BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2021 Bc. Tomáš Horeličan

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

POSITION MEASUREMENT OF MOVING OBJECTS USING A ROBOTIC TOTAL STATION
MĚŘENÍ POZICE POHYBUJÍCÍCH SE OBJEKTŮ POMOCÍ ROBOTICKÉ TOTÁLNÍ STANICE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Tomáš Horeličan**
AUTOR PRÁCE

SUPERVISOR — **Ing. Tomáš Jílek, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Cybernetics, Control and Measurements**

Department of Control and Instrumentation

Student: Bc. Tomáš Horeličan ***ID:*** 195315

Year of study: 2 ***Academic year:*** 2020/21

TITLE OF THESIS:

Position measurement of moving objects using a robotic total station

INSTRUCTION:

The aim of the thesis is to evaluate the parameters and behavior of a robotic total station during continuous position measurement of moving objects. In relation to the total station, the author is expected to assess the properties or parameters that are not defined by the manufacturer (such as time delays and non-synchronous data output). The intended application is UAV navigation inside buildings.

1. Search for and investigate the principles of robotic total stations, together with different methods for verifying their parameters and behavior in the measurement of moving objects.
2. Explore the control elements and perform basic hands-on operation of the Trimble S7 and S9 total stations.
3. Design and implement experiments to define the main aspects of the device's behavior in moving object measurement.
4. Process and evaluate the experimentally obtained data.
5. Suggest options for suppressing the identified negative properties.
6. Discuss the possibilities of using robotic total stations for UAV navigation inside buildings.

RECOMMENDED LITERATURE:

ROBERTS, Craig & BOORER, Peter. Kinematic positioning using a robotic total station as applied to small-scale UAVs. Journal of Spatial Science. 2016, 61(1), 29-45. ISSN 1449-8596. Available at: doi:10.1080/14498596.2015.1068232

Date of project specification: 8.2.2021 ***Deadline for submission:*** 17.5.2021

Supervisor: Ing. Tomáš Jílek, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
Chair of study program board

ABSTRACT

This thesis deals with an emerging unconventional use case for modern theodolites, also known as *Robotic Total Stations* (RTSs), as a tracking and guidance system, by measuring the precise position of a dynamically moving object. This applies especially to situations where conventional positioning systems such as GNSS are insufficient or completely unavailable. A kinematically acquired position from a constantly tracking RTS could be used for real-time autonomous navigation of small *Unmanned Aerial Vehicles* (UAVs), essentially providing them with a reference coordinate frame and an immediate position within it. A significant part of this thesis is dedicated to the design and realization of suitable experiments, which would estimate the reliability of this precise position measurement in a precise moment in time. The S7 and S9 series stations from the Trimble company were evaluated and an S9 HP RTS, which provides a continuous measuring frequency of up to 10 Hz was then predominantly used for experiments. The local time of a TSC7 controller, interfacing with the RTS, was being synchronized through *Precision Time Protocol* (PTP) with the local time of a Raspberry Pi mini-computer, which then provided a reference measurement of an object's true position in time. The conclusion summarizes all obtained results.

KEYWORDS

theodolite, robotic total station, tacheometry, position measurement, kinematic mode, navigation

ABSTRAKT

Práca sa zaoberá objavujúcim sa nekonvenčným využitím moderných teodolitov, tiež známych ako Robotické Totálne Stanice (RTS), ako sledovací a navádzací systém, určením presnej pozície dynamicky sa pohybujúceho objektu. To sa týka najmä situácií, kde nie je možné využiť konvenčné polohovacie systémy akým je napríklad GNSS. Kinematicky určená poloha objektu kontinuálne sledujúcou RTS môže byť následne v reálnom čase využitá pre autonómnu navigáciu malých bezpilotných leteckých prostriedkov (UAV) poskytnutím referenčného súradnicového systému a okamžitej polohy v ňom. Podstatná časť práce je venovaná návrhu a realizácii vhodných experimentov, ktoré overia spoľahlivosť určovania presnej polohy objektu v presnom časovom okamžiku. Boli preverené stanice série S7 a S9 od spoločnosti Trimble a stanica S9 HP, ktorá disponuje frekvenciou kontinuálneho merania až do 10 Hz bola napokon využitá pre experimenty. Lokálny čas ovládacieho panelu TSC7, ktorý zabezpečuje komunikáciu so stanicou bol pomocou protokolu PTP synchronizovaný s lokálnym časom mini-počítača Raspberry Pi, ktorý následne poskytoval referenčné meranie skutočnej polohy objektu v čase. V závere sú zhrnuté výsledky experimentov.

KĽÚČOVÉ SLOVÁ

teodolit, robotická totálna stanica, tachymetria, meranie polohy, kinematický režim, navigácia

ROZŠÍRENÝ ABSTRAKT

Úvod

Teodolitické systémy merajúce uhly a vzdialenosti sú štandardným nástrojom v geodetických a konštrukčných odvetviach. Moderné prístroje, známe ako *Robotické Totálne Stanice* (z ang. *Robotic Total Station* (RTS)) umožňujú okrem bežných meraní aj vykresľovanie geometrických tvarov v priestore, vytváranie detailných 3D máp alebo určovanie presných polôh objektov v určenom súradnicovom systéme. Robotizované stanice sú zároveň schopné sledovať pohybujúce sa objekty vo svojom lokálnom priestore. Je to práve táto schopnosť, ktorá ponúka nové možnosti ich využitia aj mimo štandardné aplikácie. Ponúka sa napríklad riešenie navigácie pre malé bezpilotné letecké prostriedky (ang. *Unmanned Aerial Vehicle* (UAV)) v interiéroch budov alebo v situáciách, kedy nie je možné využiť štandardné navigačné prostriedky, akým je napríklad GNSS. Takáto aplikácia však nie je primárnym účelom totálnych staníc a je nutné overenie ich parametrov pre zhodnotenie reálnej aplikovateľnosti. V tejto práci boli testované stanice *S7* a *S9 HP* od spoločnosti *Trimble*.

Analýza problému a návrh vhodnej koncepcie

Pre účely navigácie pohybujúceho sa objektu je nutné aby systém čo najpresnejšie vystihoval jeho pozíciu v priestore a čase. Prvotná rešerše ukázala hlavné problémy vyskytujúce sa pri kinematickom meraní polohy pomocou RTS. Jeden problém spočíva v nejasnosti pri stanovovaní časových značiek, ktoré stanica posiela spolu s polohovými údajmi. Nie je úplne známy okamžik vyhotovenia časového údaju, čo môže viesť k nesprávne určenej polohe. Ďalej, keďže RTS pozostáva z dvoch primárnych častí, ktoré samostatne merajú uhol a vzdialenosť s rôznymi periódami, výsledná poloha určená ich kombináciou nemusí správne odzrkadľovať skutočnú polohu objektu ak sa táto poloha v čase mení (viď. Obr. 2.4). Preferované sú vysoké frekvencie merania a maximálna rýchlosť prenosu dát z RTS. Základné koncepcie experimentu spočívajú v paralelne prebiehajúcom referenčnom meraní, ktoré určuje objektívne správnu polohu alebo čas pohybujúceho sa objektu počas merania s RTS. Zvolená koncepcia hlavného experimentu spočíva v presne vytýčenej konkrétnej polohe, ktorú objekt pri svojom pohybe opakovateľne dosahuje. Referenčné meranie udáva vždy skutočný čas, v ktorom je táto poloha dosiahnutá, a ten je potom možné porovnať s časom, ktorý tejto polohe pripisuje RTS. Týmto sa stanoví celkové oneskorenie určenej polohy, ktoré už v sebe môže zahŕňať aj interné nepresnosti totálnej stanice a je teoreticky možné ho integrovať do navigačnej úlohy.

Realizácia experimentov a testov

Pri počiatočných testoch bolo zistené, že stanica *S7* dosahuje frekvencie merania maximálne 2.5 Hz a stanica *S9 HP* podporuje zvýšenú frekvenciu až do 10 Hz (viď obrázky 4.3 a 4.4). Pre pohyb bola použitá dostupná rotačná konštrukcia, ktorá svojim umiestnením vykonávala tlmené kmity (viď Obr. 5.5) a bola stabilne upevnená tak aby ich čo najmenej ovplyvnila.

Referenčné meranie vykonával senzor so svetelnou bránou (*Panasonic EX-Z11*, viď Obr. 5.2) nastavenou na dĺžku 5 mm, ktorá spolu s 2 mm úzkou tyčinkou (Obr. 5.3) presne vytyčovala pokojovú polohu kyvadla. Referenčný čas bol tejto polohe priraďovaný počítačom *Raspberry Pi* s operačným systémom *Raspberry Pi OS Lite*. Merací program bol napísaný v jazyku C s knižnicou *PiGpio*.

Komunikácia s RTS bola vykonávaná pomocou ovládacieho panelu TSC7 so systémom Windows 10 cez štandardnú aplikáciu *Trimble Access*. Stanica bola umiestnená v niekoľkých rôznych vzdialenostiach od kyvadla kolmo na rovinu kmitov a kontinuálne sledovala hranol umiestnený na ramene kyvadla (Obr. 5.6, vpravo). Počiatok súradnicového systému bol vždy od samotnej polohy stanice a nulový horizontálny uhol bol definovaný s kyvadlom v pokojovej polohe.

Merací program v hlavnom vlákne cez systém prerušení priraďoval časovú značku každému prechodu cez bránu (pokojovú polohu kyvadla). Dátový výstup z RTS (viď Výp. 5.1) prichádzajúci na UART vstup RPi bol paralelne zaznamenávaný druhým vláknom a zároveň bola každému prichádzajúcemu bodu merania priradená časová značka. Lokálny čas panelu TSC7 bol pomocou protokolu PTP (z ang. *Precision Time Protocol*) počas každého merania synchronizovaný s lokálnym časom RPi a priebeh tejto synchronizácie bol zaznamenávaný v log súboroch.

Stanovené časové značky prechodov cez bránu mohli byť porovnané so značkami z RTS a tým určené celkové oneskorenie merania od skutočného momentu pokojovej polohy k tomu, ktorý tejto polohe priradila stanica. Ďalšie oneskorenie vyplýva z rozdielu času, kedy je niektorý stanicou zmeraný bod prístupný na koncovom zariadení a opäť časom ktorý mu bol stanicou priradený, teda oneskorenie prenosu (viď Obr. 4.2). Okrem tlmených kmitov bol vykonaný experiment s manuálnym náhodným pohybom kyvadla a dodatočné testy, ktoré mohli určiť vplyv natočenia hranolu a dynamické obmedzenia pre pohyb.

Výsledky vykonaných experimentov

Časové značky získane z RTS boli interpolované k nulovým (pokojovým) polohám tak, aby bol určený ich presný časový okamžik. Všetky časové hodnoty z RTS a RPi boli prevedené na spoločný základ tak by mohli byť následne vzájomne porovnávané.

Na obrázkoch 6.13, 6.14 a 6.15 sú súhrnné grafy pre jednotlivé sady experimentov. Je vidieť, že vo všetkých prípadoch sa oneskorenia merania polohy pohybujú okolo 110 ms. Mierne vyššie hodnoty na Obrázku 6.13 boli zmerané pri konfigurácii, kde bola veľmi presne dosiahnutá kolmá orientácia RTS, avšak nie je zrejmé či práve to bola príčina vyšších hodnôt. Na Obrázku 6.16 je celkový histogram oneskorení pre všetky zobrazené sady. Obrázok 6.17 potom ukazuje oneskorenie prenosu, blížiace sa k 26 ms. Obrázky 6.21, 6.22 a 6.23 zobrazujú obdobné výsledky pre experiment s manuálnym pohybom ramena.

Na obrázku 6.19 je vidieť zdanlivú koreláciu medzi frekvenciou časových značiek určených pri UART vstupe RPi a frekvenciou vyplývajúcou zo samotných časových značiek z RTS. Je možné domnievať sa, že stanica priraďuje tieto časy buď neskôr až v momente odoslania dát na výstup alebo sú značky priradené v momente merania ale okamžite odoslané na výstup.

Pri väčších počiatočných rozkmitoch dochádzalo k úbytku dát z RTS (viď Obr. 6.28). Na Obrázku 6.29 je vidieť stratu dát so zväčšujúcim sa počiatočným rozkmitom a s tým súvisiacou rýchlosťou hranolu. Tá bola počítaná zo samotných nepresných dát z RTS a tým značne obmedzené možné závery. Na Obrázku 6.30 je vidieť značný nárast oneskorenia merania pri veľmi malých rýchlostiach, kedy dochádzalo k zastaveniu kmitania. Presnejšie experimenty s nezávislým určením rýchlosti by mohli spoľahlivejšie stanoviť tieto závislosti.

Na obrázkoch 6.25, 6.26 a 6.27 je možné vidieť vplyv horizontálnej rotácie hranolu. Tieto výchylky pravdepodobne spôsobuje prechod medzi jednotlivými reflexnými elementmi hranolu, avšak rozsiahlejšie experimenty sú potrebne pre robustnejšie určenie závislostí.

Záver

V tejto práci bolo vykonaných niekoľko experimentov pre overenie parametrov merania polohy pohybujúceho sa objektu pomocou RTS.

V úvodných dvoch kapitolách práce boli najprv popísané základné princípy merania s teodolitmi a načrtnutá problematika merania s dynamickým pohybom objektu. Následne boli popísané parametre dostupných totálnych staníc a ich príslušenstva využívaného v tejto práci.

V štvrtej a piatej kapitole bol vykonaný podrobnejší rozbor danej problematiky, stanovené parametre a požiadavky, a zvolená koncepcia pre hlavný experiment. Následne boli vybrané konkrétne komponenty, ich rozloženie a konfigurácia pre daný experiment. Bol tiež uvedený podrobný postup jeho realizácie.

Šiesta kapitola poskytla podrobný postup spracovania získaných dát a ich finálnu prezentáciu a zhodnotenie.

HORELIČAN, Tomáš. *Position measurement of moving objects using a robotic total station*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation, 2021, 121 p. Master's Thesis. Advised by Ing. Tomáš Jílek, Ph.D.

Author's Declaration

Author:	Bc. Tomáš Horeličan
Author's ID:	195315
Paper type:	Master's Thesis
Academic year:	2020/21
Topic:	Position measurement of moving objects using a robotic total station

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno 16.5.2021 .....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would, hereby, like to express my gratitude to my master's thesis supervisor Ing. Tomáš Jílek, Ph.D. for his expert mentoring, readily available consultations, patience and incentivising suggestions concerning the thesis.

Brno 16.5.2021 .....

author's signature*

*The author signs only in the printed version.

Contents

List of Figures

List of Tables

Listings

Introduction

This thesis builds upon experience with and previous work on my semestral thesis, essentially providing a comprehensive and finalized document with many enhancements and additions [1]. The initial results were also published in [2].

Modern measurement systems based on the principles of theodolite position measurements are being greatly utilized within the geodetic or construction industries. Rapid progress in the development of these systems has led to a shift from just simple devices that could measure angles in the horizontal and vertical axes towards fully automatic total stations that can provide both angle and distance measurements, create three-dimensional maps of the environment or track and follow moving objects. These devices are more generally referred to as *Robotic Total Stations* (RTSs). The term *total station* refers to a combination of an angle measuring element (classic theodolite) with a unit measuring the distance from the equipment into a single integrated device. This allows the total station to determine an absolute (*total*) position of virtually any point in its surroundings. Elements inside contemporary total stations are exclusively electronic as opposed to the earlier much more rudimentary mechanic theodolites. In addition, modern total stations are being equipped with servomotors, enabling their fully automatic and remote operation.

With the advent of robotized automatic total stations, new forms of their utilization are emerging, even in non-standard fields. The automatic control of a total station brings new functionalities, such as the mentioned following of moving objects. In a standard scenario, this would enable only a single operator to completely operate the total station and therefore perform geodetic measurements much more effectively. However, an interesting byproduct of this functionality is the ability to now have a continual stream of a moving object's positional data, which is gathered from a device generally capable of very precise measurements. This capability can be used in many other applications, where such position information is required.

One such application is, for example, using a total station as a tracking and guidance system in environments, where conventional positioning systems such as *Global Navigation Satellite Systems* (GNSS) are limited or completely unavailable. This mostly includes building interiors, where reliably obtaining a stable signal from positioning satellites is very difficult, or other places, which do not have sufficient satellite coverage. This positional data acquired from the total station can be subsequently used for real-time autonomous navigation of small *Unmanned Aerial Vehicles* (UAVs) inside buildings or even during GNSS signal loss.

For this data to be really useful, however, its credibility and precision in space and time relating to the true object location must be guaranteed. In a standard total station operation, a position measurement is performed statically, on a stationary

object, where high position stability and accuracy are guaranteed. However, a dynamic application, such as drone navigation, requires continual position estimation, where stopping the movement and waiting for measurement is simply not feasible. This raises many questions and complications, where possible discrepancies between the true object's position at a given time and a position assigned to it for this time instant as a result of a measurement can occur.

1 Thesis objectives and motivation

This issue has been the topic of several research papers over the years [10, 30] - [18, 34]. The aim of this thesis is to evaluate the usability of robotic total stations from the *Trimble Inc.* company, available at the *Department of Control and Instrumentation*, *i.e. Ústav automatizace a měřicí techniky* (UAMT), at the *Faculty of Electrical Engineering and Communication* (FEEC), *Brno University of Technology* (BUT) for kinematic position measurement. Overall, three total stations available, which are the S7, S9 and SX10 series. Within this master's thesis, the main focus and tests with namely the Trimble S7 and Trimble S9 HP have been performed. It has been found that the S7 series total station in the existing configuration does not provide measurement frequencies higher than 2.5 Hz. Following practical experiments have, therefore, been performed with the Trimble S9 HP, which supports frequencies of up to 10 Hz. The Trimble SX10 was not used in this thesis and might provide an interesting option for future research. A detailed description of the hardware used and all available total stations is provided in Chapter 3.

In order to determine the parameters of an RTS measurement of a moving object, it is necessary to firstly, in a suitable manner, define a real objective position in a specific time instant. Together with its timestamp, this position can be then compared with data acquired from the RTS. Since the monitored object is moving dynamically, a type and shape of motion that takes into account expected experiment results and its feasibility within the thesis need to be selected. Section 2.3 deals with the issue of dynamic RTS measurements in more detail and gives reasoning behind the selection of such a motion for the experiments.

Section 4.2 describes the structure and realization procedure of the experiment as well as the selection of suitable components for the reference position measurement. Parameters and requirements, which the experiment must meet in order to ensure correct RTS parameter determination for the purposes mentioned in the introduction are discussed. In short, the expected result of the experiments is an objective determination of the deviation of data acquired by the RTS from a real (reference) position in an exact time instant. Subsequent evaluation of these results is described in the conclusion (see p. 100).

2 Position measurement with a total station

This chapter will familiarize the reader with the topic of measurement with a total station. A short overview of the historical evolution and elementary principles, and functionalities of total stations and theodolites will be explained. Several complications and issues with kinematic measurements will be outlined. Attention is focused mainly on problems concerning the subject of this thesis.

2.1 Historical background

Devices measuring angles, heights, distances or providing horizontal (level) and vertical (plumb) alignment functionalities have been known since the antic times. Notable are for example the roman *groma*, *chorobrates* or *dioptra*, each performing different specific operations. Several sources date the term *theodolite* to first start appearing no earlier than around the 16th century. Devices referred to by this name where usually simpler (in their capabilities, but certainly not in their complexity) single purpose instruments with a gradated scale, from which an angle value in the range of 0 to 360 degrees would be read out.

It was only later, when devices serving different independent functions started to be combined, as for example enhancing a telescope with angle measurements, a spirit level, compass or other elements. Reading out the angle measurements was usually accomplished by what is called a *vernier scale*, which enabled more precise measurements than regular scales. The theodolite, which was at first limited to rotation only in the horizontal plane, was later enhanced to also rotate vertically. Due to their construction however, the ability for vertical rotation was largely limited (see Fig. 2.1, left) and later in the 19th century, first devices that could perform full 360 degree rotations in the vertical started appearing in the United States. These were called *vernier transits* or simply *transits*. In the 1950s, the *Geodimeter* device provided a first application of *Electronic Distance Measurement* (EDM). Classical direct angle readings were soon phased out in favor of more advanced optical techniques. Apart from even more modern methods, the scale could for example be safely enclosed within the device and read out by an optical guiding and magnifying apparatus, which allowed the addition of a *micro scale* with a substantially greater resolution and again more precise measurements. An important aspect in any kind of a theodolite measurement was correct synchronization of the zero value of a scale with what is called a *horizontal* or *vertical circle*.

In the later half of the 20th century, theodolites performing electronic angle readings with photoelectric sensors and rotary encoders first started appearing. With this advancement now, two previously independent and unrelated measuring devices,

Fig. 2.1: A visual comparison of an old and modern theodolite. Left: taken from [3], Right: taken from [4].

the EDM and the new electronic theodolite, could be combined into a single unit called a *total station*. Moving further, advancements in the microelectronic industry enabled the total station to be equipped with an internal microprocessor, extending its functionalities even beyond regular measurements. It was then able to perform on-device data processing and memory storage, so the range of possible kinds of obtainable data was extended to, for example, projections of measured positions into a chosen coordinate frame. The data could be stored and kept on the device for later processing or exported to another instrument at any time. External sources of information could also now be connected to the total station, providing for instance the current date or localization data from GNSS, which enabled unification of the measured local positional data with geodetic coordinate frames. The initial calibration of the device was substantially simplified and many procedures were now done automatically. The latest step in the evolution of total stations was equipping the rotary mechanisms with servo motors at the beginning of the 21st century. While before that, manual precise orienting by mechanical gearings was necessary, now even this operation could be done automatically. It enabled further expansion of possible features by adding automatic object tracking. This fully automated device is known by the term *Robotic Total Station* (RTS), an example of which is shown in Figure 2.1 on the right. The field which specializes in geodetic measurements, mainly utilizing total stations, is also called *tacheometry* [5].

2.2 Fundamental measurement principles

Typically in a tacheometric measurement, an angle is given in the units of *gon* (*gradian*). This unit is formally defined as nine-tenths of a degree, meaning it divides a whole circumference into 400 equal parts, where 90 ° = 100 gon. A clear advantage of such a formalization is very efficient, fast and intuitive mental reconstruction of orientation. Figure 2.2 illustrates the basic concept of theodolite angle measurements. As was stated above, the device can rotate in the horizontal and vertical planes and the measured (horizontal and vertical) angles can be either read out directly or optically from a certain scale, or provided by electronic measuring equipment. Modern RTSs provide an easy way of setting up the reference, or zero, *azimuth* (0,000 gon, as shown in Fig. 2.2), from which the horizontal angles are measured either clockwise or counterclockwise. The reference for vertical angles is usually fixed, either pointing up (towards the *zenith*) or down (towards *nadir*), which are shown in Figure 2.3. Specifically for the *Trimble S7* and *S9 HP*, the reference will always be at the zenith [6], [7]. Servo drives are another mentioned enhancement of modern RTSs. They enable automatic rotation around both, the vertical and horizontal axes. Precise angle measurements are nowadays usually done electronically, by incremental rotary encoders for example, which provides higher flexibility and overall enhancement of the station's functionalities.

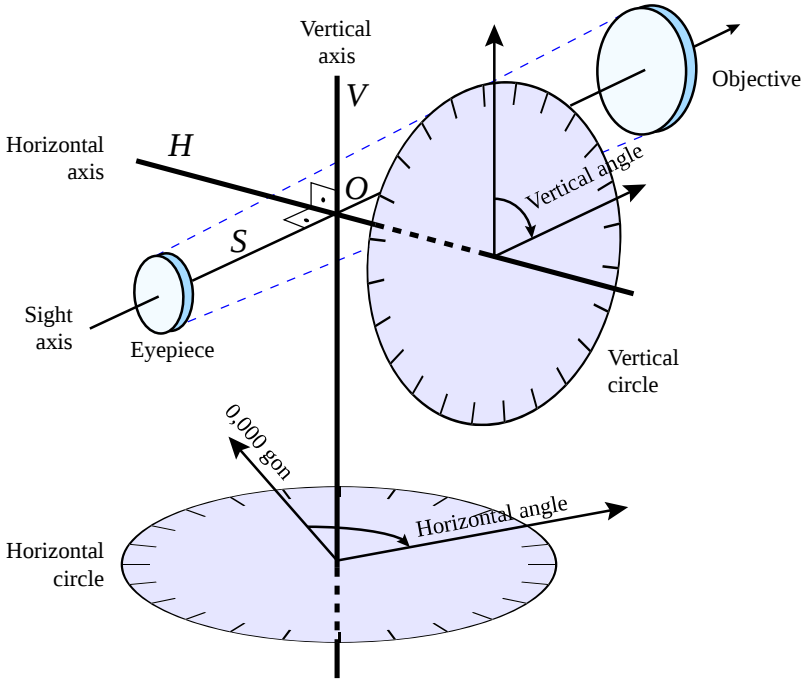


Fig. 2.2: Theodolite axes and angles representation. Taken from [8].

Distance (also referred to as *slope distance*) measurement with EDM units is

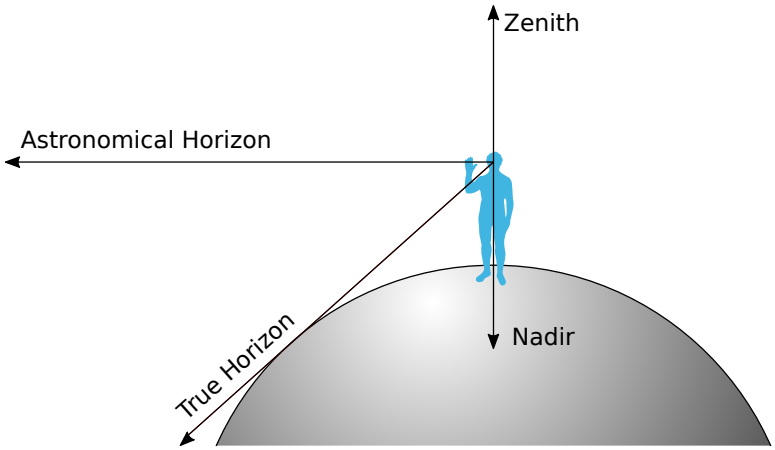


Fig. 2.3: Zenith and Nadir representation. Taken from [9].

based on transmitting and receiving signals of a certain spectrum of radiation and comparing their difference. Commonly, these signals lie within the optical or microwave band. Another common practice is emitting a focused laser beam. The main difference between all of these types of signal sources is that while for a microwave ray, separate devices for receiving and emitting of the light have to be used, the optical light as well as laser beams allow for a single device to serve both as an emitter and a receiver. Hence, the complexity and number of necessary components can be minimized. In this case an emitting device sends out a ray (or beam), which is then reflected from the measured object back to the same device which also receives it and a phase shift between the sent and received signal is calculated. Alternatively a *Time of Flight* (TOF) measurement principle is used, where a time difference between the moment of dispatch and the moment of acquisition is calculated instead. Measurement can be done either by reflecting the light from an opaque surface or a specialized light-reflecting prism, which directs the light back towards the device can be used. Modern total stations, such as the *Trimble S7* or *S9 HP*, usually utilize a laser light source and enable both operation modes, providing the user with an option to choose between a *Direct Reflex* (DR) or a reflexive prism target-based measurement method.

RTSs equipped with an internal microprocessor can perform further data processing and control of its individual components, continuous monitoring of its state and error compensation, processing of external GNSS localization data or numerous other functions. Apart from standard angle and distance measurement functions, a modern RTS can, for instance, measure and lay out different geometrical shapes in 3D space, measure areas and volumes, create dense precise 3D maps of its surroundings or provide positions in a coordinate frame [5].

2.3 Measurement with dynamic object movement

As was outlined in Section 2.1, latest modern RTSs can provide not only static measurements of non-moving objects, but also dynamically track the position of an object during its motion. This is done using a matrix image sensor, which picks up deviations of the incoming light from a central position, from which angle corrections can be made. For a standard use case, the benefits are clear. In the construction industry, for example, work can be carried out much more effectively by requiring only a single person to operate the total station for all of its measurements. The RTS is first locket onto a prism target and the operator can then freely relocate the prism to any desired position without requiring a second operator to reorient the RTS accordingly. After a satisfactory prism placement, the operator can initiate its precise position measurement from a distance.

Since the RTS can also provide a continuous output data stream while it's following the prism, a notion of utilizing this data for other means arises. However, as can be evident, this process employs fast measurements in rapid succession and the final accuracy of such acquired positions can be degraded, compared to precise static measurements which can take up to several seconds. *Trimble* total stations can perform measurements in a *Tracking* (TRK) mode with sub-second periods of up to 100 ms (for a more detailed description see Chapter 3). In some cases, a 50 ms measurement period is available with a *Trimble Universal Total Station* (UTS) (see Sec. 4.1).

The measuring process is in its principle the same as with a static measurement. An often discussed problem concerning this rapid data acquisition was time synchronization between the individual RTS components. Namely, the distance measuring EDM unit and the angle measuring component. The problem lies within a different time period required by each component to perform its own measurement. Typically, an angle can be acquired much sooner then a distance value, therefore the approximated position of an object can divert slightly from a real one, as the resulting position is estimated from both of these measurements. This limitation does not really effect static measurements, as there will be no expected change in position within this time interval, however for fast moving objects it can cause deviations from their true positions. An illustrative example of this effect can be sen in Figure 2.4.

Several studies performed by Lenda *et al.* [10, 11] show that with a favorable placement of the RTS and by using modern total stations with high sampling periods, these deviations can be reduced to the sub-millimeter regime. In these cases they also present favorably higher positioning accuracies when compared to GNSS-based *Real Time Kinematic* (RTK) methods. An interesting solution proposed by Kerekes *et*

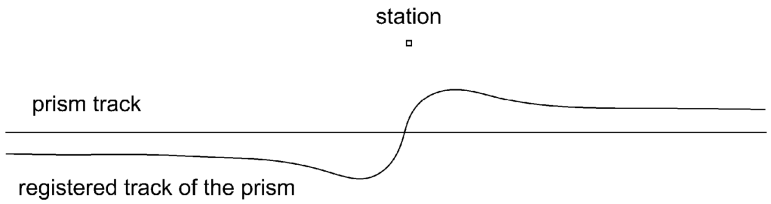


Fig. 2.4: Position measurement deviation. Taken from [10].

al. [12] lies within avoiding the slower distance measurement completely and instead utilizing a network of two or possibly more RTSs, using the faster and arguably more precise angle measurements only. Their method was also able to achieve accuracies to within several millimeters and allowed to nearly double the acquisition rate of measurements.

The desired *AutoLock* tracking functionality is only available with prism-based measurements [6], [7]. Since light used to perform the distance measurement has to be reflected and directed back towards the instrument, the main evident limiting factor of using these devices for positioning is the necessity of direct line of sight at all times. An otherwise useful feature of modern RTSs is a predictive tracking capability in the event of a lost line of sight, where the station predicts the next position of a target based on its previous movement. This can be useful if the navigated drone, for example, flies behind an obstacle and its trajectory is not expected to change during this time. However, it might ultimately lead to a worsened tracking ability, when such changes in direction of motion are expected or for generally unpredictable movements. A more consistent proposed method for dealing with the loss of line of sight is presented by Kerekes *et al.* [13], where a network of multiple total stations tracking the same prism from different angles can provide sufficient redundancy.

Another factor influencing the quality and accuracy of kinematic object tracking and position measurement is the type of prism used. It is clear, that standard circular types of prisms (Fig. 2.5, left) will not be well suited for multi-directional tracking, as they can reflect light rays only from a specific orientation range. When mounted on a drone, a correcting gyroscope mechanism might perhaps be used, however this increases complexity and a much simpler solution presents itself by using specialized 360 ° prisms (Fig. 2.5, right). These contain reflexive elements around their whole perimeter, effectively making them usable from any orientation and angle. Conversely, as is shown by Lackner *et al.* [14], the circular type typically provides higher angle and distance accuracies, especially when aligned correctly. This is attributed to the neighbouring elements influencing the RTS locking mechanism at certain orientations. However, maintaining perfect constant alignment, in

particular during dynamic motion of the target might be difficult to achieve and modern prism designs are shown provide acceptable accuracies. The biggest influence appears to be moments when the prism orientation facilitates changes between these elements. Active prisms providing better recognition of a target in highly reflexive environments might also be useful, however their benefits might be negated their increased weight. A passive *Trimble VX/S 360* prism, which can be seen in Figure 2.5 on the right was used exclusively in all experiments within this thesis.

Fig. 2.5: An example of a circular (left) and 360 (right) prism. Left: taken from [19].

During a continuous measurement, the RTS assigns a timestamp to each final measured position. This is also done during static measurements, however the implication of these values is, more or less, only informative, as they bear little to no relevance when the position of the object doesn't change. For a kinematic measurement however, these timestamps gain much higher significance, since at any instant in time the position of the object can be different. Especially for the purposes of real-time navigation of small UAVs, the necessity of acquiring not only an accurate position, but also an accurate time in which the object occupied that position arises. It is evident that an incorrect or shifted timestamp assigned to a specific position might also cause errors in navigation and an incorrect localization despite having an accurate absolute position. Since for typical industry applications, these time values bear little significance in measurement precision, exact parameters of how and when they are attached are not always evident and definite. Manufacturers might not include the details in their specifications and it is also apparent that given the different measuring periods of the individual RTS components, these values cannot, in principle, be absolutely definite. The question, in which part of the measuring process was this timestamp created, arises. The theodolite can, for example, acquire angles with a high frequency and subsequently use several samples for calculating an average or other suitable value. The same can, in principle, be done by the EDM for distances as well. These raw angle and distance values might in addition be further processed by the internal processor for calculating other data values, such as

the object's coordinates in a given reference frame. It is not clear at which point in this process a timestamp was associated with the final data point. Deeper investigations into the internal timing parameters of total stations can be seen in research by Stempfhuber *et al.* [15], Böniger *et al.* [16], Gojcic *et al.* [17] or Thalmann *et al.* [18], however these deal with specific scenarios and RTS models.

