Location techniques for pico- and femto-satellites, with applications for space weather monitoring

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

Ian Michael Griffiths MPhys

Department of Engineering
University of Leicester

2017

Abstract

Space weather phenomena have a significant impact on satellite communications but are not well understood. *In-situ* measurements of the ionospheric environment would significantly improve the understanding of the origins and progressions of these phenomena. Whilst previous scientific satellites have measured the ionospheric plasma, they only provide a limited view due to their small number. It has previously been suggested that a swarm of femto-satellites (PCBsats) could be used to collect high quality temporal and spatial measurements, whilst being financially effective.

To give the measurements any scientific value, the location and time of each measurement needs to be accurately recorded. The PCBsat prototype used a solution that, due to export requirements and fundamental limitations with the device, would not be capable of working in space. Several location and timing solutions have been investigated, with none matching the precision, accuracy, power consumption and physical size of a GNSS receiver (i.e. a receiver of GPS, GLONASS, Galileo etc. signals). To further reduce the power consumption, a novel distributed GNSS receiver has been designed and built, where the largest computational burden (calculating the receivers position) is offloaded to a relaying node. This use of distributed computing has been shown to reduce the power consumption of the receiver by between $5.6\,\%$ and $13.3\,\%$ - which is equivalent to between 2 and 5 times the power consumption of the PCBsat's main processor.

In addition to this, this novel approach has the additional benefit of being used in a hybrid scheme. Where information required to calculate a receiver's position is stored so that it can be used with higher precision ephemerides that are publicly available but are delayed by up to three weeks. This has many applications as it can increase the utility of collected data, at a reduced cost.

As the intended femto-satellite application relies on a link to relaying satellites, the dynamics, in particular the dispersion, of the intended constellation needs to be known. This has been modelled using a novel orbit simulator. The orbit simulator is the first of its kind to model multidimensional free molecular drag to simulate the effects of the low density atmosphere on a satellite. This allows the dispersion of a constellation of satellites to be investigated with maximum separations for the PCBsat being presented.

Acknowledgements

Many people have helped and assisted me over the years with this work and whilst I would like to thank all the current, and past, members of the embedded systems and communications group, there are some that need to mentioned by name. I would like to thank Tanya, my supervisor, for allowing me the space and freedom to conduct my own research, as well as for her guidance. I would also like to thank both Fayyaz and Felix for the numerous conversations that ultimately helped shape this work.

It is beyond necessary to thank Andrew Addy and the rest of the team at Spirent Communications, who arranged for the University to be loaned one of their signal simulators. The use of the simulator helped this research to such a great extent that I am incredibly indebted to you.

Most importantly, I would to thank my fiancée Claire, for her encouragement and support, especially on the many occasions where I have been rambling about nonsense in the small hours of the morning.

Contents

List of Tables

List of Figures

Listings

Abbreviations

ADC	Analog to Digital Converter
AHB	Advanced High-performance Bus
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface
BPSK	Binary Phase-Shift Keying
CAN	Controller Area Network
CCD	Charge-Coupled Device
CDMA	Code Division Multiple Access
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DLL	Delay Locked Loop
DMA	Direct Memory Access
DMIPS	Dhrystone MIPS
DSP	Digital Signal Processing
DSSS	Direct-Sequence Spread Spectrum
EM	Electro-Magnetic
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
I^2C	Inter-Integrated Circuit
LED	Light Emitting Diode
LEO	Low Earth Orbit
LPDDR	Low-Power Double Data Rate
LUT	Look-Up Table
MAC	Media Access Control
MIPS	Million Instructions Per Second

MMC	MultiMedia Card
MQFP	Metric Quad Flat Package
NCO	Numerically Controlled Oscillator
OBC	On-Board Computer
PATA	Parallel AT Attachment
PCB	Printed Circuit Board
PLC	Power Line Cycles
PLL	Phase Locked Loop
PRN	Pseudo-Random Noise
PSRR	Power Supply Rejection Ratio
RAM	Random Access Memory
SATA	Serial AT Attachment
SD	Secure Digital
SDHC	Secure Digital High Capacity
SDXC	Secure Digital eXtended Capacity
SEB	Single Event Burnout
SEL	Single Event Latch-up
SEU	Single Event Upset
SNR	Signal to Noise Ratio
SoC	System on-Chip
SPI	Serial Peripheral Interface
SVD	Singular Value Decomposition
TID	Total Ionising Dose
TLE	Two Line Elements
UART	Universal Asynchronous Receiver/Transmitter
UHF	Ultra High Frequency
USB	Universal Serial Bus
VHF	Very High Frequency

