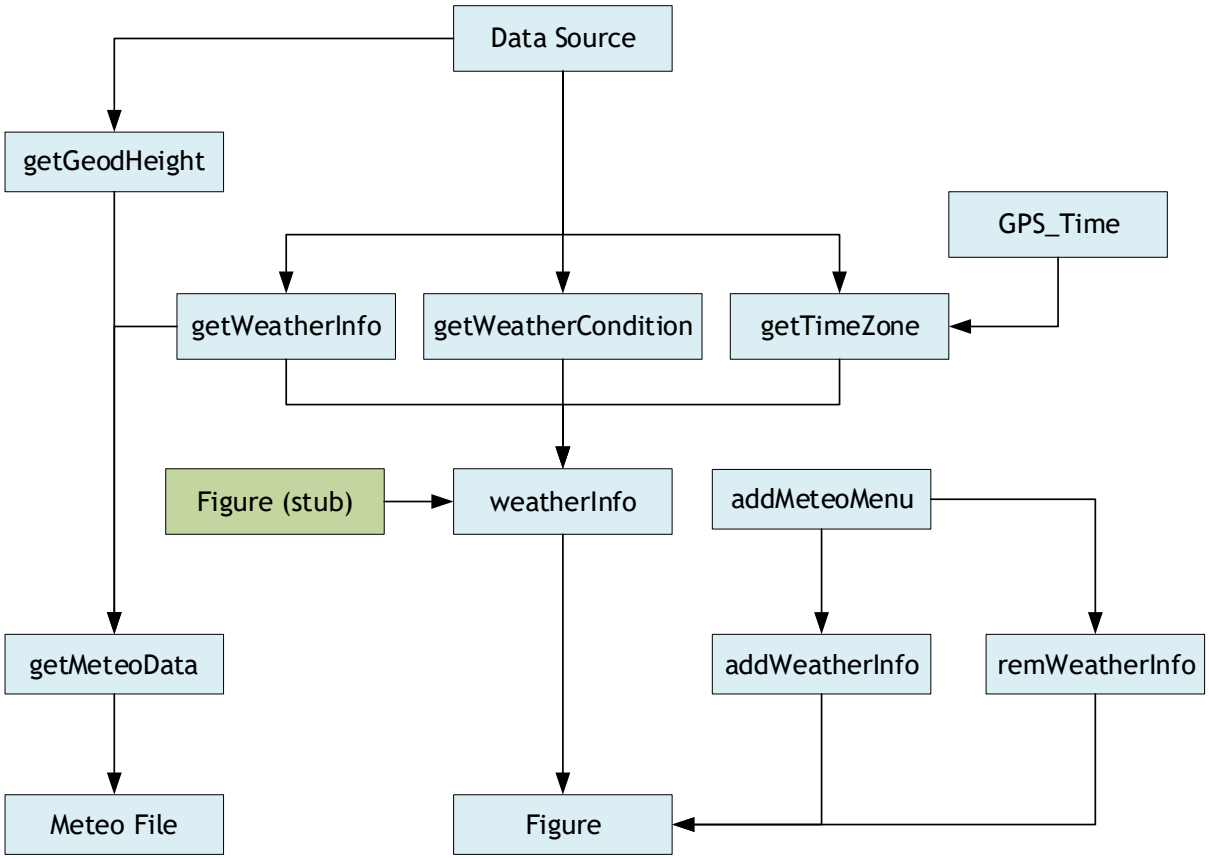


Figure 40 ***Test plan***

5.5 Deployment

The developed component will be deployed through the GitHub repository of goGPS MATLAB.

5.6 Maintenance

The initial version of the software component has been developed and deployed. The reliability and availability of the developed components depend on the Data Source component (DarkSKY API) making its data available 24/7. However, it is anticipated that through daily usage, additional requirements might evolve. Thus, in cognisance of the laws of software evolution by Lehman (1980), continuous communication with GReD s.r.l. will be ensured for maintenance to be made whenever a change in the requirement is recognised. In this case also, regression testing techniques will be employed to ensure that the software performs after the maintenance task. It will not be essential to rerun all the tests previously performed but a selection of some of the test cases will be carried out. However, to reduce the effort required for the maintenance of the software, the initial development of the program was carried out bearing in mind the following solutions proposed by Vliet (2007) for reducing maintenance problems.

i. By writing higher-quality code, better test procedures, better documentation and adherence to standards and conventions;
ii. By anticipating changes during requirements engineering and design and by taking them into account during realisation, future perfective and adaptive maintenance can be realised more easily.
iii. Fine tuning to user needs may lead to savings in perfective maintenance.
iv. By writing less code

5.7 Case Study

The purpose of this case study is to relate the ZWD derived from observations made by GNSS stations with weather parameters. The behaviour of ZWD and weather parameters were considered during an incident that happened on July 13, 2021 at Malpensa Airport, in which Emirates' Boeing 777-300ER (A6-ECF) was damaged by hail storm on its way to John F. Kennedy (JFK) International Airport, New York City. The airline suffered serious damages with shattered windshields, nose cone, engine

cowlings and wings. The flight departed Milan Malpensa airport at 16:23 but returned at 18:04.

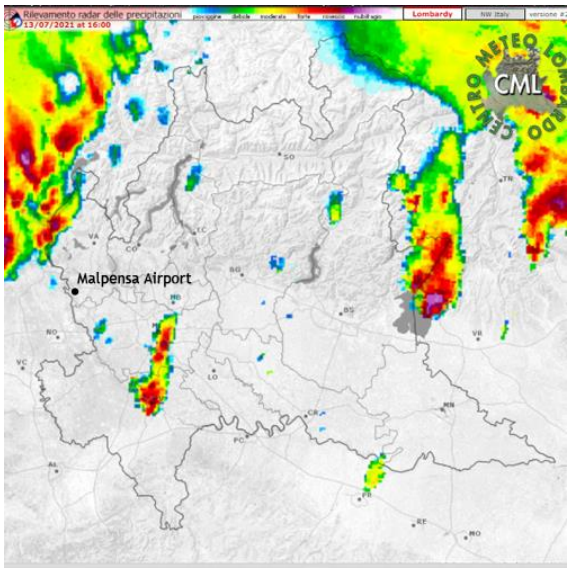
Weather radar maps have the capability of detecting areas of precipitation as well as its intensity, type and direction. On a weather radar map, different colours (Table 5) are used to represent the type of precipitation when one is detected over an area, Hebby (2022).

Table 5 Precipitation type according to colour on radar map

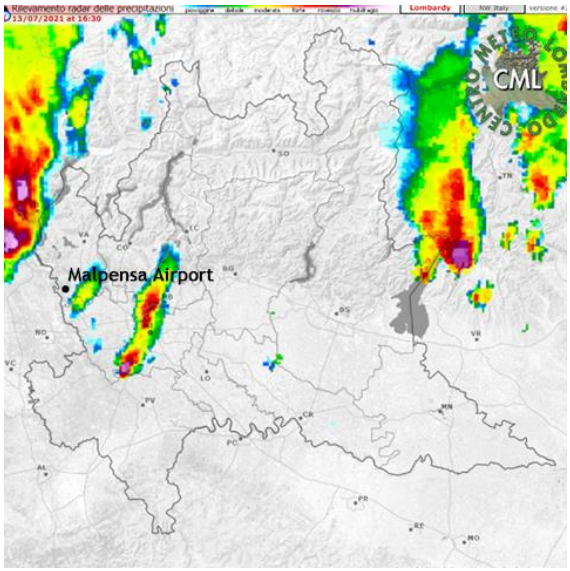
Colour on Radar Map	**Precipitation Type**
Light green	Light rain, or light rain aloft not reaching ground
Dark green	Light to moderate rain
Yellow	Moderate rain
Orange	Heavy rain
Red	Very heavy rain or rain and hail
White or blue	Snow
Pink	Freezing rain or sleet or mix of winter precipitation types

