Development of Telemetry Ranging for Small Satellites

A Ranging technique for Delfi-PQ

Vikram Srikanth

Delft University of Technology

Development of Telemetry Ranging for Small Satellites

A Ranging technique
for Delfi-PQ

by

Vikram Srikanth

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday November 30, 2023 at 10:00 AM.

Student number:	5507901	
Project duration:	March 15, 2023 – November 30, 2023	
Thesis committee:	Dr. Stefano Speretta,	TU Delft, Supervisor
	Dr. Alessandra. Menicucci,	TU Delft, Chair
	Dr. Ir. B.C. Root,	TU Delft, Examiner

Cover: Cover: CSIRO Parkes Radio Telescope
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Acknowledgements

`That's one small step for man, one giant leap for mankind`

-Neil Armstrong

These immortal words of Neil Armstrong represent that space is a collaborative endeavor. Any work for space starts in a group and any achievement made is the achievement of a great number of people.

Similarly, this thesis would not have begun or be complete without acknowledgement of the contributions and support of the amazing people through life, especially in past two years at Delft.

I would like to thank my supervisor, Stefano Speretta, for the opportunity to work on this topic. His incredible knowledge and palpable enthusiasm for research made the thesis an absolute pleasure to work on. The thesis meetings were always invigorating and taught me the importance of planning and mentality in achieving goals in the thesis and in life. Next, I would like to thank Alessandra Menicucci and Bart Root from TU Delft for being a part of my thesis defense committee, taking the time and interest to read and assess my work.

Further, I thank Erdem Turan for his guidance and support throughout my thesis in helping me understand the topic. I also would like to thank Sevket Uludag for guiding me and helping me get started on the labs at TU Delft Space Systems Engineering. I also thank all the amazing people at the TU Delft Space Systems Engineering department for the fun and inspiring discussions we had.

Nearing the end, I thank my spectacular gang over at master rooms 2.44, who made thesis life a lot more fun and enjoyable. I also thank my roommates, flatmates and friends from Structures, Aerodynamics and FPP for the memories we shared these two years. I express my gratitude to all my friends I met at Aerospace and at TU Delft, for their support and all the good times we had.

Finally, I thank my family, my father, mother and brother, who enabled my dreams to study space and for their unwavering support through the years. I would have to write a separate thesis to express all my thanks. Last, but not the least, I thank my aunt, uncle and cousin over at Den Haag, who made the Netherlands feel like a second home.

Vikram Srikanth
Delft

Summary

Satellite ranging is one of the methods used to track satellites. By calculating the time of flight of electromagnetic signals, the distance of the satellite from a ground station or another satellite can be found. This information can be used to

1. Determine the position/orbit of the satellite, which helps
   - Conduct satellite navigation around Earth and in deep space.
   - Set up scientific experiments in space.
2. Supplement other tracking methods such as Doppler and Laser tracking allowing for a better overall tracking system.

However, a ranging system requires the satellite to have special equipment and bandwidth dedicated to powering and processing a ranging signal.

Small satellites form factors such as CubeSats and PocketQubes have been gaining popularity in recent years due to their low cost and faster developmental cycle. This also allows the design of distributed systems, which open up opportunities for improved performance and unique functionalities using multiple small satellites. However, due to Size, Weight, and Power (SWaP) constraints, many small satellites forgo the use of a ranging system.

A literature study of the state of the art in ranging systems revealed the invention of a new class of ranging techniques in the past decade. These are the telemetry ranging techniques[3, 33, 75], which eliminate the use of a dedicated ranging signal and instead use the telecommand and telemetry communication channels to derive range information. This leads to many advantages, such as not requiring dedicated ranging signal equipment, allowing use of bandwidth and amplifier efficient modulation techniques and simplifying satellite operations. This frees up SWaP and bandwidth for use in other applications while still providing accurate (<1m) range measurements.

It was identified that this technique had not been implemented in a small satellite system. Therefore, the thesis explored the capabilities of a telemetry ranging technique in small satellite scenarios using Delfi-PQ as a target platform. The main research question was whether it was feasible to implement telemetry ranging in Delfi-PQ with no additional hardware being added (using only software changes) and using open source, commercially available software and equipment. The goal for the range accuracy was targeted as <1km (1σ) to achieve better than Two-Line Element set (TLE) tracking accuracy, which is the accuracy many small satellites have in near Earth orbit[43].

Telemetry ranging, being a hybrid ranging method using both phase and time measurements to determine the range, needed a way to determine the time of flight of radio signals. For this purpose, a correlation algorithm was developed, tested and validated using GNU Radio, an open source radio software development toolkit.

For the thesis, the hardware in the loop testbed at TU Delft[65] which uses the commercially available Universal Software Radio Peripheral (USRP) B200 Software Defined Radio (SDR) was adapted to simulate the ground station radio and conduct ranging. Surface Acoustic Wave (SAW) delay lines were used to induce a delay in the radio signals and sequential ranging, which is a proven two way ranging technique, was implemented to find this delay. Since the range is given as the speed of light multiplied by the delay, any errors in determining this delay get multiplied by the speed of light too. Therefore, this test validated that the USRP-GNU Radio setup was capable of timing signals with an accuracy of <150m.

Following this, the thesis adapted the telemetry ranging technique to Delfi-PQ. Initial tests were carried out to ensure the ground station setup was capable of ranging with the Gaussian Frequency Shift Keying (GFSK) modulated telecommand codewords of Delfi-PQ. These tests further revealed a ground station system accuracy of <1m. An initial prototype telemetry ranging method was developed, where the radio sends commands to the satellite and the satellite sent a marker back along with the telemetry for each command. By correlating for this marker and knowing the sent time of the command, the range was determined.

This method reveled the presence of clock recovery errors and onboard processing delays, leading to ranging accuracies of the order of 100s of km (117.5km at 9600bps). Uplink bitrate and onboard processing were identified as primary drivers of performance, with doubling bitrate nearly halving the range error and increasing onboard processing increasing the range error.

Since one of the goals was to achieve a better than 1km accuracy, a timer onboard the satellite microcontroller was used to measure the processing time of the satellite. The time spent by the signals in the satellite modem were estimated and compensated for. Doing so revealed the presence of bit shifts in the satellite where the telemetry or telecommand data were received/transmitted a few bits earlier or later than they should have been. These bit shifts were compensated for based on statistical reasoning. This almost eliminated the onboard processing error and left the clock recovery error as the limit for range accuracy. The overall range method was validated using delay line based ranging tests.

The final range accuracy was determined to be around 10s of km (8.92km at 9600bps) for a single range measurement and still halved with a doubling of the bitrate. Range averaging, which works based on the property of Gaussian distributions that data is more likely to lie near its true value than away, was used to improve this accuracy to achieve the <1km goal. It was determined that at an uplink bitrate of 9600bps, <1km accuracy can be attained using 80 measurements which can be taken in 4.27s of ranging. Alternatively, at a data rate of 85.63 kbps, a single measurement or 53ms of ranging will give a <1km accuracy.

This obtained result was around the same order of performance as previously achieved in Time-Derived Ranging[38], where the system used ping packets to range and was completely redesigned for to conduct only ranging. Due to the time measurement onboard eliminating the processing delays, a 47% better performance than this literature was achieved. Further, the method retains all the advantages of telemetry ranging methods discussed above.

Overall, it was found that telemetry ranging is a viable technology for use in small satellite ranging. It did not place any requirement on the hardware of Delfi-PQ (meaning a software update can enable this method now) and achieved accuracies meeting the goal of <1km with a few seconds of ranging. One of the main advantages with telemetry ranging is that the ranging can be conducted whenever the satellite is communicating with the ground, meaning it can be done for an entire pass of a satellite (10mins for Delfi-PQ) with minimal penalty to telemetry data speeds. Averaging over an entire pass gives a 106 times range accuracy improvement leading to a range accuracy of 84.15m at 9600bps for Delfi-PQ.

The clock recovery was found to be the limit for ranging performance in a small satellite. This arises from the uplink symbol tracking loop. Therefore, future research to improve telemetry ranging performance in Delfi-PQ and small satellites is suggested on methods to improve the clock recovery. The main constraint here is due to the requirement that no hardware changes be made. In case additional hardware can be added to the satellite or the satellite has a software defined radio, the demod remod clock recovery error estimation method suggested in telemetry ranging literature[75] can be used for this purpose.

Contents

List of Figures

List of Tables

Acronyms

ΔDOR Delta Differential One Way Ranging. 2

ADEV Allan Deviation. 7, 113, 114
AGC Automatic Gain Control. 46, 51, 63
ASM Attached Synchronization Marker. 68, 70–72, 74–76, 87, 88
AWGN Additive White Gaussian Noise. 41

CCSDS Consultative Committee for Space Data Systems. 11, 16, 71
CERN European Organization for Nuclear Research. 114
CLTU Communication Link Transmission Unit. 16–18
CPU Central Processing Unit. 36, 67
CRC Cyclic Redundancy Check. 56, 67

DAC Digital to Analog Converter. 33
DLL Delay Locked Loop. 21
DPLL Digital Phase Locked Loop. 14
DSAC Deep Space Atomic Clock. 6, 7
DSN Deep Space Network. 11, 16, 65
DTTL Data Transition Tracking Loop. 19, 61, 62, 117

FFT Fast Fourier Transform. vi, 33, 37, 38, 66, 67
FOV Field of View. 3
FSK Frequency Shift Keying. 57

GFSK Gaussian Frequency Shift Keying. iii, 27, 54–61, 63, 65, 74, 86, 87
GMSK Gaussian Minimum Shift Keying. 12, 14, 27, 54, 56–59, 61–63, 86
GPS Global Positioning System. 2, 3, 7, 24, 49, 50, 52, 62, 88, 90
GPSDO Global Positioning System Disciplined Oscillator. 49, 88, 114
GUI Graphical User Interface. 29, 41

IDE Integrated Development Environment. 66
IF Intermediate Frequency. 20

MI Modulation Index. 58, 60, 61, 63, 64
MSB Most Significant Bit. 75
MSK Minimum Shift Keying. 57

NCO Numerical Controlled Oscillator. 21
NMEA National Marine Electronics Association. 50
NORAD North American Aerospace Defense Command. 3
NRZ Non Return to Zero. 56
NRZI Non Return to Zero Inverted. 56

PCC Pearsons Correlation Coefficient. 61
PLL Phase Locked Loop. 21
PN Pseudo Noise. 10–12, 14

PPS Pulse Per Second. 49, 50
PRN Pseudo Random Noise. 11
PTP Precise Time Protocol. 114

RAM Random Access Memory. 38
RMS Root Mean Squared. 3, 48
RMSE Root Mean Squared Error. 19, 78

SAW Surface Acoustic Wave. ii, vi, 28
SDR Software Defined Radio. ii, 4, 29
SGP4 Simplified General Perturbations 4. 3
SNR Signal to Noise Ratio. iv, 10, 11, 18, 47, 52, 53, 55, 59, 61–63, 82, 83, 89, 91
SPI Serial Peripheral Interface. 55, 74, 91
SWaP Size, Weight, and Power. ii, 2, 6, 7, 25

TLE Two-Line Element set. ii, 3, 13

USB Universal Serial Bus. 66
USO Ultra Stable Oscillators. 7
USRP Universal Software Radio Peripheral. ii, 4, 27–29, 40, 42, 44, 46–49, 51–53, 58–60, 62, 63, 65, 67, 69, 77, 79, 86–89
UTC Coordinated Universal Time. 34

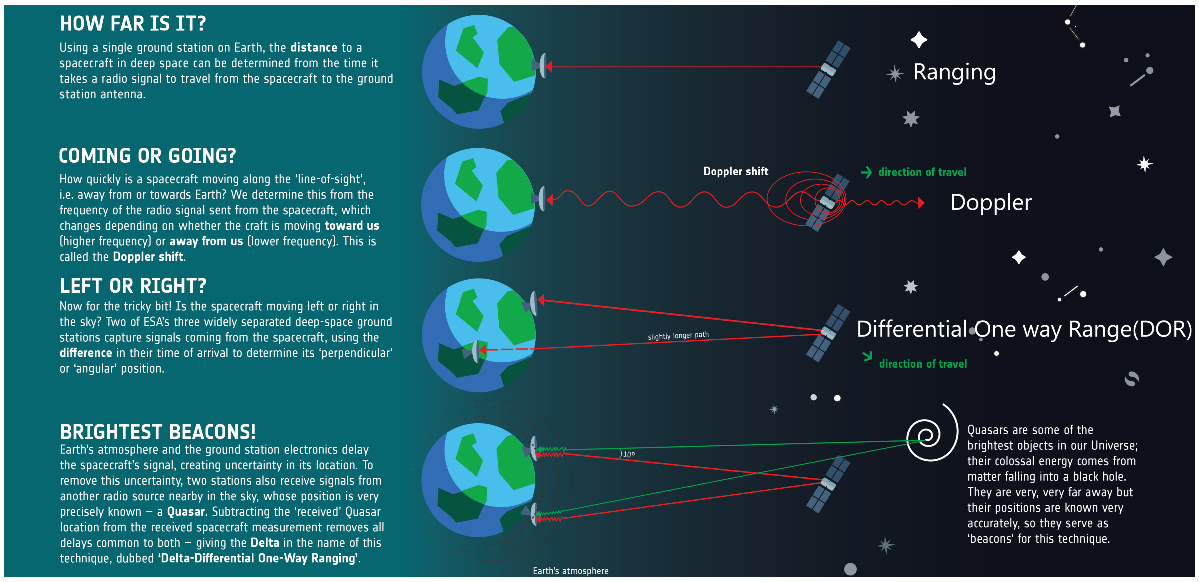
VCO Voltage Controlled Oscillator. 21

1 Introduction

How can we find out where a satellite is? Satellites are launched into pre-designed trajectories, so there is always a certain level of knowledge about its location. These trajectories follow the laws of physics, so using these laws a predication can be made about the location of a satellite. Such a knowledge based predictive approach to problems is called modelling, and the predictions themselves are called models. These predictions or models start off very accurate and eventually develop errors that drift their predicted satellite position away from the true position of the satellite. This drift is influenced by various factors such as the knowledge of physics used for the model, approximations employed and the accuracy of the computers used for running the calculations[77].

Thus, to counteract this drift, these satellite navigation models are regularly updated with position data obtained from other sources. With these frequent updates, the model's accuracy resets to the accuracy of the measurement provided, extended the duration for which the model's predictions can be used.

There are numerous methods which obtain data about satellite position or 'track' them, out of which radiometric tracking techniques have been dominant in the past decade for deep space[25]. These are of three types (Figure 1.1):


HOW FAR IS IT?

Using a single ground station on Earth, the **distance** to a spacecraft in deep space can be determined from the time it takes a radio signal to travel from the spacecraft to the ground station antenna.

COMING OR GOING?

How quickly is a spacecraft moving along the 'line-of-sight', i.e. away from or towards Earth? We determine this from the frequency of the radio signal sent from the spacecraft, which changes depending on whether the craft is moving **toward us** (higher frequency) or **away from us** (lower frequency). This is called the **Doppler shift**.

LEFT OR RIGHT?

Now for the tricky bit! Is the spacecraft moving left or right in the sky? Two of ESA's three widely separated deep-space ground stations capture signals coming from the spacecraft, using the **difference** in their time of arrival to determine its 'perpendicular' or 'angular' position.

BRIGHTEST BEACONS!

Earth's atmosphere and the ground station electronics delay the spacecraft's signal, creating uncertainty in its location. To remove this uncertainty, two stations also receive signals from another radio source nearby in the sky, whose position is very precisely known – a **Quasar**. Subtracting the 'received' Quasar location from the received spacecraft measurement removes all delays common to both – giving the **Delta** in the name of this technique, dubbed **'Delta-Differential One-Way Ranging'**.

Quasars are some of the brightest objects in our Universe; their colossal energy comes from matter falling into a black hole. They are very, very far away but their positions are known very accurately, so they serve as 'beacons' for this technique.

Figure 1.1: Type of Radio Tracking[5]

1. Ranging: Satellite ranging works on the principle of radio waves (or any electromagnetic wave) travelling at the speed of light. In ranging, a known electromagnetic wave or 'ranging signal' is

transmitted from the satellite to the ground station at a known point in time. In the ground, the time this wave arrives at is noted. Thus, the time of flight of this electromagnetic wave can be determined. From this time of flight, the 'range' of the satellite or its **distance** from the ground station antenna is determined as

$$Range = c * (t_g - t_s) \tag{1.1}$$

where c is the speed of light, t_g is the time the signal is received at the ground and t_s is the time the signal is received at the satellite. The Global Positioning System (GPS) constellation of satellites also use this concept to provide positioning services to everyone.

2. Range Rate or Doppler Tracking: The spacecraft due to movement causes a frequency shift in the radio signals sent from it to the ground. This change in frequency is called Doppler effect and by measuring this frequency shift, the spacecraft **velocity** can be estimated. The observed frequency is given as:

$$f_o = \frac{c + v_o}{c + v_s} \times f_i \tag{1.2}$$

where v_o is the velocity of the observer, f_i is the initial frequency, v_s is the velocity of the spacecraft, and c is the speed of light. The Doppler effect can be seen in Figure 1.2. This method provides the velocity of the spacecraft towards or away from the observer.

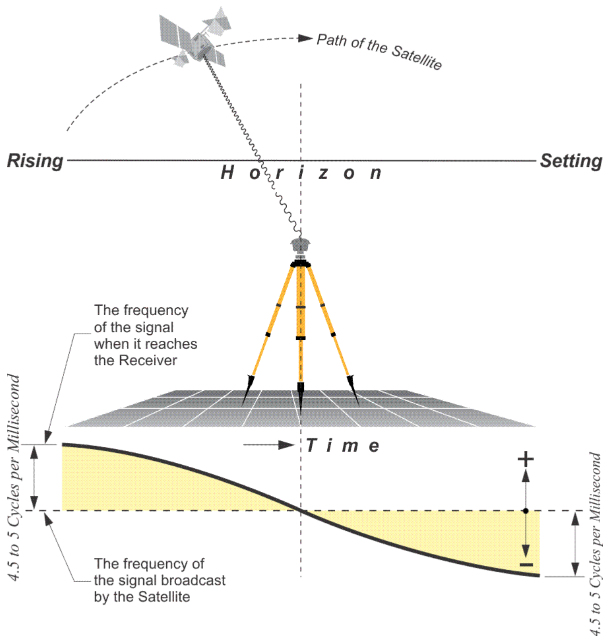

Path of the Satellite
Rising
Horizon
Setting
The frequency of the signal when it reaches the Receiver
4.5 to 5 Cycles per Millisecond
Time
+
−
The frequency of the signal broadcast by the Satellite
4.5 to 5 Cycles per Millisecond

Figure 1.2: Doppler Shift of Spacecraft Radio Wave [78]

3. Delta Differential One Way Ranging (ΔDOR): The radio waves from a spacecraft are measured at two different locations. The time difference between the signals in reaching the two locations is measured. This, along with the distance between the two stations, gives the **angular position** of the spacecraft. The same technique can be used on a natural radio source such as a Quasar simultaneously to detect and mitigate errors in these measurements. This comparison with the Quasar gives adds the $delta$ to ΔDOR.

These three tracking methods are used in conjunction with each other to determine the distance, velocity and angular position of satellites. Of these methods, ranging has had new research in the past decade that enables new possibilities for use in small satellites. With the advent of CubeSats and PocketQubes, small satellite missions have been gathering interest due to the low cost and faster development times of these small systems. Further, the small satellite enable distributed systems, which have benefits in terms of scientific return and mission security. These small satellites have stricter limitations on the SWaP available onboard.

Conventionally, satellite ranging required specially designed hardware and software onboard the satellite[34, 10]. A dedicated ranging signal was also needed, which took up bandwidth and power in the satellite communication system. Further, they did not work well with higher order data modulations, which can improve the signal bandwidth efficiency and signal amplifier efficiency. This essential means that the telemetry transmission speed is heavily restricted and power efficient equipment cannot be used when ranging is conducted.

For these reasons, it is not uncommon for small satellites to forgo the inclusion of a ranging system and rely on other tracking methods. Efforts have been made to implement a ranging solution for small satellites without dedicated hardware, which yielded usable range accuracies of (1 standard deviation or 1 σ) 650m[28] and 150m [38]. However, these made use of a fixed ping signal for ranging, which still had the drawback of not allowing data to be transferred while ranging was conducted and requiring a complicated time measurement system. Therefore, many small satellites in Earth orbit make use of the TLE data generated by the North American Aerospace Defense Command (NORAD) tracking systems, which provide a best case range accuracy of 1km.

In the past decade, a novel new method was invented which eliminates the presence of a ranging signal altogether and makes use of the satellite command signals and telemetry to communicate the time information needed for two-way ranging[3, 32, 30, 29, 33, 31, 75]. These are known as telemetry ranging techniques due to them making use of the telemetry channels to transfer ranging information. By eliminating the dedicated ranging signal, all the above discussed disadvantages are eliminated. Due to these factors, telemetry ranging was developed and validated in a lab test by NASA[31] and showed possibilities for excellent ranging results with better than conventional ranging performance(1m accuracy) being achieved in the lab. Therefore, it is set to be tested in flight for the first time in the Europa Clipper mission [33].

Telemetry ranging techniques do not place any specific requirements on the hardware on board the satellite. Thus, even without special hardware, a ranging solution can be enabled for small satellites using this technique. Theoretically, it is also possible to make small modifications to a satellite already in flight to enable telemetry ranging to be conducted. Despite these advantages, such an implementation is yet to be done.

Therefore, this thesis aims to research the use of a telemetry ranging system for small satellites and identify the main drivers/limitations for the performance. In the long term, doing so will expand the tracking design space for small satellites, allowing an additional choice for designers. In the short term:

1. Many amateur satellites in Earth orbit currently rely on NORAD tracking for positional data[43]. NORAD tracks satellites using radar and generates, TLE which are a representation of the satellite position at the point of tracking. Using these TLEs with the NORAD defined Simplified General Perturbations 4 (SGP4) orbital model, satellites can be tracked with a best case Root Mean Squared (RMS) accuracy of 1km. Achieving an accuracy greater than this with the telemetry ranging method will enable better orbit determination for all these satellites, and hence better navigation and scientific capabilities. With sufficient improvements to this system, better accuracies than from GPS could also be attainable, at which point other satellites can forgo the use of power hungry GPS receivers.
2. While radio measurements dominated the previous decade of tracking, the current one is of laser tracking, which offers magnitudes better performance[25]. One of the issues with these systems is the long lock on time due to their narrow Field of View (FOV). With a better position estimate from this method, laser lock on time for small satellites can be reduced. Further, it is also possible to implement the concepts of telemetry ranging with lasers[50].
3. Deep space missions need the use of a specialized tracking system, as neither GPS or TLEs are available. Having this ranging method for small satellites would allow them to carry out deep space missions with little (if any) additional requirements on the system. Satellites with this system will also be able to carry out science missions requiring higher range accuracies (such as BepiColumbo[79])

The Thesis started with a literature study in chapter 2 to explore the state of the art in ranging techniques. Here, the research gap was identified that telemetry ranging techniques are yet to be applied to small satellites.

For this purpose, the Delfi PQ satellite, whose developmental and testing equipment are readily available at TU Delft, was targeted as the platform of choice for this research. To guide the thesis, research questions were derived from the above goal in **chapter 3** as:

- **RQ-001: How can telemetry-based ranging methods be implemented using commercial hardware and software?**
  a) **What are the requirements placed on the hardware and software used?**
  b) **What are the main challenges, if any, affecting the viable use of telemetry ranging for Delfi-PQ and other small satellites?**
- **RQ-002: How do the telemetry ranging technique and the obtained ranging results compare to the results in published literature?**
  a) **What are differences, if any, between the research and literature?**
  b) **How do these differences affect the results obtained?**
- **RQ-003: What are the main parameters in Delfi-PQ that affect the performance of the method?**
  a) **By adjusting these parameters, are we capable of reaching a viable ranging solution?**
  b) **Why do the parameters affect the range performance in the Delfi-PQ satellite?**

The system requirements are also defined here based on stakeholder needs and expectations. Finally, this chapter ends with a description of the tools and software such as Delfi-PQ that are to be used for the thesis.

The thesis focuses on developing a viable ranging method, which involves creating the necessary algorithms and validating the available hardware for use with telemetry ranging. Therefore, **chapter 4** develops an algorithm based on signal correlation which will enable detection of signal departure and arrival times. The choices for the correlation method and its performance are discussed here, contributing to research questions RQ-001 and RQ-003 and RQ-003 (b).

This correlation code is further validated for use with the USRP B200 SDR, which is part of the TU Deft software defined radio testbed[65]and is the ground station for the developmental cycle. Using the sequential ranging method, **chapter 5** validates that the ground station is capable of ranging. The optimal setup and operational parameters for the USRP are also explored here. This contributes to RQ-001 and RQ-003 and also contributes to RQ-002 by validating sequential ranging performance with known performance metrics from literature.

Chapter 6 incrementally develops the telemetry ranging method for the Delfi-PQ. This provides a complete answer to all the three research questions.

The overall findings are summarized in **chapter 7** and recommendations for future work are provided.

2 Literature Study

This chapter presents a literature survey to obtain a clear picture of the state-of-the-art in the field of satellite ranging. It starts with discussing the background information necessary to understand the thesis in section 2.1, followed by a deeper dive into state of the art in ranging research. Section 2.2 explores the basics of signal processing. These two sections enable the identification of research gaps towards which the thesis will be directed. Finally, a tradeoff is conducted in which the most suitable ranging method for small satellites is identified for thesis research.

A few research questions were identified which were used to guide the literature study:

- Q1: What are the basics of satellite ranging and signal processing? (section 2.1 and section 2.2)
- Q2: What is the state-of-the-art in satellite ranging? (section 2.1.2)
- Q3: In the state of the art, are there any research gaps that are interesting to explore? (section 2.3)

2.1. Satellite Ranging

This section focuses on the relevant background information necessary to understand the context and content of the literature review.

Satellite Ranging is fundamentally a simple process: a radio signal is sent from one radio to another, and the travel time of this signal is measured. This time, multiplied by the speed of light, gives the distance between the radio antenna:

$$(Pseudo)Range = c * (t_r - t_t) \tag{2.1}$$

where c is the speed of light, t_r is the time the signal is received at the 2nd radio and t_t is the time the signal is sent from the first radio. A version of the process as described can be seen in Figure 2.6. This is called pseudo-range because the obtained range has in it a few sources of error in the time delay measurement. It is important to calibrate the measurements for these error sources to obtain the real range.

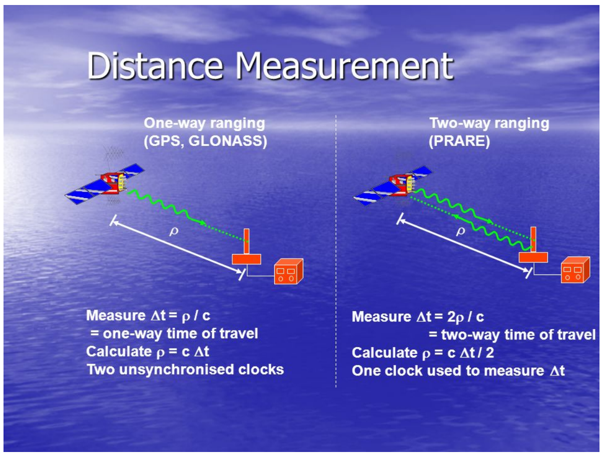


Figure 2.1: Satellite Ranging

This range value is then used to verify and correct orbit models, make corrections to the trajectory of the satellite and more. Signals are usually modulated onto a carrier wave as seen in Figure 2.2 in order to make it easier to capture and track the signal.

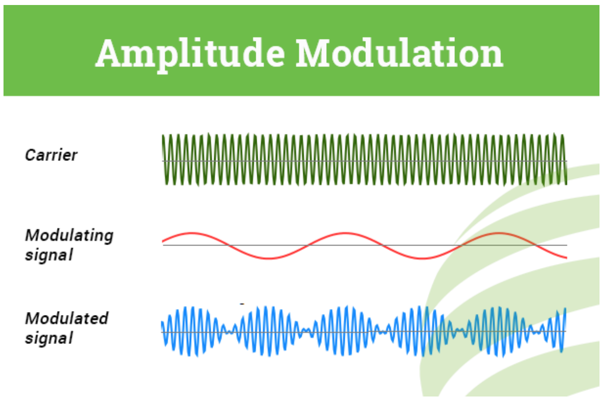


Figure 2.2: Phase Modulated Waveform [68]

Ranging is divided into a number of categories:

2.1.1. One Way Ranging

In one way ranging, the ranging signal is sent from one radio to another along with the transmission time. This information can then be used to estimate the travel time of the signal and find the range. This range information can then be used to guide satellite navigation or scientific missions. There are several benefits to one way ranging[62]:

- Path dependent effects such as ionospheric, tropospheric and solar plasma delays are reduced.
- Latency of obtaining navigation data is reduced.
- The amount of time available for data transfer is increased due to lesser ranging time.
- In case of ground to satellite transmission, the need to point the spacecraft antenna at the ground station is eliminated.

However, until the Deep Space Atomic Clock (DSAC)[62], one way ranging in space was not really feasible in deep space environments such as the Moon, Mars and beyond. This was due to lower long term stability of space based clocks. Ground based clocks have the advantage of essentially having endless freedom in its SWaP consumed. It is easier to design suitable ground based enclosures, so they can operate in ideal conditions.

Clock stability is measured through the use of Allan Deviation (ADEV), which is a statistical measure used to estimate the stability of noisy processes(originating in [2]). Allan variance is defined as:

$$\sigma_y^2(\tau) = E(\sigma_y^2(2, T, \tau)) = \frac{1}{2} * E((\bar{y}_{n+1} - \bar{y}_n)^2)) \tag{2.2}$$

where E is the expectation operator, which is the mean of all the values in the set.
and $\bar{y}_n$ is defined as:

$$\bar{y}_n = \frac{x(nT + \tau) - x(nT)}{\tau} \tag{2.3}$$

where x(t) is the time series containing all the data points for which the variance is estimated and where T is the dead time for which no measurements are taken and
τ is the time between the two measurements. Therefore, half the mean of the individual variances gives the Allan variance.

ADEV is given as the square root of Allan variance

$$ADEV = \sigma_y(\tau) = \sqrt{\sigma_y^2(\tau)} \tag{2.4}$$

The ADEV of clocks is defined as being over a period of time, such as a deviation of $1×10−15$ per day which is the value for hydrogen maser based atomic clocks [69]. Whereas crystal oscillators typically used have an ADEV of about $1×10−12$ per day. This is sufficiently low error for ranging happening over just minutes or hours. However, this over time builds up a difference in time or offset between multiple clocks. Typically, clocks on the ground are based on a high stability frequency standard such as hydrogen maser or cesium beam standard (ADEV of around 1×10^{-15} per day for Maser[69, 26]). Higher level standards are also available such as the National Institute of Standards and Technology's Ytterbium lattice atomic clock which has a one day Allen Deviation of $1 * 10^{-17}$[62].

In space, however, SWaP availability is restricted, and precise environmental control is expensive in terms of the power to implement. With the DSAC, novel mercury ion trap clock technology is leveraged to make a clock requiring a lower SWaP while maintaining a high ADEV of $3 * 10^{-15}$per day. This is achieved at SWaP of 19L, 19 kg and 43W, which allows it to be used in deep space missions. The SWaP of the clock is expected to be improved in later revisions without compromising performance, and it has been demonstrated that the ranging performance degradation due to frequency over time can be calibrated for[62].

Clocks in space prior to this operated with higher ADEV of the order of $1e^{-11}$(see [1]) over 1000 seconds, which is decent short term accuracy. This accuracy gives a range error of

$$RangeError = Clock\_Offset * c = 1e^{-11} \times 3e^8 = 0.003m \tag{2.5}$$

over 1000 seconds, which is pretty good. However, over a longer period such as over a day, the longer term accuracy of the clock is lower, in the order of $1e^{-9}$ over 1 day[42]. Calculating the range error here gives

$$RangeError = Clock\_Offset * c = 1e^{-9} \times 3e^8 = 0.3m \tag{2.6}$$

and the error keeps adding up over time unless corrected. Thus, this causes a problem when tracking is resumed after a long period of no tracking, as the clock offset will have added up as they were not compensated regularly, High accuracy clocks which are Ultra Stable Oscillators (USO) have been implemented in satellites in some missions such as the GPS and Cassini. Considering the case of the GPS satellites, these clocks must be updated several times a day to ensure they do not drift beyond predetermined requirements. This process of correction can be time-consuming depending on the travel time of a signal to the satellite and affects the capability of a satellite using one way range to be truly autonomous. The high stability of the DSAC is a big step in enabling one way range as a long term viable ranging technology. Further, the chip scale atomic clocks which were developed with a mass of 55g have also been tested to provide a worst case range estimate in the order of 100s of

[1]https://www.microsemi.com/product-directory/clocks-frequency-references/3824-chip-scale-atomic-clock-csac

meters in lunar scenario[60]. This does employ complex time correction methods . On the other hand, conventional two-way ranging accuracy is of the order of 1m [11].

Therefore, one-way ranging loses out in absolute performance to other ranging techniques due to the limitations of clock technology in small satellites.

2.1.2. Two Way Ranging

In two-way ranging, the Radio waves are sent from the ground station to the satellite and are received, retransmitted from the satellite. The idea of this method is to use a single clock to measure the transmission and receive time so that the clock errors seen in one way ranging are eliminated. These can further be subdivided into[72]:

1. Phase based Ranging.
2. Time Derived Ranging.
3. Hybrid Ranging.

Phase based Ranging

The ranging signals change their phase (see Figure 2.3) as they travel through space. This change is phase is related to the frequency of the signal, the propagation velocity of and distance travelled by the signal. Using methods such as correlation algorithms and symbol tracking loops, this phase change can be estimated. Based on this phase change, the time of flight of the signal can be estimated.

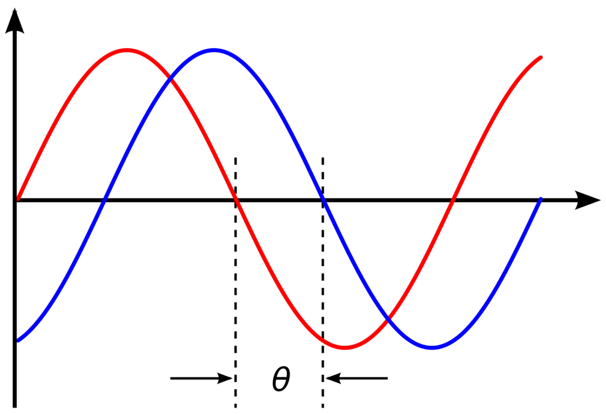


Figure 2.3: Phase Shifted Wave

Commonly used Phase based Techniques are:

1. Tone Ranging: Tone based ranging methods were the ranging methods used in the initial days of spaceflight. Among the different types of tones, using sinusoidal tones of increasing frequency or sequential ranging emerged as a widely used standard due to its use by NASA. Therefore, sequential ranging will be described here, but the concepts are applicable to other tone based ranging methods too.

   Sequential Ranging was used as a ranging technique by NASA and ESA in the 20th century, according to the timeline of standards documents[9, 26]. The idea of sequential ranging was that the ranging signal, which were tones of successively changing frequencies, were transmitted to the satellite. These tones were then received, demodulated, filtered, amplified and sent back.

   Initially, sequential ranging was done using square waves [9]. Later, sine waves were used instead, as they improved the spectral efficiency of the ranging process[76]. The sequential ranging signal consists of a series of varying frequency components. The first and highest frequency component is known as the range clock. The frequency of the nth component is given as

$$f_n = 2^{-7-n} \times f_t \tag{2.7}$$

for S band uplink and

$$f_n = 221/749 * 2^{-7-n} \times f_t \tag{2.8}$$

for X band and

$$f_n = 221/3599 * 2^{-7-n} \times f_t \tag{2.9}$$

for Ka Band, [11, 26] where f_t is the carrier wave frequency and n is the component number of the range clock, which is limited by the bandwidth of the satellite. This structure is used to resolve the ambiguity in the range measurement. Since range signals repeat after a certain period of phase, it is necessary to have a structure large enough to resolve the distance within expected values based on orbit simulations.

This coherence enables the use of the Doppler rate aiding technique to compensate for the change of phase during ranging measurement due to movement of the satellite. If the initial frequency of the signal is f_t and the uplink signal is Doppler shifted by a value d_u, then the received signal frequency is

$$\alpha_u \times f_t \tag{2.10}$$

This is then multiplied by β for the range clock(or G which is the transponding ratio for the carrier wave) and the downlink frequency at the spacecraft is

$$\beta \times \alpha_u \times f_t \tag{2.11}$$

This value is Doppler shift by α_d and the final frequency measured at the end of the ranging process is

$$\alpha_d \times \beta \times \alpha_u \times f_t \tag{2.12}$$

for the signal, and

$$\alpha_d \times G \times \alpha_u \times f_t \tag{2.13}$$

for the carrier wave.

Since β, G and f_t are known, a local model can be constructed to estimate the rate of change of phase. The rotation of the satellite will add a component to the relative velocity between the satellite and the observer, which cannot be accurately predicted unless the attitude of the satellite is very well known or controlled. This component will contribute to the relative velocity and hence the Doppler shift between the satellite and the observer. This will be present as an error which cannot easily be estimated[11, 76].

It is also possible to have a non-coherent ranging signal that also enables high performance ranging, as described in [14]. In this method, the ranging uplink clock is not coherent with the downlink carrier, and this causes drift errors from the clock of the spacecraft in generating the downlink signal, which causes frequency offset between uplink and downlink signals which cannot be completely compensated for. These measurement errors will arise from the frequency offset affecting the range value measured directly and from the loss in correlation amplitude due to the frequency offset affecting the measured correlation peak. The frequency offset error is given as [18]:

$$Range\ error\ due\ to\ \delta f_{RC} = \frac{c}{4}.\frac{\delta f_{RC}}{f_{RC}}.Tm \tag{2.14}$$

The amplitude loss is given as

$$A_c = |sinc(2\delta f_{RC} T)| \tag{2.15}$$

This will feed into the thermal noise equation for ranging (see section 5.2). where f_{RC} is the range clock and T is the integration time. Doppler rate aiding is still applied, but with the use of a predictive model that reduces the frequency offsets[11].

For the sequential range, the range ambiguity resolution is given as,

$$K = 2^{6+L}\ Range\ Units \tag{2.16}$$

where L is the component number of the last component. Each Range unit corresponds to about 0.94 ns of delay. So if a resolution of 2 sec is needed, then minimum L is 25. A resolution of 2 seconds means that the range of the spacecraft is known within $c * 2 = 6 * 10^8$m. Usually, the known range is bracket is much smaller.

In this method, the highest frequency tone is used to obtain high range resolutions, whereas the tones of lower frequency and hence lower range resolution are used to resolve ambiguity. The lowest frequency tone has a wavelength greater than the prior known range of the satellite. Once the approximate position with respect to this tone is identified, the closest higher frequency tone (which can range up to half the previous distance but more accurately) is sent to further refine the position estimate. This is done for all the ambiguity resolution tones until the range estimate builds up to a stage where the main high frequency tone can be used. Some lower frequency components of the signal are also multiplied by a higher frequency component to allow for spatial separation. This is known as chopping and essentially modulates the lower frequency signals into a higher frequency subcarrier known as the chop component, which is usually the range clock[11].

For the actual ranging process, each ranging signal component is tracked for an integer number of seconds. This is usually a number T_1 for the range clock and T_2 for the rest of the components. A one-second delay in switching between the components also exists. The range clock starts before transmit time to ensure ranging does not start before it is received and persists for 1 second afterward. This along with a second of switching time gives us the 3 in equation 2.17. Therefore, the total ranging time is

$$T = T_1 + 3 + (L - C) * (T_2 + 1)\ s \tag{2.17}$$

By correlating the initial and final transmitted sequential wave, the phase change and hence the range of the spacecraft can be determined.

2. PN Ranging: This ranging method was initially proposed as τ ranging as competition for sequential ranging decades ago. During those times, establishment of parallel correlators which were necessary to allow for efficient function of the technique was not possible due to economic considerations. However, in the 1990s, when correleators became cheaper due to improvements in digital signal processing[66, 15]., it became possible to implement the technique to its fullest extent. Conventionally, sequential ranging makes use of only one simple correlator, with the disadvantage being the more complicated timing algorithm. When this timing is improper, the range measurement is affected adversely. In the case of PN ranging, however, multiple correlators are needed to receive the code in parallel, increasing the complexity of the correlators but removing the strict timing requirement[67, 23].

The phase delay of the ranging signal is measured in terms of Range Units. The conversion from Range Units(RU) to time is given as

$$Two\ Way\ Time\ Delay = \frac{2 \times RU}{f_s} \tag{2.18}$$

for S band,

$$Two\ Way\ Time\ Delay = \frac{749}{221} \times \frac{2 \times RU}{f_x} \tag{2.19}$$

for K band,

$$Two\ Way\ Time\ Delay = \frac{3599}{221} \times \frac{2 \times RU}{f_{ka}} \tag{2.20}$$

for Ka band, where the f is the carrier frequency. This equation arises from coherence of the signal frequency with the carrier frequency.

Pseudo Noise (PN) ranging has a few key advantages over sequential ranging

(a) Firstly, at Signal to Noise Ratio (SNR) ratios higher than a certain level, PN ranging has a performance advantage over sequential ranging.

(b) Secondly, and more importantly, PN ranging allows for regenerative ranging due to its signal structure. In regenerative ranging, once the ranging signal is acquired by the satellite, a low noise delay locked loop is used to track the phase of the received signal. Since the PN code has a Pseudo Random Noise (PRN) format, a fresh copy can be cheaply regenerated and have its phase matched to the measured phase on board. This new noise free copy of the signal is sent on the downlink. The advantage of regenerative PN ranging is that it increases the SNR of the signal by several tens of db [23](exact number not specified). This can either be used for the ranging signal, which directly translates to a reduction in the jitter of the ranging process, or power can be reduced to maintain the ranging accuracy and improve the telemetry SNR[18].

The PN ranging code is a series of 1s and 0s arranged in a specific order. It is built from 6 basic components, which can be seen in Table 2.1. From these components, different PN noise structures are possible, such as the Deep Space Network (DSN) range code [18], Consultative Committee for Space Data Systems (CCSDS) T4B, CCSDS T2B[56].

Table 2.1: Components of Pseudo Noise Ranging Code[18]

bn	Ranging Code Component	Length(λ_n)
b1	1,0	2
b2	1, 1, 1, 0, 0, 1, 0	7
b3	1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0	11
b4	1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0	15
b5	1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0	19
b6	1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0	23

The DSN range code is the first code tested and verified at NASA. The code is defined as

$$b'(i) = b'_1(i) \cup [b'_2(i) \cap b'_3(i) \cap b'_4(i) \cap b'_5(i) \cap b'_6(i)] \tag{2.21}$$

where $\cup$ is the logical OR operator and $\cup$ is the logical AND operator and

$$b'_n(i) = b_n(i \times mod\lambda_n) \tag{2.22}$$

$b'_n(i)$ is a periodic code extended to period λ_n by repetition of the initial code[18]. The total length of the code is given as the product of the individual lengths of the components. The code repeats after its length.

$$Length\ L = \prod_{n=1}^{6} \lambda_n = 1,009,470 \tag{2.23}$$

The ambiguity resolution of this code is given as

$$ambiguity\ resolution = \frac{c \times L}{4 f_{RC}} \tag{2.24}$$

where f_{RC} is the frequency of the range clock, which is the length 2 component. The DSN PN ranging codes themselves have a few advantages over sequential ranging tones. The $\cap$ terms for the codes $b2$ to $b6$ become zero most of the time and the final sequence is somewhat regular. The main sequence is dominated by b'_1. This also decreases the tracking noise in the signals as a larger amount of the sideband power is contributed to the highest frequency spectral lines [23].

These CCSDS T4B and T2B codes make use of a weighted range clocks. Higher weights for the range clock allow for better range performance, whereas lower weights allow for a quicker acquisition time at the cost of lower range performance. The T2B code is mainly made for use in lower T (acquisition time) or SNR environments[56].

3. GMSK+PN Ranging: The use of PN ranging signal eliminates the use of some types of modulation for the telemetry data. This is non-ideal, as bandwidth is of limited availability and can

get crowded very quickly when considering popular mission destinations such as the moon or Mars[52]. Thus, the exploration of bandwidth-efficient modulation techniques gave rise to Gaussian Minimum Shift Keying (GMSK) + PN ranging[52].

This method makes use of GMSK to modulate the telemetry, while the PN ranging signal is used to conduct ranging. This adds a few advantages to the resolution of the ranging and performance at high data rates.

GMSK + PN ranging arose from the need to efficiently use the bandwidth available in popular space destinations such as Mars. Thus, the use of bandwidth efficient modulations

The schematics of the method can be seen in Figure 2.4.

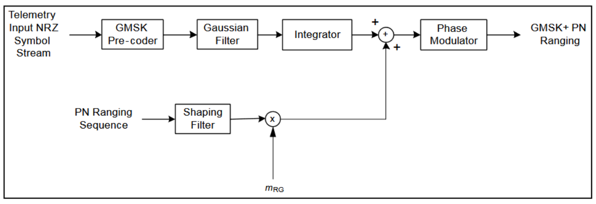


Figure 2.4: Schematics of GMSK+PN Ranging [58].

GMSK+PN ranging offers a few advantages compared to PN ranging. Its exact characteristics are not discussed here and can be found in [52].

The phase based ranging techniques have a few drawbacks. For example, while designing signals for use in sequential and PN ranging, it is important to prevent interference of ranging and telemetry signals in the modulated carrier wave[11]. This happens when their spectral components overlap in frequency. What does a spectral overlap look like? Consider Figure 2.5, where the orange bumps and blue bars represent waves of different frequencies. If part of one of them occupy the same frequency with another, the parts which are together (or spectral components) will add up and change the signal which is being transmitted, which is the interference that needs to be avoided.

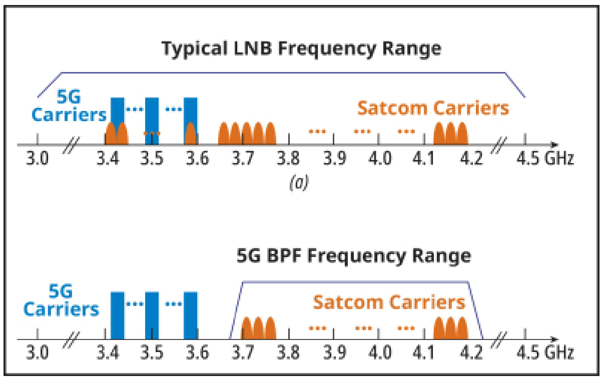


Figure 2.5: Communication signals plotted in the frequency domain[19]

This effect is less pronounced in deep space ranging, where the ranging sidebands are not large enough to cause a noticeable change in the noise floor of the telemetry signals, meaning the amplitude of the ranging signal (or size of signal as in how the orange signals are smaller than the blue signals in Figure 2.5) is small enough to not change the size of the telemetry data noticeably. The telemetry signals do cause a change in the ranging floor, or the parts of the ranging signal, which must be and are usually countered by changing the modulation indices of the modulation scheme used. However, in near Earth ranging, this effect is much more pronounced [41]. The data rate of telemetry is also reduced when ranging is done, as the ranging signal occupies part of the bandwidth.

Time Derived Ranging

The time of transmission and of reception of the ranging signals are recorded in the ground and on the satellite, as in Figure 2.6. The time spent by a signal on board the satellite can instead also be calibrated for using a large number of measurements. Time derived ranging techniques have been explored for small satellites in [28] where satellites were pinged with a ranging signal and the satellite responded with the same packet. The goal was to achieve a better than TLE (1KM) range performance, and a 1σ ranging accuracy of 650m for the satellites was achieved. Further research[38] built on this method, and with optimized satellite parameters achieved a 1σ ranging performance of 155m per measurement. These time derived ranging methods are advantageous in that they are simple and can be implemented in any satellite with varying results and offer reasonable performance. On the downside, they occupy the entire bandwidth for conducting ranging, need the system to be coded to allow low system latency and do not allow for data transfer on the downlink or the uplink.