3 Trimble total stations and accessories

In this chapter, known parameters and functionalities of the available total stations will be described. The reader will be acquainted with their basic operation sequences and procedures for performing measurements. The parameters and operation descriptions will focus mainly on the functionalities, which are significant to the aims of this thesis.

3.1 Main measurement options

Two measurement modes are available, which are *Standard* (STD) and *Tracking* (TRK). These mainly affect how the EDM instrument operates. Angles are always being averaged during the course of a distance measurement. STD mode provides a single precise measurement, which can be accepted by the operator or discarded. The distance value itself can also be averaged over several measurements for higher accuracy. A continuous data output is only available while in TRK mode, where the EDM continually provides measurements as soon as they are finished. This also means that the dynamic target following functionality only works with TRK mode. Since automatic prism tracking is the main focus of this thesis, TRK will be the preferred mode used during experiments.

Available target options are either a reflexive prism, where a specific type with its parameters can be selected or a DR target, which is a general non-reflexive surface. Both measurement modes are available with either target, however the *AutoLock* function, which enables object tracking can only be used with reflexive prisms [7, 6].

3.2 Trimble S7 Robotic Total Station

The *S7* series (see Fig. 3.1) total stations come equipped with *MagDrive* servo motors and absolute encoders with diametrical reading and with precisions given at either 0.3 mgon, 0.6 mgon, 1.0 mgon or 1.5 mgon. The stations provide automatic level compensations with a precision of 0.15 mgon. The maximum given angular velocity of the servos is 115 °/s.

The given measuring accuracy for distance values in TRK mode with a reflexive prism is 4 mm + 2 ppm with a measurement period of 400 ms. Non-reflexive DR targets are not considered in this thesis. The usable distance range is from 20 cm up to 2,5 km. However, when using the *AutoLock* mode with a passive reflexive prism, the maximum gets limited to 500 m - 700 m and pointing precision is given to be better than 2 mm, at a 200 m distance. A pulsed laser diode (905 nm) with

Fig. 3.1: Trimble S7 Robotic Total Station. Taken from [20].

a horizontal beam divergence of 2 cm/50 m and a vertical divergence of 4 cm/50 m is used as a light source for the EDM.

A digital image sensor with a resolution of 2048 x 1536 pixels can be used as well. Power is provided by a 6.5 Ah battery or an external 12 V power supply. Connectivity is provided through a 2.4 GHz antenna, *Bluetooth* or a direct serial connection. The internal COM port can also be used for data output [21]. Table 3.1 summarizes the relevant or used parameters.

Angles	**Sensor type**	absolute encoder
	Accuracy	0.3 mgon, 0.6 mgon, 1.0 mgon or 1.5 mgon
Distances	**Sensor type**	905 nm pulsed laser diode
	Accuracy	4 mm + 2 ppm
	Precision	$<$ 2 mm @ 200 m
	Range	from 20 cm to 500 m - 700 m
Max. angular velocity		115 °/s
Measurement period		400 ms
Measurement modes		TRK, AutoLock

Tab. 3.1: Relevant parameters of the Trimble S7.

3.3 Trimble S9 HP Robotic Total Station

The *Trimble S9 HP* (Fig. 3.2) RTS utilizes the same *MagDrive* servo motors as the *S7*. Exact types of angle sensors are not specified, but it is assumed that they are also a similar type of absolute encoders. Given angle measurement accuracies are 0,15 mgon or 0,3 mgon. Level compensator accuracy is not specified. Maximal servo angular velocity is again 115 °/s.

Fig. 3.2: Trimble S9 HP Robotic Total Station. Taken from [4].

When using a reflexive prism target in the TRK mode, the given distance measurement accuracy is 5 mm + 2 ppm and a standard measuring period of 400 ms. A valid distance range for measurements with a passive reflexive prism is 150 cm to 3 km. Again, the *AutoLock* functionality limits the maximum effective usable range to only 500 m - 700 m with a pointing precision of less than 2 mm at a 200 m distance. A 660 nm laser diode with a horizontal and vertical beam divergence of 4 cm/100 m is used as the EDM light source. An increased measurement frequency of 10 Hz when in TRK mode is also mentioned within the documentation.

The *S9 HP* also is equipped with a digital image sensor with a 2048 x 1536 resolution. Power is again provided by a 6.5 Ah battery or an external 12 V power supply. A 2.4 GHz antenna, *Bluetooth* or an internal serial port provide connectivity. The S9 HP can also deliver data output through the internal COM port [22]. Table 3.2 again summarizes the relevant or used parameters.

Angles	**Sensor type**	absolute encoder assumed
	Accuracy	0.15 mgon or 0.3 mgon
Distances	**Sensor type**	660 nm laser diode
	Accuracy	5 mm + 2 ppm
	Precision	$<$ 2 mm @ 200 m
	Range	from 150 cm to 500 m - 700 m
Max. angular velocity		115 °/s
Measurement period		400 ms, 100 ms available
Measurement modes		TRK, AutoLock

Tab. 3.2: Relevant parameters of the Trimble S9 HP.

3.4 Trimble SX10 Scanning Total Station

This station belongs to the *Scanning Total Station* (STS) category with its high speed 3D scanning capability. It is, again, equipped with similar *MagDrive* servo motors and absolute encoders with diametrical readings as the *S7* and *S9 HP*. The given angle measurement accuracy is 0.3 mgon. Its centered dual-axis level compensator accuracy is 0.15 mgon. Maximal servo angular velocity is not specified.

Fig. 3.3: Trimble SX10 Scanning Total Station. Taken from [23].

With reflexive prism targets in the TRK mode, the given distance measurement accuracy is 2 mm + 1.5 ppm. The measuring period for the TRK mode is not specified, however a scanning rate of 26.6 kHz is achievable. The available range for standard reflexive prism measurements is from 1 m up to 5.5 km. The maximum is again limited, when used in conjunction with the *AutoLock* functionality, to 300 m

- 700 m, for 360 prisms specifically. The pointing precision is 0.88 mm at a 50 m distance. The EDM light source is a 1550 nm laser diode.

It contains three image cameras with a 2592 x 1944 pixel resolution. A 6.5 Ah battery or an external power supply providing 11.1 V can be used. Wi-Fi or 2.4 GHz radio and a USB are stated as communication options. A serial COM interface, similar to that of the *S7* and *S9 HP* is visibly present [24]. The relevant parameters are summarized in Table 3.3.

Angles	**Sensor type**	absolute encoder
	Accuracy	0.3 mgon
Distances	**Sensor type**	1550 nm pulsed laser diode
	Accuracy	2 mm + 1.5 ppm
	Precision	0.88 mm @ 50 m
	Range	from 100 cm to 300 m - 700 m
Scanning rate		26.6 kHz
Measurement modes		TRK, AutoLock

Tab. 3.3: Relevant parameters of the Trimble SX10.

This station was, unfortunately, not used or tested during this thesis, however several interesting factors might make it a promising option for any future research. Most notably, as is mentioned by Lenda *et al.* [11], its high measurement rates might be beneficially utilized for the navigation task. This might also point to a generally better inter-component synchronization and more accurate timing capabilities. The high measurement rate, however, is connected specifically to the 3D scanning functionality and might not be utilizable for standard prism tracking. Nonetheless, custom solutions, which will be presented in Section 3.7 might enable full utilization of this capability and therefore it might be an interesting path for any future research.

3.5 TSC7 controller

The TSC7 is a handheld device used for remote operation of *Trimble* total stations. It has a 17.8 cm diameter, 1280 x 800 resolution touchscreen and a full keyboard with a numpad and navigation buttons, as can be seen in Figure 3.4.

Standard connectivity features such as Wi-Fi, *Bluetooth* or WWAN are available. Internally, it is equipped with an Intel Pentium 1.1 GHz *Apollo Lake* 64-bit quad core CPU, 8 GB of LPDDR4 RAM and an eMMC storage of 64 GB. It is powered

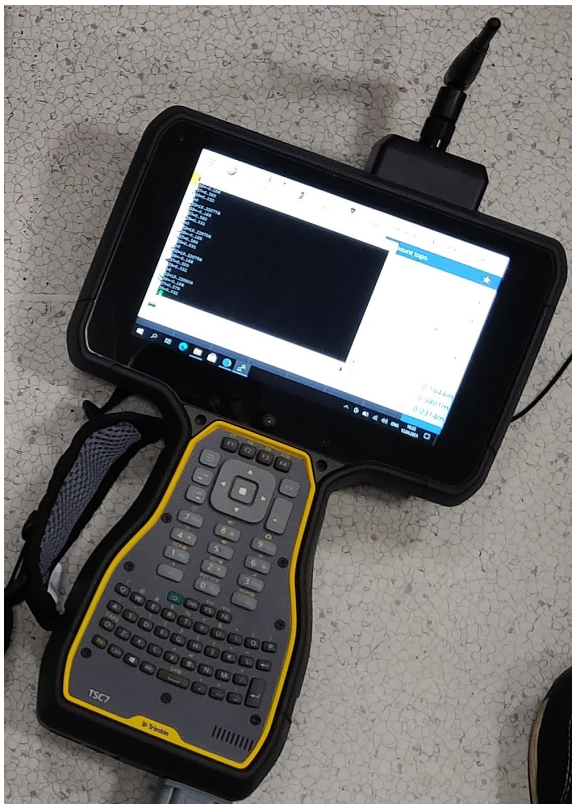


Fig. 3.4: TSC7 controller.

by two 3.1 Ah hot-swappable batteries or a 19 V/5 A charging power supply. The controller runs on a full-fledged Windows 10 Pro 64-bit operating system, essentially making it a fully independent handheld *Personal Computer* (PC). Communication and control of an RTS is managed by a *Trimble*-developed application called *Trimble Access*, that comes preinstalled with the device. Two external *EMPOWER* modules can be attached to extend the device's functionalities (see Chap. 3.7). The controller used had a radio antenna module, that was managing the connection to a total station, attached. A peripheral USB and an RS232 serial port can be used for data transfer [25].

3.6 Basic instrument operation for measurements

For remote RTS operation a TSC7 controller (see Sec. 3.5) and a 2.5 GHz radio connection is used. After powering on, the RTS waits for connections at a specified channel ID and frequency. For an initial connection, these parameters have to be set accordingly and saved in the *Trimble Access* application, after which all future connections are done automatically.

Once a radio link to the total station has been made, the user can proceed to

perform their measurements. Basic angle and distance data is available right away, however for additional values a coordinate system and a location of the RTS within it have to be defined. Several methods are available, nonetheless for the purposes of this thesis a basic *Station Setup* process was sufficient. If the RTS level compensator is out of tolerance, meaning the station is not leveled properly, a calibration screen with a graphical representation of spirit levels for the horizontal plane is displayed. After adjusting the level to acceptable limits, the level compensation function can be used. Afterwards, environmental quantities such as temperature and atmospheric pressure can be entered. After that, the user can move on to define the coordinate system.

First, a *Base Station* point has to be set. This point represents the location of the RTS within the coordinates. The *Base Station* point can be set explicitly with the x, y, z values or one of several methods, such as resection, can be used to calculate the location. Another value that can be set is the height of the point. This helps to easily define a ground level for the coordinates. The total station calculates all coordinate values using its distance and angle measurements, which means the measured height is going to be influenced by the actual height of the measuring equipment. This value is therefore used to compensate for the influence. If the value is set to zero, the origin of the vertical axis will be coincident with the optical axis.

After the *Base Station* point is set up, a *Back Sight* point has to be measured. This essentially defines the orientation of the defined coordinate system in the horizontal plane. This point is measured from the instrument, and its azimuth and height can be set explicitly. The height here serves the same purpose, only from the target point of view. If the *Back Sight* is, for example, a prism attached to a pole of certain length, but we want to ignore the pole, the height can be set to compensate for the pole's length. The azimuth can be set to define the orientation of the coordinates within a global existing frame.

After the user accepts the *Back Sight* point measurement, the total station is configured and ready for real measurements. Further *Back Sight* points for higher precision (see Sec. 5.3) or other topological points can be measured. An example of a layout after the setup procedure, not representing the actual experiment setup, can be seen in Figure A.1.

When the station icon (Fig. A.1 up) is clicked a menu with several RTS features is displayed. Here, one of the measurement modes described in Section 3.1 can be selected. TRK mode, which allows high frequency measurements and the *AutoLock* feature, which automatically locks onto a target within the station's field of view can be selected from this menu. The prism icon provides options for selecting a target type (see Sec. 3.1) and its height.

The last option relevant to the purposes of this thesis is data output. This can be found after clicking the three lines icon at the top left (Fig. A.1) and navigating to `Instrument Settings`. Three options for data output format (Fig. 5.8) are available, `GDM user defined`, `Pseudo NMEA GGA` or `GDM HA VA SD`. The *pseudo NMEA* is a format based on the equally named standard from the *National Marine Electronics Association* (NMEA), which is commonly used in navigational and maritime applications. Transmission parameters such as the used COM port, baud rate or flow control can also be specified. One important fact to keep in mind is that this screen must not be exited with the escape button, otherwise the transmission will be closed. Navigating to different screens while this one is opened in the background is of course possible.

3.7 Alternative operation options

Many studies, some of which were mentioned in Section 2.3 or will be talked about further in Section 4.1, performed their research using total stations developed by the *Leica Geosystems AG* company. A common trend was utilizing their available *Application Programming Interface* (API) for direct control of the RTS, called *Leica GeoCOM*. It is an ASCII-based communications protocol, which provides basic operation commands to, for example, directly query the EDM or theodolite instruments for distance or angle measurements. On a higher level, custom programs developed with C/C++, MS-VBA, Matlab or LabView can use this API for direct access and operation of the instrument. As it is presently understood, this system is licensed by *Leica* and needs to be acquired from the appropriate channels [26].

An investigation was conducted into similar possibilities for the available total stations from *Trimble*. Three possible third-party development solutions were found. First is the *Trimble EMPOWER* platform. This is an option that provides the ability to enhance the *Trimble* controllers (such as the TSC7) for different proprietary applications. Custom hardware modules and software programs can be developed to create new functionalities, such as adding *Near Field Communication* (NFC) communication capabilities. Access to the necessary *Software Development Kit* (SDK) needs to be first consulted and approved by the company [27]. This option is, however, not suited for the purposes stated in this thesis.

The second option is the *Trimble Access SDK*. This provides software developers access to the core of the *Trimble Access* application used to control their RTSs in order to enhance its functionalities or develop new custom solutions utilizing the API of the program. New workflows, *User Interface* (UI) elements or measurement routines can be developed. However, this still relies on the core application philosophy with an abstraction layer between the real RTS hardware. Access to this SDK,

again, needs to be acquired by the appropriate *Trimble* communication channels [28]. This option might still not be the best suited solution for the aims presented by this thesis.

The last and most promising option is called the *Trimble Precision SDK*. This seems to provide an API for direct access to the *Trimble* hardware, such as their total stations. Custom applications, not bound to any already existing solution, can be created and control of individual station interfaces is available. The API is designed specifically for development in the *Microsoft* Windows environment with C++ or C#. Thus any limitations stemming from the available *Trimble Access* application can be circumvented by a direct custom solution. Similarly as with the *Leica GeoCOM* interface, independent control of the distance an angle measuring components should be available. The SDK is provided through *Trimble*'s *Installation Manager* application and a necessary license must first be acquired from *Trimble* [29]. This option seems to be the most suitable for any RTS-based navigation solutions outlined in this thesis. Acquiring the necessary license was not achieved during the time frame of this thesis, therefore a practical implementation is not presented. However, it provides an interesting direction for any future research.

4 Problem analysis and concept design

This chapter will provide detailed clarification for the experiment methodology and design. The specific scenario concerning this thesis will be analyzed and requirements that need to be satisfied will be outlined. A suitable experiment structure, from the physical and hardware up to the software design will be specified. Each aspect will be further elaborated providing a full justification for the proposed experiment concept.

4.1 Analysis of the kinematic RTS measurement problem

Section 2.3 already provided an overview of the principles and setbacks of a an RTS measurement for dynamically moving objects. Here, a further dissection of the problem will be presented and specific experiment design properties will be stated.

In order to effectively utilize the continually acquired RTS data in real time, it is necessary for them to accurately reflect the real physical properties of the monitored object, namely position and time, and to have as little as possible diversions from these real properties. In other words, each measured position and its timestamp must describe the real location, in which the object was at that time as accurately and reliably as possible. Or in reverse logic the real time, in which the object was at that particular location.

Many research papers, now spanning over more than a decade have been dealing with the applicability of kinematic total station measurements and their parameters for several distinct applications. Lienhart *et al.* [30] discuss the potential of kinematic RTS measurements to be used for dynamic monitoring of vibrations of constructions, such as bridges. Several interesting parameters, which are also of interest for real time object tracking and navigation, notably a need for high data transfer rates and measuring frequency were discussed in this research.

Multiple factors can influence the requirements on the positional and temporal data, being for example the velocity and character of motion, distance from which the object is observed or time synchronization of different measuring components. Given, for example, the maximum angular velocity of the available total stations' servo motors, a maximal target motion speed can be estimated from equation 4.1

$$v_{\max} = \omega_{\max} \cdot r, \tag{4.1}$$

where

- $v_{\max}$ is the maximal prism velocity,

- $\omega_{\max} = 115\ ^\circ/\mathrm{s} = 2$ rad/s is the maximal angular velocity given by the *Trimble* documentation,
- r is the distance from the RTS.

It can be seen that the maximal prism velocity depends linearly on distance by a factor of two in the specific case of *Trimble S7* and *S9 HP* stations.

The impact of dissociation of measurements by different parts of the RTS is most evident in a dynamic scenario where the moving object can be in different positions when each instrument accomplishes its measurement. This was problematic especially in the early days with older hardware and insufficient inter-component synchronization. Stempfhuber *et al.* [15] and Lenda *et al.* [10, 11] describe these influences in detail and show acceptable results within a millimeter for modern total stations that have been developed in the 21st century. These studies however, have been performed with movement velocities of only up to 3 m/s, therefore applications requiring higher target speeds might necessitate more investigation before usage, as the expected errors might also be higher. The experiments were performed with a linear motion and deviations from expected lateral and vertical positions were observed.

Additional experiments with actual UAV navigation were also performed by Roberts *et al.* [31], Maxim *et al.* [32] and Hankus-Kubica *et al.* [33]. These again focused on positional deviations from expected reference trajectories.

The mentioned articles so far focused on positional data evaluation with the time aspect being controlled for. Gojcic *et al.* [17] and Thalmann *et al.* [18] perform deeper investigations into the temporal parameters of RTSs and present synchronization routines for different dynamic applications. Interesting and relevant insights into the internal workings of the measurements and time characteristics can be seen, however their results are aimed at specific scenarios and the exact details might also vary between different instruments available for this thesis.

Additionally, Stempfhuber *et al.* [15] and more recently Paraforos *et al.* [34] have performed experiments with *Trimbe* UTSs, which allow high frequency measurements of up to 20 Hz. A higher measurement rate is generally favorable for lowering of inter-component delays. A dependency of position errors on prism motion velocity is also demonstrated in [34], however velocities of only up to 1 m/s were tested. These total stations are considered to be at the highest development stage, offering even higher accuracies and, as stated, measurement rates. They are part of the *Trimble Heavy Industry* sub-brand. Their higher quality component design and construction might make them even more suitable for real time tracking and navigation than typical RTSs [35].

It is important to state that most of these groups performed experiments with *Leica* total stations and utilized the availability of their *GeoCOM* commands, which

provide a more direct control of the individual RTS components. Similar potential options were investigated in Section 3.7. Many research papers also present a higher positioning accuracy and precision compared to even RTK GNSS methods, when a suitable configuration is attained, which is favorable and further supports the aims of this thesis for RTS-based navigation.

Several notions arise from all of the above mentioned. First, a certain type of motion has to be defined, which ideally reflects the monitored factors. A reference providing the true quantity values needs to be established in order evaluate the quality of the RTS measurements. Finally, a relationship between the reference and the monitored quantities should be known in order to perform a valid comparison.

4.2 Experiment concepts and requirements

The performed experiment should therefore be able to safely quantify the reliability of the RTS data. Three main realization concepts are arising. Both of them incorporate two concurrently running measurements, that is the actual RTS and a reference measurement. A mutual relationship between these two processes needs to be accurately known in order to maximally isolate any undesired influences that the experiment might have on the observed parameters.

The first option would be apriori synchronization of measurement intervals between both processes. Either the acquisition times are exactly known and can be aligned for both the reference and RTS measurement or a quantifiable relationship exists between them so that measurement synchronization can be achieved. This way the time values for both measuring processes correspond to each other and a deviation of the RTS measured position data from the reference at each time instant can be objectively estimated. A similar concept of evaluating the positional data, whether by directly synchronizing the measurement times or by statistical fitting of the acquired trajectories is realized in [31], [33] and [34].

The second option is the opposite, that is the notion of perfectly known positions for both measuring processes, which are exactly coincident. These positions are know from the character and realization of motion and experiment. Therefore time deviations can be estimated from the values assigned by the RTS and the reference times, in which the object really occupied these positions. This provides a way of estimating measurement delays associated with the RTS as a whole.

Lastly, an experimental setup, which isolates the time data entirely can be realized. This incorporates an exactly defined linear track for motion with positions in the longitudinal direction being observed and lateral and vertical positions are expected to remain zero or constant. A comparison of deviations from zero can then be made with respect to the linear motion. A reference measurement is also

maintained and time data can, in principle, also still be evaluated. This type of experiment provides a way of evaluating the internal component synchronization delays and is shown in [15], [10] and [11].

4.2.1 Requirements on temporal data

An important aspect of the experiment realization is a correct formalization of time parameters and isolation of the experiment chain influences from the observed RTS parameters. The whole chain can be separated into an RTS measurement process and a reference measurement process.

Factors influencing the RTS measurement process can be the station's delay, data processing by the RTS's measurement software or data propagation delay from the RTS to a logging device. These are the parameters that are to be quantified. In order to utilize the obtained data for other purposes in real time, the overall delay from the moment of a physical position measurement to the moment of data availability at the end point is of concern. It might also be beneficial to have the knowledge of when (or at which point of the process) a timestamp was associated with a specific position measurement. These facts are not always evident in advance and should be estimated in a suitable manner.

The reference measurement process provides greater flexibility and control over individual components, hence it can be characterized more precisely. The reaction time of the end sensor performing the physical measurement and how fast it provides the results on output, both need to be considered. Similarly, the propagation delay from the sensor to an evaluating device is also important. Since the timestamp association is fully under our control, the parameters of the evaluating device and its effect on this association have to be taken into account.

Several requirements on the experiment chain follow from the above mentioned. The RTS measurement requirements are given largely by the intended final usage. Given the advertised sampling period of 100 ms or 400 ms (see Chap. 3), possible data loss and a potential delay increase from lower data propagation rates, it is desirable for the RTS timestamps to have acceptably low deviations from the true time, or for the deviations to be deterministic and exactly known for the whole application period. With typical velocities of small quadrocopters being in the range of ones to tens of meters per second (or mm/ms) [36], a delay of hundreds of milliseconds creates a position error of tens to hundreds of centimeters. Greater delay values could potentially render the navigation task difficult to accomplish, expanding the position error beyond the size of the aircraft itself. For the reference measurement, the process should reflect the true time of the position as accurately as possible. The delay from a physical detection to a timestamp association should be mini-

mal so as not to significantly skew the determined RTS measurement parameters. This delay should then, ideally, not exceed hundreds of microseconds or possibly be exactly deterministic so that it can be safely accounted for during processing. If these requirements are met, the reference data can be accepted as the true object's properties from the RTS's point of view.

Last but not least, the whole experiment chain should be considered as well, meaning the time data of both processes should be mutually synchronized and share a common frame or a direct relationship between them has to be known in order to be able to process both data sets in a comparative way.

4.2.2 Requirements on positional data

The requirements are again evaluated from two parts, the RTS and reference measurement processes. Position estimation is the main task in this point of view, ergo a certain accuracy of its measurement has to be adhered to. Again, considering the typical sizes of small quadrocopters being in tens of centimeters, the error in position estimation with an RTS should ideally be in millimeters, possibly units of centimeters. The accuracy of position estimation of a total station is a parameter explicitly given by the manufacturer and further verifications of its credibility are not a primary concern of this thesis. Additionally, for dynamically moving targets a much higher effect on the overall position estimation accuracy is expected to come from time and synchronization inaccuracies.

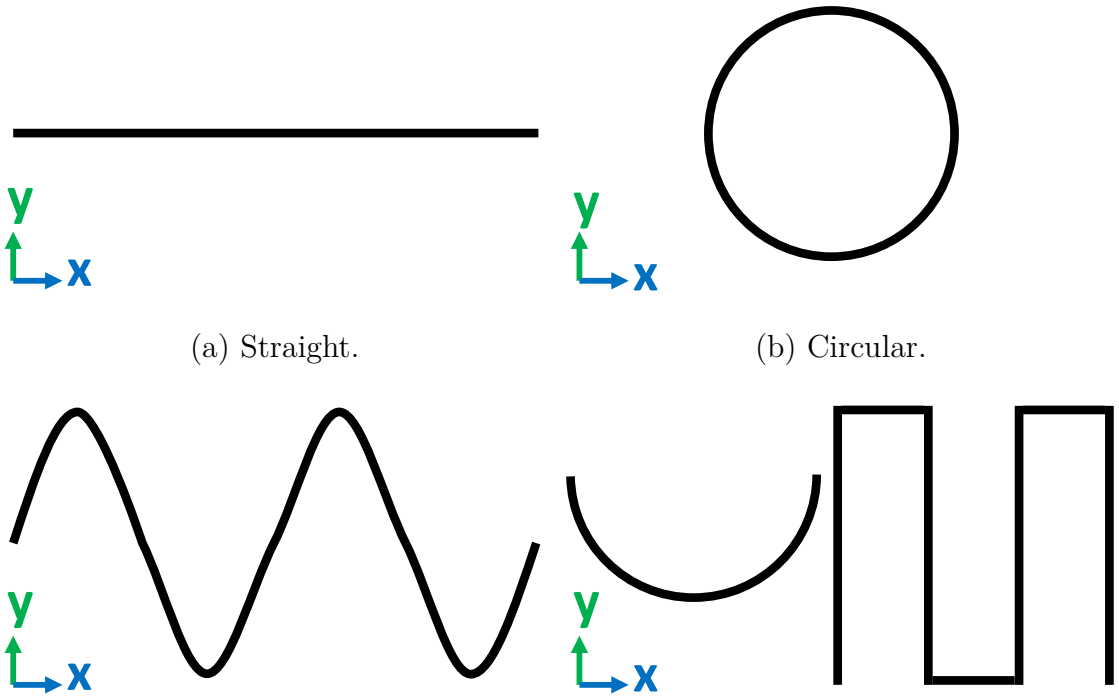
As far as the reference measurement process is concerned, again the acquired reference data point must reflect the true property of the object as accurately as possible. If the reference position is measured by a sensor, its accuracy must be high enough so as not to significantly skew the determined parameters of an RTS measurement. If the experiment setup provides inherent objectively known positions, its character and realization must allow for repeatable and reliable estimations of these positions.

4.3 Selection and evaluation of selected conception and parts

4.3.1 Target motion type

Based on the outlined requirements, parameters and available resources a suitable conception of the experiment to be performed has to be designed. Three basic shapes (or trajectories) of motion, which could be realized within an experiment in this thesis emerge from experiments performed in the aforementioned articles. As

is shown in Figure 4.1, these are a straight line, circular and sinusoidal motions, or possibly a combination or particular section of any one of them. The figure displays them in a two dimensional plane and viewed from the top. However, the particular choice of axes and orientation of the RTS in space relative to them is arbitrary and depends solely on experiment conditions and desired outcomes.



(a) Straight. (b) Circular.

(c) Sinusoidal. (d) Other parts or combinations.

Fig. 4.1: Examples of possible motion types and trajectories.

A pendulum type of motion with a spatial shape similar to that of a half-circle in Figure 4.1 (d) was chosen for the main experiment realization. An already available rotary mechanism, which was constructed as a part of a different thesis [37] could be used since it has all the necessary structural properties for the purposes of experiments performed in this thesis. The used bearings provide sufficiently low friction, which guarantees a satisfactory number of naturally damped oscillations even at lower amplitudes, provided that a suitable radius is established. By fixating the construction sufficiently, a repeatability of motion trajectory within one plane can be ensured. One source of instability is the actual pendulum arm, which in principle cannot be fixated. With an increasing radius, the rigidity of the arm decreases, however for a higher number of oscillations and greater traveled distances a larger radius is desirable. A way of increasing the arm's rigidity while maintaining sufficient length is presented further in Chapter 5. With a convenient orientation of

the RTS perpendicular to the plane of oscillation the processing task can be simplified into a two-dimensional realm. The main advantage of this character of motion is that the rest point of the pendulum can be exactly characterized by both the reference and RTS measurement. This type of motion also provides an extensive dynamic scenario with varying velocities and direction of motion changes, which can evaluate the overall RTS's capabilities in a scenario representative of a real-world motion.

4.3.2 Experiment realization concept

One of the outlined methods of experiment realization is utilizing known object positions and comparing their timestamps between both, the RTS and reference measuring processes. Given the character of the chosen oscillating pendulum motion such a precisely known position offers itself in the pendulum's rest, or zero, point. Exploiting this feature, a comparison can be performed between timestamps associated by the RTS and reference measurement at the exact moment when the pendulum crosses its rest position. This way, with a suitable experiment configuration, a direct expression of the delay between the moment of a physical event and the time the RTS associated with this event can be known. In other words, a total delay of position measurement by the total station can be expressed. This time delay can be easily converted, by integrating through the interval (eq. 4.2), into a position error and if the velocity of the object is constant during that period, relation 4.2 simplifies into 4.3. What is more, this time delay is also easily integrable into the navigation task, which permits better and more effective synchronization of all participating components.

$$\delta_{\mathrm{p}} = \int_{t_0}^{t_0+\Delta_{\mathrm{t}}} v \, dt, \tag{4.2}$$

where

- δ_{p} is the positional error,
- t_0 is the moment of the true physical position event,
- Δ_{t} is the time delay,
- v is the object's velocity during this interval.

$$\Delta_{\mathrm{p}} = v \cdot \Delta_{\mathrm{t}}, \tag{4.3}$$

where

- Δ_{p} is the positional error,
- Δ_{t} is the time delay,
- v is the object's velocity during this interval.

The perpendicular alignment of the RTS (see more in Sec. 5.2) also places it in a wort case scenario (also see Fig. 2.4), enabling a comprehensive evaluation of its real capabilities. Using this methodology, any inaccuracies emerging in the positional domain will, as a result, also be reflected and encompassed within the estimated time delay. This leads to an interpretation, where measured position values are treated virtually as absolutely accurate and the time error contains all the inaccuracies (which, ergo, can also stem from an inaccurate position measurement) within itself.

Two separate and different types of time delays will be talked about within this thesis and in order to minimize confusion, their visual representation can be seen in Figure 4.2.

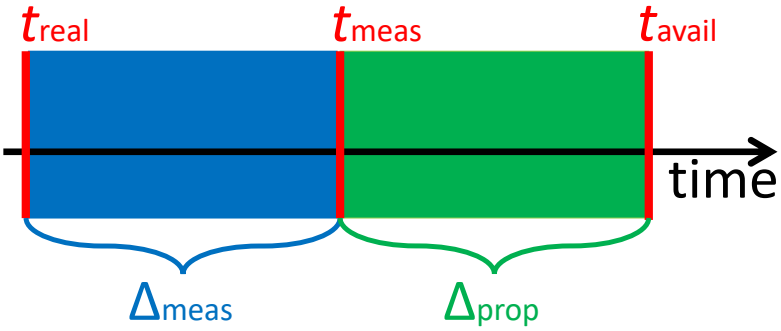


Fig. 4.2: Visual representation of estimated time delays.

The main delay emerging from the whole pendulum rest point experiment is the time difference between the real time when an object was physically occupying a certain position (t_{real}) and the time, which was assigned to this particular object's position by the RTS (t_{meas}). In other words, an overall position measurement delay is estimated (marked as Δ_{meas} in Fig. 4.2). The second one is the delay between the time that the RTS assigned to a position measurement (t_{meas}) and the time, in which the data from this particular measurement is available at the end device (t_{avail}). This will be called a data propagation delay and is marked as Δ_{prop} in Figure 4.2.

4.3.3 RTS measurement components

Evaluating the available total stations

Preliminary tests have been performed within the semestral part of this thesis to evaluate the suitability of the *Trimble S7* RTS. It had been found that this station truly does not provide a higher measurement rate during continuous tracking in the TRK mode and the frequency is limited to standard 2.5 Hz. Several different settings were tested, such as enabling and disabling the `FineLock` and `LaserLock`

functions, or changing the `Predictive tracking time` and data output baud rate, however none seemed to have an effect on the measurement frequency. Figure 4.3 displays an example of the time data from one of the tests.

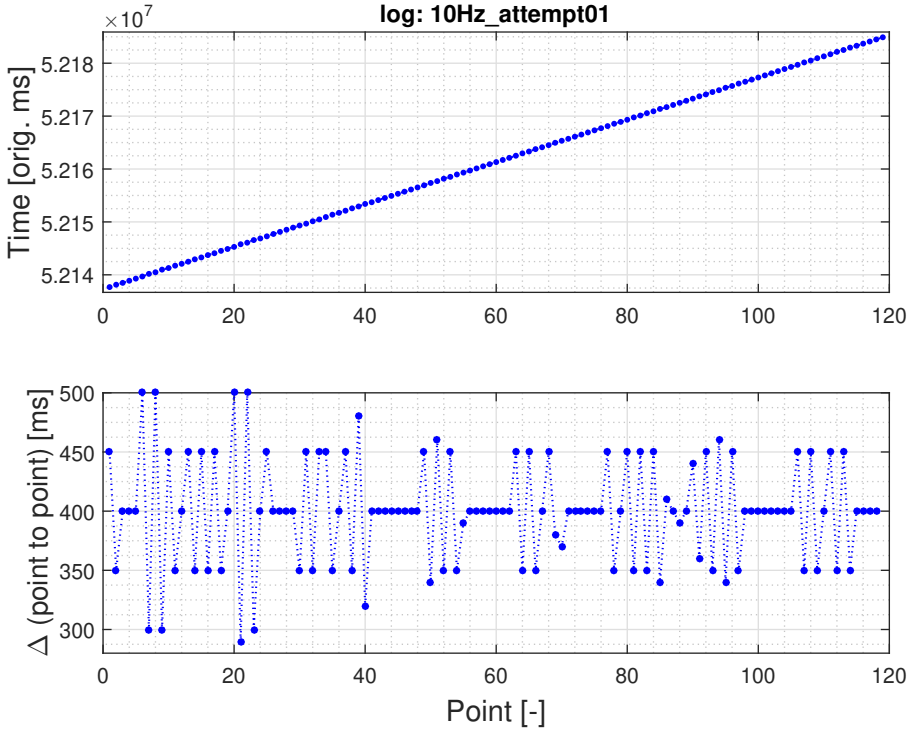


Fig. 4.3: Time data from the Trimble S7 RTS.

