Ing. Patrick Petschauer, BSc

Design and implementation of an antenna tracking system for a Ka band satellite pico terminal

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Electrical Engineering and Business

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Franz Teschl

Institute of Communication Networks and Satellite Communications

Graz, May 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.
The text document uploaded to TUGRAZonline is identical to the present masters thesis.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.
Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Date/Datum	Signature/Unterschrift

Abstract

The pico terminal is a compact and transportable satellite ground station for educational purposes. With an adjustable antenna dish, the pico terminal is used in laboratory exercises at Graz University of Technology (TUG) in combination with radio frequency (RF) equipment to illustrate and analyze the signal behavior of a satellite communication link. Laboratory exercises in the past and a pandemic have shown the limits of the pico terminal in terms of handling and integration in virtual teaching.

This thesis presents upgrades to the instrument, including hardware and software components, that allow for electrically driven antenna pointing as well as for antenna tracking. In addition to the tracking functionality based on the two-line element set and an integration into virtual teaching with remote access and control options, manual antenna alignment is simplified.

In order to implement the set goals, the special design of the antenna unit must be considered to determine the effective pointing direction. As no documents are available that specify the exact geometry of the antenna, the antenna offset angle is measured and an alignment correction algorithm is implemented. Additional hardware components, some specially designed and manufactured, as well as a software implementation for graphically supported operation are part of this work.

To verify the functionality, different test methods are described and carried out. In addition to component tests to evaluate calibration times, the satellite tracking behavior over a time period of several hours is evaluated.

Test results confirm that satellites can be tracked with the modified system. In addition, manual axis adjustments are still possible, but with much less effort. The modified system illustrates one possible method for satellite tracking and is used to impart knowledge based on practical application examples. The user interface and remote access to the system offer high added value, especially in virtual lessons.

Kurzfassung

Das Pico-Terminal ist eine kompakte und transportable Satelliten-Bodenstation für Bildungszwecke. Mit einer ausrichtbaren Parabolantenne wird das Pico-Terminal in Laborübungen an der Technischen Universität Graz (TUG) in Kombination mit Hochfrequenzgeräten (RF) verwendet, um das Signalverhalten einer Satellitenkommunikationsverbindung zu veranschaulichen und zu analysieren. Laborübungen in der Vergangenheit und eine Pandemie haben die Grenzen des Pico-Terminals in Bezug auf die Handhabung und Integration in den virtuellen Unterricht aufgezeigt.
In dieser Arbeit werden Verbesserungen des Labor-Instruments in Bezug auf Hardware- und Softwarekomponenten vorgestellt und implementiert, die eine elektrische Ausrichtung der Antenne sowie Antennen-Tracking ermöglichen. Abgesehen von dem auf *Two Line Elements* basierenden Antennen-Tracking und der Integration des Systems in den virtuellen Unterricht mit Fernzugriffs- und Steuerungsoptionen, wird die manuelle Antennenausrichtung vereinfacht.
Um die gesetzten Ziele umzusetzen, muss das spezielle Design der Antenneneinheit berücksichtigt werden, um die effektive Antennenausrichtung bestimmen zu können. Da keine Dokumente vorliegen, welche die genaue Geometrie der Antenne spezifizieren, wird der Antennen-Offsetwinkel gemessen und ein Algorithmus zur Ausrichtungskorrektur der Antenne implementiert. Zusätzliche Hardwarekomponenten, von denen einige speziell entwickelt und hergestellt werden, sowie eine Softwareimplementierung für den grafisch unterstützten Betrieb sind Teil dieser Arbeit.
Zur Überprüfung der Funktionalität werden verschiedene Testmethoden beschrieben und durchgeführt. Zusätzlich zu Komponententests zur Bewertung der Kalibrierungsdauer wird das Tracking eines Satelliten über einen Zeitraum von mehreren Stunden ausgewertet.
Testergebnisse bestätigen, dass Satelliten mit dem modifizierten System getrackt werden können. Darüber hinaus sind manuelle Achsanpassungen weiterhin möglich, jedoch mit viel weniger Kraftaufwand.
Das modifizierte System veranschaulicht eine mögliche Methode Satelliten zu tracken und wird verwendet, um Wissen basierend auf praxisbezogenen Anwendungsbeispielen zu vermitteln. Die Benutzeroberfläche und der Fernzugriff auf das System bieten einen hohen Mehrwert, insbesondere im virtuellen Unterricht.

Acknowledgement

This master's thesis was written in the 2020/21 academic year at the Institute for Communication Networks and Satellite Communication at Graz University of Technology. At this point I would like to thank everyone who supported me during my studies.

Danksagung

Die vorliegende Masterarbeit wurde im Studienjahr 2020/21 am Institut für Kommunikationsnetze und Satellitenkommunikation an der Technischen Universität Graz durchgeführt. An dieser Stelle möchte ich mich besonders bei meinem Betreuer Herrn Assoc.Prof. Dipl.-Ing. Dr.techn. Franz Teschl für die großartige Zusammenarbeit während der Entstehungsphase dieser Arbeit bedanken.

Des Weiteren möchte ich mich bei meinen Freunden und Arbeitskollegen bedanken, die mich während des Studiums immer wieder motiviert haben.

Ein herzliches Dankeschön geht an meine Eltern, Ingrid und Hermann, für die jahrelange Unterstützung in allen Lebensbereichen und für die Geduld, wenn meine Antworten zu Studienfragen recht kurz ausgefallen sind.

Ein großer Dank geht auch an meinen Bruder, der mir bei diversen Projekten immer zur Seite steht und an Tante Anni, die mich von klein auf immer begleitet und unterstützt hat.

Ein ganz besonderer Dank geht an meine Freundin, die immer zu mir steht und mich bei Rückschlägen immer wieder motiviert. Danke für deine unglaubliche Geduld und dein Verständnis.

Contents

Appendices

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Satellites orbit the Earth at different heights and speeds. Since the first start of Sputnik 1 in 1957, satellites have been continuously transported into space for applications in a wide variety of areas such as research or broadcasting. With end of 2020, around 3,400 satellites were in different orbits [1] and the number of satellites will increase rapidly in order to continue networking the world's population in the future and also to be able to offer satellite services in remote locations.

To enable short latency times, the distance between a satellite and the Earth should be as short as possible. As a result, many smaller low earth satellites are used in so-called mega-constellations, like the Starlink project. Starlink is a satellite network to provide high-speed, low-latency broadband connectivity across the globe [2]. The distance between a low earth satellite and the Earth is small, compared with other orbits such as the geostationary orbit. The downside in this case is the coverage area of a single satellite due to the short distance. To cover all regions, a large number of satellites is required to enable seamless operation.

In contrast to the geostationary orbit, in which satellites remain at almost the same position viewed from the Earth, low earth satellites orbit the Earth at higher speeds and shorter orbital times. The position of a low earth satellite changes significantly viewed from the Earth and is only visible for a short amount of time. To enable data exchange, the antenna of the ground station must be continuously adjusted to the satellite position.

The pico terminal used in this work was originally designed in the 90s of the last millennium and used for research projects at Graz University of Technology. For some years now, the pico terminal has been used mainly for training purposes in the form of a laboratory exercise on the subject of "antennas". The focus of this exercise is to understand antenna characteristics, the link budget and the principles of antenna pointing. In combination with radio frequency devices, the signal behavior can be observed during antenna alignment. Three different axes can be adjusted to align the antenna dish in the desired direction to point a satellite. These adjustments are done manually by hand via a mechanical system.

1.1 Motivation

The pico terminal as shown in Figure 1.1 is a compact and transportable instrument consisting of an offset parabolic antenna. The pico terminal is used in laboratory exercises to foster the understanding of important aspects of antenna pointing and antenna tracking. One part of this exercise is to locate a given satellite and to point it with the antenna. The laboratory exercise usually takes place on site on the university campus and offers students the opportunity to gain experience with the alignment of an antenna. Due to the *COVID-19* pandemic, many courses at the university had to be held in virtual form, including the antenna laboratory exercise. More details to the pico terminal and its components are explained in Section 2.2.1.

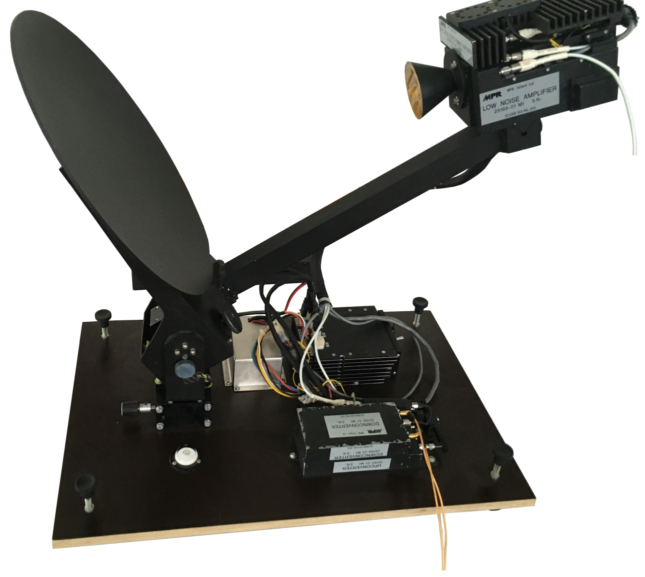


Figure 1.1: Pico terminal as used in the lab exercises. The antenna dish can be manually aligned in all directions.

Both types of lectures, on-site and virtual, have advantages on the one hand, but also disadvantages on the other. In Table 1.1 both types of lectures are compared.

Table 1.1: Comparison between on-site and virtual laboratory exercise. Both types offer different advantages (+) and disadvantages (-).

	On-site laboratory exercise
+	Hands-on feeling
+	Getting a good overview in antenna alignment
-	Focus only on one part (e.g. protocol, axis adjustment or signal behavior)
-	Axis adjustments require a lot of effort

	Virtual laboratory exercise
+	Prepared video material with detailed explanation
-	No possibility to adjust the axes

The author's accumulated experience with both types of lectures has shown that the hands-on feeling (or practical experience) during the lecture on-site is most valuable. The changes in the measured signal when the antenna dish is aligned slightly in different directions can best be transmitted on site. It is also easier to follow the required steps when standing in front of the device. The overview of the entire test setup during the exercise is very helpful in this case. One of the challenges during the on-site exercise is not to lose the sight of focus. Writing a protocol, adjusting the axes and observing the signal at the same time is almost impossible. In addition, it is very difficult to adjust the axes by hand. Adjusting the antenna in the vertical direction requires a lot of effort.
The experiences during the virtual lecture have shown, that it is easier to follow the actual slide presented via screen sharing. Previously recorded video clips of the antenna alignment were also shared during the lecture to understand the discussed topics. This is also very helpful but compared to an on-site lecture it is more difficult to follow the whole process. It is not possible to adjust the antenna dish remotely and to see the corresponding effects on the signal. This can only be discussed theoretically.

1.2 Objective

The objective of this thesis is to expand the existing hardware into a complete system that offers the following options:

- Keeping the hands-on feeling
- Easier axis adjustments
- Axis adjustments and simultaneous observation of the signal behavior
- Possibility of remote control (virtual lectures)

The hands-on feeling should be retained even after the adaptation to transmit the knowledge associated with it. The system should also provide a possibility of an easy axis adjustment which also supports to keep the focus on all relevant things at the same time. To make future virtual lectures more attractive and to enable interactions during the exercises, the pico terminal should provide a possibility of remote control.

1.3 Approach

A key topic concerns the axis adjustment. In this case, motor drives are provided so that the axes of the pico terminal can be easily adjusted. In combination with a control unit and a graphical user interface (GUI), a basis for further operating options is created. On the one hand, axes can be adjusted manually on-site via a manual control panel and on the other hand it is possible to send commands to the motor drives via the GUI. An automatic mode based on online satellite data is planned to point a desired satellite without any effort. Remote access should offer the possibility of actively interacting with the pico terminal in virtual teaching.

1.4 Benefit

The planned adaptation also offers advantages in terms of satellite tracking. In this case, the satellite is repointed at cyclical intervals. This might be a useful option for further research projects. The integration of online satellite data for determining the position of a satellite in orbit offers additional added value. In this case, the students are shown a method of how the current satellite positions are continuously updated.

1.5 Outline

The rest of this thesis is structured as follows:

- Chapter 2 reviews the literature with fundamental concepts for better understanding of the used methodologies, introduces the pico terminal with its special antenna design and highlights states of previous research in the area of antenna alignment and tracking.

- Chapter 3 describes the procedure for determining the antenna offset parameter, which is unknown due to missing documentation. In addition, a correction model is explained. This model is necessary to overcome alignment offsets when changing the skew axis of the pico terminal. Apart from the hardware and software implementation to motorize the instrument and to provide a control interface, also test methods are explained in this chapter. These test methods are used to validate the functionality of the pico terminal.

- Chapter 4 describes the test procedures to evaluate the objectives and discusses the results of the observations.

- Chapter 5 summarizes the research and gives an overview of future work in this area.

Chapter 2

Literature Review

In the first part of this chapter the fundamental aspects in satellite communication and antenna pointing, which are used in the methodology part of this thesis, are reflected. The basic principles of antennas and wave propagation as well as different types of satellites are mentioned. For position and orientation determination different tools are highlighted and possible tracking systems are explained. Beacon signal and the basics of the spectrum analyzer as well as the basic principle of stepper motors are explained. The second part of this chapter reflects the status quo. The initial situation with the basic principle of the pico terminal, a literature research of available antenna tracking systems and related works are covered.

2.1 Fundamentals

2.1.1 Antennas and wave propagation

In the following section, parabolic antennas are discussed in more detail, as the antenna dish of the pico terminal is of this type. The general functional principle can be applied to other types. To describe antenna properties, the terms "isotropic antenna" or "isotropic radiator" are commonly used. This hypothetical model can be interpreted as a spherical light bulb that emits its power uniformly in all directions.

Definition

An antenna is defined in the IEEE Standard Definitions of Terms for Antennas as a means for radiating or receiving radio waves [3]. It is an interface between a free-space electromagnetic wave and a guided wave. There are many types of antennas in different variations, but their operation mode is essentially the same. An electromagnetic wave is emitted when a radiofrequency transmitter excites electrical currents in the conductive surface layers of the antenna. The same principle works reverse, when an incident radio wave excites currents in the antenna, which are conducted to the receiver. The ability of an antenna to work both ways is termed the principle of reciprocity. [4]
In contrast to transmission via wire, no guiding structures are required by antennas. The waves are radiated away from a transmitting antenna in all directions. To overcome long distances during radio communication high gain antennas are used. [5]

Antenna pattern and directivity

Each antenna has its specific antenna pattern, which is a ”graphical representation of the radiation properties of the antenna as a function of space coordinates” [3] and usual determined in the far field. A helpful way to visualize the radiation pattern concept is described in [5]. The hypothetical isotropic pattern, as illustrated in Figure 2.1, is represented by a spherical ball of bread dough. The spherical shape distorts by squeezing the dough. As a result, there is more dough in some direction then in others. The same principle can be transferred to antennas. The entire radiation power of an isotropic antenna is emitted uniformly in all directions. A fictitious reflector on the left side of this isotropic radiator would produce an antenna pattern as shown in Figure 2.1. The power density S at a fixed distance r measured from the antenna is used to quantify the radiation. $F(\theta, \phi)$ represents the angular variation of radiation around the antenna, where θ and ϕ are the angles to describe the direction, as often seen in spherical coordinate systems. The main beam or main lobe has a much higher power density due to the reflected waves then the side lobes. With this principle, antennas can be directed to focus the available power into a desired direction. [5]

The ratio of the power density in the direction of the pattern peak S to the average power density S_i at the same distance from the antenna is called directivity D and shown in Equation 2.1:

$$D = \frac{S}{S_i} \tag{2.1}$$

Directivity measures the degree of concentration of the emitted radiation in a certain direction. A parabolic antenna, such as the antenna of the pico terminal, has a high directivity.

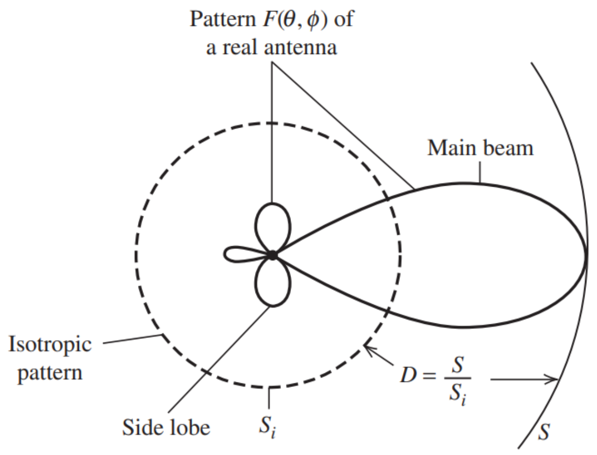


Figure 2.1: Comparison of antenna patterns. A theoretical isotropic radiator visualized as a dashed circle with a uniform power density in all directions and a typical antenna pattern of a directed antenna with a main beam in a certain direction and side lobes are shown. (Source: [5])

Antenna gain

Another term used to describe directivity, which considers losses on the antenna, is gain. Similar to directivity, gain is "expressed relative to an isotropic radiator". In Equation 2.2, the aperture efficiency η, the diameter D of the aperture and the wavelength λ are used to calculate the gain G of an aperture antenna. [5]

$$G = \eta \left(\frac{D\pi}{\lambda} \right)^2 \tag{2.2}$$

Polarization

In addition to frequency, magnitude and phase, polarization is one of the determinants of a monochromatic electromagnetic wave, which varies sinusoidally with time. Polarization is the "time-varying direction and relative magnitude of the electric-field vector" [3]. Linear polarization in horizontal or vertical direction and circular polarization are used. In Figure 2.2 an electromagnetic wave with its electrical ($\overrightarrow{E_x}$) and magnetic ($\overrightarrow{H_y}$) field behavior is visualized. In this case the electrical field component changes only in vertical direction over a time instance z. Thus, the wave is polarized vertically. [5]

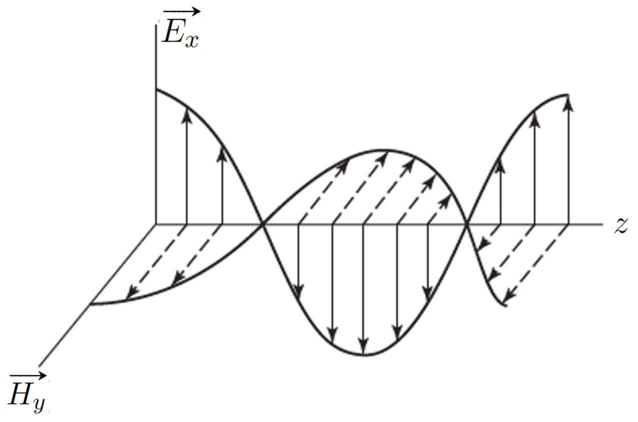


Figure 2.2: Time-space behavior of a linear polarized electromagnetic wave with the electrical field vector $\overrightarrow{E_x}$ in vertical and the magnetic field vector $\overrightarrow{H_y}$ in horizontal direction. (Adapted from [5])

An example for a circularly polarized propagation is shown in Figure 2.3. This type of polarization is used when the orientation of two corresponding antennas is not known.

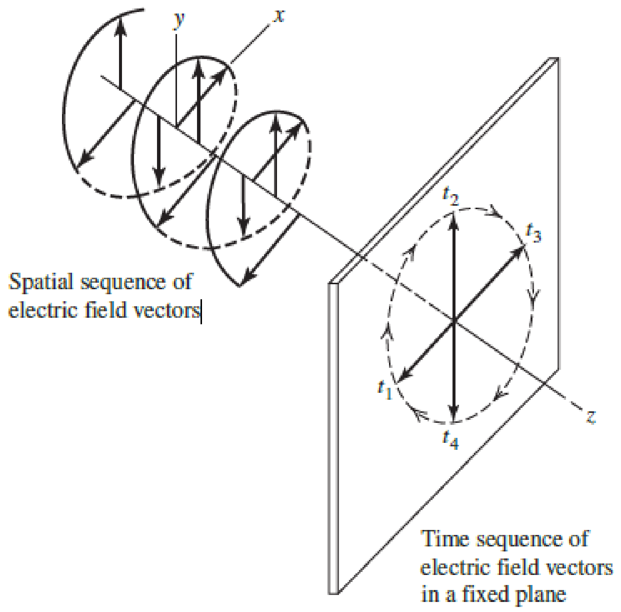

y
x
z
t_1
t_2
t_3
t_4
Spatial sequence of electric field vectors
Time sequence of electric field vectors in a fixed plane

Figure 2.3: Perspective of a circularly polarized electromagnetic wave and the time sequence of electric field vectors as the wave passes through a fixed plane. (Source: [5])

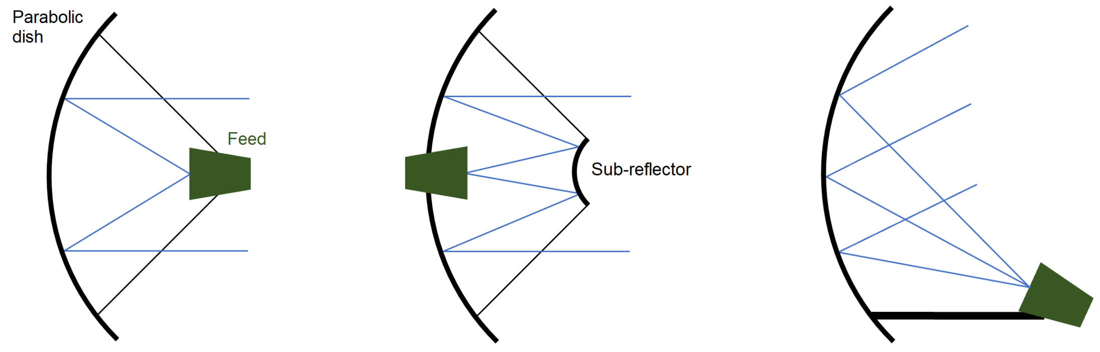
Reflector antennas

A widely used high-gain antenna is a reflector system which achieve gains of 30 dB or higher in the microwave region. The antenna of the pico terminal is of the same type. The simplest reflector antenna consists of two components:

- a reflecting surface that is large relative to a wavelength
- a much smaller feed antenna

The parabolic reflector antenna is the most popular form with a paraboloid of revolution as reflector, also called ”dish”. [5]

A horn antenna is commonly used as a feed to a parabolic dish antenna. Different feed methods are possible and visualized in Figure 2.4. The simplest method is the *front feed*. According to its position in front of the antenna, the feed and its supporting structure produce aperture blockage. The *Cassegrain* method and the not shown *Gregorian* method are used in dual-reflector antennas. These systems consist of a main reflector and a secondary subreflector with different shapes, depending on the used method. To avoid aperture blockage, the *offset feed* method is used. [6]
A conventional offset feed parabolic antenna as shown in Figure 2.4c is used when receiving satellite television. The pico terminal also uses an offset configuration but with a special behavior explained in Section 2.2.1.



(a) Front feed   **(b)** Cassegrain   **(c)** Offset feed

Figure 2.4: Different feed methods used to a parabolic dish antenna: front feed, Cassegrain and offset feed method in combination with a parabolic dish are compared. The blue lines represent a simplification of the elctromagnetic waves reflected by the parabolic antenna dish. (Adapted from [6])

Half-power beamwidth

A parameter which is associated with the antenna pattern is the term *beamwidth*. It is defined as the "angular separation between two identical points on opposite side of the pattern maximum". A widely used beamwidth to describe the angle between two directions with one-half value of the beam is the half-power beamwidth (HPBW). In this thesis, the HPBW is evaluated to validate the pointing behavior of the pico terminal. [7]

In Equation 2.3, a rough approximation to evaluate the $HPBW$ is provided [8], using the wavelength λ and the dish diameter D. With longer wavelengths, the half-power beamwidth increases.

$$HPBW \approx 70 \cdot \frac{\lambda}{D} \tag{2.3}$$

2.1.2 Satellites and orbits

In this work, the term "satellite" describes an object that has been sent into space and orbits the Earth. With on-board processing units a satellite is able to amplify or reformat received uplink data, to regenerate data directly on-board or to act as a routing device, to send data to a specific location. [4]
Satellites are used in communications, remote sensing, global navigation and meteorology [9]. Depending on their orbit, satellites can be divided into different types:

- Geostationary (GEO) satellite
- High elliptical orbiting (HEO) satellite
- Middle-earth orbiting (MEO) satellite
- Low-earth orbiting (LEO) satellite

Often the terms "geosynchronous" and "geostationary" are used interchangeably, but they are not the same. When a satellite orbits the Earth with the same angular velocity, an orbit is geosynchronous. One orbital period takes as long as a sidereal day. A sidereal day, an astronomical time scale, is the time interval during which the Earth completes one rotation on its axis and some chosen star appears to transit twice consecutively on the observer's local meridian. A **GEO** is a geosynchronous orbit with both, zero inclination angle and zero eccentricity, and is also known as Clarke Belt. Sir Arthur C. Clarke (1917-2008) mentioned the principle of a stationary "artificial satellite" in a 24-hour orbit in 1945. Satellites in geostationary orbit remain "essentially motionless above a point on the Equator" and are classified by their sub-satellite point's longitude. The sub-satellite point is the intersection of the Earth's surface and a fictitious line connecting the center of the Earth and the satellite. [10],[11],[12]

In a **HEO**, the distance from a satellite to the Earth's surface changes dramatically compared with other orbits. This behavior is visible in Figure 2.5. The focus point represents the Earth, the satellite is surrounding the focus point along the orbit. In perigee the satellite is closest to the Earth, while in apogee the distance reaches its maximum value. A HEO is designed for better coverage of regions at higher northern or southern latitudes by arranging apogee, so that a particular area is continuously covered by the satellite. [4]

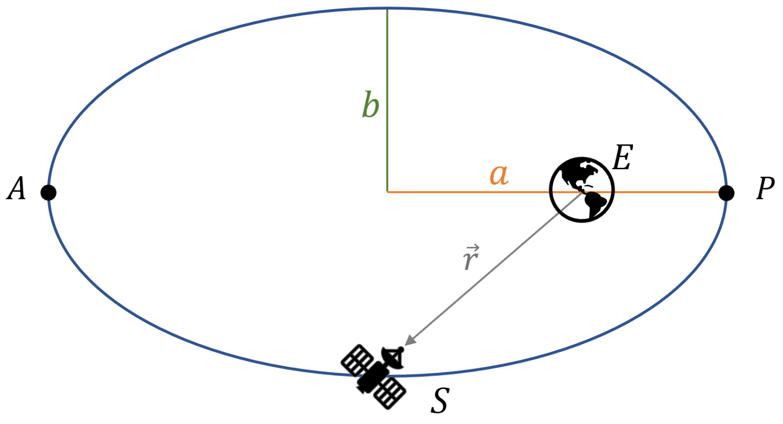


Figure 2.5: High elliptical orbit with the Earth (E) in one focus point. In such an orbit, the distance $\vec{r}$ between center of the Earth and the satellite (S) changes significantly during one orbital period. The nearest distance between satellite and Earth is in perigee (P), the farthest distance is in apogee (A). The semi-major axis (a) and the semi-minor axis (b) define the shape of the orbit.

The region between LEO and a geosynchronous orbit is called **MEO**, with satellite distances between 8,000 and 18,000 km from the Earths surface. Not only for navigation, such as the Global Positioning System (GPS), but also for communications satellites covering the North and South Pole this orbit is used. [13]

The orbit with the lowest altitudes is the **LEO**. A typical period for satellites, like the International Space Station (ISS), is approximately 90 minutes to orbit the Earth. In this orbit, observers on ground stations notice quick changes of the satellites relative positions. In addition, due to its low altitude, a LEO satellite is only visible for a short time interval at the ground station. For this reason, satellites often work in satellite constellations for missions that require uninterrupted connectivity. [13]

2.1.3 Two-Line-Elements

A definition which is often used in this thesis in combination with satellite tracking are Two Line Elements (TLE). It is a data format with a defined structure containing orbital elements. To define an orbit in the plane the parameters *eccentricity* e and *angular momentum* h are necessary. The *true anomaly* θ is the third parameter, used to locate a point on the orbit. To describe the orientation of an orbit in three dimensions, additional parameters (called Euler angles) are necessary. The first Euler angle Ω, the *right ascension of the ascending node* is a positive number between 0° and 360°. The angle between the orbital plane and the equatorial plane is called *inclination* i and measured according to the right-hand rule with positive numbers between 0° and 180°. To locate the perigee of the orbit, which is the nearest distance to Earth, the third Euler angle called *argument of perigee* ω is used. The relation between the orbital elements is illustrated in Figure 2.6. [14]

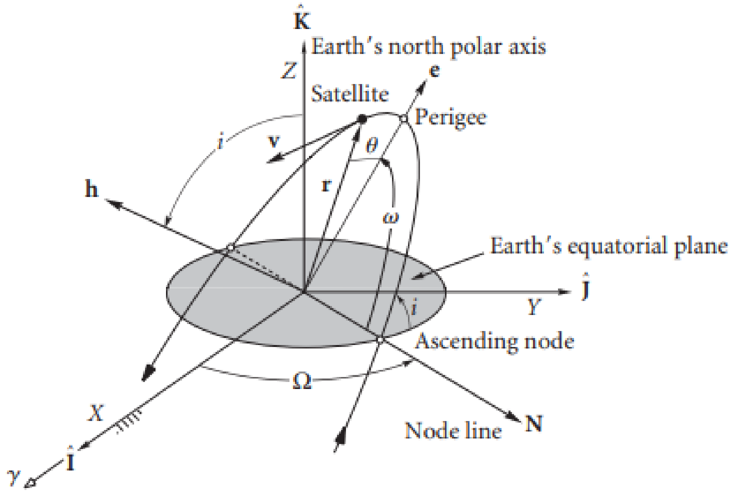


Figure 2.6: Orbital elements: the Earth's equatorial plane and a satellite orbiting the Earth at a distance r with a velocity v are shown. With orbital elements, the exact position of the satellite can be determined. (Source: [14])

TLE is a data format that describes the satellite's orbit around Earth and was first used by *North American Aerospace Defense Command* (NORAD) and *National Aeronautics and Space Administration* (NASA). Apart from the name, this format contains two lines of encoded text. [15]

In Listing 2.1 the two line element set format of the ISS is shown. This data is available online [16].

Listing 2.1: Two-Line Orbital Element Set Format of the International Space Station. Line 0 is a 24 character name followed by the two lines containing the information. The first information line starts with the character '1', the second line with the character '2'.

```
ISS (ZARYA)
1 25544U 98067A   21076.22838122  .00000001  00000-0  82009-5 0
    ↪ 9999
2 25544  51.6441  77.5551 0003453 118.6672 330.1679
    ↪ 15.48910092274347
```

In Table 2.1 the columns of the first line of the TLE format are described. In Table 2.2 the second line is described. More detailed information can be found in [17].

Table 2.1: First line of the TLE format: in total 69 characters including spaces between each field are available. The nine fields separated by spaces are numbered in the first column of this table. The second column represents the position interval of the data. In the last column a description of the data is shown. (Source: [17])

Field	Position	Description
1.1	01	Line Number of Element Data
1.2	03-07	Satellite Number
	08	Classification
1.3	10-11	International Designator (Last two digits of launch year)
	12-14	International Designator (Launch number of the year)
	15-17	International Designator (Piece of the launch)
1.4	19-20	Epoch Year (Last two digits of year)
	21-32	Epoch (Day of the year and fractional portion of the day)
1.5	34-43	First Time Derivative of the Mean Motion
1.6	45-52	Second Time Derivative of Mean Motion (decimal point assumed)
1.7	54-61	BSTAR drag term (decimal point assumed)
1.8	63	Ephemeris type
1.9	65-68	Element number
	69	Checksum (Modulo 10)

Table 2.2: Second line of the TLE format. The structure is similar to Table 2.1. (Source: [17])

Field	Position	Description
2.1	01	Line Number of Element Data
2.2	03-07	Satellite Number
2.3	09-16	Inclination [Degrees]
2.4	18-25	Right Ascension of the Ascending Node [Degrees]
2.5	27-33	Eccentricity (decimal point assumed)
2.6	35-42	Argument of Perigee [Degrees]
2.7	44-51	Mean Anomaly [Degrees]
2.8	53-63	Mean Motion [Revs per day]
	64-68	Revolution number at epoch [Revs]
	69	Checksum (Modulo 10)

2.1.4 Look angles

To communicate with a satellite, the earth station antenna such as the antenna dish of the pico terminal must be aligned into a specific direction. *Azimuth* and *elevation*, collectively called *look angles*, are the coordinates of this specific direction. The azimuth angle describes the horizontal direction and starts towards north at 0°, rising clockwise. The elevation angle θ is used for vertical alignment and starts at 0° towards the horizon rising to 90° at zenith.
Figures 2.7 and 2.8 explain the azimuth and elevation angles. In both cases, a satellite S is pointed by a ground station antenna G. When perfectly aligned, the antenna main beam of the ground station's antenna matches the green line. Due to the distance of the satellite to the earth's surface, only parts of the earth and the satellite are illustrated.

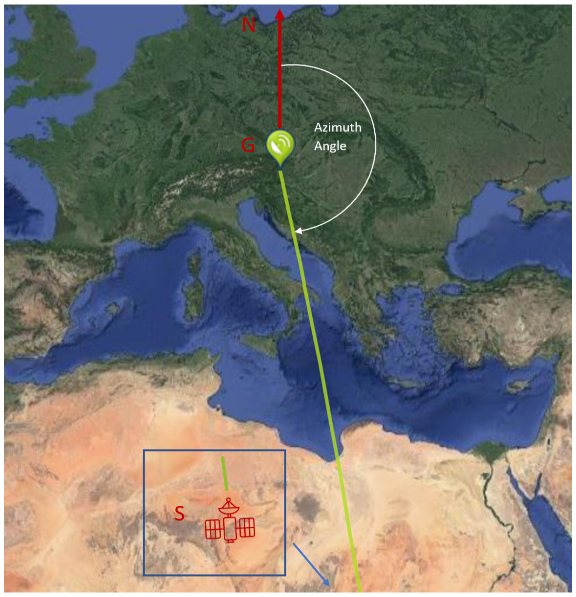


Figure 2.7: Azimuth angle of an observer at the ground station (G) in the direction of the satellite (S), measured from north (N). The green line between the ground station and the satellite illustrates the pointing direction. (Adapted from *Dishpointer*)

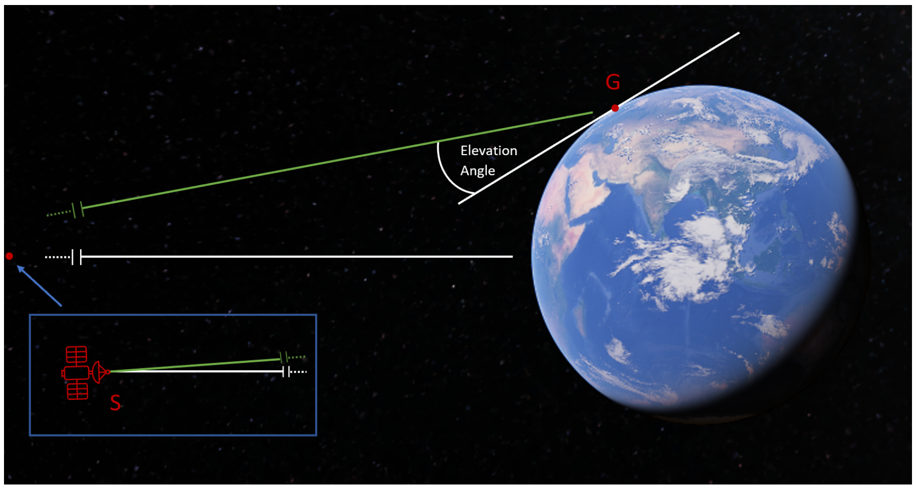


Figure 2.8: Elevation angle of an observer at the ground station (G) in the direction of the satellite (S) measured from the horizon. The green line between the ground station and the satellite illustrate the pointing direction. (Adapted from *GoogleEarth*)

To calculate both angles for a defined satellite at a certain ground station position, the approach explained in [4] can be used. With the help of the spherical trigonometric relations and the cosine law, the look angles to point a geostationary satellite can be determined using Equations 2.4 and 2.5. In this case, the Earth radius R_e, the orbit altitude h_0, the radius of the orbit measured from the center of the Earth r, the distance R_s between the satellite and earth station also known as *slant range*, the latitude of the earth station L_{ET} and the difference in longitude between the earth station and the satellite Δ are used. Positive values for latitudes in the northern hemisphere are used. [4]

$$\alpha_z = 180 + tan^{-1}\left(\frac{tan(\Delta)}{sin(L_{ET})}\right) \tag{2.4}$$

$$\theta = tan^{-1}\left(\frac{cos(\Delta)cos(L_{ET}) - \frac{R_e}{r}}{\sqrt{1 - cos^2(\Delta)cos^2(L_{ET})}}\right) \tag{2.5}$$

To transform the geodetic coordinates longitude, latitude and height to the local azimuth-elevation-range (AER) spherical coordinates, the *geodetic2aer* function in MATLAB can be used. The *ephem* library by [18] provides similar functionality in Python. For a specific position of the ground station antenna and a satellite to be pointed to, the values for azimuth and elevation can be calculated using the functions provided in the library.