To better analyse the Malpensa airport incident using the developed visualisation tool, it was very useful to have pre-knowledge of the type of precipitation that was moving across the area using radar images (Figure 41) available on the website of Centro Meteo Lombardo. By inference of the radar images, there was a frontal system that was moving to the north-west of Milan Malpensa Airport with intense precipitation and strong winds (see Section 6.6) at the time the airline set off. Also, on the north-west part of the airport, the assumed path of the flight, there were

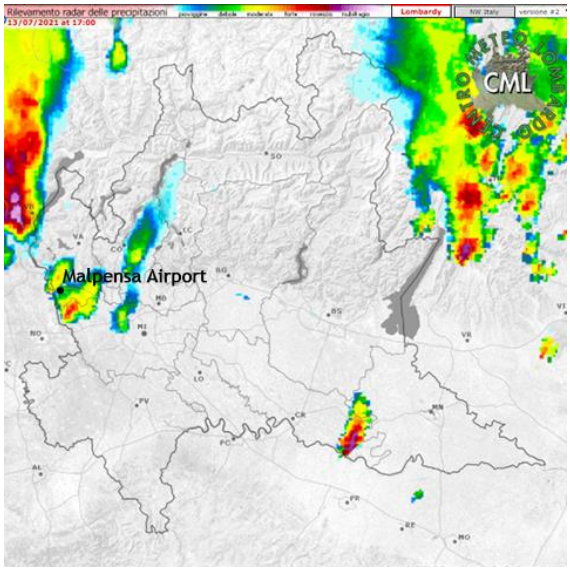
hail or rain that were also moving towards the east, in opposition to the flight trajectory.



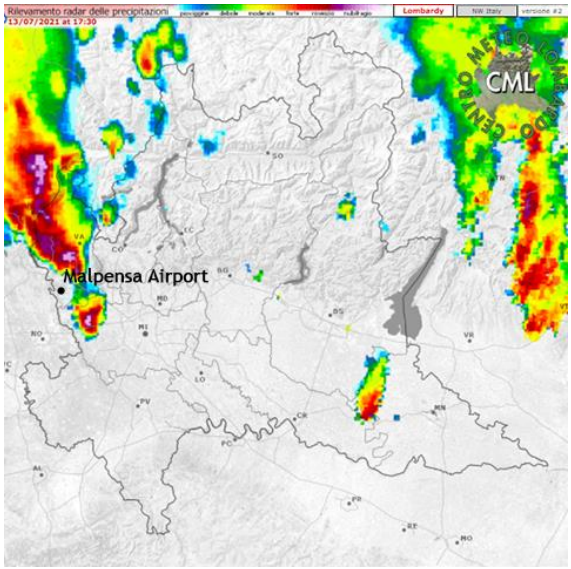
July 13, 2021-16:00



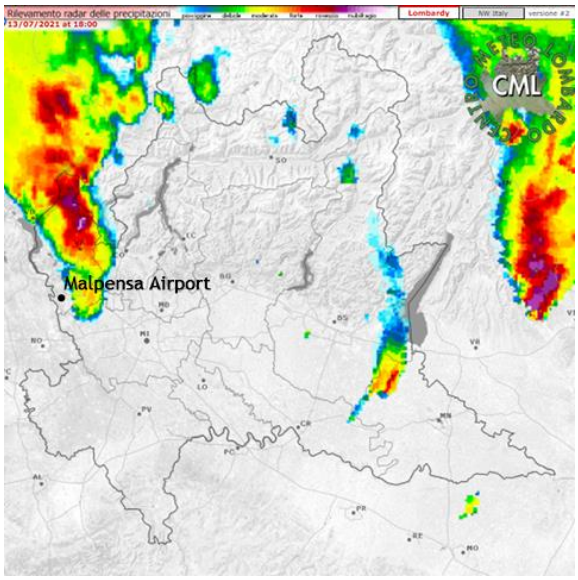
July 13, 2021-16:30



July 13, 2021-17:00



July 13, 2021-17:30



July 13, 2021-18:00

Figure 41 Progression of the hail over the Malpensa airport

(source: http://www.centrometeolombardo.com/radar/)

The considered GNSS stations (Figure 42) are 17 in all, 12 of which belong to the SPIN network (a GNSS data infrastructure of the Piedmont, Lombardy and Valle d'Aosta Autonomous Regions of Italy), one (CATU) belonging to the NetGEO network owned by Geotop society and GRTR belonging to GReD srl. These stations make observation of GNSS signals on a daily basis, are closer to the Malpensa Airport and their data are made available to the public. Thus, they were considered for this study. The data obtained from these stations were processed using the goGPS software and the ZWD derived from them were used to analyse the trend in the atmospheric water vapour content in the area during the incident. The ZWDs were obtained for a day preceding the event and on the day the event occurred. The meteorological information were then displayed on the plot of the ZWD to aid explanation of the ZWD plot. This was also to find possible correlation between the weather parameters and the ZWD plot. The results are given in Section 6.6.