Further research into the *Trimble* documentation provided an interesting revelation. Although an older version of the documentation from 2017 [6] explicitly states that this feature is only available with the *S8* and *S9* series total stations, a newer (latest as of writing this thesis) version completely omits that note. From this, it follows that either this fact is mistakenly omitted from the newer versions, or perhaps the feature can be added even into the *S7* (and possibly others) by a firmware upgrade. However, given that the data-sheet for the *S9* explicitly states the availability of a *10 Hz tracking* feature, while the *S7* one does not, it leads to a conclusion that even though possible, it might be an undocumented enhancement at this time. The present firmware version, as of testing, was H2.7.19 and further investigations into upgrades contained within the newer versions were not conducted.

For the final experiments, the *Trimble S9 HP* RTS was ultimately chosen as it had been confirmed that it truly can provide measurements with a rate of up to 10 Hz. Figure 4.4 shows time data from the *S9 HP*.

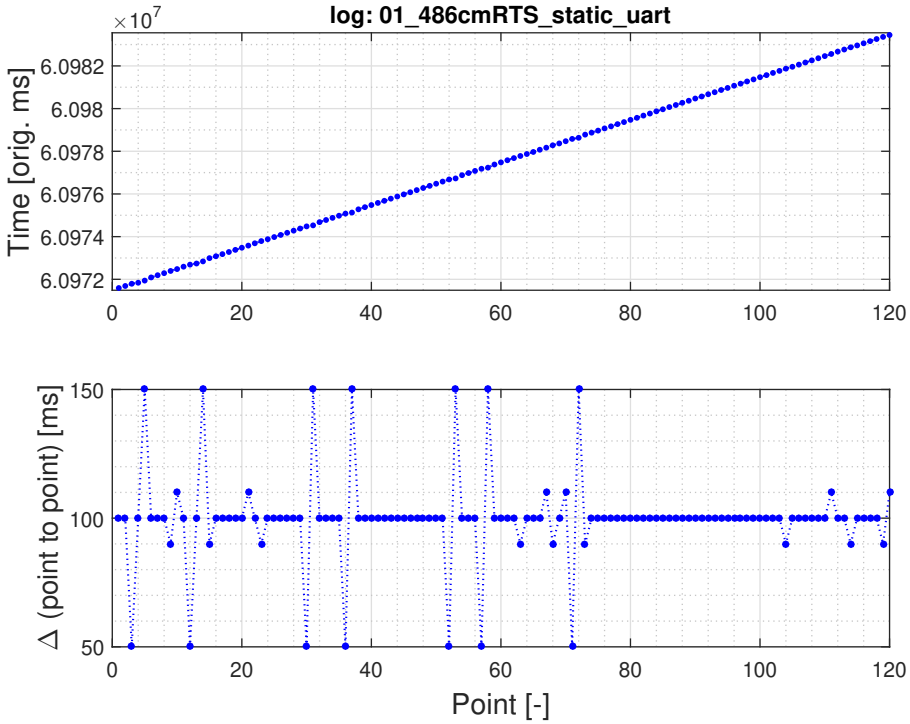


Fig. 4.4: Time data from the Trimble S9 HP RTS.

Data output settings

Data output was accomplished through the RTS's internal serial COM port. Initially in the semestral part, the data was captured by a separate PC at its USB port through an RS232 <–> USB cable. The baud rate was set to 57 600 bits/s and flow control was X_{on}/X_{off}. The number of data bits, stop bits and parity are implicitly set by the RTS to 8, 1 and none, respectively and this cannot be modified.

Given the outcomes of [30], the baud rate was subsequently raised to the maximum available 115 200 bits/s and all further experiments presented in this final thesis were performed with this transfer rate. Any other older results will only be presented in supplement B. The PC and RS232 <–> USB cable were left out completely by connecting the output directly onto the *Universal Asynchronous Receiver-Transmitter* (UART) pins of reference measurement device, which is dealt with in Section 4.3.4. In this case the flow control is set to none. This also means that the data acquisition times are within the same time frame as the reference measurement itself, ergo a further timestamp evaluation and data propagation delay from the RTS to an end device estimation can be performed. Table 4.1 at the end summarizes all the final parameters of the RTS measurement process for this thesis.

4.3.4 Reference measurement components

Measurement process performing device

The reference measurement processing device can be chosen from several distinct options. An obvious one might be direct control with a micro-controller, such as the *STM32* type. Another option is to use an industrial compact controller like the *CompactRIO* or a full-fledged *Programmable Logic Controller* (PLC). Possibly, the data can be processed directly by a *Field-Programmable Gate Array* (FPGA) module. All of these options provide real-time processing capabilities, which is beneficial to the task. Another option is a mini-PC, such as the *Raspberry Pi* (RPi). This device runs on an operating system and, therefore, its kernel handling might interfere and cause measurement delays as opposed to the true real-time solutions. However, it also provides ease of use and flexibility for various distinct tasks that might be useful for the experiment implementation. Direct hardware access is still available through its *General-Purpose Input/Output* (GPIO) pins, and by choosing a suitably low resource-hungry OS and a well implemented library for GPIO access, the negative effects of kernel scheduling should be sufficiently mitigated.

An entirely different approach is using a device, which can all by itself provide the complete required temporal and positional information with an accuracy much higher than that of the RTS. Some of the mentioned research groups were using a *Laser Tracker* device to serve as a reference source of the true object properties [13, 12, 18]. Its design is of a similar construction as an RTS, but it only uses a highly precise laser system to perform measurements. The laser beam is reflected back by a specialized reflector, similar to an RTS prism, however much smaller in size. It can also track a moving reflector and general use cases are performing highly precise alignments or surface shape measurements. A potential substitution of the entirety of an RTS by a Laser Tracker will be discussed in Section 6.4. As far as this thesis is concerned, such a device was not available at the institute and its procurement would be more complicated. From all of the above stated, the final choice was a *Raspberry Pi 3B* mini-PC, which was already available at the UAMT department.

Raspberry Pi configuration

The *Raspberry Pi OS Lite* Linux distribution was chosen as the operating system for the RPi, since it is directly developed and provided by the Raspberry Pi foundation and should provide the best optimization for the hardware. The lite version provides minimal overhead and an added benefit is an exhaustive documentation and support available for the OS. *Tickless kernel*, a mode used for power saving by reducing

the number of kernel updates when the CPU is idle, was disabled by adding the `nohz=off` parameter to the kernel *cmdline* at `/boot/cmdline.txt`. Although recent recommendations suggest that this option has minimal effect on time delays, saving power consumption is not a concern for this application and the mode was turned off to ensure no influences either way. The CPU governor was set to maximum performance by the `cpupower frequency-set -g performance` command and this setting was made permanent on boot by adding the command to `/etc/rc.local`.

As was stated in Section 4.3.3, the output data from the RTS was processed directly by the reference measurement device, which in this case is the Raspberry Pi. In order to connect the RS232 output with the UART input of the RPi, a conversion had to be made. Ready-made `RS232 <-> UART` adapters utilizing the *MAX-232* chip exist on the market, however a full both-way communication is not necessary and a much simpler one-way circuit is sufficient. The voltage levels of the RTS serial output are -5.8 V to +6.8 V for high and low logic levels, respectively. The *Transistor–Transistor Logic* (TTL) UART of the Raspberry Pi on the other hand operates with voltage levels of +3.3 V to 0 V for the same high and low logic levels. Since data is flowing only from the RTS to the RPi, voltage inversion and limitation are adequate. Figure A.2 shows these signal voltage levels. Figure 4.5 displays a simple circuit performing these adjustments.

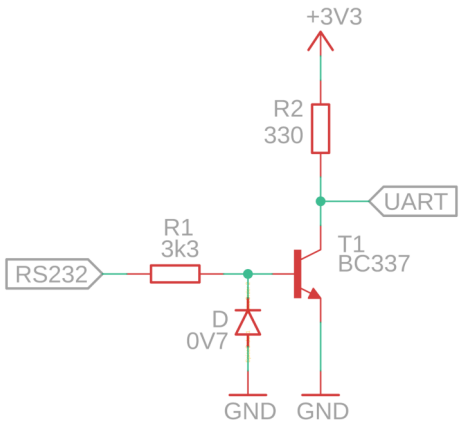


Fig. 4.5: Serial voltage adjusting adapter for GPIO UART.

Standard 3.3 kΩ and 330 Ω resistors provide suitable current values on the order of milliamperes. When the input voltage is high, the transistor opens and the output level is set to 0 V. Once the input voltage goes to negative values, all the current flows through the diode and the transistor stays closed, in which case the output is pulled to 3.3 V.

Library for implementation

A well known and widely used library for GPIO access on the Raspberry Pi is *WiringPi*. It provides a simple API to interface with the pins and is designed to be consistent with the *Arduino wiring* system style and functionality. However, *Arduino* code compatibility is not important for this thesis, and highly accurate timing and processing seem to rather not be a primary concern for the library implementation. As of 2019 it is also deprecated and is not updated anymore [38].

Another option is the *bcm2835* library. It provides a fast API for C/C++ programming, however interrupt services are not supported [39].

The measurement program was ultimately implemented using the *PiGpio* library in C language. This appears to be the lowest level and fastest GPIO access library by implementing timing and GPIO handling directly through the kernel. Interrupts are also supported. The code is open source and available on github, and it also provides a wider and more comprehensive set of functionalities. The API it provides is easy to use and understand, and it is well documented [40].

Event detecting sensor

For the pendulum rest position detection a small photoelectric light barrier sensor configuration was used. Initially, in the semestral part, an *Omron* U-type photomicrosensor was used. Its reaction time is max. 333 µs at $U_{cc} = U_{out}$. The state change reaction area is 1.1 mm from each side. The light emitting/receiving slit width and length of the barrier were 2 mm and 15 mm, respectively. It was powered from a +5 V output GPIO pin on the RPi. The NPN open collector output was connected to an input GPIO pin through an external 3.3 kΩ pullup resistor at +3.3 V. The effect of output U_{out} voltage being lower than the power $U_{cc} = +5$ V on the reaction time was not considered [41]. The configuration can be seen in Figure B.1.

Subsequently, for this final thesis implementation, a *Panasonic EX-Z11* series photoelectric sensor configuration was selected. Again, any older results achieved with the *Omron* sensor will only be provided in supplement B. Its reaction time is less than 500 µs. The light emitting and receiving slit width is 0.3 mm and the distance between the emitter and receiver can be set as required and up to 50 mm. Its connection was more involved as it requires a minimum power source of +12 V DC and the open collector output of the receiver was ultimately a PNP type transistor. The necessary 12 volts were supplied by an external switched-mode power supply. Since the transistor opens to full +12 V power, the output cannot be connected directly to a +3.3 V input GPIO pin [42]. However, a simple voltage-limiting Zener diode circuit easily solves this problem. Figure 4.6 shows the schematic, where the additional circuitry of the transistor and sensor is omitted.

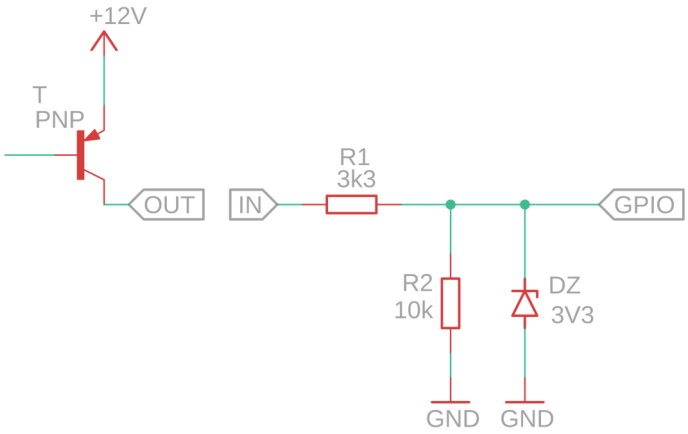


Fig. 4.6: Sensor voltage limiting adapter for GPIO.

The 3.3 V Zener diode provides a save voltage for the GPIO pin when the transistor opens. By the *Panasonic* specification, a maximum allowed output current is $I_{out,max} = 20$ mA. This leads to a theoretical minimal value for R_1 given by the Ohm's law 4.4, which is 435 Ω.

$$R_{1,min} = \frac{U_{out} - U_D}{I_{out,max}}, \tag{4.4}$$

where

- $R_{1,min}$ is the minimum resistance,
- U_{out} is the output voltage from the sensor,
- U_D is the diode voltage.

The R_2 serves as a pulldown resistor and sets the GPIO pin value low when the transistor is closed. The supply voltage of R_2 needs to be greater than U_D, *i.e.* equation 4.5 needs to be satisfied. The voltage can be calculated form the equation for a voltage divider 4.6.

$$U_{R_2} > U_D, \tag{4.5}$$

$$U_{R_2} = \frac{R_2}{R_1 + R_2} \cdot U_{out}, \tag{4.6}$$

where

- U_{R_2} is the voltage supplied by R_2.

By combining 4.5 and 4.6, the minimum value for R_2 is given by 4.7.

$$R_2 > \frac{U_D}{U_{out} - U_D} \cdot R_1, \tag{4.7}$$

As can be seen in Figure 4.6 standard 3.3 kΩ and 10 kΩ resistors were chosen for R_1 and R_2 respectively, which are well above the required minimums and still provide a high enough reverse current to the Zener diode. Again, all final reference measurement parameters are summarized in table 4.1.

4.3.5 Time synchronization

Finally, since a comparison between the time values from the RTS and reference will be made, a synchronization procedure between both of these processes needs to be maintained. Since the TSC7 and Raspbery Pi both run on an operating system, one solution might be synchronizing them to a common *Network Time Protocol* (NTP) server. This would give both devices the same time reference, however it has drawbacks. Most notably, the delays when connecting to external NTP servers are on the order of tens of milliseconds. Acquiring an internal time source would be beneficial and reaching sub-millisecond accuracies should be possible with proper implementations (such as the `chronyd` package on Linux), however a different approach appears to be more suitable.

Precision Time Protocol (PTP) is a time synchronization protocol used for high precision applications and on a local network with proper settings it can achieve sub-microsecond accuracies. For Linux operating systems two main implementations are available: *PTPd* [43] and *Linuxptp* [44]. The latter seems to be a more comprehensive implementation with more options and a wider support. Notably, a *HW timestamping* functionality is implemented directly through the Linux kernel. This means that timestamps can be acquired directly by the *Network Interface Controller* (NIC) hardware upon receiving and sending packets (if it has the necessary components) without any further overhead. This allows for the sub-microsecond accuracies mentioned earlier. However, the NIC present of the Raspberry Pi does not support this functionality and more importantly, *Linuxptp* is not compatible with the Windows OS, which is on the TSC7 controller. Enabling the processing of PTP packets on Windows is a matter of configuring the correct registries and in theory it should be able to react with messages from any Linux implementation. After further research the problem seems to lie in the way these messages are handled. Windows accepts PTP packets only in *unicast* mode and presently does not provide support for a *HW timestamping* implementation. On the contrary, packets sent by *Linuxptp* can be in *unicast* mode exclusively only with *HW timestamping*. Another thing that Windows requires is that the receiving packets must have the `ptp_timescale` flag set to `PTP`. Again, this flag is probably only set with *HW timestamping* for *Linuxptp*. Another time acquiring method is *SW timetamping*, which is a software-based approach and it is implemented within Windows. This method can provide accuracies on the order of microseconds, which is still suitable for the experiment purposes of this thesis [45]. On Linux, the latest release of *PTPd* is managed by *Aptitude* with the `apt-get` command and for *Linuxptp*, a custom script named `updateptp4l` checks for the latest source on their sourceforge repository, compares it with the currently installed and performs an update if necessary.

Attempts were made to enable the accepting of *multicast* PTP messages on Windows. A registry key called `EnableMulticastRx` exists in the PTP configuration section, however it had no effect when enabled at the time of writing this thesis. Ultimately, PTP synchronization using the *PTPd* implementation on Linux, directly between the TSC7 controller and the Raspberry Pi was performed, where the (grand) master source clock was the RPi and the controller adjusted its clock in slave mode. The synchronization was always running throughout the whole measurement process.

RTS	**Model**	Trimble S9 HP
	Controller	TSC7
	Measurement period	100 ms
Reference	**Device**	Raspberry Pi 3B
	Sensor	Panasonic EX-Z11
	Sensor slit	0.3 mm
	Sensor distance	5 mm
	Application	PiGpio, C
Data transfer	**Source**	RTS COM
	End	RPi UART
	Baud rate	115 200 bits/s
	Flow control	None
	Data bits	8
	Stop bits	1
	Parity	None
Time synchronization	**Protocol**	PTP
	Implementation	PTPd
	Master clock	Raspberry Pi 3B
	Slave clock	TSC7
	Interface	Ethernet

Tab. 4.1: Main experiment parameters and configuration.

5 Experiment realization procedure

This chapter will inform the reader of the actual physical configuration and layout of the experiment and its components. All the final chosen and argued concepts from previous chapters will be presented here in their real physical realizations and the methodology and procedure for the experiments will be shown. The device configurations, setups and developed programs will be presented.

5.1 Description and display of reference measurement components

5.1.1 Raspberry Pi 3 B

Reasoning for the selection of this device was provided in Section 4.3.4. It comes with a 1.2 GHz 64-bit *BCM2837* processor, 1 GB of RAM and storage for the OS and data is maintained by an external Micro-SD card. It provides a 100 Base Ethernet *Local Area Network* (LAN) connection, and also Wi-Fi and *Bluetooth* connectivity. Power is provided by a 5 V/2.5 A power supply. A set of 40 GPIO pins is available, which can provide reference 5 V, 3.3 V or Ground, access to peripherals such as UART or I2C, or general input/output functionality [46].

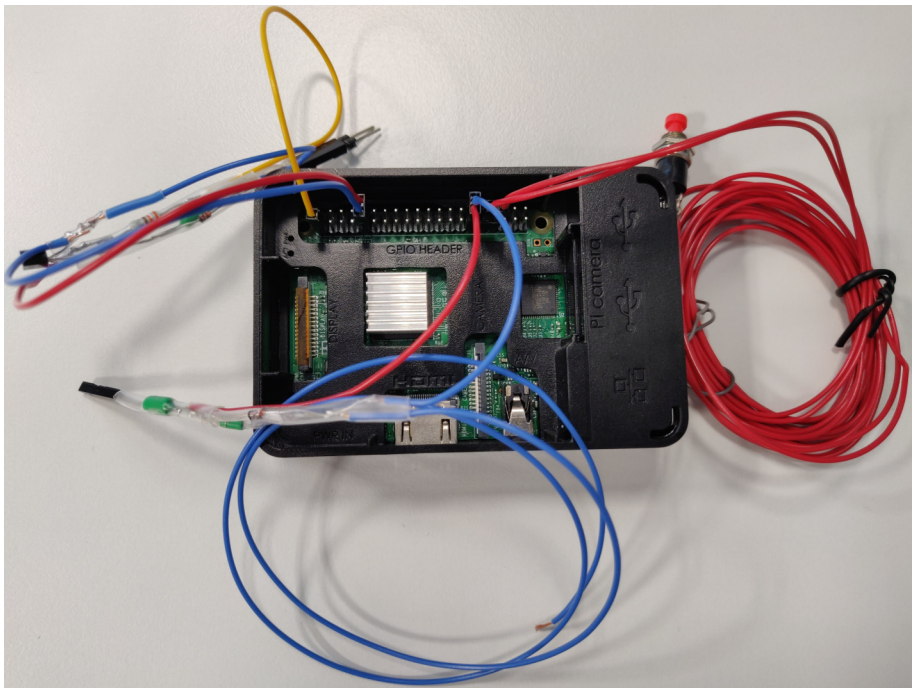


Fig. 5.1: Both adapters connected to the Raspberry Pi.

As was mentioned, for connecting with the sensor and RS232 serial output, simple custom adapters had to be made. Their description is provided in Section 4.3.4. Figure A.3 shows their realization for use and Figure 5.1 shows their connection to the RPi.

5.1.2 Panasonic EX-Z11

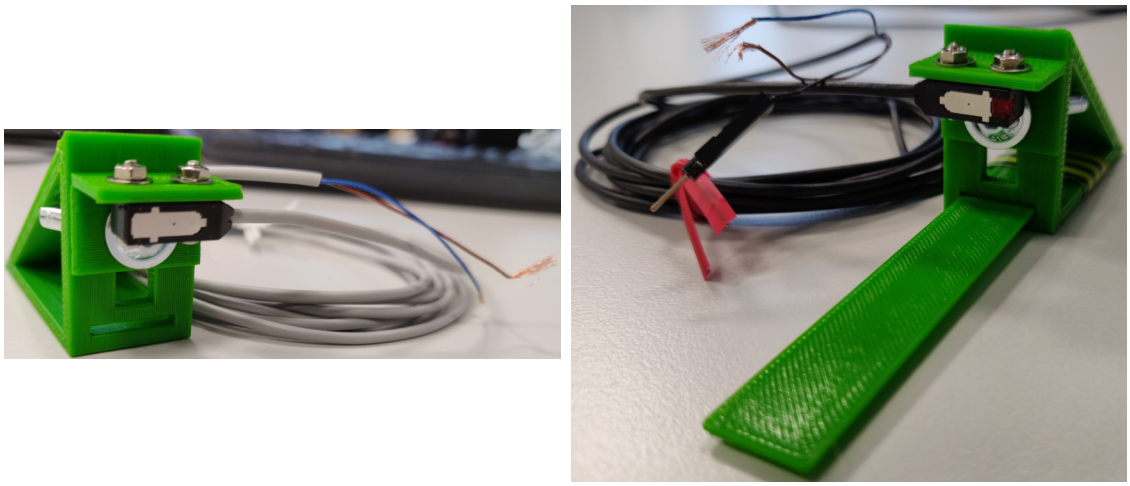


Fig. 5.2: Mounting brackets for the Panasonic EX-Z11 sensor. Left: emitter, Right: receiver.

It is a miniature photoelectric sensor that uses a 650 nm LED for motion detection. It comes as physically separate emitter and receiver units that can be mounted according to desired use. The width of the emitted light is controlled by a 0.3 mm slit and the freely adjustable emitter-receiver distance was set to be 5 mm. For this, custom mounting brackets, where the position and distance of the two elements can be adjusted, were designed and 3D-printed (Fig. 5.2).

5.1.3 Sensor-crossing prism extension

Chapter 4 provided a more abstract theoretical description of the experiment design and realization methodology. All of the individual components of the experiment have now also been described, but what remains to be further elaborated is the actual realization of the mentioned *event* or objectively known position in the pendulum rest point. From all that was mentioned so far, it is evident that some kind of an object is going to cross a light barrier sensor setup providing a reference and that the RTS will be tracking a prism constantly during the motion.

For this purpose, an attachment that is mounted onto the moving prism was designed and 3D-printed. The barrier-crossing element (a thin rod-like extension)

was created as narrow as possible in order to minimize asymmetricality of the setup, since it will be crossing the light barrier from both sides as the pendulum swings back and forth. Its circular shape allows for its invariance to orientation at which it is crossing the barrier. An earlier quickly set up proof-of-concept attachment that was used in the semestral part is shown in Figure B.1 on the right. The actual design with a 2 mm thin rod extending from the prism attachment, which was used in this final thesis, is displayed in Figure 5.3. This was the smallest thickness that could be reasonably printed and, together with the 0.3 mm slit on the sensor, should create minimal deviations between the left and right swings.

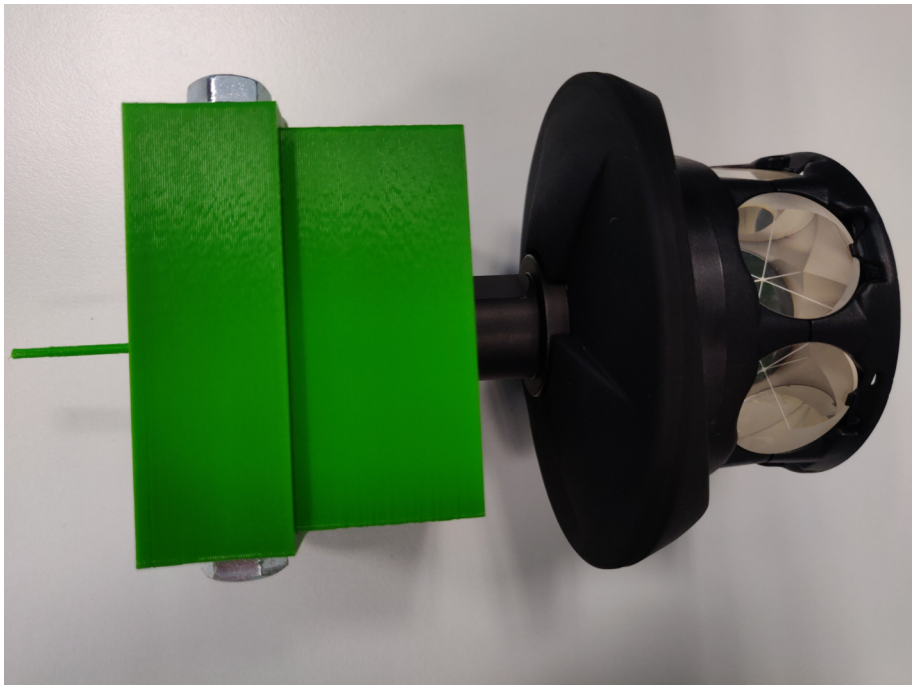


Fig. 5.3: A thin rod extension crossing the light barrier mounted on a Trimble VS/S 360 prism.

The attachment consists of two parts. The inner section mounts directly onto the prism and serves as a bracket for the actual attachment with the rod extension. The parts are secured together by two screws protruding through the whole construction. Since the desired precisions of the setup are expected to be within millimeters and milliseconds, any kind of movement at the connections between the two parts should be eliminated. For this, internal grooves and protrusions where designed to secure both parts firmly in place and prevent any rotational or lateral movement. Figure 5.4 shows the inner design of the parts.

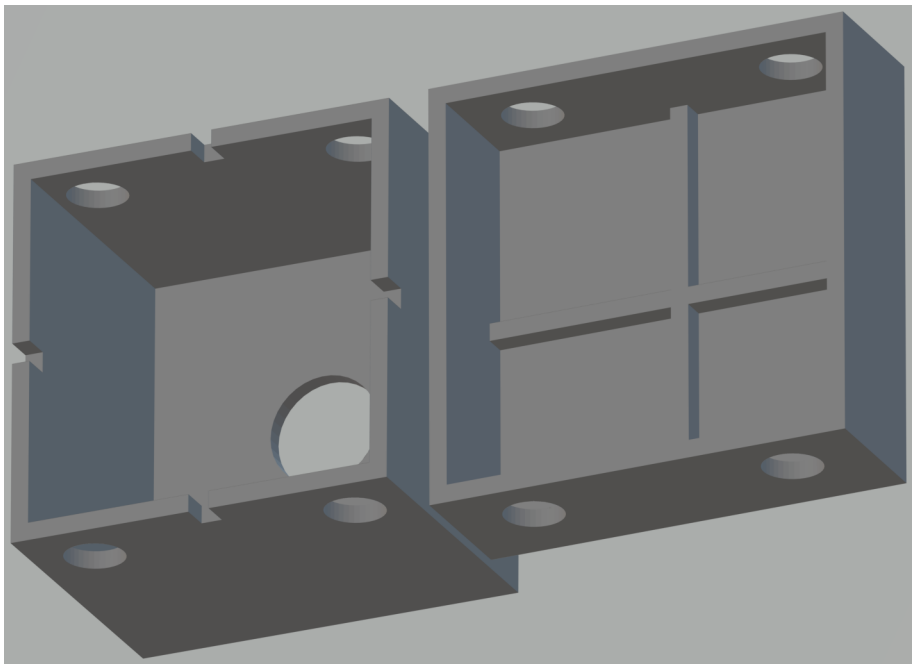


Fig. 5.4: Detail on the inner connection between the two attachment parts.

5.2 Component layout

This section will describe the physical realization and layout of the main experiment. The pendulum, prism and sensor setup can be seen in Figure 5.5 and a graphical top view of the layout in Figure 5.6, right. The extension described in Section 5.1.3 was, together with the prism, mounted onto the end of the pendulum arm. Its set length ensured a large enough number of oscillations through the sensor and also enabled the extending thin rod to be as short as possible (32 mm) in order to minimize any of its own vibrations. The whole construction was firmly fixed in place so that it didn't negatively interfere with the observed motion. As was stated in Chapter 4.3, the only unsecurable part is the arm itself, which can have unwanted vibrations in the lateral direction. Its rigidity was increased by fixing two of such pillars onto each other in that specific direction. This reduced the negative vibrations, leaving the weakest link in the connection at the center of rotation, which in principle cannot be fixed any further. The negative effects were negligible (see Sec. 6.3.3).

The reference measurement light barrier sensor setup was fixed on the ground. It was precisely positioned such that the pendulum in its rest position just breaks the light barrier with its left side, as seen from the point of view in Figure 5.5. This represents the exact true known zero position of the pendulum. The oscillating motion was always initiated from the right side, again from the view in Figure 5.5. The RTS was placed on the ground (in a position behind the camera view) at varying distances from the pendulum. It was oriented to be as perpendicular as possible to

Fig. 5.5: Experiment component layout (RTS behind camera view).

the plane of oscillations, so that the whole motion was being performed mostly in one axis. Perfect alignment can in practice never be achieved, however a method which provided an observable difference was applied. The basic concept can be seen from Figure 5.6 on the left.

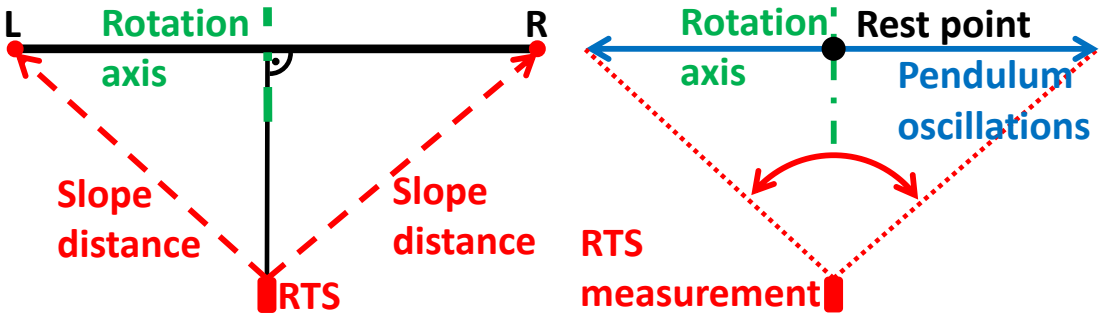


Fig. 5.6: RTS alignment principle (left) and component layout (right), top view, not the of same scale.

A slope distance value from two points lying on a horizontal line and equidistant from the axis of rotation was measured. The equidistance was achieved by observing the edges of a circular structure seen in Figure 5.5 at the top. The RTS was set to DR mode for laser-target measurements. Its vertical angle was first set to align with a specific point on one side of the circle. Subsequently, only the vertical angle was being changed (moving left and right) to align with a second point on the other side, while the vertical angle remained fixed. The RTS was positioned such that

the measured distance values from both points were equal. This procedure has its limitations and depends on the precision with which the two points can be sighted.

In summary, the pendulum arm radius was 128 cm and the RTS was placed at three distinct distances from the pendulum, where several measurements were made. The largest was at around 4.9 m, following with around 2.7 m and around 1.5 m.

5.3 Configuration and setup of experiment components

5.3.1 Total Station configuration

Whole operation of the total station was performed using the TSC7 controller, settings of which will be described in a following section. The RTS was set up as follows. After the initial orientation alignment, level calibration is performed as described in Section 3.6. Subsequently, a coordinate system suitable for the experiment can be defined. A flat linear Euclidean-space local coordinate system with a scale factor of 1.0 was used for all measurements performed within this thesis, since it is perfectly sufficient for the purposes of local navigation with an RTS. A global system with polar coordinates referenced with GNSS is not relevant for these purposes, because the RTS itself will be the source of the coordinates.

Following the basic *Station Setup* procedure then, two points have to be measured. First is the *Base Station* point, which defines the location of the RTS itself within the coordinate system. Again, the system for navigation is going to be fully defined and provided by the RTS, therefore an explicit setting for the *Base Station* point as the origin of the coordinates is preferable. The values for x, y and z were explicitly set to $[0, 0, 0]$. The height value for this point was set to 24 cm, which is the measured height from bottom (ground) to the top notch at the sighting axis and this ensures that the coordinates will originate from the ground in the vertical direction. The second point, called a *Back Sight* defines the horizontal orientation of the coordinates and its position is measured by the RTS. The *Back Sight* will also define the known zero target position for the RTS. When the pendulum was at its rest position with the prism at the bottom, the RTS was, using the *AutoLock* function, locked onto the prism. This point was then measured as the *Back Sight* in STD mode with 10 averaged measurements, ensuring a highly precise definition. The azimuth value was also set to 0 to define it as the starting orientation for horizontal angle measurements. The starting orientation for vertical angles is fixed at the zenith. Finally, the height of this point was set to 0, as the RTS was measuring the real prism location with no additional offsets.