2.1.5 NMEA protocol

GPS receiver communication, which is also used in this work, is defined within National Marine Electronics Association (NMEA) specifications. The information computed by the GPS receiver are provided in standardized NMEA sentences, each starting with the character '$' and followed by a specific identifier. Each line ends with a carriage return or line feed sequence. The data itself is ASCII text. [19]

Listing 2.2 demonstrates NMEA sentences from a GPS receiver immediately after start up. Missing entries between the commas indicate that no valid data was received at the corresponding time interval.

Listing 2.2: NMEA sentences from the GPS receiver. Depending on the identifier at the beginning of a sentence, different data are available, separated by a comma. Right after start up, only a few values are available. The red arrow indicates a line break and has been added for better readability.

```
$GPRMC,120727.000,V,0000.0000,N,00000.0000,E,0.00,0.00,170321,,,N
    ↪ *71
$GPVTG,0.00,T,,M,0.00,N,0.0,K,N*02
$GPGGA,120727.000,0000.0000,N,00000.0000,E,0,00,99.9,-17.0,M,17.0,M
    ↪ ,,0000*78
$GPGSA,A,1,,,,,,,,,,,,,99.9,99.9,99.9*09
$GPGSV,1,1,01,06,,,21*7D
$GPGLL,0000.0000,N,00000.0000,E,120727.000,V,N*40
$GPGRS,120727.000,1,,,,,,,,,,,,*7F
$GPGST,120727.000,0.0000,,,,3750000,3750000,3750000*67
$GPZDA,120727.000,17,03,2021,,*53
$GPGBS,120727.000,,,,,,,*5E
```

2.1.6 Trueness, precision and accuracy

Several international scientific committees defined many parameters to describe the performances of a measurement. Often used terms, which are also mentioned in this thesis, are *trueness*, *precision* and *accuracy*. Relevant definitions are examined by [20], such as *Vocabulary of Metrology 2012* (VIM), *ISO 5725-1:1994* and *DIN 55350-13:1987-07*. The following can be concluded from these definitions:

- Trueness is a parameter which describes the systematic error of measurement.
- Precision is a parameter which describes the random error of measurement.
- Accuracy is a parameter which describes both, systematic and random error of measurement.

Trueness applies to the "average value of a large number of measurement results" and is expressed in Equation 2.7. It can be assessed by using the difference between the average measured value $\bar{x}$ calculated with Equation 2.6 and a reference value μ. The result of an individual measurement x_i and the number of measurements n made are necessary for these calculations. [20]

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.6}$$

$$trueness = |\bar{x} - \mu| \tag{2.7}$$

To indicate how close independent measurement results obtained by replicated measurements are to one another is described by precision. It can be verified by using the standard deviation in Equation 2.8, to describe the spread of the results. [20]

$$precision = s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.8}$$

Accuracy applies to a single result of a measurement and can be evaluated using Equation 2.9. It describes how close this single result is to the expected reference value.

$$accuracy = \frac{|x_i - \mu|}{x_i} \tag{2.9}$$

Figure 2.9 visualizes the difference between trueness and precision [21].

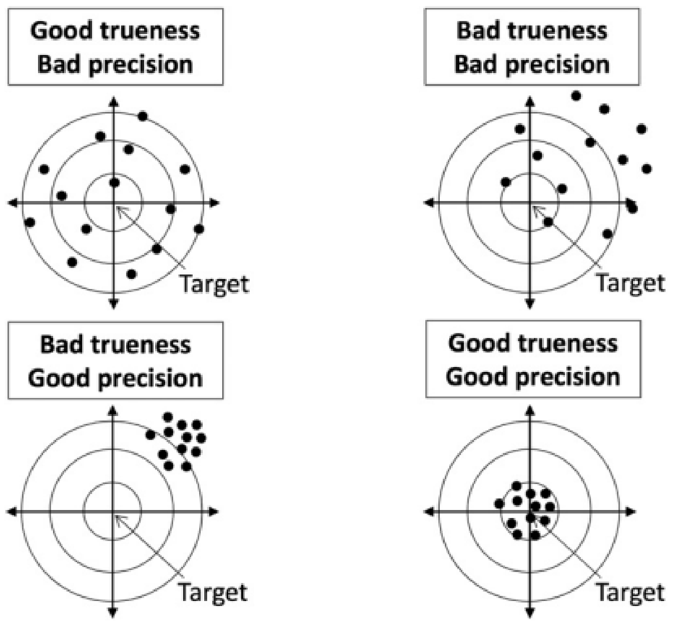


Figure 2.9: Visual comparison of trueness and precision for four assumed scenarios. The center of each inner circle is the target and the dots are notional results of independent measurements. In case of bad trueness and bad precision, the spread of the measured values is large and the geometric center of the dots has an offset from the target. If the spread between the dots is low and the geometric center of the dots lies near the target, the trueness and the precision are good. (Source: [21])

2.1.7 IMU

An Inertial Measurement Unit (IMU) is used to get information about the attitude of an object. In this case this sensor measures the antenna dish orientation. It combines different sensors such as accelerometer, gyroscope and magnetometer. [22]

One type of IMUs are 9-axis-sensors. There are different 9-axis-sensors available on the market, combining the mentioned sensors in one single package to evaluate *roll* ϕ, *pitch* θ and *yaw* ψ. The last term is also called *heading*. In Figure 2.10 these terms are explained with the corresponding movements of an aircraft.

Procedures for calculating the attitude from the raw sensor data are described in [23] and [24]. A calculation may or may not be required depending on the sensor used. More details to this topic are explained in Sections 3.3.3 and 3.4.3.

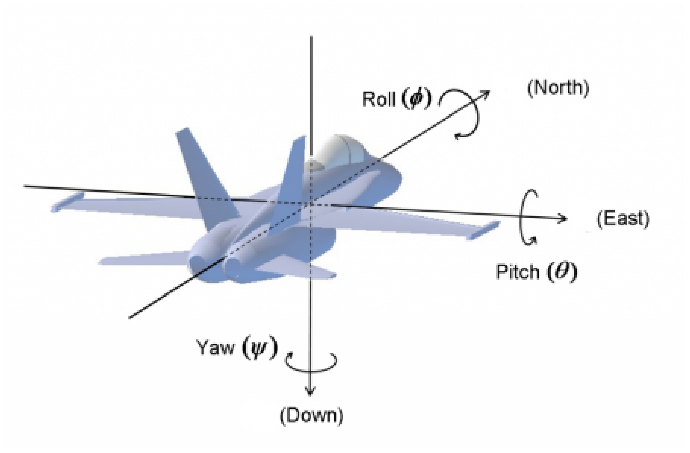


Figure 2.10: Roll, pitch and yaw explained with an aircraft as an example. The terms mentioned are typically used in the context of aviation to determine the orientation of objects in flight. Increasing the pitch raises the nose of the aircraft and lowers the tail. The roll axis is directed forward. Positive roll lifts the left wing and lowers the right wing. By changing yaw in positive direction, the nose of the aircraft moves to the right. (Source: [25])

2.1.8 Satellite beacon signal

The satellite beacon is a reference signal with a fixed frequency and power. It is sent by the satellite and usually created by a fixed crystal source. Most energy of such a beacon signal exist in a narrow bandwidth. Beacon signals are used for different applications such as precise orientation of the earth station to the satellite or propagation research.
The beacon signal of Alphasat considered in the laboratory exercise has a frequency of 19.701 GHz. Alphasat is a communications satellite launched by the European Space Agency (ESA) in 2012. The coverage is Europe-wide and shown in Figure 2.11. [26],[27]

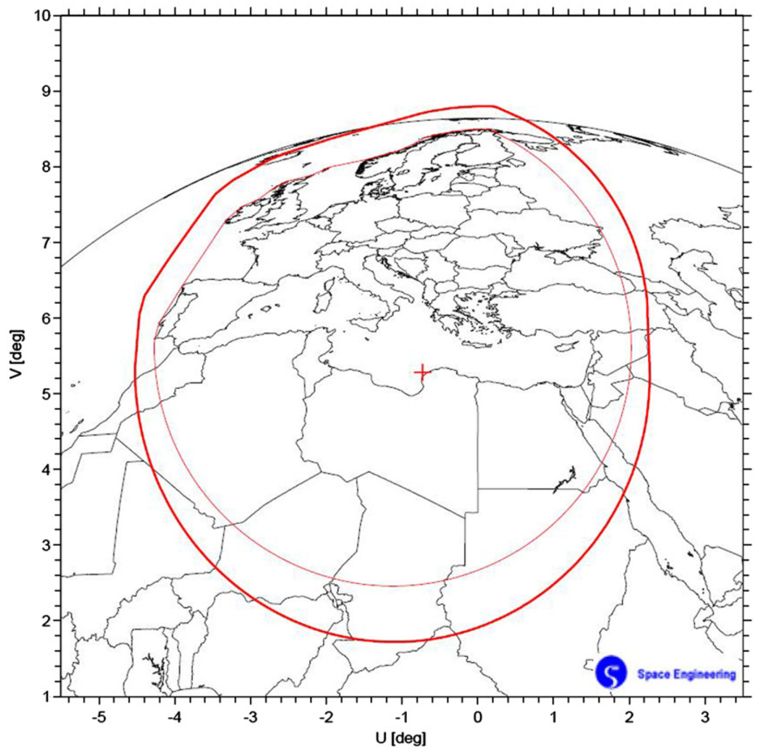


Figure 2.11: Coverage of Alphasat's beacon signal for the Ka-band. Within the inner contour line, a power multiplied by the antenna gain of at least 19.5 dBW is available. (Source: [27])

Spectrum analyzer

Electrical signals can be represented in time domain and in frequency domain, as illustrated in Figure 2.12 and they are related to each other by the Fourier transform, which is not discussed here in more detail. The frequency spectrum of a signal is the frequency range that is contained in a signal. [28],[29]

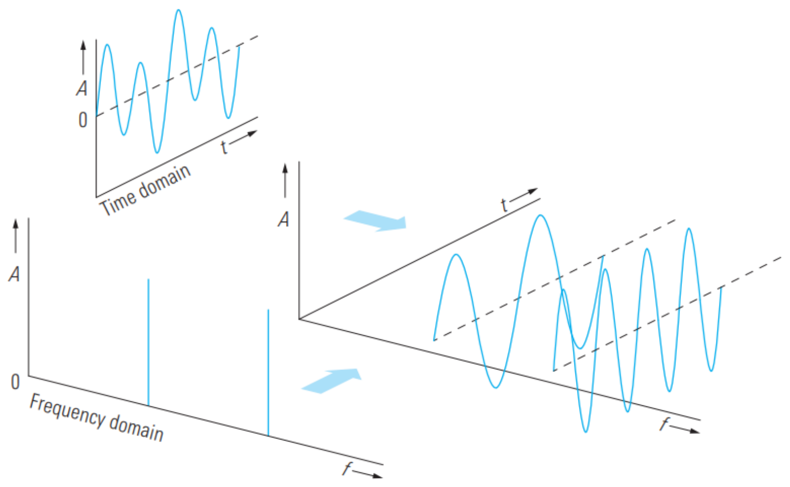


Figure 2.12: Signal representation in time and frequency domain. The added signal in time domain is achieved by adding the amplitudes at certain timestamps. In the frequency domain, the signal power is shown for the frequencies occurring in the signal. (Source: [28])

To measure the frequency spectrum of a signal, e.g. of a satellite's beacon signal, a spectrum analyzer can be used. A block diagram of such a device is explained in Figure 2.13.

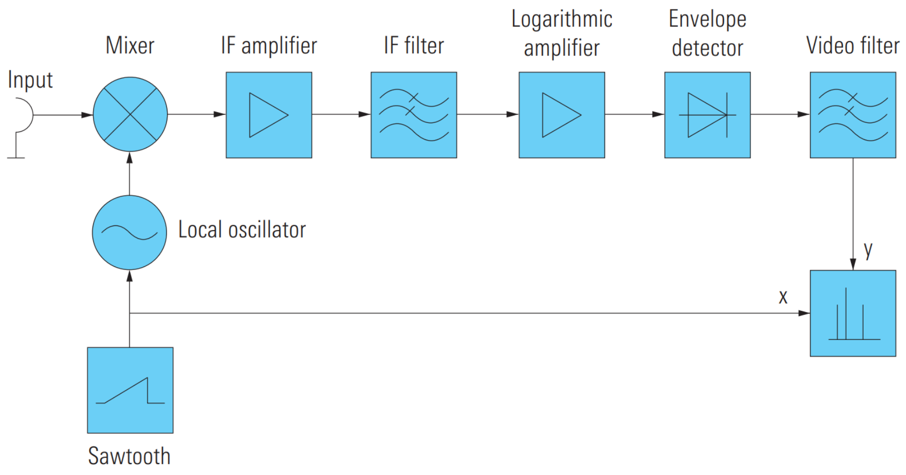


Figure 2.13: Block diagram of a spectrum analyzer: the mixer converts the RF input signal with the aid of a local oscillator to a lower or higher intermediate frequency. The local oscillator is controlled via a sweep generator with a sawtooth signal. The converted IF signal is amplified and applied to the IF filter. This filter defines the resolution bandwidth. With a logarithmic amplifier and an envelope detector the signal is compressed to allow a wide level range to be displayed simultaneously. The video bandwidth can be adjusted with the video filter. This lowpass filter can be used to smooth the signal. (Source: [28])

The following elementary setting parameters are provided:

- Frequency display range (center frequency and span)
- Level display range
- Frequency resolution

- Sweep time

The *displayed frequency range* can be set in two different ways. One possibility is by setting the minimum and maximum frequency, in which the observed signal occurs. The commonly used second option is by setting the center frequency and a desired span around the center frequency. The *level display range* is set to the maximum power level, the span and the defined attenuation of an input RF attenuator. With the resolution bandwidth (RBW) of the IF filter the *frequency resolution* can be changed. The integrated power inside this frequency range is visualized on the screen. For smoother signals, the video bandwidth (VBW) can be changed. The required time to record the whole spectrum, limited by the display frequency range, is called *sweep time*. [28]
Changing the RBW has several effects. A smaller RBW decreases the noise floor due to less power inside the swept frequency range on the one hand and increases the resolution to detect frequency components of a signal which have a small separation on the other hand. At the same time, the sweep time increases. In Figure 2.14 the parameters of a spectrum analyzer are shown.

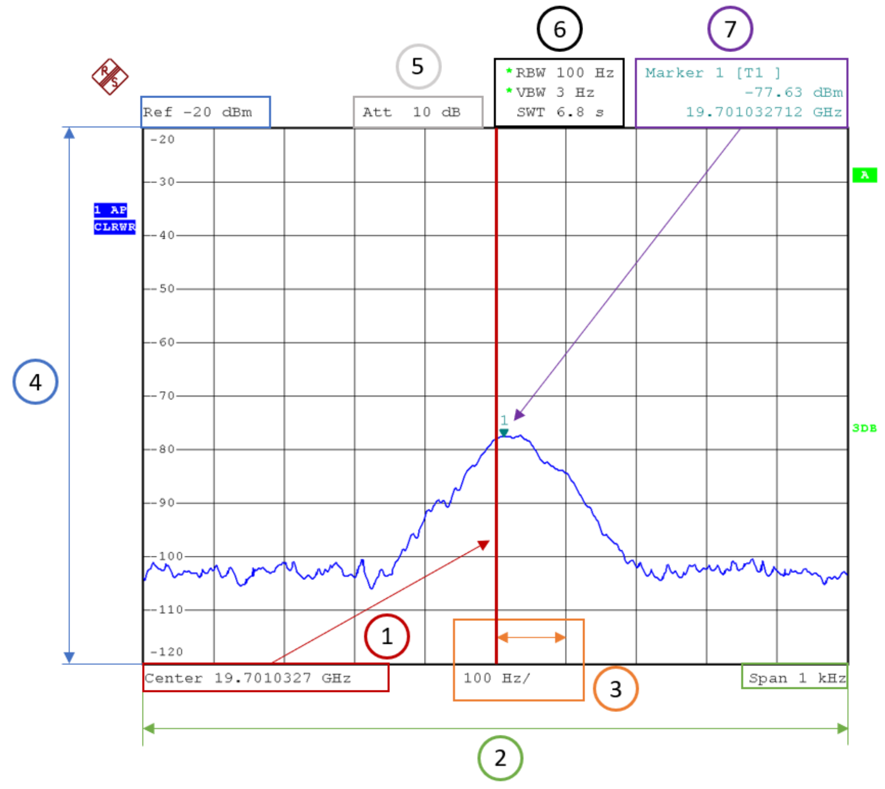


Figure 2.14: Setting parameters of a spectrum analyzer: the center frequency (1) and the span (2) limit the screen in horizontal direction. The resolution per division is visualized in (3). The logarithmic power level range (4) is set with a reference value, in this case -20 dBm, an attenuation (5) to limit signals with a very wide level range and the span. The configured resolution bandwidth, video bandwidth and sweep time are shown in (6). An optional marker (7), in this case set to peak, visualize the values for power level and frequency.

2.1.9 Stepper motor

The essential property of a stepper motor is its ability to convert electrical impulses into precisely defined increments of the rotor position. A stepper motor consists of a stationary part called the ”stator” and a rotating part called the ”rotor”. There are teeth of magnetically permeable material on both, stator and rotor which experience equal and opposite forces. These forces try to pull the teeth together and minimize the airgap between them. A schematic of the cross-section of a small part of a stepper motor is illustrated in Figure 2.15. The major force component in normal direction try to close the airgap whereas the tangential forces attempt to move the teeth sideways with respect to each other. When removing or redirecting the flux flowing between teeth, the forces of attraction drop to zero. [30]

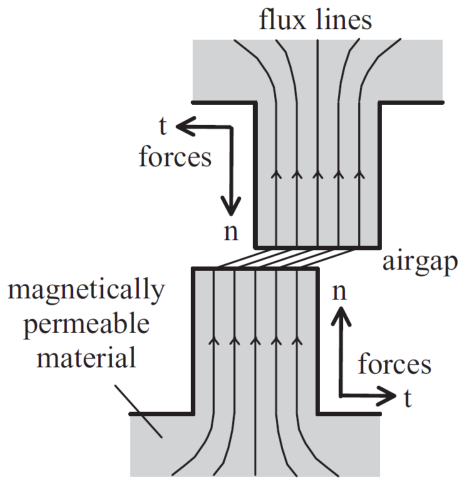


Figure 2.15: Cross-section of a small part of a stepper motor: force components in normal (n) and tangential (t) direction between two magnetically permeable teeth are shown. Magnetic flux crosses the small air gap between the teeth. The source of flux can be a permanent magnet or a current-carrying winding or a combination of both, depending on the motor type. (Source: [30])

Types of stepper motors

There are three basic types of stepper motors:

- Permanent magnet
- Variable reluctance
- Hybrid

Permanent magnet stepper motors have a magnetized rotor, while variable reluctance stepper motors have toothed soft-iron rotors. A hybrid stepper motor combines both technologies, permanent magnet and variable reluctance. The different types are distinguished by the arrangement of the windings. Multiple windings are held by the stator. [31]

The type considered in this thesis is the hybrid bipolar stepper motor. Bipolar stepper motors consist of two windings and have four wires. The current flow in the winding is bidirectional and requires polarity changes at each end. [31]

A simplified model in Figure 2.16 explains the working principle of a stepper motor.

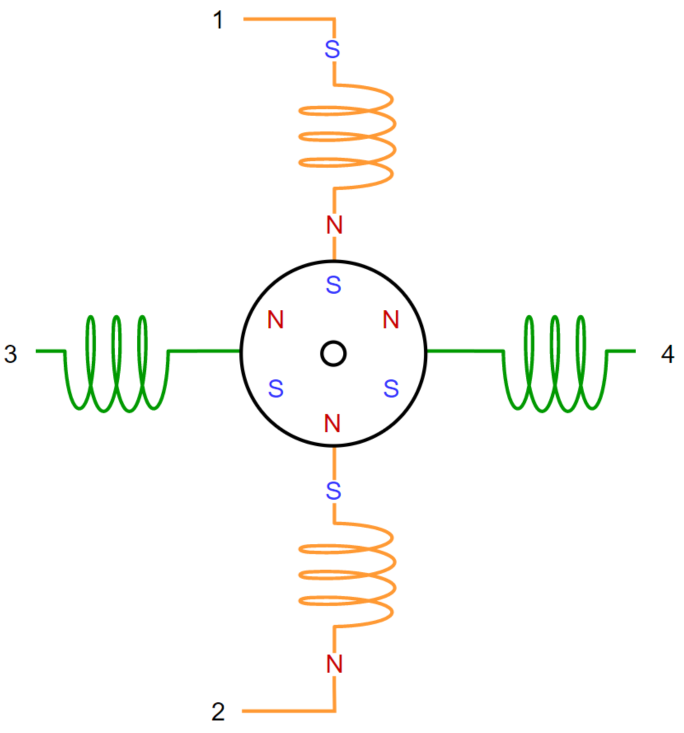


Figure 2.16: Working principle of a stepper motor: two different colored coil pairs and a rotor with six poles are illustrated. Each coil pair has two wires for energizing or de-energizing the corresponding winding. In this special case, current flows between terminals 1 and 2. The current-carrying coils cause the magnetic field shown. Switching off terminals 1 and 2 and energizing terminals 3 and 4 changes the magnetic field of the stator. The poles of the rotor are influenced by the stator magnetic field. As a result, the rotor moves 30°. If both windings are energized with different currents at the same time, the step resolution increases for a less jerky rotation.

Standardized sizes

The National Electrical Manufacturers Association (NEMA) has standardized various sizes of stepper motors. For each size different lengths of the motors are available, depending on the necessary specifications. In this thesis, stepper motors with the following NEMA standards are mentioned:

- NEMA 17 - cross section of the housing of approximately 42 x 42 mm and variable length
- NEMA 23 - cross section of the housing of approximately 57 x 57 mm and variable length

Micro stepping

The visualized motor in Figure 2.16 achieves a step resolution of 30° in its rated step size. Energizing both coil pairs at the same time allows the motor to move in steps that are half its rated step size. To increase the stepper resolution and to achieve smoother transitions between steps, ”micro stepping” can be used. This method is based on the principle of gradually transferring current from one winding to another and is achieved by pulse-width modulation. In this case, the duty cycle of the signal charging one winding is increased while the duty cycle of the other winding is decreased. [31]

Torque

A critical consideration when choosing a stepper motor is torque. Different types of rated torque, as mentioned in [31], are:

- Holding torque to rotate the shaft of the motor while the windings are energized
- Pull-in torque against a motor can accelerate from a standing start without missing any steps
- Pull-out torque as the load a motor can move at operating speed
- Detent torque to rotate the shaft of the motor while the windings are de-energized

2.1.10 Gear ratio

To evaluate the necessary torque of the required motor drives of the pico terminal, the gear ratio needs to be evaluated. According to *DIN 868*, the ratio i of a gear is the ratio of the driving speed n_1 to the driven speed n_2 and inversely proportional to the number of teeth (z_1 and z_2) of the gears [32]. In Equation 2.10 the relation between the driving speed n_x, angular velocity ω_x, diameter d_x, number of teeth z_x and rotational force (torque) M_x is shown. The subscript letter x represents either the pulley on the motor shaft of the driving axis ($x = 1$) or on the shaft of the driven axis ($x = 2$).

$$i = \frac{n_1}{n_2} = \frac{\omega_1}{\omega_2} = \frac{d_2}{d_1} = \frac{z_2}{z_1} = \frac{M_2}{M_1} \tag{2.10}$$

Due to this equation, the number of teeth is directly proportional to the torque. In this case, a smaller pulley on the motor shaft driven with a higher speed results in a higher torque on the driven axis with lower driving speed.

2.2 Status quo

After an introduction of the pico terminal in Section 2.2.1, existing antenna tracking methods are presented and the preferred method used in this work is explained in Section 2.2.2. At the end of this section, related studies concerning antenna tracking systems are discussed.

2.2.1 Initial situation

The pico terminal is a compact and transportable parabolic antenna setup with a dish manufactured from solid aluminum and an antenna gain of approximately 35 dBi at 19.7 GHz. One of the three produced pico terminals from the same manufacturer is used for training purposes in laboratory exercises at Graz University of Technology. The remaining two are disassembled and not in use. In Figure 2.17 all components of the used pico terminal are explained. One complete set of disassembled parts is the basis for this work which need to be rebuilt. Apart from hardware and software adaptations which are discussed in this work, the geometric properties and unknown antenna parameters must be determined in a first step. This is necessary because there are no detailed drawings or other documents for the devices available. Due to the geometric properties, the influence of skew angle changes on antenna alignment is illustrated in this section. Different skew angle and elevation configurations are used to explain the behavior of the effective beam path of a satellite link.

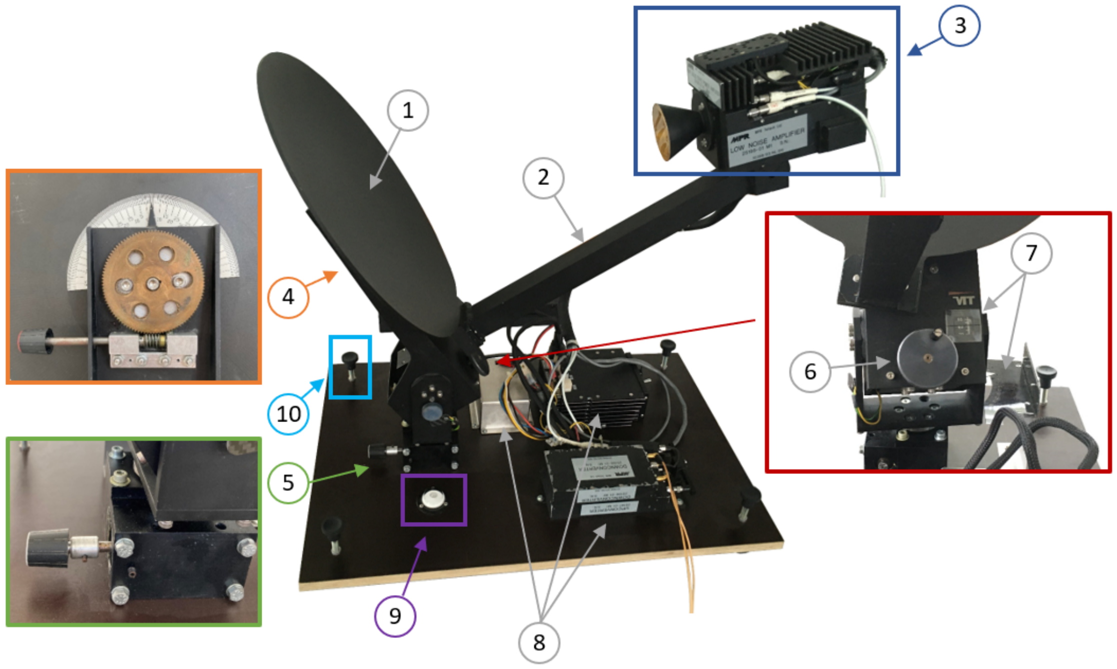


Figure 2.17: Components of the pico terminal: parabolic dish (1), the feed arm (2) and the LNA (3) are firmly connected to one another and are aligned as one unit when pointing. Worm drives are used to adjust the skew (4), azimuth (5) and elevation (6) angle of the pico terminal. Two scales (7), one for the azimuth and one for the elevation axis, are used to set the actual angles. Additional components (8) like power supply and an optional down converting stage are part of the pico terminal. A leveling bubble (9) together with four adjustable feet (10) are used to level the ground plate on the actual measurement site.

In Figures 2.18-2.20 a simplified 3D model of the pico terminal is shown. In particular, the movable unit consisting of components 1-3 as explained in Figure 2.17 is visualized in different perspectives. In the side view in Figure 2.18 the mounted LNA with an offset angle of φ can be seen. The electromagnetic waves which are reflected by the dish are represented as solid lines and have the same offset. The y-axis is the rotary axis for changing the skew angle and shows the pointing direction of the antenna dish. The z-axis is pointing upwards and the x-axis is orthogonal to the y- and z-axis in a right-handed coordinate system. Due to the offset angle φ, the effective pointing direction of the antenna might differ from the pointing direction of the y-axis. At 0° skew and an elevation angle of the rotary axis of 0°, the dish is pointing horizontally but the main antenna beam has an effective elevation angle of φ. This effect is known with conventional parabolic offset antennas, such as those used, for example, when receiving satellite television.

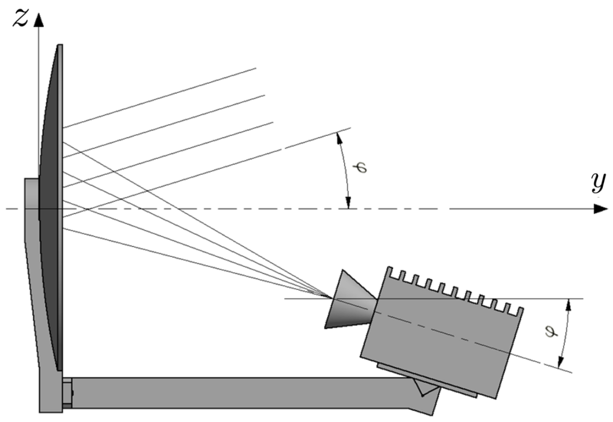
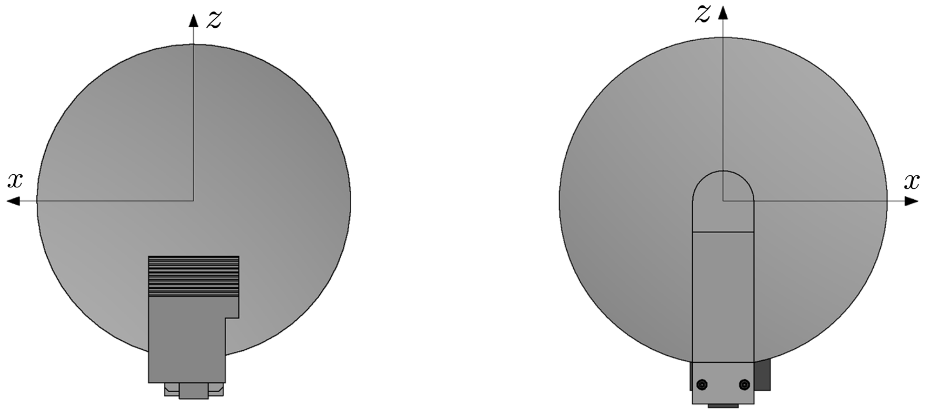


Figure 2.18: Side view of the pico terminal: the y-axis is the rotary axis of the parabolic antenna dish, the z-axis is pointing upwards. Angle φ is the offset of the parabolic antenna dish. Electromagnetic waves, represented as black lines, are reflected on the surface of the antenna dish.



(a) Front view **(b)** Rear view

Figure 2.19: Front view and rear view of the pico terminal: the z-axis is pointing upwards, the x-axis is perpendicular to the y- and z-axis.

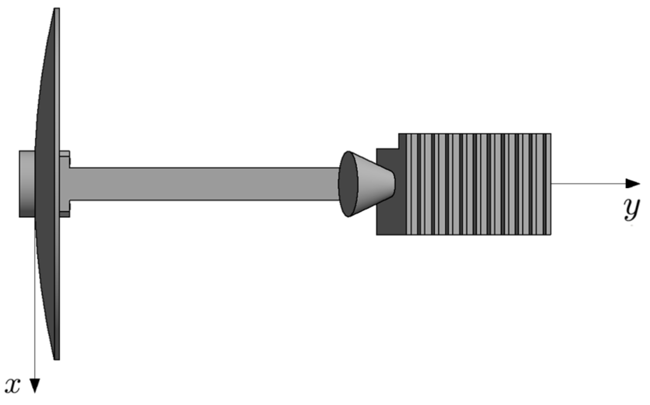
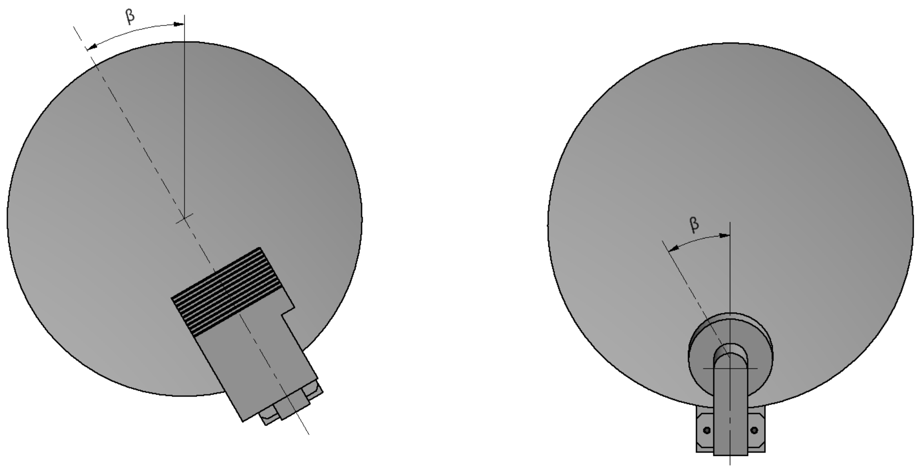


Figure 2.20: Top view of the pico terminal: at 0° skew and 0° elevation, the y-axis is pointing in the effective azimuth direction.

Influence of the skew angle on the antenna alignment

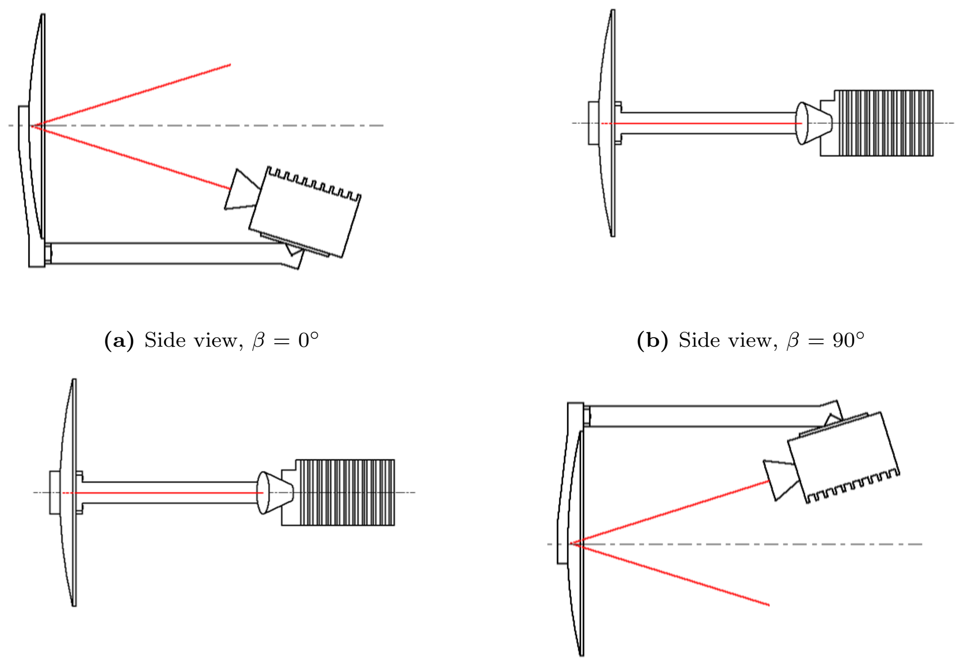
The antenna design of the pico terminal differs from a conventional offset parabolic antenna. This difference becomes noticeable when adjusting the rotary axis to change the skew angle. At 0° skew, the pico terminal behaves like a conventional offset parabolic antenna. By increasing or decreasing the skew angle β not only the LNA is rotating but also the antenna dish and the feed arm. This behavior is visualized in Figure 2.21a. Depending on the skew angle, the position of the feed arm with the attached LNA is changing. In Figure 2.21b the LNA of the conventional parabolic antenna is also rotating, but the dish and the feed arm are in a fixed nonmoving position.



(a) Pico terminal **(b)** Conventional parabolic antenna

Figure 2.21: Comparison of the pico terminal with a conventional parabolic offset antenna. In (a) the front view of the pico terminal and in (b) a conventional parabolic offset antenna with a smaller LNA are shown. In both scenarios with a positive skew angle β. When changing the skew angle of the pico terminal, the hole antenna dish and the feed arm is rotating. This leads to an offset in azimuth and elevation direction depending on the skew angle when pointing a satellite.