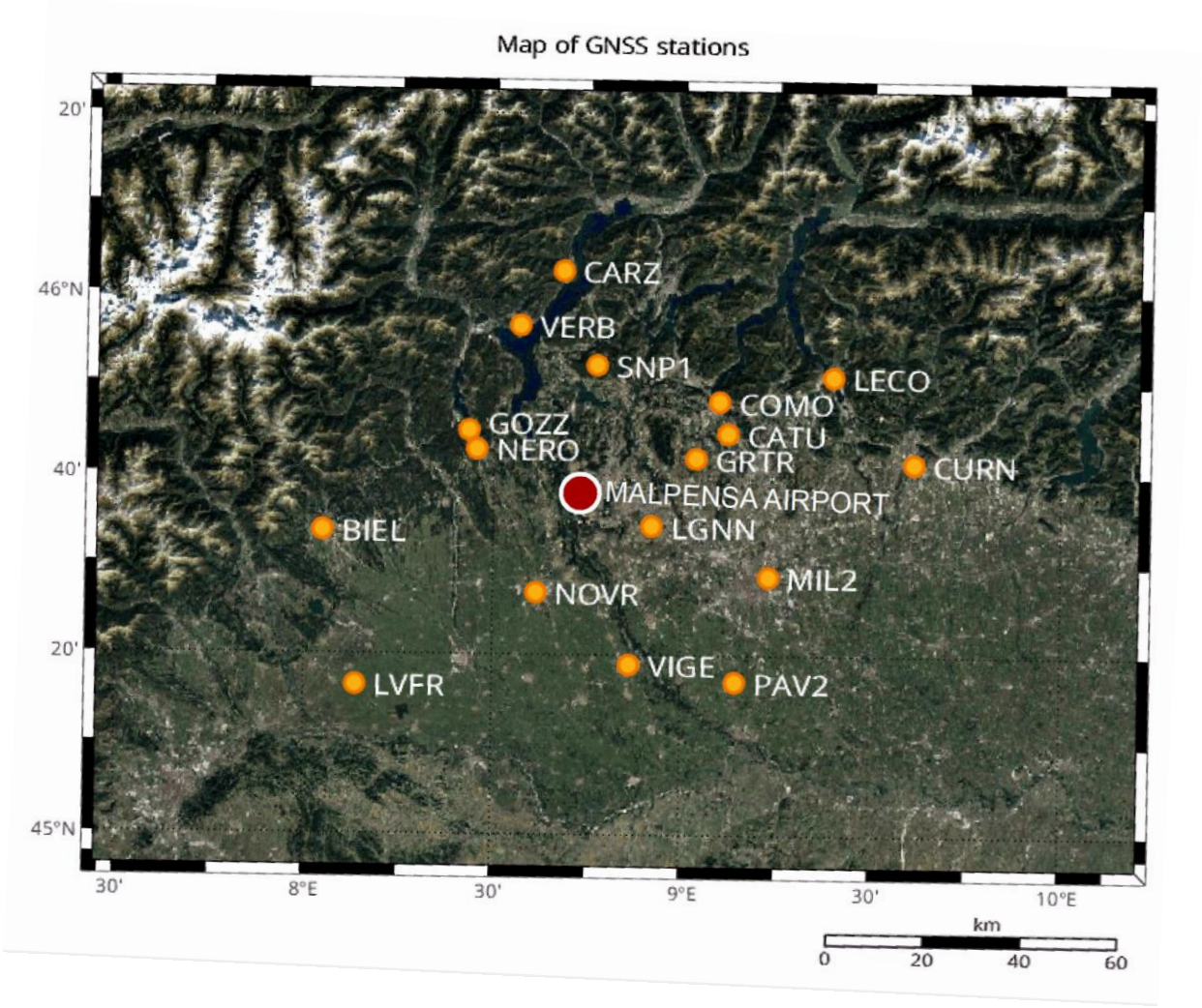


Figure 42 *Neighbouring GNSS Stations*

CHAPTER 6

RESULTS AND DISCUSSION

Adding additional information to a plot is of great importance to better explain the data contained in the plot. This has been the objective of this work, specifically, obtaining data containing global coverage of weather information from DarkSKY API, and displaying them using graphical objects of MATLAB on a plot. The plot (2D line plot) as produced by the goGPS software could contain information about any of the following parameters derived from processing of GNSS observation data:

- Zenith Tropospheric/Total Delay (ZTD), which represents the delay of a GNSS satellite signal, projected in the zenith direction above a receiver station.
- Zenith Wet Delay (ZWD) which represents the wet component of the ZTD
- Zenith Hydrostatic Delay (ZHD) representing the hydrostatic component of the ZTD and obtained from the difference between the ZTD and the ZWD
- Position coordinates (East, North or Height) of a monitoring station

As discussed in Section 5.4.1, the display of the weather data on the 2D line plots is divided into two stages, preparation and visualisation of the data. The procedures for the various stages have been discussed in Chapter 5 and the results are explained here.

6.1 Addition of the Meteo menu to figure

After the integration of the unit components with the goGPS system, the Meteo menu item (Figure 40) was added to the figure containing the plot of the monitoring data. The Meteo menu contains all the items necessary for adding weather parameters to the plot, being displayed in a first selected, first shown basis. The label of the sub items bear the corresponding weather parameter needed to show. For instance, the ‘Wind Direction’ sub item displays the wind direction weather-parameter on the plot. The menu also contains the ‘Show ALL’ sub item that can be used to display all

the weather parameters at the same time. The sub items of the Meteo menu have appropriate call back functions that are called to update the plot when they are selected. Thus, by flagging an item, the plot is updated with the corresponding data and by unflagging, they are removed from the plot. This is made possible by tag identifiers set on the function calls for the various graphical objects for displaying the weather parameters.

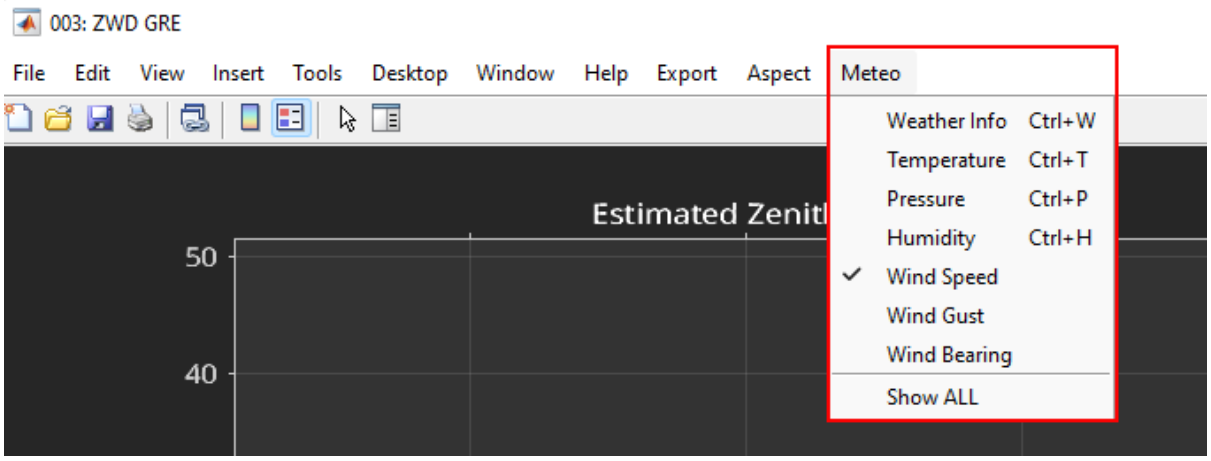


Figure 43 ***Meteo menu item and its sub-items***

6.2 Organisation of data in structure

After parsing the raw data obtained from the Data Source component, the raw data is organised in a structure to be accessed for creating the graphical objects. It is organised at an hourly rate for the number of days in which the analysis is made. For example in Figure 41 is shown the data structure for the first hour of a 7 days observation dataset. The number of fields might vary depending on the station being observed and the time. However, through trial and error, the following fields were always found to be present: time, icon, temperature, pressure, wind gust, wind bearing, humidity, wind speed, and dew point.

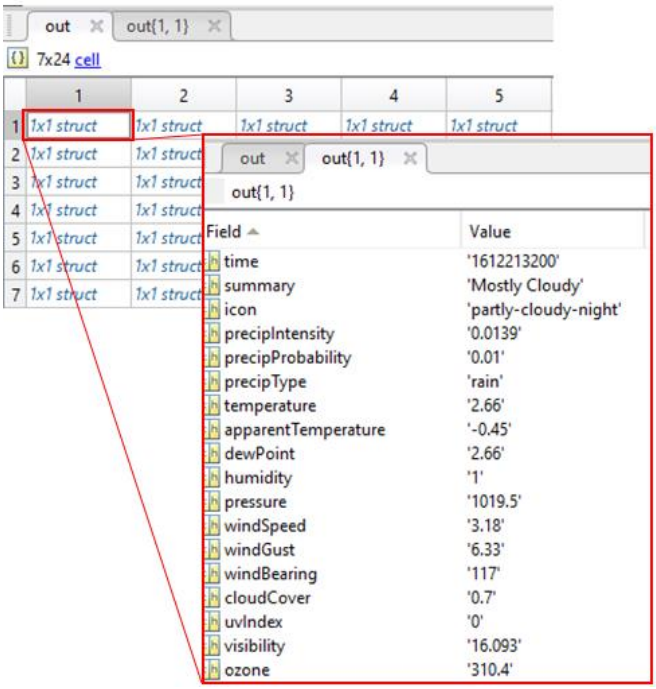


Figure 44 ***Organisation of data in structure***

6.3 Visualisation of output

By selecting an item from the Meteo menu, the values of the selected parameter are displayed over the plot for the same time period as the data on the plot. The corresponding graphical objects are shown on the plot. However, for the sake of readability, the pressure values are shown at 2-hour intervals while for the other parameters, they are spaced at an hour interval. All parameter values are horizontally aligned for easier reading. Also, for easier identification of the parameters, a label has been set in relation to each parameter on the left side of the display. Most likely weather conditions to be observed are cloudy, sunny, partial-cloudy, rainy, snowy, etc. Most common weather conditions have been listed in Appendix 1. However, additional weather conditions could be added to the list.