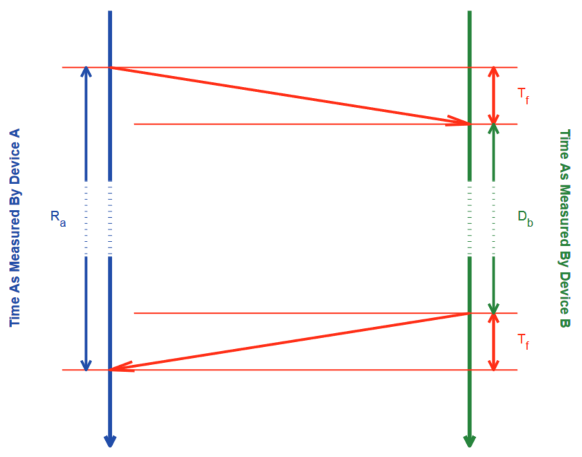


Figure 2.6: Two-way ranging between two devices[49]

Hybrid Ranging

These ranging methods arise from the telemetry ranging class of methods where both time measurements are taken on the ground and phase information of signals are used to estimate the time spent by signals on board the satellite.

1. Telemetry Ranging: This was proposed as a novel technique by NASA was proposed to eliminate the use of a ranging signal on the downlink and rely on the telemetry signal to transmit the phase information at the spacecraft down to the ground station [3]. The working of this technique is given in Figure 2.7.

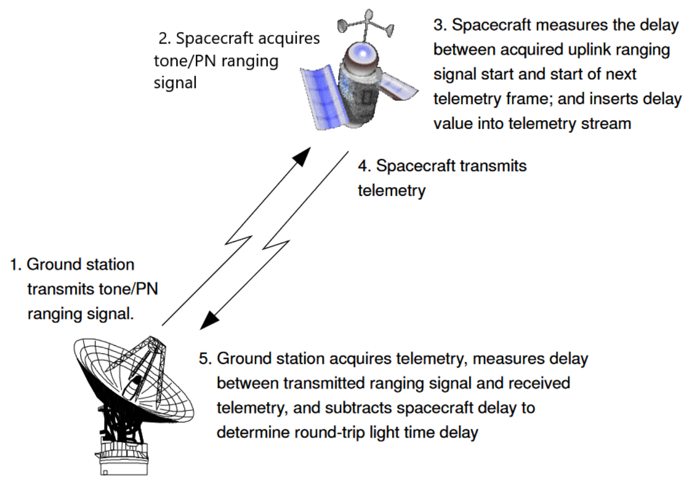

2. Spacecraft acquires tone/PN ranging signal
3. Spacecraft measures the delay between acquired uplink ranging signal start and start of next telemetry frame; and inserts delay value into telemetry stream
4. Spacecraft transmits telemetry
1. Ground station transmits tone/PN ranging signal.
5. Ground station acquires telemetry, measures delay between transmitted ranging signal and received telemetry, and subtracts spacecraft delay to determine round-trip light time delay

Figure 2.7: Telemetry Ranging as in literature[3].

Following the initial paper, several improvements were proposed that allowed the technique to be implemented using software defined radios[33]. The main improvement was that the spacecraft no longer measured the delay between ranging signal arrival and telemetry frame transmission (step 3 in Figure 2.7 was deleted). Instead, a measure of the phase of the tone/PN ranging signal was made at discrete time intervals using specially made Digital Phase Locked Loop (DPLL) as shown below in Figure 2.8.

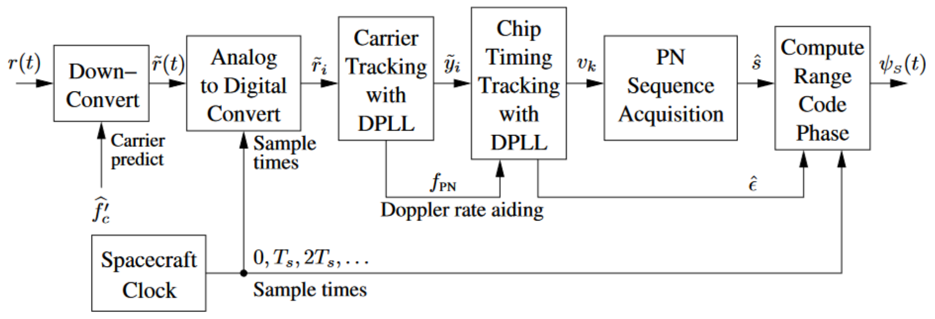


Figure 2.8: Telemetry Ranging uplink signal processing[32].

These discrete time intervals were chosen to coincide with the transmission time of downlink telemetry frames. The satellite associated the phase measured with the frame number of the telemetry frame whose transmission triggered the measurement. The pairs of phase value and corresponding frame number were sent down with later telemetry data and used to correct the range measurement in post-processing. By measuring the onboard time in terms of the phase, time tagging errors (see section C.1.3) in the satellite were eliminated.

The technique was validated in lab tests and is set to fly in the Europa Clipper Mission[31].

Telemetry ranging offers the following advantages over the prior methods, because it eliminates the ranging signal on the downlink (see Figure 2.9):

(a) Increases the bandwidth available for telemetry data, which is important in scenarios where multiple satellite have to share a limited bandwidth for communication such as the Moon[52].
(b) Allowing use of higher order modulations which further increase available bandwidth.
(c) Allows use of continuous envelope modulations such as GMSK which allow for linear operation of amplifiers, reducing design complexity and power usage.

(d) Activating ranging whenever telemetry and telecommands are being transferred, eliminating the need to actively switch on and off ranging, simplifying operations. Previously, ranging and telemetry passes had to be traded off against each other because the use of one significantly affected the performance of the other.

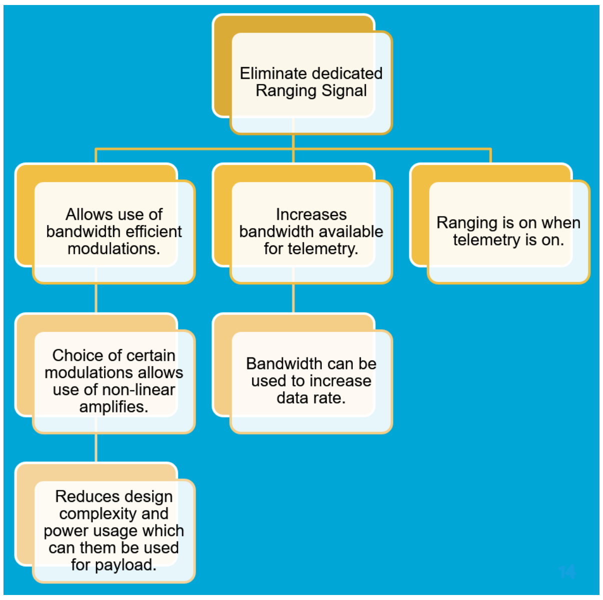

Eliminate dedicated Ranging Signal
Allows use of bandwidth efficient modulations.
Increases bandwidth available for telemetry.
Ranging is on when telemetry is on.
Choice of certain modulations allows use of non-linear amplifies.
Bandwidth can be used to increase data rate.
Reduces design complexity and power usage which can them be used for payload.

Figure 2.9: Telemetry Ranging advantages

The limiting factors for the technique were identified as the accuracy of the timings estimates in the ground and in the spacecraft[32, 74]. These timing estimates are made using digital transition tracking loops on the ground and specially designed phase locked loops for the uplink P-N noise code, as in [23]. The jitter of these loops in estimating the structure of the original signal gives us the error in the range measurements. With telemetry ranging, accuracies of 1 m were achieved at data rates as low as 2Kbps[31].

Figure 2.10 shows the timeline and structure of the ranging process. The locations P_T,P_S and P_R refer to the processing stations on the ground, spacecraft, and ground receiver units respectively. A_{DSS} and A_{SC} are the ground station and Spacecraft antennas.

The two-way geometric delay of the time is given as,

$$\tau(t) = \tau_u(t) + \tau_d(t) \tag{2.25}$$

where τ_s is the time delay on the uplink and τ_d is the time delay on the downlink. From Figure 2.10, we can derive the geometric delay as the total time delay measured minus the internal delays,

$$\tau(t_R) = (t_R - t_T) + (\tau_u^{SC} + \tau_d^{SC} + \tau_u^{DSS} + \tau_d^{DSS}) \tag{2.26}$$

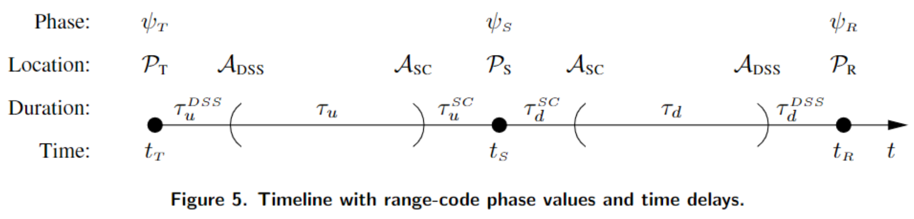


Figure 2.10: Timeline of Telemetry Ranging [30]

Two phase measurements need to be done in order to measure the delay. Usually, the derivative of the phase is available in post-processing. The derivative is given as

$$\int_{t_T}^{t_R} \dot{\psi}(t)dt = \psi_t(t_R) - \psi_T(t_T) \tag{2.27}$$

In order to determine the phase delay, measurements are done at t_S and t_R respectively. The time t_S is not time tagged on board the spacecraft. Its value is not known or needed, and it is only used here for notational purposes. The phase sample measured at t_S which is $\phi_s(t_S)$ is inserted into the telemetry frame following the frame whose start triggered its measurement. This phase corresponds to the transmitted phase at the start as seen in Figure 2.11.

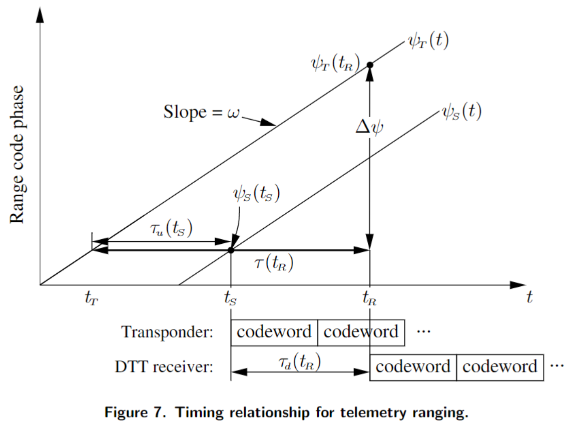


Figure 2.11: Telemetry Phase Relationship [30]

$$\psi_S(t_S) = \psi_T(t_T) \tag{2.28}$$

Following this phase being received at the ground station at time t_R, the signal phase in the uplink array is measured at time, t_R which gives us the value of $\psi_t(t_R)$. Since we know the phases and their derivative, we can estimate t_T numerically from Equation 2.27. Since the two-way delay is now known, only the calibration of each instrument delay is to be done before each tracking pass. Then, using Equation 2.26, we can estimate the total phase delay using telemetry ranging.

These phases are then time tagged using a 80MHz reference for DSN. Time tag accuracies of 1 ns can be achieved on the ground through a least squares fit. Following this, the phase is evaluated at the tagged time using the 1pps clock epoch samples and the model of $\dot{(\psi)}_T(t)$ along with the known time tag accuracy. Using this evaluated phase, the two-way timing delay can be estimated[30].

2. Telecommand/Telemetry Ranging: In this method, the uplink signal is also eliminated, and the uplink phase data is sent through the use of modifications to the telecommand data[75]. This brings an additional advantage in that the specialized tracking loops used to handle the ranging signal on the uplink are no longer needed. Telecommand data is sent in the form of Communication Link Transmission Unit (CLTU), according to CCSDS protocol as in Figure 2.12.

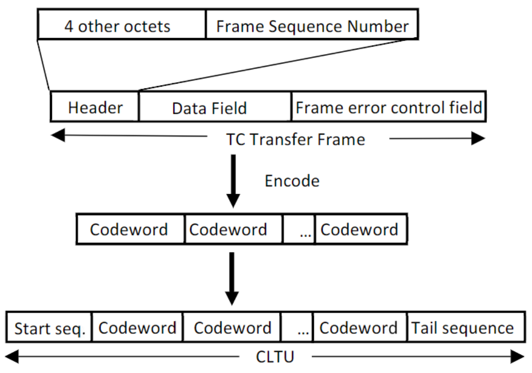


Figure 2.12: CLTU structure [75]

The ground station records the time the CLTU are sent, and the spacecraft keeps a track of the number and fractional part of bits arrived after the latest CLTU it received. When the satellite transmits a telemetry packet or frame, this integer/fractional count, the CLTU's serial number and the telemetry frame's serial number are stored into memory and associated with each other. These are then sent down later with the telemetry. The ground knows the time at which each command was sent (say t_t) and each telemetry frame was received (say t_r) as in Figure 2.13. The integer/fractional bit count represents the time passing between the arrival of command at the satellite and transmission of telemetry frame from the satellite as:

$$t_{sr} - t_{st} = t_d = \frac{Integer\_count + Fractional\_count}{Bitrate} \tag{2.29}$$

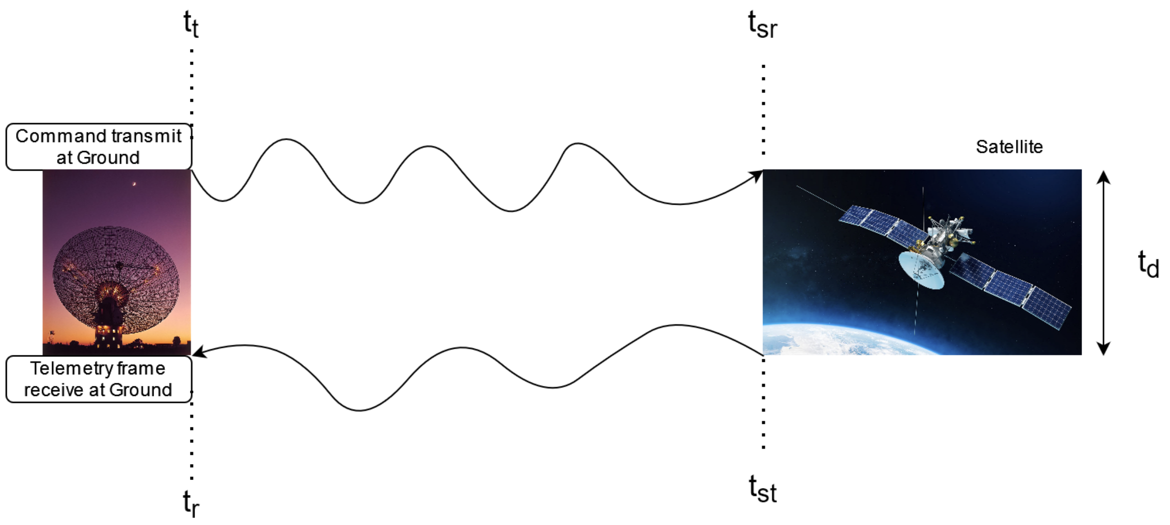


Figure 2.13: Telecommand Ranging Timing

However, practically, the arrival time of the signal (t_{sr}) is not registered perfectly at the satellite. This is because tracking loops (see section 2.2.3), which recover the clock (timing) of signals to decode the received signal, do not do this recovery perfectly. There are errors upto 1/10th of a bit[75] in the new signal's timing as compared to the received signal. The estimate of t_{sr} which is made by the satellite tracking loop will have this error (which flows down to t_d).

[75] get around this issue by creating a new method to accurately determine the clock recovery error. This is called the demod-remod correlation method. The demod-remod method stores the non demodulated signal received by the satellite. This is the signal before it is passed into the

tracking loops ($\tilde{r_i}$ in Figure 2.8). After passing through the clock recovery loop and demodulator, a copy of the signal is remodulated to obtain a noise free, but slightly time shifted copy. These two signals are now correlated with each other to obtain the time shift. This is then corrected for in the range measurement. However, the process typically takes a few microseconds, which will need to be compensated for in the final range measurement. To estimate the additional time spent in the system, the noiseless copy is correlated with itself and with the noisy copies to obtain two correlation peaks as in Figure 2.14. The distance between the correlation peaks is the time spent in the system.

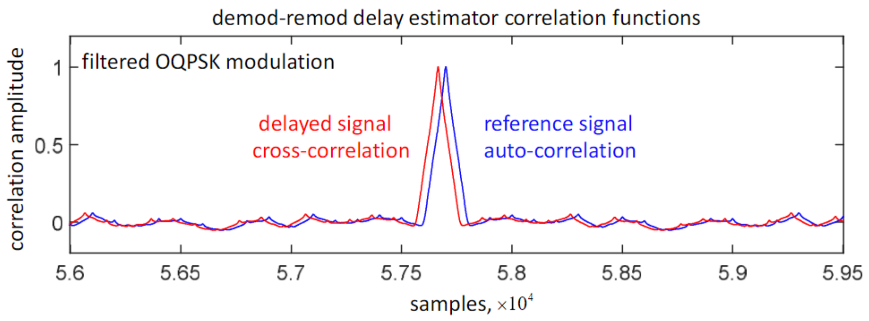


Figure 2.14: Demod-Remod Delay Estimator output[75]

Since the system can estimate arrival time better, the error in determining the arrival time comes from the correlation error between the noiseless and noisy signal which estimated the time spent in the system. The maximum achievable performance of such a system (which is an unbiased estimator) is given by the Cramer Rao lower bound. This bound is determined by the amount of noise in the system and is given as

$$var(\tau - \tau') >= (E\{|\frac{\partial}{\partial\tau}\Lambda(r|\tau)|^2\})^{=}\frac{\sigma_n^2}{4}(\sum_{i=0}^{N-1}|\frac{\partial\tilde{s_i}(\tau)}{\partial\tau}|^2)^{-1} \tag{2.30}$$

Since the noisy signal is processed before correlation, it is a known waveform. The transition regions can be identified and the signals in the regions outside for the transition regions can be set to either 1 or 0 based on this knowledge. This process is called windowing, and this improves the correlation performance by reducing the noise in the signal. Using this, performance was achieved as seen in Figure 2.15. It can be seen that the Cramer Rao bound is achieved for medium to high SNR values and even for medium low values.

On the ground, the telemetry is received and decoded as in telemetry ranging. In post processing, each CLTU is associated with a telemetry frame. Since the arrival time of CLTU having a specific fractional phase was synchronized with the departure time of a telemetry frame, the point on that exact CLTU having the same phase will be the time of departure and the corresponding telemetry frame's time of arrival is known. The time between the two gives the time the signals spent in air and hence give the ranges[75].

The range accuracy is dependent on the sample rate usable on the satellite for the demod remod correlation and is given as

$$Range\ accuracy = \frac{c \times standard\ deviation}{sample\ rate} \tag{2.31}$$

For an SNR of 30db and an assumed sample rate of 10Mega samples per second, this gives

$$Range\ error = \frac{3 * 10^8 \times 0.001}{10^7} = 0.03m \tag{2.32}$$

Thus, for high enough sample rates and SNRs, the method produces a low error in the range measurement.

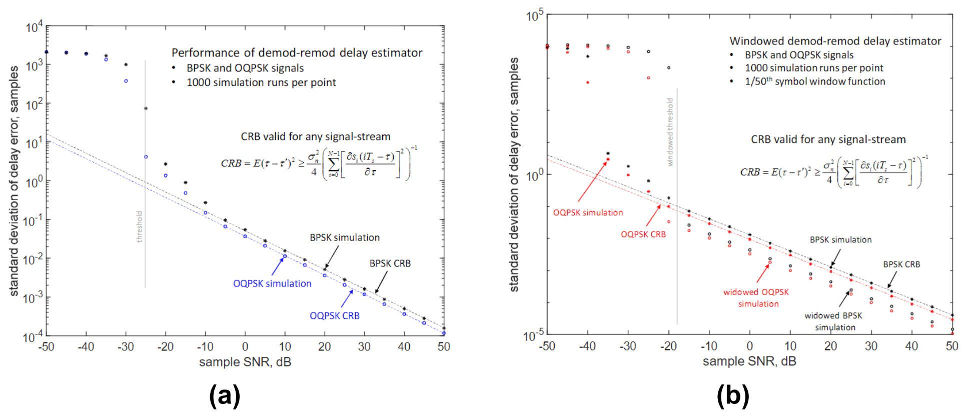


Figure 2.15: Cramer Rao Bounds for Delay Estimator **(a)**without and **(b)**with windowing

On the downlink, the signal is tracked using a Data Transition Tracking Loop (DTTL), which also adds to the delay error in the range measurement. The DTTL Cramer Rao bound is given as,

$$\sigma_d^2 = \frac{W \times B_L}{2 \times S_L \times \frac{P}{N_0}} \tag{2.33}$$

where the squaring loss is given in equation 12 of [75].

For high SNRs $S_L -> 1$, therefore

$$\sigma_d^2 = \frac{W \times B_L}{2 \times \frac{P}{N_0}} \tag{2.34}$$

For Low SNRS, the Cramer Rao bound is approximately

$$\sigma_d^2 = \frac{W \times B_L}{2 \times R_s \times \frac{P}{N_0}} \tag{2.35}$$

The end to end Root Mean Squared Error (RMSE) is given as the square root of the sum of squares of individual errors of the demod remod delay estimator error on the uplink and the DTTL on the downlink.

$$RMSE = \sqrt{\Sigma_i^n x_i^2} \tag{2.36}$$

The overall performance can be seen in comparison with PN ranging in Figure 2.16. Here it can be seen that the performances are similar over different SNR regions. The dotted lines represent the uplink (demod-remod estimator) and downlink (DTTL) CRBs. Equation 2.31 can be applied to find the ranging error.

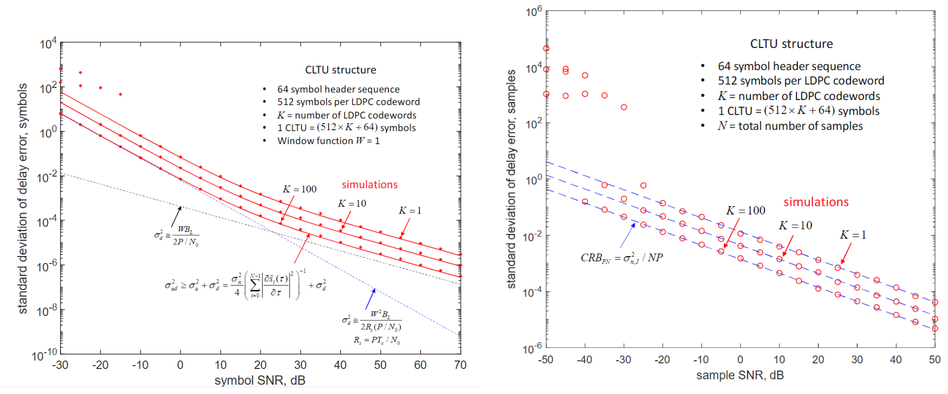


Figure 2.16: End to End Performance and CRB for Telecommand Ranging and PN ranging[75]

2.1.3. Three Way Ranging

The radio waves are sent from ground station to satellite and from satellite to another ground station or satellite as visible in Figure 2.17. This is done in specific cases which place requirements on line of sight or to increase the amount of time available for ranging. Two way ranging methods are used in three-way ranging with the added disadvantage of using two clocks to measure the time taken for the waves to travel. However, modern methods exist to synchronize the ground clocks to a higher accuracy than in one way ranging[47, 64].

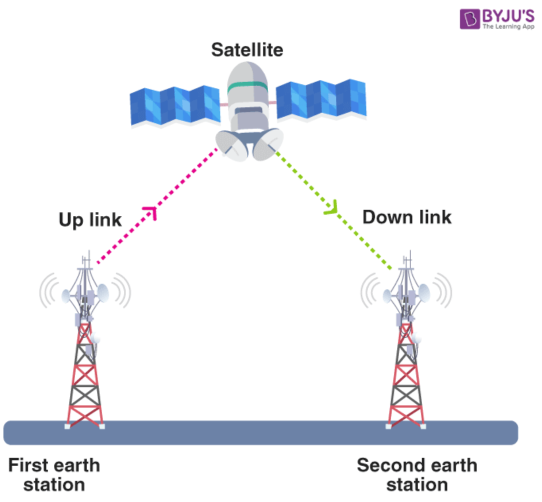


Figure 2.17: Three Way Ranging[20]

2.1.4. Range versus Range-Rate

It is possible to estimate the range or orbital position of a satellite using range or range-rate (Doppler) observables. Normally, range-rate observables are preferred for many applications, as it is easier to correct for errors in them compared to range measurements [36]. This is seen in scenarios where ranging is performed between a facility in Earth and the satellite, where the error from the atmosphere is more easily compensated for the range-rate measurement.

However, there are some scenarios where range measurements provide better performance in estimating the position of the satellite. When considering a cross-link ranging case where the ranging is done between two satellites and the relative velocities between the satellites are sufficiently low, range observables perform better than range rate observables[71, 73]. However, range rate measurements supplement and are supplemented by range measurements, therefore it would be ideal to use both in case equipment allows. Especially since telemetry ranging does not require specialized hardware, it would be easier to enable a system to use telemetry ranging and Doppler together.

2.2. Signal Processing

This section discusses some of the signal processing used in satellite communication systems. Satellite communication signals and sent and received at very high carrier frequencies. They are down-converted to a lower frequency for processing. Then, a process known as quantization is done in order to convert an analog signal to a digital signal before further processing is done.

2.2.1. Down-conversion

The uplink signal at radio frequency is converted to an intermediate frequency on arrival. This is done as the design of radio frequency filters is more flexible at the lower intermediate frequency. This is done through signal mixing operations such as heterodyning or homodyning. Further processing includes conversion of signals from analog to digital using quantization[5]. This Intermediate Frequency (IF) signal is then processed, and the signal information is extracted to obtain the original information, which is called the baseband signal. The baseband signal is then corrected for Doppler and integrated over a period of time. The integration time is an important design parameter for the signal processing, as the factor $Bandwidth \times Integration\ Time$ affects the amount of noise in the signal. Post this, the carrier phase is tracked using phase locked loops and the signal phase is tracked using delay locked loops[5]. This can be seen in Figure 2.18.

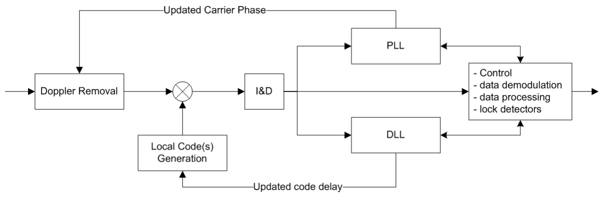


Figure 2.18: Generic Baseband Processing Flow Diagram for GPS Radars[5]

2.2.2. Quantization

Quantization means mapping an infinite set of values(analog signal) by a smaller, finite set of values(which form the digital signal). This is done by sampling the signal at different times. The Nyquist criterion states that the signal sampling frequency must be at least twice the highest frequency contained in the signal in order to adequately capture the signal information. Otherwise, signal aliasing will occur which introduces errors into the signal as seen in Figure 2.19.

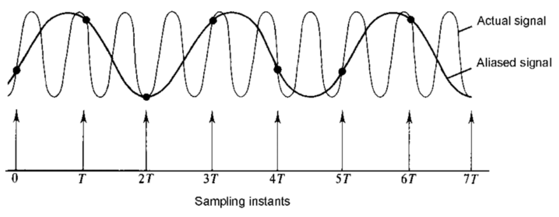


Figure 2.19: Signal Aliasing [44]

2.2.3. Carrier and Data Signal Tracking

The carrier and signal waves are tracked post conversion to baseband using Phase Locked Loop (PLL) and Delay Locked Loop (DLL).

Phase Locked Loop

A PLL produces another signal having the same phase or a phase proportional to the phase of an incoming signal. It makes use of negative feedback to enable generation of a tunable output with good accuracy and noise performance[8]. The elements of building up a PLL are simple, as seen in Figure 2.20. The input signal to the oscillator is divided by the R divider to a lower phase detector frequency. This phase is compared with the phase of the N divider, and correction pulses proportional to the error in phases are generated. These correction pulses are then sent to a loop filter, which is a specially designed low pass filter, whose output voltage is used to drive the Voltage Controlled Oscillator (VCO). The output of the VCO is sent through the N divider and divided down to a new phase through the N divider, which is sent back into the phase detector. This creates a negative feedback loop where the VCO frequency is changed to keep the output phase locked if it drifts.

The specific design of each of the PLL elements is application dependent and more involved, therefore is not discussed in the scope of the study.

There are several types of phase locked loops. For ranging applications, all digital phase locked loops (ADPLL) which use a Numerical Controlled Oscillator (NCO) and in some cases software phase locked loops (SPLL) are used. A block diagram of such a loop is given in Figure 2.21.

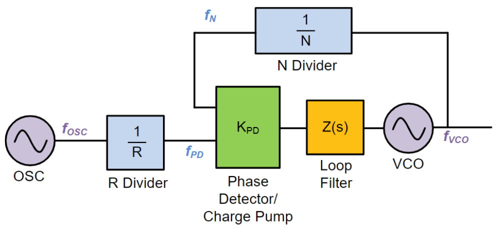


Figure 1.1 *The Basic PLL*

Figure 2.20: Basic PLL block diagram [8]

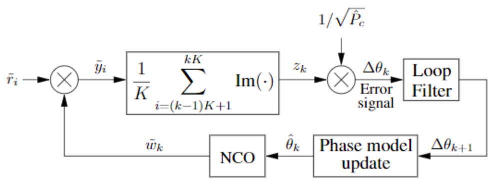


Figure 16. DPLL.

Figure 2.21: Block Diagram of Digital Phase Locked Loops [32]

Delay locked Loop(DLL)

A delay locked loop provides a copy of the signal with its phase delayed by a specified value. The difference is that a delay locked loop produces the lock differently, This is used in certain circuits to produce a delayed replica of the signal either for correlation purposes as in Figure 2.18 or to produce a lower noise replica as in Figure 2.22.

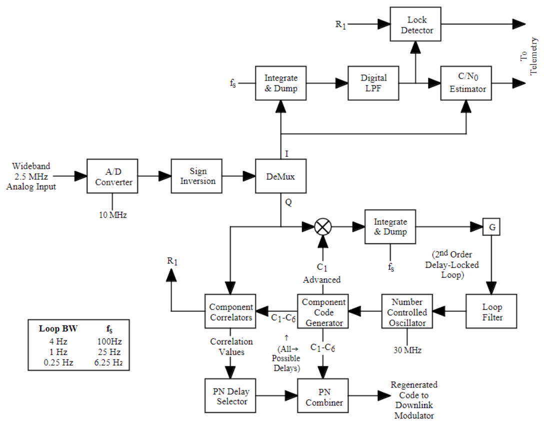


Figure 2.22: Delay Locked Loop of New Horizon used for regenerative ranging[23]

Symbol Tracking

Post lock to a carrier using phase locked loops (or Costas loop named after its inventor for suppressed carrier modulations), it is necessary to track the baseband data's timing using a symbol (or baseband) tracking loop. The symbol tracking loop, when used in ranging, in order to achieve less than 1 m resolution ($\simeq$ 3.3ns of time), makes use of data-transition tracking loops (DTTLs) for telemetry data and specially designed phase locked loops for PN sequences[74, 23].

There are tradeoffs to be done in tracking loop design. When operating with weak signals in a noisy environment, it is often desirable to optimize the loop bandwidth to minimize the rms phase error. This optimization depends on the phase spectrum of the oscillator, especially on spacecraft where the reference oscillator is often an ultrastable oscillator (USO) with significant phase noise at low frequencies.

Phase-locked loops track out phase instabilities within the closed-loop bandwidth, suggesting that the loop bandwidth B_L should be made as large as possible in order to track out both low- and high-frequency phase noise contributions to the total phase error. However, additive noise entering the tracking loop also contributes to the total phase error, and this component increases with B_L. Therefore, an optimum value of loop bandwidth exists that minimizes the total phase error, by excluding noise while tracking the oscillator's phase fluctuations. The goal of digital loop design is to determine the gain coefficients to achieve the desired loop bandwidth, once the performance has been optimized [74].

2.3. Tradeoff of Ranging Process

Out of the methods discussed above, a method is chosen for further work in the master thesis. The aim is to apply the method in a more accessible environment (without expensive DSN equipment) and test the limits of ranging process, identifying the primary factors affecting success, accuracy in the meantime. In this section, a trade-off table is presented in the end. From this tradeoff, a research gap is identified and proposed for future research answering question Q3.

2.3.1. Tradeoff between the methods

The above discussed methods are now compared using a tradeoff table. The parameters are assigned a weight from 1-10 based on their importance.

Performance (7)

The performances of the ranging methods are an important criterion in choosing the method to use. Certain mission scenarios require a greater ranging accuracy, whereas other mission can make do with lower accuracy but may place a stricter burden on other ranging parameters like acquisition time. Since it is important for a method to maintain or improve on the performance of prior methods,

System Complexity (6)

Each method adds a layer of complexity to the system software and hardware. An idea of these costs are essential. More expensive missions can afford to use expensive hardware and software on the satellite and in the ground for dedicated ranging support. Cheaper missions may prefer methods that can be implemented within the available constraints.

Compatibility with Modulations (7)

Different modulation schemes bring different advantages to the ranging process. They allow for greater efficiency in the use of the bandwidth. Some modulations also allow for more efficient hardware to be used, which increases the amount of power available for use. Therefore, compatibility with modulations is important for the method as it would future-proof it for different use cases.

Eliminate Ranging Signal(8)

Eliminating the ranging signal makes room in the signal bandwidth which can be allocated completely to the telemetry and carrier. This also reduces the complexity in the design of the ranging signal in that the interference between ranging and telemetry signals are eliminated. Finally, this also eliminates the need to choose between ranging and telemetry.

Age (3)

The age of the ranging method may indicate the amount of the research done into the method. For our purpose, we would prefer methods with more promise than well explored methods for further development, as it is more likely to lead to avenues of research.

Overall, the following methods are now traded off:

1. Sequential Tone Ranging
2. Pseudo Noise Ranging
3. GMSK+Pseudo Noise Ranging
4. Telemetry Ranging
5. Telecommand/Telemetry ranging

Table 2.2: Trade off Scoring

Green - Excellent; exceeds requirements
Blue - Good; meets requirements
Yellow - Correctable deficiencies
Red - Unacceptable

The Trade-off matrix is scored according to table 2.2 and is given in table 2.3.

Table 2.3: Trade off Matrix for ranging techniques

Method\Criterion	Performance (7)	System Complexity (6)	Compatibility with Modulations (7)	Eliminate Ranging Signal(8)	Age (3)
Sequential Ranging	Worse than Pseudo Noise Ranging	Slightly complex due to signal timing requirements	Not compatible with higher order modulations	No	Used for Ranging in many missions
Pseudo Noise (PN) Ranging	Considered as conventional Ranging Performance, sub meter accuracy	High complexity due to multiple correlator and regenerative circuit.	Not compatible with higher order modulations	No	Used for Ranging in many missions
GMSK+PN Ranging	Same as Pseudo Noise Ranging	High complexity due to multiple correlator and regenerative circuit.	Compatible with GMSK	No	Used for Ranging in a few missions
Telemetry Ranging	Better than conventional performance at sufficient data rates	Lower complexity than PN on downlink due to signal tracking DTTL	Fully compatible with all modulation techniques	Yes on downlink	Lab tested and set to fly soon.
Telecommand/Telemetry Ranging	Better than conventional performance at sufficient data rates	Lower system complexity on both uplink and downlink.	Fully compatible with all modulation techniques	Yes	Proposed

2.3.2. Tradeoff Results

Telemetry ranging and telecommand/telemetry ranging offer similar, if not better, performances than the other methods and have additional advantages that arise from eliminating the ranging signal as discussed in section 2.1.2. As can be seen in table 2.3, telecommand ranging is the best technique as it eliminates the ranging signal entirely and places no requirements on the satellite hardware due to lower complexity on both uplink and downlink. Thus, even without special hardware, a ranging solution can be enabled for small satellites using this technique. This is especially useful outside of Earth orbits and in deep space, where GPS tracking is not an option. This is also useful in Earth orbit as satellites that do not have a ranging solution can theoretically perform a software update to enable telecommand ranging.

Considering these advantages and possibilities, a research gap has been identified in that telemetry ranging techniques (notably telecommand ranging) are yet to be applied to small satellites. Therefore, it is interesting to explore the adaptation of telemetry ranging class of techniques to a small satellite. For this purpose, the Delfi PQ satellite, which is already in Earth orbit and whose developmental and testing equipment are readily available at TU Delft, is targeted as the platform of choice for this research. The goal would be to demonstrate that a viable ranging solution can be enabled on Delfi-PQ with only software changes.

3 Methodology

The conclusions from the literature study revealed a research gap, which was interesting to explore in the master thesis (see section 2.3.2). This chapter discusses the research methodology followed for the thesis. Initially, the research goal, research questions for the thesis which arise from the literature study are defined and the planning for the thesis is elaborated in section 3.1. Following this, the system requirements of the thesis goal are defined in section 3.2. Finally, the tools and software to be used in thesis are explained in detail in section 3.3.

3.1. Research Goal and Planning

From the literature study, telemetry ranging was identified as the state of the art in satellite ranging[3, 29, 75]. A research gap was identified in that telemetry ranging offers several advantages that will be useful to small satellites such as not requiring special hardware or software, allowing use of bandwidth efficient modulation techniques and not requiring power for a dedicated ranging signal (section 2.1.2). However, an implementation of this on a small satellite platform has yet to be explored. Given that TU Delft has its own group of small satellites in space, including the Delfi-PQ small satellite (see section 3.3) and is set to explore satellite miniaturization and distributive space systems with future missions, telemetry ranging could enable a ranging solution for these satellites without requiring additional SWaP.

Therefore, it is interesting to see what requirements an implementation of telemetry ranging would place on these small satellites systems and what the obtained accuracy would be? Thus, the goal of the research is determined to be:

Develop a telemetry based ranging system for use in the Delfi-PQ and future small satellite systems.

The research goal focuses on Delfi-PQ as the primary platform of testing as the hardware, software, and knowledge/documentation is readily available at the university, which will help accelerate development and predict/avoid possible research dead-ends. To achieve this research goal, it is subdivided into various research questions and components of the research are planned to answer these questions. The research questions are as follows:

- **RQ-001: How can telemetry-based ranging methods be implemented using commercial hardware and software?**
  a) **What are the requirements placed on the hardware and software used?**
  b) **What are the main challenges, if any, affecting the viable use of telemetry ranging for Delfi-PQ and other small satellites?**
- **RQ-002: How do the telemetry ranging technique and the obtained ranging results compare to the results in published literature?**
  a) **What are differences, if any, between the research and literature?**
  b) **How do these differences affect the results obtained?**

- **RQ-003: What are the main parameters in Delfi-PQ that affect the performance of the method?**
    a) **By adjusting these parameters, are we capable of reaching a viable ranging solution?**
    b) **Why do the parameters affect the range performance in the Delfi-PQ satellite?**

- RQ-001 arises from the fundamental question of how the method will work. Since the hardware and software available at TU Delft varies from that available at a research lab in NASA or at another company, the differences are noted and identified. Predominantly commercial hardware and software (mostly open source) will be used as they are freely customizable to the limit of the programming language used, giving us a lot more parameters to work with, reducing the probability of the research running into a dead end. This has the added benefit that the research is in theory reproducible and usable to anyone with the same commercially available equipment.
- RQ-002 arises from the need to verify and validate the working of the research solutions developed for RQ-001. That is, how to be sure that the results obtained are valid? For this, the working and performance of telemetry ranging are compared to those in literature. Additionally, range tests using delay lines will be conducted, which will contribute to the validation effort. It is likely that the method used here will need modifications to the one found in the literature, so the effect of the modifications will also be characterized.
- RQ-03 seeks to parameterize the method and identify the main drivers affecting the method performance. Doing so will provide the main variables that can be tweaked so that designers can explore the use of the method for their use cases.

To achieve the goals in a timely manner, a Gantt chart was made for planning the flow of the thesis. It is given in Appendix B and the research was divided in three main subsections as in Figure 3.1:

1. Correlation Algorithm: A method to determine the phase change of a radio signal is needed. This is because determining the phase change of a signal is more accurate than determining the time taken for a signal to travel a distance. Therefore, it is necessary to make a method to determine the phase of a signal. This will form the base of the ranging process, therefore will affect the performance of the system (RQ-003) and the setup of the method(RQ-001).
2. Hardware and software validation using sequential ranging: Ranging places strict requirements on the capabilities of the hardware. A millisecond of time uncertainty from the hardware can cause a range error of 300 Kilometers. Therefore, the fundamental capabilities of the hardware and software to be used apart from the satellite will be probed using a prior existing ranging method. For this purpose, sequential ranging and pseudo noise ranging were options. Sequential ranging was chosen based on the simplicity of operation and of setup required to successfully range (for more information on sequential ranging, see section 5.1.1). Validating and mapping the system performance will contribute towards RQ-003 and RQ-001.
3. Finally, once the base correlation method is established and hardware/software used the ground side are validated, the telemetry ranging technique will be developed for Delfi-PQ. This will enable a complete answer to all three research questions.

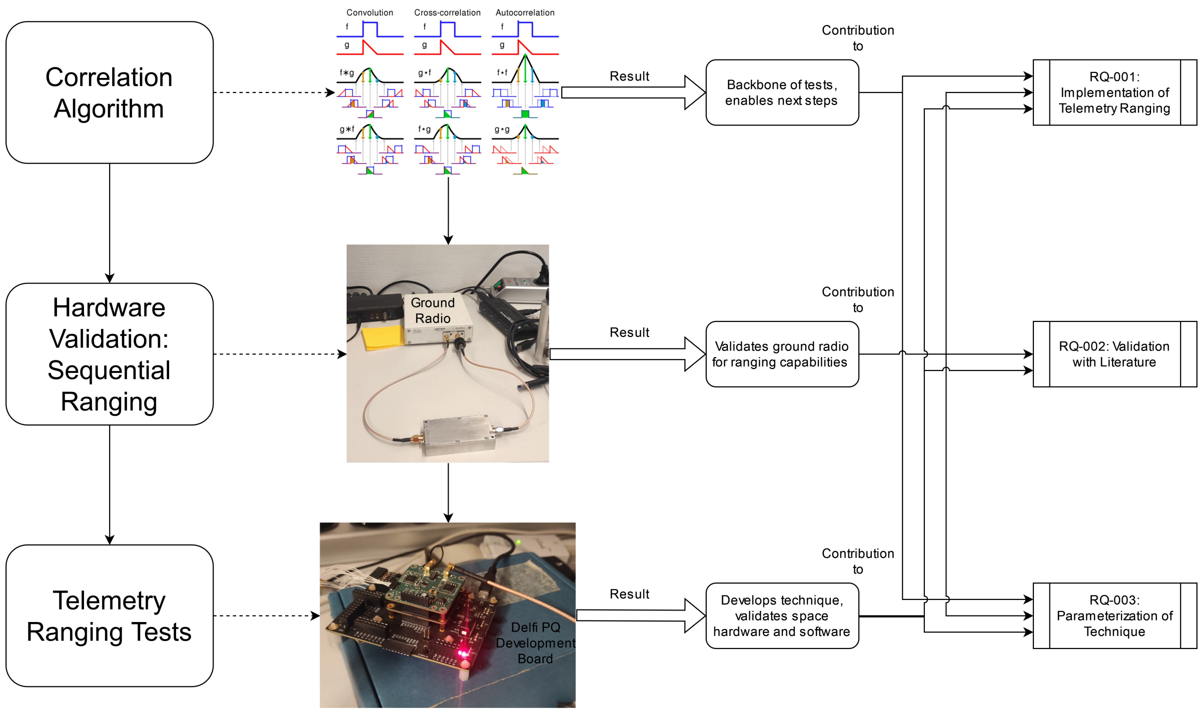

Correlation Algorithm
Hardware Validation: Sequential Ranging
Telemetry Ranging Tests
Convolution
Cross-correlation
Autocorrelation
Ground Radio
Delfi PQ Development Board
Result
Result
Result
Backbone of tests, enables next steps
Validates ground radio for ranging capabilities
Develops technique, validates space hardware and software
Contribution to
Contribution to
Contribution to
RQ-001: Implementation of Telemetry Ranging
RQ-002: Validation with Literature
RQ-003: Parameterization of Technique

Figure 3.1: Thesis steps

3.2. System Requirements

The system requirements which the ranging system will need to meet in order to be useful for the Delfi-PQ satellite are defined here. These arise from the functional requirements of the satellite and stakeholder requirements, where the stakeholder is the supervisor and other developers and users of the Delfi-PQ satellite.

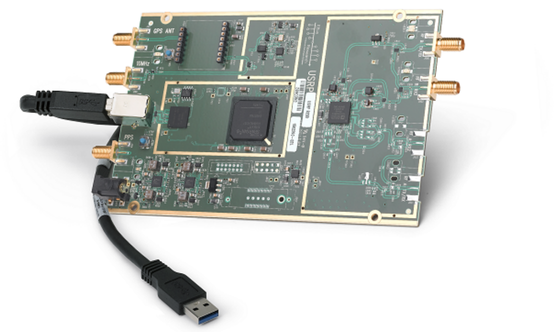
1. SREQ-001:The system shall be capable of achieving better than TLE range performance(>1Km).
2. SREQ-002:The system shall be capable of operating continuously to enable ranging throughout a Delfi-PQ satellite pass ($\simeq$10 mins).
3. SREQ-003:The system shall work with commercially available radio equipment and open source software on the ground.
4. SREQ-004:The system shall not require any hardware modifications to the satellite.
5. SREQ-005:The system shall enable simultaneous telemetry and ranging.
6. SREQ-006:The system shall enable bandwidth efficient modulations such as GMSK and GFSK

3.3. Tools and Software

A variety of tools will be used in the thesis to effectively achieve the goals.

The hardware and software to be used are:

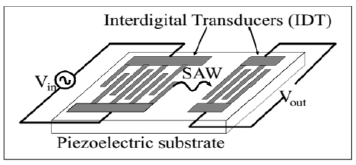
1. USRP - USRP is a software defined radio, which is a radio that implements conventionally analog functions implemented by hardware in the digital domain using software. The advantage of a software-defined radio is that it can implement a large variety of functions depending on the software. A USRP B200 board along with its enclosure is given in Figure 3.2 which provides frequency coverage from 70MHz to 6GHz, contains a direct conversion and digital baseband processor controlled by an FPGA which is connected to a PC using USB3.0. The USRP interfaces with GNU Radio software using Ettus Research's proprietary UHD software, thus enabling rapid development of radio applications with GNU Radio.


(a)

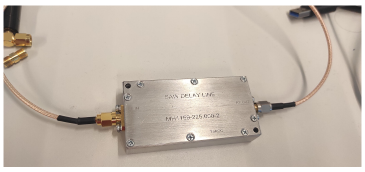
(b)

Figure 3.2: **(a)**USRP B200 Circuit Board[27] and **(a)**USRP B200 enclosure[59]

2. SAW Delay Line – as delay element for testing ranging (Figure 3.3). SAW stands for surface acoustic wave, which are sound waves that travel on the surface of an elastic material. Electrical signals are converted into sound waves using an interdigital transducer. These sound waves have the same amplitude decay properties as electromagnetic waves, but travel a lot slower. These waves are induced onto an elastic material through the use of a piezoelectric substrate. This wave is reconverted into an electromagnetic wave at the end through the use of another interdigital transducer. Thus, this slower speed and decay properties of the acoustic can be used to delay and attenuate an electromagnetic wave a known amount in a delay line[22]. The SAW MH1159-225.000-2 with 13600ns of delay and MH1110-452.600-2 with 19990ns of delay were used in the thesis.