The difference becomes even clearer when comparing the LNA position at 0° skew and 90° skew, as shown in Figure 2.22. In the side view at 0° skew, an offset between the rotary axis and the effective elevation of the main antenna beam is noticeable. For simplicity reasons one red line represents the main antenna beam. In the top view at 0° skew, the direction in azimuth direction is the same as the direction of the rotary axis. At 90° skew the situation changes. In this case no elevation offset is noticeable, but the azimuth direction of the main antenna beam changes. By changing the skew angle starting at 0° in the positive direction the effective elevation angle decreases, whereas the effective azimuth angle increases. By changing the skew angle in the opposite direction when starting at 0°, the effective elevation angle decreases again, but in this scenario also the effective azimuth angle decreases. In both scenarios only the skew axis is modified, the azimuth and elevation axes are in exact the same initial position. With a conventional parabolic antenna, the effective pointing direction changes neither in azimuth nor in elevation when modifying the skew angle. This major difference between both antenna designs effects the pointing procedure in terms of an offset in azimuth and elevation direction.

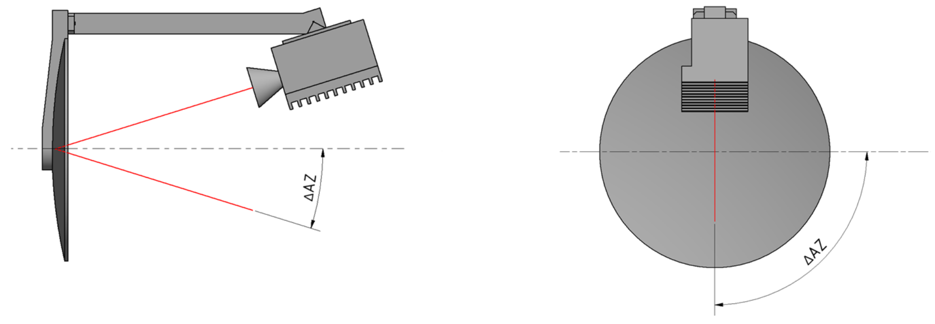


(a) Side view, $\beta = 0°$ **(b)** Side view, $\beta = 90°$

(c) Top view, $\beta = 0°$ **(d)** Top view, $\beta = 90°$

Figure 2.22: Comparison of the side and top view of the pico terminal at 0° skew and at 90° skew. The red lines visualize the simplified electromagnetic beam. With a skew angle of 0°, only an offset in elevation direction is noticeable. With 90°, the effective elevation angle is equal to the angle of the rotary axis, but in azimuth direction an offset occur.

Not only a change of the skew axis has an impact on the effective beam path. Also, a change in elevation leads to a pointing offset in azimuth direction. In Figure 2.23a the pico terminal at 90° skew and 0° elevation is shown. In this case the offset of the main antenna beam to the rotary axis, marked as ΔAZ, has the same value as the offset angle φ, mentioned in the previous section. In comparison, Figure 2.23b represents the pico terminal at 90° skew and 90° elevation. The offset ΔAZ in this scenario is 90°. It can be seen that a change in elevation also effects the effective azimuth direction of the main antenna beam, as long as the skew angle is not equal to 0°.



(a) Top view, 0° elevation **(b)** Top view, 90° elevation

Figure 2.23: Top view at 90° skew: in (a) at 0° elevation and in (b) at 90° elevation. The electromagnetic beams are visualized as red lines. Depending on the elevation angle, azimuth is changing significantly. At 0° skew, the offset ΔAZ in azimuth direction is equal to the offset anlge φ. At 0° skew and 90° elevation the offset ΔAZ increased to 90°.

In summary, the effective elevation direction is dependent on a skew angle change and the effective azimuth direction is dependent on both, a skew angle change and an elevation angle change. To come over this offset effect with a non-zero skew angle an alignment correction algorithm is necessary.

2.2.2 Antenna tracking systems

When tracking, the axis of the antenna beam is kept in spite of the movement of the satellite or the station in the direction of the satellite. Tracking usually becomes necessary when the angular motion of the satellite is of the same order of magnitude as the antenna beamwidth. Without tracking "the ground station would soon loose contact and the communications link would suffer". Various types of tracking are possible and depend on the antenna beamwidth and the amount of apparent movement of the satellite. Considering Equation 2.3, beamwidth depends on the wavelength. For higher frequencies, the wavelength becomes smaller and the beamwidth also becomes narrower. In addition, the dish diameter of the antenna also influences the beamwidth. For bigger diameters, the beamwidth becomes narrower too. The angular width of the beam directly affects the tracking selection. [33],[34]

Four classes of tracking systems are described in [34]:

- manual/programme tracking
- monopulse or simultaneous sensing
- sequential amplitude sensing
- electronic beam squinting

Manual and programme tracking

A commonly used method, especially in satellite television, is **manual tracking**, or in this case, manual pointing. An operator controls the antenna movement until the received signal level is maximized.

With **programmed tracking** the antenna control system is provided with appropriate values of azimuth and elevation angles at each instant. For successive instants, these angles are calculated in advance and stored in a memory. The pointing is done in open loop without determining pointing error between the actual direction of the satellite and the pointing direction. This method is mainly used for ground stations with antennas with a large beamwidth that do not require high pointing accuracy. It is also possible to preposition the antenna in an area of the sky where a non-geostationary satellite will appear. In combination with a closed-loop tracking system operating on the satellite beacon, also high pointing accuracy can be achieved. [33]

Automatic tracking

Automatic tracking is a general term for the remaining tracking techniques, mentioned in the list. With automatic tracking, the antenna direction is continuously aligned to that of a satellite's beacon signal. One method, the so-called **sequential amplitude detection tracking**, makes use of variations in the received signal level, which is a consequence of commanded displacement of the antenna pointing axis. Different procedures are used: conical scanning, step-by-step and electronic tracking. [33]

With *conical scanning*, the antenna beam rotates continuously about an axis. This rotation makes a given angle with respect to the axis of maximum gain. The received level is modulated at the rate of rotation, when the direction of the satellite differs from the direction of the axis of rotation. With generated orientation control signals, the antenna is realigned. [33]

With *step-by-step tracking*, antenna pointing is achieved by searching the maximum received beacon signal. Steps, which are successive displacements, about each of the axes of rotation are used in this case. When comparing the received signal level before and after a step, the subsequent displacement can be determined. Depending on whether the signal level is increasing or decreasing, displacement is either made in the same direction or the direction is reversed. [33]

Another technique is *electronic tracking*, which is comparable with step-by-step tracking. In this case, successive displacement of the beam is realized electronically. This is done by "varying the impedance of four microwave devices coupled to the source waveguide". Depointing can be evaluated by the successive deviation of the beam if the received signal does not arrive along the central axis (boresight) of the antenna. [33]

In a **(multiple source) monopulse system** each source has a slightly shifted radiation pattern with respect to the principal axis of the antenna. Waves arriving parallel to the principal axis have the consequence that the difference between the two received signals from both sources becomes zero. This difference is (approximately) proportional to the depointing angle. [33]

Preferred tracking method for the pico terminal

Comparing manual or program tracking with automatic tracking, it becomes clear that the latter method requires the measured signal for further position determination. This also means that data must be exchanged between the control unit and the RF equipment via an interface. The implementation of an interface represents an additional effort, is device-dependent and has the consequence that no data is displayed on the RF equipment when the interface is active. Program tracking is based on precalculated position data and can be used without additional measuring equipment. This makes handling easier in laboratory lessons and offers the possibility of tracking satellites without measuring the received signal. Due to the advantages of program tracking, this method is used in the present work, as it is more suitable for the laboratory exercise.

2.2.3 Related work

In this section, various scientific papers in the field of antenna tracking in combination with ground stations are presented. The focus is on the method of program tracking, as this method is used in the present work. Also papers in determining antenna parameters, object attitude and object position are mentioned.

Antenna parameters such as the offset angle of a parabolic antenna dish can be determined in various ways. One way to calculate the offset angle based on the antenna geometry is shown in [35]. With the minimal diameter of the antenna as width and the largest antenna diameter as height the offset angle can be calculated. This calculation is based on two properties, mentioned in [35]:

- A projection of any flat section of the paraboloid of revolution onto the axis of the surface is an ellipse.
- An orthogonal projection of this ellipse onto a plane at right angles to the axis of the surface is a circle.

On the one hand, using this calculation is a good method to determine the offset angle by just taking two measurements of the antenna. On the other hand, this method has some limitations. Depending on the geometry, the result can vary by several degrees if the measurement is inaccurate. It is not always clear which area of the antenna is being used effectively. This can also lead to errors when determining the width and height. In this work an optional method is shown, to determine the offset angle of a parabolic antenna dish. More information to the equation and the alternative method can be found in Section 3.1.

To be able to track satellites with an antenna, the position of the satellite must be known. A common method to determine this position is based on the NORAD TLE set format. In previous works various mathematical models are described to calculate and visualize the position of a desired satellite [36], [15].

An algorithm based on TLE to estimate the satellite's position and to update the orbital elements is shown in [37]. This is also a software implementation without any hardware components. In comparison, position calculations in this work are realized with available libraries. It is also not required to implement update procedures for better pointing or tracking results. Other papers based on TLE including hardware components describe the implementation of ground stations and are similar to the project in this work in terms of determining the position of a satellite and aligning antennas via motor drives [38], [39]. A serial connection between the control unit and a PC running Orbitron is used in both references. Orbitron is a pre-passade and real-time prediction program based on TLE which is used to deterine azimuth and elevation data. To adjust the antenna axis, stepper motor drives on the one hand and servomotor mechanism on the other hand are used. In one configuration an additional compass sensor is used to read actual azimuth and elevation angles.

The pico terminal is a transportable device and can be used at different locations. For this reason, the actual position is not fixed and need to be measured. An object tracking system based on GPS and IMU sensor is discussed in [22], processed by a microcontroller. With this system it is possible to determine the static position and the attitude of an object. The IMU used in this system is a MPU6050, combining an accelerometer and a gyroscope in one board. With a similar setup, the position of the pico terminal and the axis orientation of the antenna can be determined. In Section 3.3.3 the used sensors in this work are introduced and in Section 3.4.3 the software implementation is shown.

Chapter 3

Methodology

As part of a literature search in the previous chapters, satellite tracking methods used in ground stations were searched and which option can best be implemented with the given means. The search terms were satellite tracking, antenna tracking system, signal-based tracking, program-based tracking, two-line elements, NORAD, GPS, IMU and antenna offset. Related works found in the literature with the used search terms are mentioned in Section 2.2.3.
The literature search has shown that a tracking system based on TLE data is used successfully in similar applications. The implementation of this method on the pico terminal can also be realized in a time frame suitable for this thesis. Additional hardware such as stepper motors and a compass sensor, as mentioned in [39], are inexpensive and can be added to the existing hardware with appropriate holding parts. A control unit with the necessary interfaces was selected to control the motor drives and to communicate with the sensors. In addition, the device should be network-compatible and provide a graphical user interface for easy operation.

This chapter describes the hardware and software implementation according to the requirements.
Before the adaptations could begin, the first step was to inspect the pico terminal. Then considerations were made as to how additional hardware components could be combined with the existing system.
Before the implementation of the control software could begin, the offset of the parabolic antenna had to be determined and a mathematical model for calculating the correction angle based on the geometry of the pico terminal had to be created. The aim of the correction angle calculation is to correct the antenna alignment depending on the set skew and elevation axes.
During the first tests, the necessary parameters for setting the axes were determined. A relationship between a specified number of steps and the resulting change in the angle of the axes was determined.
In the last section, test methods are described which are used to check the functionality of the modified pico terminal.

3.1 Determination of the antenna offset

When aligning an antenna, the effective direction of the electromagnetic wave is important and need to be considered. In an offset configuration, as used at the pico terminal, the axis of the incoming or outgoing electromagnetic waves is inclined at an angle to the plane of the parabolic dish. This offset angle is explained in Section 2.2.1 and visualized in Figure 2.18.
Due to missing information to the antenna, the unknown antenna offset is determined in this section using various approaches. These approaches are:

- Calculation of the offset angle based on the antenna dish diameter (*Method 1*)
- Determination of the offset angle based on the LNA feed arm geometry (*Method 2*)
- Determination of the offset angle with a digital protractor (*Method 3*)

The offset angle can be calculated using the geometry of the antenna dish, explained in *Method 1*. For *Method 2* and *Method 3*, a few considerations must be made before the determination of the offset angle can be done.

To evaluate if the following measurements deliver the correct results, the point of intersection of the beam on the surface of the antenna dish is determined which hits the center of the antenna feed. This is done, as shown in Figure 3.1, by using the lower edge of a bubble level as a reference plane. The distance l_2 is measured from to the lower edge of the bubble level to the center of the antenna feed. It is important to hold the ruler perpendicular to the bubble level surface to get the correct result. The distance l_1 is determined in the same way, this time measured from the lower edge of the bubble level to the surface of the center point of the antenna dish.

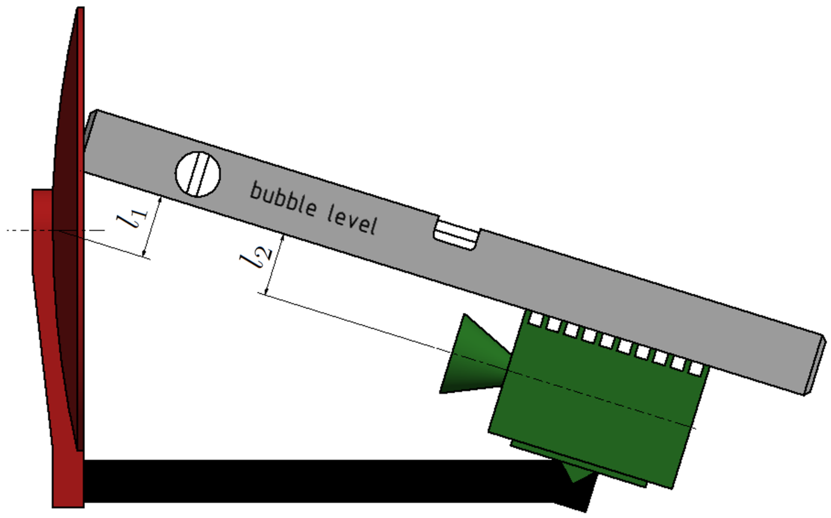


Figure 3.1: Concept to determine the point of reflection of the beam on the antenna surface that is radiated along the rotary axis of the LNA. A bubble level was used to measure the distance l_1 between the center of the antenna dish and the bubble level and l_2 between the center of the LNA feed and the bubble level. If both distances are equal, the beam will be reflected at the center of the LNA along the rotary axis.

If both distances, l_1 and l_2 have the same value, the center of the antenna feed intersects with the rotary axis of the antenna dish. In this case, the consideration shown in Figure 3.2 for determining the offset angle can be used. The thick blue line represents the tangent plane to the antenna dish at the point of reflection. This point of reflection is also the intersection point between the antenna dish and the rotary axis. In other words, the blue line represents an imaginary mirror, reflecting the beam with the same offset angle φ around the rotary axis.

The offset angle φ between the red solid line and the rotary axis of the antenna dish must be determined. This determination is made indirectly by finding similar angles that can be measured. The desired offset angle φ between the red solid line and the rotary axis has the same angle as φ between the rotary axis and the red dashed line. In Figure 3.3 the orange line is perpendicular to the blue line. The angle between the rotary axis and the blue line is also 90°. This results in an angle between the red dashed line and the blue line of 90°-φ. According to the "angle sum theorem" all angles in a triangle add up to 180°, as shown in Equation 3.1. It is obvious, that this angle is φ. Due to the fact, that the green dashed line is parallel to the red dashed line and the orange line of Figure 3.3 is parallel to the antenna feed arm, also the angle between the green dashed line and the feed arm has the same angle φ. Summarized, the offset angle φ between the red solid line and the rotary axis can be determined by measuring the angle φ between the green dashed line and the feed arm.

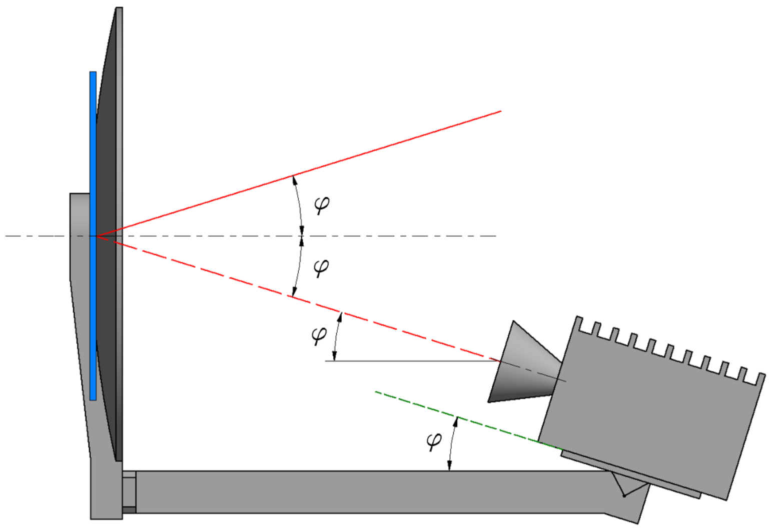


Figure 3.2: Offset angle of the pico terminal: the thick blue line represents an imaginary mirror, reflecting the beam at the center of the antenna dish. It can be seen that the offset angle φ occurs several times.

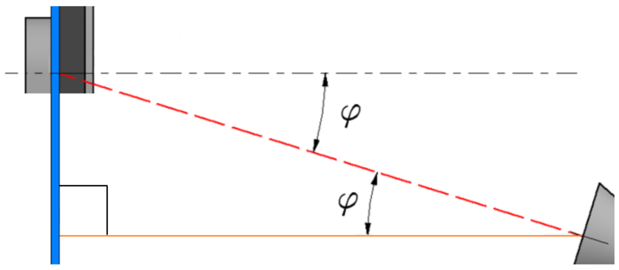


Figure 3.3: Explanation of the offset angle using a triangle. The thick blue line, the reflected beam illustrated as a red dashed line and the orange line parallel to the rotary axis form a right-angled triangle. In this case, the offset angle between the rotary axis and the reflected beam is equal to the angle between the orange line and the reflected beam. For this reason, also the green line shown in Figure 3.2 has the same offset angle φ.

$$\varphi = 180^\circ - 90^\circ - (90^\circ - \varphi) \tag{3.1}$$

Calculation of the offset angle based on the antenna dish diameter

Numerous calculation programs are available for determining the offset angle φ. These calculations are based on the geometry of the antenna dish and use the height h and width w of the dish as calculation parameters, as shown in Equation 3.2 [35].

$$\varphi = arccos\left(\frac{w}{h}\right) \tag{3.2}$$

To calculate the offset angle using Equation 3.2, the calculation parameters need to be determined with a ruler. The height corresponds to the largest diameter, the width to the smallest diameter of the antenna.

Determination of the offset angle based on the LNA feed arm geometry

With this approach, the geometry of the LNA feed arm is used to determine the offset angle. This is done by mounting the feed arm on a solid reference surface, in this case a table top, as shown in Figure 3.4. A clamp holds the feed arm in a stable position when measuring the heights h_1 and h_2 between the lower edge of the feed arm and the table top. This measurement is done at two positions, marked on the reference surface with a predefined distance. To achieve a plausible result, the distance between both markings should be as large as possible. With both heights, Equation 3.3 can be used to calculate the angle φ, which corresponds to the offset angle of the antenna.

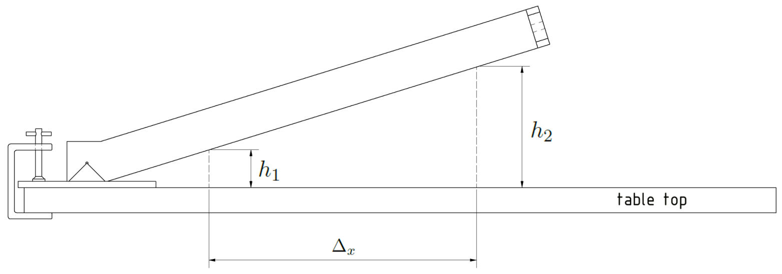


Figure 3.4: Setup to determine the offset angle: with a clamp, the feed arm is hold in place on top of a reference surface, in this case a table. Markings at two positions in a distance of Δ_x on the table top illustrate the measuring positions to determine the corresponding heights.

$$\varphi = arctan\left(\frac{h_2 - h_1}{\Delta_x}\right) \tag{3.3}$$

Determination of the offset angle with a digital protractor

In this method the offset angle is determined using a digital protractor. The feed arm is mounted in the same way as in the previous section shown in Figure 3.4. This time, no reference markings are necessary and instead of a square a digital protractor is set on the feed arm. To ensure a valid measurement, the protractor needs to be calibrated as recommended in the user manual and the table top must be leveled.

3.2 Alignment correction model and calculation

Due to the special design of the antenna unit as explained in Section 2.2.1, alignment correction is required. Depending on the actual skew and elevation angle, azimuth and elevation need to be corrected in a way, that the outgoing or incoming beams are pointing in the desired direction. With the information about the effective direction, the difference to the desired direction can be corrected in both, azimuth and elevation to achieve the best possible antenna alignment. To better understand the different directions and to create sketches for the subsequent calculation, a simplified 3D model of the moveable unit of the pico terminal is created. This moveable unit consists of components 1-3, shown in Figure 2.17.

Simplified 3D model of the pico terminal

A simplified 3D model with the parabolic dish, the feed arm and the LNA is designed with a 3D computer-aided design (CAD) software, as visualized in Figure 3.5. With this model it is possible to simulate different antenna alignment scenarios by changing azimuth, elevation and skew to understand the behavior of the effective beam path with different look angles. The effective beam path represents the effective pointing direction of the antenna. In addition, the model is not only used to create different perspectives and sketches for the correction calculation in the following section but also to measure the effective angles at different scenarios with built-in measuring tools. These measured

angles can thus be compared with the calculated results.
The 3D model is a simplification of the pico terminal with the following opportunities:

- Simplified beam path as a red solid line in the direction of the satellite and a red dashed line between the dish and the LNA.
- Enabling and disabling of different reference planes
- Different parameters for azimuth, elevation and skew to simulate different alignment scenarios
- Possibility to project the beam path onto reference planes
- Possibility to measure effective angles and distances for different look angle settings

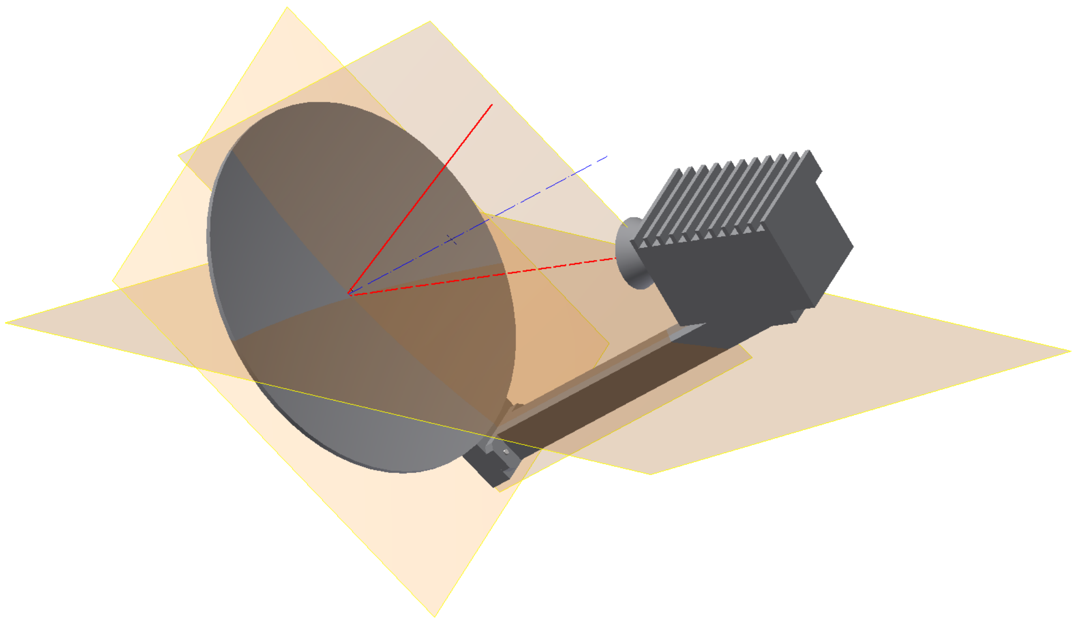
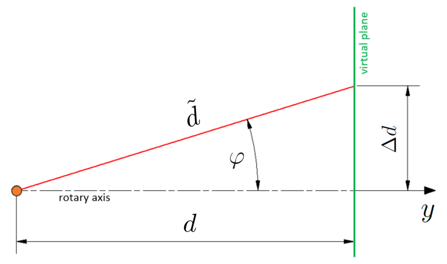


Figure 3.5: Simplified 3D model of the pico terminal. The red solid line represents the simplified beam path, pointing the satellite. This beam path is reflected on the surface of the antenna dish in the direction of the LNA feed and illustrated as a red dashed line. The blue dashed line represents the rotary axis of the antenna dish.

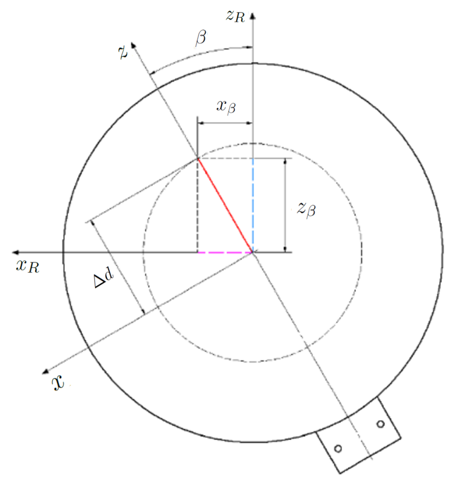
Calculations based on the 3D model

This section describes the calculation method for determining the effective elevation angle EL and the azimuth correction angle α. Two different coordinate systems are used. The x-y-z-coordinate system describes the orientation of the antenna dish whereas the x_R-y_R-z_R-coordinate system describes the reference orientation of a configured azimuth direction at 0° elevation of the rotary axis and 0° skew.

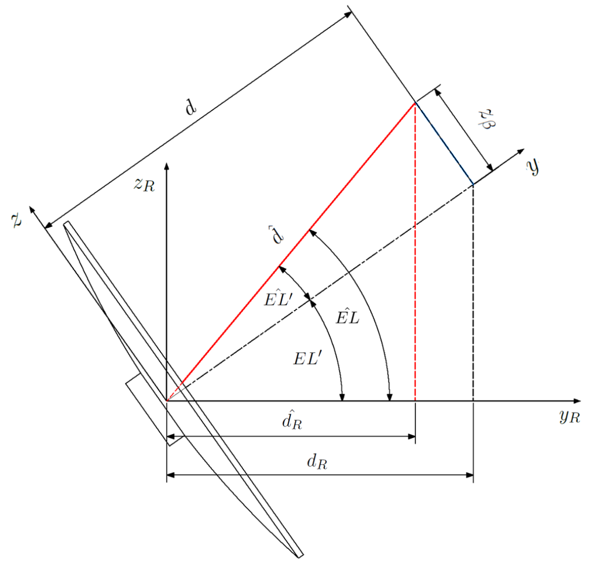
In Figure 3.6 different views of the model, the previously determined antenna offset angle φ and the effective elevation EL are sketched. The red solid line represents the main antenna beam. The blue dashed line and the pink dashed line are projected offsets in azimuth and elevation according to the configured skew and elevation angles. The z_R axis of the reference coordinate system is always pointing to the zenith. The y_R axis is pointing in the configured azimuth direction along the horizon if the skew angle is $0°$. The x_R axis is perpendicular to the z_R and y_R axis in a right-handed coordinate system. A virtual plane parallel to the antenna dish along the rotary axis at distance d to the antenna surface and a horizontal plane parallel to the Earth's surface are also shown in Figure 3.6.



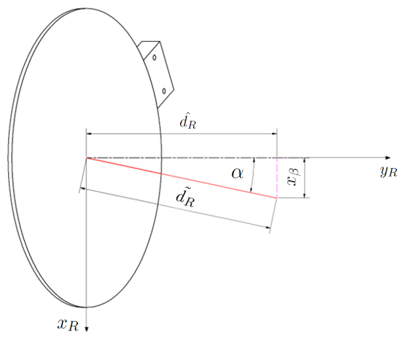
(a) Offset angle



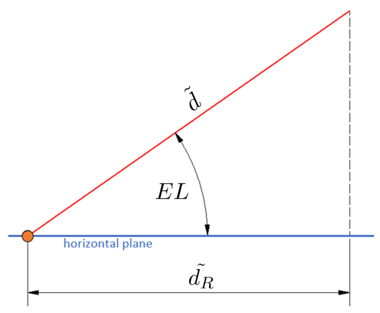
(b) Front view



(c) Side view



(d) Top view



(e) Effective elevation

Figure 3.6: Different views of the alignment correction model: the effective beam path is visualized as a red solid line. The side view represents the projection of the beam path with an offset $\hat{EL'}$ onto the ZY_R plane, where z_R is pointing to the zenith. The front view represents the projection along the y axis. In this view, the beam offsets z_β in z_R direction and x_β in x_R direction are shown. In the top view, the offset angle α in azimuth direction is shown.

In Figure 3.6 all relevant information required for the calculation are visualized. The side view represents a projection of the antenna beam's effective elevation angle onto the YZ_R plane. The total angle of the projected beam measured from the y_R axis is the sum of the configured elevation angle of the rotary axis EL' and the projected offset angle $\hat{EL'}$. EL' is adjusted by the motor drive of the elevation axis. $\hat{EL'}$ is a projection of φ onto the YZ_R plane and depend on the configured skew. At 0° skew, $\hat{EL'}$ has the same value as α. At ±90° skew, $\hat{EL'}$ disappears and the effective elevation angle equals the configured elevation angle.

In the front view the skew angle β, measured from the reference axis z_R, is shown. The length Δd of the projected beam is limited by the virtual plane, as visualized in Figure 3.6d.

In the top view the effective azimuth of the antenna beam is projected onto the XY_R plane and the azimuth correction angle α between the projected beam and the y_R axis is shown.

Two additional sketches illustrate the antenna offset angle φ, measured from the rotary axis in Figure 3.6d and the effective elevation angle, measured from a horizontal plane in Figure 3.6e. The length of the electromagnetic beam $\tilde{d}$ is measured from the center of the parabolic antenna, which is marked as an orange point. The geometric length is limited by the virtual plane, as mentioned earlier. The exact value of the distance d of the virtual plane to the antenna surface measured from the center of the parabolic antenna is not important, during the calculation this value cancels out.

The following equations based on trigonometric functions can be used to evaluate the effective elevation angle of the antenna beam and the azimuth correction angle.

With distance d and the fixed offset angle φ, the projected geometric length of the considered beam vector onto the front view plane Δd as well as the geometric length of the beam vector $\tilde{d}$ can be calculated with Equations 3.4 and 3.5.

$$cos(\varphi) = \frac{d}{\tilde{d}} \tag{3.4}$$

$$tan(\varphi) = \frac{\Delta d}{d} \tag{3.5}$$

Rearranging Equations 3.4 and 3.5 result in:

$$\tilde{d} = \frac{d}{cos(\varphi)}$$

$$\Delta d = d \cdot tan(\varphi)$$

Knowing the projected geometric length Δd and the configured skew angle β the projected offsets x_β in the x-direction and z_β in the z-direction onto the virtual plane can be calculated with Equations 3.6 and 3.7.

$$sin(\beta) = \frac{x_\beta}{\Delta d} \tag{3.6}$$

$$cos(\beta) = \frac{z_\beta}{\Delta d} \tag{3.7}$$

Rearranging Equations 3.6 and 3.7 result in:

$$x_\beta = \Delta d \cdot sin(\beta)$$

$$z_\beta = \Delta d \cdot cos(\beta)$$

Combining the results of Equations 3.5 and 3.6 as well as Equations 3.5 and 3.7, the projected offsets result in:

$$x_\beta = d \cdot tan(\varphi) \cdot sin(\beta)$$

$$z_\beta = d \cdot tan(\varphi) \cdot cos(\beta)$$

Knowing z_β and d, the projected offset angle $\hat{EL'}$ onto the YZ_R plane can be calculated with Equation 3.8.

$$tan(\hat{EL'}) = \frac{z_\beta}{d} \tag{3.8}$$

Rearranging Equation 3.8 results in:

$$\hat{EL'} = arctan\left(\frac{z_\beta}{d}\right)$$

The projected geometric length $\hat{d}$ of the considered beam vector onto the YZ_R plane can be calculated with Equation 3.9. In Equation 3.10 the sum of the elevation angle of the rotary axis EL' and the projected offset angle $\hat{EL'}$ is calculated.

$$\hat{d} = \sqrt{d^2 + z_\beta^2} \tag{3.9}$$

$$\hat{EL} = EL' + \hat{EL'} \tag{3.10}$$

Knowing $\hat{EL'}$ and $\hat{d}$, the length $\hat{d_R}$ can be calculated, which is the projection of $\hat{d}$ onto the y_R axis as shown in Equation 3.11.

$$cos(\hat{EL}) = \frac{\hat{d_R}}{\hat{d}} \tag{3.11}$$

Rearranging Equation 3.11 results in:

$$\hat{d_R} = \hat{d} \cdot cos(\hat{EL})$$

In Equation 3.12 the projection $\tilde{d_R}$ of the beam vector onto the XY_R plane is calculated and depends on the projected offset x_β and the projected beam vector at distance $\hat{d_R}$, measured in y_R direction.

$$\tilde{d_R} = \sqrt{\hat{d_R}^2 + x_\beta^2} \tag{3.12}$$

The angle α between the y_R-axis and the projected beam vector is calculated in Equation 3.13.

$$tan(\alpha) = \frac{x_\beta}{\hat{d_R}} \tag{3.13}$$

Rearranging Equation 3.13 results in:

$$\alpha = arctan\left(\frac{x_\beta}{\hat{d_R}}\right)$$

The effective elevation angle EL is calculated in Equation 3.14. It is the angle between the beam vector and the XY_R plane.

$$cos(EL) = \frac{\tilde{d_R}}{\tilde{d}} \tag{3.14}$$

Rearranging Equation 3.14 results in:

$$EL = arccos\left(\frac{\tilde{d_R}}{\tilde{d}}\right)$$

By combining Equations 3.4 - 3.14, the azimuth correction angle and the effective elevation angle of the antenna beam for given skew angle and elevation angle of the rotary axis result in:

$$\alpha = arctan\left[\frac{tan(\varphi)sin(\beta)}{\sqrt{1+tan^2(\varphi)cos^2(\beta)} \cdot cos\Big[EL' + arctan\Big(tan(\varphi)cos(\beta)\Big)\Big]}\right] \tag{3.15}$$

$$EL = arccos\left[cos(\varphi) \cdot \sqrt{\Big(1+tan^2(\varphi)cos^2(\beta)\Big) \cdot cos^2\Big[EL' + arctan\Big(tan(\varphi)cos(\beta)\Big)\Big] + tan^2(\varphi)sin^2(\beta)}\right] \tag{3.16}$$

3.3 Hardware design

This section describes the hardware components used to upgrade the pico terminal. Main parts are described in more detail, whereas add-on parts are listed and briefly explained at the end of this section.

3.3.1 Choice of motor drives and their position

The most important components of the upgraded pico terminal, apart from the control unit, are the motor drives. These drives are used to align the azimuth and elevation axis. Depending on the axis, different torque is required to rotate the axis shaft. The torque of the elevation axis was determined using a rod connected with the axis and a hand scale. According to the lever principle a torque of 2 Nm was determined.
Due to the good positioning properties of stepper motors, which are available in different sizes, these motors are a possible option to motorize the pico terminal. In combination

with micro step drivers, stepper motors are simple to control and position changes can be determined by counting steps. In combination with encoders, step losses can be recognized and compensated to increase positioning accuracy. In addition, with absolute encoders the current axis position can be determined at any time without the need of defining a reference position.

To connect the motors to the corresponding axes and to combine different operating modes, different approaches are considered and explained in the following subsections. These approaches are divided into two variant groups. In the first group, Variants 1-3 describe the mechanical implementation of the motor drives, while in the second group, Variants A and B describe methods for combining manual and automatic operation. The result is a combination of one variant per group.

The available operating modes are:

- **Manual mode** - axes of the pico terminal can be adjusted by moving a handwheel or a rotary encoder. This operating mode is required to align the antenna dish in any direction. More information can be found in Section A.7.
- **Automatic mode** - axes of the pico terminal are automatically adjusted by selecting a desired satellite with the control unit. This mode was not available on the original pico terminal.

Variant 1: Directly mounted stepper motors using shaft couplings

With this approach, the stepper motors are connected via shaft couplings to the axis of the pico terminal. A 3D model of the shaft coupling, manufactured with a 3D printer is shown in Figure 3.7. This coupler in combination with a NEMA 17 stepper motor is one option to motorize the azimuth axis of the pico terminal. Because of the size of a NEMA 17 stepper motor, the height of the motor shaft fits the height of the pico terminal's azimuth axis shaft. With a L-bracket mounted on the ground plate, the motor can be held in place. For the elevation axis a directly mounted stepper motor is only possible at the antenna front but difficult to realize, because there is no possibility to hold the motor in place.