To remove a graphical object from the plot, one has to select the checked item. This finds the handle of the object using its tag identifier and the object is removed from the plot.

6.3.1 Display of data for less than 30-hour period

When a user displays weather information for a plot containing less than 30 hours of data, icons representing the weather conditions are shown at hourly intervals (Figure 42 and 43). The size of the icons are set relative to the dimensions of the axes object in which the data is displayed. However, they can be created or viewed at any zoom level of the plot, especially to see greater details for a short time period of data analysis.

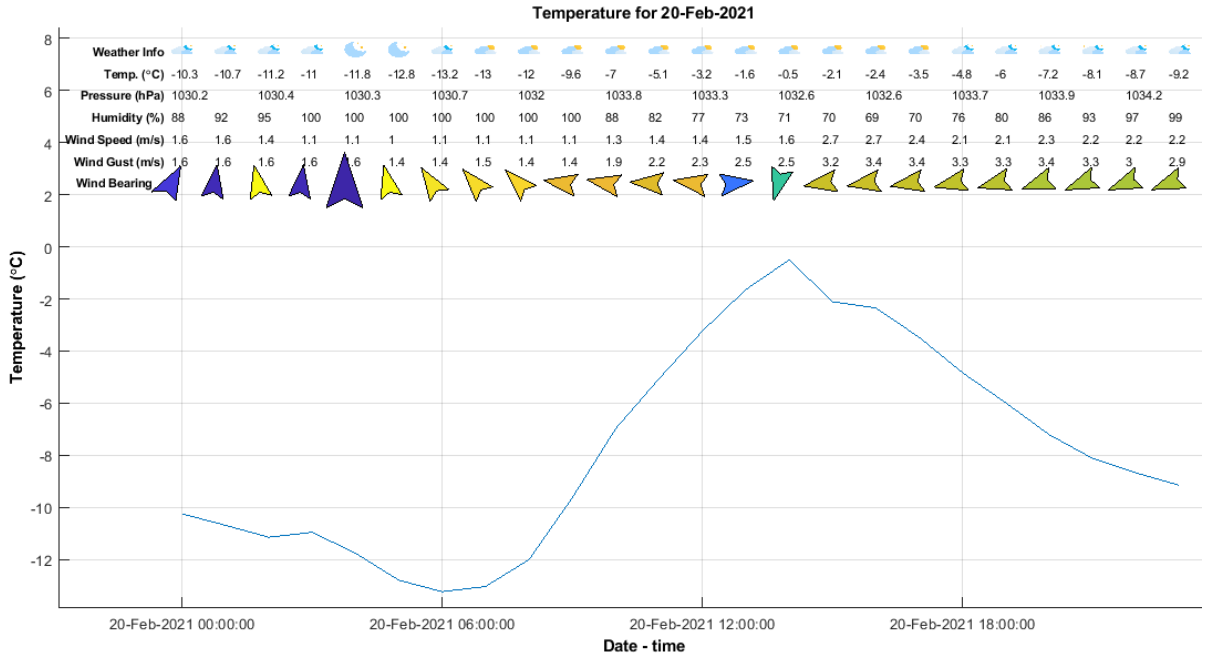


Figure 45 ***Display of data for less than 30-hour period***

6.3.2 Display of data for more than 30-hour period

When a user displays weather information for a plot containing more than 30 hours, of data, a rectangular patch is shown instead of the icons (Figure 44). This feature has been used in order to avoid clumsy or overcrowding of the weather icons and to enhance the appearance of the graph. A legend is added to the graph to explain the colours being used as face colour for the patches. The items in the legend vary depending on the number of weather conditions available for the epoch at the

station being analysed. The legend is also positioned in the best location so as to avoid it overlaying or intersecting with another graphical object.