(a)



(b)

Figure 3.3: **(a)**A SAW delay line, **(b)**MH1159 SAW delay line

The Microsaw company MH1159-225.000-2 delay line which provides, 13600ns$\pm$20ns of delay was used due to its ready availability at the TU Delft radio testing labs.

3. Windows 10 – Operating system of PC. Windows was chosen as the operating system due to its easy setup and operation. However, Linux offers a potential performance advantage with the USRP and GNU radio, but is more cumbersome to set up.
4. GNU Radio – Open-source radio ecosystem for digital signal processing. GNU Radio allows for signal processing either for simulations or for software-defined radio systems using the radio as a front end. Most of the work was carried out in GNU Radio versions 3.10.6.0 and with the systems being upgraded to 3.10.7.0 when it came it out.

   Any program set up in GNU radio is called a 'flowgraph'. GNU Radio presents many functions for signal processing in the form of blocks. These can be source blocks, from which data originates, data processing blocks, which do operations on the signals and sink blocks which mark the end of the data flow. GNU Radio blocks and flowgraphs are written using Python or C++ and have a graphical use interface in the GNU Radio Companion, which is based on Qt. Installation can be done in a straightforward manner using the Windows installer present on the GNU Radio website. The advantages and disadvantages of GNU Radio are as follows:

+ Enables quick prototyping based on the large number of inbuilt commonly used signal processing functions such as signal generation, modulation, code tracking loops, etc.
+ Has inbuilt support for USRP and other SDR systems.
+ Based on C++ and has support for Python and a Graphical User Interface (GUI).
- Does not allow for easy programmatic in-run changes of parameters once a program has started. Native support for this is not present, but there are cumbersome ways to work around it. That is, some GUI blocks enable change in variables during a flowgraph's run. It may be possible to recreate this manually and to set this to trigger on a timer. This was not done in the thesis however, as it was more effective to plan around not changing variables mid run. However, this may be an important consideration for other applications.

Why use GNU Radio instead of MATLAB or Python? GNU Radio is completely open source whereas MATLAB is proprietary. GNU Radio is also based on Python wrapped over C++ so the work can be shifted to these languages at any point in the thesis.

5. Python – For additional signal processing and graphing. Python is a popular general purpose programming language which is widely used throughout the world. For numerical and scientific analysis, the Python NumPy and SciPy libraries offer useful premade functions. GNU Radio Python library can be installed using the radioconda repository. This enables manual editing of GNU radio's flowgraphs. This also enables programmatic iteration of variables during runs and repeated runs to explore the variation of results without requiring manual input.
6. Microsoft Excel – For data analysis and graphing. Excel offers tools for quick data analysis, such as determining the standard deviation of a dataset.
7. Delfi PQ: Delfi-PQ is a 3U PocketQube satellite developed by Delfi University of Technology. The aim of the PocketQube is to demonstrate a reliable core bus technology along with testing of various experimental payloads in space[54]. The aim of the satellite is to build up technology to create distributed space systems using small satellites. Delfi PQ is targeted as the platform for this research due to ready availability of the satellite developmental models and documentation/-knowhow at TU Delft Space Systems Engineering department, where this research is conducted.

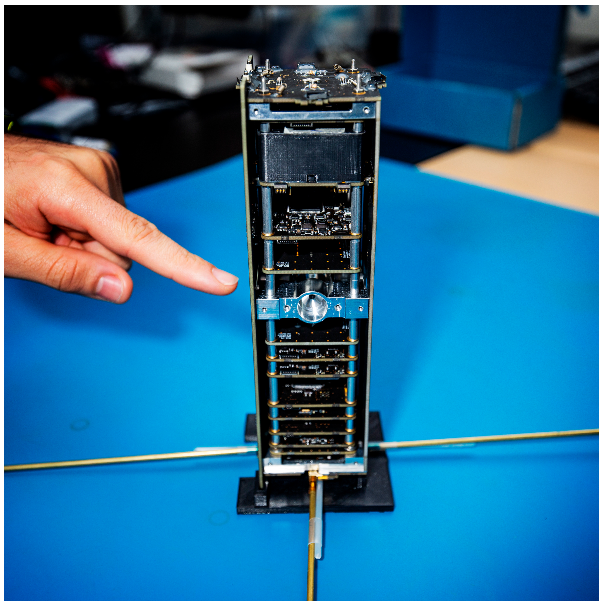


Figure 3.4: Delfi-PQ satellite[78]

An engineering model of Delfi-PQ was built to allow rapid testing of features. This was called the flatsat, and it will be used to develop the telemetry ranging method for Delfi-PQ (Figure 3.5).

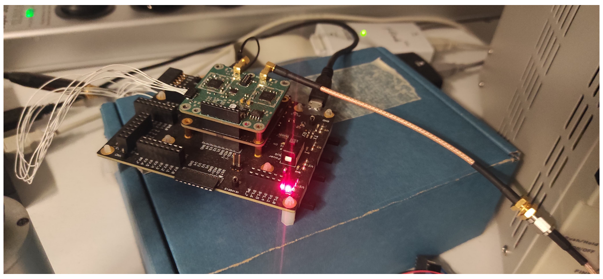


Figure 3.5: Flatsat Developmental Board for Delfi-PQ

4 Correlation Algorithm

A correlation algorithm will form the backbone of the ranging process, enabling the detection of phase differences between the sent and received signals. This chapter starts by defining what correlation is and discusses the different algorithms available for correlation in section 4.1. Section 4.2 discusses how the algorithm can be embedded into GNU radio. Finally, the parameters affecting the performances of the correlation algorithms are discussed in section 4.3, and an algorithm is chosen for further use.

4.1. Correlation Algorithm

What is correlation? Correlation is the process of multiplying one signal with another to check for their level of alignment. Let's define two signals as data sets f and g containing discrete values:

$$f = [f_1, f_2, f_3, ..., f_n] \tag{4.1}$$

and

$$g = [g_1, g_2, g_3, ..., g_m] \tag{4.2}$$

Digital representations of electromagnetic waves are of the form of f and g, where the elements of the array are called samples and represent points on the electromagnetic wave. With a sufficient number of these points, or samples. Figure 4.1 shows points or samples on a signal being recorded. Using these samples and straight lines joining thesis samples, a representation of the signal can be recreated. For proper signal processing, the number of signal samples should be greater than twice the maximum frequency being processed. This is known as the Nyquist criterion. However, the greater the number of samples, the better the signal representation and more accurate the correlation process would be in determining the phase delay

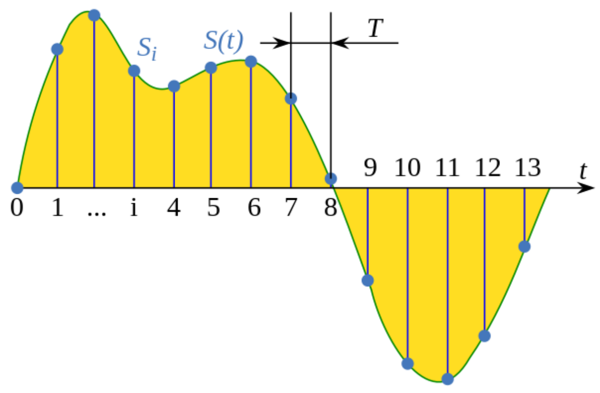


Figure 4.1: A radio signal being sampled at different points[84]

The correlation of f and g, also given by f*g in Figure 4.2 is defined as

$$CorrelationValue = f * g' = [f_1, f_2, f_3, ..., f_n] * [g_1, g_2, g_3, ..., g_m]' \quad (4.3)$$

where g' is the transpose of array g.

The graph for correlation shown in Figure 4.2 is obtained by sliding one array over the other by shifting the elements of that array and obtaining the correlation value at each point. This shifting by one element is given mathematically as:

$$f_{new} = f_{delayedby1element} = [0, f_1, f_2, ..., f_{n-1}, f_n] \quad (4.4)$$

This operation is done for all possible delay values to obtain a correlation value matrix which when

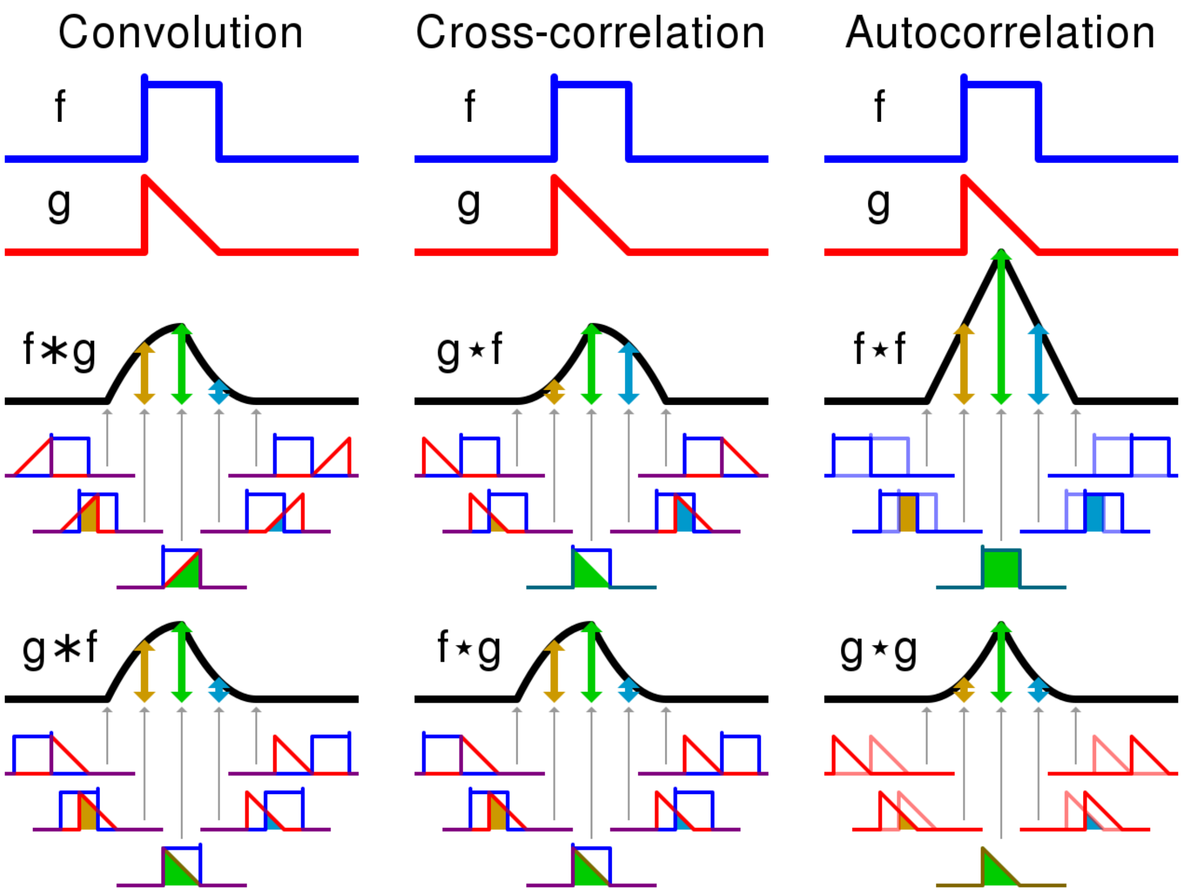


Figure 4.2: Auto Correlation and Cross Correlation[83]

plotted gives a plot similar to Figure 4.3. In this, a signal f is correlated with itself. The maximum value of the correlation is visible at the edges, which indicates that the signal is not shifted. When the correlation of the signal is with a copy of itself, this is also known as autocorrelation. Therefore, by checking for a signal autocorrelation by sliding it over an array of received signals, its presence and location within a group of received signals can be detected.

However, the sliding correlation can be a slow process, as mathematically its computation cost is O($N * M * N$) when using direct multiplication algorithms where N and M are the length of the two signals (say f and g respectively) to be correlated. This considers the cost of the multiplication of the signals of size N M times, giving O($N * M$) as complexity (considering operation f*g is being done). This is followed by addition of the signals, adding another order of N. The computational efficiency can be better when using more efficient algorithms, as seen in Table 4.1.

The correlation process, though acceptable for smaller signal lengths, can take minutes to find the correlation peak for signal lengths of 512,000 samples. Radio signals in the test were sampled at Digital to Analog Converter (DAC) sample rates greater than 250,000 samples per second, with the USRP B200 hardware to be used not supporting rates less than 200,000 samples per second. This quickly adds up, and typically the requirement is to search for the presence of the ranging signal in received signal recordings of up to many minutes, corresponding to system requirement 2 (see section 3.2). The normal correlation algorithm as defined above will take a long time to find the correlation matrix. Therefore, a faster algorithm is necessary at this point to meet the requirement. The total computational complexity for the normal (non-FFT) correlation is given as

$$O(N * M * N) \tag{4.5}$$

By FFTing the signals into the Fourier domain, multiplying them with each other and inverse FFTing the signals to obtain the correlation matrix, the properties of the Fourier transform can be exploited to get a faster correlation.

A Fourier transform transforms the signal from a function of time to a function of frequency. This means that the recorded signal components will be gathered according to their frequency, irrespective of the time at which they were recorded. A multiplication of two signals in the Fourier domain is equal to their convolution in the time domain[63], therefore, the FFTed signals with one of them conjugated can be multiplied with each other in the Fourier domain to calculate the correlation. Then an inverse FFT can be carried out to regain the time domain representation. As the computational complexity of anFFT is only O(N$*$log(N)), which is way cheaper than O(N^2), doing so improves the speed of the correlation, and hence this is called high-speed convolution or fast convolution. For FFT, apart from the signals having to be the same length, the computation is faster and avoids cyclic convolution artifacts if the signal lengths are a power of 2. Therefore, the signals will be padded to the closest higher power of 2 followed by a fast correlation. The overall complexity is given as:

$$O(2 * Nlog(N) + 2 * P - N) \tag{4.6}$$

where P is the closest power of 2 higher than N. As can be seen from Table 4.1, this is much lower than the complexity for normal correlation. For further reading on computational complexity, see a well-cited article in [82]. Therefore, the fast correlation algorithm has a big time advantage over the normal correlation algorithm for correlation sizes larger than 50000, which were tested for and will be used in ranging.

For example, Figure 4.3 represents the auto-correlation of a signal of length 84000, or the correlation with itself. Since this signal is aligned with itself, the peak reaches a maximum at 0 and 84000 samples when the signal repeats itself. The correlation algorithm is capable of detecting the delay in

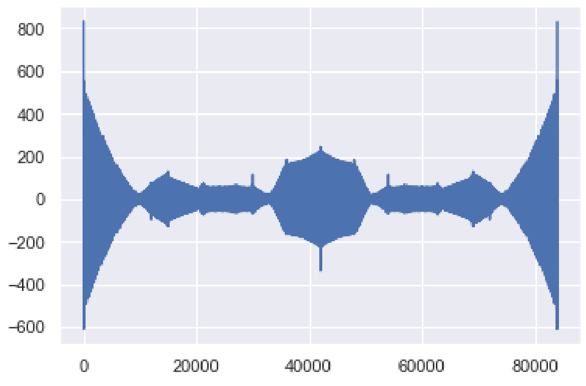


Figure 4.3: FFT-correlation of signal with length 84000 samples

the signal up to the limiting sample rate. Data interpolation can be used to increase the amount of samples available improving the resolution.

The addition of noise introduces an error in the correlation algorithm due to spreading in the correlation peaks. This has the effect of reducing correlation values at some places and increasing them at other. This change is visible in Figure 4.4 where the correlation peaks are not as well-defined in the presence of noise.

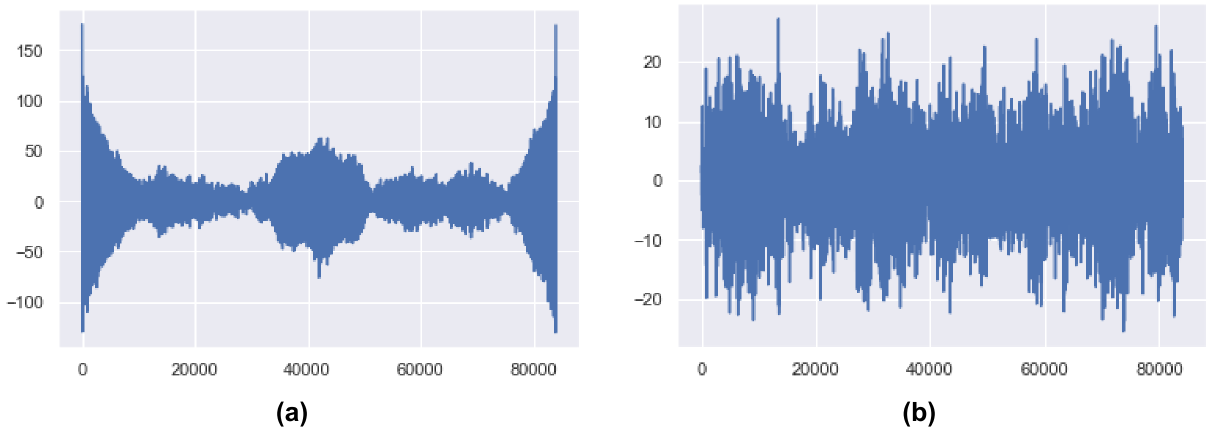


Figure 4.4: Fast Convolution for signal of length 84000 with **(a)**some noise and **(b)**overwhelming noise

How would this correlation translate into ranging measurements? GNU Radio will be used to time tag (associate each signal with a sent and received time based on Coordinated Universal Time (UTC)) and store the sent signal samples and received signal samples into a computer. By correlating these stored samples, the difference between sent and received phase and hence time can be determined.

However, when using repeating signals, such as ranging signals, the correlation algorithm is susceptible to cyclic convolution, also called ambiguity. Cyclic convolution occurs when the periodic signals are correlated with each other in normal correlation or when fast correlation is done with signals of equal length. The correlation process causes an overlap from the repeating signal affecting the correlation results. In Figure 4.5, the 5th graph represents the correlation of sequence h[n] (graph 1) with repeating sequence $X_N[n]$ (graph 4). Here, the overlap from the repeating correlations is visible between the red and blue lines. They are added together and give a larger correlation value in graph 6. These are the cyclic convolution artifacts and to avoid them, the signals must be padded to at least twice their length minus-1.

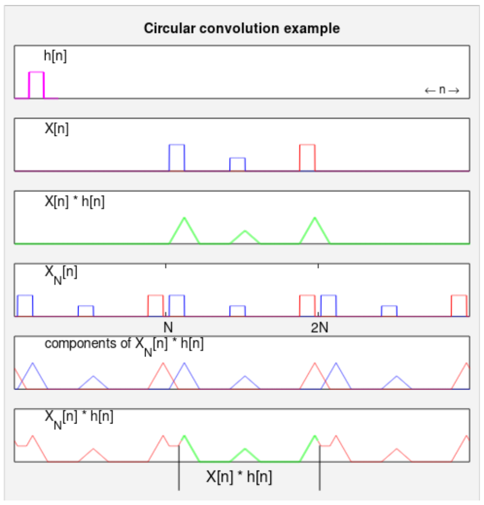


Figure 4.5: Effect of cyclic convolution[6]

The effect of padding on fast correlation can again be seen in Figure 4.6, where there are three different correlations of the same signal with no padding and different amounts of padding. When the signals are the same length, the correlation acts as if the signals wrap around themselves when delayed by one sample. For example, considering element f, f_{new} is given as

$$f_{new} = f_{delayedby1element} = [f_n, f_1, f_2, ..., f_{n-1}] \tag{4.7}$$

This has the effect of the correlation values having an additional contribution from the wrapped around version of the arrays which should not be there, which is seen in (a) in the figure. When the array is padded with zeros before correlation, these contributions are separated enough that they do not affect the correlation peaks as seen in (b) and (c).

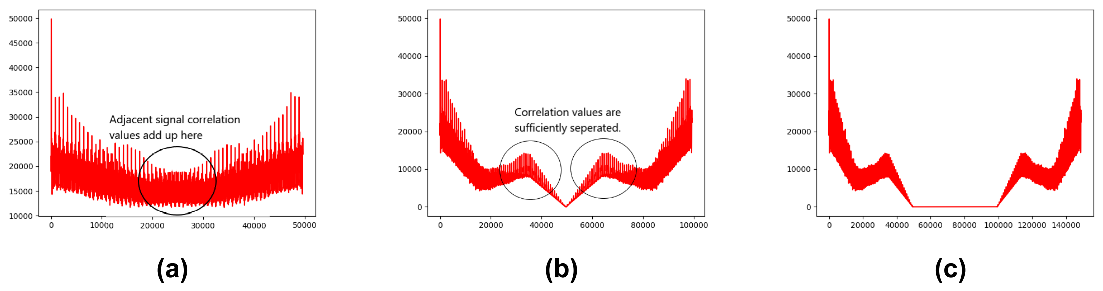


Figure 4.6: Fast Correlation for signal of length 49600 with **(a)**no padding noise, **(b)**minimum required padding of size of signal and **(c)**extra padding than needed

4.2. Working in GNU Radio

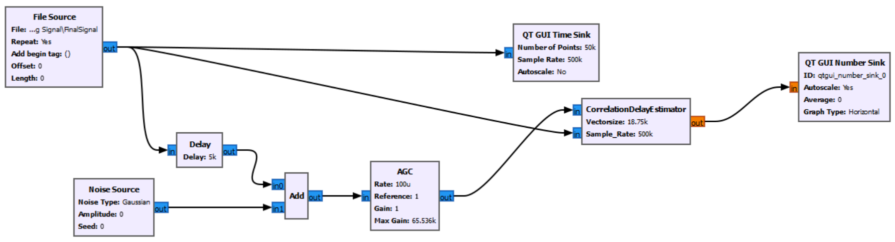
For use in GNU Radio, the Python language based implementation of the algorithm is given as follows, the complete code is given in the appendix Listing A.1:

```
correlation_values =np.fft.ifft(np.fft.fft(signal1) * np.conj(np.fft.fft(signal2)))
sample_delay_count = np.argmax(np.abs(correlation_values))
delay=self.sample_delay_count
```

where signal1 and signal2 are the (padded) signals whose phase difference to identify. Line 1 converts the signals from the time domain into the Fourier domain. Due to this system being run digitally, the maximum resolution of this correlation process is limited by the sample rate allowed by the system.

GNU Radio itself offers a correlation block, however, this is set to use the direct correlation algorithm. A fast correlation block is provided, but it does not provide any mechanism to transfer the correlation information to other blocks for further processing. Therefore, it makes sense to write the fast correlation algorithm into a GNU Radio embedded Python block.

The embedded Python block takes in defined variables as input and outputs and provides a Python function where operations can be performed upon the data. The block was configured to take in two signals of complex datatype and output the range as a floating point variable in Figure 4.7


File Source
File: ...g Signal\FinalSignal
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

QT GUI Time Sink
Number of Points: 50k
Sample Rate: 500k
Autoscale: No

QT GUI Number Sink
ID: qtgui_number_sink_0
Autoscale: Yes
Average: 0
Graph Type: Horizontal

CorrelationDelayEstimator
Vectorsize: 18.75k
Sample_Rate: 500k

Delay
Delay: 5k

Noise Source
Noise Type: Gaussian
Amplitude: 0
Seed: 0

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 65.536k

Figure 4.7: Testing of custom correlator in GNU radio

The custom correlator block was configured to decimate the output, meaning the sample rate was reducing on the output side of the block to a specified amount. This means that the block will output (for a decimation of N) 1 sample for every N samples input. The decimation code is not fully supported in GNU Radio Python based script, and the documentation only holds examples for the C++ version. However, the code which worked for the Python version was derived through trial and error and is given in the appendix in Listing A.1. Notably, the inheriting block was changed to specify that the block is a decimating block, a decimation factor was specified and the return function for the block was modified to ensure only the decimated number of samples were returned.

```
1.class CorrelationDelayEstimator (gr.decim_block):
2.decim=vectorsize
3.return num_samples // self.vectorsize
```

Making a custom block has a few advantages in that it makes it easy to integrate the correlation algorithm into GNU radio flow graphs and hence into the satellite operational workflow. It also enables real-time ranging as long as the computer is capable of handling the Central Processing Unit (CPU) and memory load. However, GNU Radio comes with a few caveats:

1. Requires a certain amount of oversampling to work properly: Oversampling means to use a higher sampling rate than the Nyquist rate required by a signal. Without an oversampling of a rate 5 times higher than the Nyquist rate, GNU Radio signals run into errors. At this point, the correlation values obtained were just wrong.
2. The correlation length is restricted for use in the flowgraph. Decimation support for GNU radio's Python block is not complete and runs into errors at higher correlation lengths, around 2million samples. Therefore, beyond this point, it is necessary to shift the correlation algorithm to a Python script.

4.3. Performance

The performance of the correlation is vital as it will be directly in the pipeline of the ranging process, so any errors will add to the overall range error. Primarily, three parameters are of interest to us:

Correlation Resolution

The resolution will determine the maximum phase change that can be detected, and thus the maximum ranging accuracy. This resolution is tied to the sample rate at which the signal is processed, as the correlation algorithms detects any changes down to the sample.

Integration Factor

Here, K is defined as the integration factor for a signal and corresponds to the number of copies of a signal sent, received and correlated together. Therefore, a signal of length 51,200 samples with k=10 integrates or gathers 10 copies of the signals before correlation, giving a total correlation length of 512,000. This has the effect of increasing the height and sharpness of the correlation peaks, but not of the noise because noise picked up by the different copies of the signal are uncorrelated. Thus, the correlation errors reduce with increasing integration factor (k) which is reflected in Figure 4.9.

Figure 4.8 shows the effects of increased integration time on the correlation graph for a signal of unit length 49,600, padded to avoid cyclic convolution. Here is also a good place to explain why the specific shape is obtained in the graph. In (a), as the signal rolls away from perfect alignment with itself, the correlation values reduce until the minimum correlation value is seen at the center. Then the signal starts to wrap around itself and starts aligning with itself again. Imagine a game of musical chairs where after a sufficient amount of rotation (delay), the people (the array values) start reaching the initial position. In the figure, the biggest peak seen is considered as the exact point of alignment or the delay value. This peak for each graph is seen near the start, as the delay is set to 50 samples for this test.

In graphs b and c where the integration times are higher, smaller but distinct peaks are also seen apart from the biggest peak. Since increasing integration factor (and hence integration time) is nothing but increasing the number of signals correlated. the smaller peaks are the caused by some of the initial signals perfectly aligning with some of the later signals. The rest of the signals are outside and do not contribute to the correlation. As visible, the integration factor increases the correlation peak height by the number of units integrated (o(n)).

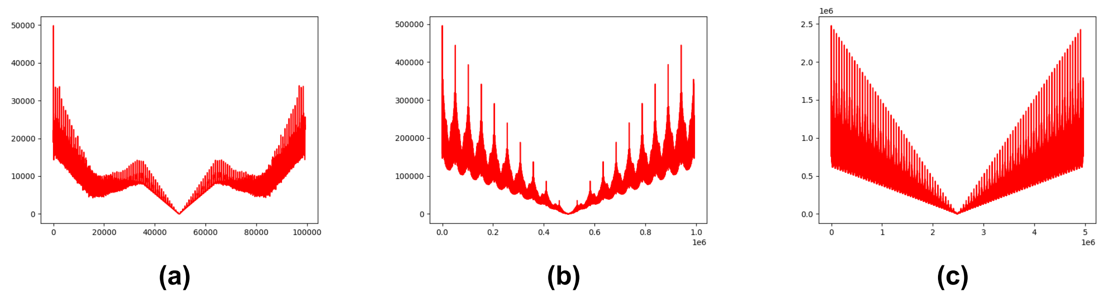


Figure 4.8: Correlation Matrix of signal with different Integration Factors of **(a)**k=1, **(b)**k=10 and **(c)**k=100

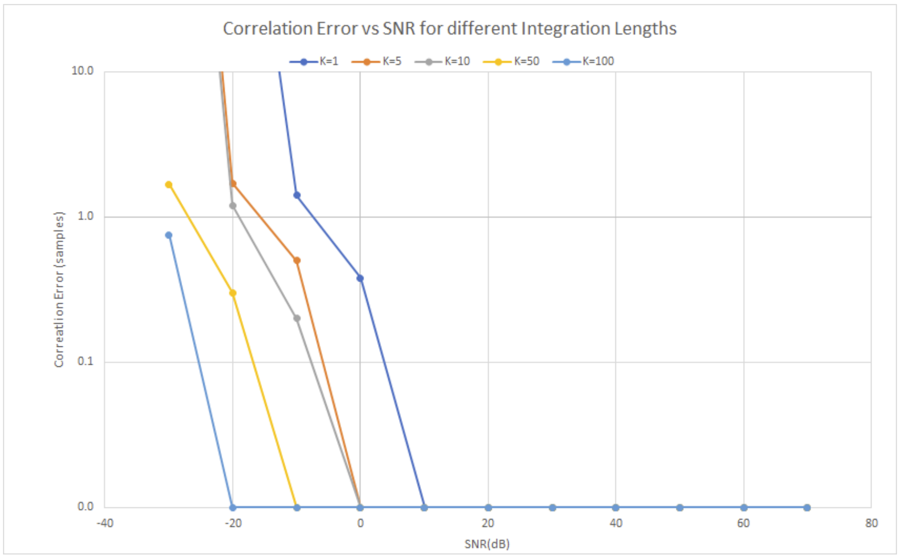


Figure 4.9: Correlation Errors vs Integration Factor

Time taken for Correlation

The time taken to correlate signals of varying lengths is a good piece of information to plan tests around. The fast convolution algorithm is relatively faster and simple to use. The correlation time is compared for

1. NumPy convolution- The numerical python or NumPy library is a collection of functions offering high level array and matrix based math operation support for Python. Normal correlation is also one of the mathematical function offered.
2. SciPy fast convolution- Scientific Python or SciPy is a library offered for scientific and technical computing which offers a FFT based correlation option.
3. Manual fast correlation operation- A manual implementation of fast correlation was made based on NumPy functions to obtain a better understanding of the correlation process. The advantage here is that the process and output can be modified for the application intended.

The results from the comparison are shown inTable 4.1. It can also be seen here that correlating a longer sequence takes more time than correlating a number of shorter sequences. It is clear that fast correlation is much faster than normal correlation and therefore will be chosen for use in the thesis.

Table 4.1: Correlation Time of different algorithms (Signal Length =51200 samples)

Algorithm / Integration Factor	NumPy Convolution	SciPy Fast Correlation	Manual Fast Correlation using Numpy
1	0.3119s	0.0059s	0.0069s
5	0.03488	0.0348s	0.0369s
10	67.3870s	0.0756s	0.0687s
50	>>100s	0.4146s	0.6359s
100	>>100s	0.8432s	1.2608s

The FFT correlation algorithms are faster than normal correlation. Therefore, their performances were compared as in Figure 4.10. The SciPy implementation gives better performances than the manual implementation in both speed and accuracy at lower SNRs. However, at higher SNR values, both give identical performances and the SNR threshold at which both methods start to make mistakes are at -10dB (k=1) and -20dB (k=10) respectively.

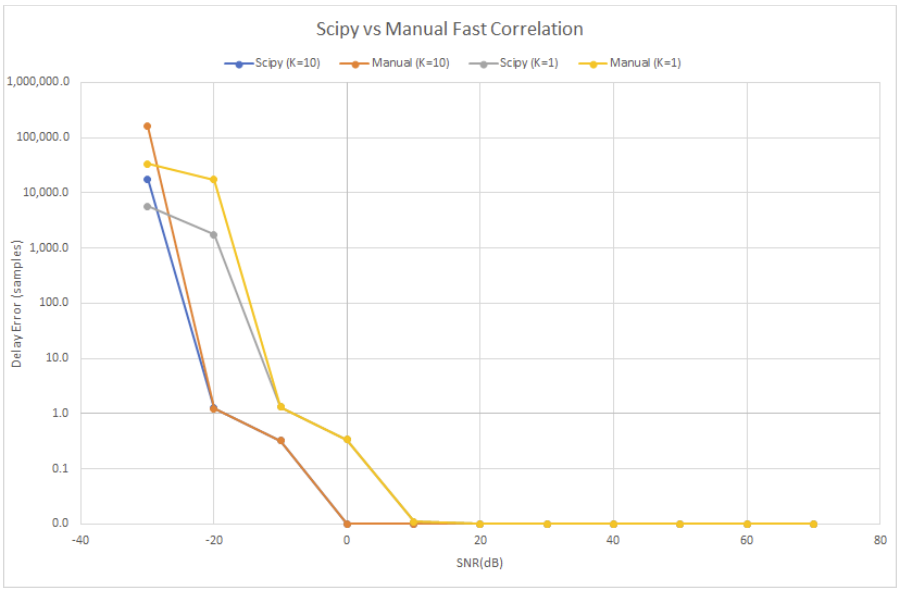


Figure 4.10: Scipy FFT correlation vs. Manual FFT correlation

Among the SciPy and manual fast correlation algorithms, though SciPy is faster and slightly more accurate, the manual algorithm allows for more control of the obtained results and better understanding of the underlying processes and does not compromise any meaningful accuracy. That is, in Figure 4.10 the error existing in the low SNR regions make no physical difference between SciPy and Manual Implementations, and in the medium and high SNR regions (>0dB) both methods are equally accurate. Therefore, the manual correlation method was used in the thesis.

Limiting Length of Correlation

How big a sequence can be correlated is dependent on the specifications of the system, notably the RAM. With a 8 GB ram system, this turned out to be around 50 Million Samples or 400 million bytes (1 sample of complex64 variable used to store samples is 8 bytes, so 4 GB). This was the limit where the system was reasonably responsive. This length can be easily increased by adding more Random Access Memory (RAM) or offloading the data storage to page memory on the hard disk, which is of course much slower.

4.4. Conclusion

In this chapter, the correlation method was defined in section 4.1. The characteristics of correlation such as normal correlation, fast correlation, difference between convolution/correlation and cyclic convolution errors were described here.

Following this, the method of implementing the algorithm into GNU Radio was discussed, along with its advantages and challenges in section 4.2.

Finally, the factors affecting the performance and the obtained results were discussed in section 4.3, resulting in the choice of the manual correlation method for use in the thesis. The next step would be to validate the hardware and software to be used for ranging using a well-known ranging method.

5 Sequential Ranging

This chapter discusses the validation of hardware and software to be used for ranging through the use of a well-known ranging method-sequential ranging. Firstly, the sequential ranging method is introduced.

Following this, the details of the working of the method such as signal generation, signal timing and setup in GNU Radio are discussed in section 5.1.

Further, the performance parameters of the method are discussed and ideal parameters chosen for a ranging test in section 5.2.

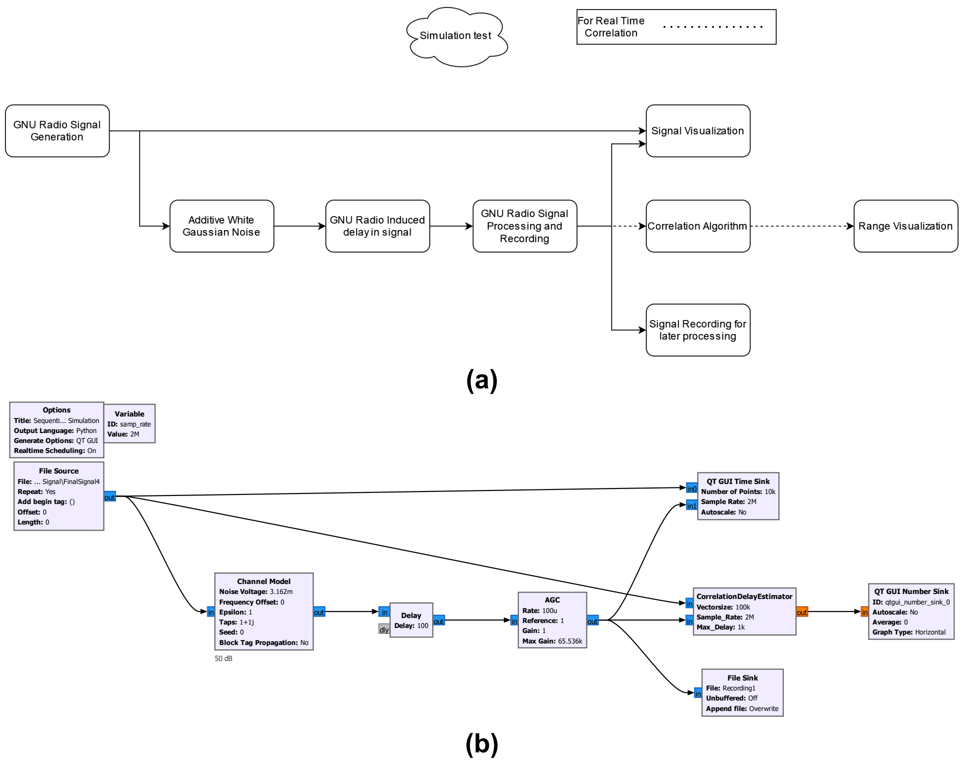
Finally, the results of simulation and USRP range test are discussed in section 5.3.

Sequential ranging or tone ranging is a well-known method that has been in use for decades. The method consists on sending a series of sinusoidal radio signals of consecutively lower frequencies to the satellite. The satellite receives these tones and re-generates a copy having the same phase through correlation, and sends this copy back to the ground station. The specific structure of the sequential ranging wave allows for a better correlation than a random signal[9]. The highest frequency signal is known as the clock signal and is used to obtain the highest possible accuracy for the range measurement. Consequently, lower frequency signals are sent to resolve the range ambiguity, having a larger wavelength and hence have a larger distance they can travel before their phase start to repeat. The lowest frequency signal is chosen to have a wavelength greater than the prior known range of the spacecraft.

If the spacecraft is known to be around 400 km away, then the lowest frequency signal wavelength is chosen as greater than 400 km, say 450 km or 500 km. The exact frequency does not matter as long as the wavelength is greater and is chosen based on what frequency the ranging system is most equipped to handle. In deep space scenarios, the resolving the ambiguity simply based on frequency would be impractical due to the very low frequencies that would be needed, thus higher frequency signals are used with prior knowledge of the spacecraft's position being used to resolve ambiguity.

To verify the capabilities of the hardware and software (see section 3.3), it was decided to use the USRP and delay line test setup with a well-known method such as sequential ranging. In order for sequential ranging to be conducted, it is necessary to suitably generate a ranging signal, time the setup appropriately and correlate to obtain the range.

The first test will be a simulation based test for the software to verify the correlation method, explore any limitations from GNU Radio, and identify the best setup of the sequential ranging system. The setup of the GNU Radio based simulation is given in Figure 5.1. In a GNU Radio flowgraph, there always exist two fundamental blocks, options and samp_rate which stands for the sample rate at which the system will be run. The options block has the configuration settings including whether to use C++ or Python as the programming language, choosing the type of GUI and more. Initially, Additive White Gaussian Noise (AWGN) is added to the recorded sequential ranging signal to simulate the addition of noise into the signal. Then this signal is delayed by a fixed number of samples (100 in Figure 5.1) using the delay block and then either (a) stored for correlation in non-real time or (b) sent to the correlation block for real-time correlation if system performance allows.


Simulation test
For Real Time Correlation
GNU Radio Signal Generation
Signal Visualization
Additive White Gaussian Noise
GNU Radio Induced delay in signal
GNU Radio Signal Processing and Recording
Correlation Algorithm
Range Visualization
Signal Recording for later processing
(a)
out
in
dly
in0
in1
50 dB
(b)

Options
Title: Sequenti... Simulation
Output Language: Python
Generate Options: QT GUI
Realtime Scheduling: On

Variable
ID: samp_rate
Value: 2M

File Source
File: ... Signal\FinalSignal4
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

Channel Model
Noise Voltage: 3.162m
Frequency Offset: 0
Epsilon: 1
Taps: 1+1j
Seed: 0
Block Tag Propagation: No

Delay
Delay: 100

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 65.536k

QT GUI Time Sink
Number of Points: 10k
Sample Rate: 2M
Autoscale: No

CorrelationDelayEstimator
Vectorsize: 100k
Sample_Rate: 2M
Max_Delay: 1k

QT GUI Number Sink
ID: qtgui_number_sink_0
Autoscale: No
Average: 0
Graph Type: Horizontal

File Sink
File: Recording1
Unbuffered: Off
Append file: Overwrite

Figure 5.1: **(a)** Simplified flowgraph and **(b)** GNU Radio flowgraph for sequential ranging simulations.

Following the simulation test, a real world delay line test will be conducted to validate the performances of the radio on the sequential ranging system. The delay line induces a known delay of 4.08 km into the signal along with 30 dB attenuation. Using this known delay, the performance of the correlation method can be estimated. Through changing the gain of the transmitter and receiver before transmission, an SNR analysis of the system can be performed.

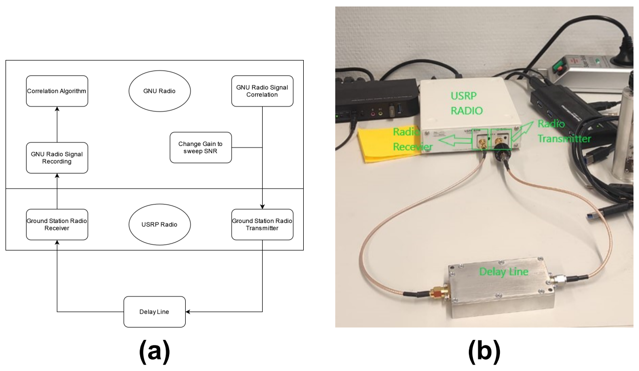


Figure 5.2: Radio ranging setup **(a)** flowgraph and **(b)** picture

5.1. Working in GNU Radio

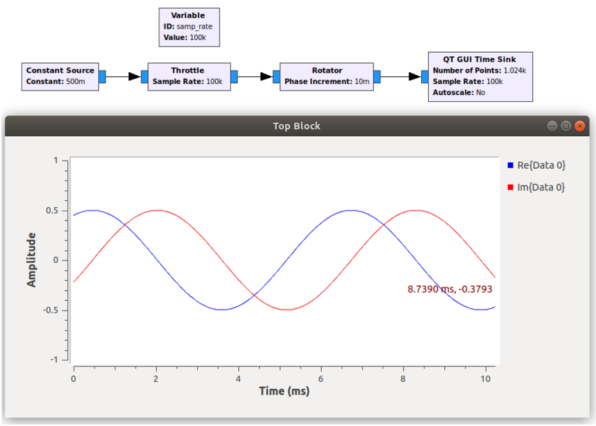
The ranging signal are sinusoidal signals that need to be generated according to the needs of the system. The ranging signal used by NASA has a range clock or highest frequency signal with a frequency coherent with (a proportion of) the carrier frequency[17]. This coherence is established for a reason. In satellite sequential ranging, the same signal is handled by the two systems (ground and satellite) each with their own clock which runs at slightly different speeds. This difference in clock speed will change the generated frequencies slightly and this mismatch will cause a reduction in the correlation peak height, which will increase the thermal noise in the system according to Equation 5.2. This coherence is not required in this test because only a single radio is used to generate the signal, so there will be no frequency offset.

One of the factors affecting the choice of highest frequency signal is the highest frequency achievable by the USRP system. From a sample rate analysis in subsection 5.2.3, this was determined to be 2MHz, which places the max frequency signal at 1MHz based on the Nyquist frequency criterion. Apart from the highest frequency signal or clock signal, the subsequent lower frequency signals have their frequencies as half of the signal preceding them. The NASA tone ranging system has 14 tones starting from 1MHz down to $1/2^{13} = 122Hz$. This gives the lowest frequency signal a wavelength of $c/125 = 2,400,000m$, which is the maximum range that can be determined unambiguously with this system. However, such an ambiguity resolution is not necessary for this delay line test. The delay line length of 4.8km requires an ambiguity resolution a bit higher, say 5 km arbitrarily, giving a tone frequency of $c/5000 = 60000Hz$. Therefore, an ambiguity resolution tone of frequency, 60000 Hz or lower is needed to unambiguously determine the delay line delay.

5.1.1. Ranging Signal Generation

Generating a sinusoidal signal can be done using the rotator block as in Figure 5.3. A constant source block is used to generate a constant signal of the given amplitude, which is 0.5 or 500m in the figure. This signal has a frequency of 0. A throttle block is used to ensure the system does not run the graph faster than it needs to, which saves on cpu resources. A rotator block has the effect of shifting the signal frequency by a given amount, which results in the formation of a sinusoidal signal. In the figure, the rotator block phase increment is set to 0.01 radians. Since the sample rate is 100kHz, that gives 100 KHz$*0.01$ =1000 radians every second, or 1000/(2pi) = 159 Hz frequency, corresponding to a wavelength of 1,886,792m (6 ms in time), which is seen in the graph. This gives the argument given to the block as:

$$Phase\_Increment = \frac{Desired\_freq \times 2\pi}{samp\_rate} \tag{5.1}$$


Variable
ID: samp_rate
Value: 100k

Constant Source
Constant: 500m

Throttle
Sample Rate: 100k

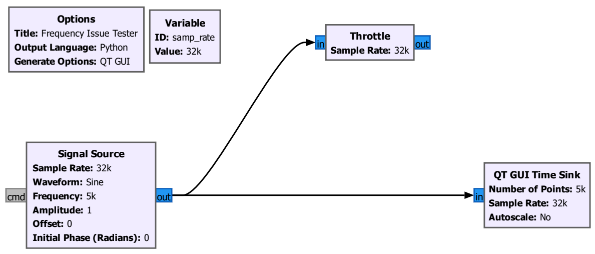
Rotator
Phase Increment: 10m

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 100k
Autoscale: No

Figure 5.3: GNU Radio Sequential Signal Generation[48]

A signal source block exists in GNU radio which encapsulates this function and generates the signal based on the given information such as frequency. The rotator and the signal source block were tested to provide the same output. For this thesis, the rotator block was used as the underlying processes were better understood. However, using both rotator or signal source blocks, GNU Radio runs into an issue which causes a signal of inaccurate frequency when the signal to be generate is of a frequency lower than 5000 Hz.

To check this, a signal visualization block was set to hold exactly the number of points that correspond to one wavelength for the signal and was set to update every second in Figure 5.4. When the frequency was higher than 5000 Hz, this signal did not shift in the figure, which showed that the signal was repeating exactly every second. In the lower frequencies, the signal showed a small shift every second, which meant that the signal frequency was slightly wrong. Therefore, GNU Radio places a restriction of 5000 Hz on the minimum frequency of a signal when the frequency of the signal needs to be accurate. The cause of this issue is unknown as an exploration was not required as this restriction is not an issue for the use case in the master thesis.



Options
Title: Frequency Issue Tester
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

Throttle
Sample Rate: 32k

Signal Source
Sample Rate: 32k
Waveform: Sine
Frequency: 5k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

QT GUI Time Sink
Number of Points: 5k
Sample Rate: 32k
Autoscale: No

Figure 5.4: Frequency Issue tester graph

The rotator block was used to generate the lowest frequency signal to be used. Since the requirement for ambiguity resolution of the delay was determined to have a frequency lower than 60000 Hz, the lowest frequency signal was generated as 12500 Hz as a starting point. Now, the higher frequency signals cannot be generated using a rotator block, this is because, even though the source may be the constant source, the different rotator blocks all have a different effect on the signal and cause the frequency and phase generated to be slightly different. This results in a phase-discontinuous wave. If the signal tones are phase discontinuous, they will have frequency components that push the radio components into non-linear operation, which will affect the accuracy of ranging. In the real world tests, this was reflected in the method giving wrong range values when the tones were discontinuous. The difference between a phase continuous and discontinuous signal can be seen in Figure 5.5.