Chapter 1

Introduction

Femto- and pico-satellites, satellites with wet masses between $10\,\mathrm{g}$ and $1\,\mathrm{kg}$ are becoming increasingly more useful for many scientific missions. In particular, their lower manufacturing and launch costs, which are predominately determined by weight, make them very attractive for multi-node satellite sensor networks. However, being able to accurately and reliably locate each individual node is critical to many tasks, but there is currently a lack of systems that are able to do this on a very restricted power budget, such as that of a femto-satellite. Therefore, the aim of this research was to devise a method of locating a femto-satellite that would fit within these restrictive power constraints.

This work investigates and evaluates the available technology and, consequently, designs and implements a novel technique for GNSS receivers that can reduce the power consumption on a node by between $5.6\,\%$ and $13.3\,\%$. In addition to this, a novel hybrid design is suggested that can, with a small overhead, increase the accuracy of the receiver's calculated positions, and an information based acquisition method is presented, which, through the use of successive acquisition scans, can acquire visible satellites which are below the search threshold.

To complement this, a novel orbit simulation is presented that can accurately model the drag effects on femto-satellites, with the maximum dispersion amount for a femto-satellite swarm being given.

This chapter discusses the background of this work, with the second chapter describing the mission concept. The third chapter discusses the design of GNSS receivers, the novelty in the distributed design and the initial prototypes, with the fourth chapter detailing the tools that were constructed for this work. The fifth and sixth chapters present the software and hardware versions of the receiver design and the seventh chapter details the orbit simulation that was designed to model the dispersion of femto-satellites. Finally, chapter eight concludes the work and suggests how it could be furthered in the future.

During the course of this work, a paper on the design of the novel distributed receiver was accepted by and presented at the 65th International Astronautical Congress (IAC14) held in Toronto, Canada - Griffiths et al. (2014).

1.1 Background

The effects of space weather phenomena can interfere with satellite communications, with a variety of implications. One such phenomena, that has only been investigated to a limited extent, is the formation of ionospheric plasma depletions, which are described in section 1.2.1. Due to the highly dynamic nature of plasma depletions, frequent measurements are required to understand their formation, propagation and environmental dependencies. Whilst previous measurements have provided useful data, they lack temporal or spatial resolution, with high temporal and spatial resolution, practically, requiring large-scale *in-situ* measurements. This could be achieved through the use of several traditional mini-, or larger, satellites, however, a more cost-effective, and potentially higher resolution, solution is possible through the large-scale use of pico- and femto-satellites. This, along with satellite design and construction, is described in section 1.3, with a potentially suitable plasma sensor discussed in section 1.2.2. As such a large sensor network relies on a large number of nodes, and to make the collected data meaningful, it is important for each node to be able to locate itself, to reduce the complexity in the management of the satellites. The most practical location method, for this application, is through the use of GNSS, which is discussed in section 1.5.

1.2 Space weather

Space weather is the category of research that covers the effects that the Sun and the Earth's upper atmosphere, in particular the magnetosphere, ionosphere and thermosphere, have on space and terrestrial technology (Bothmer and Daglis, 2007). This broad categorisation is due to the effects being caused by the solar activity of the Sun and, to a significantly lesser extent, cosmic radiation, which is, largely, from outside the solar system. With the increasing use of both terrestrial and space technologies, in a vast array of applications, the effect that space weather has cannot only disrupt day-to-day life, with the associated financial costs, but also has the potential to endanger life, making it a particularly significant area of research (Bothmer and Daglis, 2007).

The thermosphere is a part of the upper atmosphere, between the mesosphere and exosphere, that is characterised by its high temperature. It contains the entirety of the ionosphere and parts of the magnetosphere, with the magnetosphere beginning above the ionosphere and extending into the exosphere (Bothmer and Daglis, 2007). The magnetosphere is a plasma and covers a large part of the atmosphere, where it mainly interacts with the solar wind, which is primarily studied using magnetohydrodynamics (MHD). The ionosphere is characterised by its partial ionisation by solar ultra-violet radiation, which results in a neutral gas and a plasma (Kelley, 2009). This dependency on the Sun, results in a large difference between the day and night sides of the ionosphere, as well as