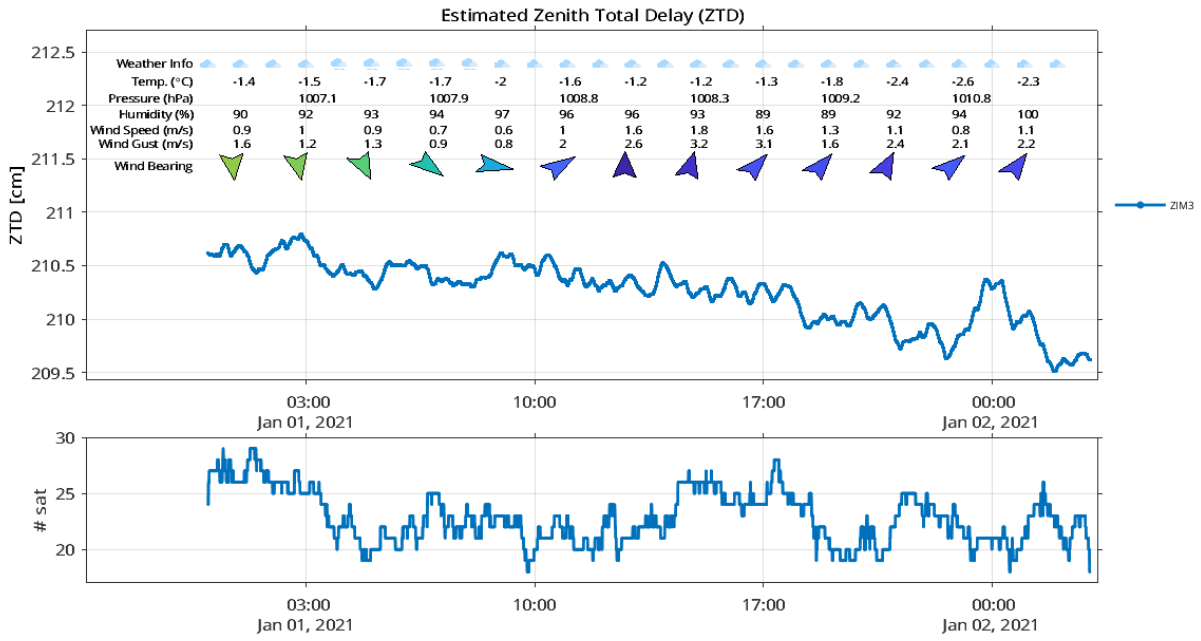


Figure 46 ***Display of data for less than 30-hour period on ZTD plot***

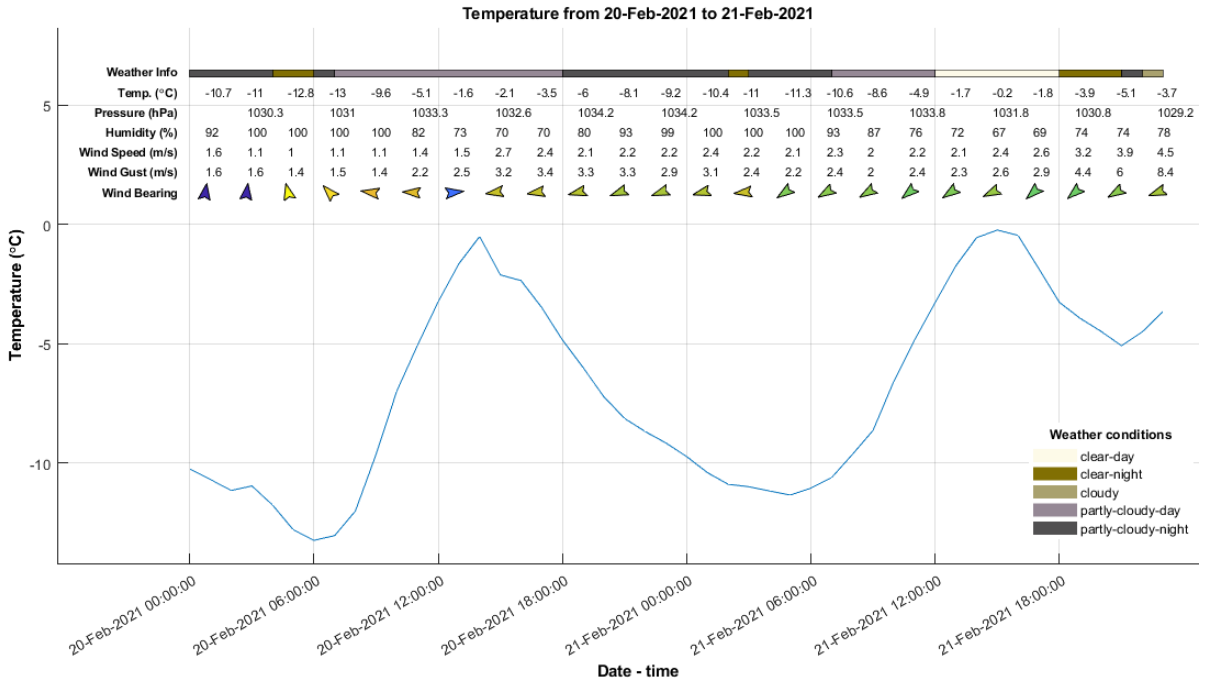


Figure 47 ***Display of data for more than 30-hour period***

6.4 Test results

The development of the program proceeded with monitoring of the command window for print of errors. Especially with the unit components building, variables and function return values were printed to the command window to ensure that they produce the correct values. Also, in the integration, it was ensured that a component calls the appropriate function or component that it interacts with for the satisfaction of specified requirements. In the performance testing, it was however realised that because a default font colour of white has been specified for the text graphical objects, they do not contrast with the light background of the plot when the object is to be displayed on the light aspect mode of goGPS. However, this has been communicated to GReD s.r.l. and efforts are being made to correct this. The display of the weather parameters runs correctly without any error being thrown to the command window, (Figure 45).

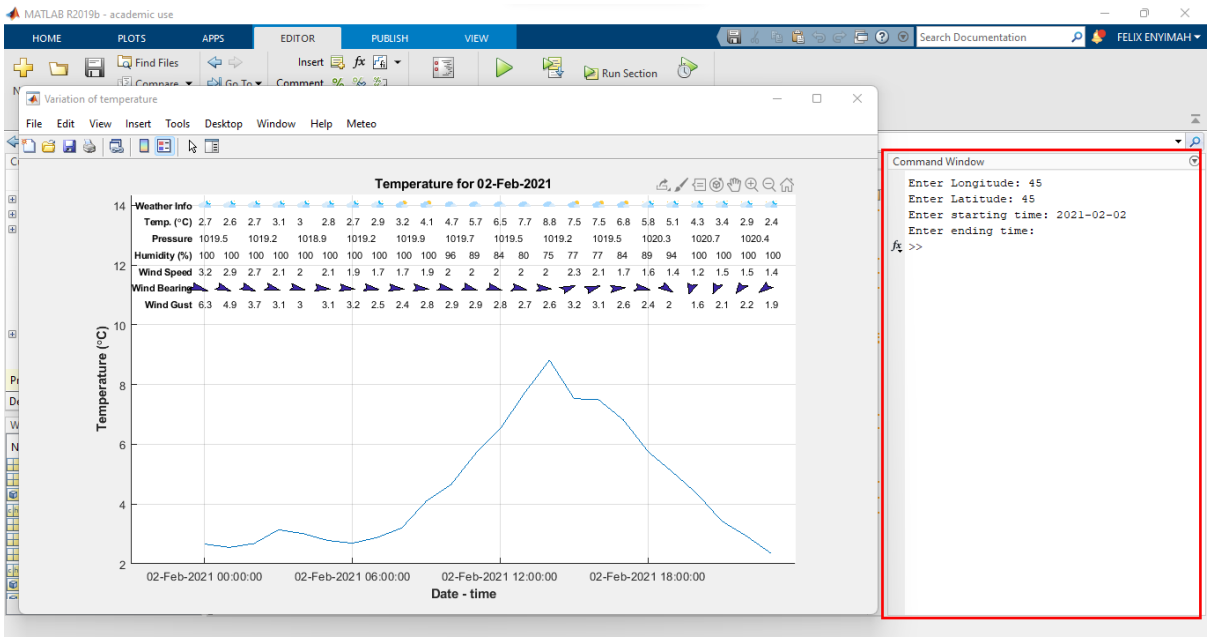


Figure 48 ***Monitoring of the command window for print of errors***

In the acceptance testing that was performed by GReD s.r.l., additional features were identified such as the need to provide a readme file for the work and to create a demo for the work. These were added to complete the project.

6.5 Meteorological Data File

The generated file contain meteorological data per hour for the station and period being analysed. The file is generated per day. Thus, data containing 3 days of observations will have 3 generated files. The file is subdivided into two main sections: the header section and the data record section. The header section contains global information for the entire file and is placed at the beginning, in accordance with the standards for RINEX file generation. The data record section contain the meteorological data for a particular station. The following are the meteorological observation data types that can be found in the data record section.

- Pressure (mbar)
- Dry temperature (˚C)
- Relative humidity (%)
- Wind azimuth (˚) from where the wind blows
- Wind speed (m/s)
- Rain increment (1/10 mm)

Figure 48 shows a sample meteorological file contents. Each observation record is read in the following order: epoch (yyyy-MM-dd HH:mm:ss), pressure, temperature, humidity, wind azimuth, wind speed and rain increment.

ZWD_194.21m

```
     3.03           METEOROLOGICAL DATA                     RINEX VERSION / TYPE
EXPORTED MET FILE FROM METEO_DATA MATLAB CLASS              COMMENT
ZWD_                                                        MARKER_NAME
     1    PR                                                # / TYPES OF OBSERV
  4429544.5026   626321.1633  4531501.0754      372.3413 PR SENSOR POS XYZ/H
                                                            END OF HEADER
 2021 07 13 00 00 00  1009.5   19.1    0.8  334.0    2.4    0.4
 2021 07 13 01 00 00  1009.2   18.7    0.8  321.0    2.1    0.4
 2021 07 13 02 00 00  1008.8   18.3    0.8  325.0    2.0    0.3
 2021 07 13 03 00 00  1008.4   17.4    0.8  328.0    1.9    0.5
 2021 07 13 04 00 00  1007.9   17.0    0.9  355.0    1.8    1.2
 2021 07 13 05 00 00  1007.7   16.7    0.9  213.0    1.9    2.9
 2021 07 13 06 00 00  1007.9   17.0    0.9  244.0    1.8    4.5
 2021 07 13 07 00 00  1008.1   17.2    0.9  278.0    1.7    4.1
 2021 07 13 08 00 00  1007.9   17.5    0.8  267.0    1.7    2.9
 2021 07 13 09 00 00  1008.0   18.2    0.8  214.0    1.6    2.0
 2021 07 13 10 00 00  1007.7   18.6    0.8  151.0    1.4    1.6
```

Figure 49 Sample Meteorological Data File Content

6.6 Case Study Analysis

From Figure 50 and considering the combined ZWD plot of all the stations, the wet path delay increased from midnight of July 12, 2021 with an approximate value of 18 cm to a higher value of about 25 cm at 21:00. The trend gradually decreased the following day to about 12 cm at a period around which the incident involving Emirates flight EK205 happened.

From the weather information displayed on the plot, the direction of the wind changed suddenly from south-east to the northern direction in the Airport, in the direction of traverse of the hail progression seen from the radar graphs in Figure 41.