After accepting of the measured *Back Sight* point, the station now provides location data of the measured target within the defined coordinate system. The RTS defines these local coordinates in the *East, North, Up* (ENU) framework. Figure 5.7 displays the resulting configuration and Figure A.1 shows a particular example with several other points on the TSC7 screen. The RTS there was not placed directly on the ground and that is why the points are located below the origin with negative height values. The location ENU values sent out as data output are of the *Easting*, *Northing*, *Elevation* convention.

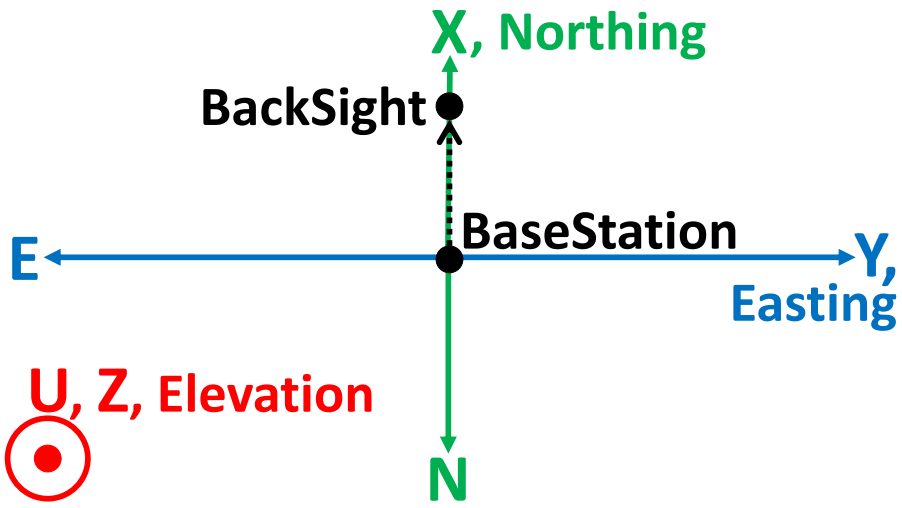


Fig. 5.7: Representation of the RTS local coordinate frame.

This configuration implies extrapolation of the measurements from the reference, which as an effect might provide worsened accuracies as is stated in [47]. The proposed configuration is with several *Back Sight* points enclosing the desired measurement perimeter. However, this seems to apply more to general static *topo* or *stakeout* measurements and *resection* techniques, which are not relevant for the purposes of this thesis with dynamic motion.

EDM settings

The final step for the RTS is enabling TRK mode and the *AutoLock* function. The 10 Hz measurement functionality can be enabled by a checkbox in the settings, but is only available when *AutoLock* is on and the station is in TRK mode. The target type has to be a prism and whenever the station is set to DR measurements, it will automatically disable the functionality until a prism target is selected. The *FineLock* function, which is used for static measurements with multiple close by prisms and does not work with TRK mode, should also be disabled. It might also be beneficial to disable the *LaserLock* function as it interferes with *AutoLock*.

Data output

For data output it is necessary to keep the screen opened in the background. The output was set to `Continuous` mode and a `GDM user defined` structure is selected where all the necessary values can be set for output. The data was being sent in a format shown in Figure 5.8, where each measured value is specified by its label number and a complete data point is ended with the `'>'` termination character. An example of acquired data from the RTS can be seen in Listing 5.1. Highest resolutions of all displayed values were set in the global job settings. In the data output section, the most important is the resolution of time values that will be sent out. The highest possible resolution in 10 ms. Parameters of the data transfer were summarized in Table 4.1.

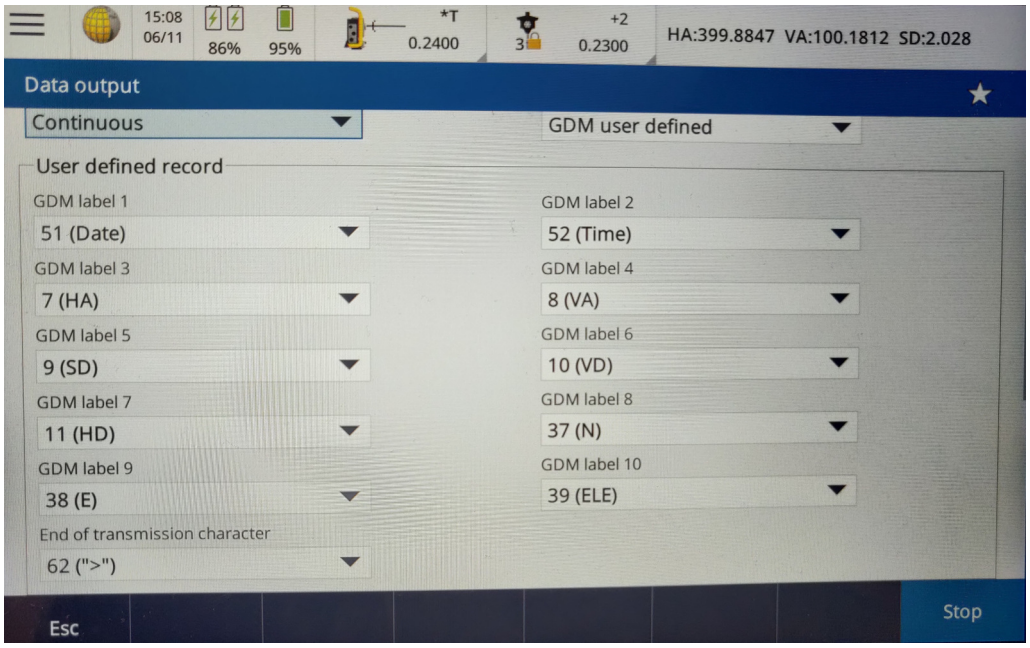


Fig. 5.8: Configured RTS data output labels.

As was mentioned, no flow control was used for transmission. The hardware flow control RS232 outputs were, therefore, interconnected with their corresponding inputs (see Fig. 5.9) so that transmission was always running and the RTS was continually sending all of its output from the COM interface immediately as it appeared. The Raspberry Pi was then immediately processing this data on its UART input. The conversion from RS232 to UART was explained in Section 4.3.4.

With all this, the RTS is finally configured for the experiments and is constantly sending data through its COM serial port.

Listing 5.1: Example RTS data output, showing one point.

```
0
51=2021.0421
52=18.411712
7=0.00006
8=102.00073
9=2.660
10=-0.084
11=2.659
37=2.659
38=0.000
39=23.916
>
```

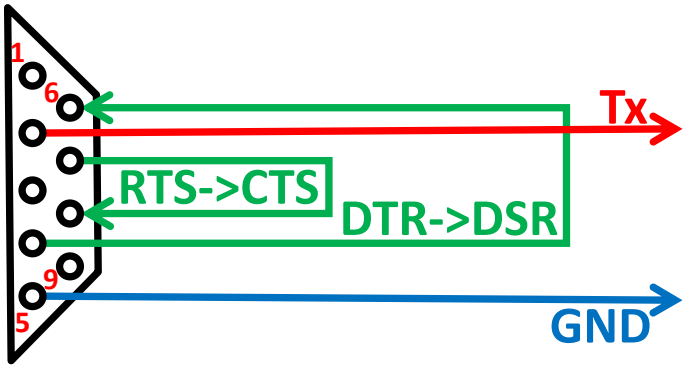


Fig. 5.9: RS232 DB9 connector pin connections.

5.3.2 TSC7 controller configuration

General Windows settings

Some additional modifications, for the purposes of experiment, were also made to the TSC7 controller. The version of the Windows operating system, which was used at the time of performing all experiments was `20H2`, build `19042.867`. The *Trimble Access* application, used for controlling the RTS, was updated to its latest available version `2021.00`. A basic user-space configuration of the OS was made. Features such as transparency, animations and hiding of scroll bars were disabled for high responsiveness. Most settings in the `Privacy` and `Gaming` sections were turned off or disabled and some services which were not relevant for this thesis, such as `Connected User Experiences and Telemetry` or `Distributed Link Tracking Client` were also disabled in order to assure as minimal as possible OS interference with the experiment. All general setting changes are documented in the `TSC7_changes.txt` file.

PTP configuration for the experiment

Since the TSC7 also provided its time to the RTS and the acquired timestamps reflected this time, the controller was synchronized with the reference measuring Raspberry Pi. A more involved configuration was needed to set up the PTP synchronization. Basic guidelines however, are available at the *Microsoft* github repository

[45] and a set of custom *PowerShell* scripts was developed to easily configure the controller for experiment and revert all changes back to the original configuration after its completion. The execution of unsigned local *PowerShell* scripts first had to be enabled in the `Update & Security -> For Developers` settings section. The lowest level scripts:

- `EnableSWTimestamping/DisableSWTimestamping`,
- `EnableW32TMlogging/DisableW32TMlogging`,
- `SetUpPTPFirewall/RestorePTPFirewall`,
- `SetUpPTPWinTime/RestorePTPWinTime`,
- `QuerySyncStatus` and `LogSyncStatus`

perform the necessary configurations and display the running state. The higher level ones:

- `StartMeas/StopMeas` and `SetUpEnvironment/RestoreEnvironment`

then call them to quickly prepare the device and manage the measurement process.

Three general actions were necessary. First, the `SoftwareTimeStamping` module had to be installed, if not present already, and the mode had to be enabled on the desired network interface. The synchronization was performed through Ethernet on both devices (TSC7 and RPi) and since the TSC7 does not have an Ethernet port, a converter from USB 3.1 to Ethernet was used. The influence of this converter on the quality of synchronization was imperceivable. Second, ports 319 and 320 had to be enabled in the firewall. Lastly, correct registry key values had to be set to enable high accuracy timing for the *Windows Time Service*, enable the PTP provider and disable other time providers. All modified registry values are, beforehand, saved to `.reg` files to be later restored. These values were set according to the *Microsoft* recommendations, where a couple of changes were made. The `DelayPollInterval` was changed from 16 seconds to 1 second and the `AnnounceInterval` from 4 to 0.5 seconds. Also, the `PollInterval` was changed from default 64 to the lowest possible 4 seconds. More about these settings is also at [48].

An officially undocumented registry key called `AllowAnyMaster` was found within the PTP configuration section. On older versions of Windows this had no effect, however when set on the `20H2` version, it enables the computer to accept correctly configured PTP packets from any device that is casting them towards it. This might lead to incorrect clock adjustments and time errors in multi-device setups, however the scheme in this thesis is a simple two-device setup and no other interfering clock sources were present on the network. It is still a *unicast* process, nonetheless it simplifies the configuration as, ultimately, only a single *unicast* destination address has to be configured.

A *Windows Time Service* log was also maintained during the synchronization process with the `w32tm /debug /enable` command to assess the quality and reli-

ability of the synchronization. Additional useful information, such as error states, was also obtained from the `w32tm /query /status /verbose` query command and so its output was also periodically being saved into a log file.

All of the mentioned configuration is performed automatically by the developed *PowerShell* scripts. `QuerySyncStatus` executes the mentioned query command periodically and could be used to quickly verify the state of the synchronization. The `LogSyncStatus` script can save this periodic output into a log file. Executing `SetUpEnvironment` enables *SW Timestamping* on the Ethernet port and sets the appropriate firewall rules. Subsequently, the `StartMeas` script can start the logging and set the necessary registry key values, by which the synchronization process is initiated. The log filename can be provided as an argument and since two files are being handled at the same time, appropriate modifiers (`_w32tm` for time service logs and `_w32tm_cst` for query logs) are applied to the name automatically by the scripts. The `Restore, Stop` and `Disable` counterpart script versions restore all the modified settings and configuration back to a state before any of them were made.

It is important to state that the presented Windows PTP configuration only accepts packets, effectively, in slave mode and it could not serve as a master clock. That is why the Raspberry Pi was the master source clock within the scheme.

The TSC7 controller is now ready to receive synchronization PTP packets from the Raspberry Pi on the Ethernet port.

5.3.3 Raspberry Pi configuration

PTP configuration for the experiment

The RPi was, through its Ethernet port, connected to the same LAN network as the TSC7, from which it was accessed and it also could connect to the internet. For easier subsequent data processing and reference, the internal time of the RPi was first synchronized with an NTP server (`time.google.com`). For this, the `chronyd` package was used with a single run of the `chronyd -q -m 'server time.google.com iburst'` command from the `sync_ntp` script. Subsequently, only PTP is used as the protocol for any further synchronization. The approach of custom *Shell* scripts for the various configurations and setups was adopted here as well.

The `sync_ptp_master` script starts the PTP synchronization using the `ptpd -c ptpdwindows.conf -V` command. As can be seen, PTP synchronization settings are provided by the `ptpdwindows.conf` configuration file. The *Microsoft* github repository [45], again, provides recommended settings. Most of these were, however, already default in the used `ptpd` package. A full breakdown of all the recommended settings compared with the default values is provided in the

- `ptpdunicast_fullexample_descriptions.txt`

file. The most important ones are:

- `ptpengine:ip_mode`, which must be set to `unicast`,
- `ptpengine:unicast_destinations`, which contains the IPv4 address of the TSC7 controller,
- `ptpengine:ptp_timescale`, which must be set to `PTP` and
- `ptpengine:clock_class`, which must be set to the value 10.

The other two important settings are `ptpengine:log_announce_interval` and `ptpengine:log_delayreq_interval`, which must both be set to the same values as their corresponding counterparts on the Windows side (`AnnounceInterval` and `DelayPollInterval` registry keys, respectively), otherwise the synchronization will not work. It is important to keep in mind that while on the Windows side, these values are set as hexadecimal numbers representing milliseconds (where 0x01f4 => 500 ms), here they are set as exponents of 2 representing seconds (where -1 => $2^{-1} = 0.5$ s). Some of the other non-default recommended settings relevant to this thesis were also set. Otherwise, all was left default as in the `ptpd` package.

All in all, after executing the `sync_ptp_master` script, the Raspberry Pi is sending master PTP packets to the TSC7 controller though its Ethernet port.

Initiating the measurement

After confirming that the PTP synchronization is running and functional, the measurement program on the RPi can be started. A detailed description of its workings will be presented in Section 5.3.4. The program was written to also process command line arguments and it can be started by executing either the `measure` or `measurewargs` script. The former was used mainly during testing and it already supplies a default output file name argument. The latter was used for actual experiments, where the file name and other arguments are passed through from the user. In both cases the program is compiled from source by the `gcc -Wall -pthread -o main main.c storetime.c storetime.h -lpigpio -lrt` command and run.

After this, the physical pendulum motion (or other experiment) was initiated. When it was done, the measurement was stopped, synchronization ended and the TSC7 controller reverted back to its original configuration.

5.3.4 Custom measurement program

The program was developed using the *PiGpio* library in C language. Since the *PiGpio* version managed by *Aptitude*, with the `apt-get` command, is sometimes behind the latest available on github, a custom script called `updatepigpio` was created, which checks for the latest release directly on github, compares it with the currently installed one and performs the update if necessary. The basic functionality can be

divided into two parts. First is the main thread, which handles the *Panasonic EX-Z11* sensor and assigns the pendulum event timestamps. The second is a separate parallel thread, which handles UART communication by logging all data output from the *Trimble* RTS and time stamping each arrived measurement point.

Main thread

This thread acquires the main reference timestamps, which are used for comparison with the RTS time and a subsequent estimation of a delay, with which the RTS measures positions (t_{real} in Fig. 4.2). It first configures the GPIO pins for the intended use and allocates a buffer to be used for the sensor timestamps. The secondary thread for UART is initiated and subsequently an *Interrupt Service Routine* (ISR) handles the sensor light barrier crossing events (on a configured GPIO pin) until either the buffer is filled or a termination sequence is initiated. This is done either with a signal message from pressing the `ctrl+c` combination or with a physical button connecting specific GPIO pins. The interrupt handler immediately acquires a timestamp using the `clock_gettime()` function from the `sys/time.h` library and adds it to the buffer. The code snippet in Listing 5.2 shows an example of this assignment.

Listing 5.2: Light barrier event timestamp assignment.

```
struct timespec current_time;
clock_gettime(CLOCK_REALTIME, &current_time);
VOID2TIMELB(aData)->
  buffer_ns[VOID2TIMELB(aData)->val_idx] =
  (uint32_t)current_time.tv_nsec;
VOID2TIMELB(aData)->
  buffer_s[VOID2TIMELB(aData)->val_idx] =
  (uint32_t)current_time.tv_sec;
VOID2TIMELB(aData)->
  buffer_isrticks[VOID2TIMELB(aData)->val_idx] = aTick;

++(VOID2TIMELB(aData)->val_idx);
```

After a termination condition is met, the ISR is disabled and the main thread waits for the UART thread to gracefully finish and exit. Subsequently, all the buffer contents are dumped into a *Comma-Separated Values* (CSV) format log file. The length of the timestamps buffer and name of the log file can be specified by

input arguments with keywords `-l` and `-o`. Otherwise, a default buffer length is used and the program asks for an output file name. The nanosecond precision time values acquired from the `clock_gettime()` function are used for the eventual time evaluation. The *PiGpio* ISR implementation itself provides a microsecond timestamp of when the interrupt call was detected, however this value is given as an elapsed period from OS boot time. Since this moment is not known globally, the resulting timestamp could not have been used for comparison with the actual RTS time values.

UART thread

This thread saves incoming data from the RTS and assigns timestamps, which are used to evaluate the time it takes for the RTS measured data to arrive and be available at the end device (t_{avail} in Fig. 4.2). All the incoming UART data from the RTS is saved to a log file and an arrival timestamp is assigned to each full measurement point, which is then being saved to a CSV format log file. Names for these files are created from the single file name provided to the program by adding a `_uart` and `_uart_t` modifier to the data and UART timestamp log, respectively. The extension for the data log is by default always replaced with `.log`.

Listing 5.3: Synchronization of the UART transmission.

```
fprintf(stdout, "\n%s: Synchronizing UART transmission
 ...\n", __argv[0]);
uart_helper.curr_read_byte = serReadByte(handle);
while(((char)uart_helper.curr_read_byte !=
UART_RTS_EOT_CHAR) && start_logging)
  {
  if ((uart_helper.bytes_available =
  serDataAvailable(handle)) > 0)
    uart_helper.curr_read_byte = serReadByte(handle);

  terminate = gpioRead(GPIO_TERMINATE_IN);
  if(terminate || *(int*)arg)
    start_logging = FALSE;
  }
```

The thread first waits for the beginning of a point by detecting the end '>' character (see Lis. 5.1 and 5.3). The RTS is sending its data continuously and

this way, whenever the measuring program starts, the log will always begin with a full point. After this, a continuous `while` loop checks for any available bytes on the UART input. Whenever more than one byte is available, a time value is immediately taken, again, using the `clock_gettime()` function. Now, if the read byte is a valid character, it is saved into the data log file and if the previously read byte was the end `'>'` character, meaning the current byte is the first character of a new point, the acquired time value is saved into the timestamp log file. A predefined number of initial characters from the incoming data is also displayed for verification purposes. The code in listing A.1 shows an example of these procedures. If a termination condition is met (by a `ctrl+c` signal message, button press or a call from the main thread), the thread again waits until an end `'>'` character is detected, then saves all opened log files and exits. This way, the log will also always end with a full point. Abrupt physical disconnections might, therefore, cause the loop to hang indefinitely, however these are not planned to happen. This time-stamping implementation has a drawback however. As it is described, the saved timestamps are really the times, in which the first byte of a measurement point was read by the program from the UART buffer, and not when the particular byte was actually presented into the buffer. This means that an error resulting from the time it takes to read out a character from the UART buffer is added to the time in which it actually was first available.

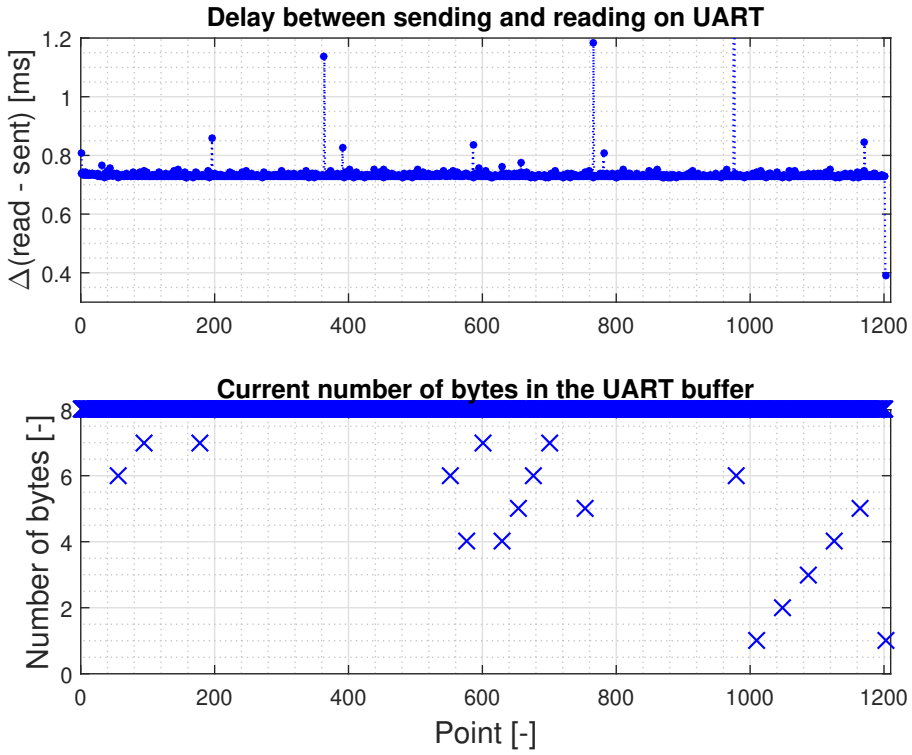


Fig. 5.10: Loop-back test of RPi UART read delays.

This error however, is not expected to reach the order of milliseconds. To evaluate this, loop-back testing was done with the RPi UART port, where the output Tx pin was connected directly to the input Rx one. A simulated RTS data stream was continuously being sent by the main thread and the secondary UART thread was, in parallel, continuously reading the arrived bytes. Each sent out point had an associated sending timestamp and each received point was being ordinarily timestamped by the process described above. From this, an internal RPi-related delay could be estimated, representing the difference between the time of sending a character and the time of reading the character from the UART buffer. The result of one test can be seen in Figure 5.10. The delay is generally below 800 µs and it is reasonable to assume that the time it takes to read a character from the UART buffer, after it had already appeared, should be even smaller that this. Also, a maximum number of 8 bytes was left in the buffer at any point in time, which at a baud rate of 115 200 bits/s would lead to a discrepancy of about 556 µs until a byte is actually read out from the buffer. The total length of the UART buffer is 4096 bytes.

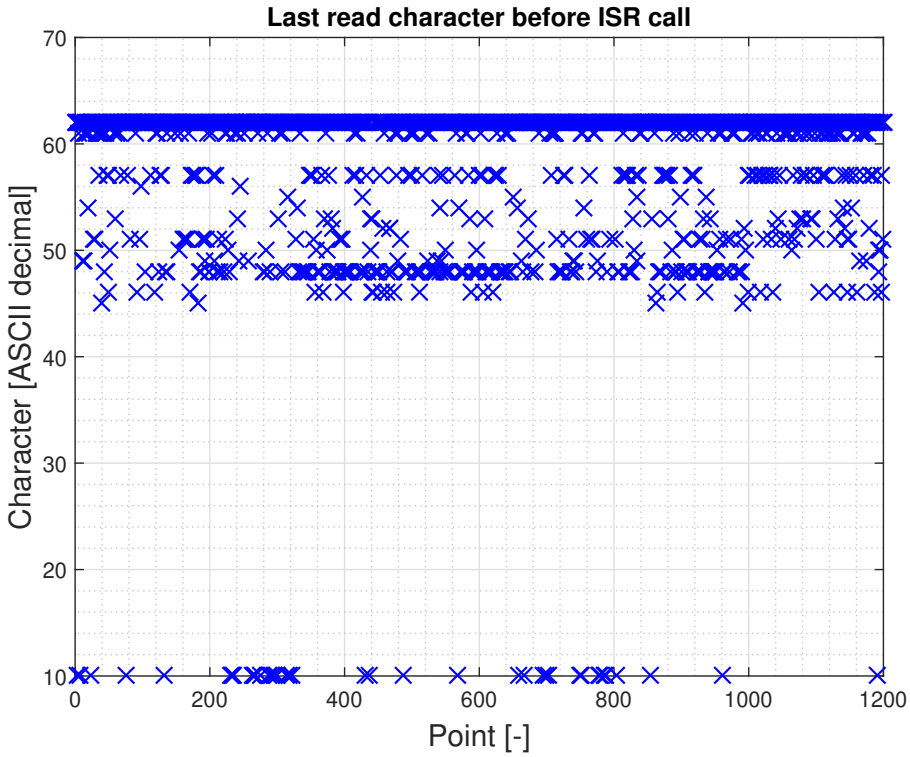


Fig. 5.11: Testing interrupt-driven time-stamping on UART.

An attempt was made to make the UART time-stamping also interrupt-driven, however shortcomings of the chosen HW solution have shown up. The premise was as follows. The incoming data would be split into two paths. One would go directly to the UART interface and the other to a GPIO pin, which would have

an ISR handler associated with it. Whenever an end `'>'` character was detected on the UART interface, the ISR, which would detect the first falling edge and immediately acquire a timestamp, would be enabled. After the handling would be done, the ISR would be disabled and enabled again only when the next end `'>'` character arrives at the UART interface. This way, true time values of when the starting character physically appears on the end device could have been acquired. However, after initial tests, the process of disabling and re-enabling the ISR on the Raspberry Pi ultimately took longer than was the actual period between two arriving data points. The mechanism was not able to keep up and often timestamped characters much later within the point or skipped the full point altogether (see Fig. 5.11). This method was, therefore, abandoned and not pursued any further. The approach was taken from Sama *et al.* [49]. However, they were using a dedicated *Digital Signal Processor* (DSP) with specialized *Input Capture* (IC) interfaces for the time-stamping and the whole process, together with UART handling, was fully ISR-driven.

5.3.5 Summary of the main procedures

A fully connected reference measurement setup ready for experiment can be seen in Figure 5.12. The experiment component layout and data flow can be seen in Figure 5.13. To summarize, the main procedure for the pendulum experiment was as follows.

1. The *Panasonic* light barrier sensor setup was placed on the ground, defining the exact pendulum rest position (see Fig. 5.5).
2. The RTS was placed at the desired distance from the pendulum and aligned as described in Section 5.3.1 (see Fig. 5.6).
3. The *Base Station* point was set as the origin of the coordinate system and the *Back Sight* was measured at the pendulum rest position.
4. Data output from the RTS was started with labels shown in Figure 5.8 and TRK mode was set.
5. Executing the `SetUpEnvironment` script, necessary configuration on the TSC7 was performed.
6. PTP synchronization between the RPi and TSC7 with both Windows logs (see Sec. 5.3.2) was initiated with the `StartMeas` script and provided log file name. The Linux master node was started by the `sync_ptp_master` script.
7. After a first successful clock correction, confirmed using the `QuerySyncStatus` script, the measurement program on the RPi was started by executing the `measurewargs` script and providing the output file name.
8. After the measurement process was finished, first, PTP synchronization was

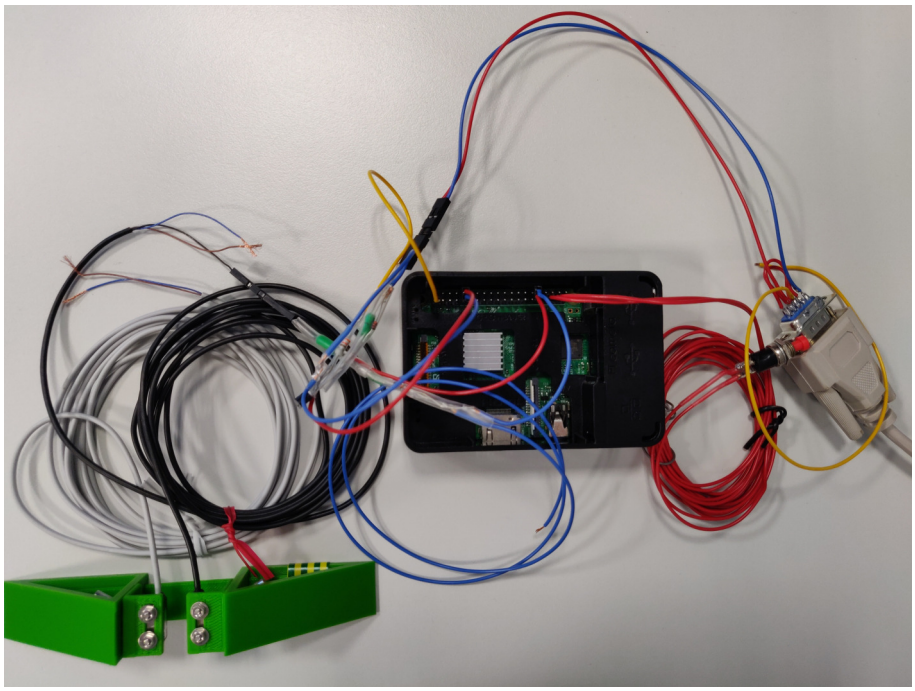


Fig. 5.12: Fully connected reference measurement setup. Panasonic sensor: left, Raspberry Pi: middle, RTS data output: right.

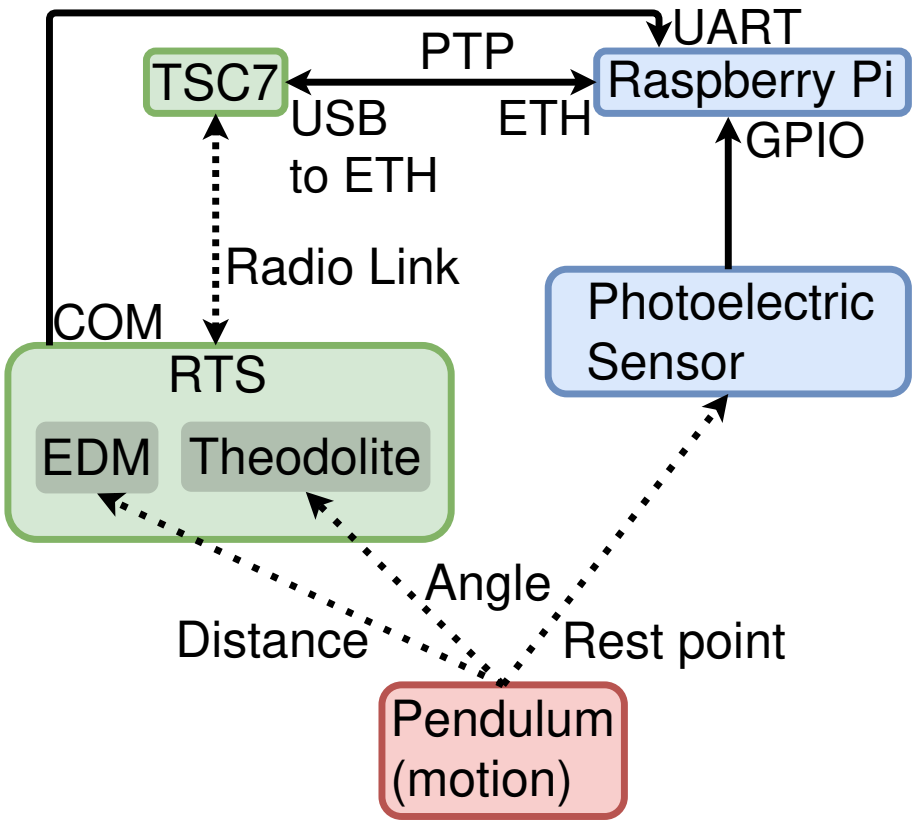


Fig. 5.13: Block diagram of the experiment layout and data flow.

stopped by executing the `StopMeas` script in Windows. This saved the necessary Windows log files (see Tab. 5.1). The Linux master node was left running if any subsequent measurements were to be made.

9. Finally, the measurement program was stopped (by `ctrl+c` termination or a button press). The RTS data output, light barrier timestamp and UART timestamp logs were saved (see Tab. 5.1).
10. If no further measurements were being made, the Linux master PTP node was stopped and the TSC7 controller reverted back to its original configuration by the `RestoreEnvironment` script.

For each of the three measurement distances, at which the main experiment was being done, static measurements were made as well. The pendulum, together with prism, was left stationary at its rest point and a continuous data stream from the RTS was collected with no reference data from the Raspberry Pi. This was done once right after the station setup and alignment (before the experiments) and also after all the measurements were finished. This provided a static baseline for the RTS data and the particular configuration of the components at the time of each measurement.

All the relevant different log types gathered during the experiment are shown in Table 5.1.

Log file	**Associated device**	**Description**
`*.csv`	Raspberry Pi/Sensor	Timestamps of sensor crossings, containing t_{real}.
`*_uart.log`	RTS	Output data from the RTS, also containing t_{meas}.
`*_uart_t.csv`	Raspberry Pi/UART	Data logged by the RPi UART thread, also containing t_{avail}.
`*_w32tm.log`	TSC7	Data logged by the *Windows Time Service*.
`*_w32tm_cst.log`	TSC7	Data logged from the query command output.

Tab. 5.1: Log files acquired during an experiment.

5.4 Additional experiments and evaluation

Apart from the main pendulum motion experiment, several other tests to verify different influences or parameters were done as well.

5.4.1 Measurement rate analysis

A variability of the measurement rate could be seen in figures 4.3 and 4.4. As the data is arriving, it seems that the measuring frequency is not constant, but rather oscillates around the expected value, creating an average that approaches the desired measurement rate. This variability was also seen in direct oscilloscope readings of the outgoing RTS data (see Fig. A.9). Frequency of the RPi timestamps acquired upon data point arrival on the UART interface (t_{avail}) could be compared with the apparent frequency of the RTS time data (t_{meas}) to evaluate their assignment point (see Tab. 5.1).

5.4.2 Aperiodic manual prism motion

Another set of experiments was made for each of the three distances. The basic procedure was the same as for the main experiment. However, the pendulum arm was being moved manually by hand at irregular intervals and different velocities. Reference rest point crossing data was being acquired by the Raspberry Pi and the same timestamp evaluation was performed as well. This provided a more chaotic and unpredictive type of motion in addition to the periodic pendulum oscillations.