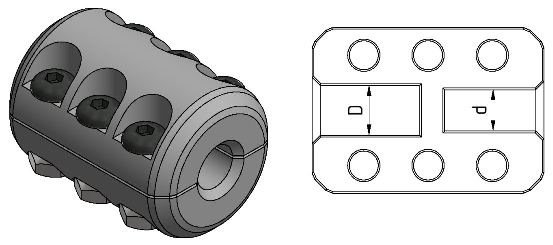


Figure 3.7: 3D printed coupler for azimuth axis. This part was used in first tests to connect a NEMA 17 stepper driver to the azimuth axis. At one end the shaft of the stepper motor and at the other end the shaft of the azimuth axis were clamped with six screws.

Variant 2: Using gear wheels to move the skew axis

This approach was considered specifically for the skew axis. A smaller gear on the stepper motor shaft transfers the movement to a larger gear. A faster rotation with a lower torque on the drive side thus leads to a slower rotation with a higher torque on the skew axis. For this purpose, the stepper motor must be mounted at a certain distance so that the teeth of both gears mesh with one another.

Variant 3: Combination of a timing belt and pulleys

A combination of different sized pulleys on the shafts connected via a timing belt is considered. One pulley is placed on the stepper motor shaft, the second one on the pico terminal's axis shaft. With this approach there is a higher flexibility in terms of positioning the stepper motors. Timing belts are available in different lengths and must match the used pulley type. As in variant 2, the transmission ratio can also be varied by using different pulley sizes. A pulley with less teeth on the motor shaft compared with the pulley on the axis shaft, reduces the number of revolutions of the pico terminal axis but increases the torque.

Variant A: Independent manual and automatic operating

To ensure manual adjustments using a handwheel in the same way as in the previous version of the pico terminal, two independent operating modes are considered. This means that the axes can be set either manually with a handwheel mounted on the shaft or automatically via the control unit. In this case, the stepper motors must be deactivated when the axis is set manually, otherwise the holding torque of the stepper motors will act against the rotary movement. Rotary encoders are also required for position changes in manual mode. Otherwise, the axes would have to be referenced each time when switching back to automatic mode.

Variant B: Semi-automatic mode for manual adjustments

With the semi-automatic approach there is also the possibility for manual adjustments. In contrast to Variant A, no handwheel is mounted on the shaft of the axis. In this case, an external encoder is used to recognize the user's adjustments. This encoder is located on a manual control panel connected to the control unit. The rotary movements of the encoder are converted into a configurable number of steps by the control unit.

3.3.2 Processing unit

The processing or control unit acts as a Human Machine Interface (HMI). It communicates with the stepper motor drivers and all sensors and can handle user input and output. To control the pico terminal, a processing unit is required that offers the following options:

- I/O capability to connect additional hardware
- Support of a Graphical User Interface (GUI)
- Internet capabilities for remote access

Different single board computers with the mentioned requirements are available on the marked.

3.3.3 Sensors

9-axis-sensor (IMU)

A 9-axis-senso, mounted on the feed arm of the pico terminal delivers actual values for roll, pitch and yaw. Roll is equivalent with the skew angle, pitch with the elevation angle and yaw with the azimuth angle. The measured values are only visualized in the GUI and have no effect on the positioning process during antenna alignment. The sensor can also be used as a reference device to determine the azimuth and elevation starting points in the configuration phase after switching on the pico terminal. To get valid data out of the sensor, calibration is necessary. The exact procedure can be found in the operating instructions of the sensor.

GPS receiver

To determine the actual GPS position of the pico terminal, a GPS receiver is used. The location is necessary to calculate the required look angels using TLE data set format. When connected to a Linux system, a file named *ttyACM* is created and located in the */dev/* directory. In this file the updated NMEA sentences are available and can be accessed with a terminal or by a program.

Limit switch

To avoid collisions between different hardware parts, a limit switch is used on the lower position of the elevation axis. One possibility to connect the limit switch is shown in Figure 3.8. The limit switch is located between the enabled signal of output DO1 and the ENA+ pin of the micro step driver. The ENA+ pin enables the stepper motor if ENA- pin is connected to GND. The dashed line illustrates a possible method to short the limit switch if the lower position is reached and elevation should be increased. Otherwise, the limit switch in the lower position always interrupts the ENA signal in the normally closed mode and the position can never be left.

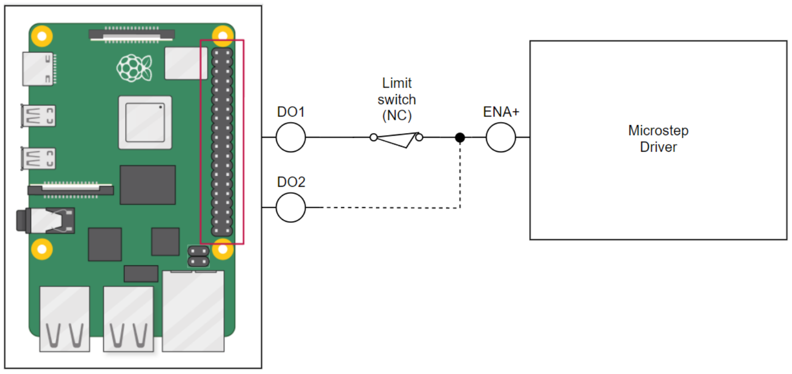


Figure 3.8: Limit switch in normally connected (NC) configuration. In the lower position the connection between the micro step driver and the control unit is interrupted. To release the end position, an additional signal (dashed line) is necessary to enable the micro step driver.

Another option to implement a limit switch is shown in Figure 3.9. In this case, the limit switch closes the circuit in the normally open mode and pulls the level on the digital input (DI) of the processing unit high. This high level is recognized by a subroutine and the stepper motors are disabled.

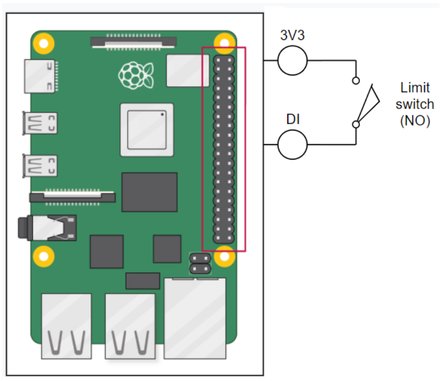


Figure 3.9: Limit switch in normally open (NO) configuration. The control unit disables the micro step driver, if the lower position is reached. In this case, the limit switch has no direct connection to the micro step driver.

Control panel with rotary encoder for manual adjustments

A manual control panel with an axis selection switch and a rotary encoder is used to detect user inputs to change azimuth or elevation manually. The control panel is connected to the control unit and replaces the previously used handwheels. The axes can be adjusted more comfortably and without physical effort. More information can be found in the user guide in Section A.7.

3.3.4 Additional hardware

Additional hardware parts are either available in local hardware or web stores or need be be designed and manufactured. Table 3.1 gives an overview of all parts used in this project and their source of supply.

Table 3.1: List of additional hardware parts used on the pico terminal. Some parts are purchased, others manufactured via CNC milling or 3D-printing (illustrated with checkmarks (✓) in Columns *Purchased* and *Manufactured*.

Component	Quantity	Purchased	Manufactured
Ground plate	1	✓	
Adjusting screws	4	✓	
Circular level	1	✓	
Pulleys	4	✓	
Timing belts	2	✓	
Aviation connectors	2	✓	
Wires	N/A	✓	
Motor mount (elevation)	1		✓
Motor mount (azimuth)	1		✓
Motor connector adapter	2		✓
9-axis-sensor mount	1		✓
End stop mount	1		✓
Feet	4		✓
Clamps	2		✓

The following manufacturing processes are used to create the parts:

- Computerized Numerical Control (CNC) milling
- 3D printing

3.4 Software design

Software is necessary to run different instructions on the single board computer and to control the general-purpose input/output (GPIO), which is the interface to external hardware components like sensors. In this section the basic functions of the pico terminal program are described and the program sequences are graphically represented with simplified flow charts. A widely used programming language to create the software source code on single board computers is Python. ”Python is an interpreted, interactive, object-oriented programming language. [...] Python combines remarkable power with very clear syntax.” [40] ”Python comes preinstalled on most Linux distributions, and is available as a package on all others.” [40]

3.4.1 GUI elements

Apart from the manual control panel, the operator interacts with the hardware via a Graphical User Interface (GUI). With the programming language Python, the build-in GUI framework *Tkinter* can be used to program a GUI. Due to the high number of GUI objects, the external software package *PAGE* is used to create the Layout. With the Drag & Drop function each object can be placed on the appropriate position. The finished layout can be exported as a Python script, which consists of the code for the GUI. This script can be customized later using the appropriate instructions.

3.4.2 Accessing the GPIO to communicate to external hardware

Depending on the single board computer, a different number of GPIO pins are available. Each pin can be accessed by the program via a GPIO library. With the library functions, the behavior of the pins can be adapted to set a pin either as an input or an output. Specific GPIO pins support additional functionality, which can be configured with the corresponding interface library. In this project, Inter-Integrated Circuit (I^2C) bus was used to communicate with the IMU. I^2C is available on most single board computers and widely used to communicate to external periphery.

3.4.3 Program sequence and functions

Various instructions in different Python scripts are executed in parallel to control the stepper motors and to visualize the actual position or sensor values. This is achieved by running multiple instructions at the same time, called threads. Some threads are started once at the beginning to update process data. Others are started via bindings by clicking on buttons.

Get actual IMU values

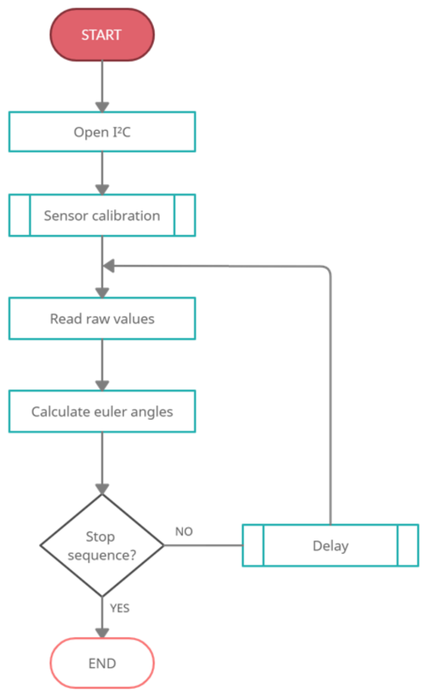
One Python script is executed to determine the actual Euler angles, measured with the IMU sensor. This script is called by the main program as a thread. Depending on the used IMU sensor and the corresponding library, different function calls are available. Sensors in a lower price range like the MPU9250 only deliver raw values for acceleration, angular velocity and magnetic field on the one hand. In this case, additional calculations are necessary to get the desired values. More expensive sensors on the other hand use a build-in micro controller for additional calculations. In addition to raw values, also the Euler angles and actual calibration statuses are delivered over the interface.

A sensor with a build-in micro controller is the BNO055. Depending on the configured sensor mode, different sensor values are transmitted with different data rates. Configured in $NDOF\_MODE$, a fusion mode with nine degrees of freedom is used. In this mode, the fused absolute orientation data is calculated from the internal sensors. More information can be found in [41].

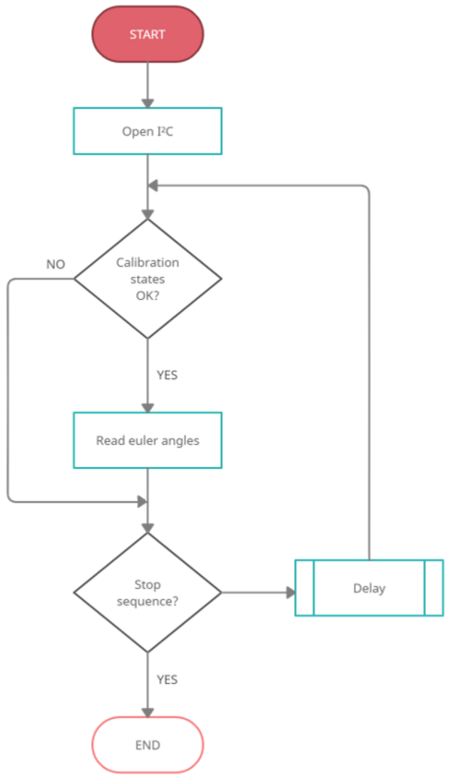
The flow charts in Figure 3.10 represent the simplified program sequence used to communicate with the two IMU sensors.

Get actual GPS values

A GPS sensor delivers actual values for date, time and the geographic position. To access all necessary GPS values in the main program, a Python script is called as a thread. In Figure 3.11 the simplified program sequence of this script is shown. GPS data is packed in different NMEA sentences. Each sentence starts with an NMEA ID, which is compared to a list of possible IDs. Depending on the ID, values of different sentence positions are combined and formatted to the desired values.



(a) MPU9250



(b) BNO055

Figure 3.10: Flow charts of the IMU sequences. In first tests, the MPU9250 was used. This sensor needs to be calibrated once at the beginning with a function call. In comparision to the BNO055, there is no build in microcontroller. Euler angles need to be calculated. The BNO055 delivers actual values without additional calculations later on. In addition, the calibration process always runs in the background and the actual calibration statuses can be checked to identify valid data.

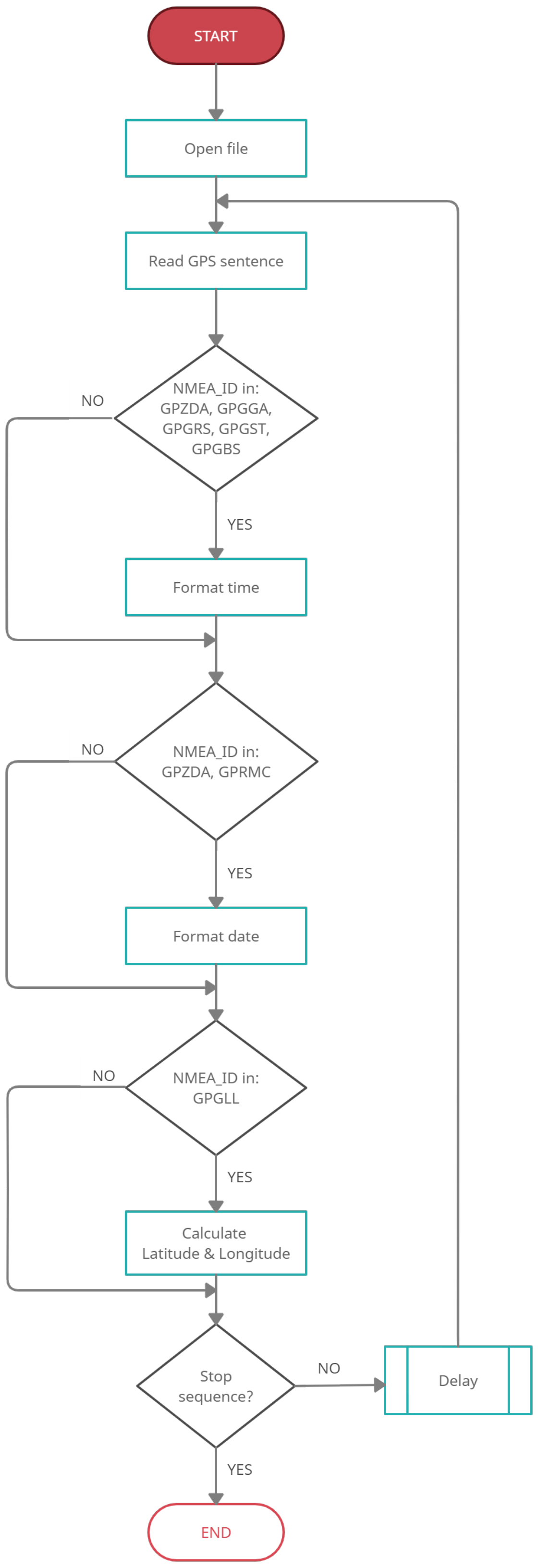


Figure 3.11: Flow chart of the GPS sequence. At the beginning the */dev/ttyACM** file (see Chapter 3.3.3) is opened. A while loop is running, until the sequence is stopped by the main program. In the while loop, the identifier of the *NMEA* sentences is compared to a list of known codewords. Depending on the identifier, the record is either extracted or ignored.

The longitude and latitude position values are available in the *GPGLL* sentence. Additional calculations are necessary to convert the data into a format that is easy to read. Two formats are used to visualize the GPS location in the GUI, decimal degree and the sexagesimal format in degree, minutes and seconds.

Reading TLE data and calculating satellite position

To calculate actual satellite positions, a library is used. For the azimuth and elevation calculations, the library requires current TLE and GPS data for the calculations. For this purpose, actual TLE files can be copied or directly downloaded from an online database. The effective values for azimuth and elevation are returned to the main program for further operation.

Moving the stepper motors and calculate actual positions

To change the angular position of the azimuth or elevation axis, the stepper motors have to move a defined number of steps. A single step is an impulse, which is created by switching a certain digital GPIO pin between high and low and vice versa. The number of necessary steps to change the axis position need to be calculated. Due to the hardware design with different pulley sizes and a transmission belt, the gear ratio must be considered in the calculations. A soft start/stop function is used to avoid jerking by reducing the speed of the motors at the start and the end of the movements. To change the motor speed, the delay between two impulses can be configured.

Settings and calibration statuses

Additional information and sensor settings can be accessed through the menu. For more details, see Section A.8.3 of the user guide.

3.5 Test methods

This section describes various test methods to verify the functionality of the pico terminal or to determine the time it takes to calibrate the sensors before the pico terminal can be used. Some tests relate to specific hardware components, others check the pointing and tracking behavior of the system.

3.5.1 Method to determine the average calibration time of the 9-axis-sensor

To determine the average calibration time of the 9-axis sensor, the internal calibration statuses of the accelerometer, gyroscope and magnetometer are evaluated during the calibration process. The time difference between the beginning and the end of each calibration process is measured and indicates the respective calibration time. To ensure a defined start situation for the measurements, the sensor is de-energized before each test.
The overall calibration time, as the sum of all individual calibration times, divided by the number of measurements is the average calibration time of the 9-axis-sensor. To avoid inaccuracies due to reaction time when measuring by hand, a Python script is used.

3.5.2 Method of determining the average time until valid GPS data is received

Depending on the environment, it can take a few minutes until valid data can be read from the GPS receiver. A status entry in a specific NMEA sentence indicates whether the data is valid. In this test the time from cold start with the unplugged GPS receiver to valid GPS data is measured. To validate the GPS data, the sensor values must be compared with previously determined position data of the measurement location.

3.5.3 Test method to evaluate trueness, precision and accuracy of the elevation axis

To validate the configured settings of the motor drives, this test method is used. Due to the geometry of the pico terminal, the leverage of the feed arm is the highest at 0° elevation of the rotary axis. This behavior might affect the accuracy of the elevation axis, depending on the actual elevation angle. All measurements start at the same reference position defined by the limit switch. To move the stepper motors, a defined angular difference is used as a reference value. To compare trueness, precision and accuracy at different elevation ranges, measurements are carried out at two different reference values for the desired elevation angle movement.

3.5.4 Determination and validation of the angle error during the automatic alignment

In this test setup the automatic alignment of the pico terminal, based on TLEs, is compared to an alignment of the pico terminal in manual mode. In both scenarios additional RF equipment is attached to the LNA to measure the signal power level of the received satellite beacon signal. To ensure the best possible communication between a satellite and the ground station, different aspects are important. Apart from atmospheric influences, misalignment of the antenna dish has a negative effect on the power level of the received signal. A parameter to determine antenna alignment, which is used in this test method, is the HPBW. With this test setup the following points where examined:

- Alignment offset from best possible signal power level
- Alignment position within HPBW

The first part of this test setup evaluates the angular offset of azimuth and elevation from the automatic pointing alignment to the intended alignment with the best possible signal power level. The results can then be used to determine the functionality of the alignment process.
For the second part of this test, the HPBW must be determined immediately after the automatic alignment and can then be used to evaluate the signal level at different antenna alignments. To ensure a meaningful comparison of the results, the weather conditions must be stable and due to the satellite movements, the entire test procedure should be carried out quickly.

Procedure

After pointing a geostationary satellite in automatic mode, the received signal on the RF equipment and the actual values for azimuth and elevation displayed on the GUI are stored for later comparisons. Right after the automatic pointing the manual mode is used to modify one of the two motorized axes in a way, to achieve the highest possible signal power level. This is done by adjusting the elevation axis while the azimuth axis remains in its fixed pointing position. The alignment with the highest signal power level needs to be stored. Based on this antenna orientation the elevation axis is adjusted with a fixed step interval in both directions until the signal power level drops by 3 dB to evaluate the HPBW. For each step interval the power level is stored. At the same time, the azimuth axis is still in its fixed pointing position.
The same procedure is repeated for the azimuth axis. In this case, the elevation axis remains in its fixed pointing position after the second automatic pointing attempt.

3.5.5 Verification of the automatic tracking function

Geostationary satellites remain essentially motionless when observed from a certain location on the Earth's surface but they can show a daily movement, depending on the inclination. In contrast, LEO-satellites change their relative position quickly. Especially in the second case a periodic alignment update is necessary to maintain an uninterrupted communication link during the visibility time slot of a few minutes.
In this particular test case the daily movement of a geostationary satellite is tracked over a time period of half a day. During the tracking procedure the actual values of azimuth, elevation and the 9-axis-sensor are logged. This test is used to verify the calculated azimuth and elevation angles at certain timestamps with available online data and to compare the calculated values with the measured values of the 9-axis-sensor. It can be assumed that the value changes between calculated and measured values are constant. In addition, a graphical representation of the daily movement of the satellite can be created.

3.5.6 Evaluation of the maximum signal power level drop of an inclined GEO satellite over half a day

Similar to the previous test the daily movement of a geostationary satellite is observed. This test should illustrate the maximum signal power level drop, which occurs without tracking. In this test scenario the antenna dish of the pico terminal is aligned only once at the beginning. With additional RF data logging equipment, the beacon signal is measured over a time period of half a day. During the whole measurement, no alignment updates are done. The satellite movement causes a misaligned pico terminal antenna. This misalignment affects the measured signal power level. Depending on the starting orientation of the antenna, the signal power level can either increase, decrease or both during the measurement.

3.5.7 Tracking of an inclined GEO-satellite for half a day

This test is used to validate the automatic tracking function and is based on the previous test setup. The only difference is the periodic alignment update interval during the measurement. Instead of one alignment at the beginning, the antenna of the pico terminal need to follow the satellite's movement by realigning periodically. In general, the measured signal behavior is assumed to be constant. Due to possible step losses when aligning the axes, the signal power level might decrease over time.

Chapter 4

Results and discussion

In the previous chapter, methods to determine the antenna offset angle and the implementation of an alignment correction model are explained. The hardware and software design are covered and different test methods to verify the functionality of the pico terminal are presented.
In this chapter the associated results are presented and discussed.

4.1 Determination of the antenna offset

The antenna offset φ is determined with three different methods, as described in Section 3.1.

4.1.1 Method 1

For *Method 1* the height h and the width w of the antenna dish is required. Both diameters were measured with a ruler. Using Equation 3.2, the antenna offset φ_{M1} was calculated. The measured antenna diameters and the calculated offset angle are shown in Table 4.1.

Table 4.1: Antenna dish parameters of the pico terminal. The width w and the height h of the antenna dish were measured with a ruler. The calculated offset angle with *Method 1* φ_{M1} was determined using Equation 3.1.

w	h	φ_{M1}
340 mm	355 mm	16.72°

4.1.2 Methods 2 & 3

For *Methods 2 and 3*, the distances $l1$ and $l2$ as shown in Figure 3.1 where measured. Since both distances have the same value of 65 mm, the offset angle evaluation can be carried out as explained in Section 3.1. The test setup used to measure the necessary heights and angles is shown in Figure 4.1.

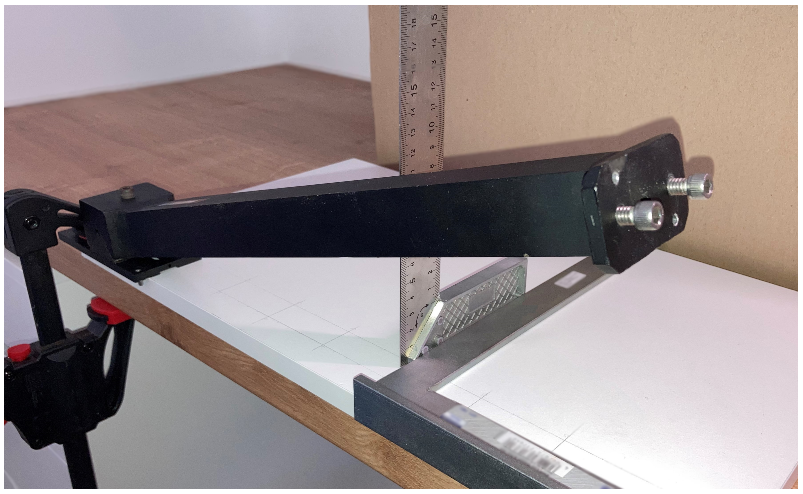


Figure 4.1: Setup to determine the antenna offset using Methods 2 & 3. With a clamp, the feed arm is hold in place. One angle is used as a stop and a second angle is used to measure the height between the bottom of the feed arm and the white plate. Markings on the left and the right side at a distance Δ_x of 150 mm on the white plate indicate the measuring positions.

In Table 4.2 the measured heights h_1 and h_2 as illustrated in Figure 3.4 and the calculated offset angles φ_{M2} using Equation 3.3 are shown.

Table 4.2: Determined offset angles with *Method 2*. The measured heights h_1 and h_2 as well as the calculated offset angle φ_{M2} are shown.

h_1	h_2	φ_{M2}
43.5 mm	89 mm	16.874°

With a similar setup but a different approach the antenna offset was determined with *Method 3*. In this case, a digital protractor was used, to evaluate the offset angle φ_{M3}. The digital protractor was placed on top of the inclined feed arm. In total four measurements were made. The average value of all measurements results in an offset angle of 16.93°. When measuring with a digital protractor, the table top must be leveled, as in this case the angle is determined directly.

Summing up, all three methods delivered an offset angle between 16.72° and 16.93°. It is assumed that the actual intended offset angle has an integer value. For this reason, the default offset angle φ for the control unit is set to 17°.

4.2 Alignment correction

To verify Equations 3.15 and 3.16, an Excel file with the calculated values for the effective elevation EL and the azimuth offset ΔAZ are compared to the measured values of the 3D CAD model.

4.2.1 Result verification with Excel datasheet

For different values of EL' and the skew angle β, the effective elevation angle EL and the angular offset of the azimuth axis ΔAZ were calculated. Some significant values of the Excel file are transferred into Table 4.3.

Table 4.3: Alignment correction for azimuth and elevation for a given skew angle β and a configured elevation angle of the rotary axis EL'. The effective elevation angle EL and the necessary azimuth correction angle ΔAZ are shown for specific values. Depending on the skew angle, EL and ΔAZ have different values for the same EL'.

no.	EL'	β	EL	ΔAZ
1	0°	0°	17°	0°
2	0°	90°	0°	17°
3	90°	0°	73°	**0°**
4	90°	90°	73°	90°
5	-17°	0°	0°	0°
6	73°	0°	90°	0°
7	35.2°	90°	33.45°	20.51°

Apart from the evaluated value for ΔAZ in line no. 3, all results seem plausible. In line no. 3, an azimuth correction of 180° might be the correct value. Considering the restricted domain for the elevation axis in the range $-\varphi \leq EL' \leq 90 - \varphi$, the parameter EL' is out of range and can be ignored. The values in line no. 7 are compared to the measured values of the 3D CAD model in the next section.

4.2.2 Result verification with a 3D CAD model

With the 3D CAD model, the parameters for azimuth, elevation and skew can be changed. By projecting the simplified beam, visualized as a red line shown in Figure 3.5 onto the horizontal plane, the angular difference in azimuth and elevation can be determined. In Figure 4.16, the effective elevation angle between the beam and the projected beam ($EL = 33.45°$) and the configured elevation angle of the rotary axis ($EL' = 35.2°$) at a configured skew angle ($\beta = 90°$) are shown. The measured value also matches with the value in line no. 7 of Table 4.3. Angular offsets in azimuth direction only appear if the skew angle is not zero. This can be seen in Table 4.3 and can be confirmed with the 3D model.

4.3 Hardware

4.3.1 Motor drives

To motorize the azimuth and elevation axes, stepper motors with the corresponding micro step drivers are used. In this case, an open-loop system with NEMA 23 stepper motors provide the necessary torque. An open-loop system has no feedback sensors to control the motor movements or to measure the orientation of the axes. Possible step losses remain undetected and lead to a positioning error.

To combine the automatic mode with manual adjustments by the operator, a combination of Variant 3 and Variant B as explained in Chapter 3.3.1 was implemented. With timing belts, more flexibility in placing the necessary motor mounts is given. The motors are held in place with manufactured motor mounts. The transmission ratio can be adjusted by selecting different pulley sizes. In this case, a stepper motor with a shorter length is sufficient to produce the same torque as a more powerful motor without a gear ratio.

In Figure 4.2 the motor drive of the elevation axis and in Figure 4.3 the motor drive of the azimuth axis are shown.

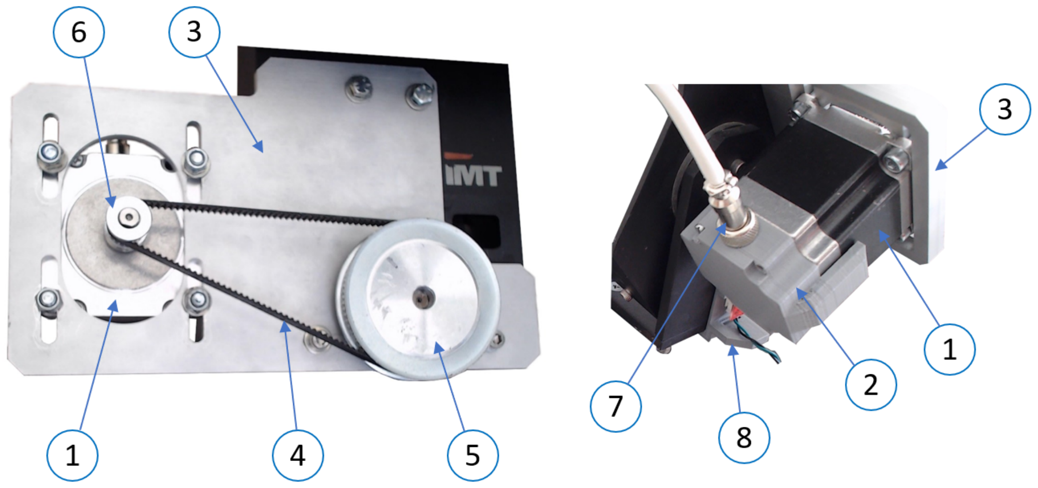


Figure 4.2: Motor drive of the elevation axis at front and back view: the NEMA 23 stepper motor (1) with an attached 3D printed motor connector adapter (2) is hold in place by the milled aluminium motor mount (3). A timing belt (4) is used for power transmission between the driven pulley of the axis shaft (5) and the driving pulley of the motor shaft (6). To adjust the tension of the timing belt, long holes in the motor mount are available. With an aviation connector (7) the cable between micro step driver and motor can be unplugged. The 3D printed end stop mount (8) holds the limit switch in place.

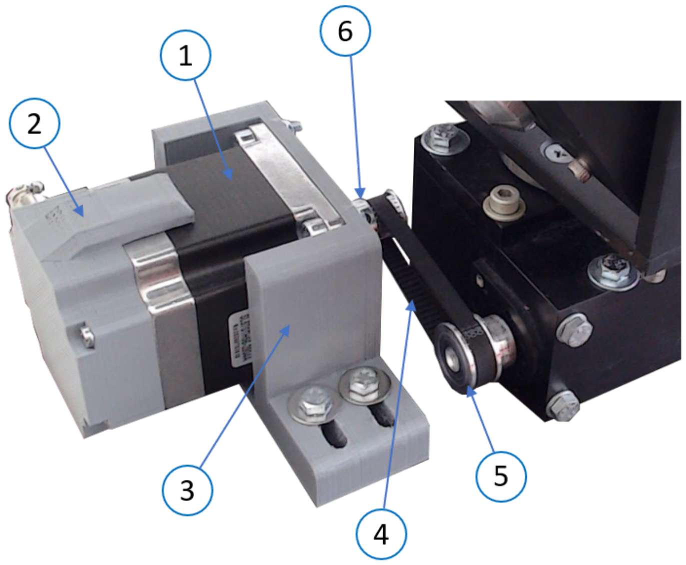


Figure 4.3: Motor drive of the azimuth axis: The same numbering was used as in Figure 4.2. In this setup, four adjustable screws hold the motor mount in place, to tension the belt. On the bottom of ground plate, clamps are used to secure the nuts, but are not illustrated.

Gear ratio

For both axes, azimuth and elevation, a pulley with 20 teeth ($z_{1,AZ}$ and $z_{1,EL}$) is mounted on the stepper motor shaft. The pulley mounted on the azimuth shaft with 30 teeth ($z_{2,AZ}$) and a pulley with 80 teeth ($z_{2,EL}$) on the elevation axis are connected to the pulley on the stepper motor shaft via timing belts. According to Equation 2.10, the gear ratio $i_{AZ} = \frac{z_{2,AZ}}{z_{1,AZ}}$ for the azimuth axis equals 1.5 and the gear ratio $i_{EL} = \frac{z_{2,EL}}{z_{1,EL}}$ for the elevation axis equals 4.

Necessary torque of the stepper motors

With loss-free transmission and a specified minimum torque on the output shaft of 2 Nm for the elevation axis and a gear ratio of 4, the torque of the motor must be at least 0.5 Nm. According to the torque curve of an equivalent stepper motor with a stepper driver of the same type and the same configuration, a minimum torque of at 0.8 Nm is provided by the motor in the lower speed range, where the motor is used. The required 0.5 Nm are thus fulfilled.

The required torque was not determined for the azimuth axis. By adjusting the axes by hand, significantly less effort was required. In the first tests with a much weaker NEMA 17 stepper motor, which has a smaller size factor compared to NEMA 23 and a direct coupling to the azimuth axis with a 3D-printed coupler as shown in Figure 3.7, the axis could be adjusted. To avoid overloading the motor, the same NEMA 23 stepper motor as on the elevation axis is used in combination with a gear ratio of 1.5.

Determination of the necessary number of steps

To change the angular position of the azimuth and elevation axes, the corresponding motors need an equivalent number of steps to move. According to the chosen stepper motors and drivers, one total rotation of the motor shaft is equivalent to 400 steps for both, azimuth and elevation.

When moving the shaft of the azimuth axis for one 360-degree rotation, the effective angular change of the antenna dish in azimuth direction is 5°. According to the gear ratio of 1.5, 600 steps are necessary for an effective change in azimuth of 5°. This results in 120 necessary steps of the stepper motor, to adjust the azimuth direction for 1°.

This result was verified with a test by moving the stepper motor by 1000 steps and measuring the average angular change with the 9-axis-sensor. As a result, the average change by moving 1000 steps was 8.385° or 119.26 steps per degree. This result is close to the expected value of 120 steps per degree.

To determine the number of steps necessary to move the elevation axis 1°, a digital protractor was used. Starting at an elevation angle of the rotary axis (EL') of 0°, elevation was changed by a defined number of steps and the corresponding angular position was determined with the protractor. In Table 4.4, actual values of the rotary axis measured with the protractor and the angular difference $\Delta EL'$ compared to the previous line are listed. From measurement line no. 1 to 6, the elevation axis was increased by 5000 steps and decreased from line no. 6 to 11. The determined average angular change of 11.26° between each line is equivalent to 444.05 steps per degree. During this test, the LNA was not mounted on the feed arm.

Table 4.4: Determining the number of necessary steps for a one-degree movement of the elevation axis. The actual elevation position of the rotary axis, measured with a digital protractor is shown in column EL'. The step size used in this measurement was 5000 steps. The total number of steps from starting position *step count* and the angular difference $\Delta EL'$ between each measurement are also shown.

no.	**step count**	EL'	$\Delta EL'$
1	0	0°	-
2	5000	11.13°	11.13°
3	10000	22.28°	11.15°
4	15000	33.72°	11.44°
5	20000	44.98°	11.26°
6	25000	56.30°	11.32°
7	20000	45.10°	11.2°
8	15000	33.84°	11.26°
9	10000	22.46°	11.38°
10	5000	11.34°	11.12°
11	0	0°	11.34°

4.3.2 9-axis-sensor (IMU)

Two different sensors were tested in the design phase, the MPU9250 and the BNO055. Both sensors combine an accelerometer, a gyroscope and a magnetometer in one package. In comparison to the MPU9250 which only delivers raw sensor values, the BNO055 uses a cortex M0+ microprocessor running a sensor fusion software on it. For this reason, the BNO055 not only provides the raw sensor values but also Euler angles and the actual calibration status of each sensor. Different sensor modes are available on the BNO055 to enable or disable the integrated sensors, depending on the application.