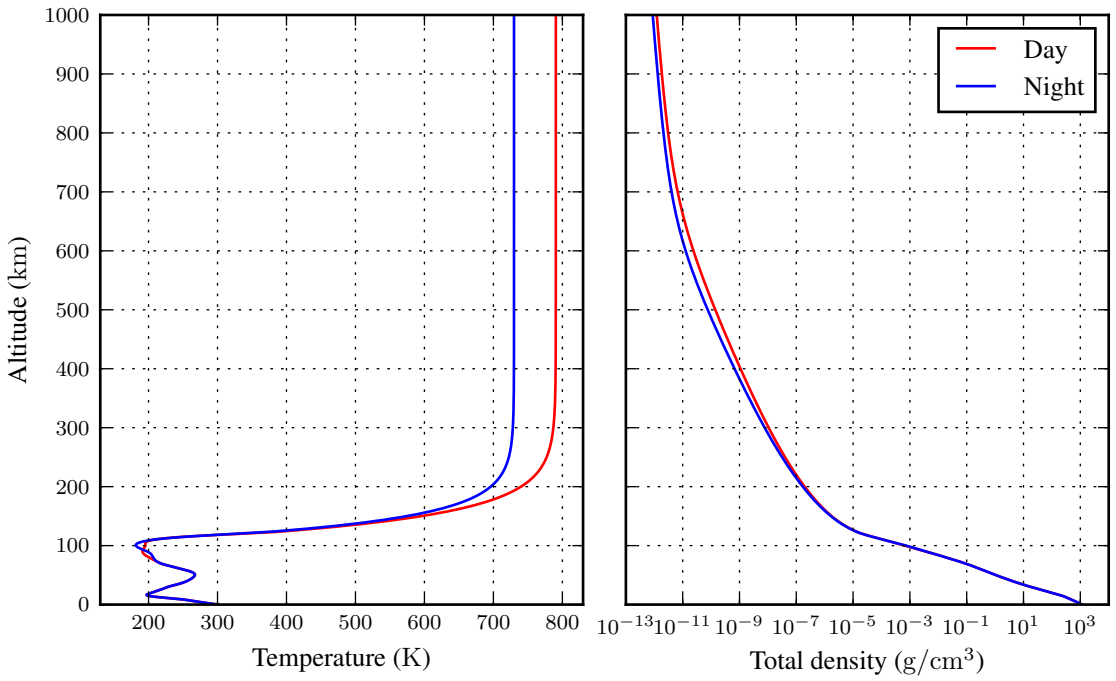


Figure 1.1: Temperature and density of the atmosphere up to $1000\,\mathrm{km}$ altitude, generated by the NRLMSISE-00 empirical model.

variations due to the solar cycle - the changing of the Sun's activity over an approximately 11 year period. Due to these variations, the ionosphere does not have an exact definition in terms of altitude, but it is often taken as being between 70 and $1000\,\mathrm{km}$ (Allnutt, 2011).

One component of space weather is the solar wind, the name given to the charged particles that are emitted by the Sun. These particles are deflected by the Earth's magnetic field into the magnetosphere, where they disrupt its balance. One well known result of this is the aurorae, where the charged particles collide with atmospheric particles, which dissipate the gained energy through the emission of light. Whilst this produces an elegant display, the injection of charged ions into the magnetosphere can pose significant problems to spacecraft. In particular, the Van Allen belt, which consists of two layers of magnetically confined charged particles. The Van Allen belt occurs above the equator between 1.2 and 7 times the Earth's radius (approximate altitudes of $1300\,\mathrm{km}$ and $45\,000\,\mathrm{km}$, respectively), however, the lower layer contains the South Atlantic Anomaly, where the radiation belt has a minimum altitude of approximately 250 km. This poses a problem for spacecraft as the charged particles can collide with electronic equipment causing both temporary and permanent damage to those that are not adequately protected. The first victim of this was the Bell Laboratories' Telstar 1 communication satellite, launched in 1962, that failed within 5 months of launch (Bothmer and Daglis, 2007).

The ionosphere is affected by space weather in two main ways, the first is directly, where ultra-violet radiation ionises the atmospheric gas, and the second is through the magnetosphere, where particle precipitation increases the conductivity of the ionosphere

and convection induces electric fields into the ionospheric plasma. This couples the ionospheric plasma to the solar wind, which has a non-constant flux. This results in variations in the ionosphere, that can induce currents into electrical equipment on the Earth's surface. These were first noticed in telegraph wires between Derby and Birmingham in 1847 (Bothmer and Daglis, 2007). Although this was non-destructive, short-term, large-magnitude variations in the solar wind, such as those caused by solar flares, can cause geomagnetic storms to occur in the magnetosphere. This can result in large induced currents, which can damage many technological systems, such as power systems. Several cases of this occurred in the late 1980s and early 1990s, with one particular example in north-eastern USA in March 1989 costing several million US dollars in damages (Bothmer and Daglis, 2007).