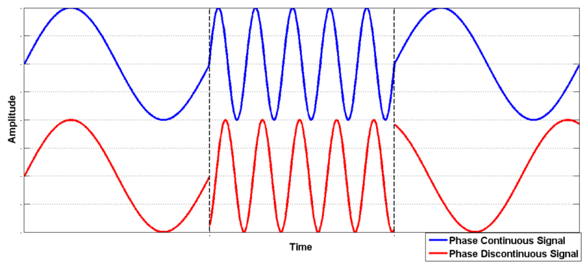
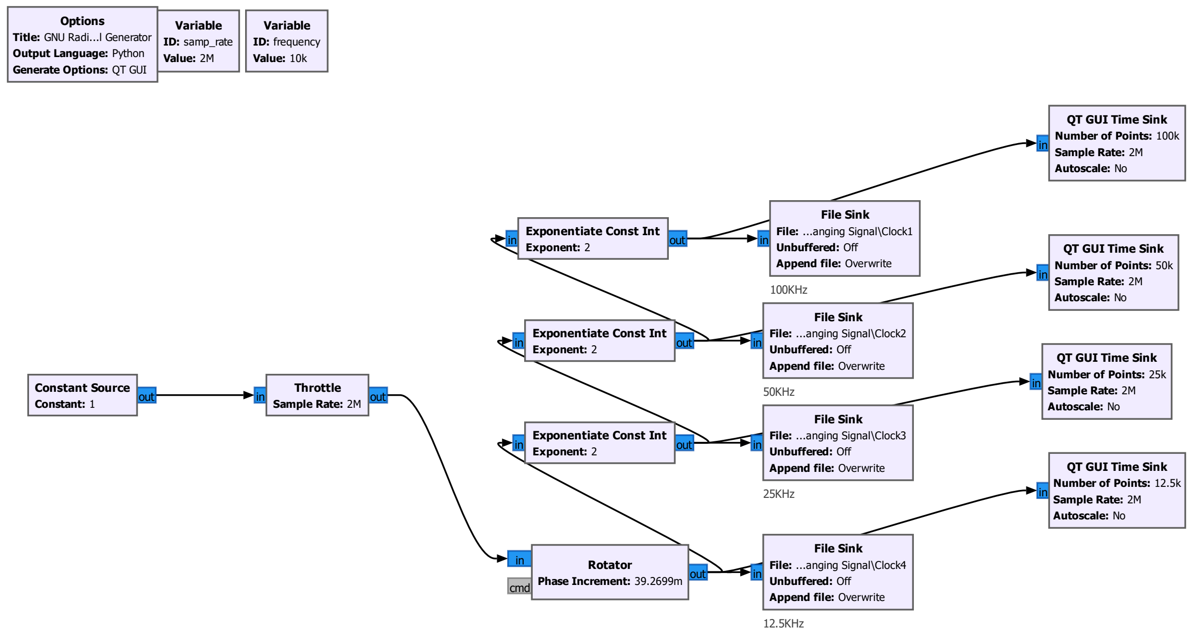


Figure 5.5: Phase continuous vs. non-continuous Signals [51]

To ensure that the signals are phase-continuous, as shown in Figure 5.6, the lowest frequency signal is generated from a single constant source and rotator block and mathematically operated on so that the rest of the waves are generated from this low frequency wave. Raising a signal to the power of 2 doubles its frequency, which can be done using the exponential const int block in GNU Radio[24]. This ensures that the signals are generated from one another and hence are phase-continuous in GNU Radio. To verify there are of the correct frequency, the signals were checked visually using a visualization block set to show exactly one wavelength worth of points for each signal every second. If this block did not show any change in the signal every second, this meant that the signal repeated perfectly according to the specified frequency and are correct. This method of generation may be the practical reason behind why the sequential ranging tones are multiples of one another.


Options
Title: GNU Radi...l Generator
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: frequency
Value: 10k

Constant Source
Constant: 1

Throttle
Sample Rate: 2M

Rotator
Phase Increment: 39.2699m

Exponentiate Const Int
Exponent: 2

Exponentiate Const Int
Exponent: 2

Exponentiate Const Int
Exponent: 2

File Sink
File: ...anging Signal\Clock1
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...anging Signal\Clock2
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...anging Signal\Clock3
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...anging Signal\Clock4
Unbuffered: Off
Append file: Overwrite

QT GUI Time Sink
Number of Points: 100k
Sample Rate: 2M
Autoscale: No

QT GUI Time Sink
Number of Points: 50k
Sample Rate: 2M
Autoscale: No

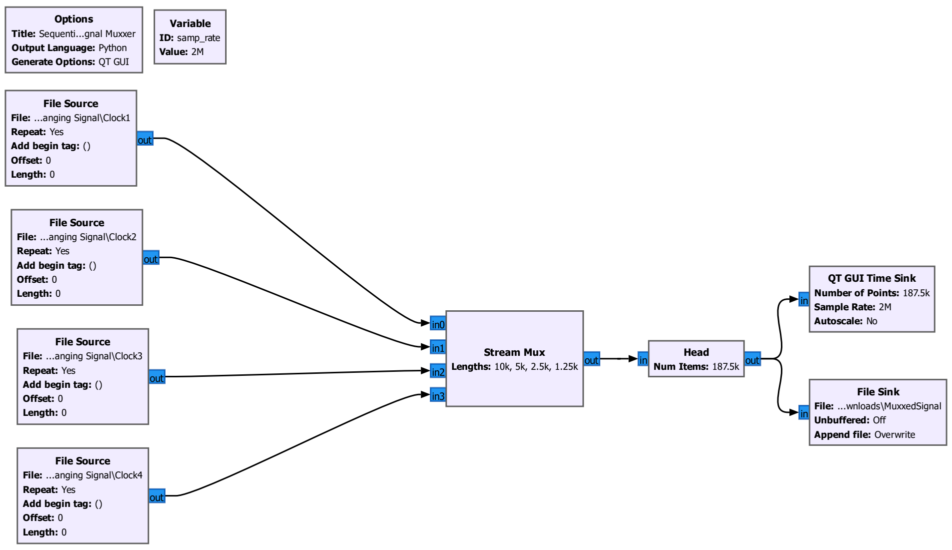
QT GUI Time Sink
Number of Points: 25k
Sample Rate: 2M
Autoscale: No

QT GUI Time Sink
Number of Points: 12.5k
Sample Rate: 2M
Autoscale: No

Figure 5.6: Generator flowgraph for sequential ranging signal

As discussed in this section, the maximum possible frequency that could be used for sequential ranging is 1MHz due to the Nyquist frequency criterion and the USRP-Windows system sample rate limit. The lowest signal frequency needs to be lower than 60kHz for ambiguity and higher than 5kHz to be accurately generated. Additionally, an oversampling of 5 times the Nyquist rate is needed for GNU Radio correlation to work properly (see section 4.2), which places the upper frequency signal bound as 250MHz, due to the sample rate limit (see section 5.2). Therefore, $f_{highest} < 250kHz$. $f_2 = f_{highest}/2, f_3 = f_2/2$ and so on to ensure continuous phase generation and $5kHz < f_{lowest} < 60kHz$ due to GNU Radio limitation and for ambiguity resolution. To simplify the signal generation and reduce the processing load on the computer when correlating the tones, 4 tones were chosen to be used ($f_{highest} = f_1, f_{lowest} = f_4$) and 100 kHz was chosen as the maximum frequency given the tones as waves of frequency 100 kHz, 50 kHz, 25 kHz and 12.5 kHz.

Finally, to reduce pre-processing setup and post-processing process to find out the range, the signals were multiplexed together or, in other words, appended behind one another using the flowgraph in Figure 5.7 starting from the highest frequency signal to the lowest. By doing so, any phase discontinuities in the signals are immediately visible, and the GNU Radio setup goes from having 4 signals strictly timed to just one signal timed. The correlation peak for this signal will also only be attained when all 4 tones are perfectly aligned, thus giving the range directly as the correlation measurement. Otherwise, each tone will only give the range measurement up to its range resolution and ambiguity, which will need to be added up according to how the method was set up to obtain the range. The pre- and post-processing are reduced because instead of having to set up 4 tones according to a strict timing in GNU Radio, now there is only a need to set up and send one longer signal having all the 4 tones.



Options
Title: Sequenti...gnal Muxxer
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

File Source
File: ...anging Signal\Clock1
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

File Source
File: ...anging Signal\Clock2
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

File Source
File: ...anging Signal\Clock3
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

File Source
File: ...anging Signal\Clock4
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

Stream Mux
Lengths: 10k, 5k, 2.5k, 1.25k

Head
Num Items: 187.5k

QT GUI Time Sink
Number of Points: 187.5k
Sample Rate: 2M
Autoscale: No

File Sink
File: ...wnloads\MuxxedSignal
Unbuffered: Off
Append file: Overwrite

Figure 5.7: Muxxer block setup GNU Radio

5.1.2. Timing

Typically, sequential ranging has each tone sent one after another to the satellite and back. The timing needs to be suitably chosen to make sure each tone can be captured individually. It is necessary to integrate for longer than the length of the signal to ensure that the ranging signal is captured. This can be seen in Figure 5.8 where the additional 1-second gaps are provided to ensure the received sequence is completely received.

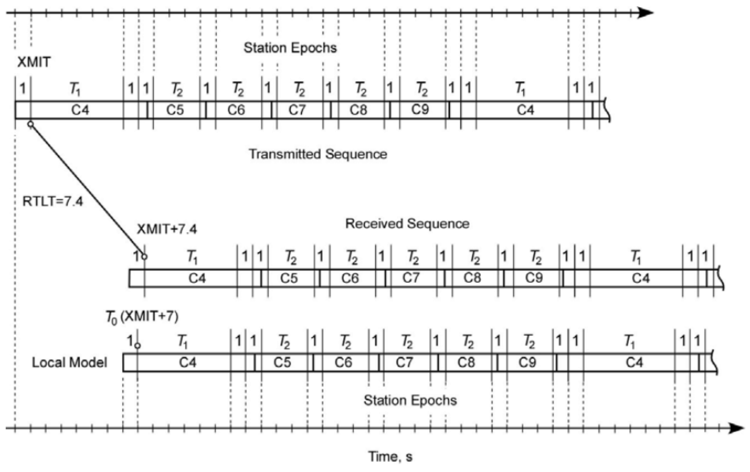


Figure 5.8: Sequential Ranging Timing, as Carried out in NASA DSN with $T_1 = 6s$ and $T_2 = 3s$[17].

For our use case, since the signals are appended together, the timing algorithm is simplified as in Figure 5.9. Each signal length is correlated to find the correlation peak.

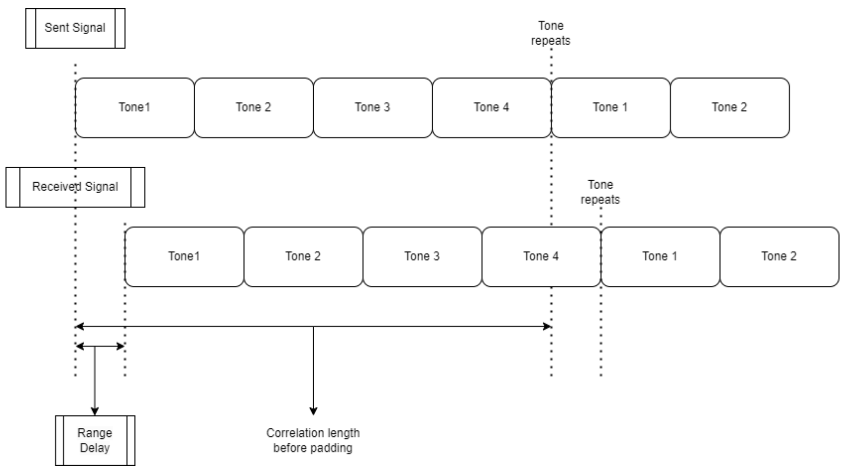
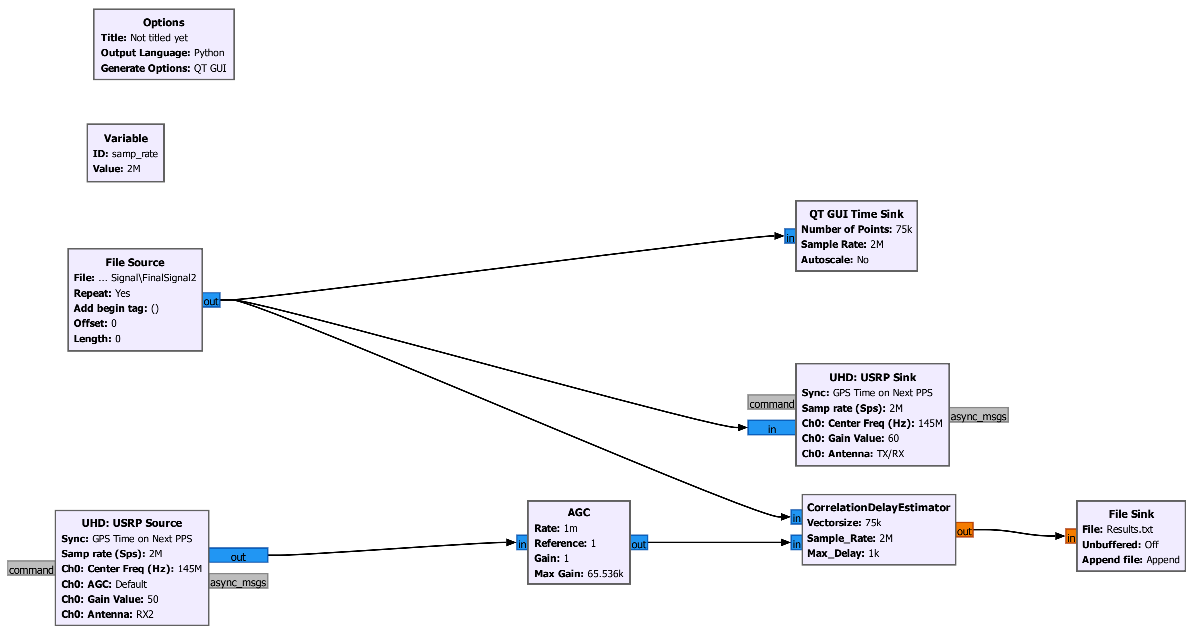


Figure 5.9: Sequential Ranging Timing For Tests

5.1.3. GNU Radio Setup

The processing is done in GNU radio for the ranging. The GNU radio setup is visible in Figure 5.10. In this figure, a pre generated sequential ranging signal is sent to the USRP sink block from the file source block. The USRP sink block interfaces between gnu radio and the USRP radio transmitter. Using this USRP block, different options for radio signal transmission such as the sample rate, carrier frequency and gain can be tweaked.

The signals picked up by the receiver emerge from the USRP source block. The source block is connected to an Automatic Gain Control (AGC) which matches the amplitude of the received signal with the reference amplitude of 1 (same as transmitted signal). Finally, the transmitted and received signals are sent to the custom correlation delay estimator block containing the correlation code as described in section 4.2. The correlation block decimates the output and returns 1 sample containing the delay value for every signal length. Depending on the computer performance, this real time correlation quickly becomes too expensive which was the case for correlating signals having length greater than 100,000. Therefore, it is necessary to store the signals offline and process them later in a separate GNU Radio flowgraph. This was used until the limit of GNU Radio decimation support was reached (see section 4.2). Beyond this point, GNU Radio saved files, which were in binary format [80], were directly read using the NumPy library and post-processed in Python.



Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

File Source
File: ... Signal\FinalSignal2
Repeat: Yes
Add begin tag: ()
Offset: 0
Length: 0

QT GUI Time Sink
Number of Points: 75k
Sample Rate: 2M
Autoscale: No

UHD: USRP Sink
Sync: GPS Time on Next PPS
Samp rate (Sps): 2M
Ch0: Center Freq (Hz): 145M
Ch0: Gain Value: 60
Ch0: Antenna: TX/RX

UHD: USRP Source
Sync: GPS Time on Next PPS
Samp rate (Sps): 2M
Ch0: Center Freq (Hz): 145M
Ch0: AGC: Default
Ch0: Gain Value: 50
Ch0: Antenna: RX2

AGC
Rate: 1m
Reference: 1
Gain: 1
Max Gain: 65.536k

CorrelationDelayEstimator
Vectorsize: 75k
Sample_Rate: 2M
Max_Delay: 1k

File Sink
File: Results.txt
Unbuffered: Off
Append file: Append

Figure 5.10: Real Time Sequential Ranging Set up GNU radio

Outside GNU Radio, the transmitter is connected to the receiver with the delay line in between as seen in Figure 5.2.

5.2. Performance

The different parameters that can affect sequential ranging accuracy are discussed in this section and chosen for use in the ranging test.

5.2.1. Frequency

The frequency of the sequential ranging signal is chosen for ambiguity resolution and noise performance. Instruments introduce a thermal noise into the system which is directly proportional to the temperature and inversely proportional to the frequency of the signal[17]:

$$\sigma = \frac{c}{f_{RC}.A_c.\sqrt{32.\pi^2.T_1.(P_R/N_0|_{D/L})}} meters, rms \tag{5.2}$$

where

f_{RC}=Range Clock Frequency.
A_C=Correction factor for amplitude being low during correlation($A_C <= 1$).
T_1= Integration time.
$(P_R/N_0|_{D/L})$=Downlink total downlink signal power to noise spectral density ratio.

In the best case scenario, this is also the theoretical limiting source of error for ranging. Practically, this limit was not seen in the tests because of limited usable resolution (sample rate) of the USRP-pc system.

5.2.2. Signal to Noise Ratio

The SNR feeds into Equation 5.2 affecting the thermal noise faced by the system, thus affecting the correlation process. Thus, the variation with SNR will determine the operable range of link budgets for the system. The SNR can be defined in terms of the voltage (amplitude) based SNR. Alternatively, literature, such as the telemetry ranging literature and NASA reports [75, 9], define SNR in terms of the signal energy A_{sig} and noise variance σ_n^2 . Both these are equivalent and given as:

$$SNR_{dB} = 10\log_{10}\frac{(A_{sig})^2}{(\sigma_n^2)}dB = 10\log_{10}\frac{(A_{sig})^2}{(A_{noise})^2}dB \tag{5.3}$$

So, the variance is nothing but the square deviation of the noise level from the mean. For Additive White Gaussian Noise (AWGN) used in simulations, the mean is the level of signal and the deviation is the noise amplitude specified and is equal to A_{noise}.

Practically, the number of symbols representing a peak is more than 1, which means that the amplitude of the peak would be given as the average of each of the samples representing a peak or a symbol. The relation between SNR of a symbol and a sample is defined in [75] as:

$$SNR_{Symbol} = SNR_{Sample} \times sps \tag{5.4}$$

The amplitude used for the sample's SNR is the RMS amplitude of the sample. where sps= no of samples per symbol.

5.2.3. Sample Rate

The sample rate at which the USRP runs determines the minimum measurable time, and hence also the raw range resolution of the ranging system. Samples are measurements of the signal value take at regular time intervals as described in section 4.1. In a system with a sample rate of 1000Hz, the minimum length of time which can be differentiated is given as $1/1000 = 0.001seconds$. Thus, the raw range resolution of this system will be approximately $0.001 * speed\ of\ light(c) = 3 * 10^5$ meters.

Different sample rates were tested to determine the ideal sample rates. The final sample rate needs to be suitable related to the supported clock rates of the USRP for ideal performance[59]. For this, it is useful to keep the selected sample rates as multiples of 2. The USRP will issue a console warning in case an unsuitable sample rate is chosen. The range resolution is related to the sample rate as

$$Range\ Resolution(sec) = \frac{1}{Sample\ Rate} \tag{5.5}$$

The variation of range error vs sample rate is given in Figure 5.11 and the histogram of the error is given in Figure 5.12. From the two graphs, it is apparent that 2MHz is the frequency with the least amount of range error. The lower frequencies just have a lower range resolution and this reflects in the error decreasing as can be seen in the histogram. The higher frequencies start to run into the limits of the USRP-Windows system and the errors are caused due to overflow, where the USRP's buffers become full, and it stops streaming data until the buffers are free again ('o' warning)[59]. This error becomes more and more ubiquitous at the higher sample rates affecting the range readings as visible in the graph.

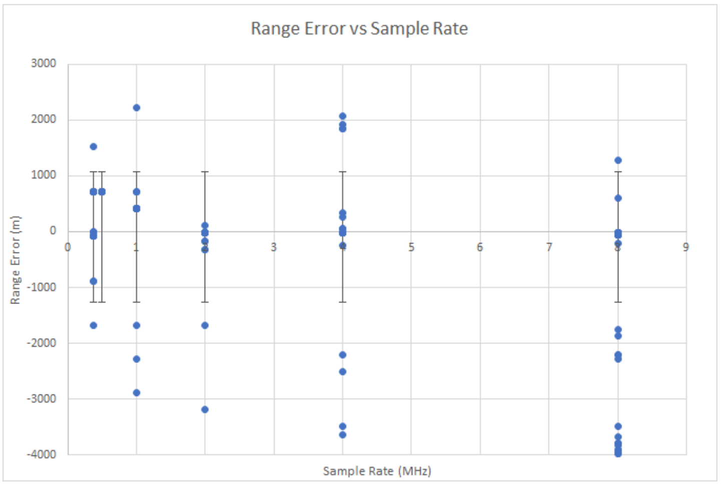


Figure 5.11: Range Error vs. Sample Rate for Sequential Ranging

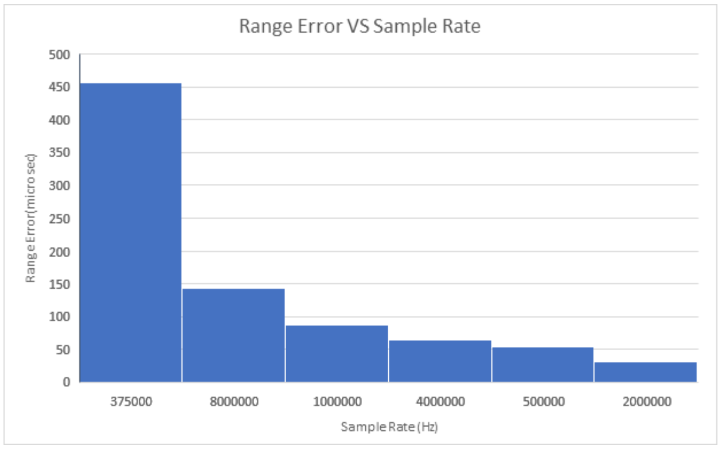


Figure 5.12: Histogram of Range Error variation with Sample Rate

5.2.4. Clock Synchronization

A block diagram of the USRP used for testing is given in Figure 5.13. In the USRP, the receiver and transmitter run on two separate chains of processing and have their own local oscillators ('LO' under AD9364, which is the direct conversion transceiver and baseband processor of the radio). These local oscillators need to be synchronized with one another to ensure that the transmission data stream and the reception data stream are aligned with one another. Any difference between these local oscillators will be visible as an offset between the transmitted and received stream, and hence as a range error. These local oscillators are derived from the radio's in the radio[37]. However, when using GNU Radio, this synchronization needs to be specified in GNU Radio's arguments.

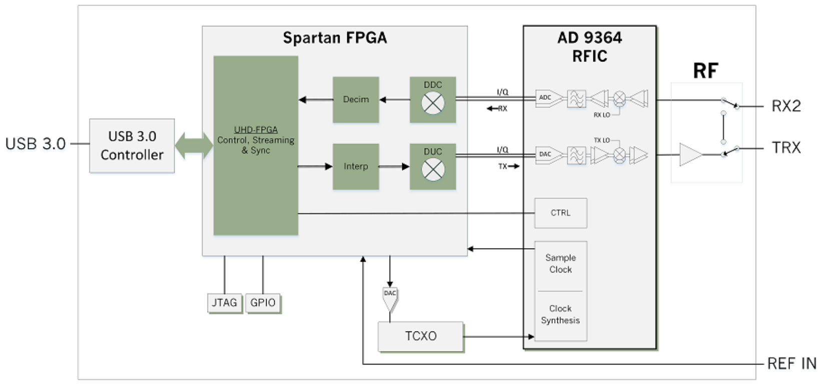
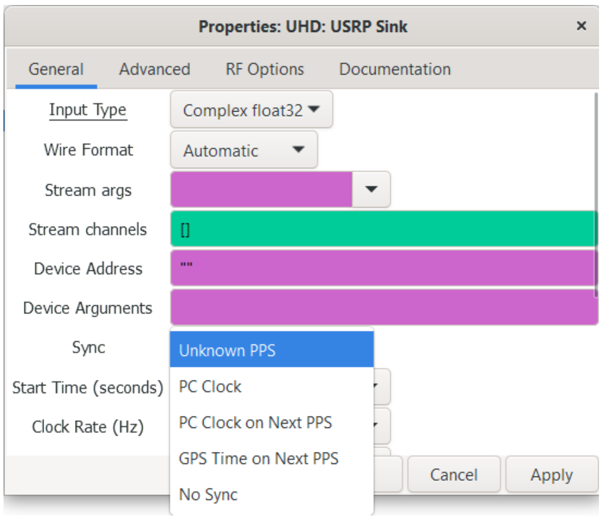


Figure 5.13: USRP B200 Block Diagram[27]

GNU Radio offers options such as synchronization to an external Pulse Per Second (PPS) Source (unknown PPS), sync to the PC clock (Immediately and on next PPS), and sync to GPS time as shown in Figure 5.14. The GPS time option is only usable when the USRP has a Global Positioning System Disciplined Oscillator (GPSDO), which is a system where the GPS time is used to synchronize the clocks to eliminate any long term drift while using a local oscillator to maintain the short term errors to a low level. This is because the short term drift of GPS errors are higher ($10^{-8}s$) due to noise effects on the received PPS signal from the GPS satellites when compared to ground local oscillators which only have their own drift errors(10^{-11}). Therefore, the local oscillators are better in the shorter term. However, when the time of operation goes over a certain limit (typically around 1000 seconds [37], but depends on the local oscillator), it is more effective to resync to GPS.



Input Type: Complex float32
Wire Format: Automatic
Stream args:
Stream channels: []
Device Address: ""
Device Arguments:
Sync: Unknown PPS
Start Time (seconds)
Clock Rate (Hz)

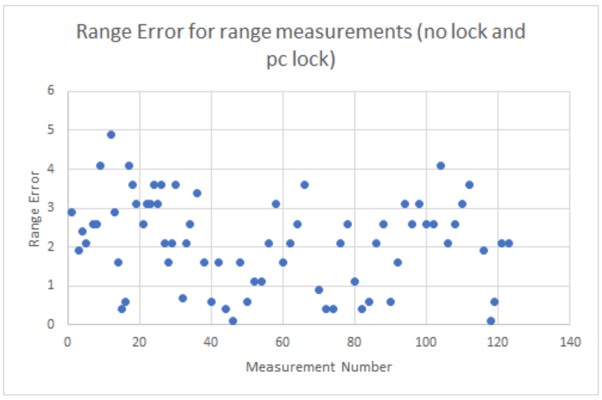
Unknown PPS
PC Clock
PC Clock on Next PPS
GPS Time on Next PPS
No Sync

Figure 5.14: USRP Clock Sync Options

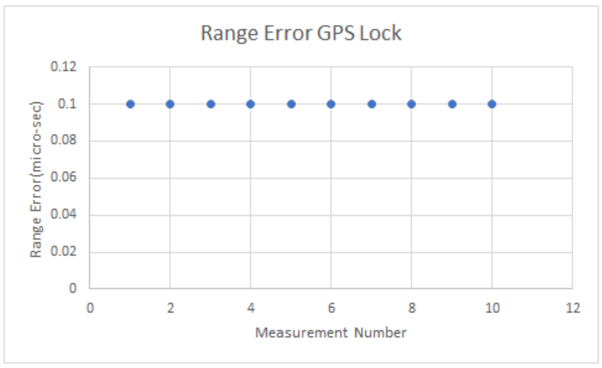
Without clock synchronization, the sequential ranging process showed a jitter in the order of microseconds. Since there was no external PPS reference at hand, the unknown PPS options could not be used and PC clock sync, PC PPS clock sync did not improve the jitter. However, with GPS time sync, the jitter reduced to the order of nanoseconds (3ns according to section 6.1.4). However, the GPS sync needed to be manually configured further before this was possible. The default GNU radio arguments just set each of the clocks to GPS time. However, this does not provide time for the National Marine Electronics Association (NMEA) string received from the GPS satellite to propagate from the through to the local oscillator. Further, the system just starts the receiver and transmitter to start whenever they can. This can also put the two stream out of sync in case they can only start at different times. Therefore, two measures were taken. One, once the GPS times were set, the code was set to sleep for a second to allow for the time sync to propagate. Two, the receiver and transmitter streams were set to start a fixed amount of time later (5 seconds), which would eliminate system latency effects on the system. These were incorporated by manually adding the following code into the GNU Radio Python file:

```
self.uhd_usrp_sink_0_0.set_clock_source('gpsdo', 0)
self.uhd_usrp_sink_0_0.set_time_source('gpsdo', 0)
self.uhd_usrp_source_0.set_clock_source('gpsdo', 0)
self.uhd_usrp_source_0.set_time_source('gpsdo', 0)
print(self.uhd_usrp_sink_0_0.get_mboard_sensor("gps_locked",0))
print(np.float64(self.uhd_usrp_sink_0_0.get_time_last_pps().get_real_secs()))
gpstime=self.uhd_usrp_source_0.get_mboard_sensor("gps_time").to_int() + 1.0
print(np.float64(gpstime))
starttime=gpstime+ 5
#sync to PPS for usrp source and sink
self.uhd_usrp_source_0.set_time_next_pps(uhd.time_spec(gpstime))
self.uhd_usrp_sink_0_0.set_time_next_pps(uhd.time_spec(gpstime))
#Set start time for usrp source and sinkv
self.uhd_usrp_source_0.set_start_time(uhd.time_spec(starttime))
self.uhd_usrp_sink_0_0.set_start_time(uhd.time_spec(starttime))
# Sleep 1 second to ensure next PPS has come
time.sleep(1)
print(self.uhd_usrp_sink_0_0.get_mboard_sensor("gps_locked",0))
print(np.float64(self.uhd_usrp_sink_0_0.get_time_last_pps().get_real_secs()))
print(np.float64(gpstime))
```

where $uhd\_usrp\_sink\_0\_0$ refers to the radio transmitter chain and $uhd\_usrp\_source\_0$ refers to the radio receiver chain.



(a)



(b)

Figure 5.15: Range measurement error in micro sec for **(a)**no sync and pc sync options and **(b)** GPS sync with fixed start time

5.2.5. Interpolation

The raw system resolution from the USRP radio is limited to the maximum sample rate the USRP and the connected system are able to handle. From our tests, the maximum value before errors start to dominate was determined to be 2MHz. However, at this sample rate, the minimum range that can be detected is 150 meters. Therefore, it is useful to obtain higher resolutions, which can be done through interpolation. Interpolation is done through the interpolating FIR filter block in GNU radio, which provides. Using this, quite high resolutions can be achieved. However, the presence of noise in the system will have the algorithm interpolating between noise affected samples, which will cause the interpolated samples to be also shifted, therefore interpolating can only increase the resolution but not reduce the effects of noise on the signal.

5.2.6. Automatic Gain Control

The AGC's gain feeds into Equation 5.2 through the amplitude correction factor and therefore affects the ranging result obtained. The ideal gain was determined to be 0.001 from Figure 5.16.

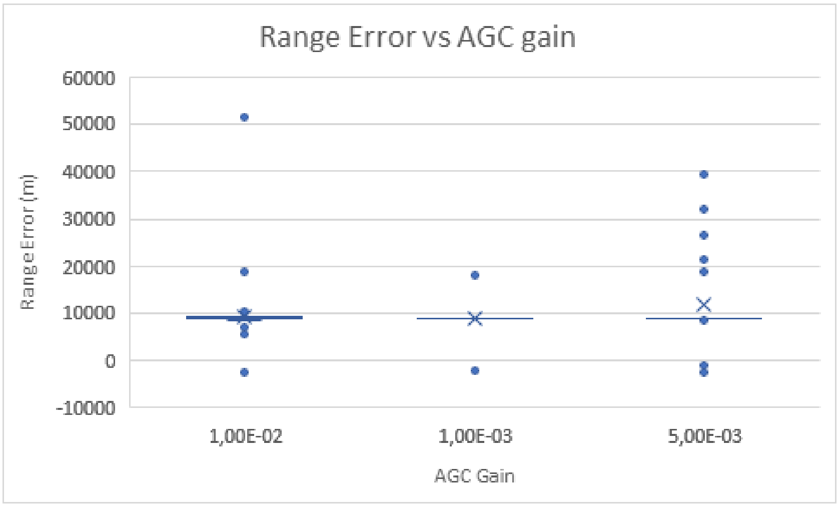


Figure 5.16: Range Error vs AGC Gain

5.3. Results

Sequential ranging simulations validated the correlation algorithm for use in GNU Radio. Following this, the USRP tests validated the use of USRP as a ground radio system.

5.3.1. Simulation

A simulation was carried out to test that the correlation was indeed working. The simulation was set up as described in the introduction of chapter 5. The SNR of the system was varied between -50dB to +50dB, as described in Figure 5.1. With the sequential ranging signal, the results were obtained as in Figure 5.17.

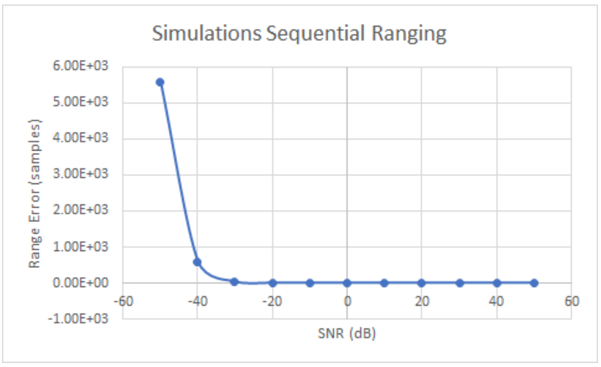


Figure 5.17: Sequential Ranging Simulations

The results show that the delay was determined to a high sample accuracy above -20dB SNR. Thus, the correlation algorithm is validated for further usage.

5.3.2. USRP Test

The USRP tests were carried out using a setup similar to the one described in Figure 5.2. Instead of adding noise, the gains of the receiver and transmitter were reduced instead to decrease the SNR.

The SNR of the system was varied by reducing the antenna gains in the radio and calculating the resulting SNR using the MPSK SNR estimator block[81]. The tests were carried out at 2 MHz sample rate, which was determined to be ideal above. This gave the system a resolution of 0.5μs or 150m. The system determined the error up to the system resolution for the higher SNR values.

The transmitter and receiver clocks of the USRP were synchronized using to GPS time and set to start at the same time. This eliminates any error due to the clock offsets in the range measurements (see section 5.2.4)

Practically, when processing signals, a shift in the timing is introduced into the signals by GNU Radio and the processing hardware. These time shifts have a constant component (bias) and a varying component (jitter). The bias can be estimated by conducting the overall range process without a distance between the transmitter and receiver. This is known as a calibration pass. For the calibration pass, a through connection is used to connect the USRP transmitter to the receiver as shown in Figure 5.18 (a). An attenuator is used here to ensure that the signal is not strong enough to damage the receiver.

For the calibration pass, a delay value is obtained as d_c. Following the calibration, the delay line element is connected as in Figure 5.18 (b). Ranging now gives another delay estimate, d_f. Taking d_f-d_c gives an estimate of the delay line delay whose true value is 13600ns $\pm$ 20ns[46].

Then the delay line is connected to the USRP and the correlation is run to obtain the delay value. The baseline is subtracted from this delay to obtain the real delay in the system.

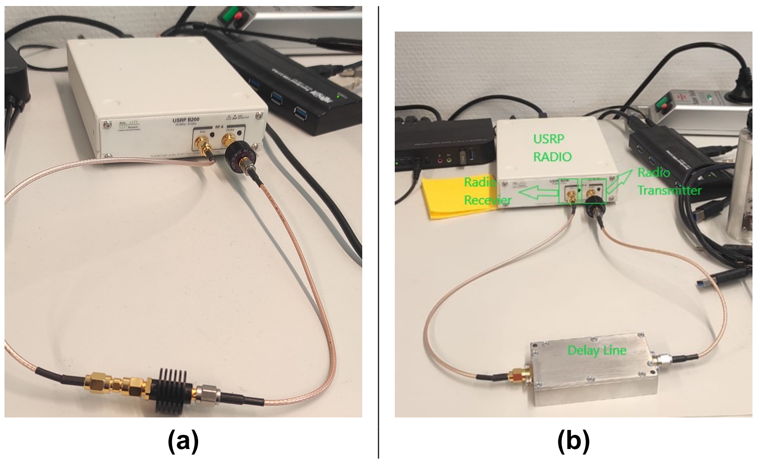


Figure 5.18: USRP with **(a)**Through Connection and **(b)** delay line

In the USRP test, the range performance improves at higher SNRs as visible in Figure 5.19. The system resolution was reached at the higher SNRs, which is why the value is not closer to zero. Most of the results were at the SNR limit, however, these tests had minor errors due to cyclic convolution of the correlation algorithm which is seen as the larger error values in higher SNR levels.

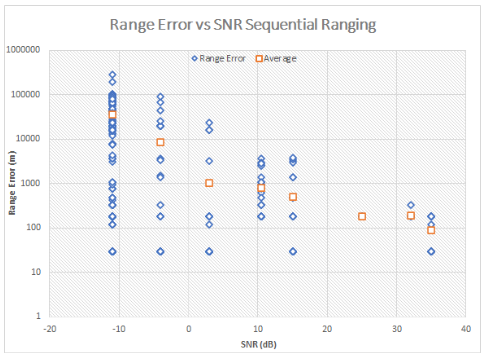


Figure 5.19: Range Error vs SNR

5.4. Conclusion

The system managed to find the delay from the delay line upto the system resolution in the best case. This means that it is feasible to range using the USRP as a ground radio system.

The next step would be to implement telemetry ranging in the simplified set-up as seen in Figure 5.1 with the telemetry ranging method.

6 Telemetry Ranging

The idea of telemetry ranging is to eliminate the presence of a dedicated ranging signal, which brings a number of benefits, as discussed in section 2.1.2. In order to do so, modifications are made to the data sent back from the satellite.

The chapter starts with section 6.1 describing the modifications necessary to use the telemetry ranging technique in the Delfi-PQ satellite. An exploration of the satellite command signals and the effect of higher order modulations (GMSK and GFSK) in GNU Radio on signal correlation performance is also done here.

In addition, in section 6.2, the modified telemetry ranging method is developed and tested using Delfi-PQ's engineering model available at TU Delft. The test setup is described, following which the results are presented.

Furthermore, in section 6.3, an improvement to the initially proposed method is suggested and incorporated into the satellite. The improvements in performance and additional requirements placed on the satellite due to this improvement are discussed.

Finally, in section 6.4, this chapter is recapped, and it is discussed how the obtained results meet the requirements.

6.1. Telemetry Ranging for Delfi PQ

Recapping the discussion on hybrid ranging (section 2.1.2), there exist two techniques-telemetry ranging and telecommand/telemetry ranging. Telemetry ranging eliminates the ranging signal on the downlink, but still needs a dedicated ranging signal and processing equipment on the uplink. This technique was validated in a lab test and is set to be flight tested in the Europa Clipper mission[33].

In the telecommand ranging [75], the method was further refined by eliminating the dedicated ranging signal on the uplink too and using satellite telecommand codewords that were already being sent instead. This method makes use of a specially designed demod-remod technique to improve the satellite clock recovery on the uplink. The telecommand ranging method was tested in simulations[75]. However, there is no information on whether this technique was lab tested or used in flight.

Telemetry ranging and telecommand/telemetry ranging are the two available choices for use in the thesis.

With the Delfi-PQ satellite, there are a few features that affect the implementation of telemetry ranging.

1. The satellite does not support the use of a ranging signal on the uplink communication channel. It does not have the necessary equipment. Therefore, only a technique along the lines of

telecommand/telemetry ranging, which eliminates both uplink and downlink ranging signals can be used.

2. The satellite uses the Serial Peripheral Interface (SPI) data communication standard to connect the modem (Semtech SX1278) to the microcontroller (MSP432P401 for the flatsat and MSP432P411 for flight model) of Delfi-PQ. This SPI is configured to transfer the demodulated data 8 bits at a time to the microcontroller using an interrupt. This is great for the data processing but also means that the exact arrival time and departure time (t_{sr} and t_{tr} in Figure 2.13) of each individual bit of data from the satellite cannot be directly measured and can only be estimated. The means that an accurate fractional bit count cannot be maintained as in telecommand ranging (section 2.1.2), where they have access to the signal demodulated version of the signal, which they ostensibly use to maintain the bit count (though it is not specified how this is implemented in [75]).

These reasons prevent the use of telecommand/telemetry ranging as it was presented in the literature.

Therefore, a modified method is suggested where the satellite pings back received commands in the telemetry stream (or attaches a pre decided marker behind the telemetry data marking that a command was received recently) and these are received on the ground. Once the ping of the command or the marker is received, the command header or the known marker can be correlated for. Based on prior knowledge of the range or logical reasoning (such as the obtained range for each set of measurements should be similar), each codeword can be associated with the correct marker on the ground. The entire process would be as in Figure 6.1.

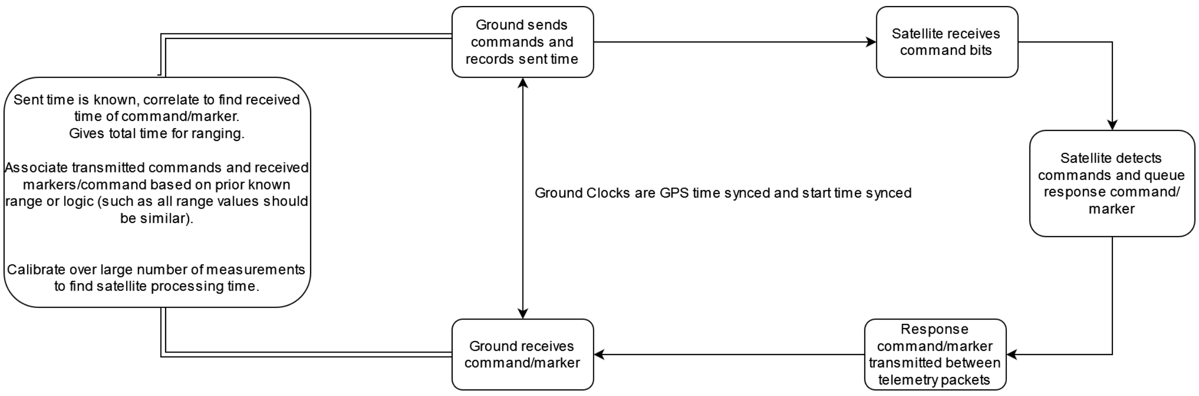

Ground sends commands and records sent time
Satellite receives command bits
Sent time is known, correlate to find received time of command/marker. Gives total time for ranging.
Associate transmitted commands and received markers/command based on prior known range or logic (such as all range values should be similar).
Calibrate over large number of measurements to find satellite processing time.
Ground Clocks are GPS time synced and start time synced
Satellite detects commands and queue response command/ marker
Ground receives command/marker
Response command/marker transmitted between telemetry packets

Figure 6.1: Telemetry Ranging suggested for Delfi-PQ

This method has a few advantages and disadvantages:

+ This retains the advantages of telemetry ranging such as not requiring a dedicated ranging signal, which gives improvements in bandwidth and allows higher order modulations.
+ In sending back the signal, the codeword/marker is regenerated, meaning a fresh copy is sent, this has improvements in noise levels for the signal similar to regenerative pseudo-noise ranging (upto 40dB of gain compared to transparent ranging[34], see section 2.1.2).
+ Since a known sequence is present in the downlink, it can be directly correlated for, thus eliminating the need for tracking loops on the ground, which eliminates the tracking loop symbol timing errors on the ground. Instead, the correlation algorithm error is now added.

  The correlation performance for GFSK signals used in the thesis follows from the blue line in Figure 6.10. For example, at a SNR of 30dB (which follows from Figure 5.19), this will give a deviation of 0.0001symbols. Since 100 samples per symbol are used, this gives a sample deviation of $100 \times 0.0001 = 0.01$ samples. At a sample rate of 2Msps, 1 sample is equal to $1/2Msps = 5 * 10^{-7} seconds$. The range error for this timing is given as $0.01 \times 5 * 10^{-7} \times speed\ of\ light(c)$= 1.5m.

- The jitter of the satellite in processing (reception, detection, regeneration of the command and sending the command) will carry over to the range measurement because this channel is not designed with ranging levels of precision in mind.

- The clock recovery error from the tracking loops will remain because it is present in the hardware and cannot be changed mid-flight.
- The telemetry channel will have a minor impact (impact discussed in later sections) on the telemetry transfer speed.

Therefore, the initial step in implementing telemetry ranging would be to transmit and receive modulated codewords in the setup of Figure 5.1 and correlate them to obtain the delay of the delay line. This would demonstrate that it is feasible to

1. Use GMSK/GFSK modulation with the signal for ranging.
2. Range with the command codewords of the satellite. This may also be inferred to be possible from the sequential ranging tests, however, it is necessary to validate each part of the system one step at a time so that all issues that could arise are identified at an early stage.

6.1.1. Delfi PQ Command Codewords

The codewords in use in the Delfi-PQ satellite are of a specific form as given in Figure 6.2. First, there is a pilot sequence indicating the start of a command.. This is followed by a header indicating the source and destination of the command. The 4 byte command data along with a 2 byte Cyclic Redundancy Check (CRC) sequence follow the header. This is then rounded off by a tail sequence indicating the end of the command.

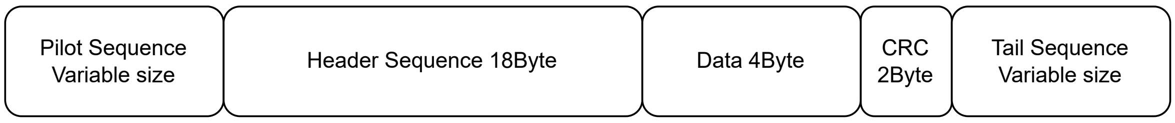


Figure 6.2: Delfi PQ command sequence

The pilot and tail sequences are sequences containing a number of hexadecimal ’7E’s (number 126 in decimal and 1111110 in binary). The pilot and tail sequences can be of different lengths. Lower lengths allow more packets to be sent and hence more measurements to be taken. Higher lengths reduced the probability that the satellite misses the command. 20 Bytes of pilot and 20 Bytes of Tail were found to be a good balance between packet length and command recognition, with the command misses being very rare (a numerical measurement of the hit/miss ratio was not done). When the pilot sequences are detected, the satellite checks the incoming data stream for the presence of a command. The command data and header sequences are pre-defined by the satellite designers. There is one header which indicates that the transmission is from ground to satellite and another indicating vice versa. The command data directs the satellite to execute one of several pre-defined functions.

The commands are encoded, scrambled and modulated before transmission and are demodulated, descrambled and decoded on the satellite. The encoding and scrambling are implemented using Python on the ground and C++ on board the satellite.

Encoding changes the data sent according to an algorithm. Only those having knowledge of the algorithm used, and the settings employed can read the signal. This is done for security because radio signals are sent over the air and anyone can intercept and read them. This also prevents the satellite from mistakenly recognizing stray radio signals as commands. Encoding on Delfi-PQ is done using the AX.25 data link layer protocol, which was written to allow robust data transmission and error correction between devices using amateur radio communication channels. This allows for standardized transfer of information.

Further, Delfi-PQ uses Non Return to Zero Inverted (NRZI) differential encoding scheme as part of the AX.25 encoding. In, NRZI encoding, the presence of bits are represented by a change in state of a signal rather than the absolute value of the signal. This is visible in Figure 6.3, where the signal state changes for the presence of 1 and stays the same to indicate the presence of zero. This makes the signal immune to polarity changes in the electrical system. The Non Return to Zero (NRZ) part stands for the signal having two states which are non-zero voltages. This signal does not have a return to zero state.