For the MPU9250 additional calculations for roll, pitch and heading were necessary, as mentioned in Section 2.1.7. To validate the delivered sensor values for roll and pitch, a triangle ruler was used to compare the measured values with the expected 45° angle. The results for roll and pitch were promising. The measured values only differed in the first decimal place, which is due to the positioning with the free hand. No usable result could be obtained for the yaw, since the calculated values fluctuated strongly despite different calibration attempts.

The BNO055 on the other hand delivers all necessary values without additional calculations. The calibration status of the sensors is updated by the internal microprocessor and can be accessed via the interface. This status indicates whether the transferred values are valid or not. To calibrate the sensors, no additional function calls are necessary. The sensor can be calibrated at any time by rotating the sensor.

To validate the delivered sensor values, the same method with the triangle ruler was used. In this case, the values for roll and pitch were in the same range as before. This time, the transmitted values for yaw were stable but there was a certain offset compared to the compass function of a smartphone. By repeating this test, the values from the smartphone and the BNO055 are changing significantly. One reason for this behavior is due to the test environment. The test was carried out at a desk in an office where other electrical devices could influence the measurements.
When mounted on the feed arm of the pico terminal, the measured values for yaw were changing even in an outdoor environment. In this case, other influences such as the magnetic fields of the motors have an impact on the measured values.

4.3.3 GPS receiver

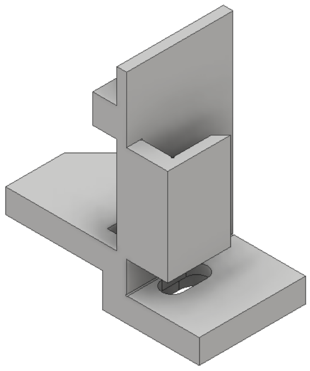
To determine the actual GPS position of the pico terminal, a Navilock NL-602U USB receiver is used. The receiver is connected via USB to the processing unit and provides actual GPS information by updating a certain file which can be accessed by other programs. A subroutine reads the NMEA sentences and provides the prepared data back to the main program.
Depending on the environment, it can take a few seconds - in certain cases up to several minutes - until valid values are received. During various function tests indoors and outdoors, it was found several times that no valid date values were transmitted during operation. In general, this is not a critical issue, since invalid GPS data will be ignored during a tracking process and the steppers will not move. To avoid this behavior, the GPS receiver should be disabled in the software during a tracking process. In this case, the configured position data and the system time are used instead of the data of the GPS receiver. More information can be found in the user guide in the appendix.

4.3.4 Limit switch

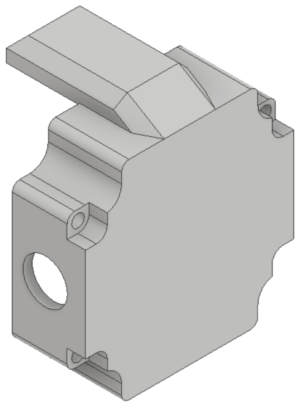
A limit switch in the "normally open" configuration, shown in Figure 3.9, is used for the elevation axis to prevent a collision in the lower position. In this case, the lower position can be left without pressing additional buttons. No limit switch is used for the azimuth axis in order to ensure more flexibility in the alignment. When configuring the azimuth reference position, it is important to ensure that the cables are not overstretched during operation.

4.3.5 Additional hardware parts

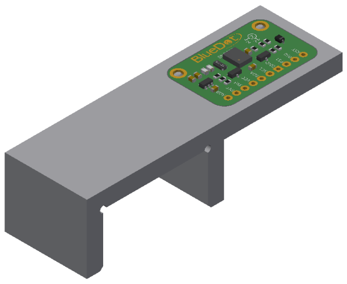
Apart from the motor mount, additional hardware parts are used on the pico terminal. In Figure 4.4 the models of the 3D printed parts are shown. All parts were designed with Autodesk Inventor and sliced with Cura. Autodesk Inventor is a CAD application for 3D mechanical design and Cura is an open-source slicing application to generate the machine code for 3D printers. Some of these printed parts are also shown in Figures 4.2 and 4.3.
The 3D printed end stop mount holds the limit switch of the elevation axis in place. The mount is attached to the transition part of the azimuth axis to the elevation axis with one of the existing screws. To connect the cables with the stepper motors, a 3D printed motor connector adapter with a hole for an aviation connector is used. One end of the cable with the corresponding counterpart is connected to the motor side, while the other end of the cable is connected to the micro step driver. Inside the adapter, the aviation connector is soldered to the four wires of the stepper motor. To hold the 9-axis-sensor in place, the 9-axis-sensor mount is used. With this mount, the sensor can be attached to the feed arm without any screws. Four feet are used to level the ground plate of the pico terminal. With two clamps, the motor mount of the stepper motor of the elevation axis is held in place.



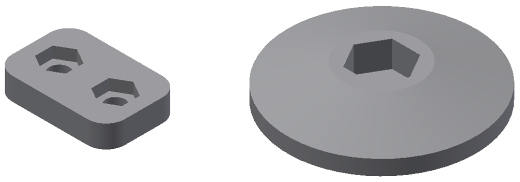
(a) End stop mount



(b) Motor connector adapter



(c) 9-axis-sensor mount

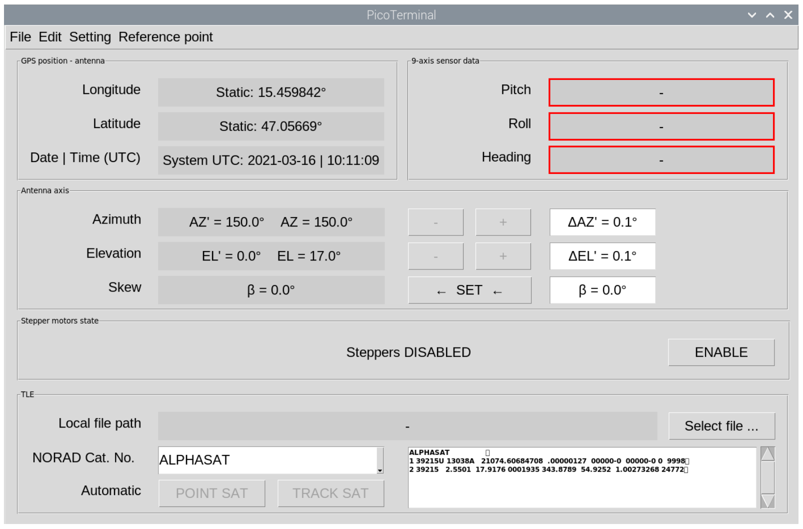


(d) Clamp and foot

Figure 4.4: 3D models of printed parts used on the pico terminal. With the end stop mount, the reference switch of the elevation axis is hold in place. On each stepper motor, a motor connector adapter is used to easily connect the stepper motor with the motor driver. A 9-axis-sensor mount which fits on the feed arm of the pico terminal is also shown. In total four feet and 2 clamps are used to adjust the ground palte and to hold the motor mount of the azimuth axis in place.

4.4 Software

The program to control the motor drives and to read the sensor values is split into several subroutines in different Python scripts. With a GUI, the operator is able change different settings and parameters. In Figure 4.5, the main screen is shown. The GUI is organized with group panels to separate the different hardware components from each other. In addition to current GPS and axis position data, measured values from the 9-axis sensor are visualized. Position adjustments of the azimuth and elevation axes as well as the loading of TLE data are possible with this screen. A user manual with more details to all GUI elements and the corresponding source code can be found in the appendix.



Longitude: Static: 15.459842°
Latitude: Static: 47.05669°
Date | Time (UTC): System UTC: 2021-03-16 | 10:11:09
Pitch: -
Roll: -
Heading: -
Azimuth: AZ' = 150.0° AZ = 150.0° ΔAZ' = 0.1°
Elevation: EL' = 0.0° EL = 17.0° ΔEL' = 0.1°
Skew: β = 0.0° β = 0.0°
Steppers DISABLED
Local file path: -
NORAD Cat. No.: ALPHASAT
Automatic

```
ALPHASAT
1 39215U 13038A   21074.60684708  .00000127  00000-0  00000-0 0  9998
2 39215   2.5501  17.9176 0001935 343.8789  54.9252  1.00273268 24772
```

Figure 4.5: Main screen of the GUI to control the antenna unit of the pico terminal. The screen is split into different sections to visualize the position and orientation, to control the stepper motors and to change settings via the menu bar. In this case, the 9-axis-sensor and the GPS receiver are disconnected.

4.5 Test results

4.5.1 Evaluated average calibration time of the 9-axis-sensor

A specific Python script was programmed to evaluate the average calibration time of the 9-axis-sensor. The flow chart of this script is shown in Figure 4.6. When starting the script, the *I2C* connection to the sensor is established and the system time is stored into a start time variable. After the initialization process, the calibration statuses of the gyroscope, the accelerometer, and the magnetometer are checked cyclically. At the same time the whole sensor has to be calibrated by rotating the device around all axes, as explained in the user manual. If one of the calibration statuses reaches the fully calibrated status with value '3', the system time is stored into the corresponding variable. Thus the calibration time of gyroscope, the accelerometer, and the magnetometer are determined individually. Once all sensors are fully calibrated during the measurement, the calibration is printed in the console window. In addition, the sensor statuses at the end of the measurement are printed for comparison reason.

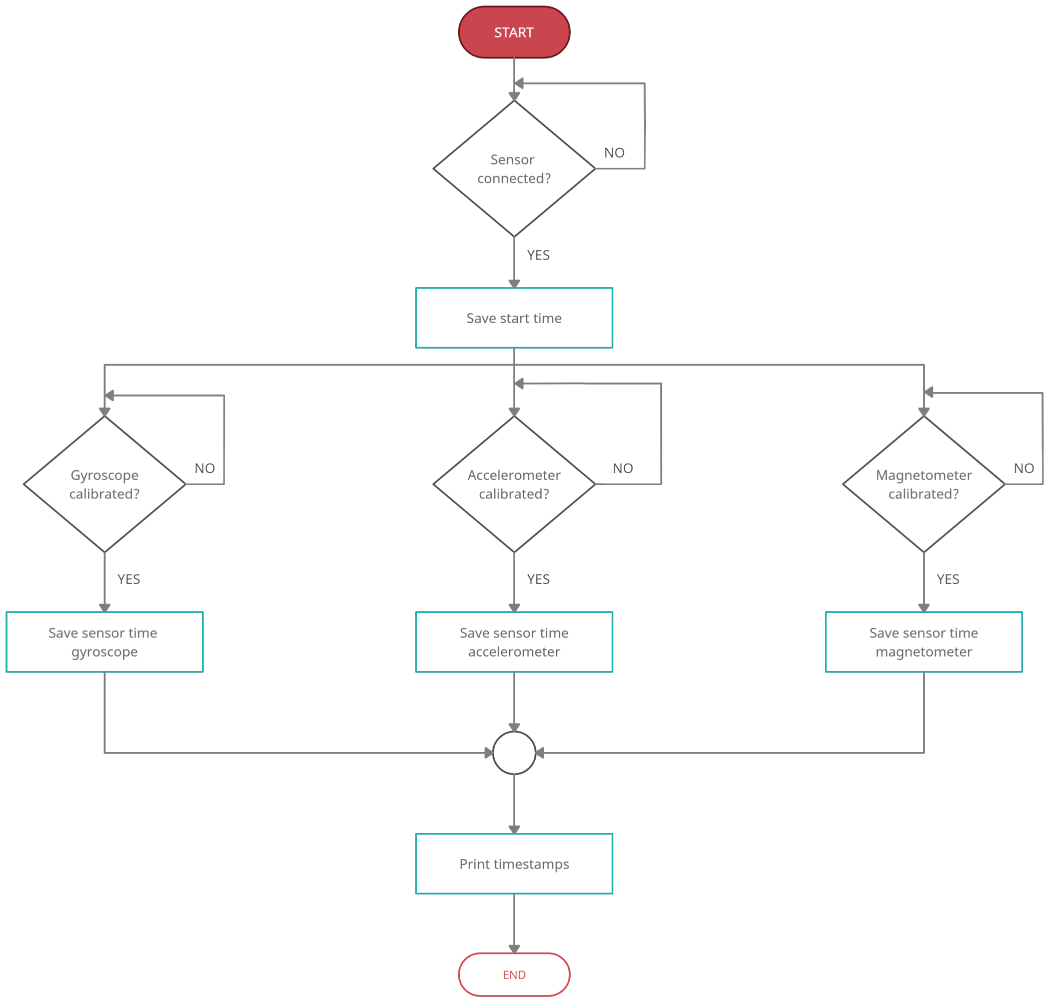


Figure 4.6: Flow chart of the Python script to determine the calibration time of the 9-axis-sensor. While loops are used to check the sensor connection and actual sensor statuses. The time difference between successful communication establishment and the fully calibrated status of each sensor is determined.

The determined time differences of the individual measurements are listed in Table 4.5. It is noticeable that the calibration time of the gyroscope is quite stable and significantly shorter compared to that of the accelerometer, and the magnetometer. To calibrate the gyroscope, the sensor needs to be in a ”single stable position for a period of few seconds” [41]. Due to the stable position of the sensor on the feed arm at the beginning of the measurement, a fast calibration of the gyroscope was assumed. The measurements confirm this assumption. The calibration of the accelerometer and the magnetometer need more effort. As described in the user manual, the necessary calibration steps have been carried out for both sensors. For the accelerometer ”6 stable positions for a period of a few seconds” and ”slow movement between 2 stable positions” are necessary. For the magnetometer ”some random movements” are sufficient. [41]

Table 4.5: Determined calibration times for gyroscope (GYRO), accelerometer (ACCEL) and magnetometer (MAG). For each sensor and the overall system (SYS), the calibration status at the end of the GPS receiver measurement is shown. The calibration status is an integer number between 0 and 3. A value of '3' indicates a fully calibrated sensor, '0' a not calibrated sensor.

no.	**Calibration time in sec**			**Calibration status**			
	GYRO	ACCEL	MAG	GYRO	ACCEL	MAG	SYS
1	0.82	14.65	11.32	3	3	3	2
2	0.82	19.06	20.17	3	3	3	0
3	0.81	18.06	19.85	3	3	3	2
4	0.82	16.75	18.39	3	3	3	3
5	0.82	14.54	4.30	3	3	3	0
6	0.83	33.30	29.4	3	3	3	3

The average calibration times for each sensor are shown in Table 4.6. A calibration time of 4.30 sec at measurement no. 5 for the magnetometer seems a little bit too fast, compared to the average value of 17.24 seconds. When looking on the system status SYS, the value was 0 at the end of the measurment. According to [42], the north pole was not found, if the system calibration status is 0.

Table 4.6: Average calibration times of the 9-axis-sensor. The results are determined independently for the gyroscope, accelerometer and magnetometer. The comparison shows a significantly shorter calibration time for the gyroscope.

	Gyroscope	**Accelerometer**	**Magnetometer**
Average time	0.82 sec	19.39 sec	17.24 sec

During different tests with the pico terminal, it could be observed that the calibration status can change - after successful calibration. As a consequence, the magnetometer delivers incorrect values. The determination of the magnetic north pole using a sensor generally turns out to be difficult, since environmental influences distort the measurement. The comparatively low field strength of the Earth's magnetic field can therefore only be determined by means of compensation calculations. Modern smartphones also have sensors for determining the location or position. Such a smartphone was used to find reference values that can be compared with the measured values of the 9-axis-sensor. It turned out that the values of the smartphone also fluctuated strongly and were therefore of only limited use.

4.5.2 Determination of the average time until valid GPS data is received

Another Python script was created to evaluate the average time it takes until the GPS data is valid. In Figure 4.7 the flow chart and in Listings 4.1 and 4.2 code snippets of this script are shown. After the initialize section of the script, it is checked whether a data file with the name *ttyACM** has been created in the */dev* directory of the RaspberryPi. If a file is located in the directory, the system time is stored into a start time variable and all lines of the data file are read cyclically. Each line of this file represents a NMEA sentence with its identifier at the beginning of the line. Data in the NMEA sentence that begins with *GPGLL* applies to latitude and longitude. If the character "A" is in the data status field of such a sentence, the values are valid. In this case, the system time is stored into an end time variable and the time difference between start and end time variables is the evaluated test time of each measurement.

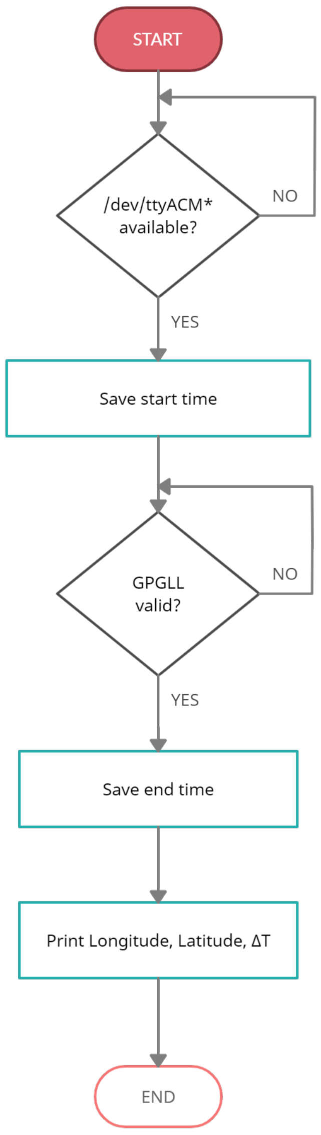


Figure 4.7: Flow chart of the Python script to determine the average time until valid GPS data is received. *While loops* are used to check if the receiver is connected to the control unit and if the *$GPGLL* sentence is valid. The time difference ΔT between successful communication establishment and valid GPS data is determined.

Listing 4.1: Code snippet to determine the file path of the GPS data file. If there is a file with a specific name convention, the loop exits.

```
gps_path = ''
while gps_path == '':
        for file_paths in glob.glob(self.path):
                gps_path = file_paths
```

Listing 4.2: Code snippet to extract longitude and latitude values from the *NMEA* sentences. A status variable is set if the data in the NMEA sentence is valid.

```
if data.split(',')[0] == "$GPGLL":
        lat = data.split(',')[1]
        lat_dir = data.split(',')[2]
        lon = data.split(',')[3]
        lon_dir = data.split(',')[4]
        status = data.split(',')[6]

        lat_deg = int(float(lat) // 100)
        lat_min_tmp = float(lat) % 100
        lat_min = int(lat_min_tmp // 1)
        lat_sec = round(((lat_min_tmp % 1) * 60), 1)

        lon_deg = int(float(lon) // 100)
        lon_min_tmp = float(lon) % 100
        lon_min = int(lon_min_tmp // 1)
        lon_sec = round(((lon_min_tmp % 1) * 60), 1)

        self.gps_latitude = str(lat_deg)+u'\xb0'+"␣"+str(lat_min)+"
           ↪ '␣"+str(lat_sec)+"''␣"+lat_dir
        self.gps_longitude = str(lon_deg)+u'\xb0'+ "␣"+str(lon_min)
           ↪ + "'␣"+ str(lon_sec)+"''␣"+lon_dir
        self.gps_longitude_degree=round(lon_deg + (lon_min_tmp /
           ↪ 60), 6)
        self.gps_latitude_degree=round(lat_deg + (lat_min_tmp / 60)
           ↪ , 6)

        if status == 'A':
                self.gps_valid = True
        elif status == 'V':
                self.gps_valid = False
        else:
                self.gps_valid = None
```

In Table 4.7 the evaluated test time and the corresponding values for longitude and latitude are shown. The measurements were taken at two different locations outdoors with almost the same result. Location 1 represents the *Institute of Communication Networks and Satellite Communications* of TUG. Location 2 represents the private home of the author and thus will not be disclosed for data protection reasons.

Table 4.7: Determined time difference ΔT between successful communication establishment and valid GPS data at different locations. Values for longitude and latitude are shown in degrees (°), minutes (') and seconds (").

no.	Location	ΔT	Longitude	Latitude
1	1	0.186642 sec	15° 27' 34.3" E	47° 3' 31.9" N
2	1	0.862223 sec	15° 27' 34.3" E	47° 3' 32.0" N
3	1	0.556774 sec	15° 27' 34.4" E	47° 3' 32.0" N
4	1	3.561892 sec	15° 27' 34.4" E	47° 3' 31.9" N
5	1	0.768409 sec	15° 27' 34.6" E	47° 3' 31.9" N
6	2	0.231791 sec	15° E	47° N
7	2	4.438948 sec	15° E	47° N
8	2	0.990122 sec	15° E	47° N
9	2	0.552716 sec	15° E	47° N
10	2	0.680697 sec	15° E	47° N

In Listing 4.3 the log file of measurement no. 5 of Table 4.7 is shown. Valid position data are printed in degrees, minutes and seconds.

Listing 4.3: Output of the console window at the end of a measurement. Actual values for longitude and latitude in degrees, minutes and seconds are shown. Start time, end time and the time difference ΔT are determined and visualized in the output of the console.

```
============ RESTART: /home/pi/picoTerminal/GPS_validDataTest.py
    ============
Time started
None 13:46:22 LON: 15    27' 34.6'' E  LAT: 47    3' 31.9'' N
2021-02-23 13:46:20.193259+00:00
2021-02-23 13:46:20.961668+00:00
0:00:00.768409
>>>
```

In Table 4.8, the values for longitude and latitude of the GPS receiver $P_{receiver}$ converted into the decimal degree format are compared to evaluated position data from online maps P_{online}. The difference in longitude ΔLON and the difference in latitude ΔLAT between $P_{receiver}$ and P_{online} are listed.

Table 4.8: Deviation of the determined GPS position via online maps P_{online} and the determined GPS position via the GPS receiver $P_{receiver}$. For each measurement at both locations the deviations ΔLON for longitude and ΔLAT for latitude between P_{online} and $P_{receiver}$ are calculated.

		Longitude		**Latitude**	
no.	**Location**	decimal degrees	ΔLON	decimal degrees	ΔLAT
1	1	15.459528°	0.000121°	47.058861°	0.000073°
2	1	15.459528°	0.000121°	47.058889°	0.000045°
3	1	15.459556°	0.000093°	47.058889°	0.000045°
4	1	15.459556°	0.000093°	47.058861°	0.000073°
5	1	15.459611°	0.000038	47.058861°	0.000073°
6	2	15	0.000107°	47	0.000088°
7	2	15	0.000004°	47	0.000033°
8	2	15	0.000135°	47	0.000033°
9	2	15	0.000079°	47	0.000033°
10	2	15	0.000032°	47	0.000033°

It can be seen that the deviation values ΔLON and ΔLAT are close to zero and in an acceptable range of less than 10 meters, as shown in Figure 4.8.

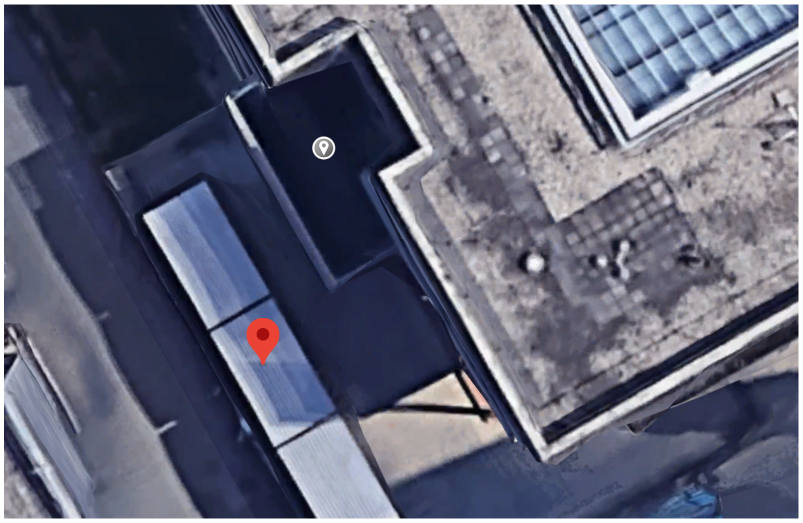


Figure 4.8: Test location 1 of the pico terminal during the measurements. The grey pin represents the actual position (P_{online}), where the pico terminal was used. The red pin visualizes the measured position of the GPS receiver ($P_{receiver}$). The distance between these two points is approximately 7 meters, measured with available online maps [43],[44].

The average time it takes to receive valid GPS data is shown in Table 4.9. With average times under 1.5 seconds GPS position data is immediately available. It is important to note, that these measurements were taken outdoors. In an indoor environment the times would be significantly higher, if the position information can be determined at all, due to blockage of the GPS signals. While position data is available right after plugging the USB cable into the USB port of the processing unit, it could be observed, that date information took much longer to be up to date. In contrast to this, the time information is available as quickly as the position data but was not recorded by measurement.

Table 4.9: Determined average times to valid GPS data. The average time of all measurements per location and the average time of all measurements (location 1 and location 2) are shown.

	Location 1	**Location 2**	**Overall**
Average time	1.19 sec	1.38 sec	1.28 sec

4.5.3 Evaluated trueness, precision and accuracy of the elevation axis

To check the pointing quality of the motorized antenna of the pico terminal, pointing tests were carried out and trueness, precision and accuracy of the elevation angle were evaluated. For this purpose, the elevation angle was increased by a certain reference value, beginning with a defined starting position. Both, the angles of the starting position EL_0 and the end position EL_x were measured with a digital protractor. The determined values are listed in Tables 4.10 and 4.11. To achieve a uniform starting position, the axis was moved to the lower end position. When the limit switch was actuated - this process is recognized by the control software - the motor was stopped.

Two different reference values were used for the tests. These reference values are the targeted angles to which the elevation is supposed to raise.

In Table 4.10 the reference value x_{ref} with a desired angle difference of 20° and in Table 4.11 with a desired angle difference of 45° were used. In both cases the evaluated angle differences x_i after the adjustments were calculated by using Equation 4.1.

$$x_i = EL_x - EL_0 \tag{4.1}$$

Table 4.10: Determined values during the elevation pointing test. The column EL_0 denotes the measured elevation angle at the starting position before the adjustment. The measured end position, EL_x, and the calculated angular difference x_i of each measurement are also listed. In this test setup, the targeted angular difference x_{ref} was always 20°.

no. (i)	EL_0	EL_x	x_i	x_{ref}
1	-5.92°	13.83°	19.75°	
2	-5.94°	13.74°	19.68°	
3	-5.94°	13.68°	19.62°	
4	-5.96°	13.62°	19.58°	
5	-5.97°	13.65°	19.62°	20°
6	-5.87°	13.62°	19.49°	
7	-6.03°	13.60°	19.63°	
8	-6.08°	13.54°	19.62°	
9	-6.03°	13.54°	19.57°	

Table 4.11: Determined values during the elevation pointing test. The column EL_0 denotes the measured elevation angle at the starting position before the adjustment. The measured end position, EL_x, and the calculated angular difference x_i of each measurement are also listed. In this test setup, the targeted angular difference x_{ref} was always 45°.

no. (i)	EL_0	EL_x	x_i	x_{ref}
1	-6.03°	38.95°	44.98°	45°
2	-6.03°	38.87°	44.90°	
3	-6.04°	38.99°	45.03°	
4	-6.04°	38.87°	44.91°	
5	-6.03°	38.97°	45.00°	
6	-6.08°	38.87°	44.95°	
7	-6.04°	**35.97**°	42.01°	
8	-6.04°	**38.35**°	44.39°	

For both measurements the mean value $\bar{x}$ of the calculated angular differences x_i was calculated using Equation 2.6. By replacing μ with x_{ref}, the trueness of both measurements were determined using Equation 2.7. By dividing the determined trueness with the reference value x_{ref} and multiplying with 100%(see Equation 4.2), the percentage error $\%error$ was calculated.

$$\%error = \frac{trueness}{x_{ref}} \cdot 100\% \tag{4.2}$$

The precision of both measurements was calculated using Equation 2.8. To determine the accuracy, Equation 2.9 was applied to every measurement line in Tables 4.10 and 4.11. In Table 4.12, the results of both measurements can be compared.

For the second measurement with a reference value of 45°, two result lines can be seen. When looking at measurement lines 7 and 8 of Table 4.11, a relatively high deviation of x_i compared to the other measurement lines is noticeable. At line 7 some files were opened in the background of the control unit during the measurement. As a result, the motor control sequence was affected by other processes and the motor could not rotate continuously. Instead, jerky movements were made. As a result, the losses when starting from the resting position came into play several times and falsified the measurement. At measurement line 8 the motor stopped for a short period of time, slightly after releasing the limit switch. Via the console output it could be determined that a new end position impulse was registered after leaving the end position. Since the switch could not be operated mechanically, this behavior is plausible with regard to a possible voltage induction by the motor currents. At the time of the test, the limit switch was connected to the control unit with two non-twisted wires.

Table 4.12: Evaluated values for trueness, precision and accuracy for both reference values. In column *trueness*, the difference between the average measured values from column EL_x and the desired reference value x_{ref} is shown. The $\%error$ is shown in the same column. The evaluated standard deviation is used to describe the spread of the measured results and is shown in column *precision*. The difference between the evaluated angular difference x_i and the desired value x_{ref} in relation to x_i is the accuracy of the individual measurement. In column *Accuracy range* the minimum and maximum deviation in percent of all measurements are shown. The results in the marked line (*) do not consider measurement lines 7 and 8 of Table 4.11.

Reference value	Trueness		Precision	Accuracy range
20°	0.38°	1.91%	0.41°	1.3 - 2.6%
45°	0.48°	1.06%	1.15°	0 - 7.1%
45°(*)	0.04°	0.09%	0.07°	0 - 0.2%

4.5.4 Evaluated angle error during the automatic alignment

To evaluate the angle error when pointing a geostationary satellite with the automatic pointing function compared to manual alignment, in this case *Alphasat*, a *Rohde Schwarz FSP* spectrum analyzer with a frequency range between 9 kHz and 40 GHz was used to measure the 19.701 GHz beacon signal of the satellite. Due to the frequency range of the measurement equipment, the signal can be measured without a down converter.

To use the automatic axes adjustment, the corresponding TLE data was downloaded and selected in the combo box of the GUI. This procedure is explained in the user guide in the appendix. By clicking on the "start pointing" button, the necessary steps are calculated and sent to the micro step drivers to move the stepper motors and to adjust both axes, azimuth and elevation, in the desired direction. To achieve a meaningful measurement, the spectrum analyzer had to be configured. Starting with a higher RBW and a larger span, the signal was located and then shifted to the center by adjusting the center frequency. To lower the noise level, the RBW was decreased. A smaller span and a lower VBW were configured to achieve a higher update rate and a smoother signal.

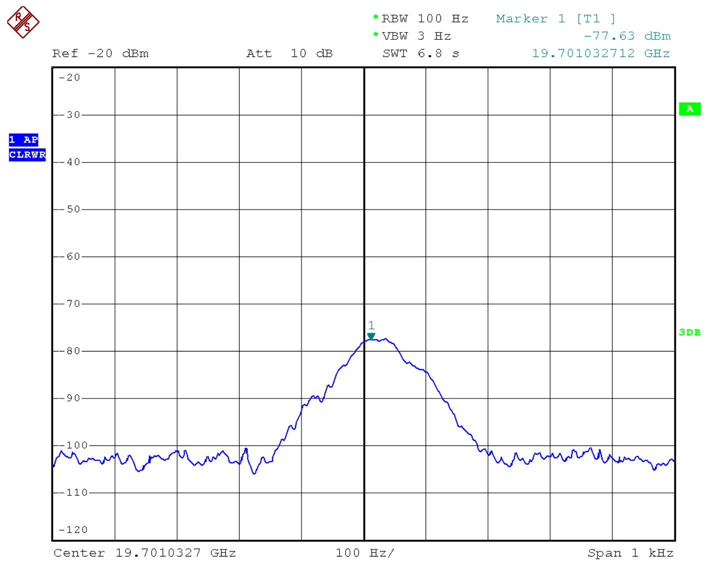
To evaluate the angle error of the measured signal level after the automatic alignment compared to the highest possible signal level, the measured signal level of both axes must be evaluated at different azimuth and elevation angles. In other words, actual values for azimuth and elevation which are displayed on the GUI right after the automatic alignment are noted and the measured signal spectrum is exported as a screenshot. In the following steps, one of the two motorized axes is adjusted in 1° steps with the manual control panel in positive and negative direction to evaluate the highest signal level. The angle error to be determined in one axis direction results from the difference between the GUI value after the automatic alignment and the GUI value of the measurement with the highest signal power level.

While one of the two axes was adjusted, the other remained in a fixed position. For both measurements, elevation and azimuth, the pointing function was started once at the beginning to update the position. The end position visualized on the GUI and the corresponding plot of the spectrum with a marker was set to peak right after pointing are listed as measurement no. 1 in Tables 4.13 and 4.14 and visualized in the following subsections.

Elevation axis

In Figure 4.9 the spectrum of the measured signal with a power level of -77.63 dBm right after the automatic pointing is visualized. By adjusting the elevation axis in 1° steps in positive and negative direction, different power levels were measured. In Measurement no. 2 and 3 of Table 4.13 the results in positive direction and in measurement no. 4 and 5 the results in negative direction are shown. Measurement no. 1 corresponds to the measured signal level after the automatic alignment.

It can be seen that the measured signal level right after the automatic pointing was the highest. It is important to notice, that adjusting the elevation influences the effective orientation of the azimuth axis as well. During the whole measurement, no movements of the AZ' axis were made, but the effective azimuth direction was changing. This change of the effective azimuth direction can be seen in column AZ in Table 4.13 and is due to the antenna geometry.



Date: 23.FEB.2021 07:52:25

Figure 4.9: Measured spectrum right after aligning the antenna dish with the automatic pointing function. The power of the Alphasat beacon signal is shown. With a marker set to peak, the highest power level was visualized in the top right corner. The spectrum shown in this figure relates to measurement no. 1 in Table 4.13.

Table 4.13: Determined signal power level (P_{dBm}) during elevation axis alignment (EL') in 1° steps. Due to geometry reasons, the effective azimuth direction (AZ) is changing without moving the azimuth axis (AZ') connected to the drive. Another effect related to the geometry is shown in column EL. The effective elevation (EL) is changing less than 1°. During the measurement, *skew* was configured to 90°.

no.	AZ′	AZ	EL′	EL	skew	$\mathbf{P_{dBm}}$
1		167.2°	35.48°	33.7°		-77.63 dBm
2		167.5°	36.48°	34.7°		-78.54 dBm
3	146.66°	167.7°	37.48°	35.6°	90°	-81.92 dBm
4		167°	34.48°	32.8°		-79.34 dBm
5		166.8°	33.48°	31.8°		-85.66 dBm

Azimuth axis

Using the same procedure as described for the elevation axis, the power levels were determined by adjusting the azimuth axis. In this case, a change in azimuth direction has no effect on the effective elevation direction. The highest power level was measured at measurement no. 6 with a power level of -74.43 dBm and can be seen in Table 4.14. In Figure 4.10 the power level right after the pointing procedure which corresponds to measurement no. 1 and the highest evaluated power level in measurement no. 6 were compared. The angular difference between both measurements was +2° and results in a power level difference of 4.36 dB. By plotting all measured power levels at their corresponding angles, the behavior of the signal at different azimuth angels can be visualized. In Figure 4.11 the blue line represents the measured power levels and the orange line represents the -3 dB offset from the highest measured power level. The angular difference between the two intersection points, one at 167.6° and the other at 171.3° of the blue and the orange line is the evaluated half power beamwidth. The measured $HPWB_{meas}$ results in 3.7°. The calculated half power beamwidth $HPBW_{calc}$ can be evaluated using Equation 2.3. With an effective dish diameter of 34 cm and an average signal frequency of 19.701 GHz, $HPBW_{calc}$ results in 3.14°. Due to frequency shifts in the measured signal and the tropospheric effects in the form of clouds during the measurement, a difference between the evaluated and calculated HPBW of 0.56° appears to be tolerable.

The noted azimuth value right after the automatic alignment, marked as a red point in Figure 4.10, is slightly outside the evaluated $HPWB_{meas}$. One possible reason for this behavior might be the initial setup of the pico terminal. The reference position of the azimuth axis was set by comparing a well-known direction of a building wall. An offset of one or two degree might be possible. Comparing the azimuth direction with a compass was not possible. When measuring the true north direction with the 9-axis-sensor or a smart phone, the evaluated values always changed by several degrees. In addition, the measured values of the 9-axis-sensor and the smart phone compass differed considerably. This difference was a multiple of the expected magnetic declination, which is the angle between magnetic north and true north, of approximately 4° in Graz.