The wind speed also increased abruptly from the lower value of 1.9 m/s at the mid-night of the day of the incident to a higher value of 10.2 m/s. There was also a decrease in the values of the humidity, pressure and temperature during the incident period. The trend of these parameters are in favour of the heavy frontal wind that was traversing the Airport area. As explained in Section 2.2.3, for instance, the reduced values of humidity during the incidence period is due to the wind moving air masses to the area, mixing with water vapour evaporating from the surfaces resulting in a reduced amount of water vapour in the atmosphere. In addition, the moving air mass indicate a loss of air column in the area resulting in the reduced air pressure seen during the incident period, as explained in Section 2.3.

Moreover, these were accompanied by a higher increase in wind gust from a lower value of 2.2 m/s at 3:00 AM of the previous day to a higher value of 10.2 m/s at the period of the incidence. Moreover, the weather condition is shown to be rain as deduced from the radar maps using the classification of radar colours in Table 5. This might also be hail as reported in news, as rain and hail belong to the same category in Table 5.

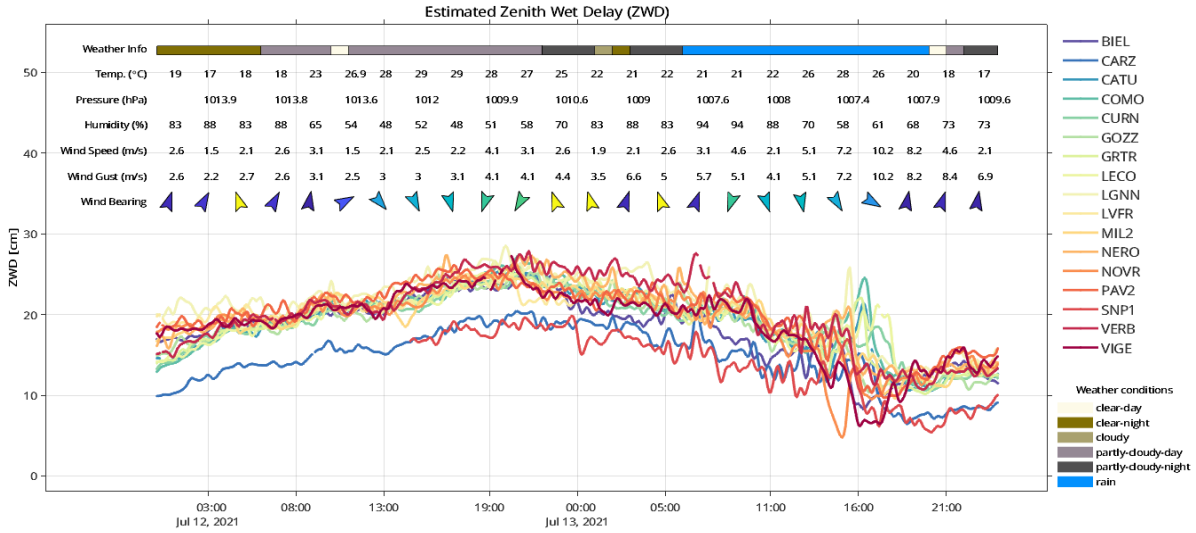
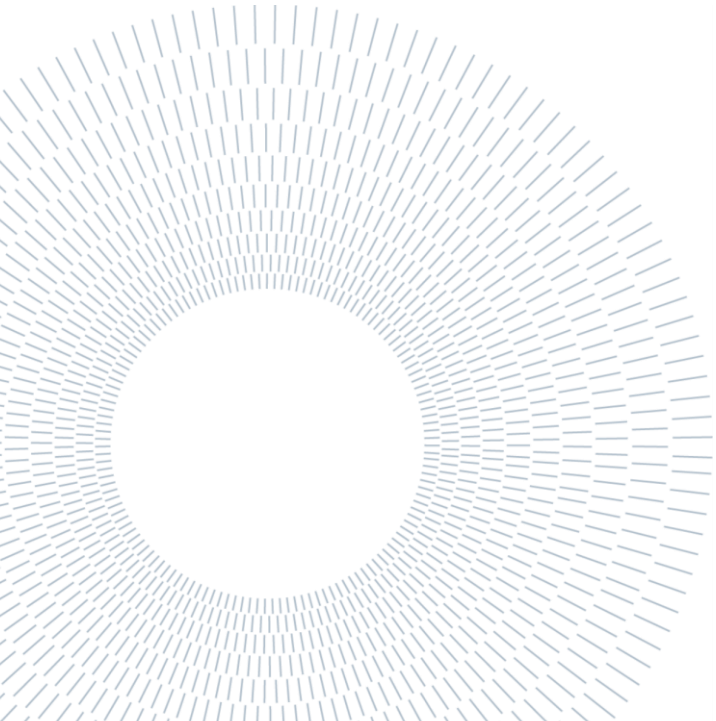


Figure 50 Display of Weather Information on goGPS ZWD Plot



CHAPTER 7

CONCLUSION AND RECOMMENDATION

7.1 Conclusion

The use of GNSS for the remote sensing of atmospheric water vapour has found applications in the field of meteorology. The satellite signals of the GNSS space vehicle traverses the various layers of the atmosphere, having many encounters with the components of the atmosphere, mainly the electrically charged ionosphere and the electrically neutral troposphere. Subsequently, the signal experiences a delay before reaching the receiver. By modelling and estimating this delay, it is possible to estimate the amount of precipitable water vapour in the atmosphere which can be useful for meteorological purposes.

goGPS is one of the software that can be used to process the GNSS observation data to derive an estimate of the amount of water vapour in the atmosphere. It has the capability to produce time series plots of station coordinates, ZWD, ZHD, ZTD etc. While analysing the plots produced by the software, it will be very useful to have an additional information of the meteorological conditions of the atmosphere or events at the observation station that will serve as an explanatory tool for the plots produced.

Matlab, the programming language used for the development of goGPS provides the capability to use graphical objects to add extra information to a plot to achieve this purpose. This provision has been exploited using the meteorological data from DarkSKY API. The information can be displayed for a day or multiple-days observation dataset, with the option for a user to display variable parameters such as temperature, pressure, humidity, and wind information. It is hoped that the extra information that can be added will provide an answer to the needs of a user in interpreting the behaviour of the estimated GNSS water vapour parameter and position coordinates of a monitoring station.

7.2 Recommendation

It is recommended that

- The component will be revised to update the ylim of the graph when a graphical object is deleted.
- The spacing between the objects will be updated when an object is deleted.
- The only limitation is that attempting to display weather information for more than 3 days makes the appearance clumsy. For better visualisation or if one wants to make plots for presentation purposes, it is recommended to display the weather info for multiple days to detect when a change in environmental variable occurs and produce the graph for the particular day in which the phenomenon occurred for a more detailed analysis and nicer appearance of the displayed graphical objects.
- It is also recommended that a Microsoft HTML Help Workshop window accessible from the goGPS GUI menu bar be developed to guide a goGPS user.

REFERENCES

Anon, (2021), *Heat capacity, the ocean, and our weather*, Available at: https://geo.libretexts.org/@go/page/795. Accessed December 18, 2021.

Anon., (2022), *"Clouds, National Meteorological Library and Archive Fact sheet 1 — An introduction to clouds"*, https://www.metoffice.gov.uk/binaries/content/assets/metofficegovuk/pdf/research/library-and-archive/library/publications/factsheets/factsheet_1-clouds.pdf. Accessed: March 22, 2020.

Barindelli, S., Realini, E., Venuti, G., Fermi, A., Gatti, A., (2018), 'Detection of water vapor time variations associated with heavy rain in northern Italy by geodetic and low-cost GNSS receivers', *Earth, Planets and Space*, Vol. 70, Num. 28, 18 pp.