5.4.3 Lateral pendulum arm displacement

The prism was left stationary in the pendulum rest point position. Data output from the RTS in TRK mode was again collected by the RPi the same way as with the main experiment, but no reference time data was being collected. The pendulum arm was then perturbed to induce as much vibrations as the structure allows in the lateral direction (*Northing* axis). Ideally, this would again be performed for each of the three experiment distances to verify the effect with each particular setup at the time of experiment. However, only one test was done, where the RTS was placed at about a 4.7 m distance from the pendulum. This test was mainly done to verify the rigidity of the setup and the expected limits for the light barrier length. It does not provide any significant conclusions for the main subject of this thesis.

5.4.4 Horizontal prism rotation

This test was inspired by the research done in [14]. However, the methodology of this test was not as precise and accurate as theirs, and these results should be taken in a more informative rather than definitive way.

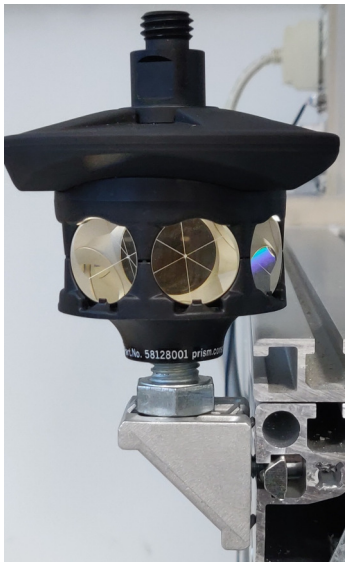


Fig. 5.14: Prism placement for horizontal rotation tests.

These tests were performed at two different distances. The prism was fully screwed down as is shown in figure 5.14 and the RTS was placed on the ground (below the height of the prism) at a horizontal distance of 4.6 m and 1.2 m. The same station setup as with previous experiments was done, the coordinate origin being at the *Base Station* point and the *Back Sight* being the measured prism location. Subsequently a 10 Hz data output, TRK mode measurement was initiated. The prism was then being slowly rotated on the screw by hand. A full 360 degree horizontal rotation (visible prism elements in Fig. 5.14 moving to the right) was performed and then it was rotated backwards (prism elements moving to the left) to the original starting position. One test was done with 720 ° rotations. This was repeated two to three times in one measurement run.

The drawbacks of this methodology are the following. First, the threads on the screw and the hand rotation might create unwanted displacements influencing the rotation-related data. Second, in conjunction with the horizontal rotation, the prism was also moving up and down vertically as it was rotated on the screw, so a full isolation of the effect of rotation might not have been achieved.

A solid relationship between the prism angle and the measurements might not have been obtained here and a different test, where the prism would be precisely

positioned at several specific known angles of incidence and measurements taken without any movement, might also provide more substantial results in the future.

5.4.5 Safe prism velocity limits

Some conclusions can also be made from the fact that the RTS starts loosing data as the initial starting angle, and with it the velocity, of the pendulum arm increases. When the starting angle gets too wide, the RTS is not able to track and looses the prism instantly. Tests with several different starting angles were run and the data already gathered from the experiment runs could also be used to estimate the velocity limits.

This test, again, is not ideal as the prism velocity is not controlled and depends on the starting angle and subsequent oscillation amplitudes. What is more, the velocity values are calculated from the actual RTS measured angles. A much more reasonable and expansive experiment could be made in the future, where a constant linear (or angular) velocity would be maintained and gradually increased to the point of measurement failure. The velocity would also need to be measured implicitly on the moving target as opposed to being calculated from the imperfect RTS measurements.

6 Result processing and evaluation

This chapter will provide the final results from the above extensively described experiments. The data processing methodology will first be explained and then conclusions that can be made from the performed measurements will be stated. Concluding, perhaps unconventionally, with also some other potential alternatives to the presented navigation problem solution.

6.1 Main experiment data interpretation

6.1.1 Data processing and evaluation tools

Table 5.1 showed all the acquired log files that were used for the subsequent data processing and evaluation, which was performed in the *Matlab* environment. These logs, whenever mentioned, will be from now on referred to by their associated device.

RTS log parsing and data extraction

A custom *Matlab* function had to be created to parse data gathered from the RTS, since the proprietary format (see Lis. 5.1) is not easily loaded into the environment. The function `parselog_TrimbleS` reads the RTS log file and creates a matrix from this acquired data, organizing them into a standard spreadsheet-style format, where columns represent all the different quantities (time, angle, distance, *easting*, *etc.*) and rows the individual full measurement points. The function also handles incomplete data (*e.g.* missing quantities) or dropouts in the measurement. For this, apart from the log file location, the function also takes a second argument, which is a vector of all the expected quantity labels in their expected order. A full point, ending with the `'>'` is loaded into a temporary buffer. The quantity labels and their corresponding values are extracted and the labels are referenced with the provided quantity label vector. If at any point during parsing an expected quantity label is missing or the label is missing a value, the corresponding value field is filled with a `NaN` and the user is notified by a warning about the corrupt points and their missing quantity labels (see Lis. 6.1).

Listing 6.1: Example of a found corruption in the RTS data.

```
06-May-2021 20:07:31 Parsing RTS log ...
Warning: Missing data value with label '52' at point 598
(line ~6569 in log)! Missing values filled with NaN,
check for data corruption.
```

```
> In parselog_TrimbleS (line 115)
In analyzeData_scr (line 128)
06-May-2021 20:07:32 Done, time elapsed: 0 min. 1.091648 s
```

The function also informs about an unexpected ordering of the read labels. If more quantity labels than expected are found, it exits, prompting the user to provide a different vector. Using this function, the acquired RTS data could be quickly and efficiently loaded and analyzed.

TSC7 log parsing and data extraction

The TSC7 logs contain vast amounts of information and a parsing scheme had to be devised for them as well. Contrarily, a different approach was taken, where a *PowerShell* script (`ParseLogData`) in conjunction with a simple parsing program written in C was used. The most relevant entries in these logs are the PTP synchronization statistics and error states. These are contained inside the *Windows Time Service* and *query* command logs, respectively. The log files are natively created in *Unicode* formatting and first had to be reformatted to *UTF-8* in order to be used in subsequent steps. The script then uses the `findstr /i` command (with either a `'offset:'` or `'error:'` search phrase) to retrieve content lines containing the required data values and calls the parsing program (`dataparser.exe`), which filters the remaining text, extracts the numerical values and formats them into a final CSV file. This could then be easily loaded into the *Matlab* environment and used for further processing. The parsing program is a modified version of a utility, the original of which had been created and published on github at a time before the writing of this thesis [50].

Time data interpretation

The different time values from all the acquired logs had to be converted into a common uniform frame, in which they could be compared and evaluated. Even though time data from both, the RTS and the TSC7 logs comes from the same TSC7 time source, their value format was ultimately different. Therefore, three *Matlab* functions were created:

- `parsetime_TrimbleS`,
- `parsetime_RaspberryPi`,
- `parsetime_TSC7`,

performing the necessary conversions of the RTS, Raspberry Pi and TSC7 time data respectively. As can be seen from label `'52'` in Listing 5.1, the RTS time data format is `|HH|.|mm|ss|msms|`. These are in a 24-hour local *Coordinated Universal Time*

(UTC) time, respecting the current timezone location. The `parsetime_TrimbleS` function converts them using relation 6.1, where N = $1e3$, into their native milliseconds.

$$(\text{Hours} \cdot 3600 + \text{Minutes} \cdot 60 + \text{Seconds}) \cdot \text{N}, \tag{6.1}$$

where N is the scale, which converts seconds to the desired time unit.

An RPi time data value consists of two parts: the elapsed UNIX epoch seconds and the elapsed nanoseconds since this time. These UNIX epoch time values were converted into the same UTC time frame as the RTS data. The UNIX epoch seconds are easily converted to a UTC time value using the `datetime()` *Matlab* function. Since the RTS time values account for the local timezone, a correct UTC offset has to be known to correctly perform rest of the conversion. During the writing of this thesis a *Daylight Saving Time* (DST) clock shift had occurred, resulting in some of the logs requiring different UTC offsets. So to make the conversion convenient and universal across all logs, the `parsetime_RaspberryPi` function was made to take a date value argument, from which it can automatically determine the correct UTC offset, as well. This date is specifically taken from the RTS data log (see Lis. 5.1, label `'51'`). This way, the conversion is always tied to the particular time frame of the RTS log, respecting its timezone. Finally, the resulting time values can be converted using relation 6.1, where N = $1e9$ and applying the correct UTC hour offset, into nanoseconds and joined with their nanosecond part.

The *Windows Time Service* logs maintained a timestamp value for each new appended entry. These values were not particularly used for any evaluation and they were only useful when plotting the actual relevant PTP synchronization statistics to maintain a common horizontal axis with all of the other graphs. If they were to be used this way, however, they still had to be converted since their default format was UTC+0 with a 7 decimal place fraction of a second. Hence, the `parsetime_TSC7` function also takes a date value argument to calculate the correct UTC timezone offset and converts the time values into nanoseconds using relation 6.1, where $N = 1e9$.

RTS data processing

A *Matlab* script (`analyzeData_scr`) globally handles all the data processing and outputs the resulting graphs or statistics. After all data from the logs had been parsed and loaded into the environment, the actual processing could be performed. If the data contained any initial or ending points with a corrupt or missing time value, they were cut off. A warning message informs of any such corruptions remaining in the middle of the data.

The angle values were converted from gon units to degrees (see Sec. 2.2). They were also shifted from a $\alpha \in [0, 360]$ ° range to $\alpha \in [0, \pm 180]$ °, where the zero reference azimuth had already been conveniently set up during the experiment. The vertical starting angle was shifted from the zenith to the pendulum rest point and flipped so that the angle values increase upwards from the rest point. This made for a more convenient visualization of the values.

Figure 6.1 shows an example of the performed pendulum motion in local coordinate space. In can be seen that oscillations were being done in the *Easting* axis and, in this particular example, a displacement of max. 13 mm was present in the *Northing* axis as a result of imperfect alignment of the RTS. This slight rotation should have a minimal impact on estimation of the rest position. An example result of a near-perfect perpendicular alignment can be seen in Figure A.4.

Figures 6.2, 6.3 and 6.4 show the angle, timestamp and *Easting* coordinate values from this particular run of the experiment. A clear indication of data loss is visible from all of these figures. This log did not have any missing or corrupt values and it is in fact all the data as it was arriving from the RTS. A further discussion of the this and the dependency of data fidelity on distance, prism velocity and the starting angle will be presented in Section 6.3.5.

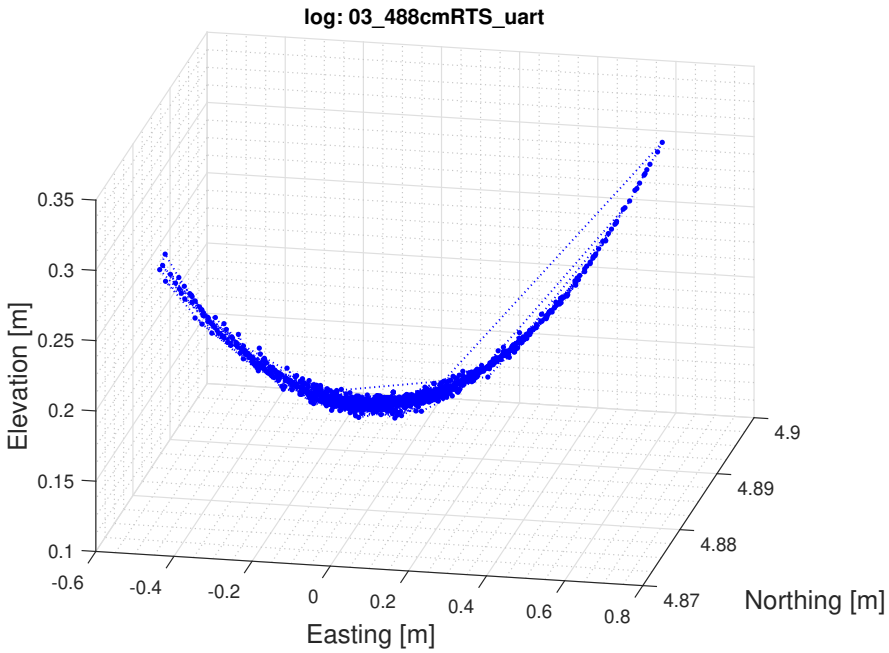


Fig. 6.1: Prism motion in local coordinates.

As can be seen in Figure 6.4, interpolation of the *Easting* data had been performed. The points arriving from the RTS were varying in frequency and there was no guarantee that they would always collide with the precise pendulum rest point.

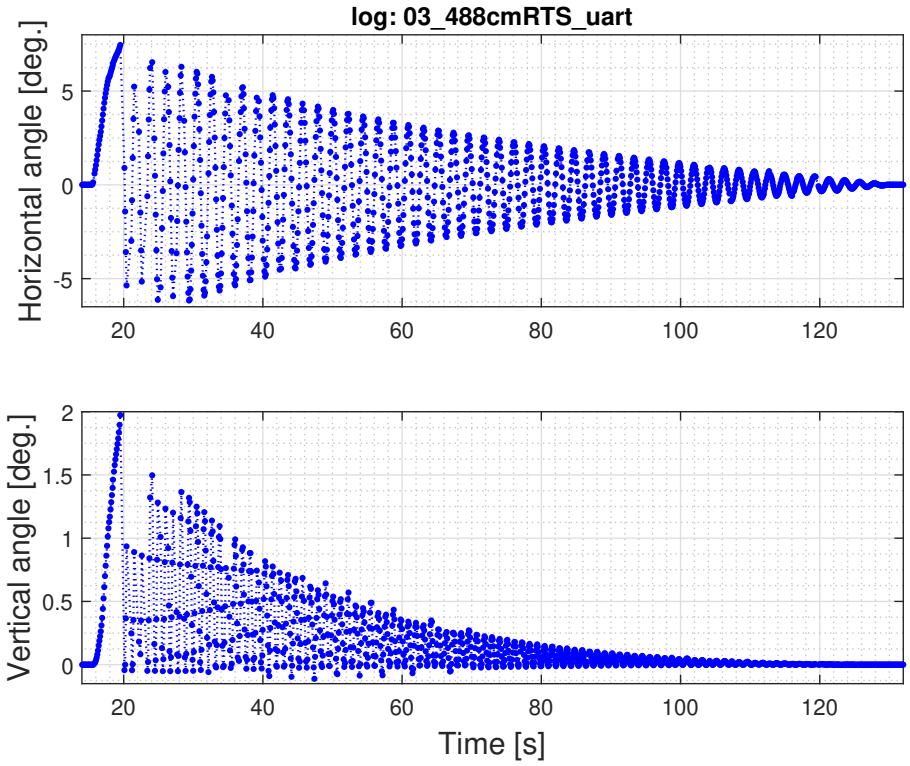


Fig. 6.2: S9 HP RTS angle values in time (from motion start to end).

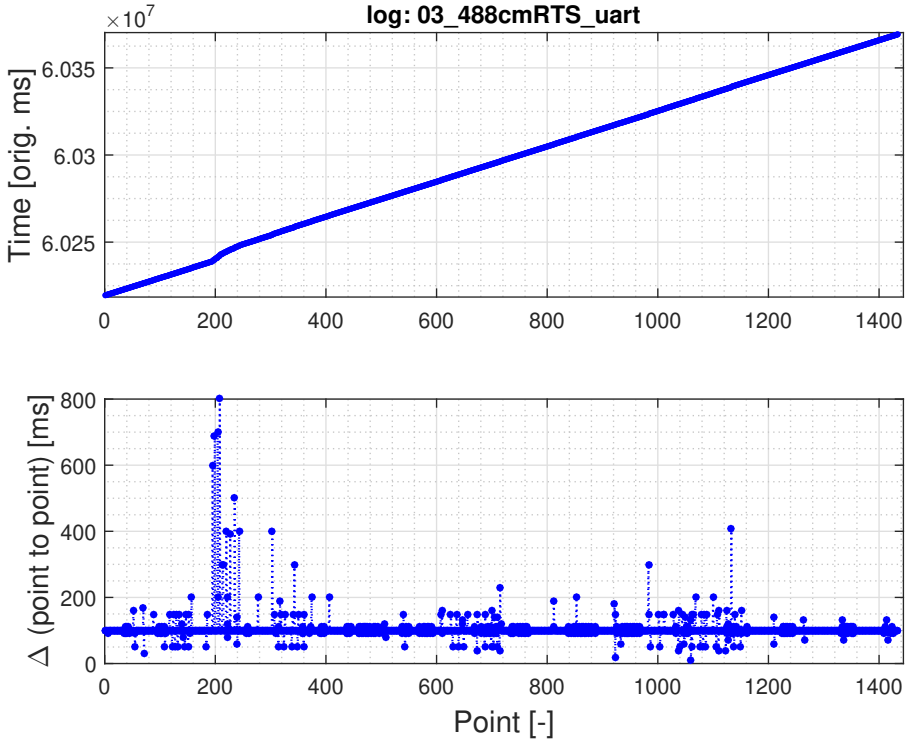


Fig. 6.3: S9 HP RTS time data (full log duration).

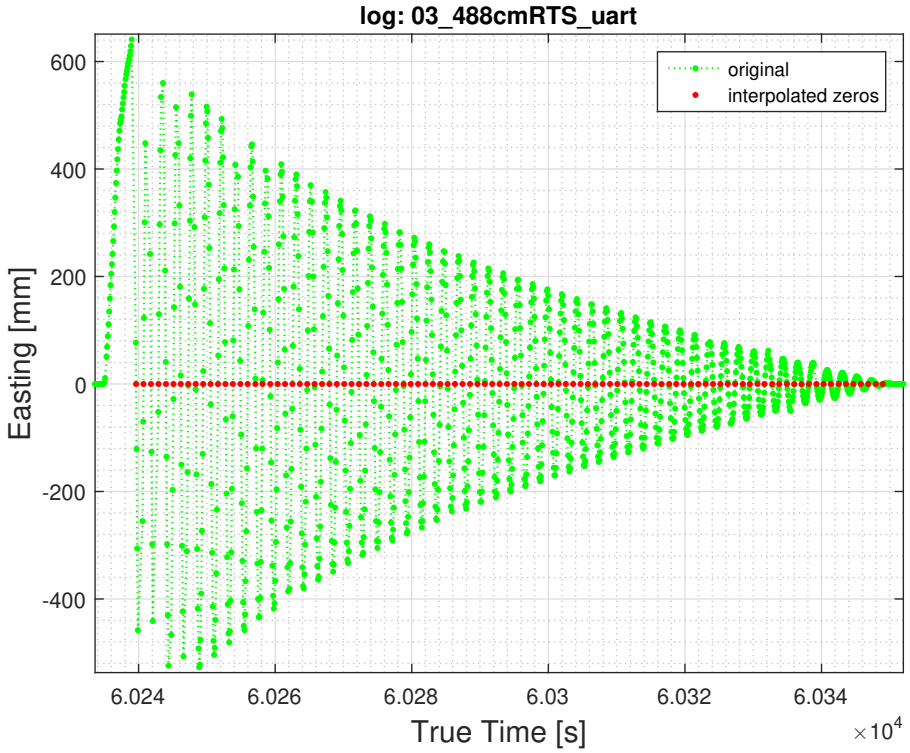


Fig. 6.4: S9 HP RTS Easting data with interpolated zeros.

That is why the exact rest point values were obtained by interpolating between each zero-bounding pair of the RTS data points. A closeup of this is shown in Figure 6.5. Since ENU coordinate data is of particular interest for the purposes of UAV navigation, *Easting* values were used together with the timestamps for this interpolation. A point might be made that the horizontal angle values would be more accurate and reliable data to use, however no significant difference was observed in the results.

Before the interpolation begins, a starting and ending point is automatically found within the data. The beginning is easy to find as it will always be the maximal achieved distance (or angle) and the experiment was always, conveniently, started from the same direction. The end is obtained by finding the first point where no significant movement beyond a certain threshold (in order to account for noise) is observed. This step is visualized in Figure A.5. Subsequently, the closest-to-zero points within this area are found for both the positive and negative side (see Fig. A.6). In some cases there might be a point which does hit the rest position at zero exactly, in which case no interpolation is being performed for that section and the true zero is used. A simple linear interpolation is sufficient if the sections are small enough. Typically, a section was about 8 cm long.

Millisecond time values obtained from the `parsetime_TrimbleS` function and meter position values are converted to nanoseconds and millimeters respectively.

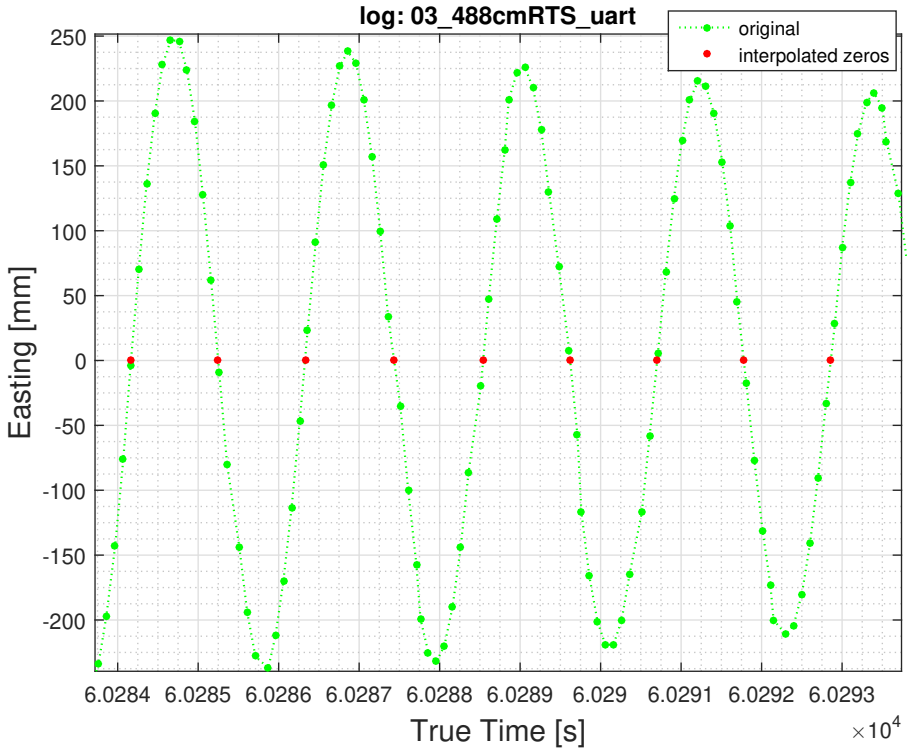


Fig. 6.5: S9 HP RTS Easting data with interpolated zeros (closeup).

Within each zero bounding two-point section, positions are linearly spaced out by a step of 1 µm (or generally one one-thousandth of the input value). Time values are then interpolated onto each of these points. This way, all zero-crossing sections are densely populated by interpolated data (see Fig. A.7). Generally, this method had always found an exact zero position, however if for any reason that was not the case, the value closest to zero was taken. All of these steps are done by the created `interp0pass` and `get0points` functions, where the final output, apart from other data, is a matrix containing the zero position with its corresponding time value for each rest point crossing. This process is also able to handle imperfect or partially incomplete forms, especially at the beginning, but naturally these points are not going to be trustworthy since the resulting sections might be too long, or a single point could be assigned where two physical passes happened. However, this would be a result of incomplete data and not an inherent fallacy of the interpolation method. The interpolated points were visible in figures 6.4 and 6.5. The resulting timestamps are already expressed in nanoseconds and can be directly compared with the RPi sensor timestamps to estimate the RTS measurement delay (Δ_{meas}, Fig. 4.2).

The last processing step is estimating the expected theoretical error given by the potential asymmetricality of the reference measurement setup. As the prism-mounted extension is crossing the light barrier from the left side (*i.e.* returning

pass), it will break the barrier at a different position (and earlier time) than from the right side (*i.e.* forward pass), and therefore slightly offset timestamps might be acquired. This is easily visible from the old setup seen in Figure B.1 on the right and Figure 6.6.

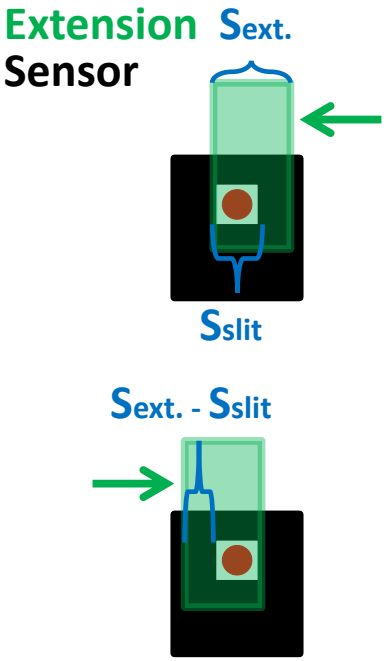


Fig. 6.6: Derivation of the setup asymmetricality error.

The extension cannot be narrower than the sensor's emitting and receiving slits since it must completely obstruct the beam in order to trigger a change. The actual setup used for these experiments, which was presented in Figure 5.3 minimizes any potential occurrence of this error by using en extremely thin slit sensor and making the extension as thin as possible (see Sec. 5.1). Since the width of the extension is known and the velocity of the pendulum near the crossing point can be estimated from the densely interpolated values, the time which it takes to move from a barrier-breaking position of the returning pass to the barrier-breaking position of the forward pass can easily be calculated. This distance is a very small portion of the oscillation and it can be considered linear without any curvature. Therefore a simple linear velocity equation can be used for the calculation (Eq. 6.2).

$$t_{\mathrm{err}} = \frac{\mathrm{S}_{\mathrm{ext.}} - \mathrm{S}_{\mathrm{slit}}}{v_{\mathrm{prism}}}, \tag{6.2}$$

where

- t_{err} is the estimated expected error,
- $\mathrm{S}_{\mathrm{ext.}}$ is the width of the extension,
- $\mathrm{S}_{\mathrm{slit}}$ is the width of the sensor's slit,

- v_{prism} is the prism velocity near the rest point.

This resulting asymmetricality error is introduced to the data as a correction, by adding it to each corresponding RPi timestamp. Both, corrected and uncorrected values will be plotted together in the following graphs.

6.2 Main experiment results

6.2.1 Raspberry Pi timestamp validation

The RPi timestamps can also be evaluated using the ISR ticks provided directly by the *PiGpio* implementation at the sensor interrupt event after shifting both of them to a zero start value. These are defined as microsecond ticks since boot time (see Sec. 5.3.4). Since both types of these stamps were taken at the same event, they should not deviate from each other significantly. Figure 6.7 shows their difference (the first point is zero since both were shifted to a zero start) and it can be seen that they did not deviate by more than 300 µs throughout a performed experiment. Other experiment runs showed the same result. As there is no significant deviation, the local time timestamps acquired by the `clock_gettime()` function could be accepted as reference for the RTS measurements.

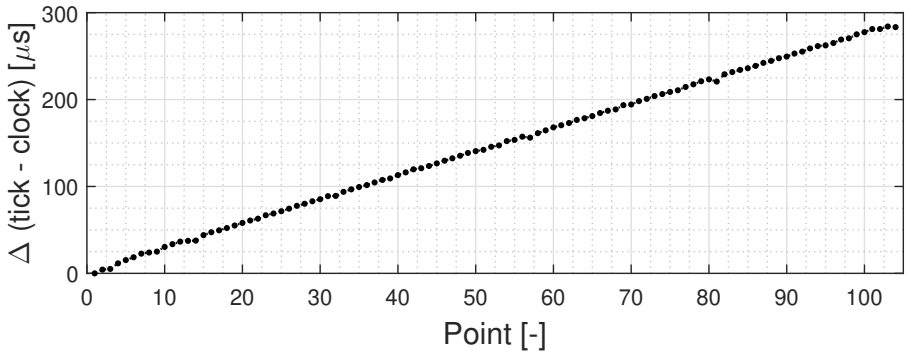


Fig. 6.7: Comparison of two different time values acquired at the RPi ISR event.

6.2.2 Single main experiment run overview

The RTS position measurement delay (Δ_{meas}, see Fig. 4.2) was established as the difference between the RTS and the reference RPi timestamp ($t_{\text{meas}} - t_{\text{real}}$). Both time series were parallel shifted to start according to the first RPi value. Since a delay is expected, these differences should be positive. Figure 6.8 shows results from the single experiment run that has been used as an example so far. Figure 6.9 presents the estimated delay in a histogram. The aforementioned corrections are also displayed.

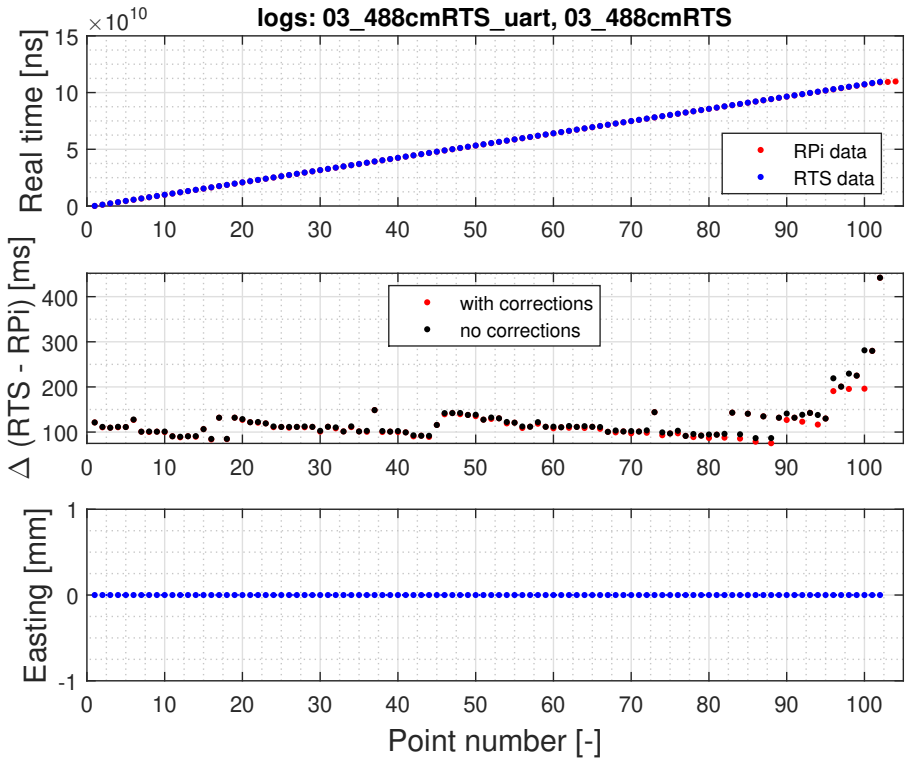


Fig. 6.8: Comparison of the RTS and RPi timestamp values (single run).

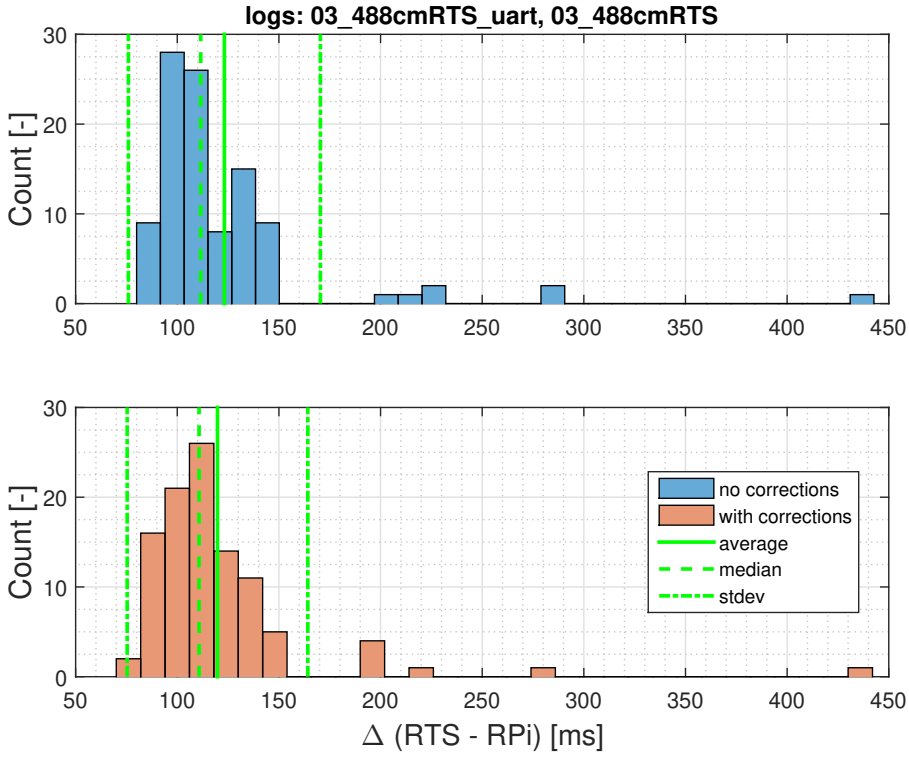


Fig. 6.9: Histogram of the RTS position measurement delay (single run).

Figure 6.10 shows a comparison between the delays of forward and returning zero point passes before any applied corrections. An expected offset with the returning passes creating a higher final delay because of the earlier timestamp acquisition is seen, however the difference is small and the setup was not far from being symmetrical.