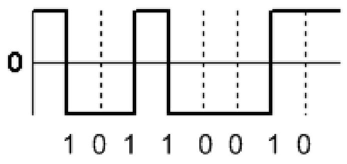


Figure 6.3: NRZI Differential Encoding

Scrambling (or data randomising) further changes the encoded data to be different at every point of time so that patterns in the data cannot be used to extract encoded information. This can be thought of as a type of encryption. The scrambler has a seed based on which it operates. The satellite, knowing the seed, can unscramble the code. Scrambling has two additional effects:

- It spreads the data over the entire frequency bandwidth, improving the power efficiency of radio components.
- It adds more state shifts to the signal, reducing the number of bit slips that occur when receiving a signal.

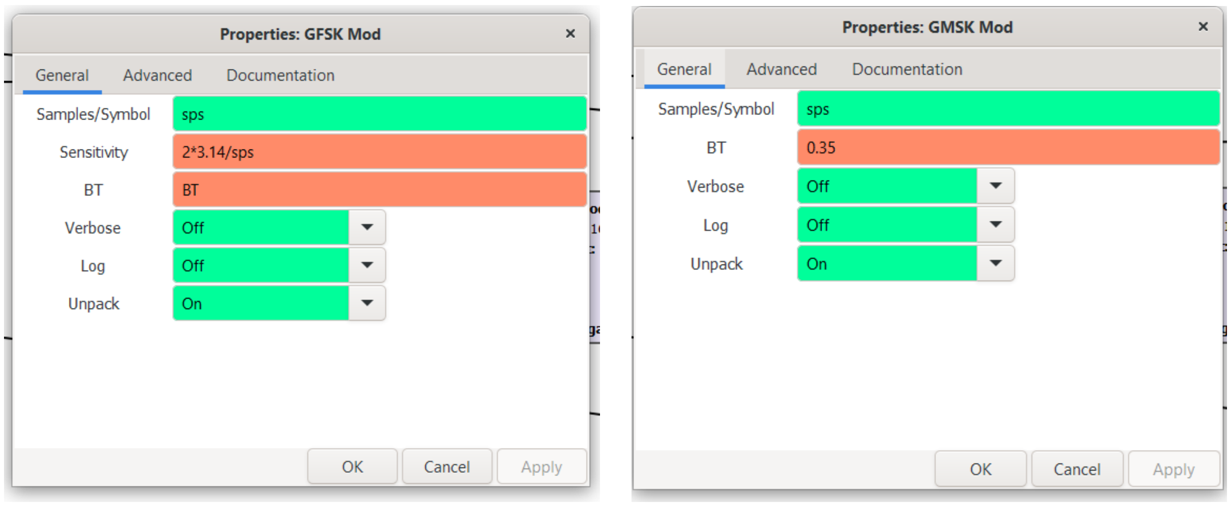
. For Delfi-PQ, a G3ruh encoder was used to scramble the command.

The code to generate a command codeword for Delfi-PQ(Listing A.6) was written based on earlier code of Delfi-PQ ground station code. The AX.25 encoding scheme and G3ruh scrambling algorithm were treated as black boxes for the purposes of the thesis.

6.1.2. GMSK and GFSK Modulation

There are two modulation options that are supported by the Delfi PQ modem, GFSK and GMSK. Frequency Shift Keying (FSK) is a modulation scheme that works by changing the frequency of the signals to encode information. GFSK works by adding a Gaussian filter over the modulated data to make frequency transitions smoother. This improves the spectral distribution of the signal, reduces the side-band power and interference with neighboring signals. Minimum Shift Keying (MSK) a special case of frequency shift keying where the difference between the higher and lower frequency is half the bitrate and GMSK adds a Gaussian filter to the transition. This is an edge case of frequency shift keying with a maximum bandwidth efficiency.

The GMSK and GFSK modulations are available in the form of mod blocks in GNU Radio. The modulation parameters of the blocks can be adjusted as in Figure 6.4.


Properties: GFSK Mod

General Advanced Documentation

Samples/Symbol: sps
Sensitivity: 2*3.14/sps
BT: BT
Verbose: Off
Log: Off
Unpack: On

OK Cancel Apply

Properties: GMSK Mod

General Advanced Documentation

Samples/Symbol: sps
BT: 0.35
Verbose: Off
Log: Off
Unpack: On

OK Cancel Apply

(a) **(b)**

Figure 6.4: Block options for **(a)**GFSK Mod and **(b)**GMSK Mod

The parameters were defined as given after initial investigations:

- Samples/Symbols (sps): Not to be confused with samples per seconds (which is the sample rate unit, see section 5.2), this value, also known as the oversampling ratio, gives the number of samples representing a bit of data. If sps is set to 100 and sample rate is 1000Hz, then 1 bit will have 100 samples representing it and a length of $100/1000 = 0.1 seconds$. Therefore, this ties the length of a modulated bit of data to the sample rate of the system. Practically, this should be set to the closest integer value such that the satellite bit length is matched by the modulated signal. It is defined as
$$sps = \frac{sample\_rate\_ground}{bit\_rate\_satellite} \tag{6.1}$$
For this section, sps=100 because telecommand ranging literature used that value.
- Sensitivity(GFSK only): The sensitivity of this block defines how much the phase of the modulated signal(in radians) changes based on the new input sample. For a sps of 100, if the desired change is 90 for a bit (or symbol)° (corresponding to a Modulation Index (MI) of 0.5 which is GMSK modulation), the sensitivity is
$$Sensitivity = \frac{\pi}{2 \times sps} \tag{6.2}$$
The only difference betwee GFSK and GMSK modulation is that MI is >0.5 for GFSK and =0.5 for GMSK. The MI represents how much the frequency of the modulated signal will shift to represent a bit. If the MI is 0.875, then the frequency changes by 87.5%.

  In testing however, for both GMSK mod block of GNU radio and GFSK mod block with MI 0.5, Delfi PQ failed to recognize the packets. To isolate this issue, the GMSK modulation was tested using the ISISpace CheckoutBox Software (which also uses the same USRP b200 as the ground radio), and the transmitted packets were recognized by Delfi-PQ. Therefore, GNU Radio being the only difference between these tests, this is likely a setup or implementation issue of GNU Radio, where it is failing to properly modulate the signals at the lower modulation index. A deeper exploration of this issue is suggested as part of future investigations if GNU Radio is still to be part of the ranging process. For the thesis, GFSK modulation with a MI of 0.875 was used, with a sensitivity of
$$Sensitivity = \frac{2 * \pi}{sps} \tag{6.3}$$
- : BT: 0.5
The bandwidth-time product determines the bandwidth the modulator will use (the symbol time

is fixed by the sps variable as $symbol\ time = \frac{sps}{sample\_rate} = \frac{1}{bitrate}$). A lower bandwidth is better for noise control, but a sufficiently high bandwidth is needed for the modulation to work. See Figure 2.2.3 for more information. This parameter was already fixed by Delfi-PQ's designers.

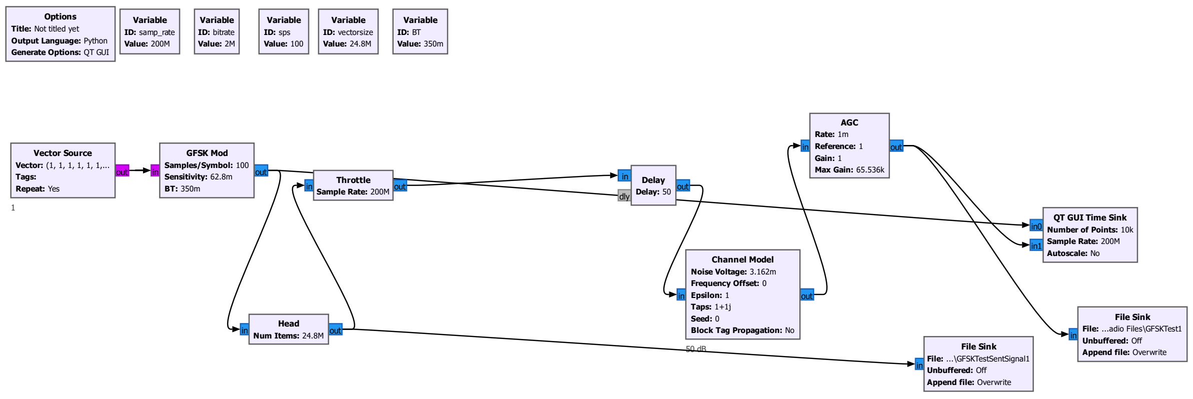
- : All other parameters in GNU Radio are set to default as in Figure 6.4.

Next, the aim of the following tests are to test the command correlation in simulations and in USRP tests.

6.1.3. Simulation and Validation with Literature

Simulations were conducted to obtain the performance of the systems under different SNR conditions and modulations. The standard deviations were calculated per bit(given as $\sigma_{bit} = \frac{\sigma_{sample}}{sps}$), to put the results on the same terms as in literature. The data was averaged from simulation runs of 500 points per SNR.

In Figure 6.5, the command signals, converted to bits by the GNU radio's vector source block are modulated using the GFSK mod (seen in figure) and GMSK mod (not seen in figure) blocks. This modulated signal is then stored using the lowermost file sink block. A copy of the modulated signal is delayed by 50 samples using the delay block and passed through the channel model block which adds a specified amount of noise to the signal. The added noise was varied between -50dB and 50dB of sample SNR (see section 5.2.2 for SNR Definition) and this noisy copy was stored to another file. Finally, using a python version of the correlation algorithm in section 4.1, the signal and its delayed copy were correlated to find the delay between them. Since the induced delay is 50 samples, the variations from this were recorded as the delay error.


Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 200M

Variable
ID: bitrate
Value: 2M

Variable
ID: sps
Value: 100

Variable
ID: vectorsize
Value: 24.8M

Variable
ID: BT
Value: 350m

Vector Source
Vector: (1, 1, 1, 1, 1, 1,...
Tags:
Repeat: Yes

GFSK Mod
Samples/Symbol: 100
Sensitivity: 62.8m
BT: 350m

Throttle
Sample Rate: 200M

Head
Num Items: 24.8M

Delay
Delay: 50

Channel Model
Noise Voltage: 3.162m
Frequency Offset: 0
Epsilon: 1
Taps: 1+1j
Seed: 0
Block Tag Propagation: No

AGC
Rate: 1m
Reference: 1
Gain: 1
Max Gain: 65.536k

QT GUI Time Sink
Number of Points: 10k
Sample Rate: 200M
Autoscale: No

File Sink
File: ...adio Files\GFSKTest1
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...\GFSKTestSentSignal1
Unbuffered: Off
Append file: Overwrite

Figure 6.5: Simulation Flowgraph for GFSK and GMSK Mod Tests

Subsample Peak Estimation

A subsample estimate of the delay was calculated by quadratic curve fitting over 5 data points using scipy.interpolate.interp1d function. The literature used three points to curve fit, but python ran into errors without 5 data points for the quadratic fit. This quadratic fit function was optimized using scipy.fminbound function(uses Brent's function[13]) to find the maximum correlation peak. These subsample peaks were slightly different from the correlation algorithm peak and are visible in Figure 6.6. Please note that the markers used in the thesis graph are different than the literature graph for better visibility.

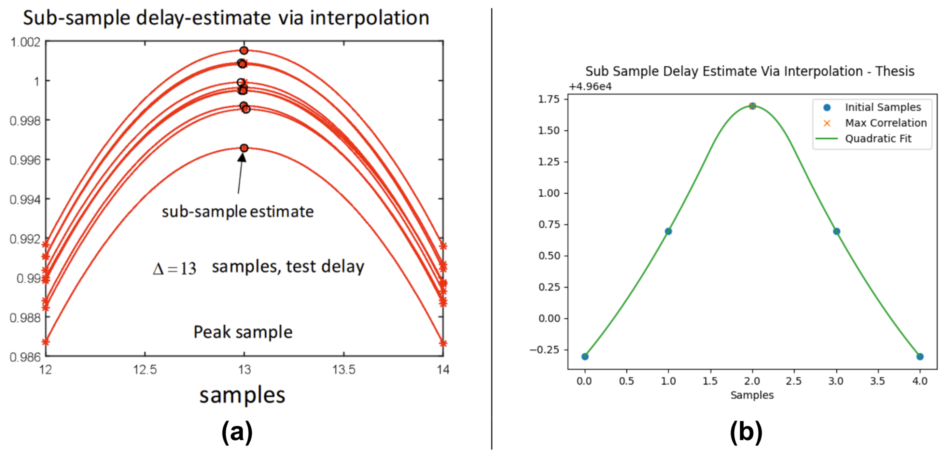


Figure 6.6: Sub Sample Peak Estimation of **(a)**Literature[75] and **(b)**Thesis

This subsample estimation allows for a quick higher resolution approximation of the delay value. It is employed in this section to put the simulations on the same level as the literature. It is also employed in the USRP tests of this section to get an estimate of the jitter due to the ground systems, and the results are presented because they are interesting to see, though they are to be considered with caution. The additional caution is because there was not sufficient time to validate to what extent the results of this subsample estimation were true. However, the results are not expected to be too wrong, because:

1. They were used in the telecommand ranging literature[75] as one of the main algorithms used to get their results.
2. Mathematically, a correlation is essentially multiplication of two 1D matrices in our case. The resulting equation of this multiplication will have an order of 2, so a quadratic approximation makes sense.
3. The obtained jitter for the GFSK modulation with MI of 0.875 gives a time jitter of $\simeq$ 2ns when no delay line is connected and $\simeq$ 18.5ns when delay line is connected (see section 6.1.4), which is close to the expected contribution of delay line ($\simeq$20ns from documentation[46]).

Nevertheless, the algorithm is not used in the future sections. As a future suggestion, it is suggested to validate this subsample estimation algorithm for use with real world tests over a different SNRs. This would provide a quick and convenient way to see beyond the raw sample rate.

Demod-Remod Process

The demod remod process of the literature discussed in section 2.1.2 was also modelled in simulation. The blocks in Figure 6.7 were added to the end of the flowgraph in Figure 6.5. The demodulated remodulated signals has an error in its timing due to the tracking loop clock recovery inside the demodulation block (as discussed in section 6.1). To estimate this time error, the remodulated signal, containing this error, is correlated with the original signal.

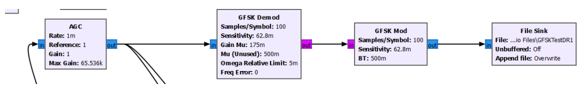
Further the literature suggests the use of windowing to improve it's correlation results. Because the received signal is processed before correlation, it is a known waveform. Therefore, based on its structure, a window function can be designed which will set the signal value away from transition points to either 0 or 1 based on known information, reducing the overall noise in the original signal.

In the simulation test, the initial signal was delayed by a known amount of samples (say $delay\_1\_2$ to obtain the received signal) The received signal is demod remodded to obtain a demod remod signal delayed by $delay\_2\_3$. The overall delay value between initial and demod remod signal is given as $delay\_1\_3$. Thus the demod remod error is given as

$$demod\_remod\_error = delay\_1\_3 - (delay\_1\_2 + delay\_2\_3) \tag{6.4}$$

$delay\_1\_3$ and $delay\_2\_3$ are determined by correlation and $delay\_1\_2$ is known as it is the induced delay.

The results of this were plotted as the green curve in Figure 6.10. Interestingly, the demod remod delay estimate is worse than the literature. This could be because of the thesis simulations using two correlations to obtain the error in the estimate, and due to GNU Radio's demod remod implementation.



AGC
Rate: 1m
Reference: 1
Gain: 1
Max Gain: 65.536k

GFSK Demod
Samples/Symbol: 100
Sensitivity: 62.8m
Gain Mu: 175m
Mu (Unused): 500m
Omega Relative Limit: 5m
Freq Error: 0

GFSK Mod
Samples/Symbol: 100
Sensitivity: 62.8m
BT: 500m

File Sink
File: ...io Files\GFSKTestDR1
Unbuffered: Off
Append file: Overwrite

Figure 6.7: Demodulation Remodulation Blocks

Results of Simulation

In Figure 6.10, the simulation results from GMSK (grey line), GFSK modulation (0.875 MI, blue line) and demod-remod correlation (green line) are plotted along with literature results (k=1 from Figure 6.8) the blue line representing the results for GFSK modulation shows a better standard deviation when compared to the grey line showing GMSK modulated simulations. The GMSK mod data has excellent agreement with the literature data (k=1 from Figure 6.8) with a square of Pearsons Correlation Coefficient (PCC) or R^2 of 0.94, indicating great correlation between the two data sets.

PCC provides a normalized value between -1 and 1 for correlation between two sets of data. 1 indicates maximum positive correlation, -1 indicates maximum negative correlation, which means if the signs of one of the sets of data were reversed, they would be perfectly correlated and 0 means there is no correlation between the data. The PCC between the datasets is given as

$$r_{xy} = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\Sigma_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.5}$$

The square of this (R^2) was calculated using the RSQ function on Excel.

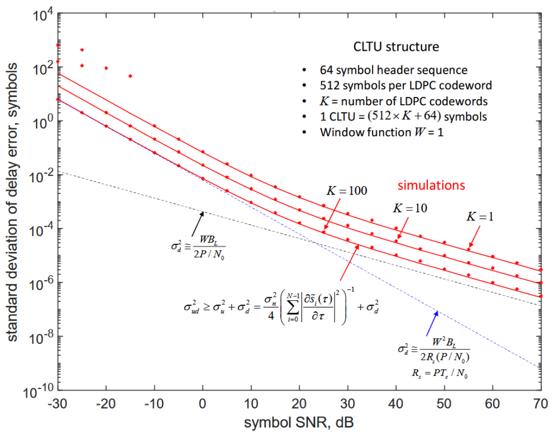


Figure 6.8: Command Signal Correlation Simulation from Literature[75]

From the graph and the obtained R^2 value of 0.94, the correlation performance of GMSK codewords matches well with the literature, albeit a bit worse. The visible deviations between the GMSK correlation and literature can be explained through theory. Looking at Figure 6.8, the blue dotted line is the performance limiting Cramer Rao Bound of the DTTL used to time the downlinked telemetry packets in telecommand ranging. The performance of codeword correlation on the uplink is limited by the correlation Cramer Rao Bound (black dotted line, orange line in Figure 6.10). The literature performance (yellow) is given as the sum of these two limiting errors.

The results obtained in the thesis simulation should match the correlation error of the literature (offset due to insufficient correlation length for K=1), meaning the slope of the results should approximately follow the slope of the correlation Cramer Rao bound (orange line). The additional error component due to the DTTL tracking loop is higher than correlation on the low SNRs. Thus, the literature results are slightly worse at low SNR. At high SNRs, the DTTL error component reduces and the correlation error becomes the limiting factor. Here, the slightly better performance of the literature correlation leads to better overall performance. This difference in correlation performance could be due to:

1. GNU Radio's implementation of modulations. They already showed issues with being recognized by Delfi-PQ.
2. The signal structure could be better for the literature. Better signal structure can lead to better correlation performance, though not better than the Cramer Rao bound. In Figure 6.9, better performance by a factor of 10 for the same SNR is seen for the sequential ranging signal as compared to the GMSK codewords (section 5.3.1) if the quadratic interpolation algorithm is applied.

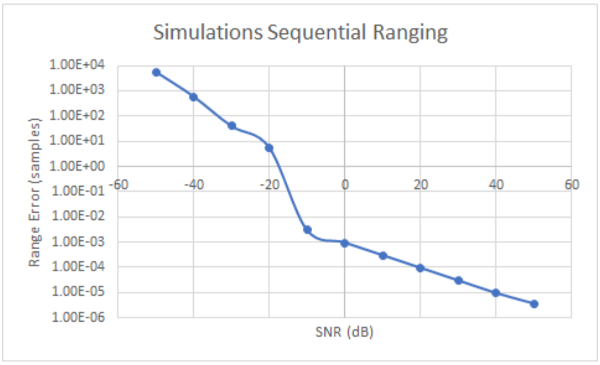


Figure 6.9: Sequential ranging simulations with quadratic interpolation

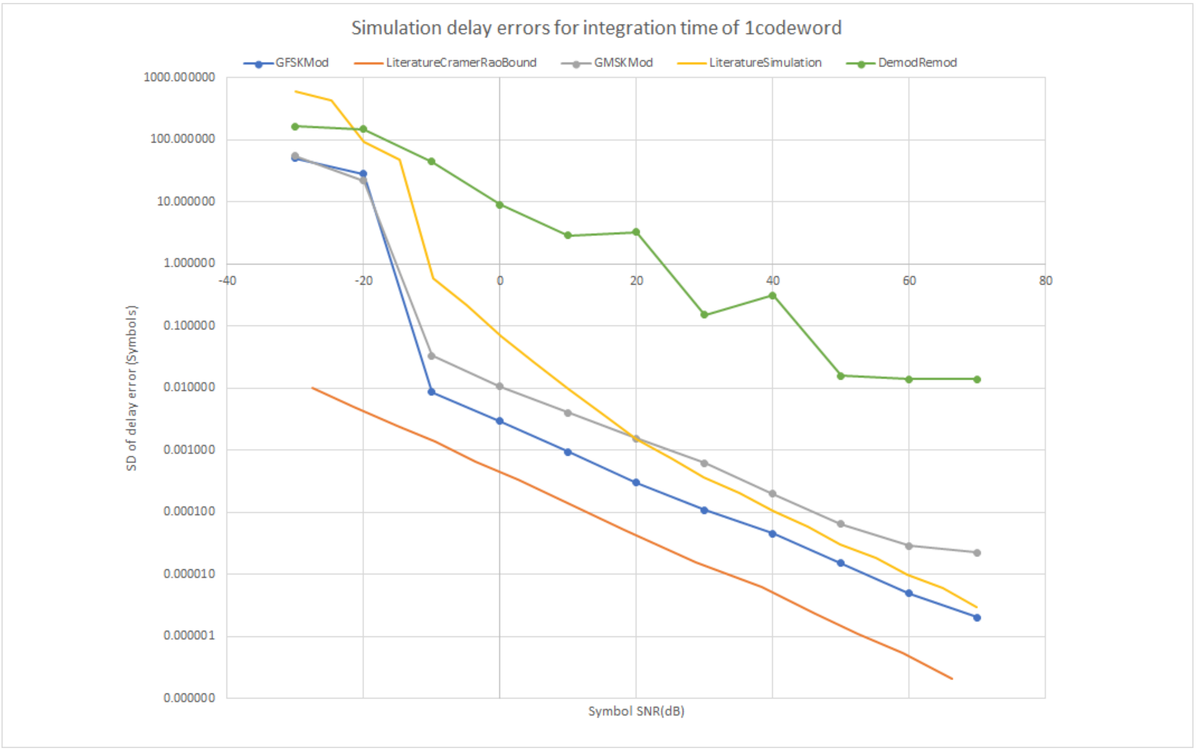
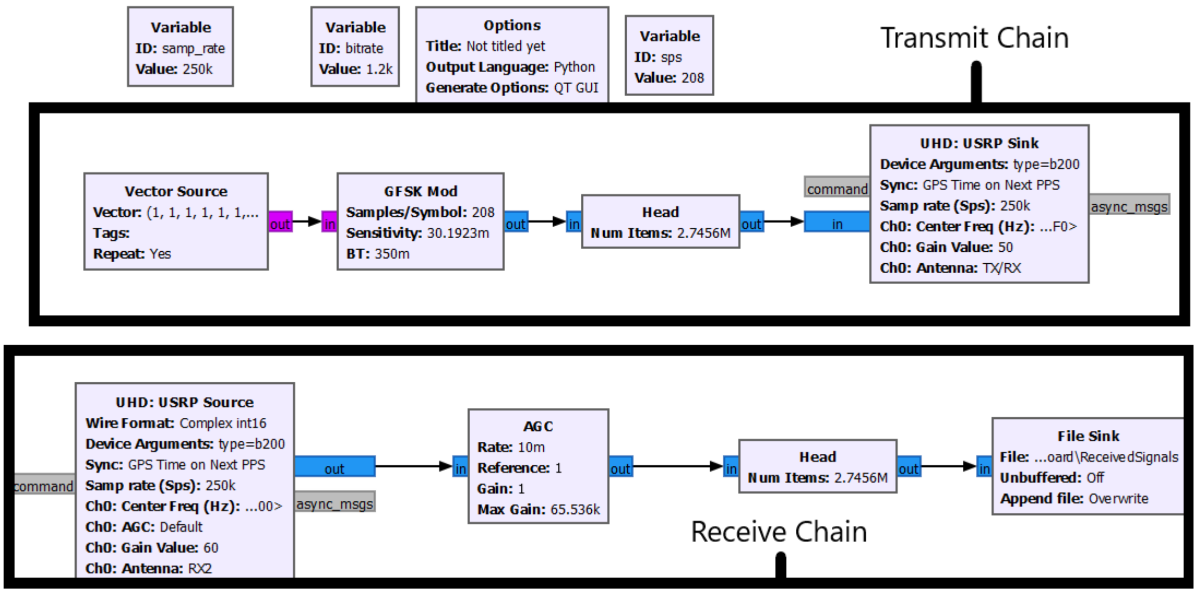


Figure 6.10: Delay Estimation error vs SNR for Modulated command correlation (sps=100, k=1)

6.1.4. Modulation Tests with USRP

Following the simulations, the signals were sent through the USRP to see if any differences arise when testing with real hardware. As in the previous USRP test, the transmitter and receiver of the radio were synchronized to GPS time and set to start at the same time to minimize time offsets between the two. Calibration passed for the radio and ranging passes for the delay line were conducted (similar to section 5.3.2).

Similar to the simulation, the decimal command sequence was converted to bits using the vector source block in GNU radio. These bits were modulated and transmitted using the UHD:USRP sink block in Figure 6.11. The head block limits the number of samples that can pass through, with it being configured here to send a fixed number of commands. In the receive chain, an AGC is used to match the amplitude of received signals with the transmitted signal. These adjusted signals are then recorded using the file sink block. The recording is done to enable offline correlation and post-processing to find the range as real time processing was found to be too computationally expensive for the PC.


Variable
ID: samp_rate
Value: 250k

Variable
ID: bitrate
Value: 1.2k

Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: sps
Value: 208

Vector Source
Vector: (1, 1, 1, 1, 1, 1,...
Tags:
Repeat: Yes

GFSK Mod
Samples/Symbol: 208
Sensitivity: 30.1923m
BT: 350m

Head
Num Items: 2.7456M

UHD: USRP Sink
Device Arguments: type=b200
Sync: GPS Time on Next PPS
Samp rate (Sps): 250k
Ch0: Center Freq (Hz): ...F0>
Ch0: Gain Value: 50
Ch0: Antenna: TX/RX

UHD: USRP Source
Wire Format: Complex int16
Device Arguments: type=b200
Sync: GPS Time on Next PPS
Samp rate (Sps): 250k
Ch0: Center Freq (Hz): ...00>
Ch0: AGC: Default
Ch0: Gain Value: 60
Ch0: Antenna: RX2

AGC
Rate: 10m
Reference: 1
Gain: 1
Max Gain: 65.536k

Head
Num Items: 2.7456M

File Sink
File: ...oard\ReceivedSignals
Unbuffered: Off
Append file: Overwrite

Figure 6.11: GNU Radio flowgraph for command based telemetry ranging

The quadratic interpolation algorithm was also employed here to get an estimate of the subsample delay to get better than sample rate resolution. As discussed in section 6.1.3 the results are to be considered with caution, but can be expected to be close to the real value.

Thedelay estimation errors wer obtained for the high SNR region and were plotted in Figure 6.12 for GMSK modulation, GFSK modulation (MI = 0.875) and GFSK modulation (MI=0.5). The bias and jitter can be seen for the different modulations in Table 6.1. The following conclusions can be drawn from these results:

- GMSK, shows incredibly high bias and jitter for all the tests, meaning the modulation completely failed to work for the ranging. Similar to when it did not work with Delfi-PQ, the reason is expected to be GNU Radio's modulation implementation.
- GFSK (MI=0.875) shows medium bias and low jitter. GFSK (MI=0.5, should theoretically be GMSK modulation), surprisingly works even though Delfi-PQ failed to recognize it and shows relatively lower bias and higher jitter than GFSK (MI=0.875). The jitter decreases with increase in MI which is expected, as higher MI use more of the bandwidth, thus allowing for better spectral separation between the bits.
- The system jitter seen in the tests was 0.31m with the MI=0.875. This would be part of the overall range error that will be seen in the later tests.

However, since Delfi-PQ only recognized the GFSK(MI=0.875) modulated commands as discussed in section 6.1.2, it is chosen for the future tests.

Table 6.1: Bias and Jitter for Different Modulations(1 sigma)

Modulation	GMSK	GFSK(MI=0.5)	GFSK(MI=0.875)
Bias (m)	1950	170.14	373.04
Jitter for range measurements (Delay Line - Baseline) (m)	20759	15.22	6.25
Jitter for Baseline measurements (m)	1497992	18.85	0.31
Jitter for Delay Line measurements (m)	224698	16.8	6.26
Number of Measurements	101	101	101

A notable issue in these tests was that the bias measured changed for every initialization of the radio (by approx 60 meters for MI=0.875). However, the change in is only visible in the delay line measurements. The baseline measurements show no such bias. It is hard to pinpoint any particular reason for this, as the only change between the two tests was the delay line being connected. This could be pointing towards an issue with the delay line, but due to limited time, Nevertheless, a suggested way to correct this would be to use another radio to receive the signals a known distance away during ranging with the satellite. By comparing the obtained range value for this second radio, this bias can be calibrated.

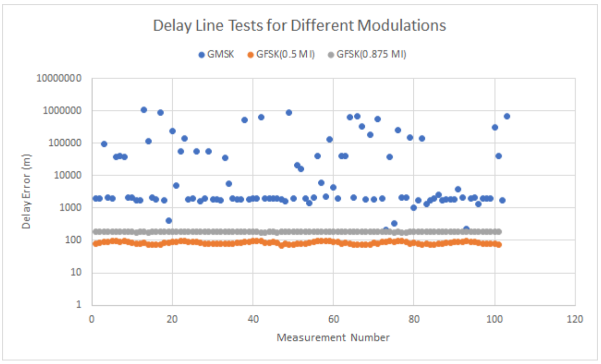


Figure 6.12: Delay Line Tests for Modulations

Therefore, in the final ranging, the procedure on the ground would be:

1. Calibration with self.
2. Calibration with second radio.
3. Final Ranging.

It is also possible to skip step 2 and conduct ranging at the cost of a few meters additional error which was done for the thesis.

6.2. Telemetry Ranging with satellite model

In the previous section, the changed telemetry ranging method was proposed for Delfi-PQ (Figure 6.1). The properties of the GFSK commands for correlation were validated in section 6.1.3. The USRP ground radio was found to give a bias of 373.04 meters for the test and jitter of 6.26 meters (section 6.1.4). This bias seen changes with every time the USRP radio is restarted and needs to be estimated through a separate test as discussed.

However, for this initial test, the jitter and bias will not be visible as they are below the system resolution of 500kHz which was used. This system resolution gives a time resolution of 4 micrometers and a range resolution of $\simeq$1.2km (motivation is given in the following subsections). For this initial test, the time spent onboard the satellite is not measured in any way. The idea is to obtain an estimate of the delays that are introduced by the satellite into the ranging process.

For the satellite, the delay error contribution is expected from two main sources:

1. The uplink clock recovery - A symbol tracking loop is used on the uplink receiver to recover the signal timing (clock) for demodulation, as discussed in section 6.1. This ambiguity or clock error can be $\pm$ half a bit length (1 bit length in total), thus giving a range error of

$$Clock\ Recovery\ Error = c * bit\ length = c * \frac{1}{bitrate} = \frac{1}{bitrate} \tag{6.6}$$

   where c is the speed of light.
2. The onboard processing jitter - the Delfi-PQ communication channel is asynchronous, meaning that command reception and telemetry transmission occur independently of each other. Thus, there may be any number of processes that occur between command reception and marker retransmission, which will all add to the jitter contribution here.

If the tests necessitated resolutions higher than the raw system resolution, which will happen in case the sum of above two errors are lower than the 1.2km range resolution, other methods such as GNU Radio interpolation (section 5.2.5) and quadratic interpolation (section 6.1.3) could have been validated and used. However, the obtained results did not need a higher resolution to interpret.

6.2.1. Test Setup

The test setup is as visible in Figure 6.17. The ground radio transmitter and the satellite receiver are set up with 145 MHz antennas, and the satellite transmitter and the ground receiver are set up with 436 MHz antennas. This is because the uplink transmission is done at different frequencies to prevent interference from each other.

The carrier frequency can affect the delay through the ground instruments. For example, the DSN baseline delay for different carrier wave downlink frequencies is given in Figure 6.13. They show a deviation of 18ns (5.4m) over between 8395 and 8450 MHz. To see if there is any deviation between the baseline delay for the 145MHz uplink signal and 436MHz downlink signal in GNU Radio, a correlation flowgraph was set up with the sequential ranging signal. The baseline calibration was run and recorded at 2MHz system resolution. During the recording, using the GNU Radio QT GUI Chooser widget, the carrier frequency was changed from 145MHz to 435MHz. For the change between 145MHz to 436MHz, there was no change in the baseline delay value upto the system resolution. This indicates that this change is lower than the range resolution of 150m.

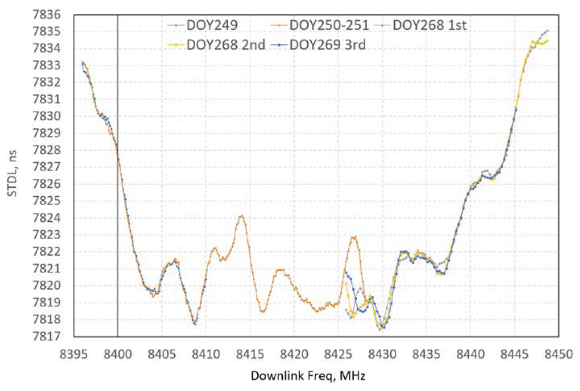
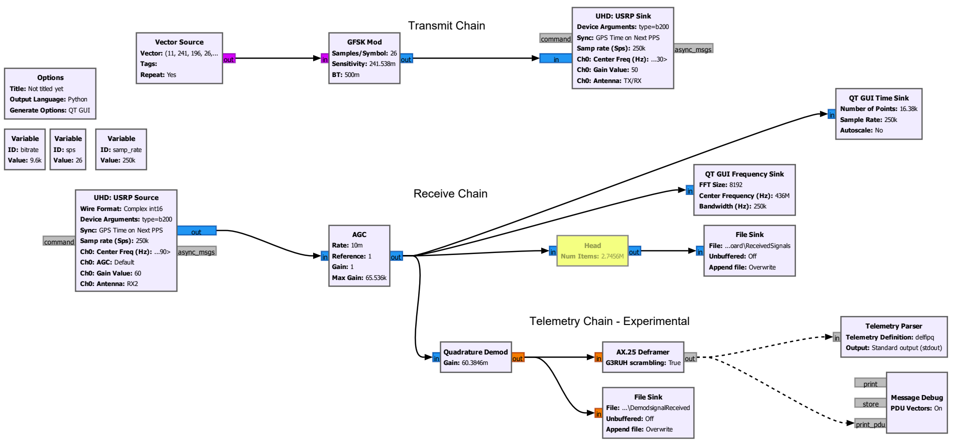


Figure 6.13: DSN baseline delay deviation for downlink frequencies[17]

The satellite test board is set about 30cm away from the radio. It is connected to a power source and to the PC using Universal Serial Bus (USB). From the PC, the satellite code can be modified, and the satellite console can be accessed. Various messages about the satellite's functioning are printed to the console and can be used to monitor the satellite's software operations.

The GNU Radio flowgraph in Figure 6.14 is set up similar to the modulation tests. A FFT graph and signal time graph are set up to see the 436MHz response signal of the satellite, as in Figure 6.15.



Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: bitrate
Value: 9.6k

Variable
ID: sps
Value: 26

Variable
ID: samp_rate
Value: 250k

Vector Source
Vector: (11, 241, 196, 26,...
Tags:
Repeat: Yes

GFSK Mod
Samples/Symbol: 26
Sensitivity: 241.538m
BT: 500m

UHD: USRP Sink
Device Arguments: type=b200
Sync: GPS Time on Next PPS
Samp rate (Sps): 250k
Ch0: Center Freq (Hz): ...30>
Ch0: Gain Value: 50
Ch0: Antenna: TX/RX

UHD: USRP Source
Wire Format: Complex int16
Device Arguments: type=b200
Sync: GPS Time on Next PPS
Samp rate (Sps): 250k
Ch0: Center Freq (Hz): ...90>
Ch0: AGC: Default
Ch0: Gain Value: 60
Ch0: Antenna: RX2

AGC
Rate: 10m
Reference: 1
Gain: 1
Max Gain: 65.536k

QT GUI Time Sink
Number of Points: 16.38k
Sample Rate: 250k
Autoscale: No

QT GUI Frequency Sink
FFT Size: 8192
Center Frequency (Hz): 436M
Bandwidth (Hz): 250k

Head
Num Items: 2.7456M

File Sink
File: ...oard\ReceivedSignals
Unbuffered: Off
Append file: Overwrite

Quadrature Demod
Gain: 60.3846m

AX.25 Deframer
G3RUH scrambling: True

File Sink
File: ...\DemodsignalReceived
Unbuffered: Off
Append file: Overwrite

Telemetry Parser
Telemetry Definition: delfipq
Output: Standard output (stdout)

Message Debug
PDU Vectors: On

Figure 6.14: Telemetry Ranging Test Setup

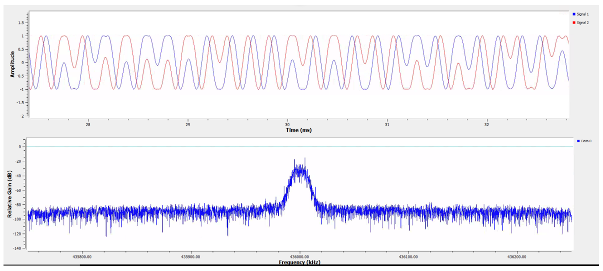
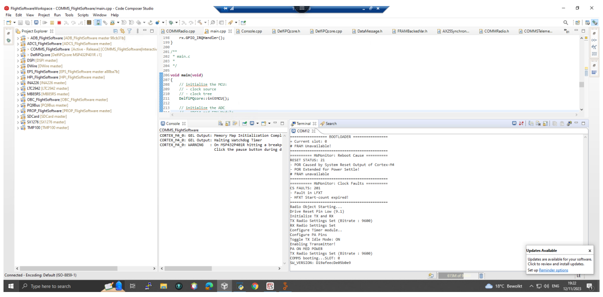


Figure 6.15: GNU Radio Output for Tests, signal and FFT

In the PC, code-composer studio application was opened to interface with the satellite (Figure 6.16). Code composer is an Integrated Development Environment (IDE) made for use with Texas Instruments microcontrollers. Since Delfi PQ runs on a version of the Texas Instruments MSP432 microcontroller (MSP432P401 for the flatsat and MSP432P411 for flight model), code-composer studio is used to update the code of the microcontroller and read the system console logs for troubleshooting. Running the above GNU Radio application, Code-Composer and the USRP at the same time increased the load on the PC's CPU. The maximum raw sample rate it could now handle with minimal errors was 500kHz, so that was used for the tests. This was sufficient to see the satellite side errors for the tests conducted in the thesis. If this is a limit in the future, switching to a more powerful PC, switching to Linux (for better scheduling and support, GNU Radio has limited support for Windows[21]), forgoing the use of GNU Radio features such as the FFT graph and validating the subsample estimators are all valid solutions.



```
    rx.GPIO_IRQHandler();
}

/**
 * main.c
 *
 */

void main(void)
{
    // initialize the MCU:
    // - clock source
    // - clock tree
    DelfiPQcore::initMCU();

    // initialize the ADC
```

```
COMMS_FlightSoftware
CORTEX_M4_0: GEL Output: Memory Map Initialization Compl
CORTEX_M4_0: GEL Output: Halting Watchdog Timer
CORTEX_M4_0: WARNING   : On MSP432P401R hitting a breakp
                         Click the pause button during d
```

```
================= BOOTLOADER ================
= Current slot: 0
# FRAM Unavailable!
=============================================
========== HWMonitor: Reboot Cause ==========
RESET STATUS: 21
- POR Caused by System Reset Output of Cortex-M4
- POR Extended for Power Settle!
# FRAM unavailable
=============================================
========== HWMonitor: Clock Faults ==========
CS FAULTS: 201
- Fault in LFXT
- HFXT Start-count expired!
=============================================
Radio Object Starting...
Drive Reset Pin Low (9.1)
Initialize TX and RX
TX Radio Settings Set (Bitrate : 9600)
RX Radio Settings Set
Configure Timer module..
Configure PA Pins
Toggle TX Idle Mode: ON
Enabling Transmitter!
PA ON MED POWER
TX Radio Settings Set (Bitrate : 9600)
COMMS booting...SLOT: 0
SW_VERSION: D19afeecDe05b0e9
```

Figure 6.16: Console in Code Composer Studio

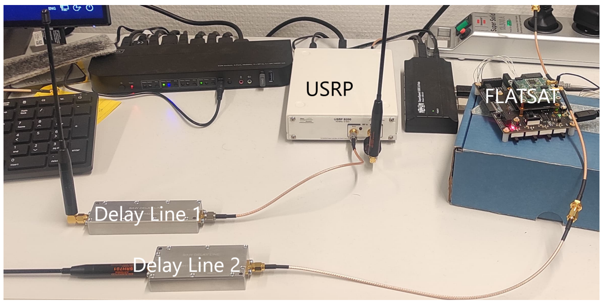


Figure 6.17: Ranging Equipment for Tests

Satellite Side Processing

When a byte (8 bits) of data is received by the satellite, it is gathered and sent through a serial to parallel interface (SPI) to the satellite by the modem using a system interrupt. This interrupt stores the data into a byte queue. A regular task service is called at repeated intervals to process the bytes inside the byte queue.

When a command is sent and the pilot sequence of 7Es are fully received, the first byte of the header sequence is encountered. Upon detecting a byte that is not a hexadecimal 7E, the program raises a flag that a signal may be present. Based on the predefined structure of the signal, the satellite checks the bytes for the presence of the full header, data and CRC (for command structure see Figure 6.2). When the whole command is detected, the corresponding function of the command is called and a response packet is queued and sent to the modem using another interrupt.

The Delfi-PQ communication system is based on the Semtech SX1278- LoRa modem, which supports a fixed number of data rates and receiver bandwidth. For Delfi-PQ's design, the satellite receiver channel filter bandwidth was set higher than twice the bitrate. This channel filter rejects noise and interference arising from outside the specified channel[61]. The series of data rates and bandwidths of the satellite used in the tests are given in Table 6.2. The lower frequency channel filter bandwidths (Snos 1, 2 and 3) were fixed by Delfi PQ's design. The higher frequency bandwidths were chosen to be the lowest available option working with the higher bitrates. Any bandwidths lower than these reduced probability of command recognition.

Table 6.2: Bitrate Setting available on Delfi-PQ[61]

SNo	BitRate of Satellite (bps)	Channel Filter Bandwidth used in tests (Hz)
1	1200	10400
2	2400	10400
3	4800	10400
4	9600	25000
5	19200	50000
7	38400	100000
8	76800	166700
9	153600	Not tested

From these settings, Delfi-PQ can only run upto 9600bps when in flight because it does not have a high-powered amplifier needed for the high data rate signals. Since the flatsat is close to the radio, a higher power amplifier is not required to test the higher bit rates. When the bitrate reaches 76800bps, this overwhelms the size of the byte queue used to store the incoming data bits onboard the satellite. This is because the flatsat has a smaller queue due to lesser memory compared to the actual satellite and also due to code never needed to be optimised for the higher bit rates because the satellite does not reach these bitrates in flight. Overall, the microcontroller is not fast enough to process the bytes beyond this data rate.

The flatsat satellite engineering model has limited performance and memory. The max CPU load reached its limits during the experiments, which presented itself in the form of artifacts in the satellite telemetry stream. One such artifact was that the marker (Attached Synchronization Marker (ASM), described in the following subsections) defined in the satellite code was significantly longer than defined in the code. When the processing burden was alleviated by removing additional debug print statements, the artifacts disappeared.

Post Processing on Ground

In the tests, the prepared command sequence was sent to the flatsat engineering model and its response markers were recorded using GNU radio as given in Figure 6.14. The received signals were then demodulated using the quadrature demod block and filtered using a low pass filter in GNU radio. These demodulated signals are stored as binary files in the PC.

A python script was made which opened these binary files using NumPy's fromfile function (code in Listing A.5). This script scans the filtered and demodulated reception stream containing the telemetry data and using another copy of the marker made in GNU Radio(using vector source block), correlates for the marker sequence. This gives the correlation matrix containing the locations of the response ASMs visible as peaks(see Figure 6.18).

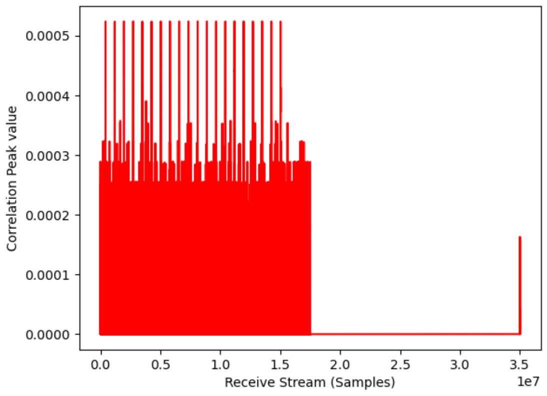


Figure 6.18: Correlation Peaks visible for correlation of ASM and Reception Stream

The start location of the sent commands are known because the signals are of known length. Thus, the start times of the commands are determined mathematically in terms of the number of samples.

Following this, the code slices the matrix into small chunks and checks each chunk for the highest peak using the NumPy argmax function. A minimum cut off threshold is set (0.0004 for Figure 6.18) beyond which a correlation peak is accepted. This cutoff value varies with USRP sample rate and satellite bitrate, so is visually determined from the graph every time parameters are changed. Now, it is also possible that the code finds the same peak in two consecutive slices, as each peak is not perfectly sharp and has a number of points close to the maximum value. Thus, another variable (delay peak breaker) is defined which is set to the length of a peak, whose value is determined visually. When two consecutive peak values fall within the distance of delay peak breaker, it means that the same peak has been detected twice, and only the higher value will be retained. By the end of this loop, the location of all the commands in the reception stream are identified.

Finally, it is necessary to associate each start location with the end location. Luckily, there exists only one approximate delay value for which all sent signal locations align with the received signal locations. To determine this and to reject a sent signal location if the corresponding received signal was dropped, a maximum delay value is set. When the delay value is higher than this max delay, which happens when the receiver stream misses a command, the corresponding sent signal location is dropped from the list. To determine the max delay value, the approximate delay that is correct can be estimated by looking at the sentsignal and received signal locations. In the code, the max delay variable can be set to a very high value. Running the code with this will show the correct delay for some of the peaks, which can then be used to set the max delay value. In case there is prior knowledge of the approximate range, that can also be used here. In case this max delay is wrong, it will be clearly visible when comparing the correlation graph and in the range values obtained.

After all these operations, an array of range values are obtained for the ranging pass. The jitter in these values is the instrument jitter from the satellite.

6.2.2. Test Results

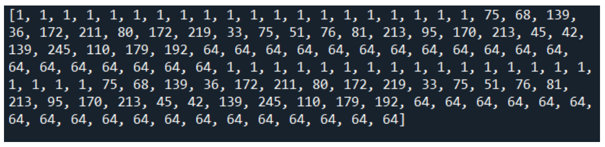
The obtained performance for the telemetry ranging was found to be dependent on the bitrate and on the internal processing load of the system. This is due to the clock recovery and processing jitter on the spacecraft, as discussed in section 6.2.