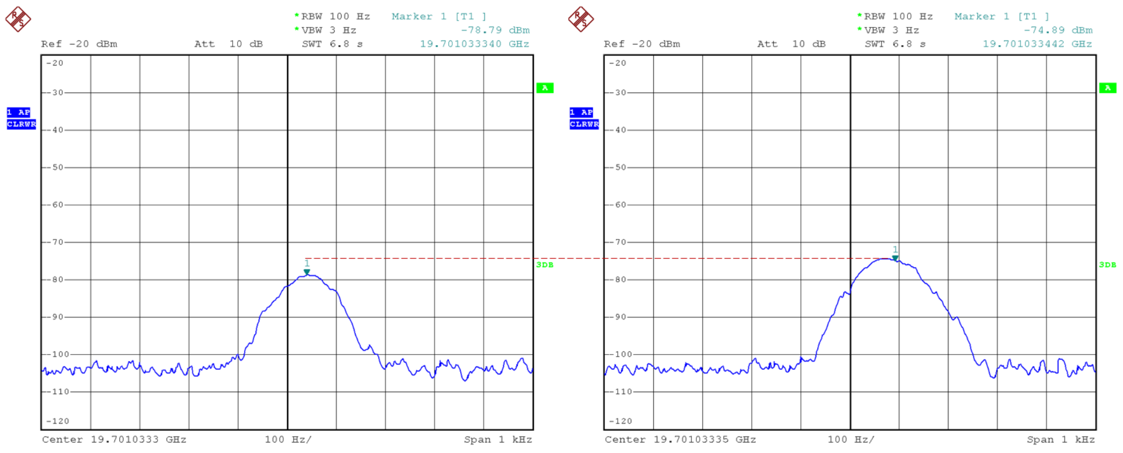


Figure 4.10: Measured spectrum right after aligning the antenna dish with the automatic pointing function, similar to Figure 4.13. This time the evaluated power level right after pointing (shown in the left spectrum) was lower than the evaluated power level a few measurements later (shown in the right spectrum). The power level difference of 4.36 dB is visualized by the red dashed line.

Table 4.14: Determined signal power levels P_{dBm} during azimuth axis alignment (AZ') in 1° steps, similar to Table 4.13. This time, the effective azimuth direction (AZ) changes with the same value as AZ'. Both elevation angles (EL' and EL) are in a fixed relative position. The configured *skew* was 90°.

no.	AZ′	AZ	EL′	EL	skew	P_{dBm}
1	150.2°	167.2°				-78.79 dBm
2	149.2°	166.2°				-83.48 dBm
3	148.2°	165.2°				-92.16 dBm
4	147.2°	164.2°				-98.90 dBm
5	151.2°	168.2°				-75.80 dBm
6	152.2°	169.2°	35.67°	33.9°	90°	-74.43 dBm
7	153.2°	170.2°				-75.03 dBm
8	154.2°	171.2°				-76.89 dBm
9	155.2°	172.2°				-82.37 dBm
10	156.2°	173.2°				-87.94 dBm
11	157.2°	174.2°				-97.04 dBm

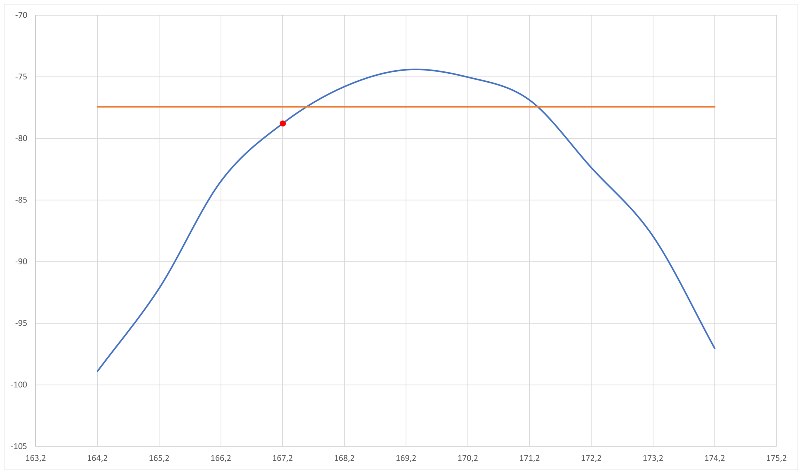
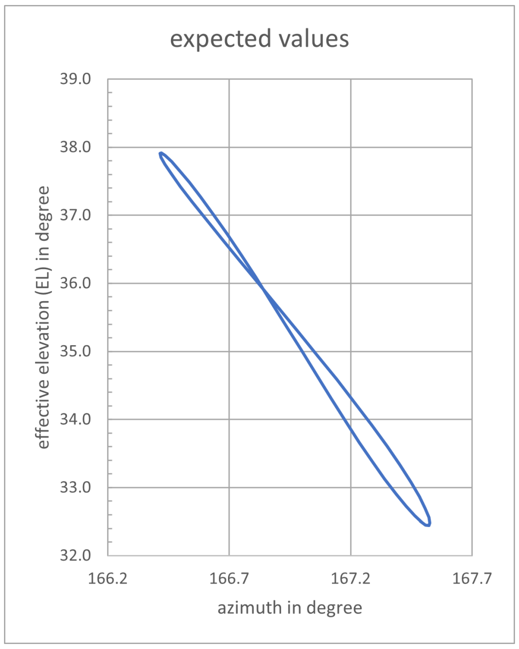


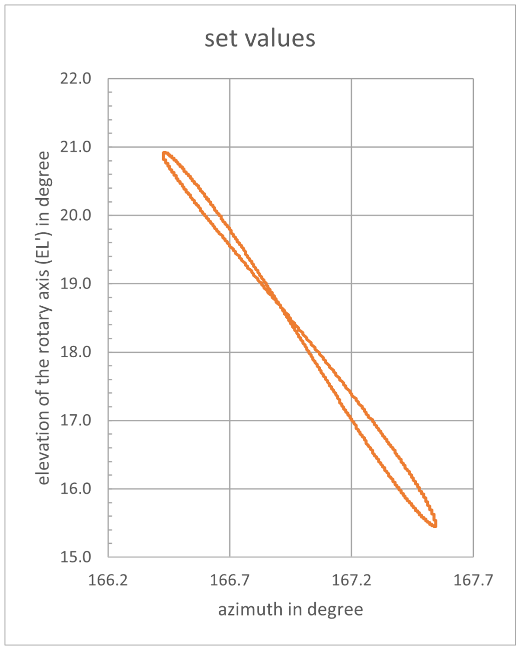
Figure 4.11: Evaluated half power beamwidth during azimuth axis alignment is shown. The blue line represents the interpolated power levels of the beacon signal to the corresponding azimuth position (AZ). The orange line represents a value of -3-dB from the highest measured power level. The angular difference of the two intersection points of both lines represents the HPBW of the antenna. Right after the automatic alignment, a power level of -78.79 dBm at 167.2° was determined, shown as a red dot which would be slightly outside the HPBW.

4.5.5 Verification of the automatic tracking function

In Figure 4.12a the expected daily movement of *Alphasat*, as seen from Graz on March 9th, 2021 is illustrated. It is a representation of the elevation angle in comparison to the azimuth angle and can be used to determine the satellite's relative position as seen from the ground station. Starting at the intersection point at midnight, elevation is decreasing and azimuth increasing until the curve reaches the bottom right position at 6:00 am. During the next half day, elevation is increasing and azimuth decreasing. At 6:00 pm at the top left position, directions are changing again. This behavior is repeated every day with slightly changed values for azimuth and elevation.



(a) expected values for azimuth and effective elevation



(b) set values for azimuth and elevation of the rotary axis

Figure 4.12: Daily movement of Alphasat for a period of one day is visualized. In (a) the expected azimuth and the effective elevation angles (EL) are shown. In comparison, azimuth and elevation of the rotary axis (EL') during tracking with an update interval of 2 minutes and a configured skew angle of 0° are shown. The angular difference in elevation is due to the 17° offset of the antenna. In both plots, the angular difference between maximum and minimum value for azimuth is 1.11° and for elevation 5.48°.

In a first test, the automatic tracking function without RF-equipment was used, to compare the actual adjustments of the axes with the expected adjustments according to the daily movement of the satellite. The update interval was set to 2 minutes for calculating new alignment data and to adjust the axis. The calculated alignment data, which are the setpoints for azimuth and elevation, are shown in Figure 4.12b. Each alignment change was logged using actual orientation data of the 9-axis-sensor and storing the data into a simple database running on the control unit in the background.

In Figure 4.13 the logged alignment changes of the azimuth and elevation axis, measured with the 9-axis-sensor, are shown for the same observation period of one day. It can be clearly seen that this shape does not match the expected adjustment shown in Figure 4.12. The determined changes in the azimuth direction of more than 20° clearly exceed the expected value. To get a more meaningful result, the measured elevation angles are plotted in comparison to the calculated azimuth angles, as visualized in Figure 4.14. At first glance, the determined shape also seems to deviate significantly. To rule out measurement errors in the data recording, the test was repeated with additional elevation angle values, measured with a digital protractor. Instead of a two-minute update interval, this time a two-hour interval was used. With a modified script, the test time was simulated to achieve the measurement of a whole day in only a few minutes. After a minute in real time, the timestamp for calculating new position data was increased by the two-hour update interval. Every change of the elevation axis was measured with the digital protractor for later comparison. The result of this measurement is shown in Figure 4.15.

The blue dashed shape visualizes the measured 9-axis-sensor values for the elevation axis in comparison to the calculated vales for azimuth. Compared to the previous discussed movement in Figure 4.14, a certain similarity can be seen. The values measured with the protractor are visualized in orange. The expected movement, shown in Figure 4.12 can be clearly seen in Figure 4.15. In conclusion, this means that the values determined with the 9-axis-sensor do not match the actual elevation angle at the time of measurement. The analyses showed that the BNO055 IMU, installed on the feed arm of the antenna, is not suitable to provide reliable data to verify the antenna pointing.

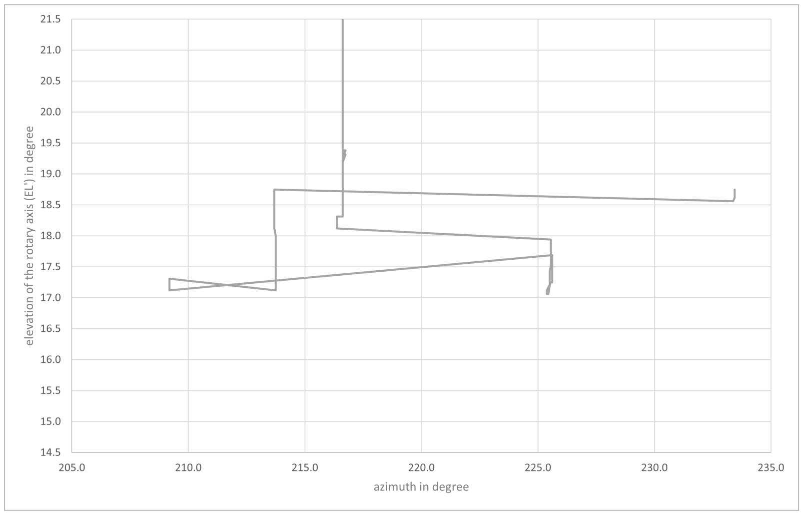


Figure 4.13: Measured values for azimuth and elevation with the 9-axis-sensor during tracking over a period of one day. The update interval was set to 2 minutes. The angular difference in azimuth differs significantly from the expected difference shown in Figure 4.12. The difference in altitude seems plausible, but it is difficult to compare.

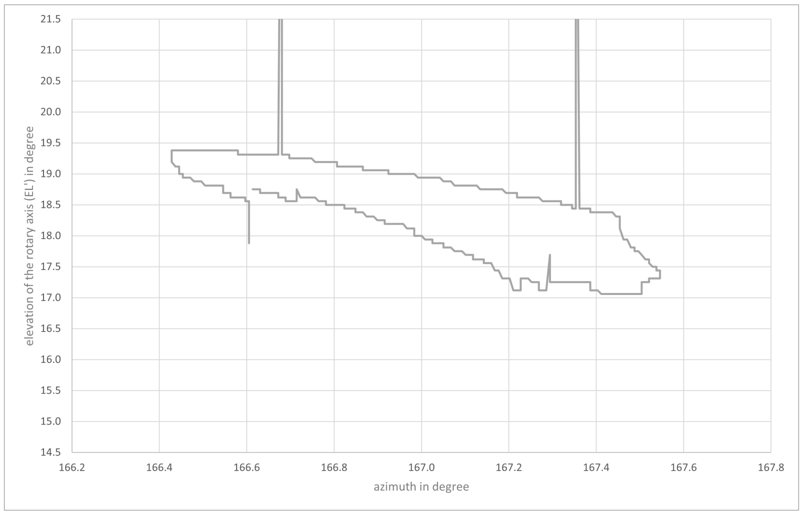


Figure 4.14: Measured values for elevation with the 9-axis-sensor in relation to calculated values for azimuth. In this case, the obviously incorrectly determined azimuth values are replaced by the calculated values, used in Figure 4.12. This eliminated the error in the azimuth direction and made it possible to compare the elevation values. Apart from a few outliers, the movement of the elevation axis can be clearly seen. In a direct comparison with the movement in Figure 4.12, it is noticeable that there is no intersection.

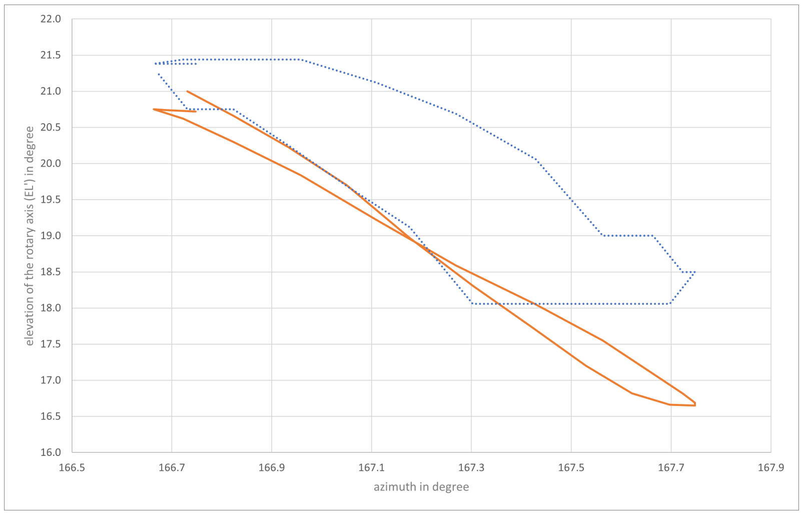


Figure 4.15: To compare measured values for elevation with the 9-axis-sensor (blue dashed line) with reference values, a simulation script with an update interval of 1 hour was used. With this script it was possible to simulate a time difference of 1 hour in under a minute in real time. This made it possible to simulate the daily movement in 24 minutes and to record the currently set elevation angle with a digital protractor after each update interval. The measured values for elevation with digital protractor (orange line) and the measured values with the 9-axis-sensor are visualized in relation to calculated values for azimuth. The evaluated values of the digital protractor match with the expected movement in Figure 4.12, whereas the evaluated values with the 9-axis-sensor look similar to thus of Figure 4.14.

4.5.6 Evaluation of the maximum signal power level drop

In this test, the maximum signal power level drop of the Alphasat 19.701 GHz beacon signal during half a day is evaluated. Once the antenna dish is aligned to point to the satellite, it remains in this fixed configuration. Azimuth and elevation are not changed during the whole measurement. Due to the daily movement of Alphasat, a certain misalignment appears. Depending on the angular difference between the configured azimuth and elevation angle to the required angles, the received signal level is changing.

For this test, the same *Rohde Schwarz FSP* spectrum analyzer as in one of the previous tests was used for measuring the received beacon signal in combination with the Spectrum Analyzer Automation Tool (SAAT) running on a PC. SAAT is a software tool which communicates with a defined number of spectrum analyzers. This tool allows changing the configuration of the analyzers and storing the measured spectrum in data files.

During this measurement running half a day, the measured spectrum is saved into the mentioned file. The file consists of several lines. Each line corresponds to a certain timestamp t and includes the signal power at different frequencies, separated by columns. With this information, the spectrum for all timestamps can be reproduced.

Before the measurement can be started, the pico terminal need to be aligned and the SAAT configured with a one-minute time interval. When the actual spectrum is visualized in the SAAT, the tool is ready and the measurement can be started. The pico terminal was aligned once in the morning at an elevation angle of the rotary axis (EL'_{fixed}) of 35.2°. The effective elevation angle (EL_{fixed}) at a skew angle of 90° is lower than the measured elevation angle (EL'_{fixed}). This effect is shown in Figure 4.16. According to this model a configured elevation angle EL'_{fixed} of 35.2° corresponds to an effective elevation angle EL_{fixed} of 33.45°.

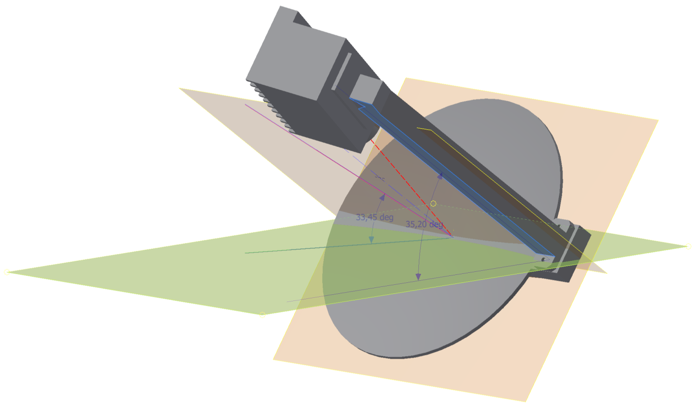


Figure 4.16: Comparison of the effective elevation angles EL (33.45°) and EL' (35.2°) in a 3D model. Due to geometry reasons, the effective elevation angle of the beam (red solid line) differs from the configured elevation angle of the rotary axis (EL'). Both angles are measured from the green plane, parallel to the horizon.

In Figure 4.17, the maximum signal power of each line of the file is plotted over time. As expected, the power level is changing, depending on the misalignment. The measurement started at 8:00 am with a desired effective elevation angle EL_{TLE} of 32.7°, determined via TLE data. Considering the antenna offset, EL'_{TLE} is the desired elevation angle of the rotary axis. Compared to the configured effective elevation angle EL_{fixed} of 33.45°, an alignment error of 0.75° occurred. The measured signal power at this timestamp was approximately -81 dBm.

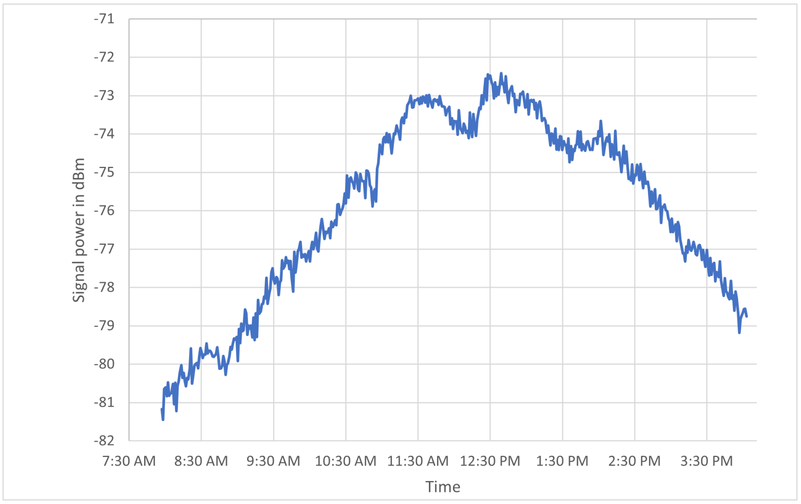


Figure 4.17: Measured signal power of Alphasat's beacon signal with fixed antenna alignment over a time interval of half a day. The spectrum of the received beacon signal was logged in a one-minute time interval and saved into a file. The maximum power level of each line in the file, representing the spectrum at a certain time stamp, is visualized. A change in power of roughly 9 dB during the measurement is noticeable.

The angular difference ΔEL between EL_{TLE} and EL_{fixed} and the angular difference $\Delta EL'$ between EL'_{TLE} and EL'_{fixed} are shown in Figure 4.18. According to the smallest alignment error it was assumed, that the differences are 0 when the signal power is at its maximum. Comparing Figure 4.17 with the highest power level in the range of 12:30 PM and Figure 4.18 with $\Delta EL \approx 0$ at 9:45 AM, this assumption is not valid. One possibility for this behavior might be a skew angle offset. A skew angle of 87° instead of 90° increases the effective elevation EL_{fixed} by nearly 1°. Recalculating ΔEL results in a new timestamp at 11:20 AM. Another possibility is an incorrect offset angle. With an offset angle in the range of 0°, the measured signal seems reasonable. Recalculating ΔEL result in a new timestamp in the range of 12:30 PM. Another aspect relates to the geometry of the antenna. At a skew angle of 90°, the weight of the LNA bends the feed arm. When the feed arm is bent down, the angle between the axis of rotation and the axis of the LNA feed increases. This leads to an increase of the effective elevation angle.
The second effect shown in Figure 4.18 concerns the offset increase of ΔEL and $\Delta EL'$ at higher elevation angles. Assuming a configured elevation angle of the rotary axis (EL'), shown in Figure 4.16, of 0°, the effective elevation angle (EL) is also 0°. When increasing EL', also EL increases, but in smaller steps. With a theoretically configured elevation angle of the rotary axis of 90°, the effective elevation angle is only 73°.

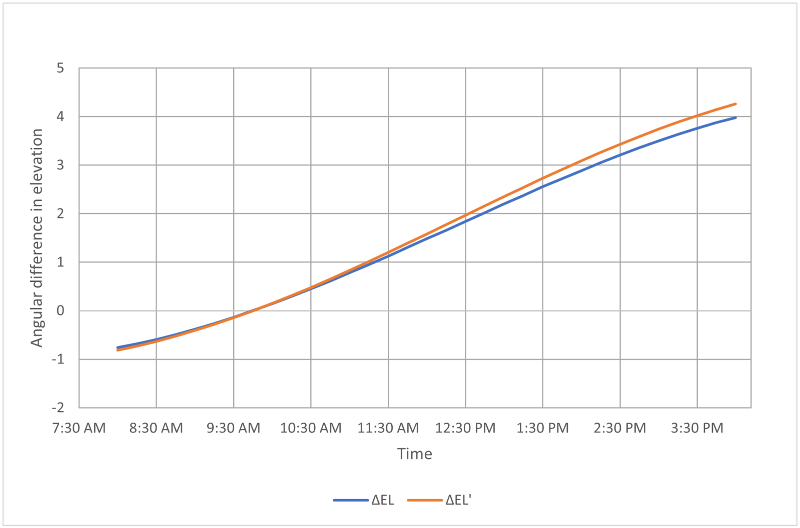


Figure 4.18: Angular differences of both, the effective elevation angle ΔEL and the elevation angle of the rotary axis $\Delta EL'$ over time with a fixed antenna alignment are shown. At a difference of 0°, the antenna alignment in elevation direction matches the desired elevation angle, calculated with TLE data. It is assumed, that the highest measured signal power occurs to this timestamp. A second effect is also shown in this figure. An increase in elevation lead to a higher offset between both, ΔEL and $\Delta EL'$.

In Figure 4.19 the frequency of the highest measured signal power is shown as a function of the time. According to the movement of the satellite, the frequency is changing. The signal frequency of Alphasat's beacon signal is 19.701 GHz. If the satellite moves towards Earth, the frequency increases. If the satellite moves away from the Earth, the frequency decreases. At first glance, this behavior would be due to the Doppler effect. A more detailed analysis of the evaluated distances between satellite and antenna with the Systems Tool Kit (STK, Ver. 11.4.1) has shown that the noticed frequency behavior is not exclusively due to Doppler. STK is "a platform for analyzing and visualizing complex systems" and interacts "with data from platforms across the aerospace, defense, telecommunications, and other industries" [45]. The exported data was printed over time and is visualized in Figure 4.20. It can be seen, that during the measurement the distance was decreasing which corresponds to an increase of the frequency. The comparison of the evaluated frequency change from SAAT data, with STK distance data, suggests that other effects influenced the measurement. The spectrum analyzer was used with the internal oscillator. Temperature changes inside the device might affect the quality of the measurement.

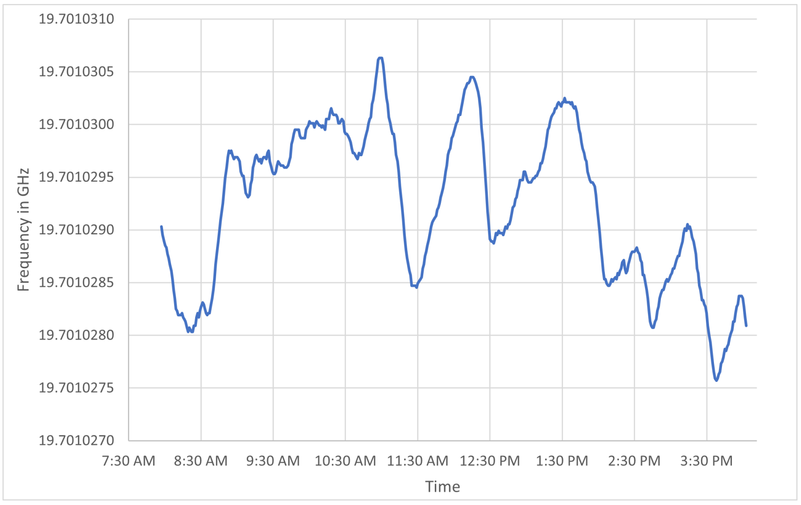


Figure 4.19: Change of the center frequency of the measured beacon signal of Alphasat over a time interval of a half day. The corresponding frequency to the highest evaluated signal power (as mentioned in Figure 4.17) to a certain timestamp was logged during the measurement. Due to movements of the satellite, the distance between the pico terminal and the satellite is changing over time. This leads to a change in the beacon signal frequency.

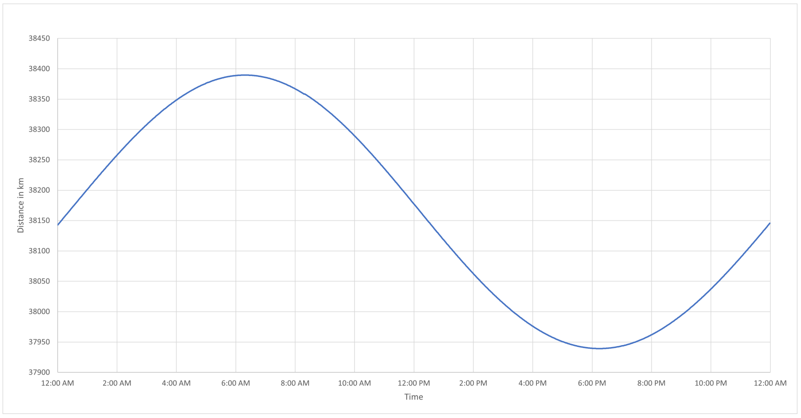


Figure 4.20: Change of distance between Alphasat and ground station in Graz during the day on March 9th, 2021.

4.5.7 Tracking of an inclined GEO-satellite for half a day

This test examines the tracking properties of the pico terminal. Like the previous test, the measured beacon signal was measured over time. In this case, the automatic tracking function with an update interval of 10 minutes was activated. All 10 minutes, the position of the azimuth and elevation axis were adjusted, according to the TLE data.

In Figure 4.21 the measured power of the beacon signal is shown over time. The measurement started at 8:30 AM with a power level of -75.23 dBm. It was assumed, that this power level should be constant over the whole measurement. Contrary to this assumption, the power level has increased steadily. Right after the first pointing procedure, a quick signal check was done. By increasing and decreasing the elevation axis, the highest signal power was evaluated. It has been shown that the elevation axis was set about 2° too high after the automatic alignment. Nevertheless, the test was carried out with the calculated values. The elevation axis was increased cyclically during the test. At the beginning of the measurement, the elevation angle of the rotary axis EL' was 34.6°. At the end of the measurement, the calculated value for EL' was 39.36°. A reference measurement with a digital protractor results in a value for EL' of 37.66°. In this case, an angle loss of 1.7° during the whole measurement over half a day occurred. This behavior can be also seen in Figure 4.21. At the start of the measurement, the angular difference of 2° was determined, as already mentioned. At this time, the configured elevation angle was too high. During the test, several steps were lost at every update interval. Due to this effect, the angular distances decreased every update interval and the measured power level was getting higher. At the end of the measurement, the angular difference was nearly in the range of 0°. This can be also seen in the measurement results.

Compared to the measured signal in Figure 4.17, in this test a much higher power level was achieved. Also, the difference of the highest and the lowest power level is only 3 dB instead of 9 dB without tracking.

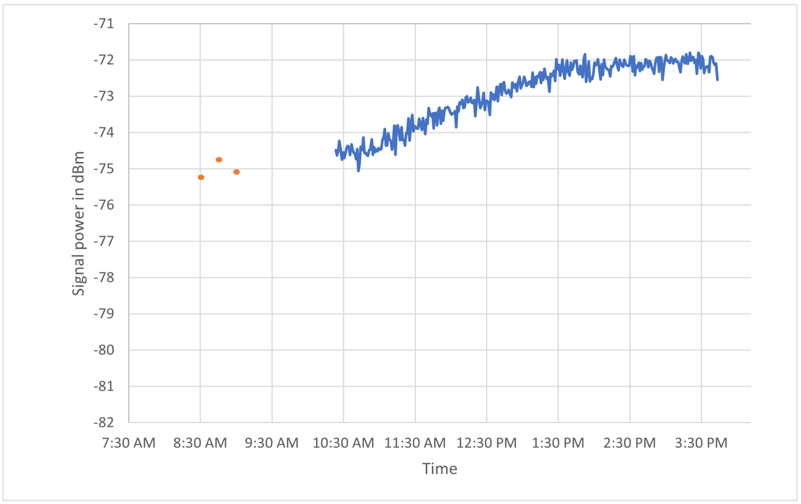


Figure 4.21: Measured signal power with activated tracking function. Similar to Figure 4.17, the highest power level at a certain timestamp was determined. No spectrum data logged between 8:30 AM and 10:30 AM due to a wrong update interval configuration. The three values read off directly at the beginning of the measurement can be seen as orange dots and mark the starting power level of the measured signal. With the logged data and the manual values at the beginning, the power curve can be used for comparison purposes.

Chapter 5

Summary and outlook

The aim of this work is to upgrade an existing satellite pico terminal so that it features both electrically operated antenna alignment and antenna tracking. For this purpose, existing satellite tracking concepts have been reviewed. This review showed that programme tracking, based on TLE data is a common and realizable method for the given hardware. Algorithms for more precise satellite pointing go beyond the scope and are not dealt in this thesis.
Since for the existing satellite pico terminal no documentation on the geometry was available, several parameters had to be determined in the first place. One of these parameters is the offset angle of the parabolic antenna dish. Other parameters are the gear ratios of the axes that are used to align the azimuth and the elevation angle of the antenna and the torque that is required for the rotation. Further, an alignment correction model to identify the effective pointing direction was established and implemented into the control software.
Besides the control software and the graphical user interface, a variety of different hardware components are a part of the upgrade. Several of these hardware components have been specially designed and manufactured as part of the thesis. With a variety of tests, that have been defined and then carried out, the functionality of all components of the upgraded pico terminal has been tested. The results show that the combination of spare parts and additional hardware and software components enables electrically operated antenna alignment and antenna tracking. Results show further that automated tracking of a geostationary satellite is possible. A comparison of the signal power change from an inclined GEO satellite over time shows that activated tracking enables significantly higher and more constant signal levels.
With the manual control panel and the installed stepper motor, the antenna can be aligned in azimuth and elevation without any physical effort. So, it is possible to align the antenna and to observe the received signal on a spectrum analyzer at the same time. This feature is especially useful for educational purposes. In this way, the focus can be placed on the essentials during a laboratory exercise. The COVID-19 pandemic has impressively shown the importance of online platforms and virtual teaching. With the upgraded pico terminal it is now possible to control the axes with remote access directly via the GUI and observe the alignment of the antenna with an optional webcam.
Nevertheless, there is room for improvement in some areas. With the currently used open-loop stepper motors, occasionally step losses occur when adjusting the axes, especially in

overload situations. Using a closed-loop motor drive might be a better choice for tracking systems compared to an open-loop system. In this case, jerky movements or lost steps due to overload can be detected. Even in a closed control loop, step losses can occur if the encoder is mounted on the motor shaft and not on the axis of the antenna movement. Upgrading the motor drives to a closed-loop system and comparing the results with the current open-loop configuration when tracking a satellite may be considered for the future.
The idea of determining the orientation of objects using an IMU has proven to be impractical for installation on the feed arm of the antenna unit. The magnetic interference that occurs during operation has a negative effect on the measurement results of the sensor.
Numerous tests were carried out as part of this work to check functionality of the pointing and tracking features of the upgraded terminal. These tests can also be implemented into future laboratory exercises. Ideas for further test setups are listed below:

- Tracking Alphasat during a rain event and discuss the behavior of the 19.7 GHz signal with respect to fades. During a precipitation event, electromagnetic waves are scattered and absorbed by raindrops, resulting in higher attenuation and a lower signal level on the receiver side.
- Tracking a LEO satellite to evaluate the tracking behavior at fast update intervals to follow the satellite's movement. Due to the LEO, the visibility of a satellite in such an orbit and thus the radio contact to a ground station is less than 15 minutes per cycle.
- Performing measurements of the building entry loss at Ka-band frequencies. When the terminal is placed indoors e.g. behind a window the loss is typically quite high. Tests with the pico terminal have shown that at a frequency of 19.701 GHz, the signal is attenuated by at least 25 dB, when the signal has to propagate through trough triple insulated glazing.

References

[1] Union of Concerned Scientists. (2021) Ucs satellite database. Accessed on 2021-04-09. [Online]. Available: https://www.ucsusa.org/resources/satellite-database

[2] Starlink. (2021) Starlink. Accessed on 2021-04-09. [Online]. Available: https://www.starlink.com

[3] "Ieee standard definitions of terms for antennas," *IEEE Transactions on Antennas and Propagation*, vol. 17, no. 3, pp. 262–269, 1969.

[4] M. O. Kolawole, *Satellite Communication Engineering, Second Edition.* CRC Press Taylor & Francis Group, 2014.

[5] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design, Third Edition.* John Wiley & Sons, Inc., 2013.

[6] H. Lehpamer, *Microwave Transmission Networks: Planning, Design, and Deployment, Second Edition.* Mc Graw Hill, 2010.

[7] C. A. Balanis, *Antenna Theory Analysis and Design, Third Edition.* John Wiley & Sons, Inc., 2005.

[8] B. R. Elbert, *Introduction to Satellite Communication, Third Edition.* ARTECH HOUSE, INC., 2008.

[9] J. N. Pelton, S. Madry, and S. Camacho-Lara, *Handbook of Satellite Applications, Second Edition.* Springer International Publishing, Switzerland, 2017.

[10] S. Kidder, "Satellites and satellite remote sensing — orbits," in *Encyclopedia of Atmospheric Sciences (Second Edition)*, 2015, pp. 95–106.

[11] NIST. (2010) Time and frequency from a to z, s to so. Accessed on 2021-04-09. [Online]. Available: https://www.nist.gov/pml/time-and-frequency-division/popular-links/time-frequency-z/time-and-frequency-z-s-so

[12] A. C. Clarke, "Peacetime uses for v2," in *Wireless World*, February 1945, p. 58.

[13] ESOA. (2021) Satellite orbits. Accessed on 2021-04-09. [Online]. Available: https://www.esoa.net/technology/satellite-orbits.asp

[14] H. D. Curtis, *Orbital Mechanics for Engineering Students.* Elsevier Butterworth-Heinemann, 2005.

[15] E.-I. CROITORU and G. Oancea, "Satellite tracking using norad two-line element set format," *SCIENTIFIC RESEARCH AND EDUCATION IN THE AIR FORCE*, vol. 18, pp. 423–432, 2016.

[16] Celestrak. (2021) Celestrak.com catnr=25544. Accessed on 2021-03-17. [Online]. Available: https://celestrak.com/satcat/tle.php?CATNR=25544

[17] D. T. Kelso. (2019) Frequently asked questions: Two-line element set format. Accessed on 2021-03-17. [Online]. Available: http://celestrak.com/columns/v04n03/

[18] Brandon Rhodes. (2020) ephem 3.7.7.1. Accessed on 2021-03-16. [Online]. Available: https://pypi.org/project/ephem/

[19] H. Si and Z. M. Aung, "Position data acquisition from nmea protocol of global positioning system," in *International Journal of Computer and Electrical Engineering*, vol. 3, no. 3, 2011, pp. 353–357.

[20] E. Prenesti and F. Gosmaro, "Trueness, precision and accuracy: a critical overview of the concepts as well as proposals for revision," *Accreditation and Quality Assurance*, vol. 20, pp. 33–40, 12 2014.

[21] O. Cartiaux, J.-Y. Jenny, and L. Joskowicz, "Accuracy of computer-aided techniques in orthopaedic surgery: How can it be defined, measured experimentally, and analyzed from a clinical perspective?" *The Journal of Bone and Joint Surgery*, vol. 99, p. e39, 04 2017.