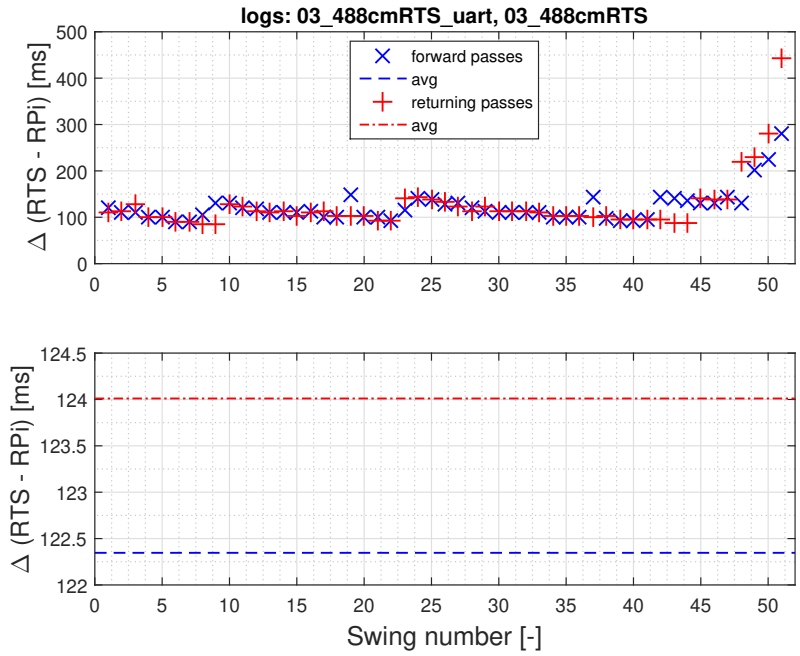


Fig. 6.10: RTS position measurement delay with split passes (single run).

These graphs contain all the experiment data together with the sparser beginning section, which might not provide trustworthy results. The time delay in this experiment run clusters around the 100 ms mark with a median of 112 ms before corrections. An average of the calculated expected asymmetricality errors was 7 ms with a maximum of 84 ms at the last point where the velocity was the smallest. It can be seen that the RTS measurement delays tend to increase rapidly towards the end section as the pendulum moves gradually slower until its complete stop. This was observed in every experiment run. This may have been caused by the prism covering gradually shorter distances from the central rest point, effectively forcing the RTS to operate further within its highest error range due to the experiment layout (see figures 5.6 and 2.4). However, a definite conclusion cannot be stated and more experiments with varying RTS positions and orientations could have been done to evaluate this hypothesis. Since the expected asymmetricality error also rises with decreasing pendulum velocity as it takes more time for the extension to cover the error distance (see Figure 6.11), the ending section was not trusted for the final evaluation.

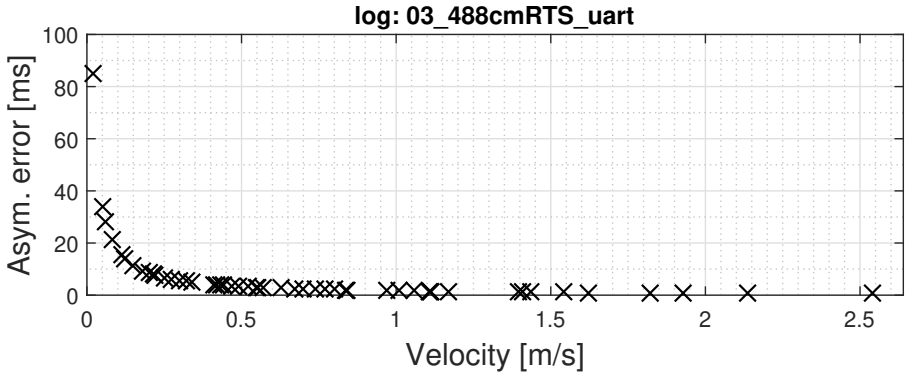


Fig. 6.11: Asymmetricality error with respect to prism velocity.

From the above stated, it is clear that both the beginning and ending sections might not provide trustworthy results, therefore a middle part from the movement was always taken, starting from a point with dense enough original data and ending with a point where the calculated asymmetricality error is not higher than 20 ms.

The data propagation delay (Δ_{prop}, see Fig. 4.2) was established as a difference between a RPi UART and RTS timestamp ($t_{\text{avail}} - t_{\text{meas}}$) for each of the points in the data set regardless of a pendulum rest position pass. This can be seen in Figure 6.12. This delay was around 24 ms.

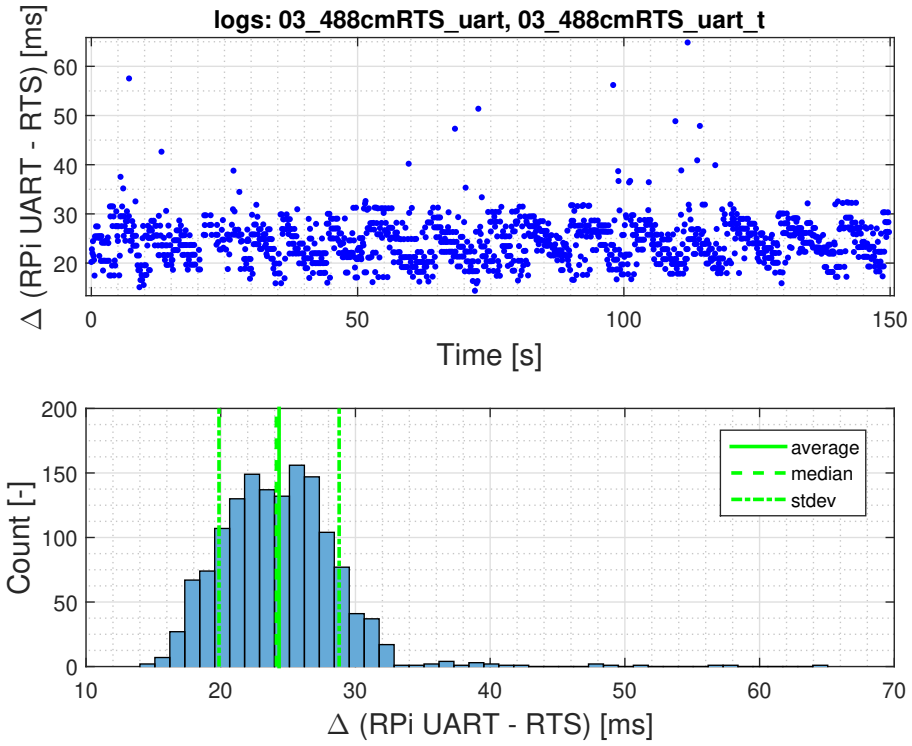


Fig. 6.12: Comparison of the RPi UART and RTS timestamp values (single run).

PTP behavior can be seen in figure A.8. After the initial spike, all synchroniza-

tion offsets were within 200 µs throughout the whole experiment. A single error was detected by the query command. Results obtained with the old setup within the semestral part can be seen in supplement B.

6.2.3 All main experiment runs evaluation

Several runs of the main experiment as it was shown in Section 6.2.2 were performed at three different distances (approximately 1.5 m, 2.7 m and 4.9 m). For each distance around four to six runs were done with a wider and narrower starting angle resulting in a total of 34 experiment runs, some of which contained unusable data. For easier comparison of the individual runs, each will be encompassed into a single boxplot, where the top and bottom box boundaries indicate the 75th and 25th percentiles, respectively. The mark shows the median value and whiskers extend to maxima and minima before outliers, which are plotted independently. Only data without the beginning and ending sections was processed for all of these results.

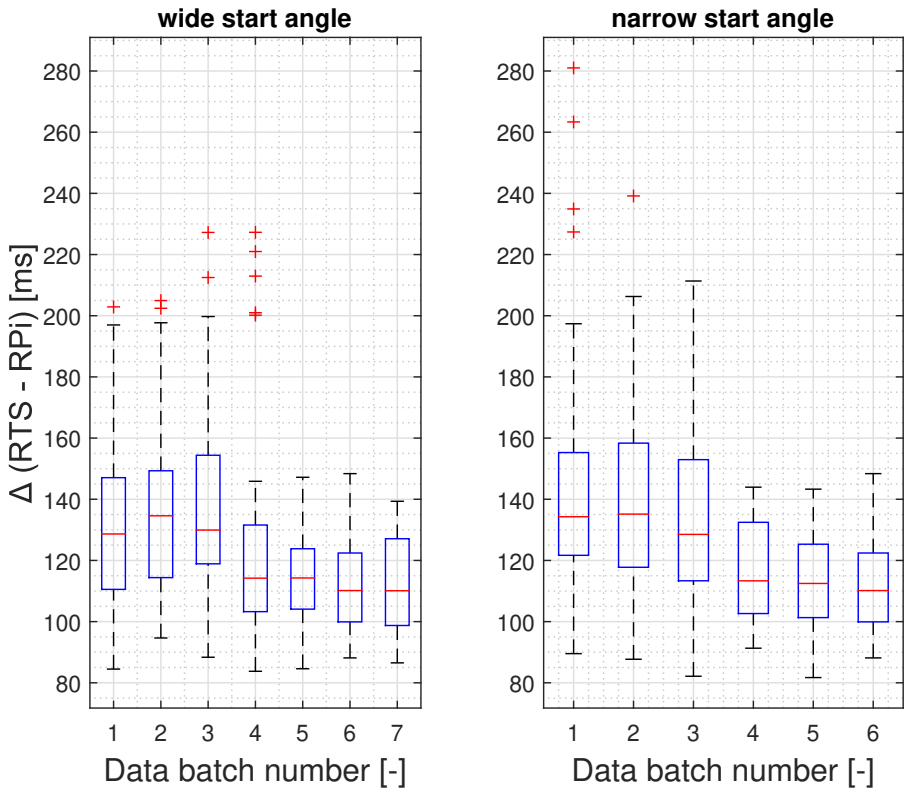


Fig. 6.13: RTS position measurement delay (all runs, 4.9 m distance).

As can be seen in figures 6.13 and 6.14, all the experiment results vary around a 110 ms mark. The first three results in Figure 6.13 in both sections were obtained with a more perpendicular RTS alignment (see Fig. A.4) than the other runs, with only a 5 mm displacement in the *Northing* axis. All of the other experiment runs had

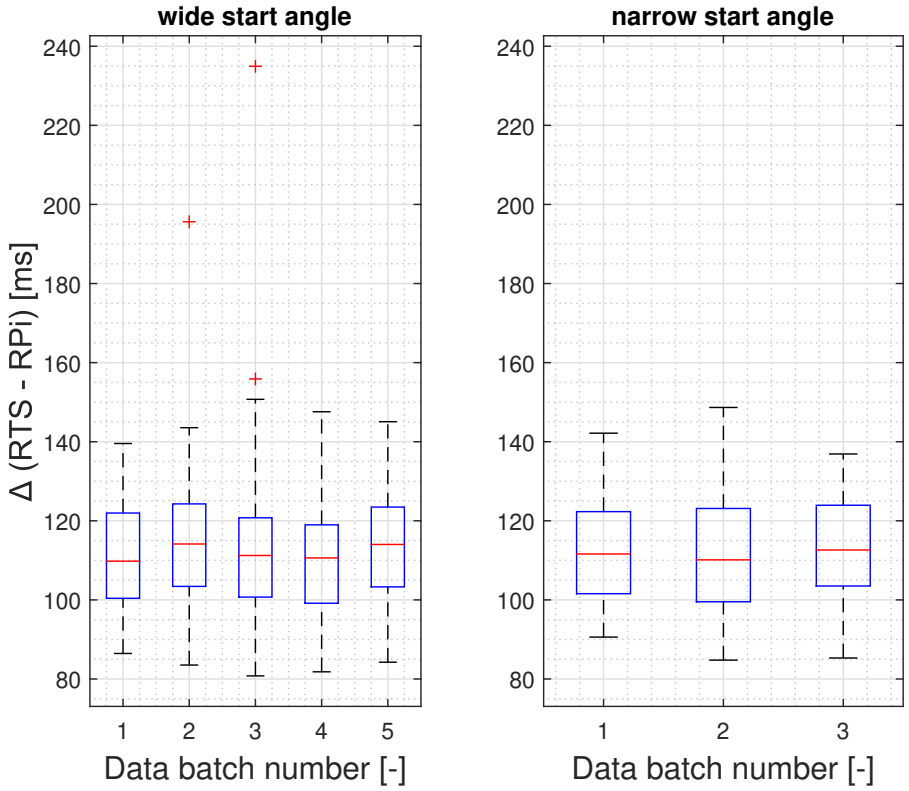


Fig. 6.14: RTS position measurement delay (all runs, 2.7 m distance).

a displacement between 10 mm and 20 mm. However, it is not clear whether only the alignment had any effect on the higher delay results. The reader might also be, misleadingly, tempted to compare the 280 ms outlier in Figure 6.13 on the right with data displayed earlier in Figure 6.8, however that particular shown experiment run corresponds to batch number 6 on the left side. Figure 6.15 only shows the narrow starting angles since at the 1.5 m distance, the RTS was not able to maintain lock on the prism with a wider start angle and always lost track of it. Some outlying values, which were contained within the max. 20 ms expected asymmetricality error (see Sec. 6.2.2) cutoff are also visible.

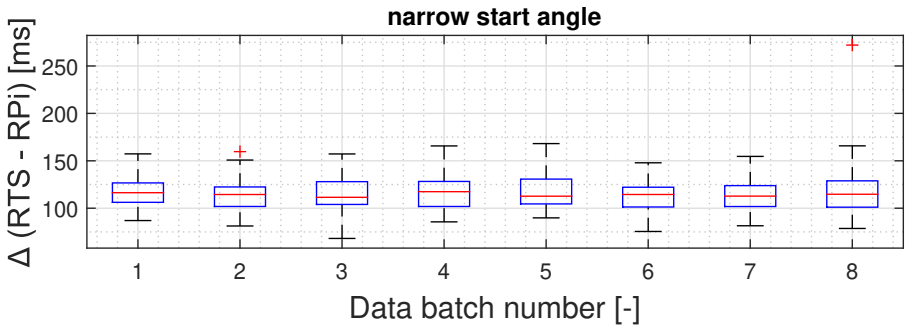


Fig. 6.15: RTS position measurement delay (all runs, 1.5 m distance).

A combined histogram of the RTS position measurement delay for all the experiment runs at all three distances is shown in Figure 6.16 and a combined histogram of the propagation delay in Figure 6.17. Beyond axis limits and not shown are outlying values with counts smaller than 5. Again, results from the old setup are shown in supplement B.

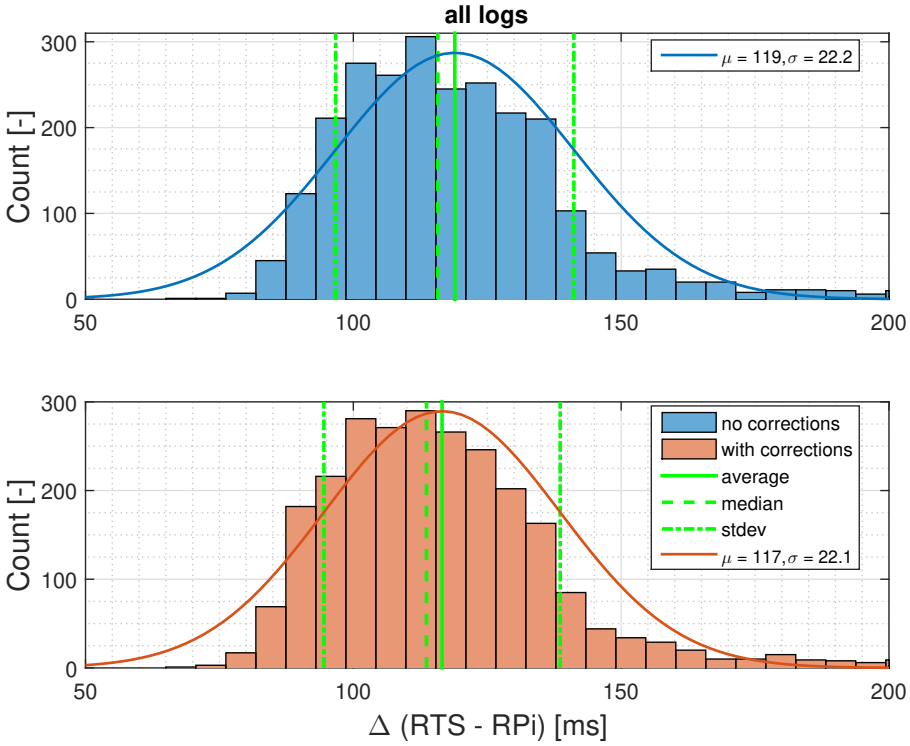


Fig. 6.16: Histogram of RTS position measurement delay (combined results).

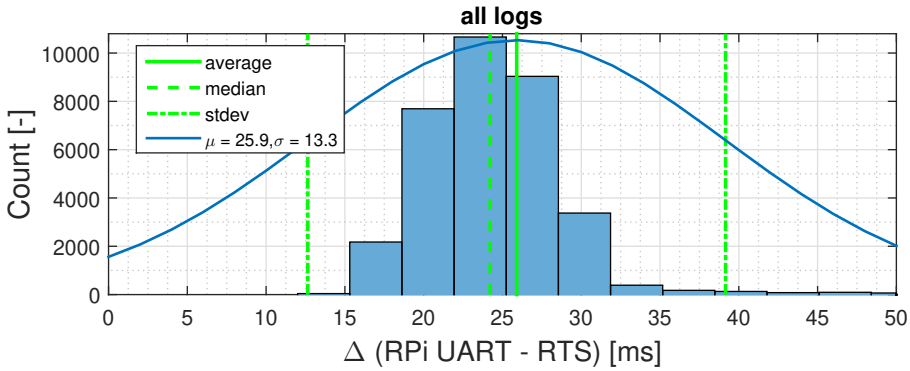


Fig. 6.17: Histogram of RTS data propagation delay (combined results).

Measurements, where the prism remained stationary at the pendulum rest point were used to verify the behavior of the setup. This had been done twice for each RTS distance, before and after any measurements. No significant differences between

these tests were observed and as expected the *Easting* and *Elevation* coordinate values stayed at a constant zero throughout the whole measurement. In each case before experiments, minor drifts were observed in the *Northing* axis with a maximum 5 mm deviation in two cases, which is also visible from the EDM slope distance measurement shown in Figure 6.18 on the bottom (values where shifted to start at zero). For measurements done after experiment runs, no drift was present and only offsets of $\pm$ 2 mm max. were seen. All angle values were staying within thousandths of a degree. The distance (and with it naturally all three coordinate) vales change with a quantization step of 1 mm, which is in line with the $<$ 2 mm precision given in the documentation. The observed accuracies were also all within the given parameters.

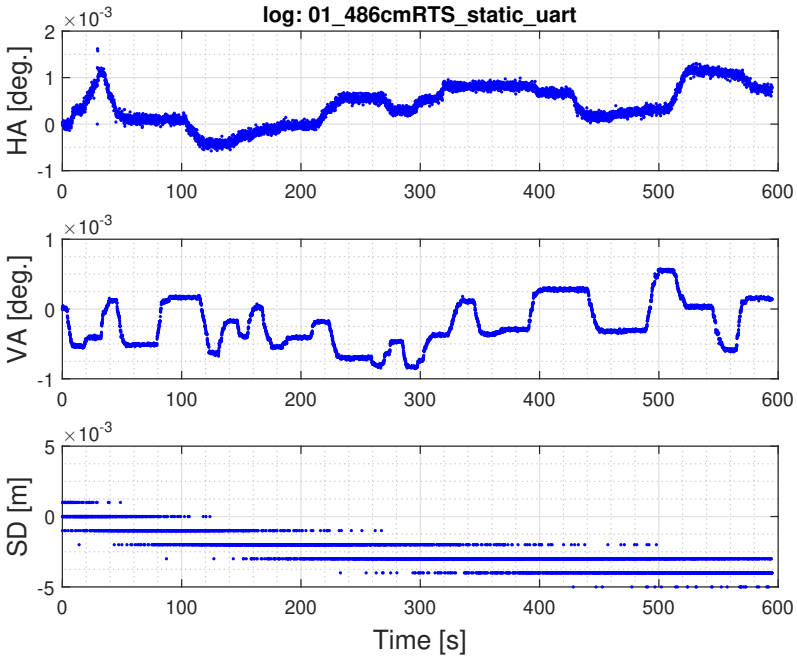


Fig. 6.18: RTS data with prism stationary at rest point.

6.3 Additional experiments results

6.3.1 Measurement rate analysis

As can be seen in figures 4.4 and A.9, the intervals between points sent out by the RTS seem to have a similar variability to that of the timestamps within the data. RPi UART and RTS timestamp values throughout the whole logging interval of the experiment runs were analyzed. Point-to-point differences of both time series were compared and Figure 6.19 shows a correlation between these two obtained

frequencies for a particular experiment run. On the right is a distribution of the difference between them, gathered from all experiment and test logs.

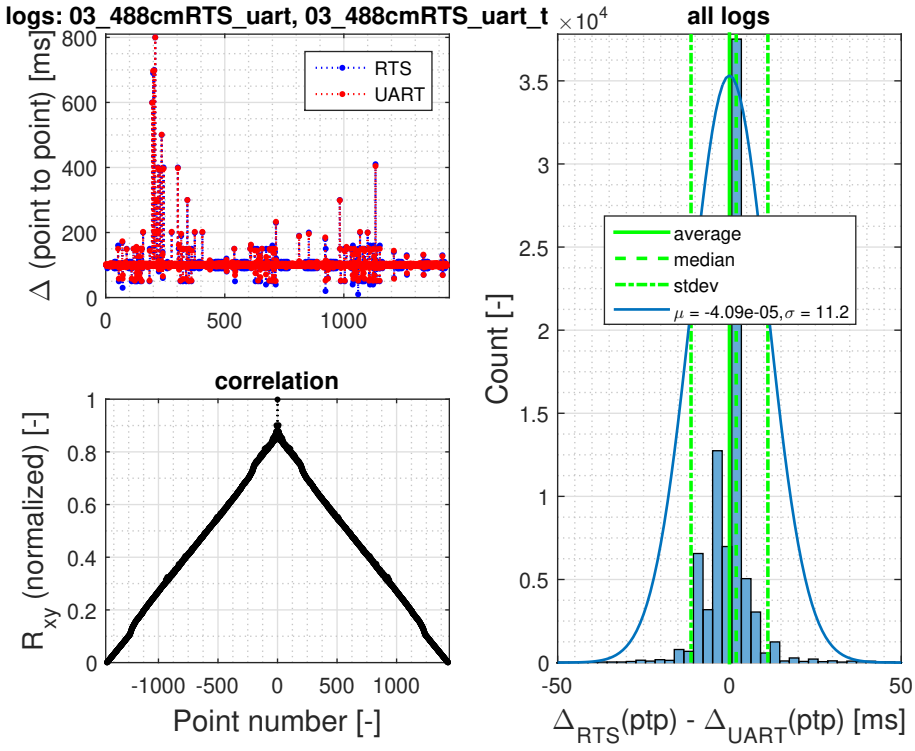


Fig. 6.19: Comparison of RPi UART and RTS timestamp frequencies.

Two assumptions can be taken from this. If the timestamps are being assigned at the moment of measurement, this means that the data has to be sent out immediately after a measurement is done. Or they are only assigned at the moment the data is being sent out through the serial interface, which would mean that the exact time of measurement is still unknown.

Additionally, these measurement periods seemed to be generally higher when the prism was moving compared to stationary measurements, however no additional tests were done to further evaluate this observation.

6.3.2 Aperiodic manual prism motion

For each RTS distance, a set of four tests was made with manual movement of the pendulum arm at varying velocities, as can be seen in Figure 6.20. From a total of 12 runs, all data was usable. Figures 6.21 and 6.22 show the results. Again, the delay values cluster around the 110 ms mark. No data is beyond axis limits. Figure 6.23 shows the data propagation delay from these tests. In this case, values with counts less than 5 are again, not shown.

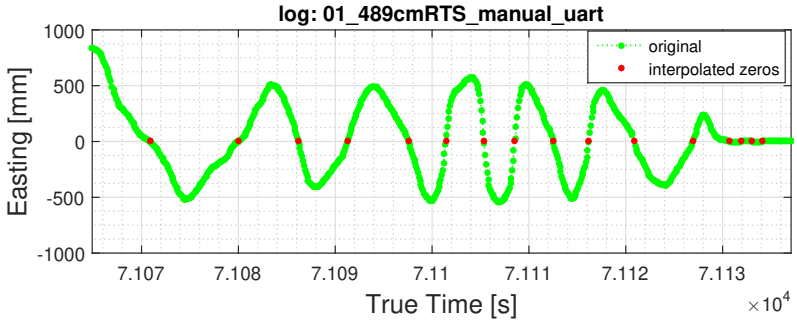


Fig. 6.20: Example of manual prism motion.

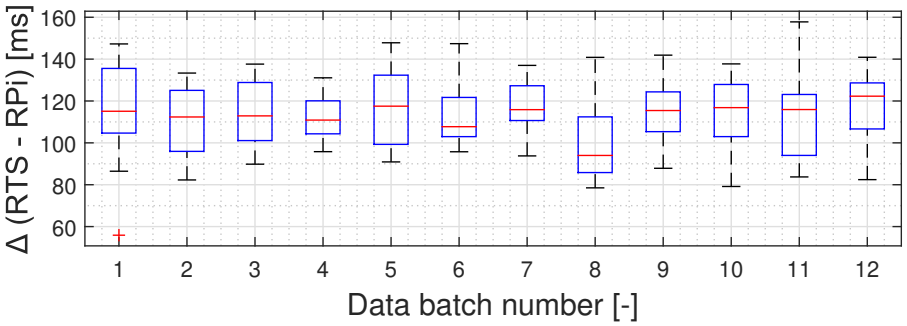


Fig. 6.21: RTS position measurement delay (manual motion).

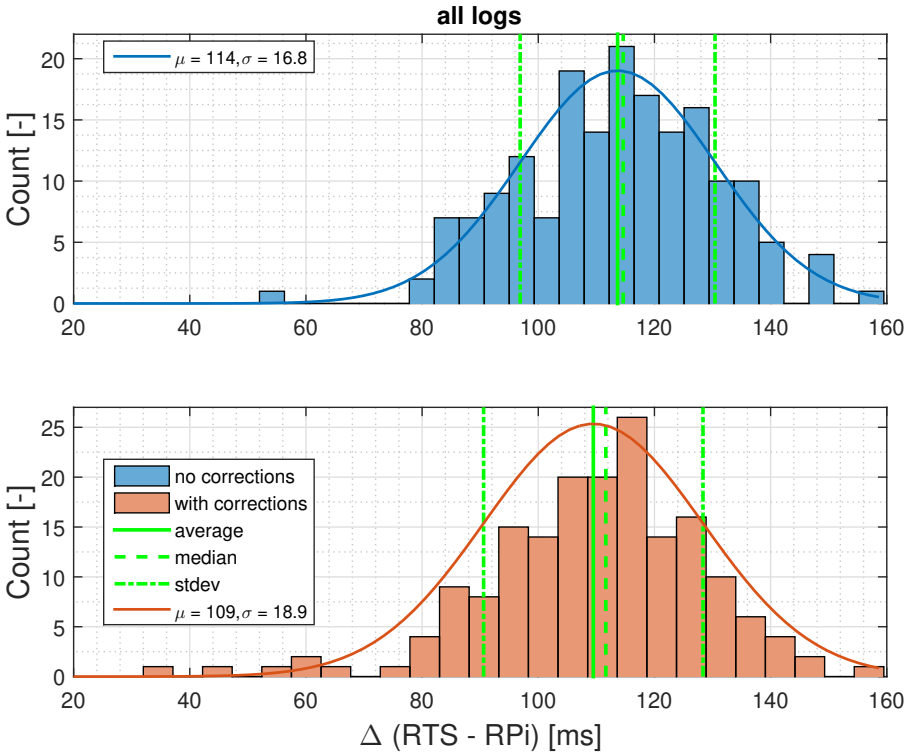


Fig. 6.22: Histogram of RTS position measurement delay (combined results, manual motion).

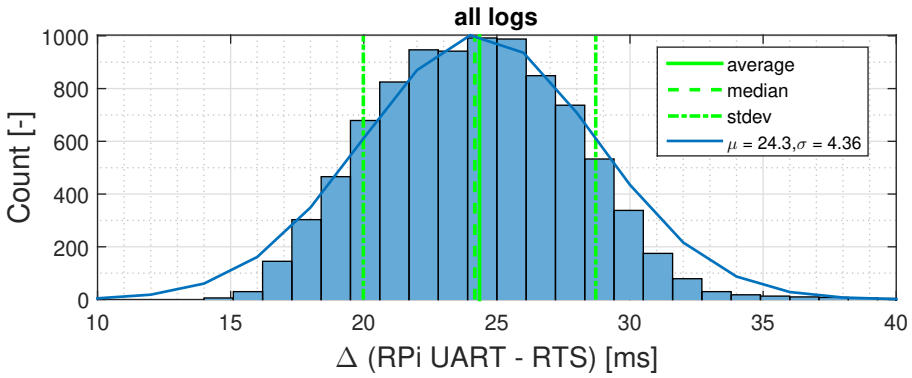


Fig. 6.23: Histogram of RTS data propagation delay (combined results, manual motion).

6.3.3 Lateral pendulum arm displacement

To test the rigidity of the pendulum arm in the lateral direction, a simple test, where it was manually perturbed to vibrate as much as possible at the rest point, was made. As a consequence, slight oscillations in the *Easting* axis were also created. However, the most relevant result shown in Figure 6.24 is that even with a forceful perturbation, displacement in the *Northing* axis was not larger than several millimeters and the construction should not have interfered with the experiment results significantly. These displacements are expected to be even smaller when the pendulum is set to oscillate naturally in the *Easting* axis. The physical light barrier length was set to 5 mm, which in conjunction with the 2 mm rod extension leaves a free space of only 3 mm for such oscillations. This was not exceeded as the rod extension was not observed to break during the experiments.

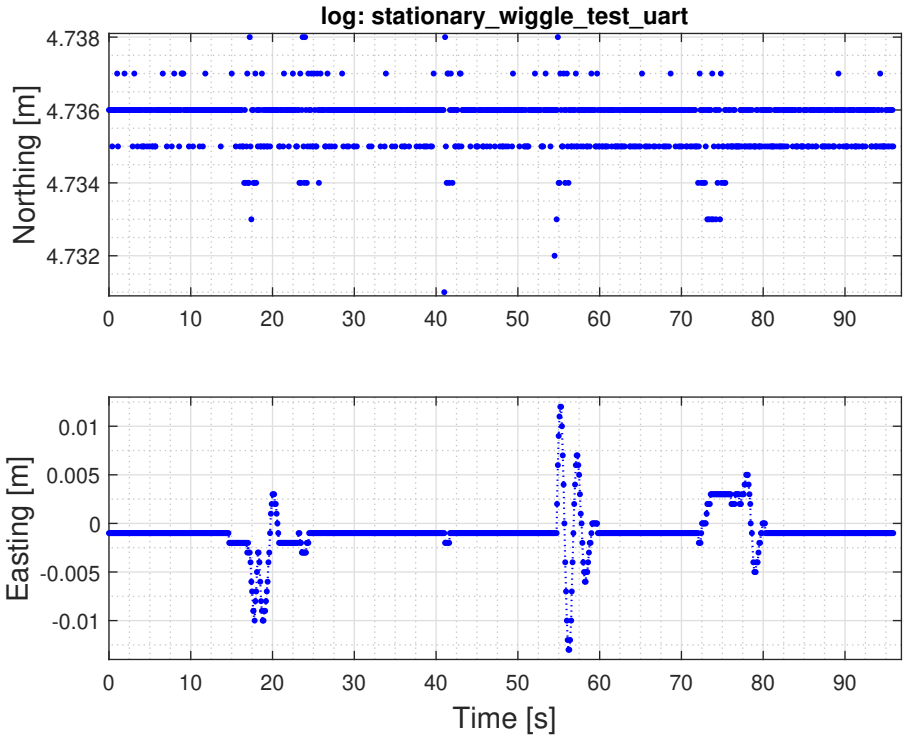


Fig. 6.24: Maximum pendulum arm displacement in the *Northing* axis.

6.3.4 Horizontal prism rotation

The final additional set of tests was done to verify the effect of prism orientation on the RTS measurements. Some conclusions might be made, however better designed experiments should be performed for stronger and definite correlation estimations (see Sec. 5.4.4). The main outcome here is a maximal measured deviation of angles and distances while the prism rotates.

Several consecutive 360 degree rotations back and forth within one test run were made, with a total of three test runs. First two were done at a 4.6 m distance, the third one at 1.2 m. Results from one of the runs are shown in figures 6.25 and 6.26. Jumping behavior between the prism elements is clearly visible, however a further rotation angle correlation was not made. A maximum deviation of both axes was 4-5mm. At 1.2 m the jumps increased up to 10 mm (Fig. 6.27). A fourth test with double rotations (720°) back and forth was made, however there was no difference in the results. Given the chosen main experiment methodology (see Sec. 4.3), these deviations might also be reflected in the resulting time delay estimations as the angle of incidence on the prism was changing during its motion.

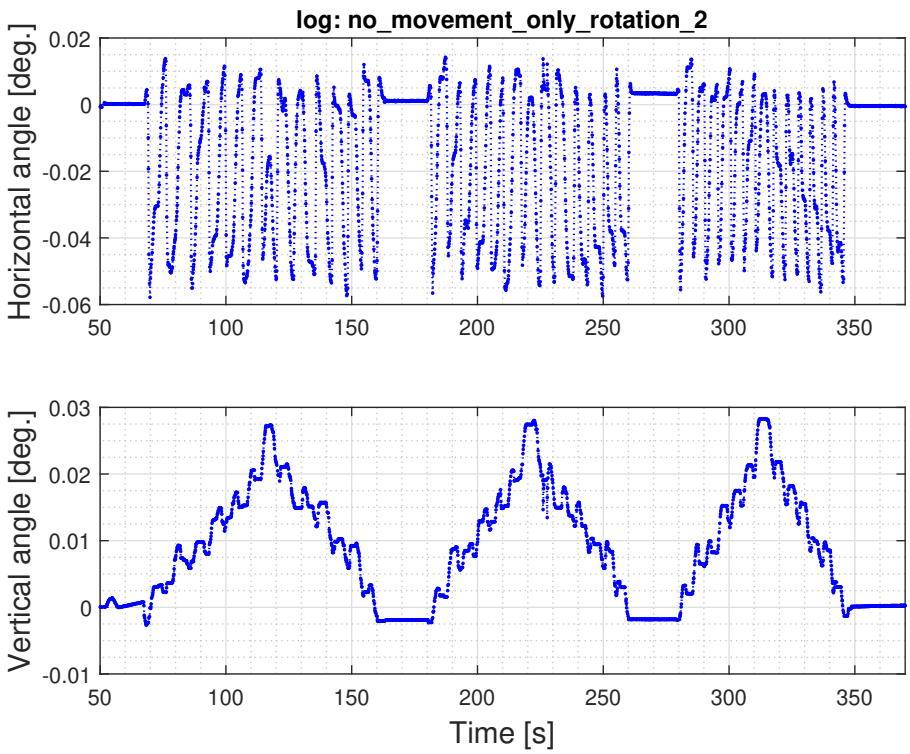


Fig. 6.25: Angle deviation with horizontal prism rotation (4.6 m distance).