The results of the tests are visible in Figure 6.22. The testing was conducted in 3 phases in the interest of incrementally raising the complexity of the ranging system:

1. Command with encoding and differential encoding violation: As discussed in section 6.1.1, Delfi-PQ uses differentially encoded and scrambled command codewords. Initially a single command was tested for ranging without radio scrambling and encoding enabled. Once the satellite re-

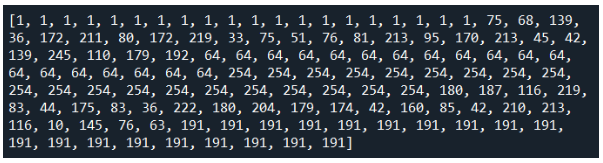
ceived this command, it sent back the same command, with a different header, as a response. On the ground, the header was correlated for, and the total ranging time was calculated. Once this procedure was validated to work, the system was expanded to handle a number of commands as in section 6.2.1. In this phase of testing, a single command codeword was repeatedly transmitted in GNU Radio which cause a violation of differential encoding scheme between consecutive commands. With this violation, the performance of the system was worse than the performance with proper differentially encoded codewords. The command recognition was also worse with the satellite missing commands more often than in the other cases.

2. Command with differential AX.25 encoding: In this phase, the testing procedure was the same as in the previous case, with the satellite response header being correlated for. However, to further ensure that differential encoding was not violated between commands, a single command was no longer repeated using GNU Radio. All the commands to be sent to the satellite were fed to the encoder one after another to obtain a series of differentially encoded commands. The series of properly encoded commands were then transmitted using GNU Radio. The differential encoding violation can be seen in Figure 6.19 **(a)** where the middle of the sequence repeats a 1 after the 64 instead of inverting (giving 255-1=254).



```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 75, 68, 139,
36, 172, 211, 80, 172, 219, 33, 75, 51, 76, 81, 213, 95, 170, 213, 45, 42,
139, 245, 110, 179, 192, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64,
64, 64, 64, 64, 64, 64, 64, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 75, 68, 139, 36, 172, 211, 80, 172, 219, 33, 75, 51, 76, 81,
213, 95, 170, 213, 45, 42, 139, 245, 110, 179, 192, 64, 64, 64, 64, 64, 64,
64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64]
```

(a)



```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 75, 68, 139,
36, 172, 211, 80, 172, 219, 33, 75, 51, 76, 81, 213, 95, 170, 213, 45, 42,
139, 245, 110, 179, 192, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64, 64,
64, 64, 64, 64, 64, 64, 64, 254, 254, 254, 254, 254, 254, 254, 254, 254,
254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 180, 187, 116, 219,
83, 44, 175, 83, 36, 222, 180, 204, 179, 174, 42, 160, 85, 42, 210, 213,
116, 10, 145, 76, 63, 191, 191, 191, 191, 191, 191, 191, 191, 191, 191,
191, 191, 191, 191, 191, 191, 191, 191, 191]
```

(b)

Figure 6.19: Two commands **(a)** not following differential encoding and **(b)** following differential encoding

3. Command with both AX.25 encoding and G3ruh scrambling: Once the ranging process was validating with encoding, the scrambler was enabled. This added a complication in that the scrambler changed the satellite's retransmitted sequence to be different at every point in time. This also changed the command header which was sent by the satellite, meaning that unless this change could be determined, the header could not be used as a marker for the ranging. Theoretically, the state (value of latest bit) of the scrambler depends on the previous 17 bits. Due to this 17 bit level of complexity, the scrambler state evolves very rapidly. It could have been possible to predict this change, but a simpler solution was available at hand.

   An ASM is used in telemetry processing for frame synchronization or detection of when a frame starts or end. They are not encoded or scrambled[57] and have a high correlation peak as seen in Figure 6.20 to allow for precise frame timing determination. Delfi-PQ does not use an ASM at all, but can be configured to add one in its data stream. The satellite was configured to send an unencoded and unscrabled ASM behind the response command in the reception stream. Using this has the added benefit that the ASM could also just be attached behind a telemetry packet in response to a command, thus making it possible to conduct telemetry ranging without having to send an entire command back, contributing to SREQ-005 (section 3.2).

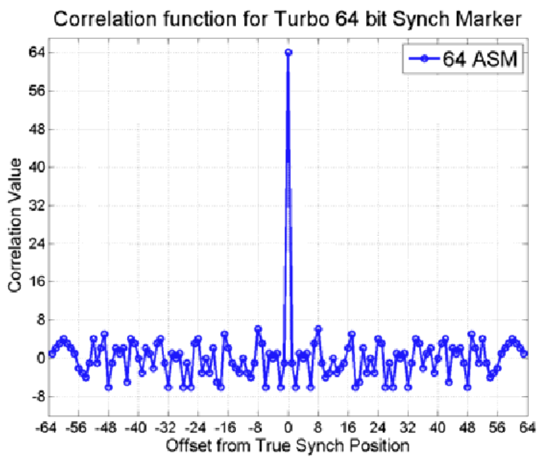


Figure 6.20: Typical ASM correlation peak[51]

Adding an ASM to the satellite was straight forward ((code in Listing A.3)). Whenever a command was detected on the satellite, a boolean variable was triggered which placed 4 ’7E’ bytes in the satellite transmit queue along with a 4 byte ASM marker. Correlating for this preset sequence, the arrival timing of the response command packet(or telemetry packet in the future) can be determined. A picture of the attached synchronization marker in the data stream is given in Figure 6.21, where the orange line represents the received data and the blue line is the copy of the ASM which was correlated to find the orange copy.

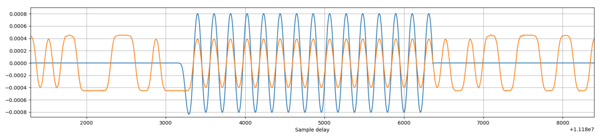


Figure 6.21: Attached Synchronization Marker in the data stream

The CCSDS ASM[57] was attempted to be used due to higher expected correlation performance, but did not work on the satellite. Attempting to use the codeword made the satellite generate a wrong marker. The use of symmetric ASM codes seemed to work with the satellite while asymmetric codes did not. However, due to time concerns, the work proceeded with the next steps and there were not enough tests to conclusively determine that this is the case. A few other ASM markers were checked, and the final marker was determined to be 4 bytes of 85 (which is also the marker in Figure 6.21), due to its known structure allowing for easier manual inspection to verify that the results obtained were accurate. This also gave sufficient correlation performance for testing purposes.

However, there are performance benefits in using a sequence having high auto-correlation and low cross-correlation coefficients, such as the CCSDS ASM in low noise conditions, and it is suggested that the problem be investigated further in the future if low noise operating conditions are expected.

From the results in Figure 6.22 and Table 6.3, the following observations can be made:

1. The bitrate is one of the main drivers of performance, with a twofold increase in bitrate almost doubling the performance in all the cases.
2. When the ASM is added, the range performance worsens by 39% for the data rate of 19200bps (and about 30% for all except 9600bps). This is likely due to the additionally processing that the satellite now does to add the ASM to the bit stream. This is supported by the fact that the only unknown in the error budget that could be this large is the satellite processing jitter (see Table 6.4 below)

3. Proper configuration of codewords, such as using the correct differential encoding scheme has a minor effect on the ranging performance.

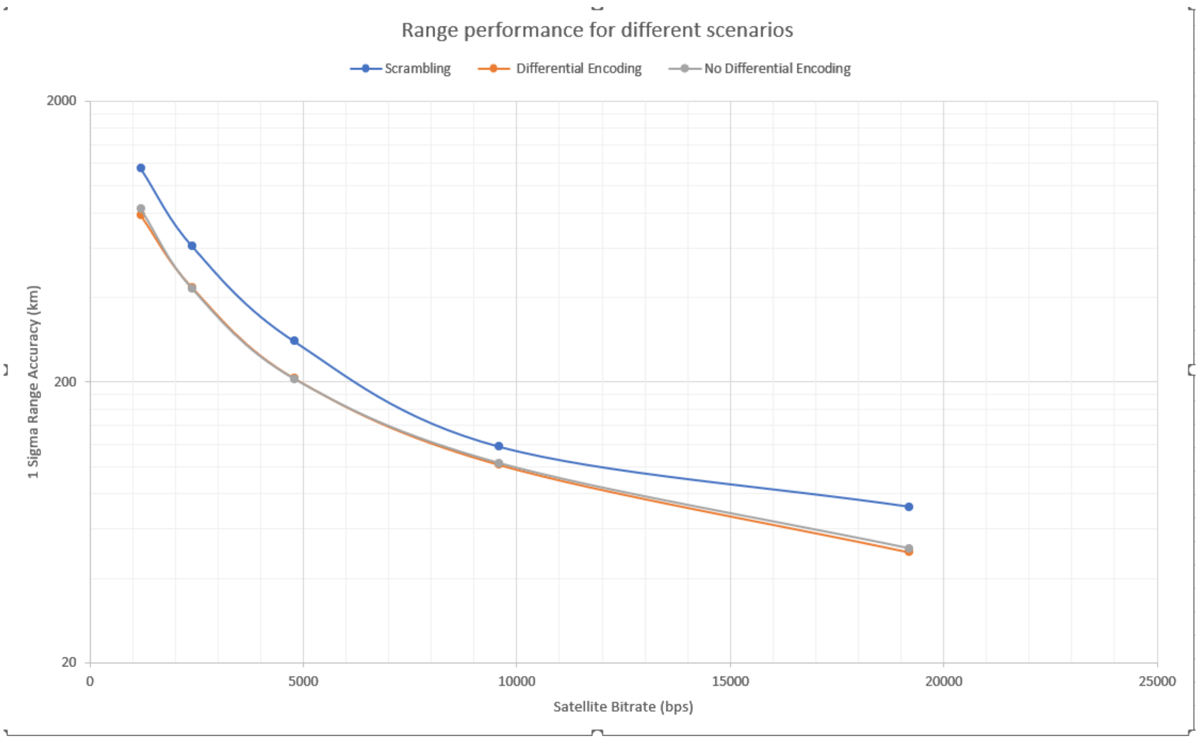


Figure 6.22: 1σ Range accuracy vs Satellite Bitrate

Table 6.3: Telemetry Ranging without Time Measurement Results (1σ accuracy

Bitrate (bps)	19200	9600	4800	2400	1200
No Differential Encoding (km)	51.02	102.75	205.01	430.76	825.47
Measurement Count	564	283	146	71	35
Differential Encoding (km)	49.43	101.30	205.69	434.04	782.79
Measurement Count	97	99	100	71	35
Encoding and Scrambling (km)	71.90	117.53	278.45	608.93	1149.62
Measurement Count	192	195	196	67	67

The usable results for Delfi-PQ will be those of the scenario with encoding and scrambling switched on, as that is the scenario of normal operation. The range accuracy attainable in this case is tens of kilometers as seen in the table. The system requirement of better than 1km range accuracy (SREQ-001, section 3.2) cannot be met with this solution.

At this point, it is interesting to look at the error budget for the ranging method in Table 6.4. The different contributions from the system are mapped out and only the satellite processing jitter is still unknown. Therefore, the next step would be to measure and mitigate this error. In order to do so, it was decided to use the Texas Instruments MSP432 microcontroller board's hardware timers to measure the time between reception of a command and transmission of an ASM.

Table 6.4: Error Budget for Telemetry Ranging

Error Source	Expected 1σ Contribution	Source
Ground System Correlation Jitter	0.31 m	section 6.1.4
Ground System Correlation Bias	6.25 m	section 6.1.4
Ground Bias due to Carrier Frequency	<150m	section 6.2.1
Round off Errors due to Sample Rate Resolution of 1.2km	$\leq$1.2 km	Sequential Ranging Tests (section 5.3.2)
Satellite Clock Recovery	$\simeq speed\ of\ light(c)/(3 * bitrate)$	section 6.1, divided by three to get 1σ
Satellite Processing Jitter	Unknown and highly variable	-

6.3. Telemetry Ranging with Time Measurement

Section 6.2 developed the telemetry ranging for Delfi PQ where a GFSK modulated command was sent to the satellite on the uplink. On receiving the command, the satellite set a ASM behind a data packet on the downlink, which was correlated for to determine the overall time of flight. The range performance was obtained in the order of hundreds of kilometers as seen in Table 6.3.

The only possible option which will not affect the other functions of the satellite (to meet SREQ-005) is to measure the time elapsed on board the satellite from the time a command was received to the time a response was sent (processing time). This will then be sent to the ground and used to correct the processing jitter in the range measurements. For this purpose, the inbuilt hardware timer available onboard the MSP432 microprocessor of Delfi-PQ was used.

The operation of the method is discussed in section 6.3.1. Initial tests revealed the presence of ”bit-shifts” (explained in more detail below) in the obtained range measurements, whose cause could not be determined. Therefore, a method based on the expected statistical distribution of the range errors was created to correct for the bit shift. The entire method was validated by conducting ranging with delay lines in section 6.3.2. Finally, the obtained performance of the method is discussed in 6.3.3.

6.3.1. Working

To measure the processing delay in the satellite, one of the two hardware timers present in the satellite were adapted. This timer, the Texas Instruments MSP432 Timer32 module, adopts the resolution of the system crystal oscillator, which runs at 48MHz[5]. This corresponds to a time tagging resolution of 20.83ns or 6.25meters (see section C.1.3). Since the system measures the time twice, one for reception time of command and once for transmission time of marker, time tag errors of upto 12.5meters will always be present in this system. This time tag error is the reason why telecommand ranging[75] measures the onboard time delay in terms of the incoming bit’s phase. By measuring the system delay in terms of an external signal whose phase can be estimated more accurately than the onboard time, the time tagging error is eliminated.

The timer’s 32BIT resolution allows it to count upto $2^{32}-1$ bits or 4294967295, which takes around 90 seconds at 48MHz. The timer counts in reverse, starting at 4294967295 and counting down. Once the timer reaches 0, it resets its value to 4294967295 and continues counting. Therefore, when measuring the time onboard, some measurements will give a wrong value due to this timer reset. In this case, just adding 4294967295 (assuming measured time is $command\ timer\ time$ - $marker\ timer\ time$, if not subtract instead) to the measured time will correct for the reset.

Measuring Time on Board

A simplified flowgraph is presented in Figure 6.23. The earliest point which can be measured without major changes to the satellite code is the interrupt generated when the modem SPI (SPI is described in section 6.1) sends the received data and commands as bytes(8 bits) to the system. The received bytes are then placed in a queue until the satellite work functions check for the presence of bytes in the queue. In order to ensure the right timing is used, a timing queue of the same length is made which associates each time measurement with the corresponding byte.

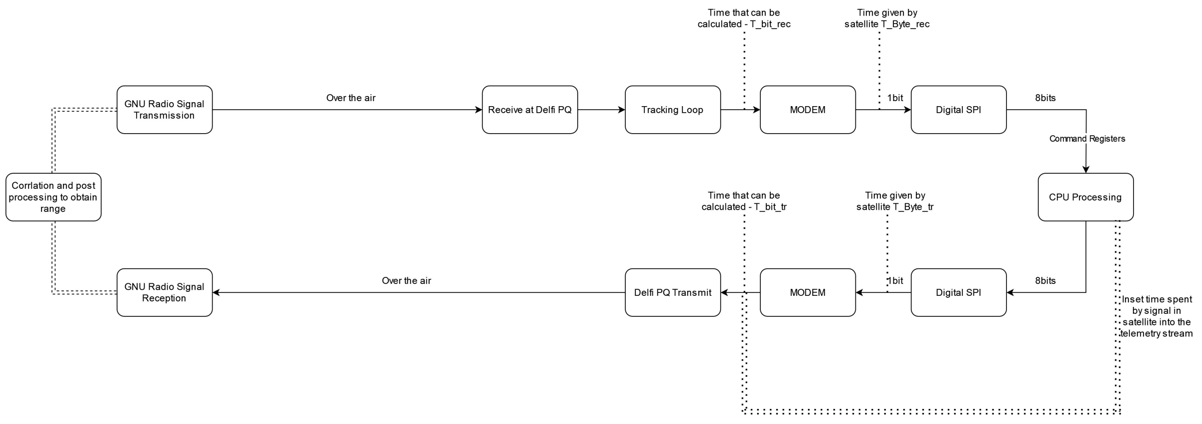

Time that can be calculated - T_bit_rec
Time given by satellite T_Byte_rec
GNU Radio Signal Transmission
Over the air
Receive at Delfi PQ
Tracking Loop
MODEM
1bit
Digital SPI
8bits
Command Registers
CPU Processing
Corrlation and post processing to obtain range
Time that can be calculated - T_bit_tr
Time given by satellite T_Byte_tr
GNU Radio Signal Reception
Over the air
Delfi PQ Transmit
MODEM
1bit
Digital SPI
8bits
Inset time spent by signal in satellite into the telemetry stream

Figure 6.23: Simplified Timing Measurement flowgraph for Telemetry Ranging

When a command is recognized by the satellite, the reception time of the byte containing the command on the microcontroller is retrieved and recorded. This time here is known as T_byte_rec and its unit is in ticks of the system crystal oscillator (which operates at 48MHz). However, it is necessary to correct the time to obtain the arrival time of each bit at the modem, which is T_bit_rec. Since the modem processes bits at the bitrate speed, the time it takes for the modem to process a bit (in terms of ticks) is given by

$$Processing\ Time = \frac{Timer\_Res}{Bitrate} \tag{6.7}$$

where $Timer\_Res$ is the resolution of the timer, which in this case is the resolution of the crystal oscillator onboard the satellite (48MHz) and Bitrate is the bitrate of the satellite.

The byte contains each bit in the order that it came into the modem, starting with Most Significant Bit (MSB) first. Once the command is recognized, the position of its first bit in the byte can be determined. This position is called the Bit_Index. With the Bit_Index and modem processing time known, a correction can be applied to find the incoming time of the bit into the satellite system as

$$T\_bit\_rec = T\_byte\_rec + (7 - Bit\_Index) \times \frac{Timer\_Res}{Bitrate} \tag{6.8}$$

Figure 6.24 represents the value of a bit index when a command is present in between a new byte.

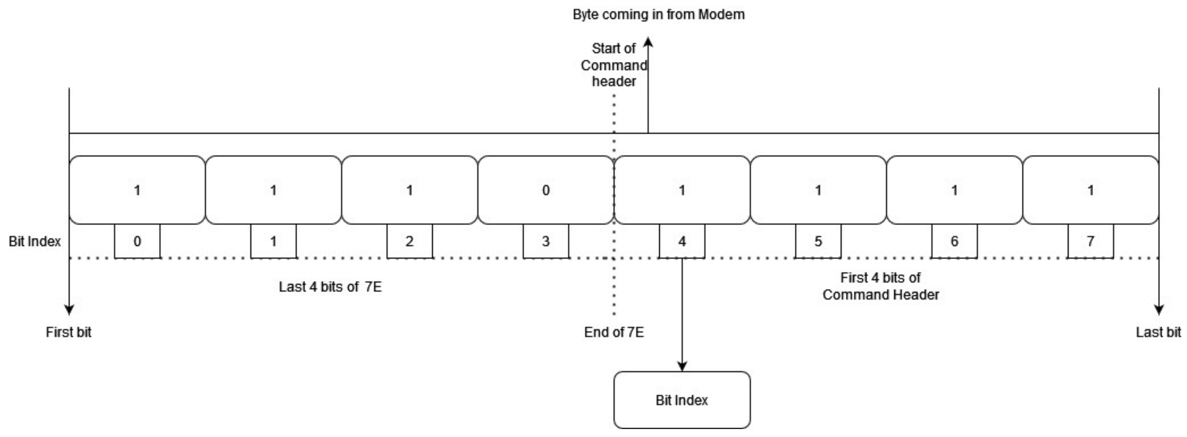


Figure 6.24: Bit Index of command onboard satellite

Similarly, the transmission time indicated by T_byte_tr is recorded on the microcontroller at the start of the interrupt which is called to send each byte to the SPI for retransmission. Based on where the ASM's first bit lies in the byte, the transmission bit index is calculated. The correction for the time each bit will spend in the modem before transmission is applied as

$$T\_bit\_tr = T\_byte\_tr + (Bit\_Index) \times \frac{Timer\_Res}{Bitrate} \tag{6.9}$$

where all other variables are the same as the previous equation. Notably, the bitindex correction is directly the value of the variable instead of being subtracted from seven. The makes sense when thinking about the order of reception and transmission of bits in the modem. The timer time is given in terms of ticks of the system oscillator. The translation to seconds is given as

$$Time\_on\_board(s) = \frac{T\_bit\_rec - T\_bit\_tr}{Timer\_Res} \tag{6.10}$$

with $Timer\_Res = 48MHz$ for Delfi-PQ.
Once the time measurement was available, the corrected range is

$$Corrected\ Range = Correlation\ Range - Time\_on\_board * c \tag{6.11}$$

where c is the speed of light.
This corrected range will not contain the onboard processing jitter errors in it.

Sending time to ground

Next is the question of how the time measured on board the satellite can be received on the ground. For the purpose of the tests, the final time was stored on board the satellite. Once the transmission of a command was completed, this value was printed to the flatsat console using the telemetry debug function, which is called every second when enabled. The value was not printed immediately after recording to ensure that the range measurement was not perturbed by the computational load of printing to console, which is significant for the pocketcube.

Since the telemetry debug function was set to be called every second, only one command was sent every second because the system was configured to only record one onboard time measurement. This was partially due to the limited memory and partially because it was unnecessary complexity to enable multiple time measurements to be taken and printed without any print statements being called in between the processing of any of them.

The print statement was controlled using flag variables. These flags were switched on after the transmission of an ASM and switched off after the onboard time measured for the latest command was printed, which happened within approximately one second.

When transferring the time in flight however, there are two options:

1. The timing information can be sent through the telemetry as in Figure 6.23 (for detailed plan see Figure B.2). These can then be decoded separately and post-processing can be done to obtain the range measurements.
2. Alternatively, for simplicity, the data can just be attached behind the ASM in the telemetry stream even without encoding. Since the ASM's location and hence the time measurement's location is known through correlation, it can be immediately extracted. An implementation of this was attempted and the code is given in Listing A.5. Due to limited time and since extracting the data is not an unsolved problem (it is known to be possible, just needs to be implemented) the focus was placed on validating the ranging technique. Thus, this is included as a future suggestion.

Bit Shifts

When the timing measurements were made on board the satellite and used to correct the range measurements, it was noted that some range measurements were shifted in value corresponding to an integer number of bits. Since unusual artifacts were previously encountered when the satellite was under too much cpu load, this could possibly be indicating the same. At higher bitrates, when the cpu load is higher, the range was found to be more prone to bit shifts. The explanation for this would be that the satellite processor is dropping cycles more often under higher loads.

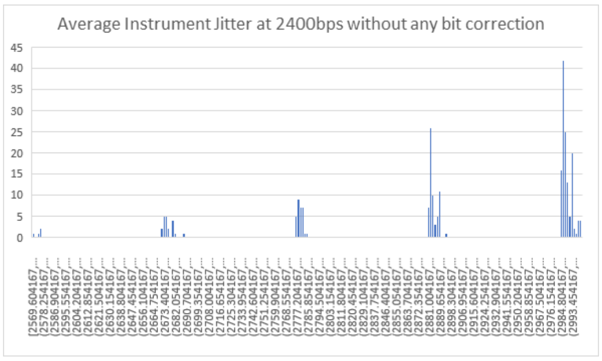
This phenomenon needs to be explored further if using the same technique in the future. An initial step would be to test with a different satellite or radio system to see if the same issue is experienced. This can be done using another USRP connected to a computer. It is also possible (though unlikely) that including the ASM without encoding causes issues with the satellite system.

However, due to time constraints near the end of the thesis, this approach was not feasible, and an alternative approach was considered to correct these bit shifts. According to the histogram in Figure 6.25 **(a)**, the measurements are clearly segregated at different bit offsets. The higher bit value bit offsets are rarer too.

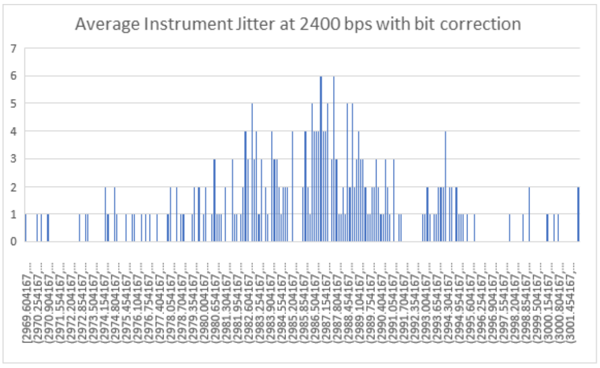
The bit offset values for the graph were calculated as a function of bitrate as

$$bitoffset\ correction = \frac{sample\ rate}{bitrate} = \frac{250000}{2400} = 104.167 samples/bit \tag{6.12}$$

Using this, the bit offsets were corrected and **(b)** was obtained, which resembles a Gaussian distribution. Since the underlying errors of the symbol tracking loop clock recovery and other instruments (due to thermal noise) are expected to be Gaussian, the correction for the bit offsets should give a valid ranging result. However, in Figure 6.25, a few measurements having higher jitter were removed from **(b)** as they were thought to be erroneous based on their high bit shift indices. However, later investigations revealed that there is no correlation between a higher bit shift index and corrected jitter but the data was already cleaned and the information was lost. So **(b)**'s jitter in graph is misleading, the actual jitter is higher and is presented in later sections.



(a)

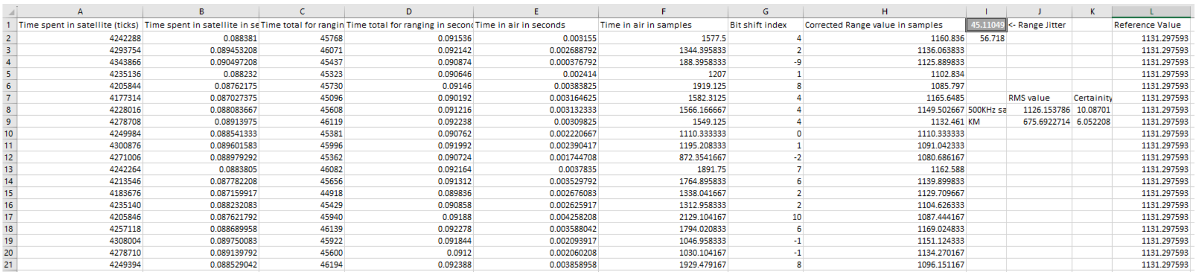


(b)

Figure 6.25: Delfi PQ timer corrected Instrument Jitter **(a)**without bit correction and **(b)**with bit correction

This graph's x axis are **(a)** [2569,3003] and **(b)** [2969,3003]. The x axis could not be corrected to be more visible without losing the granularity of the graph which shows the bit-shifts.

The correction setup for one set of measurements is given in Figure 6.26. Column A has the onboard time measurement of the satellite in ticks of the crystal oscillator. This is then converted into seconds in column B. Column C has the range measurement in samples, which is converted into the corresponding time of flight (seconds) in column D. Then column E obtains the corrected time of flight as the difference between column D (range time in sec) and column B (onboard time in sec). This corrected time of flight will have most of the processing delay from the satellite eliminated but will still have the bit shift errors. Column F translates this time of flight measurement back into range samples for easy interpretation.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Time spent in satellite (ticks)	Time spent in satellite in se	Time total for rangin	Time total for ranging in seconc	Time in air in seconds	Time in air in samples	Bit shift index	Corrected Range value in samples	45.11049	<- Range Jitter		Reference Value
2	4242288	0.088381	45768	0.091536	0.003155	1577.5	4	1160.836	56.718			1131.297593
3	4293754	0.089453208	46071	0.092142	0.002688792	1344.395833	2	1136.063833				1131.297593
4	4343866	0.090497208	45437	0.090874	0.000376792	188.3958333	-9	1125.889833				1131.297593
5	4235136	0.088232	45323	0.090646	0.002414	1207	1	1102.834				1131.297593
6	4205844	0.08762175	45730	0.09146	0.00383825	1919.125	8	1085.797				1131.297593
7	4177314	0.087027375	45096	0.090192	0.003164625	1582.3125	4	1165.6485		RMS value	Certainity	1131.297593
8	4228016	0.088083667	45608	0.091216	0.003132333	1566.166667	4	1149.502667	500KHz sa	1126.153786	10.08701	1131.297593
9	4278708	0.08913975	46119	0.092238	0.00309825	1549.125	4	1132.461	KM	675.6922714	6.052208	1131.297593
10	4249984	0.088541333	45381	0.090762	0.002220667	1110.333333	0	1110.333333				1131.297593
11	4300876	0.089601583	45996	0.091992	0.002390417	1195.208333	1	1091.042333				1131.297593
12	4271006	0.088979292	45362	0.090724	0.001744708	872.3541667	-2	1080.686167				1131.297593
13	4242264	0.0883805	46082	0.092164	0.0037835	1891.75	7	1162.588				1131.297593
14	4213546	0.087782208	45656	0.091312	0.003529792	1764.895833	6	1139.899833				1131.297593
15	4183676	0.087159917	44918	0.089836	0.002676083	1338.041667	2	1129.709667				1131.297593
16	4235140	0.088232083	45429	0.090858	0.002625917	1312.958333	2	1104.626333				1131.297593
17	4205846	0.087621792	45940	0.09188	0.004258208	2129.104167	10	1087.444167				1131.297593
18	4257118	0.088689958	46139	0.092278	0.003588042	1794.020833	6	1169.024833				1131.297593
19	4308004	0.089750083	45922	0.091844	0.002093917	1046.958333	-1	1151.124333				1131.297593
20	4278710	0.089139792	45600	0.0912	0.002060208	1030.104167	-1	1134.270167				1131.297593
21	4249394	0.088529042	46194	0.092388	0.003858958	1929.479167	8	1096.151167				1131.297593

Figure 6.26: Excel table used for bit correction

The column F measurements still have the bit shift errors, and they must be corrected. The bit shift correction process consists of choosing a reference value to which the bit shifts will be corrected to. When there are a sufficient number of measurements, this can be chosen as the value that the majority of measurements fall in (the biggest peak in Figure 6.25 **(a)**).

However, the majority peak value is not clear when there are less than 100 measurements. This was the case in the tests. Since the right value of the instrument delay is not known, the correction will be first done to an arbitrary reference value. The integer count of bit shift from the reference value is calculated as

$$Bit\_shift\_index = ROUND((Ref\ value - Current\ value)/(bitoffset\ correction), 0) \tag{6.13}$$

in Microsoft Excel, where the Ref_value is given in Figure 6.26 column L and $Current\_value$ is given in column F. This function gives the integer bit value by which the data should be shifted for correction in Column G. The shift is then applied to the data as

$$Bit\_shift = Bit\_shift\_index \times bitoffset\ correction \tag{6.14}$$

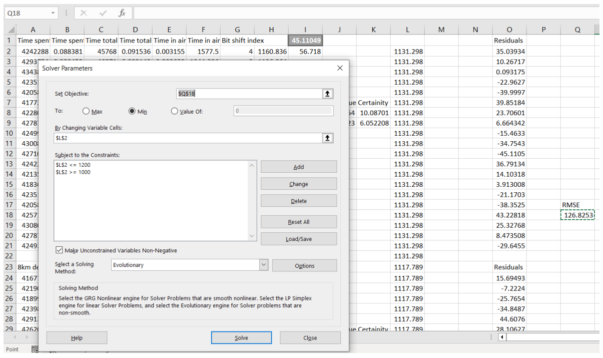
The bit shift is then used to correct the range as

$$Bit\ Corrected\ Range\ value = Uncorrected\ Range - Bit\_shift \times bitoffset\ correction \tag{6.15}$$

This gives the corrected range estimate in column H, and the standard deviation of this corrected range is the range jitter.

After correcting for the bit shift, the values at the very ends may not affect the jitter measured significantly if shifted by an additional bit. Different references values will give different positions for these end data points and will affect the jitter and average value of the corrected range measurement. Therefore, the 'true' value of the reference was defined as the one minimizing the RMSE for all the measurements. This is because when correcting for errors to make the final distribution of data Gaussian, the optimal solution is defined to be the set of parameter values that minimizes the weighted sum of squares of residuals[69]. Since the errors in each of the measurements are uncorrelated, they have equal weights and the weighted sum of squares of residuals is given by the RMSE.

Finding the optimal value for the reference was done through evolutionary optimization using the Excel solver add-in. In Figure 6.27, the configuration of the solver add in can be seen. The reference value was given as the variable to change, the root mean squared error of the measurements was given as the objective to minimize and the bounds for the optimization were set as [Initial Reference $\pm$ bitoffset correction]. This is because the optimization design space repeats after one bit, as shifting all the measurements by the same constant number will not change the distribution between the numbers but shifting only some of them will.


Q18

	A	B	C	D	E	F	G	H	I	J	K	L	O	Q
1	Time spent	Time spent	Time total	Time total	Time in air	Time in air	Bit shift index		45.11049				Residuals	
2	4242288	0.088381	45768	0.091536	0.003155	1577.5	4	1160.836	56.718			1131.298	35.03934	
3	4293											1131.298	10.26717	
4	4343											1131.298	0.093175	
5	4235											1131.298	-22.9627	
6	4205											1131.298	-39.9997	
7	4177										ue Certainity	1131.298	39.85184	
8	4228									54	10.08701	1131.298	23.70601	
9	4278									23	6.052208	1131.298	6.664342	
10	4249											1131.298	-15.4633	
11	4300											1131.298	-34.7543	
12	4271											1131.298	-45.1105	
13	4242											1131.298	36.79134	
14	4213											1131.298	14.10318	
15	4183											1131.298	3.913008	
16	4235											1131.298	-21.1703	
17	4205											1131.298	-38.3525	RMSE
18	4257											1131.298	43.22818	126.8253
19	4308											1131.298	25.32768	
20	4278											1131.298	8.473508	
21	4249											1131.298	-29.6455	
22												1131.298		
23	8km de											1117.789	Residuals	
24	4167											1117.789	15.69493	
25	4219											1117.789	-7.2224	
26	4189											1117.789	-25.7654	
27	4239											1117.789	-34.8487	
28	4291											1117.789	44.6076	
29	4262										ue Certainity	1117.789	28.10627	

Solver Parameters

Set Objective: Q18

To: Max [✓ Min] Value Of: 0

By Changing Variable Cells: L2

Subject to the Constraints:
L2 <= 1200
L2 >= 1000

[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Evolutionary

Solving Method
Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Figure 6.27: Excel Solver Add in used to minimize the RMSE

To validate that the bit-shift corrected and optimized data is indeed representing the range, a range test was conducted using delay lines. This bit shift correction algorithm brings the distribution of any set of data within a bit (or any other specified value). However, this corrected data may not hold any physical meaning. Therefore, the values of the corrections applied should be carefully calculated to be logically sound. Otherwise, any errors are only visible in the ranging test.

6.3.2. Validation with Delay Line

In order to validate the results obtained, ranging tests were conducted with and without delay lines inducing [33590$\pm$50] ns of time delay corresponding to [10.07km$\pm$0.015km] of range delay in the system. The delay lines connected were the Microsaw MH1159-225.000-2 delay line ([13600$\pm$20]ns same as previous USRP tests) for the uplink 145MHz signal and Microsaw MH1110-452.600-2 ([19990$\pm$30]ns for the downlink 436MHz signal. These delay lines were used because they were available with the radio testing workshop at TU Delft and were set up as in Figure 6.29

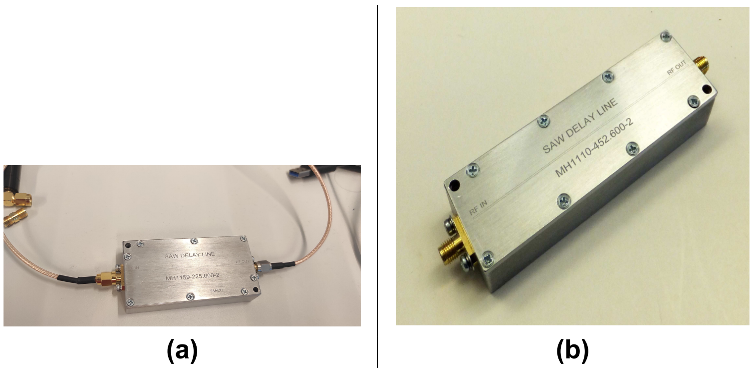


Figure 6.28: Microsaw Delay line **(a)**MH1159-225.000-2 and **(a)** MH1110-452.600-2[45]

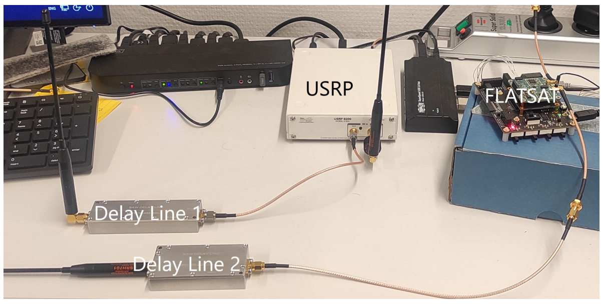


Figure 6.29: Radio testing equipment

As in section 6.1.4 and section 5.3.2, USRP delay line tests were conducted to validate that the above method works. However, in this test, as the satellite jitter is high enough such that there is a large uncertainty for each measurement. This uncertainty (with 95% confidence) is given as

$$Uncertainity\ u\ (95\%\ Confidence) = \frac{2.\sigma}{\sqrt{Number\ of\ Measurements}} \tag{6.16}$$

where σ is the standard deviation of the jitter for that measurement. This σ is calculated from a number of measurements as given in Table 6.5.

Both the jitter and uncertainty for each of the validation range tests are also given in Table 6.5. The uncertainty value says that statistically, 95% of all measurements obtained will be within [Range $-$ u, Range $+$ u]. As the expected distribution for the measured satellite delay is Gaussian, by increasing the number of measurements, u can be reduced, and the measurement can be brought closer to the real range value. This is possible as measurements in a Gaussian distribution are more probable to lie closer to the true value.

A test was carried out without the delay lines to get a baseline measurement for the satellite and ground instrument delay. This has a uncertainty in the range measurement of u_1. Thereafter, the delay lines are connected and measurements are taken to obtain the instrument delay + delay due to delay lines. This has an uncertainty of u_2. The uncertainty of the total range measurement is given as

$$u = \sqrt{u_1^2 + u_2^2} \tag{6.17}$$

and the range measurement is given as the mean of individual measurements

$$Range = \frac{\Sigma_i^n x_i}{n} \tag{6.18}$$

The range measurement is presented in the blue bars, and the known value of the delay line delay is given in the orange bar. The total uncertainty of the range measurement and of the delay line measurement is given in the black error bars in Figure 6.30. The uncertainty of the blue bars is the 95% uncertainty and the uncertainty of the orange bars is the 100% certainty (as given by the delay line data sheets[46, 45]). The orange bar uncertainty is exaggerated in the figure for visibility.

At least 20 measurements were taken for baseline and delay lines tests to reduce the uncertainty in the range measurement. The initial test was carried out at 4800bps. Since the uncertainty was very high for this data rate, higher data rates were used to get lower uncertainties in the range measurement. Two tests were carried out at 9600bps, one with more measurements to get a lower uncertainty (9600.1) and one with a higher uncertainty (9600.2). Due to high levels of uncertainty, data rates below 4800bps will not yield any meaningful results, so they were not tested. Due to time constraints, higher data rates than 9600bps were not explored.

In all the tests, once the bit correction method was carefully employed as described, the results fell within the uncertainty intervals, which validates that the bit corrected ranging method works.

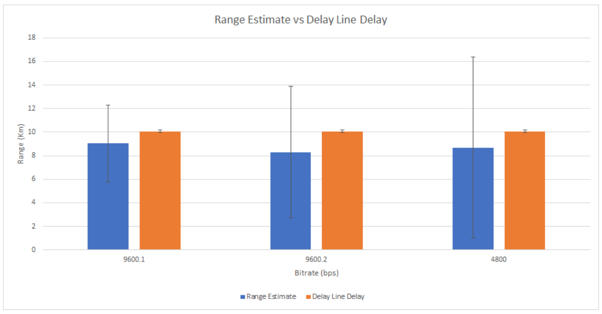


Figure 6.30: Estimated range with uncertainty window and true range

Table 6.5: Measurements for delay line based validation

Test ID	9600.1	9600.2	4800
Mean value for Baseline (km)	734.14	626.36	675.48
Number of Measurements for Baseline	40	20	20
1 sd Jitter for single measurement (km)	8.86	8.74	17.46
95% uncertainty in range (km)	2.80	3.9	6.05
Mean Value for Delay Lines (km)	743.18	634.65	684.17
Number of Measurements for Delay Lines	116	20	40
1 sd Jitter for single measurement (km)	8.86	8.93	18.54
95% uncertainty in range (km)	1.65	3.99	4.77
Range (km)	9.04	8.29	8.69
Range Error (km)	1.03	1.77	1.31
Range Uncertainty (km)	3.25	5.58	7.71

6.3.3. Performance

It is important to discuss the performance of the new system with time measurement and bit correction to see if the additional complexity pays off.

The 1σ range jitter of the system as a function of the bitrate is visible in Figure 6.31 along with the 95% uncertainty windows (uncertainty calculation in the following sections) and in Table 6.6. The jitter is represented by the 1σ of the obtained measurements, calculated using a different number of measurements for each bitrate. This is the jitter that can be expected for 68% of all obtained measurements. The limiting error for this case is expected to be the clock jitter whose 1σ is calculated as (see Table 6.4)

$$Clock\ Jitter = \frac{c}{3 * bitrate} \tag{6.19}$$

where c is the speed of light. As made in the previous sections (section 6.3.1), this equation also has the assumption that the clock jitter distributed as a Gaussian.

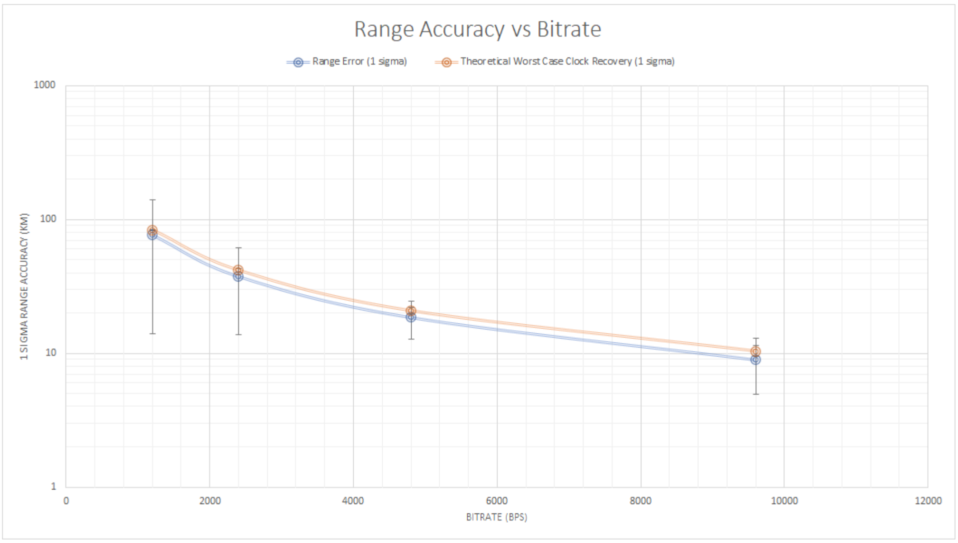


Figure 6.31: Range Jitter vs Bitrate

Table 6.6: 1σ range jitter vs bitrate

Bitrate (bps)	1200	2400	4800	9600
1 σ Range Jitter (km)	76.70	37.54	18.52	8.92
1 σ Theoretical Clock Recovery Jitter (km)	83.27	41.64	20.82	10.41
Offset between Clock and Range Jitter (km)	6.57	4.1	2.3	1.49
Number of Measurements	6	10	40	20
95% Uncertainty (km)	62.63	23.74	5.86	3.99

Here, two conclusions can be drawn:

1. An improvement in range jitter directly proportional to the bitrate is seen. Curve fitting for the available data points, the range accuracy can approximately be obtained as:

$$RangeAccuracy(km) = \frac{116701.826607}{Bitrate^(1.033129)} \tag{6.20}$$

with a curve fitting $R^2 = 1$, which means that the equation is an excellent predictor of the data (R^2 calculated by Excel and more decimal places are given for accuracy). Part of the bitrate power of 1.033 in the equation, is the presence of other errors in the range measurements (see error budget in Table 6.7). Another part of the bitrate power of 1.033 could be due to lower number of measurements used (due to time constraints) for the 1200bps and 2400bps, making the data less representative of the true jitter. Nevertheless, these results answer research question REQ-003, where it is clear that the bitrate of the systems is the main driver of the telemetry ranging performance.

2. The theoretical 1σ clock recovery error curve and the range error curve match very well with each other (R^2=0.999), indicating that the clock recovery is the dominant error in the range measurement. There is a measurable offset of a few kilometers between the two as seen in Table 6.6. This offset is larger than expected due to other error sources, and because the range data is less representative of the true jitter due to the limited number of measurements.

The overall error budget can be drawn as

Table 6.7: Error Budget for Telemetry Ranging with Time Measurement

Error Source	Expected 1σ Contribution	Source
Ground System Correlation Jitter	0.31 m	section 6.1.4
Ground System Correlation Bias	6.25 m	section 6.1.4
Ground Bias due to Carrier Frequency	<150m	section 6.2.1
Round off Errors due to Sample Rate Resolution of 1.2km	≤1.2 km	Visible in Sequential Ranging Tests (section 5.3.2), Easy to reduce by increasing sample rate
Time Tagging Error	12.5m	section C.1.3 and section 6.3.1
Satellite Clock Recovery	$\simeq speed\ of\ light(c)/(3 * bitrate)$	section 6.1, divided by three to get 1σ
Satellite Processing Jitter	Unknown	Majority was eliminated by time measurement, a small but unknown component will remain.

However, SNR also affects the performance of the method through affecting the ground correlation algorithm performance (as in Figure 6.10). Therefore, the final error would be given as

$$RangeAccuracy(km) = \frac{116701.826607}{Bitrate^(1.033129)} + Correlation\ Errors \tag{6.21}$$

There would also be a threshold SNR below which the satellite will not be able to detect the incoming data anymore, and telemetry ranging or normal communications cannot be conducted.

One of the changes in the telemetry ranging technique was that the satellite now sends down a 4 byte marker for every command received, along with a 4 byte packet of the measured time on board. These will add a minor penalty of 8 bytes per range measurement to the data transfer speed of the telemetry data. The maximum possible range speed would be to send a marker for every command received. Assuming a command length of 64 bytes (which was used in the experiments, see 6.1.1), this gives a marker count of 9600/64*8= 18 markers per seconds. This causes a penalty of 18*8*8=1152 bps, giving a final data transfer speed of 8448bps (88% of initial speed).

However, the number of range measurements in 1 second can be reduced. Say in 1 second, at 9600bps speed, the satellite only takes 5 range measurements. Then the markers will take 5*8*8=320bps to be sent and the effective data rate for telemetry would be 9600-320=9280bps (96% of initial speed). Therefore, this penalty will depend on the number of range measurements taken during a given amount of time and becomes less of an issue at higher data rates.

6.4. Results

The chapter started with section 6.1 describing the modifications necessary to use the telemetry ranging technique in the Delfi-PQ satellite.

In section 6.2, the modified telemetry ranging method was developed and tested using Delfi-PQ's engineering model available at TU Delft. The clock recovery of the satellite and the onboard processing jitter were found to be the limiting errors in this case and added upto errors in the order of 100s of km (see Table 6.3).