[22] Wahyudi, M. S. Listiyana, Sudjadi, and Ngatelan, "Tracking object based on gps and imu sensor," in *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2018, pp. 214–218.

[23] T. Yoo, S. Hong, H. Yoon, and S. Park, "Gain-scheduled complementary filter design for a mems based attitude and heading reference system," *Sensors (Basel, Switzerland)*, vol. 11, pp. 3816–30, 12 2011.

[24] M. Pedley, *Tilt Sensing Using a Three-Axis Accelerometer*, Freescale Semiconductor, Inc., 3 2013, rev. 6.

[25] A. Civita, S. Fiori, and G. Romani, "A mobile acquisition system and a method for hips sway fluency assessment," *Information*, vol. 9, p. 321, 12 2018.

[26] R. Ludwig and P. Bretchko, *RF Circuit Design, Theory and Applications.* Prentice Hall NJ, 2000.

[27] O. Koudelka, "Q/v-band communications and propagation experiments using alphasat," *Acta Astronautica*, vol. 69, no. 11, pp. 1029–1037, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576511002141

[28] C. Rauscher, "Fundamentals of spectrum analysis, germany: Rohde & schwarz, 5th edition," 2011.

[29] D. Galar and U. Kumar, "Chapter 3 - preprocessing and features," in *eMaintenance*. Academic Press, 2017, pp. 129–177. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128111536000038

[30] P. P. Acarnley, *Stepping motors: A Guide to Theory and Practice, 4th Edition.* Institution of Electrical Engineers, 2007.

[31] R. Condit and D. Jones. (2004) Stepping motors fundamentals, microchip technology inc, an907. Accessed on 2021-05-15. [Online]. Available: http://ww1.microchip.com/downloads/en/Appnotes/00907a.pdf

[32] A. Böge, *Vieweg Handbuch Maschinenbau, Band 1, 8. Auflage.* Vieweg, 2004.

[33] G. Maral and M. Bousquet, *Satellite Communications Systems, Fifth Edition.* John Wiley & Sons, Inc., 2009.

[34] G. Hawkins, D. Edwards, and J. McGeehan, "Tracking systems for satellite communications," in *IEE PROCEEDINGS, Vol. 135, Pt. F, No. 5*, vol. 135, no. 5, 1988, p. 393407.

[35] K. Kolesnyk, R. Panchak, I. Kozemchuk, and Z. Skybinska, "Development of an automated subsystem for modeling and calculating a mirror antenna from its guiding and tracking the target," in *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, 2019, pp. 1–5.

[36] A. A. Yassin Abdelkarim, M. E. Babikir Osman, and S. F. Babiker, "Satellite orbit determination and tracking model using two-line elements set," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2018, pp. 1–6.

[37] A. Khanlari and F. MansourKiaie, "A new efficient algorithm for tracking leo satellites," in *2013 IEEE International Systems Conference (SysCon)*, 2013, pp. 587–590.

[38] J. E. Espndola Daz and A. A. Rodrguez Hernndez, "Methodology to implement a satellite ground university station," in *2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 2015, pp. 261–267.

[39] A. E. Putra, B. A. A. Sumbada, and A. Nurbaqin, "Satellite tracking control system for ugm ground station based on tle calculation," in *2016 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2016, pp. 91–96.

[40] Python Software Foundation. (2021) Using python on unix platforms. Accessed on 2021-02-21. [Online]. Available: https://docs.python.org/3/using/unix.html

[41] B. Sensortec, *BNO055 data sheet: Intelligent 9-axis absolute orientation sensor*, Bosch Sensortec, 6 2016, rev. 1.4.

[42] Adafruit BNO055 Absolute Orientation Sensor. (n.d.) Adafruit bno055 absolute orientation sensor. Accessed on 2021-03-11. [Online]. Available: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/device-calibration

[43] GoogleMaps. (2021) Accessed on 2021-03-19. [Online]. Available: https://www.google.at/maps/place/47\%C2\%B003'31.9\%22N+15\%C2\%B027'34.6\%22E/@47.058902,15.4595503,49m/data=!3m1!1e3!4m5!3m4!1s0x0:0x0!8m2!3d47.0588611!4d15.4596111

[44] GIS. (2021) Accessed on 2021-03-19. [Online]. Available: https://gis.stmk.gv.at/wgportal/atlasmobile/map/Basiskarten/Basiskarte

[45] I. Analytical Graphics. (2021) Systems tool kit (stk). Accessed on 2021-05-08. [Online]. Available: https://www.agi.com/products/stk

Appendix A

User guide

These operating instructions should be read through before using the motorized pico terminal for the first time. All relevant information regarding preparation, finding the right measurement location, leveling and operation are explained in this instruction. To ensure safe handling of the laboratory instrument, the warnings listed in Section A.9 must be observed.

A.1 General information

The motorized pico terminal as explained in Figure A.1 is an upgraded version of the original pico terminal. The main difference can be found in the operation. While the original pico terminal is only aligned manually with a corresponding effort, the motorized version offers several advantages. The upgraded version is characterized by convenient operation with the manual control panel and a graphical user interface with various configuration and display options. Additional pointing and tracking functions as well as the possibility of integration into virtual lessons with the help of remote access offer new possibilities in the laboratory exercise.
The pico terminal consists of a parabolic offset antenna and a low noise amplifier (LNA) with a linear polarized feed. The feed is horizontal polarized at $0°$ skew. To change the polarization angle, the whole antenna dish needs to be moved around the skew axis by hand. Both axes, azimuth and elevation are adjusted with a motor drive. The drives are controlled via a manual control panel or the graphical user interface (GUI). As a control unit a RaspberryPi with Linux operating system is used.

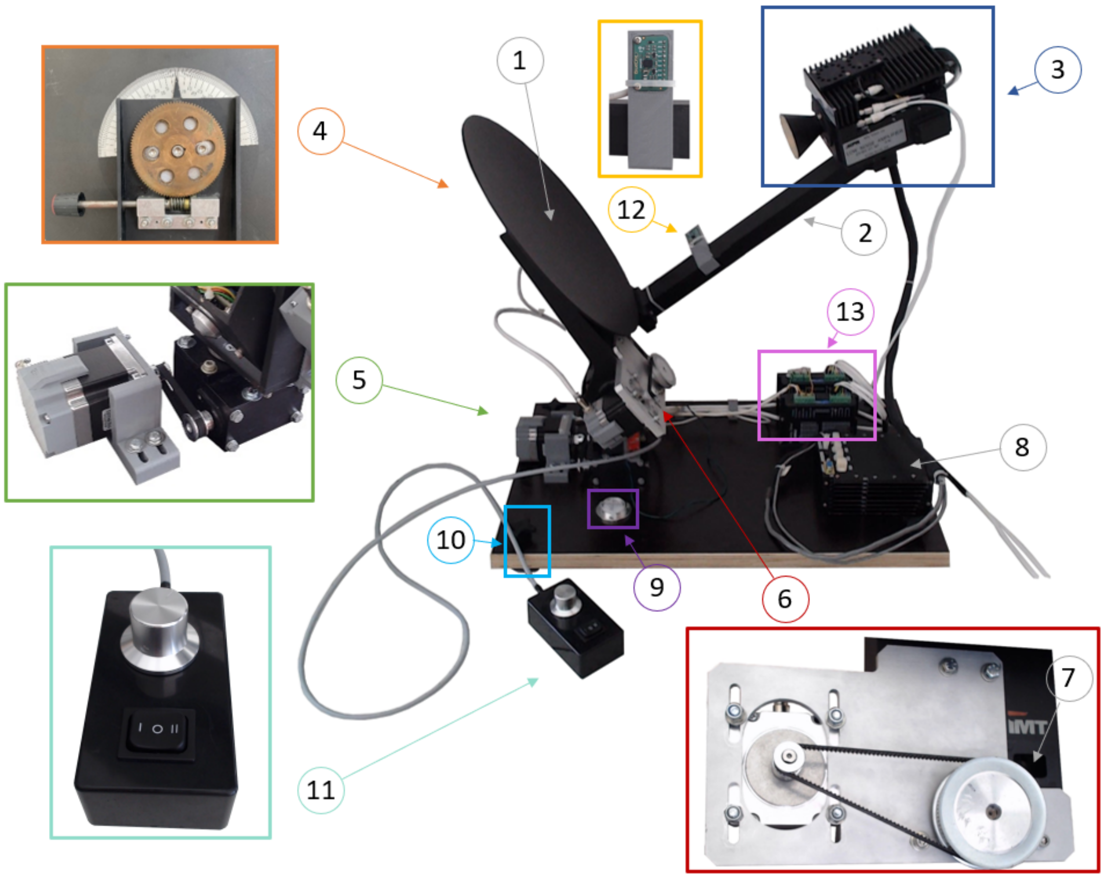


Figure A.1: Components of the upgraded pico terminal: parabolic dish (1), the feed arm (2) and the LNA (3) are firmly connected to one another and are aligned as one unit when pointing. Worm drives are used to adjust the skew (4) angle of the pico terminal. To adjust the azimuth (5) and elevation (6) angle, stepper motors in combination with different sized pulleys, a belt and the previously existing worm drives are used. The elevation scale (7) is still available, but only used for angle comparison reasons. One power supply (8) is necessary to power the LNA. A leveling bubble (9) together with four adjustable feet (10) are used to level the ground plate on the actual measurement site. The manual control panel (11) with a switch to select the axis and a rotary encoder can be used for manual antenna alignment. The IMU or 9-axis sensor (12), mounted on the feed arm, delivers optional antenna orientation data. To control the stepper motors, micro step drivers (13) are used. The control unit, a RaspberryPi with a Linux operating system, is connected to all devices. Both, the control unit and the GPS receiver are not illustrated. Additional power supplies for the micro step drivers and the control unit are also not shown.

A block diagram of the upgraded pico terminal is illustrated in Figure A.2. In this configuration the control unit can be accessed via a remote computer. Other options to control the system are explained in Section A.2.

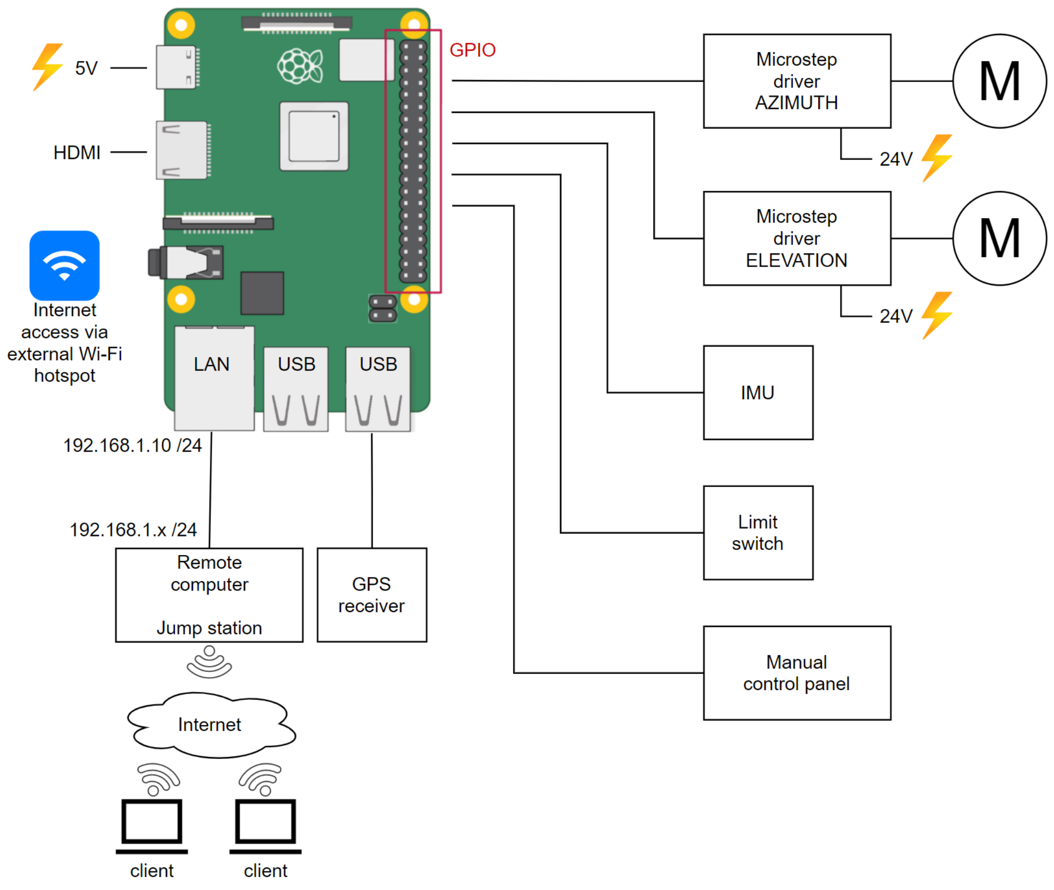


Figure A.2: Block diagram of the antenna pointing/tracking unit of the pico terminal: all hardware components, apart from the GPS receiver, are connected to the GPIO pin header of the control unit. The GPS receiver is connected via USB. The stepper motors for azimuth and elevation are controlled with additional micro step drivers. The desired step directions and impulses are generated by the control unit. Wi-Fi connection to an external hotspot is used to access the internet. A cable connection between the control unit and a remote computer via LAN is on method, to control the system. The pre-configured IP address of the control unit is visualized. To establish a connection via LAN, the remote computer needs to be in the same IP address range. For the use of a monitor, one full-size High Definition Multimedia Interface (HDMI) port is available.

An additional block diagram is shown in Figure A.3. In this diagram the RF setup is described. Due to its predominant use as a receiving unit, only the downconverting setup is considered. To support spectrum analyzers in a lower frequency range, an optional down converter can be used.
The 20 GHz signal from the LNA can also be measured directly with the appropriate RF measuring equipment, as shown in Figure A.4.

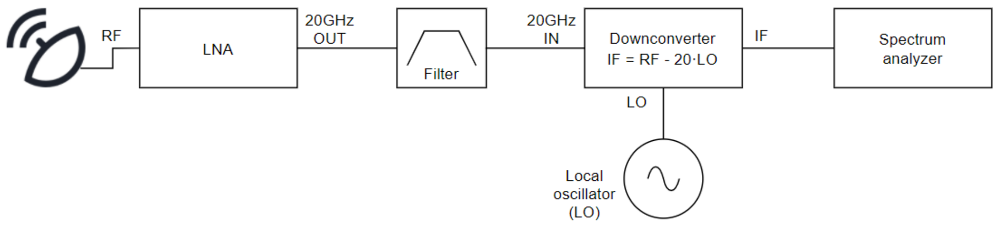


Figure A.3: RF block diagram of the pico terminal: The received RF signal is amplified by the LNA. An optional filter and a down converter in combination with a local oscillator can be used to mix the signal into a lower frequency range.

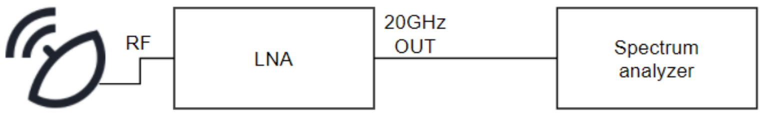
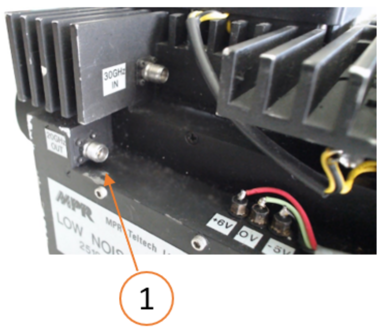
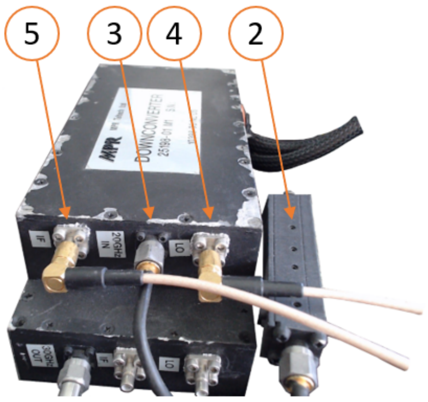


Figure A.4: RF block diagram of the pico terminal without down converter: The received RF signal is amplified by the LNA and directly measured with an appropriate spectrum analyzer.

In Figure A.5 the connectors of the LNA and the down converter are described. Each connector has a specific label. The same labels are used in the block diagrams in Figure A.3 and A.4.



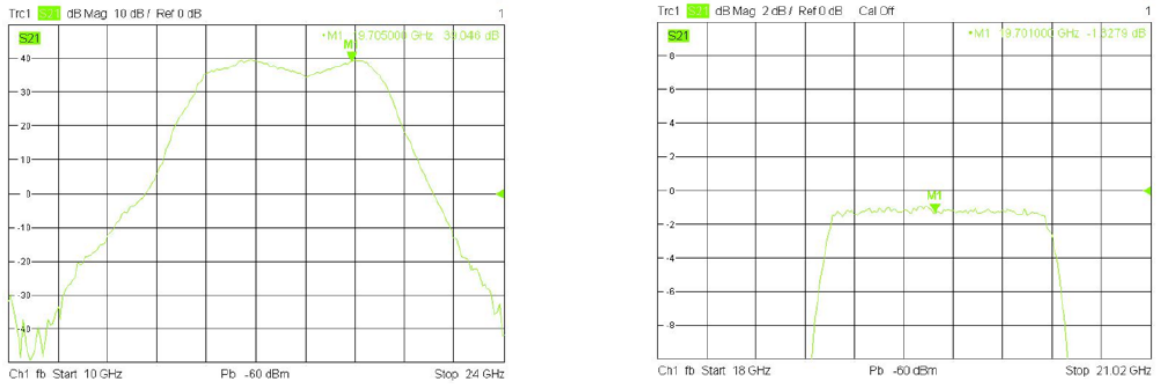
(a) LNA



(b) Down converter

Figure A.5: RF connectors of the LNA and the down converter: The amplified signal after the LNA (1) can be directly connected to an appropriate spectrum analyzer or to the filter stage (2). The filterd signal is down converted (3) with the use of a local oscillator (4). The output signal at intermediate frequency (5) can then be measured with a spectrum analyzer.

The gain of the LNA and the characteristics of the RF filter are frequency dependent and illustrated in Figure A.6.

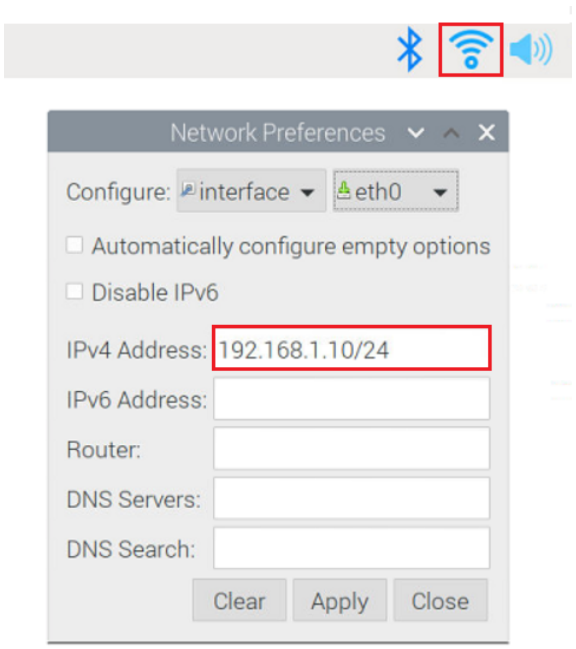


(a) LNA gain **(b)** RF filter characteristics

Figure A.6: Frequency dependence of LNA gain and RF filter. At a frequency of approximately 19.7 GHz the gain of the LNA is 39 dB. At the same frequency, the signal is attenuated with 1.3 dB by the RF filter.

A.2 Set up different operating modes

There are two possible ways to operate with the RaspberryPi. One option is to use the RaspberryPi as a normal computer with monitor, mouse and keyboard or with just a touch monitor instead. The monitor can be connected over HDMI, for mouse and keyboard the available Universal Serial Bus (USB) ports can be used. The second option to operate with the control unit is via Virtual Network Computing (VNC). A VNC server is running on the RaspberryPi, which enables incoming connections to control the device. For this purpose, the free-ware tool *TightVNC Viewer*, which is available online needs to be installed on a remote computer. To establish a connection to the RaspberryPi, the network settings of the remote computer must match the configured network settings of the RaspberryPi. The default settings for the Local Area Network (LAN) adapter of the RaspberryPi are shown in Figure A.7.


Network Preferences

Configure: interface eth0

Automatically configure empty options

Disable IPv6

IPv4 Address: 192.168.1.10/24

IPv6 Address:

Router:

DNS Servers:

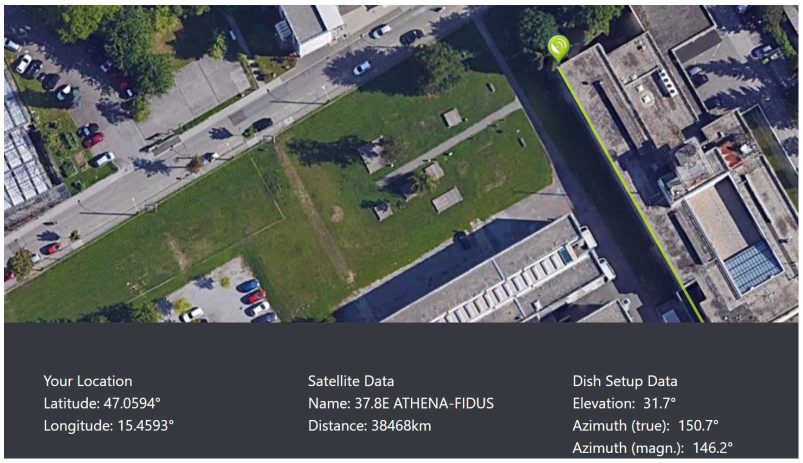
DNS Search:

Figure A.7: LAN settings of the RaspberryPi shown in the value display with the red border. Settings can be accessed via the Wi-Fi symbol in the taskbar.

For the use in virtual teaching, the remote computer can be used as a jump station. The Wi-Fi network adapter of the jump station is connected with a hotspot for internet access. At the same time, a wired VNC connection over the LAN network adapter of the jump station and the control unit of the pico terminal is used to visualize and control the software running on the control unit. Clients from remote locations can access the jump station with a desired remote access software over the internet and are able to control the jump station. With this setup it is possible to control the pico terminal from any remote location.

A.3 Recommended location and leveling

To guarantee a good satellite link, a direct connection without any obstacles like trees or walls between the pico terminal and the satellite is necessary. In most cases, data of the IMU sensor for heading are invalid. For this reason, it is necessary to find a well-known reference location which can be used as a guide to set the reference value for the azimuth direction. For this purpose, there are different platforms available online. A recommended platform is *Dishpointer*. Normally, this tool can be used to determine the parameters for aligning the antenna for a defined location and the desired satellite. But it is also possible to determine the orientation of a building wall. This principle is shown in Figure A.8.



Your Location
Latitude: 47.0594°
Longitude: 15.4593°

Satellite Data
Name: 37.8E ATHENA-FIDUS
Distance: 38468km

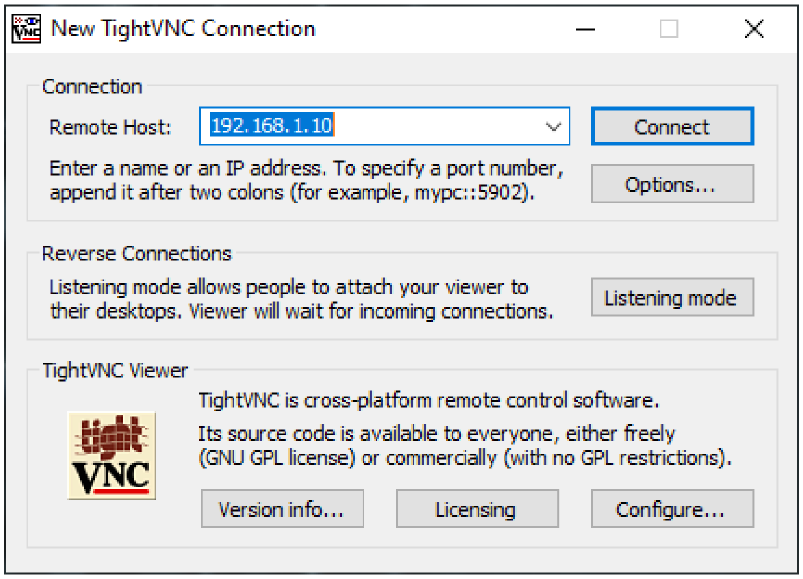
Dish Setup Data
Elevation: 31.7°
Azimuth (true): 150.7°
Azimuth (magn.): 146.2°

Figure A.8: Location estimation using the online platform *Dishpointer*. the parameters for elevation, azimuth and skew for a specific antenna position and the desired satellite are shown. It is also possible to use this platform to determine the orientation of a building wall in azimuth direction. This method is helpful when setting the azimuth reference point of the pico terminal.

The green location marker is set to the edge of a wall and different satellites from the drop-down list are selected. This drop-down list is not visualized in Figure A.8. If the green line matches the orientation of the wall of the building, the (true) Azimuth value shown at the bottom can be used for reference purposes. In this special case the satellite *37.8E ATHENA-FIDUS* can be used to evaluate the alignment of the wall of the institute building. It is necessary to level the ground plate of the pico terminal to ensure a correct pointing. This can be done with the four adjustable feet at the corners of the plate using the circular leveling aid. A more accurate option will be using a digital protractor instead of the circular leveling aid.

A.4 Start up

By connecting the power plug, the RaspberryPi starts and the stepper drivers are ready for use. A switch must be on to supply the LNA with power. When using a remote computer to operate, the VNC connection can be established. The default Internet Protocol (IP) address is shown in Figure A.9.


New TightVNC Connection

Connection

Remote Host: 192.168.1.10

Enter a name or an IP address. To specify a port number, append it after two colons (for example, mypc::5902).

Reverse Connections

Listening mode allows people to attach your viewer to their desktops. Viewer will wait for incoming connections.

TightVNC Viewer

TightVNC is cross-platform remote control software.

Its source code is available to everyone, either freely (GNU GPL license) or commercially (with no GPL restrictions).

Figure A.9: Establishing a VPN connection to the RaspberryPi via the software tool *TightVNC Viewer*. The configured IP address of the RaspberryPi is necessary. The default configuration is highlighted in blue. By clicking the *Connect* button, a pop-up occurs. With the correct password, the RaspberryPi can be accessed remotely.

To start the pico terminal software, the command line shown in Listing A.1 need to be run in a terminal. Another option is to use a Python Integrated Development Environment (IDE) like *IDLE 3* and to select the python script. To start the GUI, the module needs to be run.

Listing A.1: Command line to run the python script with the terminal. A GUI with the main screen appears right after starting the script.

```
python3 /home/pi/picoTerminal/picoTerminal.py
```

A.5 Setting up a Wi-Fi connection

If an Internet connection is required when using the pico terminal, for example to download TLE data, a Wi-Fi connection must be set up at the beginning. This can be done by clicking on the Wi-Fi symbol in the taskbar. Available connections are shown in a pop-up window which can be selected by a left mouse click (see Figure A.10). Depending on the security level, the required password must be entered. If no connection is needed, this step can be skipped.

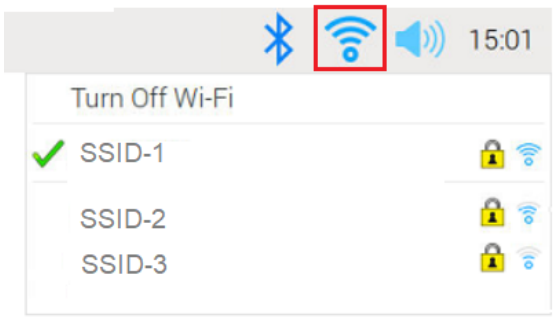
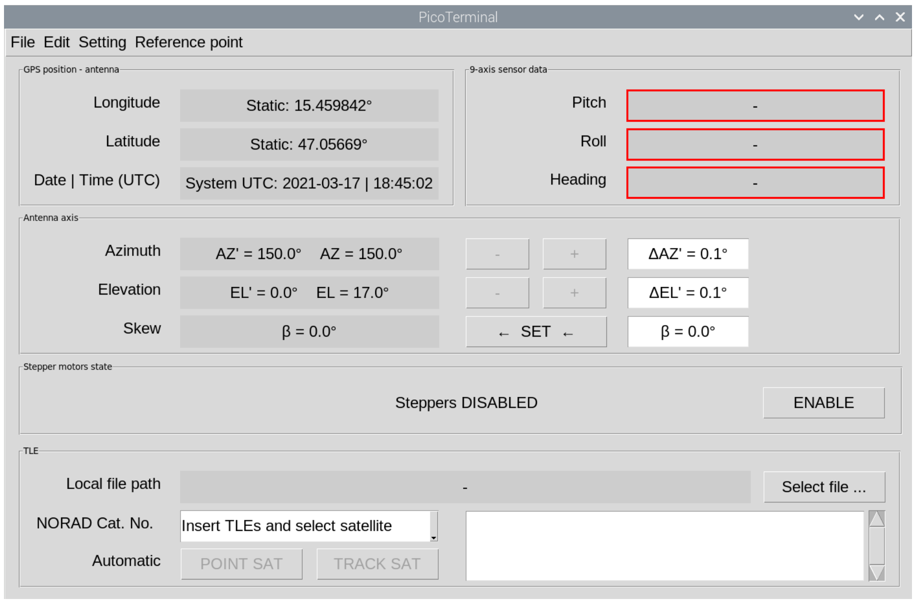


Figure A.10: Wi-Fi settings of the RaspberryPi. With a left mouse click on the Wi-Fi symbol in the taskbar, a list of available Service Set Identifiers (SSIDs) is shown. It is recommended to set up a hotspot with a smartphone and select the configured hotspot name from the list.

A.6 User interface - main screen

The main screen is split into different sections as shown in Figure A.11. On top of the screen a menu bar is available. Loading TLE data or changing settings can be accessed via this menu. Actual sensor data from the GPS receiver and the 9-axis-sensor are shown in the upper region. To change azimuth and elevation axis, the middle section is important. With the ENABLE/DISABLE button, the stepper drivers can be switched on or off. In the disabled state the stepper motors immediately stop to move. In the bottom region the automatic functions for pointing and tracking are available.



Longitude: Static: 15.459842°
Latitude: Static: 47.05669°
Date | Time (UTC): System UTC: 2021-03-17 | 18:45:02
Pitch: -
Roll: -
Heading: -
Azimuth: AZ' = 150.0° AZ = 150.0° ΔAZ' = 0.1°
Elevation: EL' = 0.0° EL = 17.0° ΔEL' = 0.1°
Skew: β = 0.0° β = 0.0°
Steppers DISABLED
Local file path: -
NORAD Cat. No.: Insert TLEs and select satellite
Automatic

Figure A.11: Main screen of the pico terminal software. The GUI is split into different sections. Value displays with a red border (9-axis sensor data) indicate a not fully calibrated 9-axis-sensor.

A.7 Adjust azimuth and elevation axis

To adjust azimuth or elevation, different options are available. One possibility is to use the + or the − button for the corresponding axis. A setpoint can be defined by clicking in the entry box (see "Antenna axis" in Figure A.11). According to touch support, a keyboard pops up and the desired value can be configured.
Another option to change the axis is by using the manual control panel, visualized in Figure A.12. The desired axis can be selected by setting the switch in position 1 for azimuth, or position 2 for elevation. With the rotary control the selected axis can be adjusted (rotate clockwise for positive adjustments). There are three different modes available to configure the number of steps per increment (see Section A.8.3). To roughly position an axis, a mode with many steps per increment is recommended. For fine tuning, a mode with only a few steps is the best option. Modes can be changed by clicking on the rotary control. The actual mode is also displayed on the GUI.

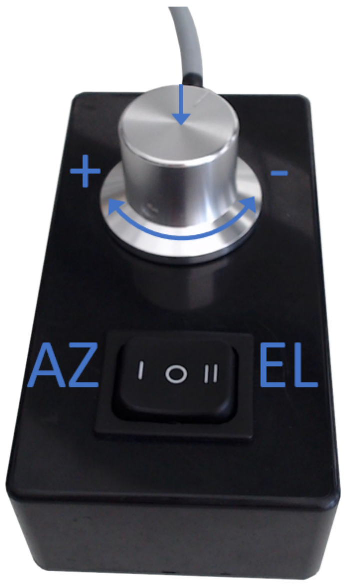


Figure A.12: Manual control panel for manual antenna adjustments. A switch for axis selection and a rotary encoder are the main components. Positive axis adjustments are made with the rotary encoder in clockwise direction. Pressing the button of the rotary encoder changes between three different step modes.

It is also possible to point or track a satellite using TLE data. More information to this topic can be found in A.8.1.

It is important to know, that there are two angular values for both axes. One value describes the actual angle of the pico terminal axis (AZ' for azimuth, EL' for elevation) set by the stepper motors and the second value is the effective angle (AZ for azimuth, EL for elevation) the antenna is pointing at. Depending on the configured skew angle, which can only be changed manually by hand, the effective direction changes without moving any motor. Due to this effect, a change by 1° (EL') does not necessarily have to mean a change in the effective direction (EL) by 1°. Apart from a possible offset depending on the skew, a change of 1° (AZ') of the azimuth axis is equivalent to an effective change of

1° (*AZ*).

It is important to configure the manually set skew angle in the entry box of the main screen. By clicking the *SET* button, the effective values for azimuth and elevation are updated. The *SET* button is highlighted yellow, if the configured skew angle has changed but was not set (see Figure A.13).

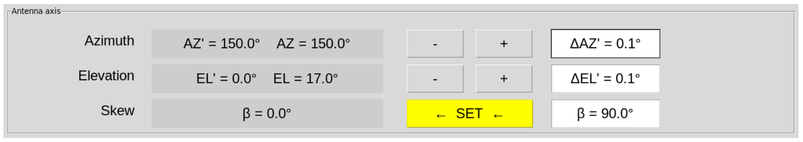


Figure A.13: *SET* button to change the skew angle. For correct azimuth and elevation calculations the manually set skew angle must be configured. In this case, a skew angle of 90° was configured, but not set. This is visualized with the yellow highlighted *SET* button. When clicking the *SET* button, the skew angle (named β) is also set to 90° and the values for *AZ* and *EL* are changing, according to the configured skew angle.

A.8 Menu

To change settings or to load TLE data, the menu bar must be used. The available options are explained in this section.

A.8.1 File

In the *File* item, the CelesTrak website can be opened to search the desired satellite and to copy and paste the TLE data into the entry box in the TLE section of the main screen. Another option can be used to download TLE data. With the menu item *Load TLE data* TLE data from a specific satellite catalogue number can be downloaded (see Figure A.14). For this purpose, the satellite catalogue number must be known (e.g. 39215 for Alphasat). In this case, TLE data is inserted automatically in the TLE entry box as shown in Figure A.15.

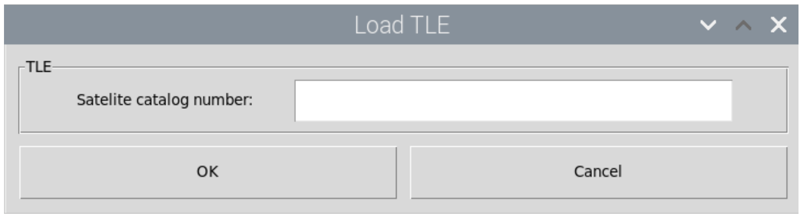
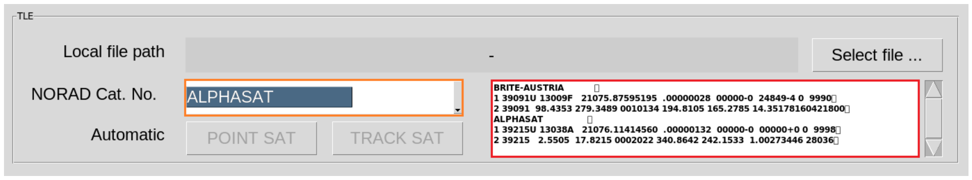


Figure A.14: Load TLE data with satellite catalogue number. By clicking in the entry box, a keyboard opens. The desired satellite catalogue number can be typed in and by clicking the *OK* button, the TLE data is downloaded from the *CelesTrak* website and inserted in the TLE entry box (see Figure A.15).



```
BRITE-AUSTRIA
1 39091U 13009F   21075.87595195  .00000028  00000-0  24849-4 0  9990
2 39091  98.4353 279.3489 0010134 194.8105 165.2785 14.35178160421800
ALPHASAT
1 39215U 13038A   21076.11414560  .00000132  00000-0  00000+0 0  9998
2 39215   2.5505  17.8215 0002022 340.8642 242.1533  1.00273446 28036
```

Figure A.15: TLE entry box (marked with a red border) with TLE data of two satellites. Data can be added via the *Load TLE* function (see Figure A.14), via copy and paste from other sources, by selecting a file with offline TLE data (*Select file ...*) or typed in by hand (last option is not recommended). It is important to select the desired satellite in the combo box, marked in orange.)