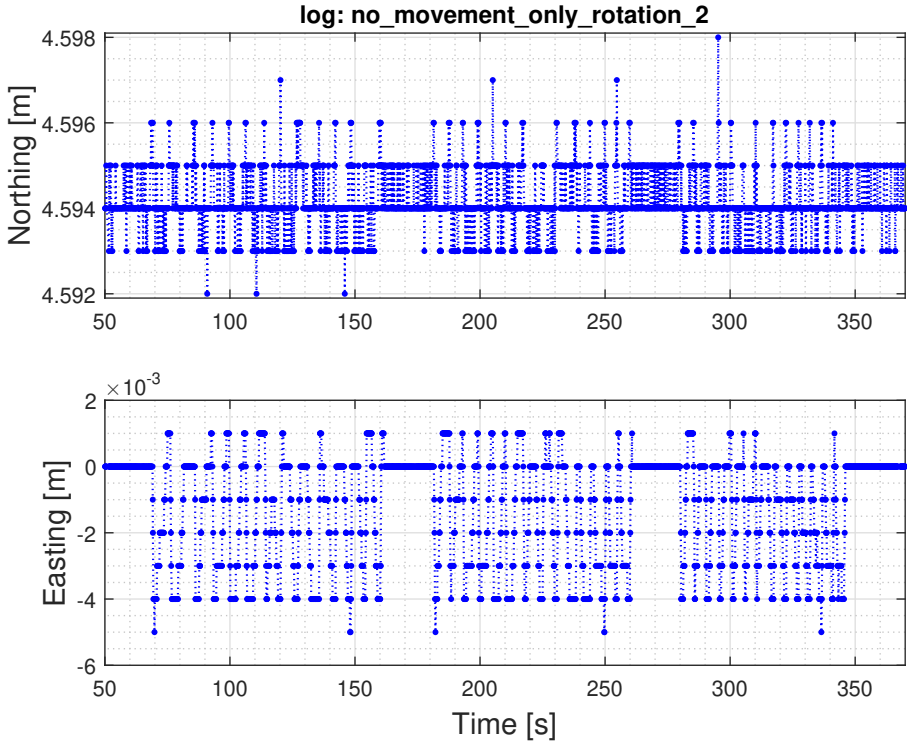


Fig. 6.26: *Northing* and *Easting* deviations with horizontal prism rotation (4.6 m distance).

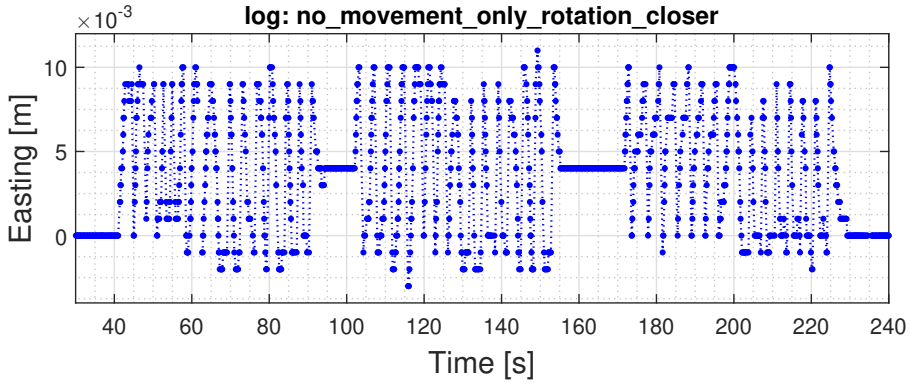


Fig. 6.27: *Easting* deviations with horizontal prism rotation (1.2 m distance).

6.3.5 Safe prism velocity limits

As is seen in Figure 6.28, the beginning sections with highest prism velocities contained sparser data than the rest of the motion. From the already acquired data and some additional tests, an evaluation of these limits could to some extent be performed. Furthermore, this evaluation does not depend on the light barrier sensor setup since only data from the RTS is being evaluated. That is why, in this case, acquired data from old experiment runs performed during the semestral part were also included. They are still labeled as old in the legend to make a clear distinction between the them and new data sets.

The first 4 periods of motion were always taken from each measurement run. The number of points contained within these periods was extracted and maximal velocities calculated. As is seen in Figure 6.28, the data acquires its full fidelity after a certain number of periods as the peaks of oscillations (and velocity) get smaller. To be able to compare relative data loss of an incomplete data portion from a full fidelity picture, only periods that do not contain the full recoveries should make it to the comparison. Otherwise the fully recovered periods would skew the results towards a lower relative data loss. That is why 4 periods of motion were chosen, where it was always guaranteed that the data, if it contained such losses, does not recover in this interval.

The expected maximal prism velocities according the the RTS specifications (see Sec. 4.1) are 3.0 m/s, 5.3 m/s and 9.7 m/s for 1.5 m, 2.7 m and 4.9 m respectively. These were calculated using the actual measured distances by the RTS rounded to cm precision. As can be seen in Figure 6.29, top left, data integrity decreases with a wider starting angle and with increasing velocities (top right). The velocity values were calculated as a gradient of the acquired angles in time, multiplied by the distance measurements. 80 to 90 points was the maximum observed within the four periods. The lowest was 30 to 40 and beyond that, with wider starting angles,

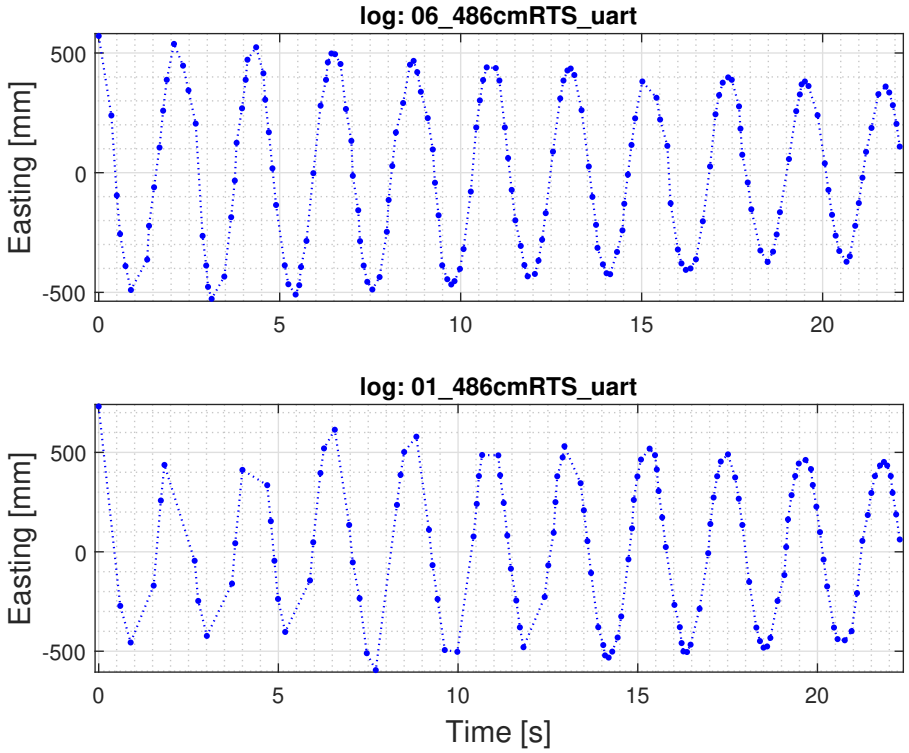


Fig. 6.28: Data reduction at wider motion starting angles.

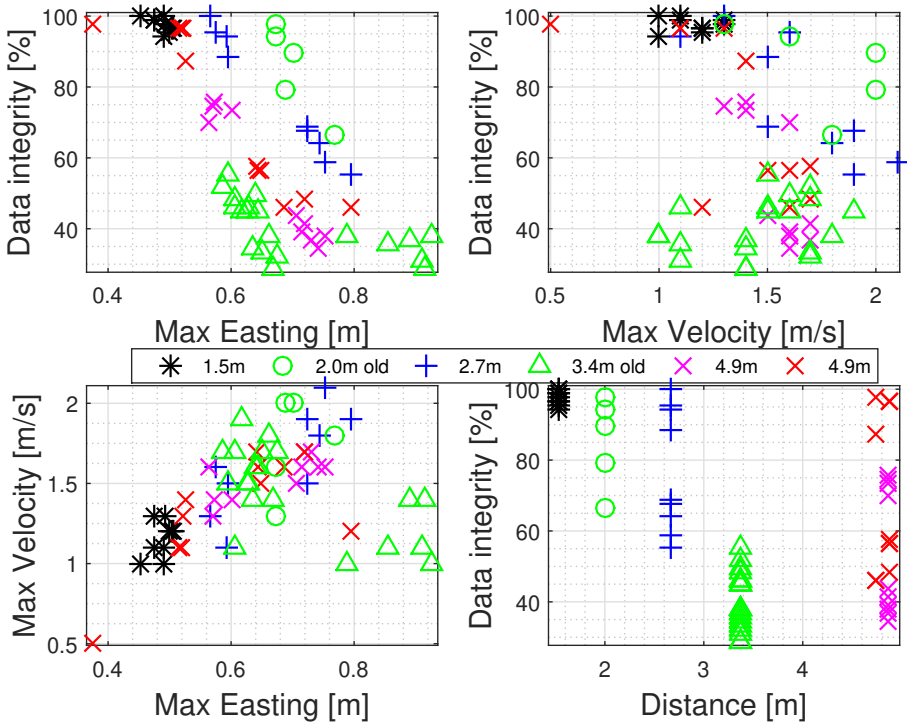


Fig. 6.29: Data integrity with motion starting angle.

the RTS completely lost track of the moving prism. Data from the old experiments also follows the general trend. A set of points at the 4.9 m distance was marked with a different color (purple). These are the runs, which contained visibly higher results of the measurement delays in Figure 6.13.

There are limitations to this procedure however. First is the obvious velocity estimation from the actual imperfect data and a lack of precise control of this velocity (also see Sec. 5.4.5). A sharp turn in the velocity data can be seen, which is a result of unreliable calculations from too sparse data. Velocities higher than around 2 m/s were not reliably calculable. However, if the trend was to be extended, it might be reasonable to assume a cutoff at around 3 m/s. Second, changing environmental conditions could have affected the results as, for example, the data integrity might have been influenced highly by changing light conditions, which was not controlled for in these experiments.

A decrease of data fidelity with increasing RTS distance is not an expected result (Fig. 6.29, bottom right) and this might have been influenced by several factors, processing errors included. One hypothesis might be that the TOF distance measurement took a longer time at the greater distances resulting in less acquired data points during the prism's motion. However, further experiments would have to be done to verify this. Also, the data from narrower starting angles tends to cluster around higher values regardless of the distance. It is only the wider starting angle data points, that seem to follow such a trend (specifically, the 2.7 m and 4.9 m results). It is also important to mention that no wider start angle data is available at the 1.5 m distance since there, the RTS was not able to maintain lock on the prism at all. Another thing which should be verified is the extent to which acceleration and rapid change of direction could have an effect on data integrity. It might be the case that the RTS is in fact able to track with higher velocities without data loss and it is these rapid changes that have a larger effect on the data.

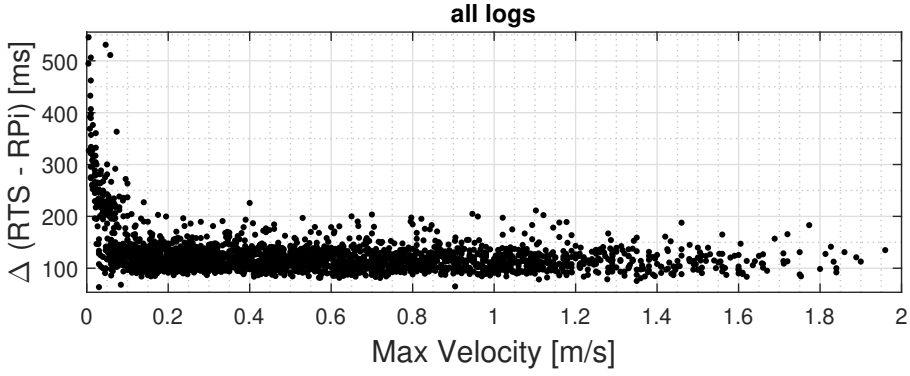


Fig. 6.30: RTS position measurement delay with max. prism velocity (combined results).

The last evaluation stemmed from the visible rise of measurement delays as the pendulum was gradually halting to a stop (see Fig. 6.8). As was mentioned in Section 6.2.2, these ending sections were not included in the final measurement delay evaluations, with a cutoff defined by the expected asymmetricality error of max. 20 ms. For this evaluation, maximal prism velocities were compared with the resulting time delays for all of the main experiment runs. Results from the old setup were not used in this case and peak velocities were calculated from the whole motion, excluding the initial segments with reduced data. Figure 6.30 shows a combined plot from all of the the main experiment results, where a sharp rise in delays is visible at very low velocities. Some points appearing beyond the 2 m/s limit are hidden from view as these were single outlying values in any particular data set. These results might be caused by the asymmetricality errors that are also increasing at lower velocities, however even the points which are not expected to have these errors follow the same trend. Nonetheless, equivalent shortcomings of the evaluation apply here as well.

6.4 Alternative navigation solutions

The results presented in the preceding chapters showed measurement delays of about 110 ms to 120 ms from true position time and limitations on target's dynamic properties. An interesting area of research is presented by Yang *et al.* [51] in using a Laser Tracker for the moving object navigation task. Several orders of magnitude higher accuracies of these devices might circumvent current limitations of RTSs, however different obstacles emerge within this solution as well, with even less tolerance to rapid dynamic target motion changes. The authors devised a new system for dynamic laser tracker measurements, which might make these devices even more favorable than RTSs for navigation tasks in the future.

A completely different and more obvious approach is presented by Retscher *et al.* [52] in using radio waves for localization and navigation inside buildings. This completely eliminates the direct line of sight necessity and would be more effectively expanded throughout the whole building interior. However, despite the favorable scalability of this solution, the positioning accuracies of these methods are still by at least an order of magnitude lower than the RTS solutions.

Conclusions and possible improvements

This thesis touched on the principles of position measurement with a total station and attempted to investigate their possible applications in more unconventional scenarios, such as UAV navigation in GNSS-denied environments (see Chap. 2). Chapter 3 described the parameters and basic operation of *Trimble* RTSs available at the UAMT institute of Brno University of Technology. Chapter 4 provided a deeper elaboration of the theoretical and practical problems of kinematic RTS position measurements, and outlined the reasoning behind and requirements for suitable evaluation experiments. A practical realization and implementation details were then shown in Chapter 5. Finally, all obtained results, together with their processing methodology, were presented and described in Chapter 6.

Initial tests showed a lack of a 10 Hz measurement rate functionality on the *Trimble S7* RTS, however it was fully available on the *S9 HP* (see Sec. 4.3.3). Potential benefits of the available *SX10* for future research were outlined and a completely new option in the *Trimble* UTS devices was briefly noted (see Sec. 4.1). Several evaluation experiments were performed with the *S9 HP* RTS, from which more than a 100 ms position measurement delay, when compared to a reference true measurement time, and a data propagation delay of around 26 ms were shown (Sec. 6.2.3). A correlation of RTS-assigned timestamps with serial port arrival rates was proposed (see Sec. 6.3.1). A limited analysis of RTS parameters' dependence on the target's dynamic properties was performed (Sec. 6.3.5). Position deviations of up to 5 mm, which could have contributed to the high delays were also shown from various additional tests. The reference measurement setup components should not have contributed to the final results with errors higher than around 1 ms. The data for processing was constrained to a max. 20 ms theoretical asymmetricality error caused by the sensor-rod setup, however the resulting offset was only at several units of milliseconds.

Alternative operation solutions for the RTSs, such as described in Section 3.7, might further mitigate the measurement delays. Generally, higher data transfer and measurement rates, which might also indicate better inter-component synchronization, are favorable. Larger distances from targets are preferred in order to maintain successful target tracking, however safe operational velocity (or possibly acceleration) ranges might be limited to only below several units and above tenths of m/s. A suitable orientation relative to the target's path should also be considered.

As it currently stands, the tested *Trimble S9 HP* in its presented configuration would be capable of navigating small UAVs at reasonably low velocities with smooth trajectories, however a further expansion of its applicability through a reduction of measurement delays by custom operation solutions might be preferable. Further

verifications of a desirable RTS position might also lead to even lower delays. Alternatively, other higher grade total station models might provide even more satisfying results. Overall, robotic total stations might provide an accurate UAV navigation solution for indoors and other GNSS-denied environments when correctly set up. Their cumbersome scalability is still balanced by higher positioning accuracies when compared to more easily scalable options.

Possible additional work and improvements

Several other experiments could have been made to verify the effect of the RTS's alignment and position on the obtained results. A further measurement rate analysis, providing a better quantification of the RTS's behavior and more tests with different RTS devices, estimating any further benefits and differences could also have been made. Limitations of the additional performed experiments could be eliminated by, for example, utilizing a second light barrier or an encoder, which would provide more accurate and independent velocity estimations. Better prism rotation tests could have estimated a precise angle relation. Different types of motion, such as linear or circular, would also have been beneficial, singling out any other potential negative interferences that the current setup might have had on the results.

Bibliography

[1] HORELIČAN, Tomáš. *Měření pozice pohybujících se objektů pomocí robotické totální stanice* [online]. Brno, 2021, 44 p. [ref. 2021-04-25]. Available at: <https://www.vutbr.cz/studenti/zav-prace/detail/131019>. Semestral thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Thesis supervisor: Ing. Tomáš Jílek, Ph.D.

[2] HORELIČAN, Tomáš. POSITION MEASUREMENT OF MOVING OBJECTS USING A ROBOTIC TOTAL STATION. In: *Proceedings I of the 26th Conference STUDENT EEICT 2020* [online]. 1. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, 2021, accepted for publication, 4p [ref. 2021-5-11]. ISBN 978-80-214-5942-7. Available at: <https://conf.feec.vutbr.cz/eeict/index/pages/view/ke_stazeni>

[3] BAUTSCH. Historic universal theodolite Pistor & Martins manufactured in Berlin (1851), exposed 2017 at an exposition of GeoForschungsZentrum Potsdam. Taken free-hand with polarising filter. *Wikimedia Commons: Universaltheodolit.14Zoll.Pistor&Martins.Berlin.1851* [online]. San Francisco, California, United States: Wikimedia, 2017, 17 April 2017n. l. [ref. 2021-4-26]. Available at: <https://upload.wikimedia.org/wikipedia/commons/9/9e/Universaltheodolit.14Zoll.Pistor%26Martins.Berlin.1851.jpg>

[4] Trimble S9 HP: Total Station. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2021, 2021 [ref. 2021-4-29]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/styles/banner/public/2017-05/S9-total-station_2_1.png?itok=ANcR10Db>

[5] KAVANAGH, Barry F. and Tom B. MASTIN. *Surveying: Principles and Applications*. NINTH EDITION. United States of America: Pearson, c2014. ISBN 978-0-13-700940-4.

[6] Trimble Access General Survey: HELP. Trimble: *Trimble Geospatial help portal* [online]. Sunnyvale, California, United States: Trimble, 2018 [ref. 2021-03-10]. Available at: <https://help.trimblegeospatial.com/TALegacy/Help%20Files/2017_20_Help/English/General%20Survey%20Help%20v2017.20.pdf>

[7] TRIMBLE ACCESS GENERAL SURVEY: USER GUIDE. *Trimble: Trimble Geospatial help portal* [online]. Sunnyvale, California,

United States: Trimble, August 2020 [ref. 2021-4-25]. Available at: <https://help.trimblegeospatial.com/TrimbleAccess-PDFs/2021.00/en/TA_General_Survey.pdf>

[8] FRED THE OYSTER. A schematic representation of a theodolite's axes and circles. *Wikimedia Commons: Theodolite vermeer* [online]. San Francisco, Kalifornia, United States: Wikimedia, 2014, 7 November 2014n. l. [ref. 2021-1-2]. Available at: <https://upload.wikimedia.org/wikipedia/commons/d/df/Theodolite_vermeer.svg>

[9] DTR. An SVG to show the relationship between Zenith, Nadir, Horizon. Based on Wikiman 1m80.svg and Horizons.svg. *Wikimedia Commons: Zenith-Nadir-Horizon* [online]. San Francisco, Kalifornia, United States: Wikimedia, 2007, 13 August 2007 [ref. 2021-5-1]. Available at: <https://upload.wikimedia.org/wikipedia/commons/4/47/Zenith-Nadir-Horizon.svg>

[10] LENDA, Grzegorz, Andrzej UZNANSKI and Michal STRACH. Comparison of Accuracy of Kinematic Methods for Localization of Mobile Targets. In: *2018 Baltic Geodetic Congress (BGC Geomatics)* [online]. Olsztyn, Poland: IEEE, 2018, 2018, p. 138-144 [ref. 2021-03-10]. ISBN 978-1-5386-4898-8. Available at: doi:10.1109/BGC-Geomatics.2018.00032

[11] LENDA, G., A. UZNAÑSKI and M. STRACH. Influence of Time Delays of Robotic Total Stations Witch High Sampling Frequency on Accuracy of Measurements to Moving Prisms. *Archives of Civil Engineering* [online]. 2019, **65**(1), 31-48 [ref. 2021-03-10]. ISSN 1230-2945. Available at: doi:10.2478/ace-2019-0003

[12] KEREKES, Gabriel and Volker SCHWIEGER. Position Determination of a Moving Reflector in Real Time by Robotic Total Station Angle Measurements. *Journal of Geodesy, Cartography and Cadastre* [online]. Bucharest, c2018, December 2018, (9), 13-18 [ref. 2021-4-28]. ISSN 1454-1408. Available at: <https://jgcc.geoprevi.ro/docs/2018/9/jgcc_2018_no9_2.pdf>

[13] KEREKES, Gabriel and Volker SCHWIEGER. Kinematic Positioning in a Real Time Robotic Total Station Network System. *Bornimer Agrartechnische Berichte: 6th International Conference on Machine Control and Guidance* [online]. Potsdam, c2018, 1 – 2 October 2018n. l., **101**, 35-43 [ref. 2020-12-30]. ISSN 0947-7314. Available at: <https://www.researchgate.net/publication/330779154_Kinematic_Positioning_in_a_Real_Time_Robotic_Total_Station_Network_System>

[14] LACKNER, Stefan and Werner LIENHART. Impact of Prism Type and Prism Orientation on the Accuracy of Automated Total Station Measurements. In: *Joint International Symposium on Deformation Monitoring* (JISDM) [online]. TU Wien: FIG, 2016, Mar 2016, 8p [ref. 2021-4-29]. Available at: <https://www.fig.net/resources/proceedings/2016/2016_03_jisdm_pdf/nonreviewed/JISDM_2016_submission_24.pdf>

[15] STEMPFHUBER, Werner. *VERIFICATION OF THE TRIMBLE UNIVERSAL TOTAL STATION (UTS) PERFORMANCE FOR KINEMATIC APPLICATIONS* [online]. Zurich, 2009 [ref. 2021-4-29]. Available at: <https://www.researchgate.net/publication/242294561_VERIFICATION_OF_THE_TRIMBLE_UNIVERSAL_TOTAL_STATION_UTS_PERFORMANCE_FOR_KINEMATIC_APPLICATIONS>. Conference Paper. Swiss Federal Institute of Technology.

[16] BÖNIGER, Urs and Jens TRONICKE. On the Potential of Kinematic GPR Surveying Using a Self-Tracking Total Station: Evaluating System Crosstalk and Latency. *IEEE Transactions on Geoscience and Remote Sensing* [online]. 2010, 48(10), 3792-3798 [ref. 2021-4-29]. ISSN 0196-2892. Available at: doi:10.1109/TGRS.2010.2048332

[17] GOJCIC, Zan, Slaven KALENJUK and Werner LIENHART. Synchronization routine for real-time synchronization of robotic total stations. In: *INGENEO 2017: Proceedings of the 7th International Conference on Engineering Surveying* [online]. Lisbon: FIG, 2017, October 18 - 20, 2017, p. 83-91 [ref. 2021-4-29]. Available at: <https://fig.net/resources/proceedings/2017/2017_10_INGEO/44PR_TS4-4_Gojcic.pdf>

[18] THALMANN, Tomas and Hans NEUNER. Temporal calibration and synchronization of robotic total stations for kinematic multi-sensor-systems. *Journal of Applied Geodesy* [online]. 2020, **15**(1), 13-30 [ref. 2021-4-30]. ISSN 1862-9024. Available at: doi:10.1515/jag-2019-0070

[19] Prisms & Targets: Traverse Prism. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2021, 2021 [ref. 2021-5-16]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/styles/large/public/2019-11/58026019%20-%20Traverse%20Prism%20with%20AR%20Coating.png?itok=qBavwDpp>

[20] Trimble S7: Robotic Total Station. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2021, 2021 [ref. 2021-4-29]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/

files/styles/banner/public/2017-05/S7-total-station_2.png?itok=05rmsPuj>

[21] Trimble S7: DATASHEET. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2015-2019, 2019, p. 1-4 [ref. 2021-1-3]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/2019-06/022516-154G_TrimbleS7_DS_USL_0619_LRsec.pdf>

[22] Trimble S9/S9 HP: DATASHEET. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2015-2020, 2020 [ref. 2021-03-10]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/2020-04/022516-155H_TrimbleS9_DS_USL_0320_LRsec_0.pdf>

[23] Trimble SX10: Scanning Total Station. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2021, 2021 [ref. 2021-5-5]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/styles/banner/public/2017-05/SX10-total-station_0.png?itok=FaEpBLrP>

[24] Trimble SX10: DATASHEET. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2016-2017, 2017, p. 1-4 [ref. 2021-1-3]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/2019-10/Datasheet%20-%20SX10%20Scanning%20Total%20Station%20-%20English%20USL%20-%20Screen.pdf>

[25] Trimble TSC7: CONTROLLER. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2018-2019, 2019 [ref. 2021-4-29]. Available at: <https://geospatial.trimble.com/sites/geospatial.trimble.com/files/2020-06/Datasheet%20-%20Trimble%20TSC7%20controller%20-%20English%20%28USL%29%20-%20Screen.pdf>

[26] Leica TPS1200: GeoCOM Reference Manual. *Arecibo Observatory* [online]. Heerbrugg, Canton St. Gallen, Switzerland: Leica Geosystems [ref. 2021-5-5]. Available at: <http://www2.naic.edu/~phil/hardware/theodolites/TPS1200_GeoCOM_Manual.pdf>

[27] Trimble EMPOWER Software Development Overview. *Trimble: Developer Home* [online]. Sunnyvale, California, United States: Trimble, c2020, 2020 [ref. 2021-5-5]. Available at: <https://mcsdeveloper.trimble.com/software-development/>

[28] Trimble Access SDK: Trimble Access Software Development Kit. *Trimble: Geospatial* [online]. Sunnyvale, California, United States: Trimble, c2021,

2021 [ref. 2021-5-5]. Available at: <https://geospatial.trimble.com/products-and-solutions/trimble-access-sdk>

[29] TRIMBLE PRECISION SDK FOR WINDOWS: Quick access to all things a developer may be interested in. *Trimble: Developer* [online]. Sunnyvale, California, United States: Trimble, c2021, 2021 [ref. 2021-5-5]. Available at: <https://developer.trimblegeospatial.com/tpsdk_windows/index.html>

[30] LIENHART, Werner, Matthias EHRHART and Magdalena GRICK. High frequent total station measurements for the monitoring of bridge vibrations. *Journal of Applied Geodesy* [online]. 2017, **11**(1), 1-8 [ref. 2021-03-10]. ISSN 1862-9024. Available at: doi:10.1515/jag-2016-0028

[31] ROBERTS, Craig and Peter BOORER. Kinematic positioning using a robotic total station as applied to small-scale UAVs. *Journal of Spatial Science* [online]. 2016, **61**(1), 29-45 [ref. 2021-03-10]. ISSN 1449-8596. Available at: doi:10.1080/14498596.2015.1068232

[32] MAXIM, Artyom, Otto LERKE, Marshall PRADO, Moritz DÖRSTELMANN, Achim MENGES and Volker SCHWIEGER. UAV Guidance with Robotic Total Station for Architectural Fabrication Processes. *DVW-SCHRIFTENREIHE: Unmanned Aerial Vehicles 2017 (UAV 2017)* [online]. 2017, 9. 2. 2017, **86**, 145-161 [ref. 2020-12-30]. ISSN 0940-4260. Available at: <https://www.researchgate.net/publication/320491203_UAV_Guidance_with_Robotic_Total_Station_for_Architectural_Fabrication_Processes>

[33] HANKUS-KUBICA, Agnieszka, Bartosz BRZOZOWSKI, Karol CHEDA, Maciej KULINSKI and Piotr WIECZOREK. Verification tests of total station usability for UAV position measurements. *2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)* [online]. IEEE, 2020, 2020, 331-335 [ref. 2020-12-30]. ISBN 978-1-7281-6636-0. Available at: doi:10.1109/MetroAeroSpace48742.2020.9160081

[34] PARAFOROS, Dimitris S., Marcus REUTEMANN, Galibjon SHARIPOV, Roland WERNER and Hans W. GRIEPENTROG. Total station data assessment using an industrial robotic arm for dynamic 3D in-field positioning with sub-centimetre accuracy. *Computers and Electronics in Agriculture* [online]. 2017, **136**(April 2017), 166-175 [ref. 2021-03-10]. ISSN 01681699. Available at: doi:10.1016/j.compag.2017.03.009

[35] Trimble SPSx30 Total Station: USER GUIDE. *Trimble: Docushare* [online]. Sunnyvale, California, United States: Trimble, June 2017 [ref. 2021-5-10]. Available at: <https://trl.trimble.com/docushare/dsweb/Get/

Document-464679/SPSx30%20User%20Guide%20Rev%20B%20June%202017.pdf>

[36] Phantom 4 Pro V2.0: Specs. *DJI: Phantom 4 Pro V2.0* [online]. Shenzhen, Guangdong, China: SZ DJI Technology Co., c2021 [ref. 2021-4-20]. Available at: <https://www.dji.com/sk/phantom-4-pro-v2/specs>

[37] ČEPL, Miroslav. *Měření přesnosti GNSS přijímačů* [online]. Brno, 2019, 51 p. [ref. 2020-12-30]. Available at: <https://www.vutbr.cz/studenti/zav-prace/detail/118986>. Bachelor's thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Thesis supervisor: Ing. Tomáš Jílek, Ph.D.