The onboard processing jitter was measured and mitigated in section 6.3, which revealed the presence of bit shifts in the ranging data. These bit shifts were corrected using statistics and the range obtained was validated using delay line range tests. The final range accuracy was found to be limited by the satellite clock recovery (see Figure 6.31 and Table 6.6).

6.4.1. Achieving the required range accuracy

Currently, at the data rates used by Delfi-PQ, the range accuracy is at the level of many kilometers per single measurement. It is necessary to obtain a higher accuracy so that the measurements are at a usable level, which is defined as better than TLE tracking measurements (1km accuracy) according to SREQ-001. This can be achieved by averaging the measurements.

Since the jitter is a Gaussian distribution after calibrating for the bits, taking the mean over a N number of measurements will give the mean a range accuracy of

$$R_{mean} = \frac{R}{\sqrt{N}} \tag{6.22}$$

where R is the range accuracy of a single measurement.

At the maximum possible in flight data rate of 9600bps, the satellite has a 1 sigma ranging accuracy of $8.92km$. To get the ranging accuracy within 1km as per SREQ-001, the amount of averaging needed is

$$N = (\frac{R}{R_{mean}})^2 = (\frac{8.92}{1})^2 = 79.5 \simeq 80 \tag{6.23}$$

Averaging over 80 measurements will be needed to obtain a sub kilometer ranging accuracy. At the data rate of 9600bps, considering a satellite command length of 64 bytes (testing gave this as a good balance between command recognition and command length), the time it would take to achieve sub kilometer accuracy is $80 * 64 * 8/9600$ =4.27seconds.
Therefore, a sub kilometer accuracy meeting SREQ-001 can be achieved in 4.27 seconds of continuous ranging.

One of the advantages of telemetry ranging is that over a pass of the satellite, ranging is active whenever telemetry is, meaning that it is feasible to range for a complete pass of Delfi-PQ (10 minutes). Therefore, integrating for the entirety of 10 minutes would give an improvement of

$$\frac{\sqrt{(60 * 10s)}}{(512bits/9600bits/s)} = \sqrt{11250} = 106times \tag{6.24}$$

improvement, which gives an accuracy of $8920/106$= 84.15 meters over an entire ranging pass.

The data rate linearly improves the performance seen in the satellite as visible from Figure 6.31. **Considering that higher data rates are achievable, the sub kilometer performance meeting REQ-001 can be achieved with a single measurement at a data rate of 85.63kbps and higher**. At data rates higher than 85.63Mbps, even sub meter accuracy can be achieved with this method. Higher data rates enable even more measurements to be taken in the same pass that could be used for range averaging to improve the accuracy achieved.

6.4.2. Comparison with Literature

An interesting comparison here would be that of time derived ranging techniques (section 2.1.2), whose results also were limited by clock recovery and onboard processing time. Using a ping packet to range in a CubeSat having a Si4464 Radio along with a MSP432 microcontroller, [38] achieved a 1σ range accuracy of 155m at 10000bps bitrate. They rebuilt the radio code to conduct only ranging but did not measure time onboard the satellite. To make sense of this result, let's take a look at the number of measurements they used to obtain this ranging accuracy. 10000 measurements were used to bring the range accuracy to a 155m level. Therefore, according to the properties of a Gaussian distribution. the accuracy of 1 measurement will be

$$Single\ Measurement\ Accuracy\ Literature = Averaged\ Accuracy \times \sqrt{N} = 155 \times 100 = 15.5km \tag{6.25}$$

at a bitrate of 10000bps, meaning the result in this literature is also limited by clock accuracy and onboard processing jitter (as in the initially proposed telemetry ranging method).
With 10000 measurements, the averaged measurement accuracy of telemetry ranging for Delfi-PQ will be

$$Average\ Accuracy\ Thesis = \frac{Single\ Measurement\ Range\ Accuracy}{\sqrt{N}} = \frac{8920}{100} = 89.2m \tag{6.26}$$

for a bitrate of 9600bps which is 42% improvement to the time derived ranging literature[38]. The similar order of accuracy of these results give confidence in the obtained performance of telemetry ranging in the thesis.

The results obtained for the initial prototype telemetry ranging method in section 6.1 had a higher onboard processing jitter due to all the other processing tasks that were also happening alongside ranging. The time derived ranging literature reduced this by only conducting ranging and reached the above results. In measuring the time onboard the satellite, this processing jitter was heavily mitigated and the accuracy obtained in the thesis for the ranging method (8.92km for 9600bps) approached the clock recovery limit, resulting in a better accuracy than the discussed time derived ranging literature.

Telemetry ranging in literature[75] manages to be better than the clock recovery limit by using a different clock recovery method (demod-remod method, see section 2.1.2) which required complete modification of the uplink processing chain. They also used phase measurement of incoming signals to avoid time tagging on board the satellite. These methods come at a greater cost of system complexity and especially onboard processing, which was not feasible on Delfi-PQ. This method also needs the hardware to support their usage.

Telemetry ranging using time measurements as developed in the tehsis finds a middle ground for small satellites where an accuracy upto the clock recovery timing was achieved while retaining the benefits of telemetry ranging (see 2.1.2) such as improved bandwidth and simultaneous telemetry/ranging without necessitating additional hardware or major changes to satellite operation and software. Due to this, the method can be implemented on Delfi-PQ with just a software update and ranging with all the above discussed benifits can be performed.

7 Conclusion

Satellite ranging is a vital tracking method frequently used to complement navigation solutions and other techniques such as Doppler and laser tracking. Two-way ranging methods have seen new research in the past decade, inventing the telemetry ranging class of techniques, which offer several important advantages over conventionally used techniques such as (section 2.1.2):

1. Elimination of a dedicated ranging signal, increasing the bandwidth available for telemetry data, which is important in scenarios where multiple satellites have to share a limited bandwidth for communication.
2. Allowing use of higher order modulations which further increase available bandwidth, reduce design complexity of and power usage of satellite signal amplifiers.
3. Enabling a ranging solution on satellites without dedicated ranging hardware.
4. Activating ranging whenever telemetry and telecommands are being transferred, eliminating the need to actively switch on and off ranging, simplifying operations.

These advantages are especially important to small nano-satellites such as CubeSats and PocketCubes which aim to become more compact and power efficient. Many small satellites skip having a ranging solution, due to the additional power and hardware requirements. Telemetry ranging can enable a ranging solution without placing any additional power or hardware requirements on the system,

Therefore, the master thesis explored the implementation of a telemetry ranging system for small satellites using the Delfi-PQ satellite as a demonstration platform. The work started with a literature study in chapter 2 which explored the state of the art in ranging and led to the choice of telemetry ranging as the research focus.

A fundamental method was needed to determine the time of flight of electromagnetic signals through the air. For this purpose, a correlation algorithm which could determine the phase difference (and through phase the time difference) between two copies of a signal was developed and its performance was explored in chapter 4.

The correlation algorithm was tested with the USRP radio, GNU Radio software and with the proven sequential ranging method in chapter 5. This proved that the correlation algorithm works, and that the USRP radio/ GNU Radio setup is capable of achieving ranging accuracies greater better than 150m. Sequential ranging theoretically can achieve high accuracies at the meter level, but this was not seen due to the limited system resolution (sample rate) of the tests.

Chapter 6 started with proposing modification to the literature telemetry ranging method in section 6.1 and exploring the effect of GMSK and GFSK modulation on the correlation accuracy. A method to see beyond the system resolution was adapted from literature (section 6.1.3) and was used here to probe the limits of the USRP-GNU Radio-Windows PC ground radio setup, which were obtained as in Table 6.1).

A prototype telemetry ranging method was developed and implemented for Delfi-PQ in section 6.2. This method was found to have an accuracy of the order of hundreds of kilometers due to variable on board processing jitter (section 6.2.2).

In section 6.3, an improvement to the previous method was implemented where the onboard processing time was measured and subtracted from the range measurement. Doing so revealed the presence of bit shifts in the range measurement (section 6.3.1). These bit shifts were eliminated in post processing using statistical reasoning and the overall method was validated using delay lines before obtaining a final range accuracy of tens of km for a single measurement as in section 6.3.3. This range accuracy was found to be largely limited by clock recovery error. Thus doubling the bitrate almost doubled the range performance.

Range averaging was proposed as a method to achieve an accuracy of 1km at lower bitrates. This works through the properties of a Gaussian distribution, where averaging increases the accuracy because the measurements are more probable to be near the true value of the range than not (section 6.4.1).

Finally, the results were compared to literature in section 6.4.2 where it was clear that telemetry ranging had an advantage compared to other time of flight measurement methods.

7.1. Answers to Research Questions

RQ-001: How can telemetry-based ranging methods be implemented using commercial hardware and software?
The telemetry ranging method used in thesis completely uses commercial and open source hardware and software as given in section 3.3. The telemetry ranging method developed in the thesis works as follows:

1. The ground station, which is a USRP radio has the ground signal processing set up with GNU Radio as given in section 6.2.1. The satellite commands are encoded and scrambled (see section 6.1.1 for more information) using Python and transfered into GNU Radio. The ground station sends GFSK modulated commands to the satellite on the uplink and receives, stores the telemetry signals from the satellite on the downlink.
2. The satellite receives the command signals and starts a timer onboard as described in section 6.3.1. Once a data packet or response command packet is sent down the telemetry channel, the satellite sends a ASM behind the packet. The timer is used to estimate the time between the reception of the command and transmission of this ASM. This estimation is then sent in the telemetry stream using one of two options defined in section 6.3.1.
3. Back on the ground, Python code using the correlation algorithm defined in chapter 4 scans through the entire reception telemetry stream and calculates the location of the ASM in this stream. Since the ground knows the transmission time of the commands and now also the reception time of the ASM, the code now calculates a range measurement (see section 6.2.1), which is corrected using the onboard time measurement.
4. In Delfi-PQ, this revealed the presence of bit shifts in the data, as described in section 6.3.1. There were corrected using statistical reasoning to obtain the range of the satellite.

RQ-001 (a): What are the requirements placed on the hardware and software used?
This telemetry ranging system does not place any additional requirements to the satellite hardware on Delfi-PQ for it to work. The requirements on the software are:

1. The satellite shall be capable of recognizing the reception of a command and transmission of a marker as defined in Figure 6.1.
2. When the satellite runs other processes between reception of command and transmission of marker, it should be capable of timing this process, either as a measure of time as defined in Figure 6.23 or as a measure of the phase of the incoming commands as mentioned in section 6.3.1.

3. The ground hardware and software used should be capable of timing or correlating the received signals to the desired level of accuracy. In the thesis, the GNU Radio-USRP-Windows PC system, when using a GPS synced GPSDO showed a correlation capability (baseline jitter) of within 1meter in section 6.1.4.

In case timing the signal onboard is not possible, jitters at the level of hundreds of kilometers as seen in 6.22 can be expected. The system's normal operation can be changed in this case to use a technique similar to the time derived ranging paper's[38] ping based ranging technique (see section 6.4.2) while sending the response as telemetry packets with an ASM attached. In this case performance that is close to the literature technique which is 79% worse than the thesis can be expected. It may also have a detrimental effect on the telemetry transfer speed.

RQ-001 (b): What are the main challenges, if any, affecting the viable use of telemetry ranging for Delfi-PQ and other small satellites?
The main factor limiting the performance of telemetry ranging for Delfi-PQ is the ranging jitter due to the clock recovery capability of the satellite's symbol tracking loop. This emerged as a hard limit to the accuracy of a single measurement at a given bitrate for Delfi-PQ and necessitated either higher bitrates or averaging over a number of measurements to achieve a 1km accuracy as in the requirements (see section 6.4.1).

This error was eliminated in literature[75] by estimating the clock recovery error with a demod-remod correlation based algorithm to obtain better performances (see section 2.1.2), but this was not feasible to implement in Delfi-PQ due to hardware constraints.

Therefore, implementing a better clock recovery method in the small satellite will be the challenge, which when solved will allow for several magnitudes better range performance.

RQ-002: How do the telemetry ranging technique and the obtained ranging results compare to the results in published literature?
The telemetry ranging technique developed in the thesis retains all the advantages of the literature telemetry ranging technique (see section 6.1). However, it loses out on performance due to the clock recovery error. Adding an improved clock recovery method to Delfi-PQ would require hardware modifications, which were precluded by the system requirements (see section 3.2).

Compared to time derived ranging techniques, the technique achieves higher range accuracies and does not require the system to be built only for ranging (see section 6.4.2).

REQ-002(a) What are differences, if any, between the research and literature?
The telemetry ranging method developed in the thesis was adapted to the capabilities of Delfi-PQ. The tracking loop clock recovery error on the uplink of the satellite could not be estimated as it was in telemetry ranging literature and was present as the limiting error. Further, the fractional bit count was not tracked on board the satellite and instead the time between the reception of a command and transmission of a marker in the satellite was measured (see section 6.3.1). The phase determination to measure the time of flight of the signals for ranging was done onboard the satellite in lite rature, whereas it was offloaded to the ground in the thesis. The performance of the correlation algorithm of the thesis was slightly worse than literature as in section 6.1.3.

When compared to time derived ranging techniques, the system uses an ASM on the downlink instead of ping packet. This was due to the presence of the scrambler (see section 6.2.2). The literature calibrates for the time between reception and transmission of command. In the thesis, this time is directly measured.

REQ-002(b) How do these differences affect the results obtained?
The difference with the telemetry ranging literature lead to the performance of the method being limited by the clock recovery. There is also a small cost to the downlink telemetry transfer speed due to the addition of the ASM (see section 6.3.3). The satellite processing burden is also reduced because it does not need to correlate any signals on board and just needs to attach an ASM to the downlink channel.

Compared to the time derived ranging literature, the time measurement allow for mitigation of the satellite onboard processing jitter's contribution, improving the performance of ranging by 46% (see section 6.4.2).

RQ-003: What are the main parameters in Delfi-PQ that affect the performance of the method?
The satellite **Bitrate** was found to be the main driver for the performance at the given SNR (section 6.3.3). The **SNR** also affects the performance of the correlation algorithm (section 4.3), with **signal integration time** improving performances at lower SNRs at the cost of having lesser measurements per ranging pass. A larger **number of measurements** can also be used to improve the overall range accuracy through range averaging.

The ground side also has a minor contribution to error from system jitter and bias in the order of meters (see section 6.1.4).

REQ-003(a) By adjusting these parameters, are we capable of reaching a viable ranging solution?
By adjusting the bitrate, range measurements better than 1KM, meeting the system requirements, can be achieved at higher data rates. Further, a number of measurements can be averaged to increase the obtained accuracy. For Delfi-PQ, a viable ranging performance (better than 1 km) is reached with 4.27 seconds of ranging at 9600bps or in a single measurement at 80.39Kbps in the tested high SNR region.

At higher SNRs, correlation performance is good (6.1.3). At lower SNRs, correlation performance can be improved by increasing signal integration time, which reduces the number of measurements that are taken during the ranging pass.

REQ-003 (b) Why do the parameters affect the range performance in the Delfi-PQ satellite?
The bitrate affects the clock recovery error and onboard processing error of Delfi-PQ. Increasing this directly reduces both errors, as the satellite runs faster, causing the tracking and processing to happen faster. The time measurement onboard the satellite eliminated the onboard processing error, leaving only the clock recovery error, which halves with a doubling of the bitrate (see section 6.3.3).

The SNR affects the performance of correlation algorithm used to determine the overall time of flight for the ranging signal (see section 4.3).

The integration time decreases the correlation error at the lower SNRs but has a threshold SNR below which it is useful. This is because it increases the height of the correlation peaks, but not the height of noise (see section 4.3 again).

Finally, averaging the number of measurements improves the accuracy of ranging due to the properties of Gaussian distributions (see section 6.4.1).

The ground USRP jitter and bias which add to the range error are possibly due to correlation performance and due to GNU Radio's modulation techniques as they change between the modulations (see subsection 6.1.4).

7.1.1. System Requirements

The systems requirements and their relevant results are given in Table 7.1.

Table 7.1: System Requirements

ID	Requirement	Result	Validation by
SREQ-001	The system shall be capable of achieving better than TLE range performance(>1Km)	Better than TLE Performance achievable at >85.6 kbps or with 4.27sec of range averaging at 9600Kbps	Delay Line Tests
SREQ-002	The system shall be capable of operating continuously to enable ranging throughout a Delfi-PQ satellite pass (∼10 mins)	No system restriction on time of ranging pass, needs minor modifications to ground correlation code	Visual inspection of system operation
SREQ-003	The system shall work with commercially available radio equipment and open source software on the ground	GNU Radio, USRP for Ground station, MSP432 microcontroller and SX1278 modem for satellite are all commercially available	Inspection
SREQ-004	The system shall not require any hardware modifications to the satellite.	No hardware modifications are needed	Testing with Delfi-PQ, no hardware modification were needed
SREQ-005	The system shall enable simultaneous telemetry and ranging	System set up to work with simulatanous telemetry and ranging, needs testing to confirm	Systems Tests demonstrated new capabilities, only integration with proven technology left, so validated
SREQ-006	The system shall enable bandwidth efficient modulations such as GMSK and GFSK	System enables GFSK and GMSK for ranging	GFSK+GNU Radio tested with Delfi-PQ, proven working. GMSK+GNU radio has issues but GMSK from ISISpace CheckoutBox works, so likely GNU Radio issue

7.2. Future Recommendations

The thesis focused on demonstrating telemetry ranging as a viable technique to enable ranging in small satellites without placing additional requirements on satellite hardware. The main focus was on proving the working of a prototype of the technique with Delfi-PQ and this was successful (see section 6.3.3). However, there are a few future recommendations that contribute to the functionality of the current technique, to the ease of use and to improvements in performance. They are presented here:

1. Functionality Improvement 1- Time transfer to ground (section 6.3.1): Currently, the technique uses the satellite console to send the time information measured onboard to the PC. However, in the future, this time information needs to be transferred over the air. For this, there are two possibilities, as discussed in section 6.3.1.
2. Functionality Improvement 2- Correlation algorithm memory efficiency: The correlation algorithm on the ground (Listing A.5) currently stores the entire receiver stream into memory for correlation. This precludes the use of longer recordings. To fix this, the correlation can be simply done on smaller parts and these can be stitched to obtain the final correlation matrix. This was not needed for the tests, so was not implemented, but can be done and validated very quickly.
3. Functionality Improvement 3- Attaching marker behind telemetry: In the testing for the thesis, the flatsat board did not have telemetry transfer occurring. Thus, data transfer was mimicked using command data packets behind which a marker was attached. The way the current code is set up, a marker will be attached behind the next data packet after a command was received. This setup should work out of the box with telemetry data packets. However, this functionality needs to be validated with tests.
4. Final Testing 1- Testing in Flight with Delfi-PQ: The telemetry ranging method in the thesis was validated to work in section 6.3.2. Once functionality improvement 1 (ideally 2 and 3 too, but not necessary) is implemented, the technique is ready for in flight testing. Next step would be to upload the software to Delfi-PQ which is in orbit and estimate its range. Using the GPS receiver on board Delfi-PQ [54], its position can be determined, and the range measurement can be validated.

5. Quality of Life 1- Quadratic Interpolation Algorithm Validation: This algorithm was used in section 6.1.3 as it was used in literature[75], whose results the section tried to emulate. This method gives a quick estimate of the subsample peak. To validate this for usage in ranging, delay line ranging tests can be carried out over a range of SNRs using different modulations and different ranging signals. If sufficiently validated, this would allow for cheap and convenient estimation of subsample delay even onboard the satellite, which could be useful for modified implementations of telemetry ranging.
6. Performance Improvement 1- Optimize the clock recovery symbol tracking loop. The tracking loop parameters such as loop bandwidth and integration time can be varied to improve its estimation performance, which will improve the overall ranging accuracy.
7. Performance Improvement 2- Remove the SPI: In a more invasive process, the SPI of Delfi-PQ can be removed, and each bit can be transferred one by one. This will allow for accurate measurement of the processing delays on board the satellite instead of the current estimation. This will provide an insight into the exact error buildup of the system (Table 6.7).
8. Performance possibility 1- Externally measure signal timing: Using an oscilloscope, the exact arrival time and departure time of the signals onboard the Delfi-PQ satellite can be probed. Doing so may reveal some patterns in the satellite side delay, which can help design a better mitigation algorithm.
9. Alternate Implementation 1- Estimate the tracking loop error: If the requirement of no hardware changes is removed, implementation of the demod-remod tracking loop error estimator as in literature(section 6.1,[75]), will provide magnitudes better range performance.

References

[1] Rajat Acharya. “Chapter 7 Errors and Error Corrections”. In: *Understanding Satellite Navigation*. Elsevier, 2014, pp. 243–279. DOI: 10.1016/b978-0-12-799949-4.00007-5.

[2] David W Allan. *Statistics of Atomic Frequency Standards*. Tech. rep. 2. 1966.

[3] Kenneth Andrews et al. “Telemetry-based ranging”. In: *IEEE Aerospace Conference Proceedings*. 2010. ISBN: 9781424438884. DOI: 10.1109/AERO.2010.5446926.

[4] J W Armstrong et al. *Reducing Antenna Mechanical Noise in Precision Doppler Tracking*. Tech. rep. 2006.

[5] ESA Authors. *Front end*. URL: https://gssc.esa.int/navipedia/index.php/Front_End.

[6] Wikipedia Authors. *Circular convolution*. Mar. 2023. URL: https://en.wikipedia.org/wiki/Circular_convolution.

[7] Dan Baker and Brian Avenell. *Phase synchronization techniques · GNU Radio*. Jan. 1AD. URL: https://www.gnuradio.org/grcon/grcon19/presentations/Phase_Synchronization_Techniques/.

[8] Dean Banerjee and Albert Einstein. *PLL Performance, Simulation, and Design 5th Edition*. Tech. rep. 2017.

[9] Harold W Baugh. *Sequential Ranging How it Works*. Tech. rep. 1993.

[10] Jeff B. Berner and Scott H. Bryant. “Operations comparison of deep space ranging types: Sequential tone vs. pseudo-noise”. In: *IEEE Aerospace Conference Proceedings*. Vol. 3. 2002, pp. 1313–1326. ISBN: 078037231X. DOI: 10.1109/AERO.2002.1035264.

[11] Jeff B. Berner, Scott H. Bryant, and Peter W. Kinman. “Range measurement as practiced in the deep space network”. In: *Proceedings of the IEEE* 95.11 (2007), pp. 2202–2214. ISSN: 00189219. DOI: 10.1109/JPROC.2007.905128.

[12] Green Book. *Report Concerning Space Data System Standards PSEUDO-NOISE (PN) RANGING SYSTEMS INFORMATIONAL REPORT*. Tech. rep. 2014.

[13] Richard P Brent. *Algorithms for Minimization Without Derivatives*. New York, UNITED STATES: Dover Publications, 2003. ISBN: 9780486143682. URL: http://ebookcentral.proquest.com/lib/delft/detail.action?docID=1900763.

[14] S Bryant et al. *A two-way noncoherent ranging technique for deep space missions*. 2002. DOI: hdl/2014/13580. URL: https://hdl.handle.net/2014/13580.

[15] S. Bryant. “Using digital signal processor technology to simplify deep space ranging”. In: *IEEE Aerospace Conference Proceedings*. Vol. 3. 2001, pp. 31277–31282. DOI: 10.1109/aero.2001.931357.

[16] Paolo Cappuccio et al. “Report on first inflight data of bepicolombo’s mercury orbiter radio science experiment”. In: *IEEE Transactions on Aerospace and Electronic Systems* 56.6 (Dec. 2020), pp. 4984–4988. ISSN: 15579603. DOI: 10.1109/TAES.2020.3008577.

[17] *Chapter 203 Sequential Ranging DSN Telecommunications Link Design Handbook*. Tech. rep. 2023. URL: http://deepspace.jpl.nasa.gov/dsndocs/810-005/.

[18] *Chapter 214 Pseudo Noise Ranging DSN Telecommunications Link Design Handbook*. 2023. URL: http://deepspace.jpl.nasa.gov/dsndocs/810-005/.

[19] Anritsu Company. URL: https://anritsu.typepad.com/interferencehunting/interference-hunting/.

[20] ByjusWeb Contributers. *Satellite Communication: Definition, block diagram, advantages, applications*. Feb. 2023. URL: https://byjus.com/physics/satellite-communication/.

[21] GNURadio Contributers. *FAQ*. URL: `https://wiki.gnuradio.org/index.php/FAQ#Which_operating_systems_are_supported?_Does_GNU_Radio_run_on_Windows_or_Mac_OS_X?_What_about_32,_64_Bits?`.

[22] Spectrum Control. *Saw technology*. URL: `https://info.spectrumcontrol.com/saw-technology-va`.

[23] Richard J Debolt et al. *A Regenerative Pseudonoise Range Tracking System for the New Horizons Spacecraft*. Tech. rep. 2000.

[24] Destevez. *JWST sequential ranging*. Dec. 2021. URL: `https://destevez.net/2021/12/jwst-sequential-ranging/`.

[25] Dominic Dirkx et al. “Laser and radio tracking for planetary science missions—a comparison”. In: *Journal of Geodesy* 93.11 (Nov. 2019), pp. 2405–2420. ISSN: 14321394. DOI: `10.1007/s00190-018-1171-x`.

[26] *DSN Telecommunications Link Design Handbook*. Tech. rep. 2023. URL: `http://deepspace.jpl.nasa.gov/dsndocs/810-005/`.

[27] National Instruments Brand Ettus Research. *USRP B200 USB Software Defined Radio (SDR)*. URL: `https://www.ettus.com/all-products/ub200-kit/`.

[28] Cyrus Foster, Henry Hallam, and James Mason. *Orbit Determination and Differential-drag Control of Planet Labs Cubesat Constellations*. 2015. arXiv: `1509.03270 [physics.space-ph]`.

[29] Jon Hamkins. *Telemetry Ranging-Initial Comparisons*. Tech. rep. 2015.

[30] Jon Hamkins et al. *Telemetry Ranging: Concepts*. Tech. rep. 2015.

[31] Jon Hamkins et al. *Telemetry Ranging: Laboratory Validation Tests and End-to-End Performance*. Tech. rep. 2016, pp. 42–206.

[32] Jon Hamkins et al. *Telemetry Ranging: Signal Processing*. Tech. rep. 2016.

[33] Joseph Hennawy et al. “Telemetry ranging using software-defined radios”. In: *IEEE Aerospace Conference Proceedings*. Vol. 2015-June. IEEE Computer Society, June 2015. ISBN: 9781479953790. DOI: `10.1109/AERO.2015.7119178`.

[34] Peter Holsters, Giovanni Boscagli, and Enrico Vassallo. “Pseudo-noise ranging for future transparent and regenerative channels”. In: *SpaceOps 2008 Conference*. American Institute of Aeronautics and Astronautics Inc., 2008. ISBN: 9781624101670. DOI: `10.2514/6.2008-3277`.

[35] HOPFIELD HS. “TWO-QUARTIC TROPOSPHERIC REFRACTIVITY PROFILE FOR CORRECTING SATELLITE DATA”. In: *J Geophys Res* 74.18 (1969), pp. 4487–4499. DOI: `10.1029/jc074i018p04487`.

[36] Luciano Iess et al. “Astra: Interdisciplinary study on enhancement of the end-to-end accuracy for spacecraft tracking techniques”. In: *Acta Astronautica* 94.2 (2014), pp. 699–707. ISSN: 00945765. DOI: `10.1016/j.actaastro.2013.06.011`.

[37] National Instruments. *Global synchronization and clock disciplining with NI USRP software defined radios*. URL: `https://www.ni.com/en/shop/wireless-design-test/what-is-a-usrp-software-defined-radio/global-synchronization-and-clock-disciplining-with-ni-usrp-293x-.html`.

[38] Grant Iraci, Chris Gnam, and John Crassidis. *An Open Source Radio for Low Cost Small Satellite Ranging*. Tech. rep. 2018. URL: `https://github.com/UBNanosatLab`.

[39] J. J. Spilker Jr. “Troposheric Effects On Gps”. In: *Global Positioning System: Theory and Applications, Volume I*. 1996, pp. 517–546. DOI: `10.2514/5.9781600866388.0517.0546`. URL: `https://arc.aiaa.org/doi/abs/10.2514/5.9781600866388.0517.0546`.

[40] Peter W Kinm and Jeff B Berner. *Two-way Ranging During Early Mission Phase*. Tech. rep. 2003.

[41] P W Kinman et al. *Mutual Interference of Ranging and Telemetry*. Tech. rep. 2000.

[42] S Knappe et al. *LONG-TERM STABILITY OF NIST CHIP-SCALE ATOMIC CLOCK PHYSICS PACKAGES*. Tech. rep. 2006.

[43] Jinghong Liu et al. “TLE orbit determination using simplex method”. In: *Geodesy and Geodynamics* 14.5 (2023), pp. 438–455. ISSN: 1674-9847. DOI: https://doi.org/10.1016/j.geog.2023.03.004. URL: https://www.sciencedirect.com/science/article/pii/S1674984723000368.

[44] Heleen Luts. *DIAGNOSIS OF HEARING LOSS IN NEWBORNS Clinical application of auditory steady-state responses*. Tech. rep. 2005.

[45] Microsaw. *MH1110-452_600-2 Delay Line Datasheet*. Tech. rep. 2022.

[46] Microsaw. *MH1159-225.000-2 Delay Line Datasheet*. Tech. rep. 2018. URL: https://www.microsaw.fi/products/saw-delay-lines/saw-delay-lines.php?id=c1-asc.

[47] David L Mills. *Computer_Network_Time_Synchronization*. 2011.

[48] Marcus Mueller. *Rotator*. URL: https://wiki.gnuradio.org/index.php/Rotator.

[49] Dries Neirynck, Eric Luk, and Michael McLaughlin. “An alternative double-sided two-way ranging method”. In: *Proceedings of the 2016 13th Workshop on Positioning, Navigation and Communication, WPNC 2016*. Institute of Electrical and Electronics Engineers Inc., Jan. 2017. ISBN: 9781509054404. DOI: 10.1109/WPNC.2016.7822844.

[50] Marc Sanchez Net and Jon Hamkins. *Optical Telemetry Ranging*. Tech. rep. 2020.

[51] Berkeley Nucleonics. *Model 855B phase coherence terminology*. URL: https://www.berkeleynucleonics.com/sites/default/files/products/resources/855b_app_note_-phase_coherence_terminology.pdf.

[52] Richard S Orr. *Combined GMSK Modulation and PN Ranging for Communications and Navigation*. Tech. rep. 2008.

[53] M S Radomski and C E Doll. *Differenced Range Versus Integrated Doppler (DRVID) Ionospheric Analysis of Metric Tracking in the Tracking and Data Relay Satellite System (TDRSS)**. Tech. rep.

[54] S Radu et al. *Delfi-PQ: The first pocketqube of Delft University of Technology*. Tech. rep. 2018.

[55] *Recommendation for Space Data System Standards BLUE BOOK RECOMMENDED STANDARD PROXIMITY-1 SPACE LINK PROTOCOL-DATA LINK LAYER*. Tech. rep. 2020.

[56] *Recommendation for Space Data System Standards PSEUDO-NOISE (PN) RANGING SYSTEMS RECOMMENDED STANDARD BLUE BOOK*. Tech. rep. 2022.

[57] *Report Concerning Space Data System Standards GREEN BOOK INFORMATIONAL REPORT TM SYNCHRONIZATION AND CHANNEL CODING*. Tech. rep. 2020.

[58] *Report Concerning Space Data System Standards SIMULTANEOUS TRANSMISSION OF GMSK TELEMETRY AND PN RANGING INFORMATIONAL REPORT CCSDS 413.1-G-2 GREEN BOOK*. Tech. rep. 2021.

[59] Ettus Research. *Tuning notes*. URL: https://files.ettus.com/manual/page_general.html.

[60] Margaret M. Rybak et al. “Chip Scale Atomic Clock–Driven One-Way Radiometric Tracking for Low-Earth-Orbit CubeSat Navigation”. In: *Journal of Spacecraft and Rockets* 58.1 (2021), pp. 200–209. DOI: 10.2514/1.A34684. eprint: https://doi.org/10.2514/1.A34684. URL: https://doi.org/10.2514/1.A34684.

[61] Semtech Authors. *Semtech SX 1276/77/78/79 Data Sheet*. Tech. rep. 2020.

[62] Jill Seubert, Todd Ely, and Jeffrey Stuart. “RESULTS OF THE DEEP SPACE ATOMIC CLOCK DEEP SPACE NAVIGATION ANALOG EXPERIMENT”. In: *Advances in the Astronautical Sciences*. Vol. 175. Univelt Inc., 2021, pp. 2965–2980. ISBN: 9780877036753. DOI: 10.2514/1.a35334.

[63] Steven W. Smith. “Chapter 18: FFT Convolution”. In: *The scientist and engineer’s Guide to Digital Signal Processing*. California Technical Pub., 1999.

[64] TimeTag Space. *Time Tagging Technology with Picosecond Resolution for Space Applications*. Tech. rep. 2020.

[65] Stefano Speretta et al. “Software-defined testbed for next generation navigation transponders”. In: *TTC 2019 - 8th ESA International Workshop on Tracking, Telemetry and Command Systems for Space Applications*. Institute of Electrical and Electronics Engineers Inc., Sept. 2019. ISBN: 9781728136998. DOI: 10.1109/TTC.2019.8895459.

[66] R. C. Tausworthe and J. R. Smith. “A simplified, general-purpose deep-space ranging correlator design”. In: *The Telecommunications and Data Acquisition Report*. Feb. 1988, pp. 105–109.

[67] Robert C Tausworthe. *Tau Ranging Revisited*. Tech. rep. 1990. URL: https://www.researchgate.net/publication/24353601.

[68] Carri Telecommunications. *What is modulation in telecommunications?* URL: https://carritechtelecommunications.quora.com/What-is-Modulation-in-Telecommunications.

[69] Catherine L Thornton and James S Border. *Radiometric Tracking Techniques for Deep-Space Navigation MONOGRAPH 1 DEEP-SPACE COMMUNICATIONS AND NAVIGATION SERIES*. 2005.

[70] P. Tortora et al. “Precise Cassini Navigation During Solar Conjunctions Through Multifrequency Plasma Calibrations”. In: *Journal of Guidance, Control, and Dynamics* 27.2 (2004), pp. 251–257. ISSN: 15333884. DOI: 10.2514/1.997.

[71] Erdem Turan, Stefano Speretta, and Eberhard Gill. “Autonomous Crosslink Radionavigation for a Lunar CubeSat Mission”. In: *Frontiers in Space Technologies* 3 (June 2022). DOI: 10.3389/frspt.2022.919311.

[72] Erdem Turan, Stefano Speretta, and Eberhard Gill. *Autonomous navigation for deep space small satellites: Scientific and technological advances*. 2022. DOI: 10.1016/j.actaastro.2021.12.030.

[73] Erdem Turan, Stefano Speretta, and Eberhard Gill. “Performance analysis of crosslink radiometric measurement based autonomous orbit determination for cislunar small satellite formations”. In: *Advances in Space Research* (Nov. 2022). ISSN: 02731177. DOI: 10.1016/j.asr.2022.11.032. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117722010638.

[74] Victor Vilnrotter et al. *Performance Analysis of Digital Tracking Loops for Telemetry Ranging Applications*. Tech. rep. 2015.

[75] Victor A Vilnrotter and Jon Hamkins. “Telecommand/Telemetry Ranging for Deep-Space Applications”. In: *2019 IEEE Aerospace Conference* (2019), pp. 1–10.

[76] W.L.Martin and J.W.Layland. *Binary Sequential Ranging With Sine Waves*. Tech. rep. JPL Deep Space Network, 1976, pp. 30–36.

[77] Karel F Wakker. *Chapter 1.3: Deterministic and Chaotic Motion - Fundamentals of Astrodynamics*. 2015. ISBN: 9789461864192. URL: http://repository.tudelft.nl..

[78] PennState WebAuthors. *Doppler shift*. URL: https://www.e-education.psu.edu/geog862/node/1786.

[79] ESA Website Authors. *Navigation*. URL: https://www.cosmos.esa.int/web/bepicolombo/more.

[80] Multiple GNU radio Wiki Contributers. *File sink*. URL: https://wiki.gnuradio.org/index.php/File_Sink.

[81] GNU radio Wiki Contributor. *MPSK Snr Estimator*. URL: https://wiki.gnuradio.org/index.php/MPSK_SNR_Estimator.

[82] Wikipedia contributors. *Computational complexity of mathematical operations — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Computational_complexity_of_mathematical_operations&oldid=1174886199.

[83] Wikipedia contributors. *Cross-correlation — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-August-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Cross-correlation&oldid=1170377241.

[84] Wikipedia contributors. *Sampling (signal processing) — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Sampling_(signal_processing)&oldid=1178840353.

[85] Simon S. Woo, Jay L. Gao, and David Mills. “Space network time distribution and synchronization protocol development for Mars proximity link”. In: *SpaceOps 2010 Conference*. 2010. ISBN: 9781624101649. DOI: 10.2514/6.2010-2360.

A
Source Code

The source code including these snippets are available at `https://github.com/Markiv999/GNURADIO_THESIS_MASTER`.

Listing A.1: GNU Radio Correlation Delay Estimator Custom Block

```python
import numpy as np
from gnuradio import gr
import scipy as scipy

class CorrelationDelayEstimator(gr.decim_block):
    def __init__(self, vectorsize=18750, sample_rate=500000, max_delay=1000):
        gr.decim_block.__init__(
            self,
            name='CorrelationDelayEstimator',
            in_sig=[np.complex64, np.complex64],
            out_sig=[np.float32],
            decim=vectorsize
        )
        self.set_relative_rate(1.0/vectorsize)
        self.decimation = vectorsize
        self.vectorsize = vectorsize
        self.range_clock_signal = np.zeros(vectorsize, dtype=np.complex64)
        self.sequential_signal = np.zeros(vectorsize, dtype=np.complex64)
        self.i = 0
        self.j = 0
        self.counter = 0
        self.sample_rate = sample_rate
        self.correlation_values = np.zeros(vectorsize, dtype=np.complex64)
        self.max_delay=max_delay

    def work(self, input_items, output_items):
        sequential_signal = input_items[0]
        range_clock_signal = input_items[1]
        num_samples = sequential_signal.shape[0]
        n=1
        # Process each sample in the input buffer
        for k in range(num_samples):
            if self.i < self.vectorsize/n:
                # Take the latest values from the buffer
                self.range_clock_signal[self.i] = range_clock_signal[k]
                self.sequential_signal[self.i] = sequential_signal[k]
                self.i += 1
            else:
                # Calculate delay
                #self.range_clock_signal=self.range_clock_signal[0 : self.vectorsize/n]
                #self.sequential_signal = self.sequential_signal[0 : self.vectorsize/n]
                self.correlation_values = np.fft.ifft(np.fft.fft(self.range_clock_signal) *
                    np.conj(np.fft.fft(self.sequential_signal)))
                #self.correlation_values= scipy.signal.correlate(self.range_clock_signal,
                    self.sequential_signal, mode='full', method='auto')
                self.sample_delay_count = np.argmax(np.abs(self.correlation_values))
               # self.sample_delay_count=np.argmax(np.correlate(self.range_clock_signal,self.sequential_signal))
                #delay = -(self.vectorsize - self.sample_delay_count)
                delay=self.sample_delay_count
                delay_in_micro_seconds = 10 ** 6 * delay / self.sample_rate
                #correlationfactor=self.correlation_values[self.sample_delay_count]/np.sum(np.abs(self.correlation_values))
                output_items[0][0] = delay
                #n=n*2
                #if n==16:
                #    n=1
                self.i = 0
        self.j = 0
        return num_samples // self.vectorsize
```

Listing A.2: USRP Initiation and Overflow Error

```
runfile('D:/GNURADIO_THESIS_MASTER/CorrelationwithDelfiBoard/DifferentRate/TransmissionTestfinal2_9600bps.py', wdir='D:/GNURADIO_THESIS_MASTER/CorrelationwithDelfiBoard/DifferentRate')
GPS lock status: locked
1698067480.9119484
1698067482.0
GPS lock status: locked
1698067482.0
1698067482.0

[INFO] [UHD] Win32; Microsoft Visual C++ version 14.2; Boost_107800; UHD_4.4.0.0-release
[INFO] [B200] Detected Device: B200
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Detecting internal GPSDO....
[INFO] [GPS] Found an internal GPSDO: LC_XO, Firmware Rev 0.929b
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Setting master clock rate selection to 'automatic'.
[INFO] [B200] Asking for clock rate 16.000000 MHz...
[INFO] [B200] Actually got clock rate 16.000000 MHz.
[INFO] [B200] Asking for clock rate 32.000000 MHz...
[INFO] [B200] Actually got clock rate 32.000000 MHz.
O
usrp_source :error: In the last 10820 ms, 1 overflows occurred.
```

Listing A.3: Delfi PQ ASM and Transmit Time Tag code

```
uint32_t time_receiver_byte=0;
uint32_t time_transmitter_byte=0;
uint32_t time_receiver_bit=0;
uint32_t time_transmitter_bit=0;



uint8_t attachsinkmarkerflag=0;
bool activateattachsinkmarker=false;
bool timerflag=false;
uint32_t telemetry_range_time=0;
int transmitindex=0;
int receiveindex=0;

bool asmtimeflag=true;
bool printflag=false;
uint8_t COMMRadio::onTransmit(){

    transmit_timecounter_telemetry_ranging=MAP_Timer32_getValue(TIMER32_1_BASE);

    //NOTE, SPI Bus is configured to MSB_first, meaning that the bit transmitted first should be the MSB.
//    Console::log("onTransmit!");
    uint8_t outputByte = 0;
    //These detect the transmission of a packet and of 5 7E packets after which the attach sink marker will be attached.



    if(txEnabled)
    {
       for(int i = 0; i < 8; i++) // Send 8bits per call
       {
           if(!txFlagQue && !txPacketsInBuffer()) //if no idle flags are queued, and no data is ready.
           {
               if(txIdleMode)
               {
                   txFlagQue++; //stay Idle by transmitting a flag
//                   Console::log("TX: stuffing");
               }
               else
               {
                   //disable Transmit
                   disableTransmit();
                   return 0x00;
               }
           }

           if(activateattachsinkmarker==true)
           {

               //Console::log("TX: SendingAttachSinkMarker byte");

                                                // uint8_t inBit = ((0x55) >> txBitIndex); //& 0x01;
                                                // outputByte = outputByte | (inBit << (7-i));
                                                // txBitIndex++;
                                                 //Console::log("TX: TxBitIndex2: %d %d", txBitIndex, attachsinkmarkerflag);


                                                 if(attachsinkmarkerflag==4)
                                                 { if (asmtimeflag==true)
                                                     {
                                                     //Bit Correction Timecounter telemetry ranging
                                                     uint32_t temp=48000000/2400;

                                                     telemetry_range_time=(AX25Sync.receive_timecounter_telemetry_ranging[0]-(transmit_timecounter_telemetry_ranging-(static_cast<uint32_t>(i)*temp)));
                                                     transmitindex=static_cast<uint32_t>(i);
                                                     receiveindex=AX25Sync.receiveindex;

                                                     //Variables to record each individual timing-receiver
                                                     time_receiver_byte=AX25Sync.timecounter_holder_for_print_byte_timing;
                                                     time_receiver_bit=AX25Sync.receive_timecounter_telemetry_ranging[0];

                                                     //Variables to record each individual timing-transmitter
                                                     time_transmitter_byte=
                                                         transmit_timecounter_telemetry_ranging
                                                         ;
                                                     time_transmitter_bit=
                                                         transmit_timecounter_telemetry_ranging
                                                          - (static_cast<uint32_t>(i)*temp) ;
                                                     //Console::log("%d",AX25Sync.queue_final
                                                         );


                                                     //pop first value in Telemetry timing
                                                         Queue
                                                         for(int tempcounterqueue=0;
                                                             tempcounterqueue<AX25Sync.
                                                             queue_final;tempcounterqueue++)

                                                         {
                                                             AX25Sync.
                                                                 receive_timecounter_telemetry_ranging
                                                                 [tempcounterqueue]=AX25Sync.
                                                                 receive_timecounter_telemetry_ranging
                                                                 [tempcounterqueue+1];
                                                         }
                                                     AX25Sync.queue_final--;
                                                     asmtimeflag=false;
                                                     //Console::log("BitCorrectionFlag");

                                                     }
                                                      uint8_t inBit = ((0x55) >> txBitIndex)&
                                                           0x01;
                                                      outputByte = outputByte | (inBit << (7-
                                                          i));
                                                      txBitIndex++;
                                                      //Console::log("Time at Transmit for 4
                                                          %d", MAP_Timer32_getValue(
                                                          TIMER32_1_BASE));
                                                      //Console::log("TX i: %d, txBitIndex:%d
                                                          ",i,txBitIndex);


                                                 }
                                                 else if(attachsinkmarkerflag==3)
                                                 {
                                                      uint8_t inBit = ((0x55) >> txBitIndex)&
                                                           0x01;
                                                      outputByte = outputByte | (inBit << (7-
                                                          i));
                                                      txBitIndex++;
                                                      //Console::log("Time at Transmit for 3
                                                          %d", MAP_Timer32_getValue(
                                                          TIMER32_1_BASE));
                                                 }
                                                 else if(attachsinkmarkerflag==2)
                                                 {
                                                      uint8_t inBit = ((0x55) >> txBitIndex)&
                                                           0x01;
                                                      outputByte = outputByte | (inBit << (7-
                                                          i));
                                                      txBitIndex++;
                                                      //Console::log("Time at Transmit for 2
                                                          %d", MAP_Timer32_getValue(
                                                          TIMER32_1_BASE));
                                                 }
                                                 else if(attachsinkmarkerflag==1)
                                                 {
                                                       uint8_t inBit = ((0x55) >> txBitIndex)
                                                           & 0x01;
                                                       outputByte = outputByte | (inBit <<
                                                           (7-i));
                                                       txBitIndex++;
                                                      //Console::log("Time at Transmit for 1
                                                          %d", MAP_Timer32_getValue(
                                                          TIMER32_1_BASE));
                                                 }
                                                 else if(attachsinkmarkerflag==5)
                                                 {
                                                     uint8_t inBit = (0x7E >> txBitIndex) & 0x01;
                                                     outputByte = outputByte | (encoder.txBit( inBit , false) << (7-i));
                                                     txBitIndex++;
                                                 }
                                                 else if(attachsinkmarkerflag==6)
                                                  {
                                                      uint8_t inBit = (0x7E >> txBitIndex) & 0x01;
                                                      outputByte = outputByte | (encoder.txBit( inBit , false) << (7-i));
                                                      txBitIndex++;
                                                  }
                                                 else if(attachsinkmarkerflag==7)
                                                  {
                                                      uint8_t inBit = (0x7E >> txBitIndex) & 0x01;
                                                      outputByte = outputByte | (encoder.txBit( inBit , false) << (7-i));
                                                      txBitIndex++;
                                                  }
                                                  else if(attachsinkmarkerflag==8)
                                                   {
                                                       uint8_t inBit = (0x7E >> txBitIndex) & 0x01;
                                                       outputByte = outputByte | (encoder.txBit( inBit , false) << (7-i));
                                                       txBitIndex++;
                                                   }
                                                 else
                                                 {
                                                     Console::log("Wrong ASMflag, shouldn't happen");
                                                 }





                                                 if(txBitIndex>=8)
                                                 {
                                                     txBitIndex = 0;
                                                     txIndex = 0;
                                                     attachsinkmarkerflag--;


                                                     if (attachsinkmarkerflag==0)
                                                     {   //Console::log("Time at ASM %d", MAP_Timer32_getValue(TIMER32_1_BASE));
                                                         timerflag=true;
                                                         activateattachsinkmarker=false;
                                                         asmtimeflag=true;
                                                         //txbitIndex2=0;
                                                         printflag=true;
                                                     }

                                                 }
          }
          else
          {

          if(txFlagQue) // if flags are queued and attachsinkmarker is not being transmitted, transmit
          {
              //check if a packet has finished transmitting and enough 7Es have passed.
```

```



                if(encoder.StuffBitsInBuffer)
                {  //send the Buffered Bit
                    outputByte = outputByte | (encoder.txBit(0, false) << (7-i));
                    Console::log("TX:␣SendingQued␣byte");

                }

                else
                {   //send part flagbit (7E)
                    uint8_t inBit = (0x7E >> txBitIndex) & 0x01;
                    outputByte = outputByte | (encoder.txBit( inBit , false) << (7-i));
                    txBitIndex++;
                }

                //check if txBitIndex is high enough to roll over to next byte
                if(txBitIndex>=8){
                    txBitIndex = 0;
                    txIndex = 0;
                    txFlagQue--;
//                    Console::log("TX: FLAG!");
                }

            }
            else if(txPacketsInBuffer() > 0) //no flags, but tx packet buffered
            {   //Console::log("TX: ActivatedAttachSinkMarker");


                if(encoder.StuffBitsInBuffer)
                { // send stuffing bit first
                    outputByte = outputByte | (encoder.txBit(0, true) << (7-i));
                }
                else
                { //send next bit of packet
                    uint8_t inBit = (txPacketBuffer[txPacketBufferReadIndex].getBytes()[
                        txIndex] >> txBitIndex) & 0x01;
                    outputByte = outputByte | (encoder.txBit( inBit , true) << (7-i));
                    txBitIndex++;
                }

                //check if bitindex is high enough to roll over next byte
                if(txBitIndex >= 8)
                {
                    txIndex++;
                    txBitIndex = 0;
//                    Console::log("TX: Byte: %d, Packet: %d", txIndex, txPacketBufferIndex);
                    //check if we are out of bytes in packet
                    if(txIndex >= (txPacketBuffer[txPacketBufferReadIndex]).getSize())
                    {
                        //roll over to next packet
                       txIndex = 0;
                       txPacketBufferReadIndex = (txPacketBufferReadIndex + 1) % TX_MAX_FRAMES
                           ;
//                        Console::log("R%d", txPacketBufferReadIndex);

                       // add ASM
                       activateattachsinkmarker=true;
                       attachsinkmarkerflag=8;

                       if(txPacketsInBuffer() > 0)
                       {
                           txFlagQue += 2; //if next packet available add, 2 flags to stack
                       }
                       else
                       {
                           txFlagQue += DOWNRAMP_BYTES;
                       }
                    }
                }
            }
            else // no flags are qued, and no data available, but radio is enabled, just idle
                 the flag
            {
                //shouldnt happen
                Console::log("onTransmit:␣Shouldnt␣happen␣-␣Nothing␣is␣Queued");
            }

}

       } // for(int i = 0; i < 8; i++)
      // if(timerflag==true)
        //      {   //Console::log("Time at Transmite from Receive %d", AX25Sync.
            timecounter_telemetry_ranging);


                  //AX25Sync.timecounter_telemetry_ranging=AX25Sync.
                      timecounter_telemetry_ranging-MAP_Timer32_getValue(TIMER32_1_BASE);
                  //Console::log("Time at Transmite for Variable %d", AX25Sync.
                      timecounter_telemetry_ranging);
          //        timerflag=false;
            //  }
        return outputByte;

    }
    else // if(txEnabled)
    {
        //TX shouldnt be asking for this function, so turn off and pass 0
        Console::log("onTransmit:␣Shouldnt␣Happen␣-␣Transmitter␣should␣be␣off");
        disableTransmit();
        return 0x00;
    }
};
```