A.8.2 Edit

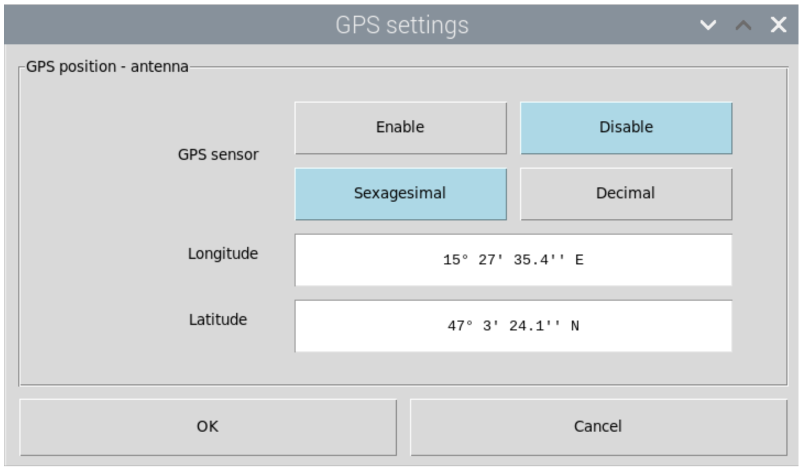
The menu item *Edit* can be used to copy, select or delete all TLE entries in the TLE entry box. When using a touch screen, this might be a useful option.

A.8.3 Settings

Different settings can be accessed with the menu item *Settings*.

GPS

It is possible to enable or disable the GPS receiver or to change the format from sexagesimal (degree, minute, second) to decimal and vice versa (see Figure A.16). Actual selections are highlighted in blue. If the GPS receiver is disabled, manual values for longitude and latitude can be set. Depending on the selected format different keyboard panels (see Figure A.17) occur to enter the desired values. Actual values for date, time and the GPS position come from the GPS receiver and are visualized on the main screen. If position data or the date are not valid, automatic pointing and tracking is not possible. In this case it is recommended to disable the GPS receiver to use the system date and time from the RaspberryPi instead. The system date and time are synchronized over the Internet connection when using Wi-Fi.


GPS settings

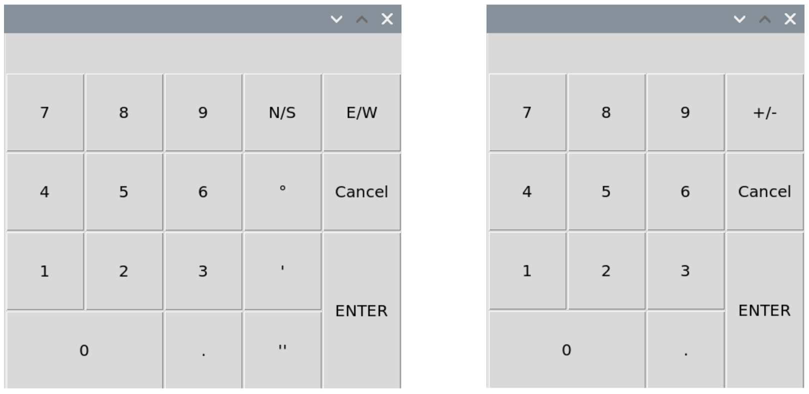
GPS position - antenna

GPS sensor: Enable | [✓ Disable]; [✓ Sexagesimal] | Decimal

Longitude: 15° 27' 35.4'' E

Latitude: 47° 3' 24.1'' N

Figure A.16: GPS settings with the possibility to enable or disable the GPS receiver or to change the format. Actual selections are highlighted in blue. If the GPS receiver is disabled, the configured values in the *Longitude* and *Latitude* entry boxes are used as position data for the TLE calculations while pointing or tracking a satellite. In addition, the system time of the RaspberrPi is used and in the main screen, the prefix "System UTC: " is added.



(a) Keyboard for sexagesimal GPS format

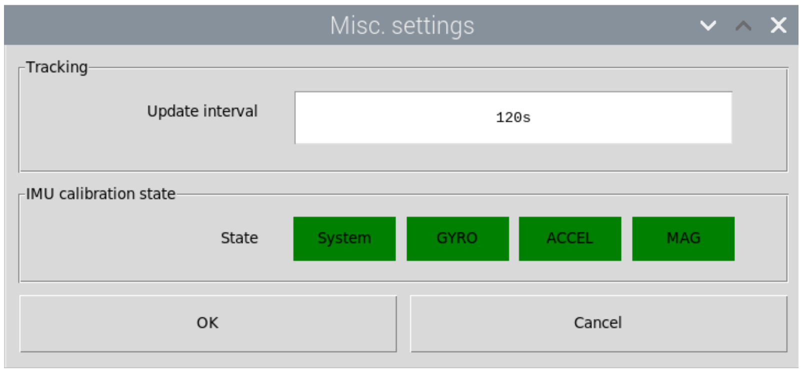
(b) Keyboard for decimal GPS format

Figure A.17: Different keyboard panels to configure values for longitude and latitude

Tracking and IMU

The tracking interval can be modified, as shown in Figure A.18. When tracking a LEO satellite, a much lower update interval is necessary in comparison to a GEO satellite. To get actual values of the 9-axis-sensor or IMU, the device must be calibrated. The IMU is a device to measure an object's orientation, in this case the alignment of the antenna. A recommended way do calibrate the sensor is available in the sensors user manual [41]. The calibration status of each sensor is visualized with a colored value display and shown in Figure A.18. At calibration status "0" the sensor is not calibrated. If the status reaches the value "3", the sensor is fully calibrated. If all value displays are green, the calibration is done. The actual calibration status is also visible in the main screen. Red borders around the measured values indicate a not fully calibrated sensor. In fact, it is not necessary to calibrate the sensor, if other measure devices (e.g. digital protractor) are used instead.

The evaluated values are only displayed on the main screen but have no influence on the program sequences.

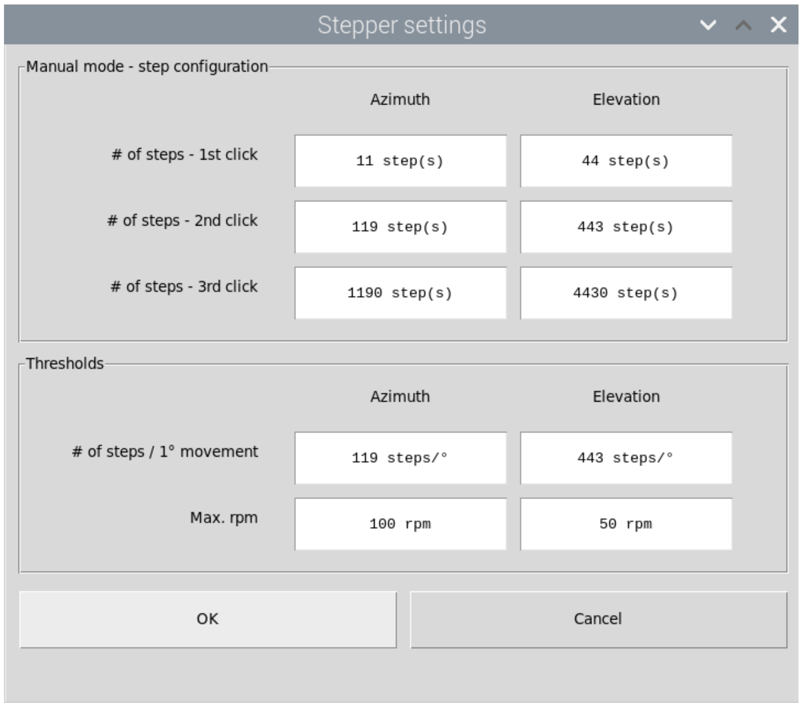


Update interval 120s

Figure A.18: Tracking interval settings and actual calibration statuses of the IMU are shown. With the default value of 120 seconds, the steppers are adjusted every two minutes according to the calculated azimuth and elevation direction using TLE data. Depending on actual calibration statuses (0 = not calibrated = highlighted red, 3 = fully calibrated = highlighted green) the background color of the value display for the desired sensor is changing.

Stepper motors

In the top region of the stepper settings, as shown in Figure A.19 it is possible to configure the manual modes, mentioned in Section A.7. For each axis, the number of steps per increment of the rotary encoder can be defined. In the bottom region the number of steps to adjust the desired axis 1° and the maximum speed in rotations per minute (rpm) can be configured. It is recommended to use the default values.



Manual mode - step configuration

	Azimuth	Elevation
# of steps - 1st click	11 step(s)	44 step(s)
# of steps - 2nd click	119 step(s)	443 step(s)
# of steps - 3rd click	1190 step(s)	4430 step(s)

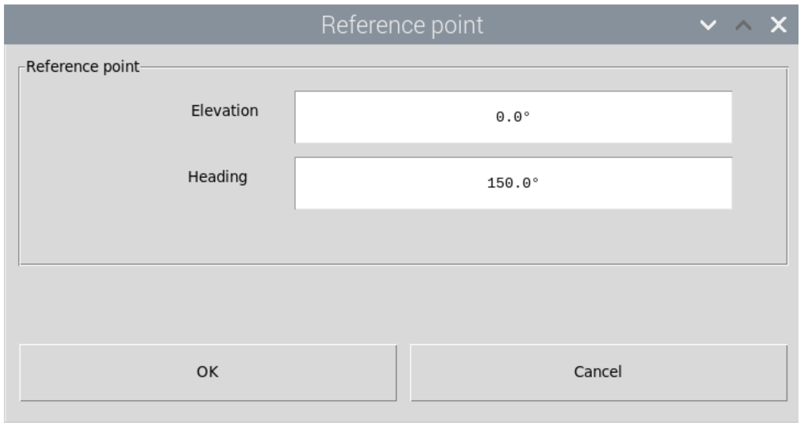
Thresholds

	Azimuth	Elevation
# of steps / 1° movement	119 steps/°	443 steps/°
Max. rpm	100 rpm	50 rpm

Figure A.19: Settings of the stepper drivers. The manual modes mentioned in Section A.7 can be configured in the *Manual mode* section. With the shown configuration, the stepper motor moves 11 steps in azimuth direction or 44 steps in elevation direction, depending on the switch position in the first mode. By clicking on the rotary encoder button, the mode changes from 1→2, 2→3, 3→1 and so on.

A.8.4 Reference Point

Right after start up, default values are visualized for azimuth and elevation. To define a well-known starting position, the actual angles for azimuth and elevation need to be configured. The configuration screen is shown in Figure A.20. For azimuth it is recommended to move the axis parallel to the evaluated reference plane, e.g. wall of a building, as mentioned in Section A.3. To evaluate the actual value for the elevation axis, a protractor can be used.



Reference point

Elevation 0.0°

Heading 150.0°

Figure A.20: Reference values for heading (true azimuth, AZ') and elevation (EL') are shown. Actual position data is evaluated by counting steps. For this reason, reference values must be configured, before operating with the pico terminal.

A.9 Safety regulations

The stepper motors are connected with a belt drive to the desired axes. It is obvious that a safety distance must be maintained during operation and neither the motors nor other parts of the drive may be touched. To stop the motors, the ENABLE/DISABLE button must be disabled. In manual mode using the manual control panel the steppers are disabled if the switch is in middle position. This behavior can also be used in automatic mode. If something is going badly wrong, switching from the middle position to position 1 or position 2 and back to middle position (switching back and forth) will also stop the motors.

Appendix B

Source code

The main program sequences of the control unit are explained in the following sections. Code for the visualization as well as the subroutines for converting data or downloading TLE data are not considered.

B.0.1 Main program

The main program is divided into two scripts. One for visualization and the second one with the program sequence, which is covered in this section.

GPIO

The GPIO is initialized immediately after starting the script. The code is shown in Listing B.1. These GPIO pins are used to control the stepper motors.

Listing B.1: Code snippet of the GPIO initialization

```
### stepper driver GPIO pins
GPIO_ENA = 16
GPIO_DIR_AZIMUTH = 20
GPIO_STEP_AZIMUTH = 21
GPIO_DIR_ELEVATION = 5
GPIO_STEP_ELEVATION = 6
GPIO_AZ_MAN = 13
GPIO_EL_MAN = 26

### stepper driver GPIO setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_ENA, GPIO.OUT)
GPIO.setup(GPIO_DIR_AZIMUTH, GPIO.OUT)
GPIO.setup(GPIO_STEP_AZIMUTH, GPIO.OUT)
GPIO.setup(GPIO_DIR_ELEVATION, GPIO.OUT)
GPIO.setup(GPIO_STEP_ELEVATION, GPIO.OUT)
GPIO.setup(GPIO_AZ_MAN, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
GPIO.setup(GPIO_EL_MAN, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

### stepper driver GPIO -> set outputs
```

```
GPIO.output(GPIO_ENA, GPIO.HIGH)
GPIO.output(GPIO_DIR_AZIMUTH, GPIO.LOW)
GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
GPIO.output(GPIO_DIR_ELEVATION, GPIO.LOW)
GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
```

Stepper motors

In Listing B.2 the stepper motor sequence is shown. Depending on the number of steps to move and the speed limit, the interval between the individual step impulses is varied.

Listing B.2: Code snippet of the stepper motor control sequence. The number of steps to move, a speed limit and the direction are considered. It is also possible to enable soft start and stop for gentler adjustments.

```
def move_stepper(self, GPIO_selection, steps_2_move, max_rpm,
   ↪ direction, azimuth_soft_limit = 1):
    #print("GPIO"+str(GPIO_selection) + " - rpm_max = " + str(
       ↪ max_rpm) + " - step(s) to move = " + str(steps_2_move))

    ## no movement, if direction is 0 or None
    if direction == 0 or direction == None:
        return

    if GPIO_selection == "AZIMUTH":
        if self.azimuth_active == 1:
            return
        else:
            self.azimuth_active = 1

    if GPIO_selection == "ELEVATION":
        if self.elevation_active == 1:
            return
        else:
            self.elevation_active = 1

    ## change azimuth direction, if limits are reached
    ## e.g.: current AZ = 340  , desired change in AZ = +90   -->
       ↪ 340   + 90   > 360   --> change stepper direction --> 340
       ↪    - (360   - 90   ) = 70   --> angle to move = 270   in
       ↪ opposite direction instead of 90
    if GPIO_selection == "AZIMUTH" and azimuth_soft_limit == 1 and
       ↪ direction > 0 and self.stepper_position_azimuth +
       ↪ steps_2_move >= self.azimuth_upper_position_degree * self
       ↪ .steps_per_degree_az:
        angle_2_move = steps_2_move * self.angle_per_step_azimuth
        steps_2_move = (360 - angle_2_move) * self.
           ↪ steps_per_degree_az
        direction = -1
        #print(steps_2_move, angle_2_move, direction)
    elif GPIO_selection == "AZIMUTH" and azimuth_soft_limit == 1
       ↪ and direction < 0 and self.stepper_position_azimuth -
       ↪ steps_2_move < self.azimuth_lower_position_degree * self.
       ↪ steps_per_degree_az:
        angle_2_move = steps_2_move * self.angle_per_step_azimuth
        steps_2_move = (360 - angle_2_move) * self.
           ↪ steps_per_degree_az
        direction = 1
        #print(steps_2_move, angle_2_move, direction)


    ## lock movement of EL axis in negative direction, if limit
       ↪ switch is active
    if GPIO_selection == "ELEVATION" and direction < 0 and (self.
       ↪ gpiobtnLimitEl.readPin() == 1 or self.
       ↪ elevation_lower_limit):
        self.elevation_active = 0
        return


    ## slowly increase of stepper speed (by decreasing delay time
       ↪ between pulses)
    ## delay depends on rpm_factor_percent if more or equal than
       ↪ 400 steps of movement are necessary
    ## fixed delay for movements with less than 400 steps
    rpm_factor_percent = 2

    for i in range(int(steps_2_move/2)):
        if self.stepper_state == 0:
            break

        if i < 200 and i % 4 == 0 and rpm_factor_percent < 100:
            rpm_factor_percent = rpm_factor_percent + 2

        if GPIO_selection == "ELEVATION":
            if direction > 0 and self.stepper_position_elevation <
               ↪ (self.elevation_upper_position_degree * self.
               ↪ steps_per_degree_el):
                GPIO.output(GPIO_DIR_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation + 1
            elif direction < 0 and self.stepper_position_elevation
               ↪ > (self.elevation_lower_position_degree * self.
               ↪ steps_per_degree_el) and not self.
               ↪ elevation_lower_limit:
                GPIO.output(GPIO_DIR_ELEVATION, GPIO.LOW)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation - 1
            else:
                break

        if GPIO_selection == "AZIMUTH":
```

```
            if direction > 0 and self.stepper_position_azimuth + 1
               ↪ <= (self.azimuth_upper_position_degree * self.
               ↪ steps_per_degree_az):
                GPIO.output(GPIO_DIR_AZIMUTH, GPIO.HIGH)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.HIGH)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth + 1
            elif direction < 0 and (azimuth_soft_limit == 0 or (
               ↪ azimuth_soft_limit == 1 and self.
               ↪ stepper_position_azimuth > (self.
               ↪ azimuth_lower_position_degree * self.
               ↪ steps_per_degree_az))):
                GPIO.output(GPIO_DIR_AZIMUTH, GPIO.LOW)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.HIGH)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth - 1
            else:
                break

        if steps_2_move >= 400:
            delay = round(0.15/(max_rpm*rpm_factor_percent/100),4)
        else:
            delay = round(0.15/(max_rpm*20/100),4)

        time.sleep(delay)

    ## slowly decrease of stepper speed (by increasing delay time
       ↪ between pulses)
    ## delay depends on rpm_factor_percent if more or equal than
       ↪ 400 steps of movement are necessary
    ## fixed delay for movements with less than 400 steps
    rpm_factor_percent = 100

    for i in reversed(range(int(steps_2_move/2))):
        if self.stepper_state == 0:
            break

        if i < 200 and i % 4 == 0 and rpm_factor_percent > 2:
            rpm_factor_percent = rpm_factor_percent - 2

        if GPIO_selection == "ELEVATION":
            if direction > 0 and self.stepper_position_elevation <
               ↪ (self.elevation_upper_position_degree * self.
               ↪ steps_per_degree_el):
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation + 1
            elif direction < 0 and self.stepper_position_elevation
               ↪ > (self.elevation_lower_position_degree * self.
               ↪ steps_per_degree_el) and not self.
               ↪ elevation_lower_limit:
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation - 1
            else:
                break

        if GPIO_selection == "AZIMUTH":
            if direction > 0 and self.stepper_position_azimuth + 1
               ↪ <= (self.azimuth_upper_position_degree * self.
               ↪ steps_per_degree_az):
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.HIGH)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth + 1
            #elif direction < 0 and self.stepper_position_azimuth >
               ↪  (self.azimuth_lower_position_degree * self.
               ↪ steps_per_degree_az):
            elif direction < 0 and (azimuth_soft_limit == 0 or (
               ↪ azimuth_soft_limit == 1 and self.
               ↪ stepper_position_azimuth > (self.
               ↪ azimuth_lower_position_degree * self.
               ↪ steps_per_degree_az))):
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.HIGH)
                GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth - 1
            else:
                break

        if steps_2_move >= 400:
            delay = round(0.15/(max_rpm*rpm_factor_percent/100),4)
        else:
            delay = round(0.15/(max_rpm*20/100),4)

        time.sleep(delay)

    ## correction step for odd number of steps
    if steps_2_move % 2 != 0 and self.stepper_state == 1:
        if GPIO_selection == "ELEVATION":
            if direction > 0 and self.stepper_position_elevation <
               ↪ (self.elevation_upper_position_degree * self.
               ↪ steps_per_degree_el):
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation + 1
            elif direction < 0 and self.stepper_position_elevation
               ↪ > (self.elevation_lower_position_degree * self.
               ↪ steps_per_degree_el):
```

```
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.HIGH)
                GPIO.output(GPIO_STEP_ELEVATION, GPIO.LOW)
                self.stepper_position_elevation = self.
                   ↪ stepper_position_elevation - 1

        if GPIO_selection == "AZIMUTH":
            GPIO.output(GPIO_STEP_AZIMUTH, GPIO.HIGH)
            GPIO.output(GPIO_STEP_AZIMUTH, GPIO.LOW)
            if direction > 0 and self.stepper_position_azimuth + 1
               ↪ < (self.azimuth_upper_position_degree * self.
               ↪ steps_per_degree_az):
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth + 1
            elif direction < 0 and self.stepper_position_azimuth >
               ↪ (self.azimuth_lower_position_degree * self.
               ↪ steps_per_degree_az):
                self.stepper_position_azimuth = self.
                   ↪ stepper_position_azimuth - 1


    if GPIO_selection == "AZIMUTH":
        self.azimuth_active = 0

    if GPIO_selection == "ELEVATION":
        self.elevation_active = 0

    return
```

Satellite pointing

To point a satellite with the automatic pointing function, the desired TLE data set must be loaded. Valid GPS data are also required to run the sequence shown in Listing B.3.

Listing B.3: Code snippet of the automatic pointing function. Valid TLE and GPS data are required for the function call.

```
def buttonTlePointSatClick(self):

    h0 = 350
    az = 0
    el = 0

    try:
        lon, lat, ran, az, el, sk = tle.calcSatellitePosition(self.
           ↪ gps_datetime, self.latitude_deg, self.longitude_deg,
           ↪ h0, self.tle_name, self.tle_line_1, self.tle_line_2)

        EL_prime_old = self.stepper_position_elevation * self.
           ↪ angle_per_step_elevation
        AZ_prime_old = self.stepper_position_azimuth * self.
           ↪ angle_per_step_azimuth
        skew = self.stepper_position_skew * self.
           ↪ angle_per_step_skew
        EL_prime_correction = calc.calcElevationCorrection(self.phi
           ↪ , el, skew, EL_prime_old)
        EL_prime_new = EL_prime_old + EL_prime_correction

        AZ_prime_correction = az - (AZ_prime_old + calc.
           ↪ calcAzimuthCorrection(self.phi, EL_prime_new, skew))
        AZ_prime_new = AZ_prime_old + AZ_prime_correction

        AZ_move_steps = abs(int(AZ_prime_correction / self.
           ↪ angle_per_step_azimuth))
        EL_move_steps = abs(int(EL_prime_correction / self.
           ↪ angle_per_step_elevation))

        ## AZ stepper direction for azimuth movement
        if AZ_prime_correction > 0:
            azimuth_direction = 1
        elif AZ_prime_correction < 0:
            azimuth_direction = -1
        else:
            azimuth_direction = 0

        ## EL stepper direction for elevation movement
        if EL_prime_correction > 0:
            elevation_direction = 1
        elif EL_prime_correction < 0:
            elevation_direction = -1
        else:
            elevation_direction = 0

        ## Start threads
        print(datetime.datetime.now(datetime.timezone.utc).strftime
           ↪ ("%H:%M:%S"), "AZIMUTH", AZ_move_steps,
           ↪ azimuth_direction)
        print(datetime.datetime.now(datetime.timezone.utc).strftime
           ↪ ("%H:%M:%S"), "ELEVATION", EL_move_steps,
           ↪ elevation_direction)
        print("-----")
        sys.stdout.flush()

        az_thread = threading.Thread(target=move_stepper, args=[
           ↪ self, "AZIMUTH", AZ_move_steps, self.max_rpm_az,
           ↪ azimuth_direction])
        el_thread = threading.Thread(target=move_stepper, args=[
           ↪ self, "ELEVATION", EL_move_steps, self.max_rpm_el,
           ↪ elevation_direction])
        az_thread.start()
        el_thread.start()

    except Exception as e:
            print("TLE␣-␣exception:␣" + str(e))
```

B.0.2 Calculation of azimuth and elevation using TLE data

When pointing a satellite with the automatic pointing function, a calculation of azimuth and elevation is required. This calculation is done with the sequence shown in Listing B.4. The desired TLE data set and the GPS position of the observer, in this case the pico terminal, are required for valid results.

Listing B.4: Code snippet to calculate azimuth and elevation when pointing a satellite.

```
import ephem
import datetime
import time
import pymap3d as pm
from math import *


def calcSatellitePosition(timestamp, lat0, lon0, h0, tle_name,
   ↪ tle_line1, tle_line2):
    satellite = ephem.readtle(tle_name, tle_line1, tle_line2)
    observer = ephem.Observer()
    observer.lon, observer.lat = str(lon0), str(lat0)
    observer.date = timestamp

    satellite.compute(observer)

    lon = degrees(satellite.sublong)
    lat = degrees(satellite.sublat)
    ran = satellite.range
    az = degrees(satellite.az)
    el = degrees(satellite.alt)
    sk = None

    return lon, lat, ran, az, el, sk
```

B.0.3 Alignment correction calculation

Due to the special geometry of the antenna unit, the effective antenna alignment depends on the configured skew angle. In Listing B.5 different functions are shown to convert the effective elevation angle into the elevation angle of the rotary axis and vice versa. It is also possible to evaluate the azimuth offset angle.

Listing B.5: Code snippet for alignment correction.

```
import math
import numpy as np

###############################################################
#   azimuth correction angle @ given skew and elevation angles
#
#   PAR phi          -> offset of the parabolic antenna
#   PAR EL_prime     -> elevation angle of the rotary (skew) axis
#   PAR skew         -> skew angle of the rotary axis
#                       (starting 0   clockwise @ zenith)
#
#   RET AZ_correction_angle
#                    -> azimuth correction angle
#                    -> AZ_effective_new = AZ_effective_prev
#                                       + AZ_correction_angle
#
###############################################################
def calcAzimuthCorrection(phi, EL_prime, skew):

    EL_hat = EL_prime + math.degrees( np.arctan(np.tan(math.radians
       ↪ (phi)) * np.cos(math.radians(skew))))
    AZ_numerator = np.tan(math.radians(phi)) * np.sin(math.radians(
       ↪ skew))
    AZ_denominator = math.sqrt(1 + np.tan(math.radians(phi))**2 *
       ↪ np.cos(math.radians(skew))**2) * np.cos(math.radians(
       ↪ EL_hat))
    AZ_correction_angle = math.degrees(np.arctan2(AZ_numerator,
       ↪ AZ_denominator))

    return AZ_correction_angle

###############################################################
#   effective elevation angle @ given skew and elevation angles
#
#   PAR phi          -> offset of the parabolic antenna
#   PAR EL_prime     -> elevation angle of the rotary (skew) axis
#   PAR skew         -> skew angle of the rotary axis
#                       (starting 0   clockwise @ zenith)
#
#   RET EL_effective    -> effective elevation angle of the beam
#
###############################################################
def calcEffectiveElevation(phi, EL_prime, skew):


    EL_hat = EL_prime + math.degrees( np.arctan(np.tan(math.radians
       ↪ (phi)) * np.cos(math.radians(skew))))
    EL_numerator = math.sqrt((1+(np.tan(math.radians(phi)))**2 *(np
       ↪ .cos(math.radians(skew)))**2) *(np.cos(math.radians(
       ↪ EL_hat)))**2 +(np.tan(math.radians(phi))**2 * (np.sin(
       ↪ math.radians(skew)))**2))
    EL_denominator = 1 / np.cos(math.radians(phi))
    EL_effective = math.degrees(np.arccos(EL_numerator /
       ↪ EL_denominator))

    return EL_effective


###############################################################
#   elevation angle of the rotary (skew) axis @ given skew and
#   effective elevation angles
```

```
#
#   PAR phi              -> offset of the parabolic antenna
#   PAR EL_effective     -> effective elevation angle of the beam
#   PAR skew             -> skew angle of the rotary axis
#                           (starting 0   clockwise @ zenith)
#
#   RET EL_prime         -> elevation angle of the
#                           rotary (skew) axis
#
################################################################
def calcRotaryElevation(phi, EL_effective, skew):

    EL_hat_numerator = np.cos(math.radians(EL_effective))**2  / np.
       ↪ cos(math.radians(phi))**2 - np.tan(math.radians(phi))**2
       ↪ * np.sin(math.radians(skew))**2
    EL_hat_denominator = 1 + np.tan(math.radians(phi))**2  * np.cos
       ↪ (math.radians(skew))**2
    EL_hat = math.degrees(np.arccos(math.sqrt(EL_hat_numerator /
       ↪ EL_hat_denominator)))
    EL_prime_hat = math.degrees(np.arctan(np.tan(math.radians(phi))
       ↪  * np.cos(math.radians(skew))))
    EL_prime = EL_hat - EL_prime_hat

    return EL_prime

################################################################
#   elevation correction angle of the rotary (skew) axis
#   @ given skew and effective elevation angles
#
#   PAR phi              -> offset of the parabolic antenna
#   PAR EL_effective     -> effective elevation angle of
#                           the beam
#   PAR skew             -> skew angle of the rotary axis
#                           (starting 0   clockwise @ zenith)
#   PAR EL_prime_start   -> starting elevation angle of the
#                           rotary (skew) axis
#
#   RET EL_prime_correction     -> elevation correction angle
#                                  of the rotary (skew) axis
#
################################################################
def calcElevationCorrection(phi, EL_effective, skew, EL_prime_start
   ↪ ):

    EL_prime_end = calcRotaryElevation(phi, EL_effective, skew)
    EL_prime_correction = EL_prime_end - EL_prime_start

    return EL_prime_correction


def calcStepsToMove(angle_per_step, angle_to_move):
    steps_to_move = angle_to_move / angle_per_step

    return steps_to_move

def calcAngleFromSteps(angle_per_step, stepper_position):

    stepper_position_angle = stepper_position * angle_per_step

    return stepper_position_angle
```

B.0.4 GPS

To evaluate the position of the transportable pico terminal, the code sequence in Listing B.6 is used to read the GPS data from the GPS receiver.

Listing B.6: Code snippet of the GPS sequence.

```
import time
from threading import Thread
import glob
import datetime

class GpsDateTimePosition(Thread):
    def __init__(self, path):
        Thread.__init__(self)
        self.path = path
        self.gps_time = None
        self.gps_date = None
        self.gps_longitude = None
        self.gps_latitude = None
        self.gps_longitude_degree = None
        self.gps_latitude_degree = None
        self.timestamp = None
        self.gps_valid = False
        self.state = 1

    def run (self):
        while True:
            if self.state == 1:
                                ## get GPS file path
                gps_path = ''
                for file_paths in glob.glob(self.path):
                    gps_path = file_paths

                try:
                                    ## set values to 'None'
                    self.gps_time = None
                    self.gps_date = None
                    self.gps_longitude = None
                    self.gps_latitude = None

                                    ## open GPS file
```

```
file = open(gps_path, 'r', encoding= '
   ↪ unicode_escape')

## read lines from file and format data as long
   ↪  as GPS file path is valid
for line in file:
    data=line.rstrip()
    gps_time_tmp = ''

    if data.split(',')[0] in ("$GPZDA", "$GPGGA
       ↪ ", "$GPGRS", "$GPGST", "$GPGBS"):
        gps_time_tmp = data.split(',')[1]
        self.gps_time = gps_time_tmp[0:2] + ":"
           ↪  + gps_time_tmp[2:4] + ":" +
           ↪ gps_time_tmp[4:6]

    elif data.split(',')[0] == "$GPZDA":
        self.gps_date = data.split(',')[4] + '-
           ↪ ' + data.split(',')[3] + '-' +
           ↪ data.split(',')[2]

    elif data.split(',')[0] == "$GPRMC":
        self.gps_date = '20' + '{:02d}'.format(
           ↪ int((data.split(',')[9])[4:])) +
           ↪ '-{:02d}'.format(int((data.split(
           ↪ ',')[9])[2:4])) + '-{:02d}'.
           ↪ format(int((data.split(',')[9])
           ↪ [:2]))


    if data.split(',')[0] == "$GPGLL":
        lat = data.split(',')[1]
        lat_dir = data.split(',')[2]
        lon = data.split(',')[3]
        lon_dir = data.split(',')[4]
        status = data.split(',')[6]

        lat_deg = int(float(lat) // 100)
        lat_min_tmp = float(lat) % 100
        lat_min = int(lat_min_tmp // 1)
        lat_sec = round(((lat_min_tmp % 1) *
           ↪ 60),1)

        lon_deg = int(float(lon) // 100)
        lon_min_tmp = float(lon) % 100
        lon_min = int(lon_min_tmp // 1)
        lon_sec = round(((lon_min_tmp % 1) *
           ↪ 60),1)

        self.gps_latitude = str(lat_deg) + u'\
           ↪ xb0' + "␣" + str(lat_min) + "'␣"
           ↪ + str(lat_sec)  + "''␣" + lat_dir
```

```
                            self.gps_longitude = str(lon_deg) + u'\
                               ↪ xb0' + "␣" + str(lon_min) + "'␣"
                               ↪ + str(lon_sec)  + "''␣" + lon_dir
                            self.gps_longitude_degree = round(float
                               ↪ (lon_deg + (lon_min_tmp / 60)),
                               ↪ 6)
                            self.gps_latitude_degree = round(float(
                               ↪ lat_deg + (lat_min_tmp / 60)), 6)

                            if status == 'A':
                                self.gps_valid = True
                            elif status == 'V':
                                self.gps_valid = False
                            else:
                                self.gps_valid = None

                                        ## close file, if file path
                                           ↪  changed
                    file.close()

                ## exception handling
                except Exception as e:
                    print("GPS␣sensor␣not␣found:␣" + str(e))
                    self.gps_time = None
                    self.gps_date = None
                    self.gps_longitude = None
                    self.gps_latitude = None
                    if str(e).find("codec␣can't␣decode") >= 0:
                        pass
                    else:
                        self.state = -1

                                ## delay until next cycle starts
                time.sleep(5)


if __name__ == '__main__':
    try:
        ## Test sequence
        sensorGPS = GpsDateTimePosition('/dev/ttyACM*')
        sensorGPS.setName('SensorThread_GPS')
        sensorGPS.start()

        while True:
            time.sleep(1)
            print(sensorGPS.gps_date, sensorGPS.gps_time, "LON:␣" +
               ↪  sensorGPS.gps_longitude, "␣LAT:␣" + sensorGPS.
               ↪ gps_latitude)
            print(sensorGPS.gps_date, sensorGPS.gps_time, "LON:␣" +
               ↪  str(sensorGPS.gps_longitude_degree), "␣LAT:␣" +
               ↪ str(sensorGPS.gps_latitude_degree))
            print("")
    except Exception as e:
        print(e)
```

B.0.5 IMU

In order to visualize the alignment of the feed arm, the sensor data of the IMU are evaluated using the sequence shown in Listing B.7.

Listing B.7: Code snippet of the IMU sequence.

```
import time
from os import system
import os
import RPi.GPIO as GPIO
import math
import numpy as np
from datetime import datetime
from threading import Thread
import sys
import board
import busio
import adafruit_bno055

try:
    i2c = busio.I2C(board.SCL, board.SDA)
except Exception as e:
    print("9-axis␣sensor␣exception␣1:␣" + str(e))

class Imu(Thread):
    def __init__(self,address):

        Thread.__init__(self)

        self.i2c_error = False

        self.magX = None    # raw x value from MAGNETOMETER
        self.magY = None    # raw y value from MAGNETOMETER
        self.magZ = None    # raw z value from MAGNETOMETER

        self.accelX = None  # raw x value from ACCELEROMETER
        self.accelY = None  # raw y value from ACCELEROMETER
        self.accelZ = None  # raw z value from ACCELEROMETER

        self.gyroX = None   # raw x value from GYROSCOPE
        self.gyroY = None   # raw y value from GYROSCOPE
        self.gyroZ = None   # raw z value from GYROSCOPE

        self.pitch = None   # pitch angle theta around y-axis
        self.roll = None    # roll angle phi around x-axis
        self.yaw = None     # yaw angle psi (heading) around z-axis
        self.magCal = None        # calibration status of the
           ↪ MAGNETOMETER
        self.accelCal = None      # calibration status of the
           ↪ ACCELEROMETER
        self.gyroCal = None       # calibration status of the
           ↪ GYROSCOPE
        self.sysCal = None        # calibration status of the SYSTEM
        self.cal = None           # sensor calibrated

        try:
            self.bno055 = adafruit_bno055.BNO055_I2C(i2c)
            self.bno055.mode = adafruit_bno055.NDOF_MODE

        except Exception as e:
            self.i2c_error = True
            print("9-axis␣sensor␣exception␣2:␣" + str(e))

    def run (self):

        init = True

        while self.i2c_error == False:
            try:

                self.sysCal, self.gyroCal, self.accelCal, self.
                   ↪ magCal = self.bno055.calibration_status
                self.cal = self.bno055.calibrated

                self.yaw_bno055, self.roll_bno055, self.
                   ↪ pitch_bno055 = self.bno055.euler
                self.accelX, self.accelY, self.accelZ = self.bno055
                   ↪ .acceleration
                self.magX, self.magY, self.magZ = self.bno055.
                   ↪ magnetic
                self.gyroX, self.gyroY, self.gyroZ = self.bno055.
                   ↪ gyro

            except Exception as e:
                print("9-axis␣sensor␣exception:␣" + str(e))
```