[38] Home: About. *Wiring Pi: GPIO Interface library for the Raspberry Pi* [online]. Gordon, c2021 [ref. 2021-5-1]. Available at: <http://wiringpi.com/>

[39] C library for Broadcom BCM 2835 as used in Raspberry Pi. *Bcm2835* [online]. Mike McCauley, 2021, 2021 [ref. 2021-5-1]. Available at: <https://www.airspayce.com/mikem/bcm2835/index.html>

[40] Pigpio: pigpio is a C library for the Raspberry which allows control of the General Purpose Input Outputs (GPIO). *GitHub: joan2937* [online]. San Francisco, California, United States: GitHub, 2021 [ref. 2021-03-10]. Available at: <https://github.com/joan2937/pigpio>

[41] Micro Sensing Device Data Book: Photomicrosensors Microphotonic Devices. *Omron* [online]. Shiokoji Horikawa, Shimogyo-ku, Kyoto 600-8530, Japan: OMRON Corporation, c2015, September 2015, p. 56-57 [ref. 2021-3-10]. Available at: <https://omronfs.omron.com/en_US/ecb/products/pdf/en-scec-001d-1.pdf>

[42] Ultra-minute Photoelectric Sensor: EX-ZSERIES. *Panasonic* [online]. 2431-1 Ushiyama-cho, Kasugai-shi, Aichi, 486-0901, Japan: Panasonic Industrial Devices SUNX Co., c2015, July 2015 [ref. 2021-4-28]. Available at: <https://mediap.industry.panasonic.eu/assets/download-files/import/ds_exz_jp_en.pdf>

[43] Ptpd: PTPd official source - master branch a.k.a. trunk. *GitHub: ptpd* [online]. San Francisco, California, United States: GitHub, 2019 [ref. 2021-03-10]. Available at: <https://github.com/ptpd/ptpd>

[44] Linuxptp: PTP IEEE 1588 stack for Linux. *SourceForge: linuxptp* [online]. San Diego, California, United States: Slashdot Media, c2021 [ref. 2021-5-2]. Available at: <https://sourceforge.net/projects/linuxptp/>

[45] W32Time: This repo provides resources for high accuracy time on Windows. *GitHub: microsoft* [online]. San Francisco, California, United States: GitHub, 2020 [ref. 2021-03-10]. Available at: <https://github.com/microsoft/W32Time>

[46] Raspberry Pi 3 Model B: Single-board computer with wireless LAN and Bluetooth connectivity. *Raspberry Pi* [online]. Cambridge, England, United Kingdom: Raspberry Pi (Trading) Limited., 2021 [ref. 2021-5-2]. Available at: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

[47] Trimble Knowledge Center. *Trimble Knowledge Center: SURVEY* [online]. Sunnyvale, Kalifornia, United States: Trimble, c2008, 21 January 2008n. l. [ref. 2020-12-31]. Available at: <http://trl.trimble.com/docushare/dsweb/Get/Document-406433/SC_Design%20of%20backsight%20point%20configuration%20in%20resection%20setup.htm>

[48] Configuring Systems for High Accuracy. *Microsoft: Documentation* [online]. Redmond, United States: Microsoft Corporation, 2018, 05/08/2018 [ref. 2021-5-4]. Available at: <https://docs.microsoft.com/en-us/windows-server/networking/windows-time-service/configuring-systems-for-high-accuracy>

[49] SAMA, Michael P., Timothy S. STOMBAUGH and James E. LUMPP. A hardware method for time-stamping asynchronous serial data streams relative to GNSS time. *Computers and Electronics in Agriculture* [online]. 2013, **97**(2013), 56-60 [ref. 2021-4-30]. ISSN 01681699. Available at: doi:10.1016/j.compag.2013.07.003

[50] SimpleDataParsing: parsing text-file data into a csv file for later processing. *GitHub: Imaniac230* [online]. San Francisco, California, United States: GitHub, 2020 [ref. 2021-5-7]. Available at: <https://github.com/Imaniac230/SimpleDataParsing>

[51] YANG, Linghui, Yuanlin PAN, Jiarui LIN, Yang LIU, Yue SHANG, Shuo YANG and Hanwen CAO. Automatic Guidance Method for Laser Tracker Based on Rotary-Laser Scanning Angle Measurement. *Sensors* [online]. 2020, 20(15) [ref. 2021-5-10]. ISSN 1424-8220. Available at: doi:10.3390/s20154168

[52] RETSCHER, Guenther, Vassilis GIKAS, Hannes HOFER, Harris PERAKIS and Allison KEALY. Range validation of UWB and Wi-Fi for integrated indoor positioning. *Applied Geomatics* [online]. 2019, 11(2), 187-195 [ref. 2021-5-10]. ISSN 1866-9298. Available at: doi:10.1007/s12518-018-00252-5

Symbols and abbreviations

API Application Programming Interface

BUT Brno University of Technology

COM Communication

CPU Central Processing Unit

CSV Comma-Separated Values

DR Direct Reflex

DSP Digital Signal Processor

DST Daylight Saving Time

EDM Electronic Distance Measurement

ENU East, North, Up

FEEC Faculty of Electrical Engineering and Communication

FEKT Fakulta elektrotechniky a komunikačních technologií

FPGA Field-Programmable Gate Array

GNSS Global Navigation Satellite Systems

GPIO General-Purpose Input/Output

HW Hardware

IC Input Capture

ISR Interrupt Service Routine

I2C Inter Integrated Circuit

LAN Local Area Network

LED Light Emitting Diode

NFC Near Field Communication

NIC Network Interface Controller

NMEA National Marine Electronics Association

NTP Network Time Protocol

OS Operating System

PC Personal Computer

PLC Programmable Logic Controller

PTP Precision Time Protocol

RAM Random Access Memory

RPi Raspberry Pi

RTK Real Time Kinematic

RTS Robotic Total Station

SDK Software Development Kit

STD Standard

STS Scanning Total Station

SW Software

TOF Time of Flight

TRK Tracking

TTL Transistor–Transistor Logic

UAMT Ústav automatizace a měřicí techniky

UART Universal Asynchronous Receiver-Transmitter

UAV Unmanned Aerial Vehicle

UI User Interface

USB Universal Serial Bus

UTC Coordinated Universal Time

UTS Universal Total Station

VUT Vysoké učení technické v Brně

List of appendices

A Supplementary graphs, images and listings

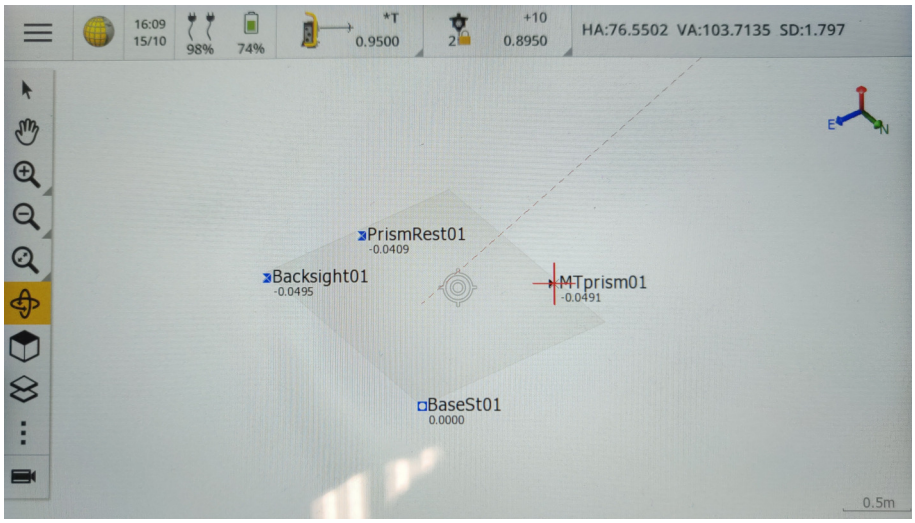


Fig. A.1: Coordinate orientation on the TSC7 (does not reflect the real experiment layout).

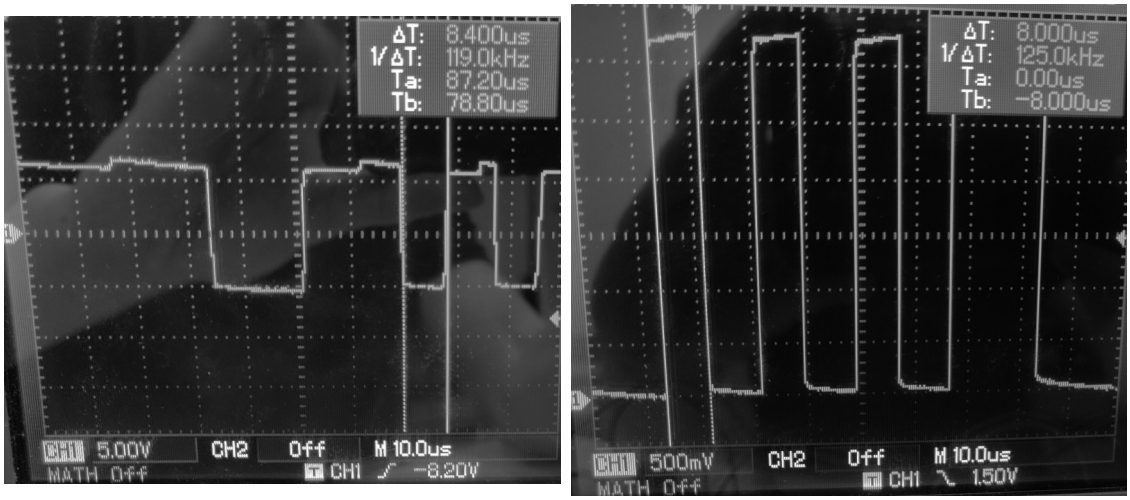


Fig. A.2: RTS data output. Left: before, Right: after conversion (bit values not the same between images).

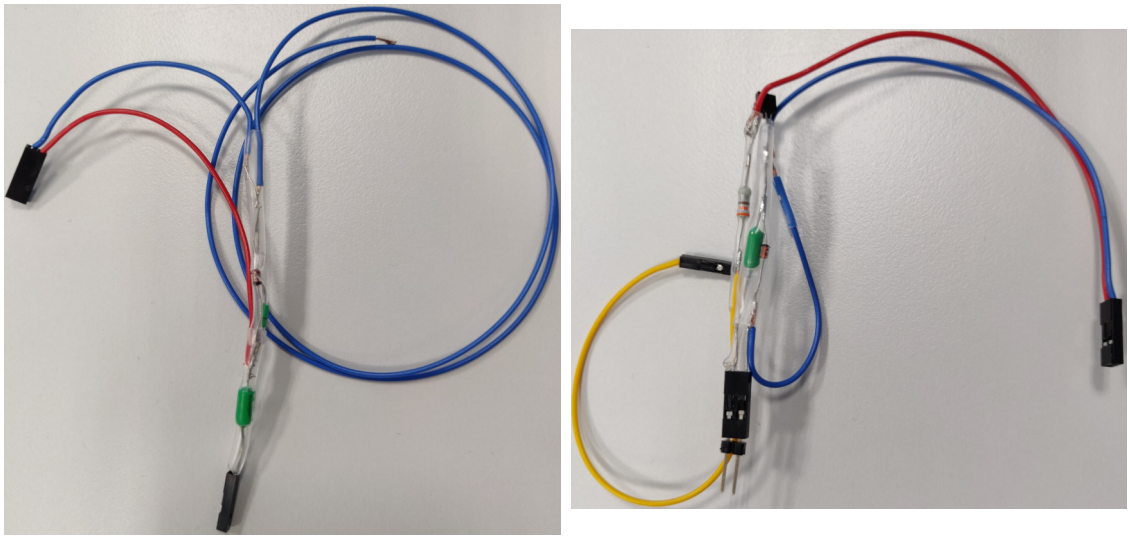


Fig. A.3: Custom adapters for Panasonic EX-Z11 (left) and RS232 (right) data output.

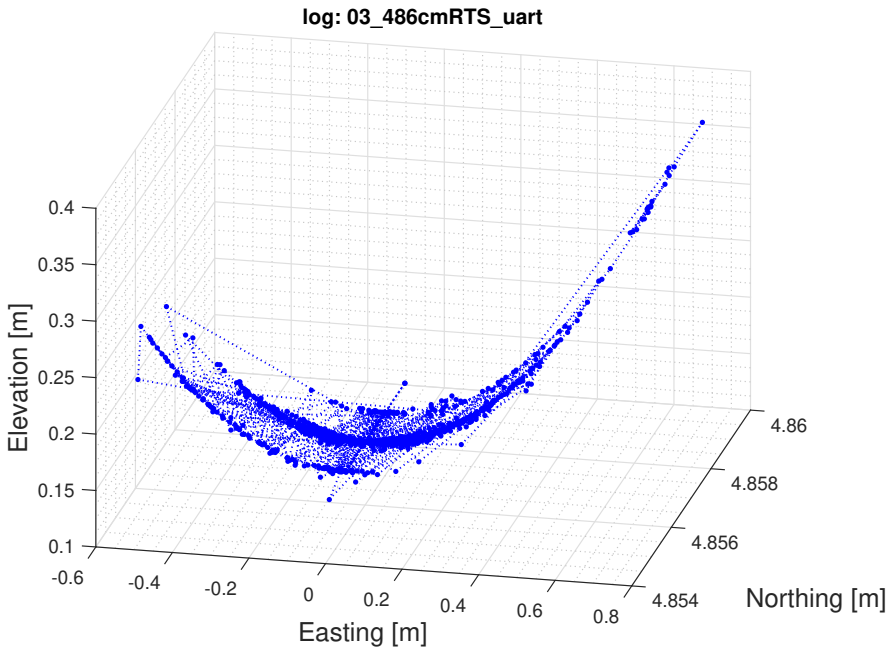


Fig. A.4: Near-perfect perpendicular RTS alignment.

Listing A.1: RTS data logging and time stamping.

```
if((uart_helper.bytes_available =
serDataAvailable(handle)) > 0)
  {
  clock_gettime(CLOCK_REALTIME, &receive_time);

  if((uart_helper.curr_read_byte =
  serReadByte(handle)) >= 0)
    {
#ifdef UART_DISPLAY_LOG
    if(b_displayed++ <= UART_DISPLAYED_BYTES)
      putc((char)uart_helper.curr_read_byte, stdout);
#endif /* UART_DISPLAY_LOG */

    putc((char)uart_helper.curr_read_byte, fout);
    if((char)uart_helper.prev_read_byte ==
    UART_RTS_EOT_CHAR)
      {
      fprintf(fout_t, "%ld%c%9ld%c%d%c%d\n",
      receive_time.tv_sec, UART_CSV_VALUE_SEPARATOR,
      receive_time.tv_nsec, UART_CSV_VALUE_SEPARATOR,
      uart_helper.bytes_available,
      UART_CSV_VALUE_SEPARATOR,
      uart_helper.curr_read_byte);
      ++uart_helper.total_handled;
      }
    uart_helper.prev_read_byte =
    uart_helper.curr_read_byte;
    }
  }
```

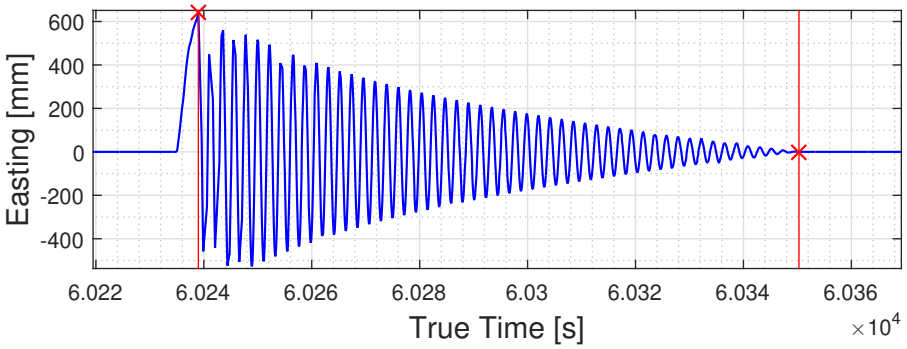


Fig. A.5: Finding the start and end point of the motion.

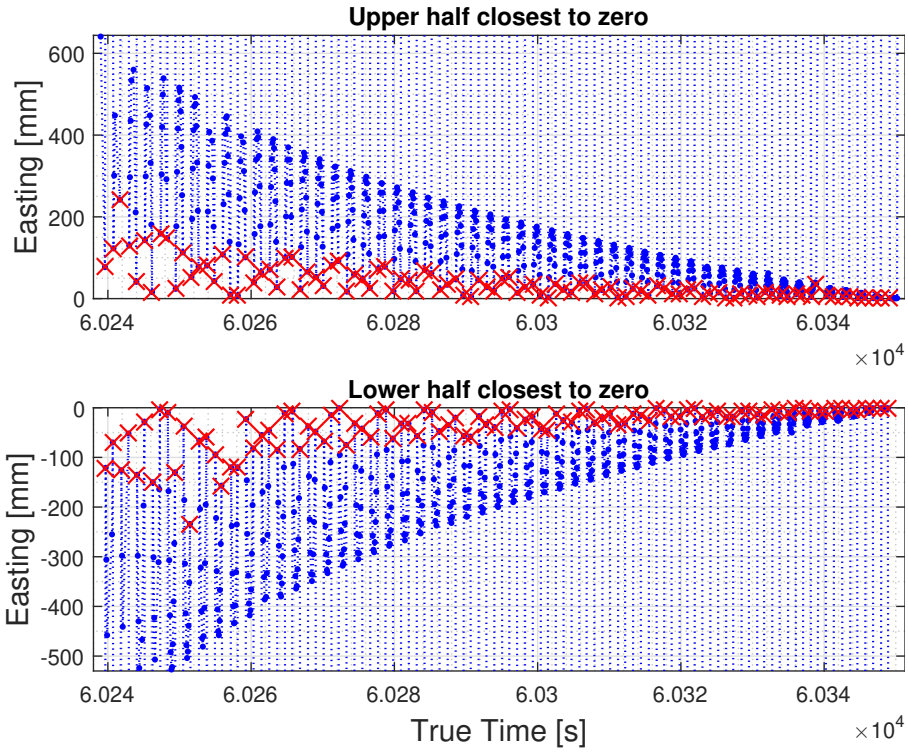


Fig. A.6: Finding the boundary points for interpolation sections.

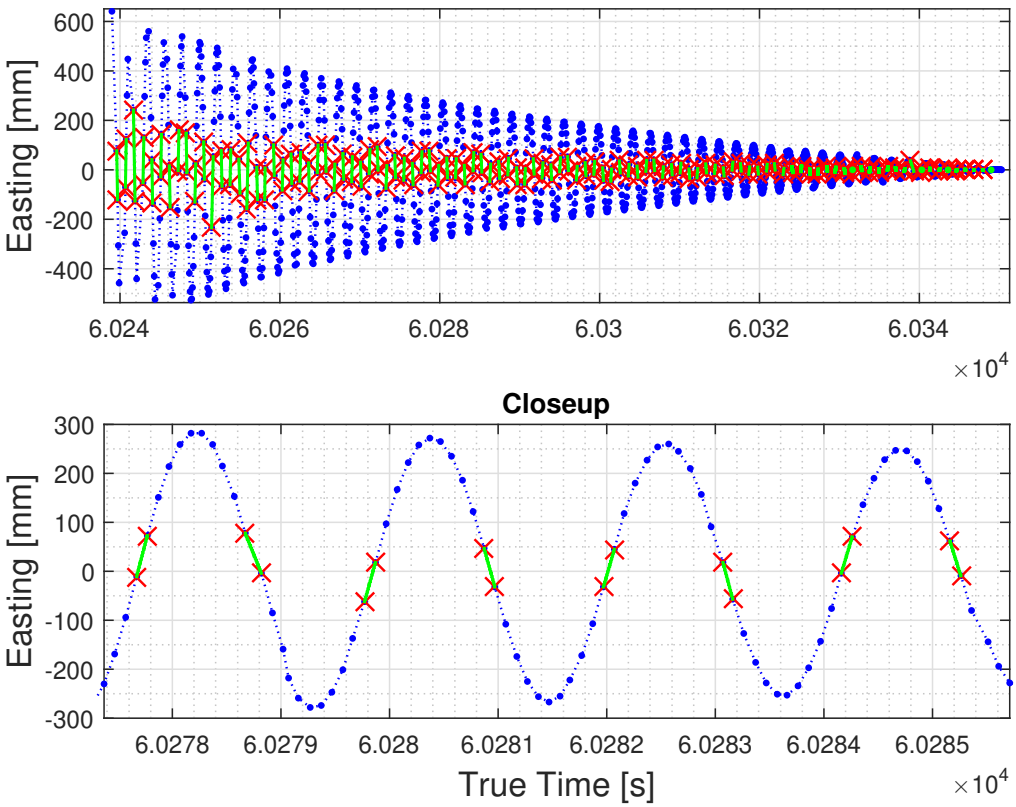


Fig. A.7: Interpolation in the zero-crossing sections.

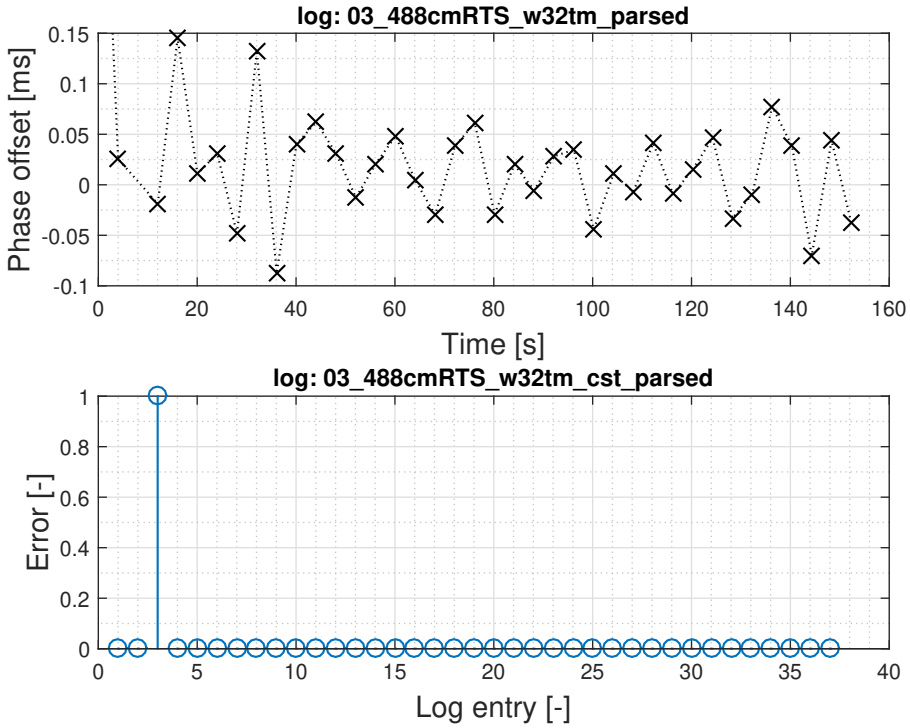


Fig. A.8: PTP synchronization status during an experiment run.

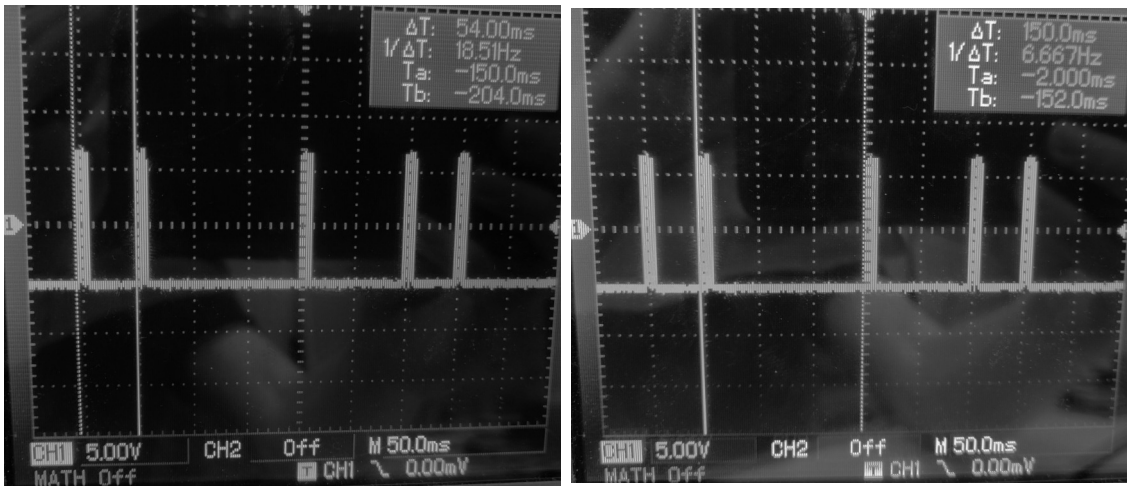


Fig. A.9: RTS data output, measurement point arrivals.

B Old experiment configuration

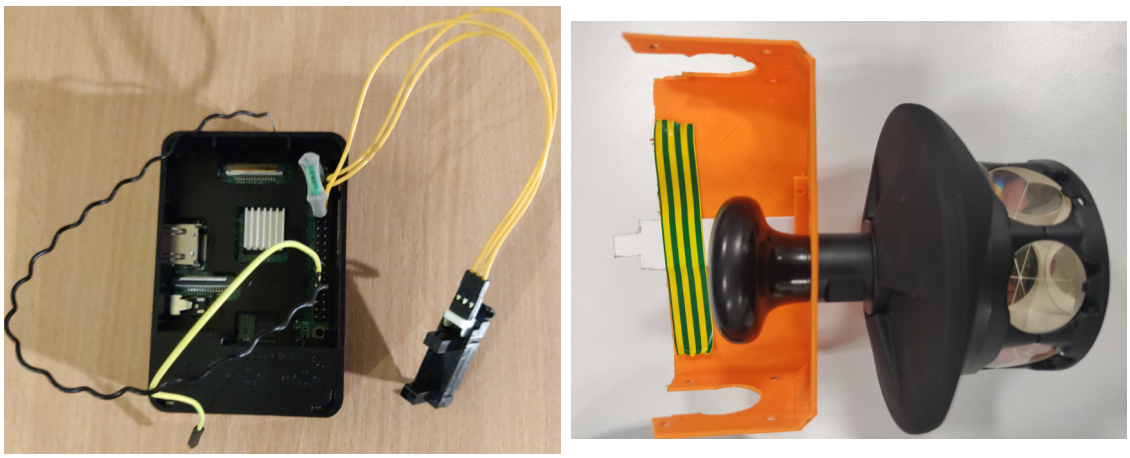


Fig. B.1: Old reference measurement setup. Left: RPi and sensor setup, Right: prism with extension.

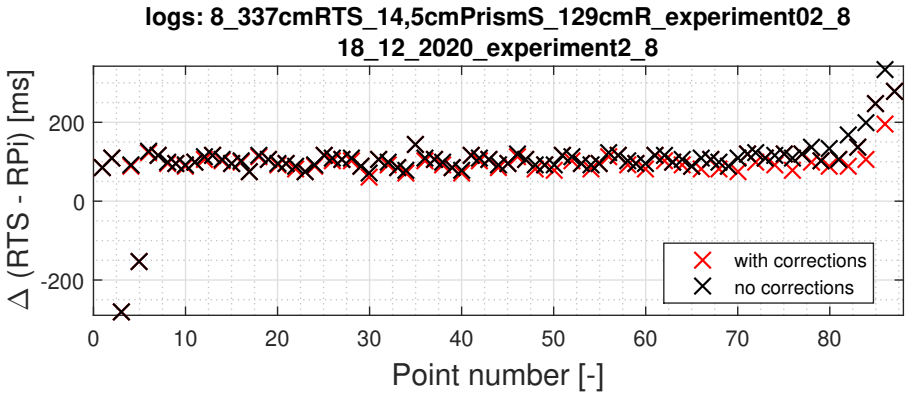


Fig. B.2: Comparison of the RTS and RPi timestamp values (old data, single run, delays only).

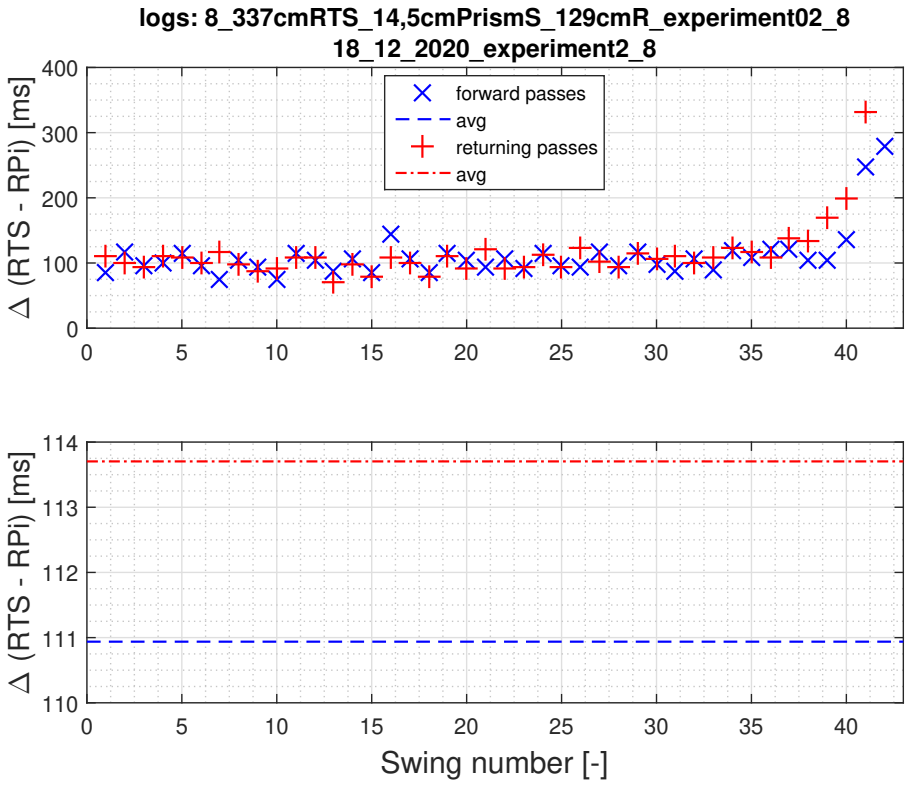


Fig. B.3: RTS position measurement delay with split passes (old data, single run).

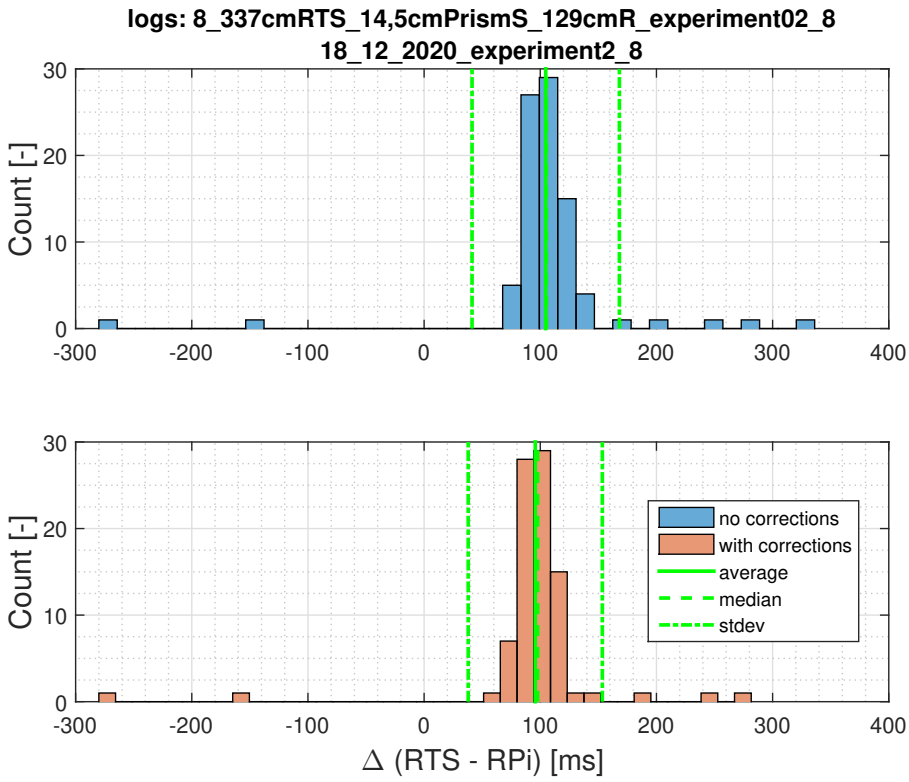


Fig. B.4: Histogram of the RTS position measurement delay (old data, single run).

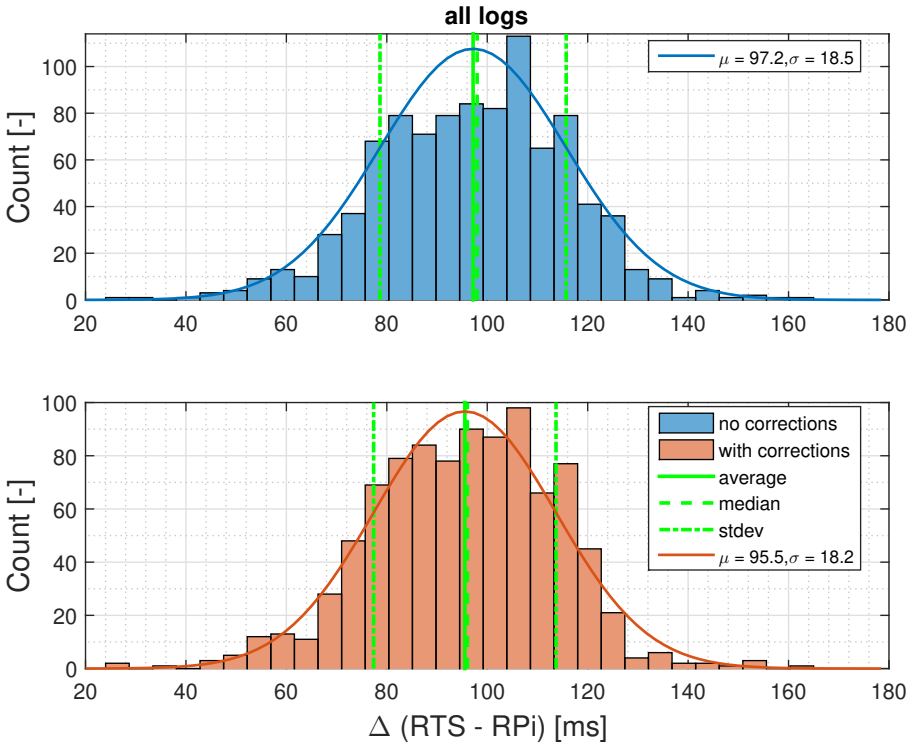


Fig. B.5: Histogram of RTS position measurement delay (old data, combined results).

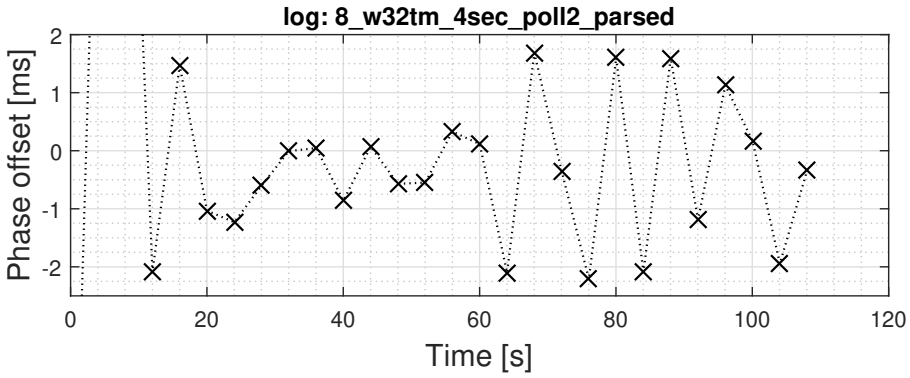


Fig. B.6: PTP synchronization status during an experiment (old data, single run).

C Electronic CD attachment contents

- `/` — root attachment folder
  - `latex` — LaTeX source files for this thesis
  - `linux` — Linux related source code root
    - `ntp_tests` — source for preliminary testing of custom NTP connections
    - `rpi` — source for the Raspberry Pi measuring program
      - `sandbox` — isolated feature testing
        - `gpiotest` — initial GPIO handling tests
        - `isrtest` — UART handling with ISR tests
        - `serialtest` — initial UART handling tests
  - `matlab` — Matlab source code root
    - `checks` — some checks/tests for correct function behavior
    - `logs` — acquired log files root
      - `RPi` — timestamp log files from the Raspberry Pi
      - `TrimbleS7` — data log files from the Trimble S7 RTS
      - `TrimbleS9` — data log files from the Trimble S9 HP RTS
      - `TSC7` — W32Time Windows service logs from the TSC7 controller
    - `old` — old development scripts
    - `saved_data` — saved processed data variables
  - `models` — source for part 3D models
  - `windows` — Windows related source code root
  - `Horelican_Tomas_DP_2021.pdf` — this thesis document