Listing A.4: Delfi PQ Receive Time Tag Code

```
    bool AX25Synchronizer::queByte(uint8_t inByte){
    if(bytesInQue() < BYTE_QUE_SIZE - 1)
    {
        receiver_timeQue[byteQueWriteIndex]=receiver_timecounter_temp;
        byteQue[byteQueWriteIndex] = inByte;
        byteQueWriteIndex = mod(byteQueWriteIndex+1, BYTE_QUE_SIZE);

    }
    else{
        Console::log("[!!]");
    }
    return true;
}

int AX25Synchronizer::bytesInQue(){
    return mod(byteQueWriteIndex - byteQueReadIndex, BYTE_QUE_SIZE);
}

bool AX25Synchronizer::rxBit(){
    bool packetReceived = false;
    if(bytesInQue() <= 1){
        return 0;
    }

    uint8_t inByte = byteQue[byteQueReadIndex];
    receiver_timecounter_temp_2=receiver_timeQue[byteQueReadIndex];
    byteQueReadIndex = mod(byteQueReadIndex+1, BYTE_QUE_SIZE);

    for(int i = 0; i < 8; i++){
//Get time of each bit coming in for ranging.




        uint8_t inBit = encoder.NRZIdecodeBit(BitArray::getBit(&inByte, i, 8));
        inBit = encoder.descrambleBit(inBit);
        switch(this->synchronizerState){
            case 0:
                //flagDetect
                BitArray::setBit(&flagBuffer, flagBufferIndex, inBit == 0x01);
                flagBufferIndex = mod(flagBufferIndex + 1, 8);
                if(BitArray::getByte(&flagBuffer, flagBufferIndex - 8, 8) == 0x7E ){
                    //Flag Detected!
                    flagBuffer = 0;
                    flagBufferIndex = 0;
                    bitBuffer[0] = 0;
                    bitBufferIndex = 0;
                    synchronizerState = 1;
                }

                break;
            case 1:
                //checkForStart
                BitArray::setBit(bitBuffer, bitBufferIndex, inBit == 0x01);
                bitBufferIndex++;
                if(bitBufferIndex == 8){
                    if(bitBuffer[0] == 0x7E ){
                        //Another Flag, Reset bitBuffer
                        bitBufferIndex = 0;
                        bitBuffer[0] = 0;
                        synchronizerState = 1;
                    }else{
                        //First Byte in! continue to the wait for end state
                        synchronizerState = 2;
                        uint32_t temp=48000000/2400;
                        //Addition because the bytes come in upstream, this correction is for
                             the modem receiving and sending the bits, also in reverse
                            because the ith bit came in 7-i bits ago
                        timecounter_temp=receiver_timecounter_temp_2+(static_cast<uint32_t
                            >(7-i)*temp);
                        timecounter_temp2=receiver_timecounter_temp_2;
                        receiveindextemp=i;

                       // Console::log("BitReceiveCorrectionflag: %d %d", timecounter_temp,
                           static_cast<uint32_t>(7-i));

                    }
                }
                break;
            case 2:
                //Wait for ending State
                //Set next bit in Buffer
                BitArray::setBit(bitBuffer, bitBufferIndex, inBit == 0x01);
                bitBufferIndex++;
                //Console::log("BitReceivecheckflag: %d %d", timecounter_temp,i);
                if(bitBufferIndex >= 8 && BitArray::getByte(bitBuffer, bitBufferIndex - 8, 8*
                    BYTE_BUFFER_SIZE) == 0x7E ){
                    // End Sequence Byte Detected;
                    // Whether this is an actual packet or not, it doesnt matter, go back to
                        flagdetect afterwards
                    synchronizerState = 0;


                    // Get length of Packet, destuff and update length
                    int rxBits_n = bitBufferIndex - 8;
                    rxBits_n = this->encoder.destuffBits(bitBuffer, rcvdFrame.data, rxBits_n)
                        ;

                    //This could be a packet?
                    if(rxBits_n % 8 == 0 && rxBits_n/8 >= 18){

                        //packet has whole bytes and minimum length of 18 bytes!
                        //reverseOrder on all bytes
                        for(int p = 0; p < rxBits_n/8; p++){
                            this->rcvdFrame.data[p] = AX25Frame::reverseByteOrder(this->
                                rcvdFrame.data[p]);
                        }
                        this->rcvdFrame.packetSize = rxBits_n/8;

                        if(AX25Frame::checkFCS(this->rcvdFrame)){
                            packetReceived = true;
                            timecounter_holder_for_print_byte_timing=timecounter_temp2;
                            //for calculation of time spent in satellite
                            receive_timecounter_telemetry_ranging[queue_final]=timecounter_temp;
                            receiveindex=receiveindextemp;
                            queue_final++;
                            if(queue_final>10)
                            {
                                Console::log("Telemetry␣Timing␣Queue␣is␣overwhelmed,␣reduce␣data␣load␣or␣increase␣queue␣size");
                            }


                            //Console::log("BitCheckconfirm Flag: %d %d", timecounter_temp,i);

                            //Bit index correction for timecounter_telemetry_ranging


                            //Console::log("Time at receive for Byte %d", timecounter_temp);
                            //Console::log("Time at receive for Bit %d", timecounter_telemetry_ranging);

                            //Console::log("Time at Receive %d", MAP_Timer32_getValue(TIMER32_1_BASE));
                            //Console::log("Time at for variable %d", timecounter_telemetry_ranging);
                            //Console::log("time receive %d", MAP_Timer32_getValue(TIMER32_1_BASE));
                        }
                    }
                }
                if(bitBufferIndex > 255*8){
                    //This Message is too long, just return to flagdetect
                    synchronizerState = 0;
                }
                break;
        }
    }
    return packetReceived;
```

Listing A.5: Ground based correlation code

```
import numpy as np
import matplotlib.pyplot as plt
import gc

plt.close('all')
gc.collect()

def next_power_of_2(x):
    return 1 if x == 0 else 2**(x -1).bit_length()

#def main():
print("Correlator␣was␣Called")
#regenerate header and demodsignals at correct sample rate and sps before correlation
##All params to change when changing any parameter
bitrate=4800
sample_rate=500000
sps=int(sample_rate/bitrate)
#have 1 trial run, set this to the double the delay, 212027(2MHZ, 2400bps), 26635(250Khz,2400bps), 108624(1MHz,2400bps), 26635 for (250Khz,4800bps),
#20000 for (250Khz,9600bps), 55000(1Mhz, 19200bps), 40000(1MHz, 76800bps), if not given or wrong set very high run once and configure based on first logical measurement.
# with differential encoding 10000 for (250Khz,4800bps), 35000 fro 1200bps 250KHz, 3500 for 19200bps 250KHz  , 7000 for 500KHz, 38400bps
```

```
#trialrundelay=6000

min_delay=1000
#Currently, set by trial, 0.000003(2MHZ, 2400bps), 0.0015(250Khz,2400bps), 0.1(250KHz,9600bps
    )
#1(250KHz, 19200bps), 0.015(1Mhz, 19200bps) , 1.0(500KHz, 38400bps), 1.0(1MHz, 76800bps),
    0.00020(250KHz, 1200bps)
#with differential encoding 0.0015 for 250Khz, 4800bps,  0.0015 for 250Khz, 2400bps , 0.00015
     for 250KHz, 1200bps, 1 for 250KHz 19200 bps., 1 for 500KHz and 38400bps(sample rate
    doubles the sample amp but bit rate halves it so cancels out also oversampling seems to
    be needed for delfi to quickly receive the packet , )
cut_off_limit_correlation=0.00045
#important to ensure that two peaks dont compete with each other. Check the distance between
    peaks and select accordingly. (when hitting a single peak twice, it reflects as wrong
    readings in the middle of the delay values array, typically negative values)
#Also a slice length needs to be bigger than a single peak.
slice_length=30000
#Peak breaker should be bigger than size of a peak, exists to avoid double and wrong inputs
    from a peak
delaypeakbreaker=75000

plt.close('all')
#DemodRemodSignal=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\
    DemodRemodSignals', dtype=np.complex64, sep='', offset=0)

ReceivedSignal=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\
    ReceivedSignals3', dtype=np.complex64, sep='', offset=0)

SentSignal=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\SentSignals5',
     dtype=np.complex64, sep='', offset=0)

DemodSignalSent=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\
    Demodsignalsent', dtype=np.float32, sep='', offset=0)

DemodSignalReceived=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\
    DemodsignalReceived', dtype=np.float32, sep='', offset=0)

HeaderSignal=np.fromfile('D:\\GNURADIO_THESIS_MASTER\\CorrelationwithDelfiBoard\\
    DelfiPQHeader', dtype=np.float32, sep='', offset=0)
#original_length=HeaderSignal.shape[0]

original_length=0

max_delay_allowed=60000+original_length

#CorrelateHeaderSequence
# HeaderSignaltemp=HeaderSignal
# DemodSignalReceived=DemodSignalReceived[1:1000000]

HeaderSignal=np.pad(HeaderSignal,(0,DemodSignalReceived.shape[0]-HeaderSignal.shape[0]))

#Pad by 2 to avoid cyclic convolution
HeaderSignal=np.pad(HeaderSignal, (0,2*(HeaderSignal.shape[0])-HeaderSignal.shape[0]))
DemodSignalReceived=np.pad(DemodSignalReceived, (0,2*(DemodSignalReceived.shape[0])-
    DemodSignalReceived.shape[0]))

#Beyond this point, you can pad to next power of 2 to improve the speed of fft.

correlation_values_3 = np.fft.ifft(np.fft.fft(DemodSignalReceived) * np.conj(np.fft.fft(
    HeaderSignal)))


delay_header= np.argmax(np.abs(correlation_values_3))
i=0

if correlation_values_3[delay_header].real<0:
    DemodSignalReceived=-1*DemodSignalReceived

plt.figure(5)
plt.rcParams['interactive'] == True
plt.xlabel("Receive␣Stream␣(Samples)")
plt.ylabel("Correlation␣Peak␣value")
#plt.xlim((164400,164700))
plt.plot(np.abs(correlation_values_3), color="red")


HeaderSignal=np.roll(HeaderSignal,delay_header)
t = np.arange(0, HeaderSignal.shape[0], 1)
fig, axs = plt.subplots(2, 1)
axs[0].plot(t, HeaderSignal, t, DemodSignalReceived)
axs[0].set_xlim(0, HeaderSignal.shape[0])
axs[0].set_xlabel('Sample␣delay')
axs[0].set_ylabel('Sent␣Signal␣and␣Received␣Signal␣Demodded')
axs[0].grid(True)
#Plot correaltion peaks between the two signals
cxy, f = axs[1].cohere(HeaderSignal, DemodSignalReceived, 256, 1. / 1)
axs[1].set_ylabel('Coherence')

fig.tight_layout()
plt.show()

#print("Delay:"+str(delay))
#print("Demod_Delay:"+str(delay_demod))
print("Header_Delay:"+str(delay_header))

#Write Result to file
#with open("Results.txt", "a") as att_file:
#    att_file.write(str(delay_header)+"\n")

#Part of associate delay with Sent Signal
#Signal Length is sps*24*8 of data sandwiched between sps*50*8 of Pilot and sps*50*8 of tail
Signal_length=sps*24*8
PT_length=sps*450*8
offset=Signal_length+2*PT_length
i=0
SentSignal_location=[]
#Find out location of Signal
while offset*i<SentSignal.shape[0]:
    SentSignal_location.append((offset*i+PT_length))
    i=i+1

ReceivedSignal_Location=[]

#Slice Length to see how much of array to slice, smaller values have lesser chance of having
    multiple peaks within the same slice.
#If an unlucky event happens where the same peak is in two slices, then change the slice
    length

p=0
delay_peak_previous=0

#Find out location of Received Peaks from Correlation_values_3
while p+slice_length<correlation_values_3.shape[0]:
    correlation_array_slice=correlation_values_3[p:p+slice_length]
    delay_peak= np.argmax(np.abs(correlation_array_slice))
   #Set peak as 0.0002 for 2million sample rate and 0.1 for 250000 sample rate
    if abs(correlation_array_slice[delay_peak])>cut_off_limit_correlation:

        if abs(p+delay_peak-delay_peak_previous)<delaypeakbreaker:
            if correlation_values_3[delay_peak_previous]<correlation_values_3[delay_peak]:
                ReceivedSignal_Location[-1]=p+delay_peak
            p+=slice_length
            continue
        ReceivedSignal_Location.append(p+delay_peak)
        delay_peak_previous=p+delay_peak

    p+=slice_length
ReceivedSignal_Location=[x+original_length for x in ReceivedSignal_Location]

#Finally Iterate through every Sentsignal location and associate it with ReceivedSignal
    Location to find the delays
#Indices for both arrays, asynchronous because we want to detect missed packets using a max
```

```
    delay threshold.
j=0
k=0

delay_values = []
j = len(ReceivedSignal_Location) - 1

while j >= 0:
    if ReceivedSignal_Location[j] - SentSignal_location[j] > max_delay_allowed:
        del SentSignal_location[j]
    elif ReceivedSignal_Location[j] - SentSignal_location[j] < 0:
        del ReceivedSignal_Location[j]
    j -= 1

    delay_values.append(ReceivedSignal_Location[j]-SentSignal_location[j])
    print("identifier")

import struct

def invert_decimal_by_binary(decimal_number):
    # Convert decimal to binary and remove the '0b' prefix
    binary_representation = bin(decimal_number)[2:]

    # Invert each bit using XOR (^) with 1
    inverted_binary = ''.join(['1' if bit == '0' else '0' for bit in binary_representation])

    # Convert the inverted binary back to decimal
    inverted_decimal = int(inverted_binary, 2)

    return inverted_decimal

Time_value_final=[]
Time_value_final_inverted=[]


def decode_bits_to_integer(bits, endianness='little'):
    # Ensure length is a multiple of 8
    remainder = len(bits) % 8
    if remainder != 0:
        bits.extend([0] * (8 - remainder))

    # Convert the list of bits to an integer
    decoded_value = int(''.join(map(str, bits)), 2)

    # Swap bytes if endianness is big-endian
    if endianness == 'big':
        decoded_value = int(format(decoded_value, '032b')[::-1], 2)

    return decoded_value


def binary_to_decimal(binary_list):
    decimal = 0
    power = len(binary_list) - 1

    for bit in binary_list:
        decimal += bit * (2 ** power)
        power -= 1

    return decimal


# Extract differentially encoded telemetry time data from stream - Experimental, not working
    as of 12/11/2023
for temp in ReceivedSignal_Location:
    Start = temp + 4 * sps * 8
    Time_stream = DemodSignalReceived[Start+3*sps : Start + 35 * sps]

    decimal_value = 0
    samples_per_bit = sps  # Adjust the number of samples per bit based on your requirements
    threshold = 0  # Adjust this threshold factor based on your signal characteristics
    time_bit_array = []

    # Iterate through the stream in chunks
    for i in range(1, len(Time_stream), samples_per_bit):
        # Extract a chunk of the demodulated signal
        chunk = Time_stream[i - 1:i + samples_per_bit]  # Consider transitions between bits
        chunkvalue=np.average(chunk)
        # Thresholding to get bits
        bits = int(np.mean(chunk) > threshold)

        # Process the bits as needed (e.g., store, analyze, etc.)
     #   print(bits)
        time_bit_array.append(bits)

    #print(time_bit_array)
    #byte_data = bytes(time_bit_array)  # Replace this with your byte data and endianness

    #decoded_value = sum([decode_bits_to_integer(time_bit_array[i:i+8], endianness='little')
        << (i*8) for i in range(4)])
    #decoded_value = decode_bits_to_integer(bits, endianness='little')
    decimal_value = binary_to_decimal(time_bit_array)
    Time_value_final.append(decimal_value)
    inverted_result = invert_decimal_by_binary(decimal_value)
    Time_value_final_inverted.append(inverted_result)



#Write Result to file
with open("Results.txt", "a") as att_file:
    for item in delay_values:
        att_file.write(str(item)+"\t")
    att_file.write(str(item)+"\n")





gc.collect()
```

Listing A.6: Delfi PQ Command Generation Code

```
import NRZI_G3RUH_Encoder
from BitLib import *
import AX25_Encoder

ax25Encoder = AX25_Encoder.AX25Encoder()
g3ruhEncoder = NRZI_G3RUH_Encoder.G3RUHEncoder()
pilot_seq = (20*[int("0x7E", 16)])
#pilot_seq= 1*[int("0x7E", 16), int("0x7E", 16)]
tail_seq = 10*[int("0x7E", 16), int("0x7E", 16)]

#88 98 8C 92 A0 A2 E0 A0 92 72 88 AA A8 61 03 F0 01 01 03 01 A3 16
#88 98 8C 92 A0 A2 E0 A0 92 72 88 AA A8 61 03 F0 01 01 11 01 C6 07
#pilot_seq=[]
#tail_seq=[]
#Delfi PQ side
#8E A4 9E AA 9C 88 E0 88 98 8C 92 A0 A2 61 03 F0
#data=[int("0x8E", 16), int("0xA4", 16),int("0x9E", 16),int("0xAA", 16),int("0x9C", 16),int
    ("0x88", 16),int("0xE0", 16),int("0x88", 16),
#      int("0x98", 16),int("0x8C", 16),int("0x92", 16),int("0xA0", 16),int("0xA2", 16),int("0
    x61", 16),int("0x03", 16),int("0xF0", 16),
#      ]

#data with header for ground side commanding
data=[int("0x88", 16), int("0x98", 16),int("0x8C", 16),int("0x92", 16),int("0xA0", 16),int("0
    xA2", 16),int("0xE0", 16),int("0xA0", 16),
      int("0x92", 16),int("0x72", 16),int("0x88", 16),int("0xAA", 16),int("0xA8", 16),int("0
          x61", 16),int("0x03", 16),int("0xF0", 16),
      int("0x01", 16),int("0x01", 16),int("0x11", 16),int("0x01", 16),int("0xC6", 16),int("0
          x07", 16)]

```

```
#Header for Attach sink marker : 1A CF FC 1D
#data=[int("0x1A", 16), int("0xCe", 16),int("0xec", 16), int("0x1D", 16)]

##data=data=[int("0x55", 16), int("0x55", 16),int("0x55", 16), int("0x55", 16), int("0x55",
    16)]
print(data)
data.extend(ax25Encoder.CalculateFCS(data))
print(data)
txBits = list2bits(pilot_seq)

txBits += (ax25Encoder.StuffBits(list2bits(flipbytes(data))) + list2bits(tail_seq))

txBytes = bits2bytes(txBits)
txBytes = txBytes*1
print(txBytes)

#txBits = [g3ruhEncoder.NRZIEncodeBit(g3ruhEncoder.ScrambleBit(x)) for x in list2bits(txBytes
    )]

#No Scrambling code
txBits = [g3ruhEncoder.NRZIEncodeBit(x) for x in list2bits(txBytes)]

#Scrambling code
#txBits = [g3ruhEncoder.NRZIEncodeBit(g3ruhEncoder.ScrambleBit(x)) for x in list2bits(txBytes
    )]

txBytes = bits2bytes(txBits)
#txBytes = txBytes*2
print('\n\n\n')
print(txBytes)
```

B
Thesis Planning and Ranging Schematic

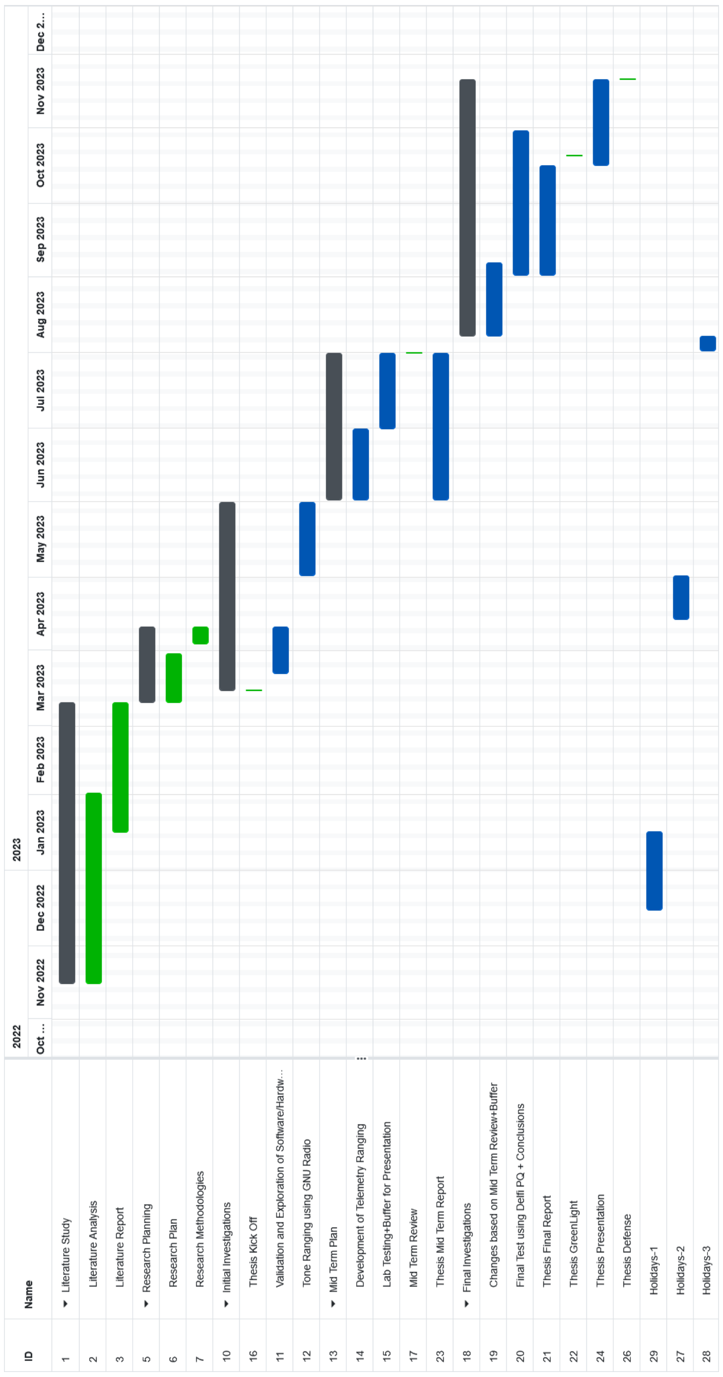
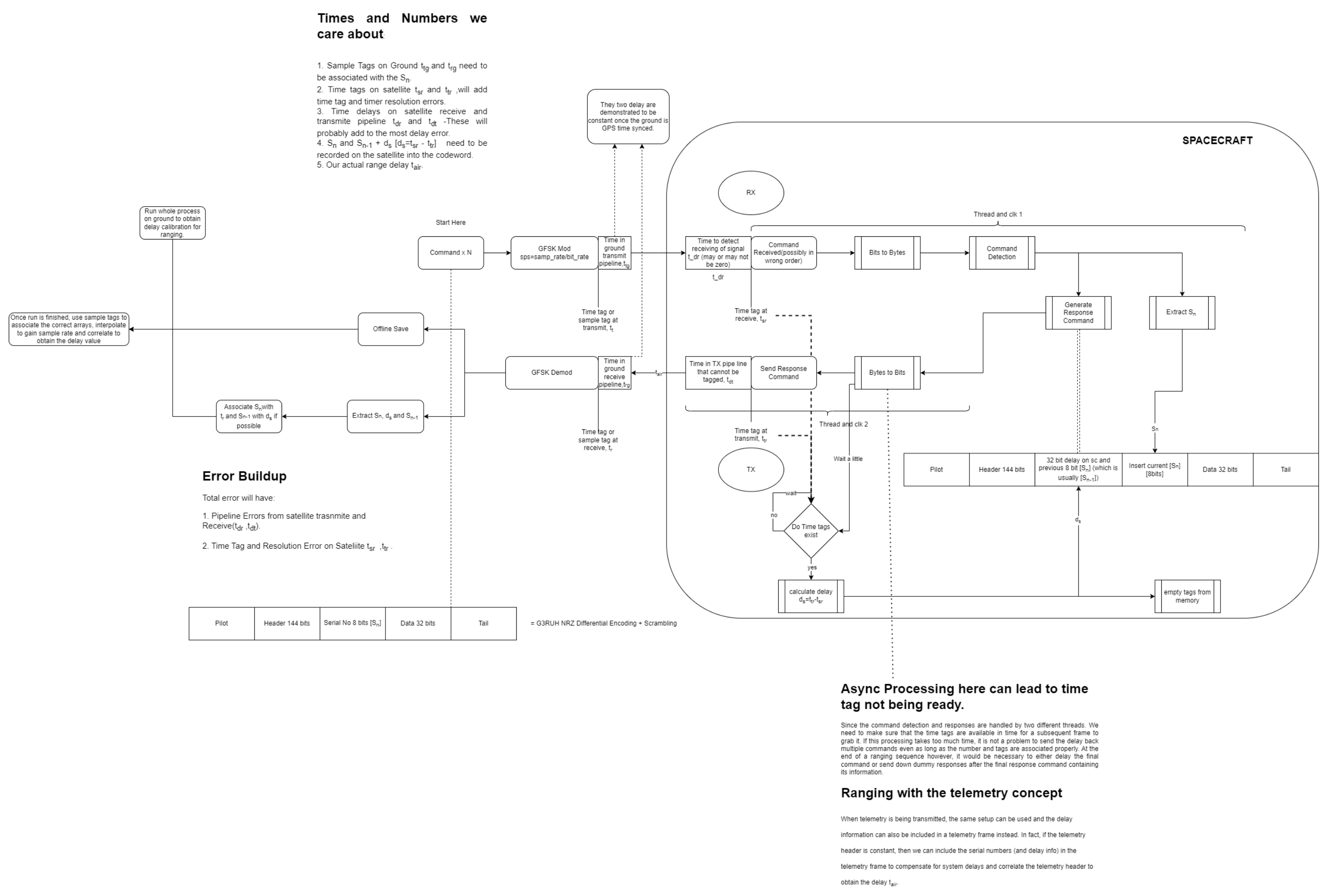


Figure B.1: Gantt Chart for Thesis Planning


SPACECRAFT
RX
TX
Start Here
Command x N
GFSK Mod sps=samp_rate/bit_rate
Time in ground transmit pipeline,t_{tg}
Time tag or sample tag at transmit, t_t
They two delay are demonstrated to be constant once the ground is GPS time synced.
GFSK Demod
Time in ground receive pipeline,t_{rg}
Time tag or sample tag at receive, t_r
t_{air}
Offline Save
Once run is finished, use sample tags to associate the correct arrays, interpolate to gain sample rate and correlate to obtain the delay value
Extract Sn, d_s and S_{n-1}
Associate $S_n$with t_r and Sn-1 with d_s if possible
Run whole process on ground to obtain delay calibration for ranging.
Pilot
Header 144 bits
Serial No 8 bits [S_n]
Data 32 bits
Tail
= G3RUH NRZ Differential Encoding + Scrambling
Thread and clk 1
Time to detect receiving of signal t_dr (may or may not be zero)
t_dr
Command Received(possibly in wrong order)
Bits to Bytes
Command Detection
Generate Response Command
Extract S_n
Time tag at receive, t_{sr}
Time in TX pipe line that cannot be tagged, t_{dt}
Send Response Command
Bytes to Bits
Thread and clk 2
Time tag at transmit, t_{tr}
Wait a little
wait
no
Do Time tags exist
yes
calculate delay d_s=t_{tr}-t_{sr}
empty tags from memory
d_s
Sn
Pilot
Header 144 bits
32 bit delay on sc and previous 8 bit [S_n] (which is usually [S_{n-1}])
Insert current [Sn] [8bits]
Data 32 bits
Tail

Times and Numbers we care about

1. Sample Tags on Ground t_{tg} and t_{rg} need to be associated with the S_n.
2. Time tags on satellite t_{sr} and t_{tr} ,will add time tag and timer resolution errors.
3. Time delays on satellite receive and transmite pipeline t_{dr} and t_{dt} -These will probably add to the most delay error.
4. S_n and S_{n-1} + d_s [d_s=t_{sr} - t_{tr}] need to be recorded on the satellite into the codeword.
5. Our actual range delay t_{air}.

Error Buildup

Total error will have:

1. Pipeline Errors from satellite trasnmite and Receive(t_{dr} ,t_{dt}).

2. Time Tag and Resolution Error on Sateliite t_{sr} ,t_{tr} .

Async Processing here can lead to time tag not being ready.

Since the command detection and responses are handled by two different threads. We need to make sure that the time tags are available in time for a subsequent frame to grab it. If this processing takes too much time, it is not a problem to send the delay back multiple commands even as long as the number and tags are associated properly. At the end of a ranging sequence however, it would be necessary to either delay the final command or send down dummy responses after the final response command containing its information.

Ranging with the telemetry concept

When telemetry is being transmitted, the same setup can be used and the delay information can also be included in a telemetry frame instead. In fact, if the telemetry header is constant, then we can include the serial numbers (and delay info) in the telemetry frame to compensate for system delays and correlate the telemetry header to obtain the delay t_{air}.

Figure B.2: Improved Telemetry Ranging using time tagging Concept for Delfi PQ

C Error Sources and Performance for Ranging

This chapter delves deeper into the sources of errors that arise in a ranging system. It discusses the offset inducing systematic errors, which shift the determined range by a constant value and jitters, which shift the determined range by a different value each time. Thus, it maps the sources of errors that would be visible in range process. This chapter is in the appendix, as most of the jitter errors are too small to be visible in the thesis experiments, mainly due to the resolution of the system. However, they still contribute to the overall errors and will be visible at some point in the future when optimizing the ranging method. The systematic errors will only be seen when conducted tests to space and in space and need to be compensated for to achieve higher range accuracies. Therefore, this is meant to be a jump start to the later error mitigation efforts.

C.1. Systematic Errors (or Biases)

There are a number of source of systematic errors in the range measurement process [36], which add to the delay in the range phase measurement and to the range error. These errors can be calibrated for through appropriate techniques, which are performed before and after a spacecraft tracking pass:

C.1.1. Clock Instability

The instability in the clock doing the range measurements contributes to the error in the ranging process. The effect of clock instability on two-way range data is

$$\Delta\rho = \sqrt{2} \times c \times \tau \times \sigma_y(\tau) \tag{C.1}$$

where c is the speed of light τ is the Round Trip Light Time (RTLT) and $\sigma_y(\tau)$ is the stability of the measuring clock. The stability is measured through the use of ADEV, which is a statistical measure used to estimate the stability of noisy processes, which apply especially to clocks based on frequency standards (originating in [2]). Allan variance is defined as

$$\sigma_y^2(\tau) = E(\sigma_y^2(2, T, \tau)) = \frac{1}{2} * E((\bar{y}_{n+1} - \bar{y}_n)^2)) \tag{C.2}$$

where E is the expectation operator, which is the mean of all the values in the set.
and $\bar{y}_n$ is defined as

$$\bar{y}_n = \frac{x(nT + \tau) - x(nT)}{\tau} \tag{C.3}$$

where x(t) is time series containing all the data points for which the variance is estimated and
where T is the dead time for which no measurements are taken and
τ is the time between the two measurements. Therefore, half the mean of the individual variances gives the Allan variance.

ADEV is given as the square root of Allan variance

$$ADEV = \sigma_y(\tau) = \sqrt{\sigma_y^2(\tau)} \tag{C.4}$$

The ADEV of clocks is defined as being over a period of time, such as Deviation of $1\times10{-}15$ per day which is the value for hydrogen maser based atomic clocks [69]. Whereas crystal oscillators typically used have an ADEV of about $1\times10{-}12$ per day. This is sufficiently low error for ranging happening over just minutes or hours. However, this over time builds up a difference in time or offset between multiple clocks.

C.1.2. Clock Offset

In three-way and one way ranging, the clock offset between the two clocks used directly translate to a larger error in the range estimate as

$$\Delta\rho = c \times \Delta T \tag{C.5}$$

Time Synchronization between clocks is of significance for data transfer and in three-way ranging schemes, For ranging especially, the error in the time sync directly translates to a large error in the ranging as a factor of speed of light.

In two-way ranging, the transmitter and receiver radios must be synchronized to ensure that they start at the same time and are running at the same speed. If not, they will misinterpret the incoming data, as the difference in start time and speed will translate into an offset between the sent and received data. That is, say the transmitter clock thinks it is starting at 10:00:01 (HH:MM:DD), and the receiver clock thinks it is starting at 10:00:01, but actually start at 10:00:02, the receiver data will be offset from the transmitter clock by 1 sec. If this is not corrected for, it will lead to an incorrect range reading

Thus, the time between the clocks in the different stations used for ranging must be synchronized upto the ranging requirement. Meaning, an error is always going to exist, but the acceptable amount of this error depends on the mission requirements and varies from case to case. There are a number of ways in which this synchronization can be done. Synchronization can be achieved by using information from a reference clock over the internet using either a GPSDO or Precise Time Protocol (PTP) or European Organization for Nuclear Research (CERN) White Rabbit protocol. Other protocols exist such as the Network Transfer Protocol(NTP) which is used as a fundamental protocol in many of Earth applications but has also flown in space missions[47]. For deep space missions, the Proximity 1 protocol was defined for an Earth Mars Space Link[55]. A further improvement was suggested in the Proximity 1 interleaved protocol(PITS)[85].

Apart from network based time synchronization methods, other local methods exist such as daisy or star chaining, where the time of one local clock is sent to the other clocks for synchronization[7]

C.1.3. Time Tagging

The time at which a certain phase is measured must also be tagged in order for the ranging to be done. This time tagging process is not instantaneous, but takes a small amount of time and has two effects. This first effect only has a small result on the range measurement, in that the distance travelled by the satellite during the time tagging period cannot be accounted for the in the ranging process (unless there is an accurate satellite model). Therefore, time tagging accuracies in the microseconds are typically acceptable, as the change in range during this period is sufficiently small[32].

An error in the time tagging causes an error proportional to the range rate of the satellite, as

$$\Delta\rho = \Delta T \times \dot{\rho} \tag{C.6}$$

where $\dot{\rho}$ is the range rate.

In the second effect, an additional timing error is introduced in the system as a function of the reference clock period. The quantization noise (minimum resolvable time difference) for the time tag is given as the period of the reference clock,

$$t_R = \frac{1}{f_{ref}} = \frac{1}{80 * 10^6} = 12.5 * 10^9 = 12.5ns \tag{C.7}$$

where phases are time tagged using a 80MHz reference for DSN. In order to reduce this further, a least squares fit assuming linear time is used to obtain a smoothed standard deviation of 1ns. Recently, with the CERN white rabbit technology, time tagging accuracies in the ps are also possible[64].

C.1.4. Clock Rate Offset

Clock rate offset is the error in the estimated clock rate. This is different from clock instability in this being the difference in the error in the rate instead of being due to missed oscillations. An error of $\Delta\dot{T}$ results in a range error of,

$$\Delta\rho = c \times \tau \times \Delta\dot{T} \tag{C.8}$$

this is also typically small (11 cm for 1 hour RTLT) with clock rate offsets being in the range of 10^{-13}.

C.1.5. Interplanetary/Ionospheric Plasma:

The plasma in the solar system cause a delay in the radio data dependent on the frequency and the total electron count in the system. This delay can be imagined as being caused due to a changing refractive index of the medium due to the presence of electrons. This changing refractive index is caused by an oscillation of electrons due to the wave producing a secondary electric field. The final velocity of the wave would be the vector sum of the initial wave and the new wave produced due to oscillation of electrons. This increases the speed of the phase components of the wave, but reduces the speed of the group component of the signal (the speed of the highest amplitude component). The changed refractive index is given as [1]

$$n = \sqrt{1 - \frac{80.6 \times N_e}{f^2}} \tag{C.9}$$

For very high frequencies, this can be approximated as[1]

$$n = 1 - \frac{40.3 \times N_e}{f^2} \tag{C.10}$$

where N_e is the free electron density of the electrons in the plasma and f is the frequency of the signals in Hertz. This causes a phase velocity, which is the speed of the phase of electromagnetic waves, to move faster than the speed of light:

$$v_p = c/(1 - \frac{40.3 \times N_e}{f^2}) \tag{C.11}$$

which is greater than c. The group velocity of

$$v_g = c \times (1 - \frac{40.3 \times N_e}{f^2}) \tag{C.12}$$

Taking range to be calculated using a constant velocity, the additional range the wave travels due to the reduced group velocity is

$$\delta l = \frac{40.3}{f^2} \int N_e \, dl = \frac{40.3}{f^2} TEC \tag{C.13}$$

where TEC is the total electron count[1]. The TEC is due to ionospheric and solar (interplanetary) plasma These contributions in the order of cm for ionospheric plasma and 1-70 m for solar plasma in the X band[69]. Using higher band radio frequencies can vastly reduce the error as seen in the equation. Using multiple radio frequency links, this contribution can be exactly estimated for a triple frequency radio link[70] and nearly completely eliminated for a dual frequency radio link [69](the different effect of electrons on the different frequencies allow us to estimate TEC). However, for single frequency links, commonly seen in satellites, alternative techniques such as Differential Range Vs Integrated Doppler (DRVID) need to be used[69, 53].

C.1.6. Tropospheric Delay

The presence of clouds and other material in the troposphere of the Earth causes a tropospheric delay in the radio signal. This is dependent on the elevation angle of the satellite with respect to the ground station.

$$\Delta\rho = \Delta\rho_d + \Delta\rho_w \tag{C.14}$$

where

$$\Delta\rho_{w/d} = z_{w/d} \times F_{w/d}(E) \tag{C.15}$$

$F_{w/d}$ is the zenith mapping function, one popular value of which is given by $1/sin(E)$. [69] More accurate calibrations can be done through line of site measurements from water vapor radiometer(WVR) or Fourier transform spectrometer, as well as GPS enabled calibrations. Tropospheric delay is non-dispersive, which means it does not depend on the frequency of the radio waveThus,us this delay is the same for all frequencies. [1]

Another Model of estimating tropospheric is given in [1] where the refractive indices of dry (N_d) and wet troposphere(N_w) are measured individually, and the overall tropospheric delay is given as

$$\delta rr_{trp} = \int_0^h N_d(h_d) \times (1 - h/h_d)^4\, dh + \int_0^h N_w(h_w) \times (1 - h/h_w)^4\, dh \tag{C.16}$$

where $h = 12km$ for wet component, $h = Height\ of\ Stratosphere$ and $N = 10^6 \times (n - 1)$ where n is the index of refraction.

Popular models that predict this delay based on tropospheric models can be found in [35, 39].

C.1.7. Instrument Delays

Instrument delay: Delay in the antenna and cables of satellites and ground station along with processing delay, this usually can be calibrated with offline measurements. The ground station instrument delay is measured with an internal loop every pass, whereas the spacecraft delay is calibrated beforehand[16]. They can also be estimated offline using different techniques and updated at occasional intervals[1]. The spacecraft side delay can also be calibrated for by using range data from a sufficient number of orbit passes. This, however, may not be feasible for a deep space mission.

C.1.8. Platform Errors

Any uncertainty in the position of the ground station platform directly adds to errors in the ranging process. Thus, the exact position and orientation of the Earth and the Ground Station with respect to the spacecraft must be tracked. Several models exist to track these errors, but they drift over time, which adds to the overall error. Platform errors are mainly due to Earths rotation, nutation, precession, and ground tides. Further, the non-rigid nature of the measuring antenna also add to these errors[36, 69]. The method of linking processing data from multiple antennas can be used using an antenna in a dry environment which will minimize this error[4].

C.2. Random Errors

Random errors are errors in the ranging process that cannot be accurately predicted. The primary source of random errors is thermal noise in the ground receiving chain. The receivers are sensitive to these noises, which shift the measured phase of the received signal ahead or behind the true phase of the signal.

C.2.1. Thermal Noise

Thermal noise is the noise generated due to the temperature exciting the electrons in a circuit, which can typically be the limiting factor for a measuring equipment[1]. For Sequential Ranging, it depends on the signal-to-noise ratio $\frac{P_r}{N_0}$, the highest frequency component of the ranging signal f_t and the integration time T[40]:

$$\sigma_r = \frac{c}{16 \times f_t \times \sqrt{\frac{P_r}{N_0} \times T}} \tag{C.17}$$

[1]https://en.wikipedia.org/wiki/Johnson-Nyquist$_{noise}$

For Pseudo Noise ranging systems, the range jitter due to thermal noise is

$$\sigma_{Range\_CTL\_sq\_sq} = \frac{c}{8 \times f_{RC}\sqrt{\frac{B_L}{(P_R C/N_0)}}} \tag{C.18}$$

in CCSDS informational report [12], the range jitter is described for different use cases and is mainly due to the chip tracking loop(CTL) used. The noise in the signals directly translate to a larger jitter error in the correlation and tracking loops, which is the limiting factor for the ranging.

For telemetry ranging, the equation for thermal jitter is not present in the surveyed literature. The papers use simulation or experimental data to obtain the performance and compare it to the cramer rao bound for the estimator (which can be a correlator or a tracking loop). Assuming a Gaussian white noise, the Cramer Rao lower bound for the tracking loop can be derived. The lower bound for a DTTL loop used on the ground is:

$$\sigma_d = \frac{W \times B_L}{2 \times S_L \times \frac{P}{N_0}} \tag{C.19}$$

where S_L is the squaring loss as given in equation 12 of [75]. For high SNRs $S_L- > 1$, therefore the CRB is

$$\sigma_d^2 = \frac{W \times B_L}{2 \times \frac{P}{N_0}} \tag{C.20}$$

For Low SNRS, CRB is approximately

$$\sigma_d^2 = \frac{W \times B_L}{2 \times R_s \times \frac{P}{N_0}} \tag{C.21}$$

C.3. Total Error Budget

All the systematic errors discussed add up to the error in the range. These errors are usually represented in the form of an error budget.

The total systematic error is given as the root of the sum of square of individual errors as the errors are uncorrelated.[1]

$$\Delta\rho = \sqrt{\Sigma\Delta\rho_i^2} \tag{C.22}$$

where $\Delta\rho$ is the total range systematic error.

The error budget of the Rosetta mission are given in Table C.1 and we can see that the systematic biases can add up to be significant, especially at the lower SEPs. Therefore, it is necessary to calibrate our range measurements for the major systematic errors.

Table C.1: Rosetta Range Residuals [36].

Source	Range Error(cm)	
	SEP 15 deg	SEP 30 deg
Interplanetary Plasma	800	350
Troposphere(Wet Only)	20	-
G/S Bias	150	-
Station Location, Earth Tides, EOP	9	-
Total	814	381

The approximate ranging error values seen in space missions can be seen in Table C.2. This table is taken from an earlier work, so the improvements over the decades cannot be seen.

Table C.2: Radiometric measurement system error characteristics[69]

Error Source	Magnitude		
	1980 S-Band	1992 X Band	2000 X Band
Random Error for 60s avg	200cm	60cm	60cm
Instrument Bias	5m	5m	2m
Instrument Stability at 8h	10^{-13}	10^{-14}	10^{-14}
Station Location			
Spin Radius	100cm	10cm	3cm
Longitude	100cm	10cm	3cm
Baseline components	30cm	5cm	2cm
Earth Orientation(1 day prediction)	100cm	30cm	7cm
Earth Orientation(After the fact correction)	20cm	3cm	1cm
Troposphere			
Zenith Bias	4.5cm	4.5cm	1cm
Line-of-sight fluctuation (over 10 min at 15 deg elevation)	1cm	1cm	1cm
Ionosphere(line-of-sight over 10 deg)	100cm	3cm	3cm
Solar Plasma			
20 - deg SEP angle			
Total line-of-sight	229m	17m	17m
Drift over 8 h	15m	115cm	115m
Station differenced	7cm	0.5cm	0.5cm
Solar Plasma			
180 - deg SEP angle			
Total line-of-sight	16m	116cm	116cm
Drift over 8 h	2m	15cm	15cm
Station differenced	1cm	0.1cm	0.1cm
Station clock			
Epoch	1 s	1 s	1 s
Rate	10^{-12}	5×10^{-14}	5^{-14}
Stability @1000s	10^{-12}	10^{-15}	10^{-15}

The systematic errors for a GPS ranging process is given in Table C.3

Table C.3: GPS Systematic Errors[1]

Error	Magnitude
Ephemeris Error	2m
Clock Error	2m
Ionosphere Error	5m
Tropospheric Error	1m
Reciver Noise	1m
Multipath, etc	1m
Total	6m

The range bias drift over a single tracking pass for Cassini was 1-2 m.[36]. The systematic range errors will be encountered when conducted a flight test as they are due to external phenomena. The random errors will be present in all the tests.