Bevis, M., Businger, S., Herring, T., A., Rocken, C., Anthes, R., A., Ware, R., H., (1992), 'GPS Meteorology: Remote Sensing of Atmospheric Water Vapour Using the Global Positioning System', *Journal of Geophysical Research*, Vol. 97, No. D14, pp. 15787-15801.

Biagi, L., (2009), *"I fondamentali del GPS"*, Geomatics Workbooks, Vol. 8, 246 pp., http://www.geolab.polimi.it/wp-content/uploads/2019/05/GW8.pdf.pdf

Biagi, L., Grec, F. C., and Negretti, M., (2016), "Low-Cost GNSS Receivers for Local Monitoring: Experimental Simulation, and Analysis of Displacements", Sensors, 16 pp., https://doi.org/10.3390/s16122140

Davis, J., L., (1985), 'Geodesy by radio interferometry: Effects of atmospheric modelling errors on estimates of baseline length', *Radio Science*, Vol. 20, Num. 6, pp. 1593-1607.

Fermi, A., Realini, E., Venuti, G. (2018), "The impact of relative and absolute GNSS positioning strategies on estimated coordinates and ZWD in the framework of Meteorological applications." *Applied Geomatics,* Vol. 11, pp. 25–38, https://doi.org/10.1007/s12518-018-0234-2 .

Flores, F., Rondanelli, R., Díaz, M., Querel, R., Mundnich, K., Herrera, L. A., Pola, D., and Carricajo, T., (2013), "The Life Cycle of a Radiosonde", *Bulletin of the American Meteorological Society* 94, 2, pp. 187–198.

Haby, Jeff. "What do the colours on radar mean?", https://www.theweatherprediction.com/basic_weather_questions/radar.html, Accessed: March 31, 2022.

Herring, T., A., (1992), 'Modelling Atmospheric Delays in the Analysis of Space Geodetic Data', *Netherlands Geodetic Commission Publications on Geodesy*, No. 36, pp. 157-167.

Lehman, M. M. (1980). 'Programs, Life Cycles, and Laws of Software Evolution', *Proceedings of the IEEE*, Vol. 68, Number 9, pp. 1060 – 1076, *doi*:*10.1109/proc.1980.11805*.

Linacre, E. and Geerts, B., (1997), "*CLIMATES AND WEATHER EXPLAINED",* Taylor & Francis e-Library, 2003, 432 pp.

Mendes, V. B. (1999), *"Modelling the neutral-atmosphere propagation delay in radiometric space techniques"*, Ph.D. dissertation, Department of Geodesy and Geomatics Engineering Technical Report No. 199, University of New Brunswick, Fredericton, New Brunswick, Canada, 353 pp.

Niell, A., E., (1996), 'Global mapping functions for the atmosphere delay at radio wavelengths', *Journal of Geophysical Research*, Vol. 101, NO. B2, pp 3227-3246.

Owens, J., C., (1967), "Optical Refractive Index of Air: Dependence on Pressure, Temperature and Composition," *Applied Optics*, Vol. 6, pp. 51-59.

Rocken, C., Ware, R., Van Hove, T., Solheim, F., Alber, C., Johnson, J., (1993), 'Sensing Atmospheric Water Vapour with the Global Positioning System', *Geophysical Research Letters*, Vol. 20, No. 23, Pages 2631-2634.

Romero, I., (2020), '*RINEX: The Receiver Independent Exchange Format, Version 3.05*', Germany, 91pp.

Saastamoinen, J., (1973), “Contributions to the theory of atmospheric refraction, Part II-Refraction corrections in satellite geodesy”, *Bulletin Geodesique*, Number 107, pp 13 - 34

Sangiorgio, M., Barindelli, s., Biondi, R., Solazzo, E., Realini, E., Venuti, G., and Guariso, G., (2019) “Improved Extreme Rainfall Events Forecasting Using Neural Networks and Water Vapor Measures”, *International conference on Time Series and Forecasting-2019, Granada,* 8 pp.

Smith, C., D., Nicholson, N., Skone, S. and Strong, G., S., (2008), ‘Evaluating regional atmospheric water vapour estimates derived from GPS and short-range forecasts of the Canadian Global Environmental Multiscale model in southern Alberta’, *Atmosphere-Ocean*, 46:4, 455-471, DOI: 10.3137/ao.460406

Tang, A., and Vliet, H. V., (2012), “Design Strategy and Software Design Effectiveness”, *IEEE Software*, vol. 29, no. 1, pp. 51-55, doi: 10.1109/MS.2011.130.

Teke, K., Böhm, J., Nilsson, T., Schuh, H., Steigenberger, P., Dach, R., Heinkelmann, R., Willis, P., Haas, R., García-Espada, S., Ichikawa, T., H., R., Shimizu, S., (2011), ‘Multi-technique comparison of troposphere zenith delays and gradients during CONT08’, *Journal of Geodesy*, Num. 85, pp 395-413.

Thayer, G., D., (1974), ‘An improved equation for the radio refractive index of air’, *Radio Science*, Vol. 9, Number 10, pages803-807.

Trenberth, K.E., P.D. Jones, P. Ambenje, R. Bojariu, D. Easterling, A. Klein Tank, D. Parker, F. Rahimzadeh, J.A. Renwick, M. Rusticucci, B. Soden and P. Zhai, (2007), ‘Observations: Surface and Atmospheric Climate Change’, *Climate Change 2007: The Physical Science Basis*, Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M. Tignor and H.L. Miller (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Vliet, H., V., (2007), *'Software Engineering Principles and Practice'*, Wiley, 752 pp.

Warren, S., G., (2019), 'Optical properties of ice and snow', *Philosophical Transactions, Royal Society publishing*, Vol. 377, No. 2146: 20180161. http://dx.doi.org/10.1098/rsta.2018.0161

Yangtze, A., (2021), 'Weather phenomenon icon', Available at https://www.figma.com/community/file/884628404763738829, Accessed: December 18, 2021.

APPENDIX I

ICONS AND COLOURS

NAME	ICON	COLOUR CODE
Blizzard		#b1fb32
Blizzard-night		#466e02
Breezy		#a1b57e
Breezy-snow		#3ba19b
Clear-day		#fdfae8

Clear-night		#816f03
Cloudy		#a9a16e
Cloudy-day		#d6d2b7
Cloudy-night		#67634c
Drizzle		#4c6762
Drizzle-day		#5a8e85

Drizzle-night		#10574a
Dust		#9ba1a1
Earthquake		#ec9558
Fire		#f4752a
Flood		#70edab
Flurries		#707271

Fog		#dce6e1
Fog-day		#adb1d2
Fog-night		#535355
Hail		#b0a0b5
Hail-day		#87728e
Hail-night		#40184d

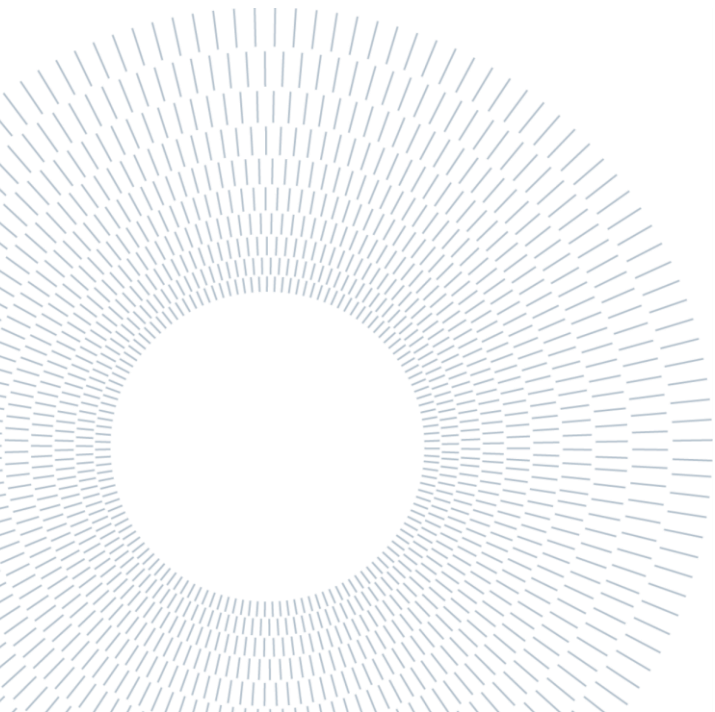
Haze		#a445e1
Heavy-rain		#1f9c9b
Heavy-rain-night		#156f6e
Heavy-sleet		#66b960
Heavy-snow		#018788
Light-rain		#8384c4

Light-rain-night		#30328d
Light-sleet		#f3bddd
Light-snow		#acf7f3
Overcast		#587270
Partly-cloudy		#d9bfda
Partly-cloudy-day		#958895

Partly-cloudy-night		#515051
Rain		#0290fd
Rain-day		#88c9fa
Rain-night		#01335a
Showers		#d1dee7
Showers-day		#829cb0

Sleet		#da6c93
Sleet-day		#afa2a7
Sleet-night		#665e61
Smoke		#c1bb8f
Snow		#2e3790
Snow-night		#232748

Sunny		#fbec02
Tornado		#975051
Wind		#57dd96



APPENDIX II

IMPLEMENTED FUNCTIONS

1.0 Function to get the time zone of a given location

```
function [outActual] = getTimeZone(latitude, longitude, time)
    baseUrl = 'https://DarkSKY.net/details/';
    httpsUrl = strcat(baseUrl,latitude,',',longitude,'/',time, '/si12/en/');
    webHTML = webread(httpsUrl);
    neededInfo = extractAfter(extractBefore(webHTML, 'units = '), 'tz_offset = ');
    outActual = replace(neededInfo, ',', '');
    outActual = str2double(outActual);
end
```

2.0 Function to get the conditions of the weather at the specified location and time

This output results vary depending on the location, time and the conditions of the atmosphere at the specified time.

```
function [structArray] = getWeatherConditions(latitude, longitude, time)
    baseUrl = 'https://DarkSKY.net/details/';
    httpsUrl = strcat(baseUrl,latitude,',',longitude,'/',time, '/si12/en/');
    webHTML = webread(httpsUrl);
    neededInfo = extractAfter(extractBefore(webHTML,'cityName'), 'conditions = ');
    out = extractBetween(neededInfo, '{', '}');
    outActual = replace(out, ':', ',');
    outActual = replace(outActual, '"', '');
    outActual = replace(outActual, '-', '_');
    structArray = struct([]);
    for i = 1:length(outActual)
        tmp = strsplit(outActual{i}, ',');
        fieldValue = tmp(2:2:end);
        fields = tmp(1:2:end);
        structArray{i} = cell2struct(fieldValue, fields, 2);
    end
end
```

3.0 Function to get the weather data needed for making the plots

```
function [structArray] = getWeatherInfo(latitude, longitude, startTime, endTime)
    baseUrl = 'https://DarkSKY.net/details/';
    structArray = struct([]);
    dayNum = 0;
    if isempty(endTime)
        endTime = startTime;
    end
    startTime = datetime(startTime);
    endTime = datetime(endTime);
    for i = startTime:endTime
        dayNum = dayNum+1;
        time = datestr(i);
        httpsUrl = strcat(baseUrl,latitude,',',longitude,'/',time, '/si12/en/');
        webHTML = webread(httpsUrl);
        neededInfo = extractAfter(extractBefore(webHTML,'startHour'), 'var hours =
');
        out = extractBetween(neededInfo, '{', '}');
        outActual = replace(out, ':', ',');
        outActual = replace(outActual, "", '');
        outActual = replace(outActual, '{azimuth', 'azimuth');
        outActual = replace(outActual, 'solar,', '');
        for i = 1:length(outActual)
            tmp = strsplit(outActual{i}, ',');
            fieldValue = tmp(2:2:end);
            fields = tmp(1:2:end);
            structArray{dayNum, i} = cell2struct(fieldValue, fields, 2);
        end
    end
end
```

4.0 Function to get the elevation of a station

```
function [stat_elev] = getTimeZone(latitude, longitude, time)
    latitude = string(latitude / pi * 180);
```

```
    longitude = string(longitude / pi * 180);
    baseUrl = 'https://api.opentopodata.org/v1/etopo1?locations=';
    httpsUrl = strcat(baseUrl,latitude,',',longitude);
    webHTML = webread(httpsUrl);
  stat_elev = webHTML.results.elevation;
end
```

5.0 Function to export the meteorological parameters to a RINEX file

```
function getMeteoData(latitude, longitude, startTime, endTime, file_name)
    % Get the weather data
    out = getWeatherInfo(latitude, longitude, startTime, endTime);
    % Get the elevation of the station
    stat_elev = getGeodHeight(latitude, longitude);
    % Get the size of the data retrieved from the web
    [numDays, numHoursInADay] = size(out);
    % Initialise the needed variables
    temperature = zeros(numDays * numHoursInADay, 1);
    pressure = zeros(numDays * numHoursInADay, 1);
    humidity = zeros(numDays * numHoursInADay, 1);
    windSpeed = zeros(numDays * numHoursInADay, 1);
    windBearing = zeros(numDays * numHoursInADay, 1);
    rainIntensity = zeros(numDays * numHoursInADay, 1);
    allHours = zeros(numDays * numHoursInADay, 1);
    % Fill the variables with the actual values obtained from the web
    for i=1:numDays
        for j = 1:numHoursInADay
            temperature(24*(i-1)+j) = str2double(out{i,j}.temperature);
            pressure(24*(i-1)+j) = str2double(out{i,j}.pressure);
            humidity(24*(i-1)+j) = str2double(out{i,j}.humidity);
            windSpeed(24*(i-1)+j) = str2double(out{i,j}.windSpeed);
            windBearing(24*(i-1)+j) = str2double(out{i,j}.windBearing);
            rainIntensity(24*(i-1)+j) = str2double(out{i,j}.precipIntensity);
```

```
            allHours(24*(i-1)+j) = str2double(out{i,j}.time);
        end
    end
    % Convert the all the hours in the days to matlab time
    allHours = GPS_Time(uint32(allHours), 0, 1, 1);
    mds = Meteo_Data;
    mds.setName(file_name);
    mds.setRinType(3);
    mds.setData(allHours, [pressure temperature humidity windBearing
        windSpeed rainIntensity], [Meteo_Data.PR Meteo_Data.TD
        Meteo_Data.HR Meteo_Data.WD Meteo_Data.WS Meteo_Data.RR]);
    mds.setValid(true);
    [x, y, z] = geod2cart(latitude, longitude, stat_elev);
    mds.setCoordinates([x, y, z]  );
    mds.export(strcat(file_name, '${DOY}.${YY}m'));
end
```

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would first of all like to thank God for sustaining me throughout this journey of pursuing a Laurea Magistrale in Italy. I am also very grateful to my supervisor, Professor Giovanna Venuti, for the opportunity given me to do the work and her patience with me. I am also thankful to my Co-Supervisor for his guidance in doing the work. I am also thankful to Professor Ludovico Biagi. Lastly, I thank my family and friends for always being there for me. I am forever grateful to you all.