1.2.1 Ionospheric plasma depletions

As the ionosphere lies entirely, or partially, between the Earth and orbiting spacecraft, variations in its composition can affect radio signals and thus communications with satellites. These variations are in the density of the ionospheric plasma, with the density decreasing by up to 3 orders of magnitude inside a depleted region (Huang et al., 2011), and are therefore referred to as ionospheric plasma depletions. As plasma depletions can vary in size from the order of metres to kilometres, depletions smaller than $1000\,\mathrm{km}$ are referred to as plasma bubbles, with depletions larger than $1000\,\mathrm{km}$ being referred to as broad ionospheric depletions (Huang et al., 2011). Despite the variation in size, both plasma bubbles and broad depletions are thought to be caused by the same effect, the Rayleigh-Taylor instability. This is where an interface is formed between two fluids with different densities, with the denser fluid being on top[1], which is an unstable configuration. The study and monitoring of plasma depletions is very important for satellite communications, with low altitude satellites, those below $1000\,\mathrm{km}$, regularly observing localised drop-outs in the plasma density along their orbital track (Krause et al., 2005).

Bubble formation

Plasma bubbles are formed in the post-sunset equatorial region of the ionosphere, in the equatorial ionisation anomaly - a channel that contains a higher concentration of ions, which is characterised by two peaks in the ion density at approximately ± 15 - 20 degrees latitude of the magnetic equator (Magdaleno et al., 2012). With bubble formation being preceded by the pre-reversal enhancement phenomena, where the evening equatorial F-layer of the ionosphere (between 150 and $800\,\mathrm{km}$) is lifted by an eastward electric field (Magdaleno et al., 2012). After sunset, the Rayleigh-Taylor instability is created in the bottom-side of the F-layer, at an approximate altitude of 250 - $300\,\mathrm{km}$ (Takahashi et al.,

[1] With respect to gravity.

2001), by ions recombining at a faster rate at lower altitudes. This results in a steep upward gradient in the density of the plasma, between the depleted bottom-side and the higher density top-side of the F-layer. With plasma bubbles being generated as narrow east-west channels in the equatorial region, that expand to higher latitudes as time progresses, along magnetic flux tubes (Takahashi et al., 2001). Additionally, a fountain effect can be generated in the F-layer, when the eastward electric and northward magnetic fields cause an upward vertical plasma drift velocity. This produces an uplift in the plasma at the magnetic equator, with the plasma being redistributed to higher altitudes along magnetic field lines (Magdaleno et al., 2012). It is worth noting that plasma bubble generation is affected by solar activity, with a greater number of bubbles being generated when solar activity is higher (Magdaleno et al., 2012).

Measurement methods

There are several methods that have been used to observe plasma bubbles, but they can be broadly categorised by how the plasma is measured, either directly, where the plasma properties - such as the density and temperature - are measured, or indirectly, where the effects of the plasma bubbles are measured and the plasma properties are inferred. The C/NOFS (Communication/Navigation Outage Forecasting System) satellite was an example that directly measured the plasma. It was launched in 2008 in to a $13°$, 405 - $845\,\mathrm{km}$ orbit (Huang et al., 2011), and measured the ionospheric plasma, neutral winds and the strength of scintillation producing irregularities (de La Beaujardière et al., 2009). It did this by measuring the electric field, using three orthogonal $20\,\mathrm{m}$ booms, and the ambient ion and electron densities, using a $512\,\mathrm{Hz}$ Langmuir probe, producing a spatial resolution of approximately $13\,\mathrm{m}$. After 7 years of operation, it burned up on re-entry in 2015 (NASA, 2015a).

An alternative measurement method of plasma depletions is to observe atmospheric air-glow, which is where ions in the atmosphere chemically react to, or cosmic rays incident on the atmosphere, produce luminescence. Takahashi et al. (2001) used oxygen emissions at 630 and $557.5\,\mathrm{nm}$ to identify plasma depletions, as the intensity at these wavelengths is significantly reduced due to the reduction in oxygen ions.

As the effect of plasma bubbles on radio signals is dependant on the frequency of the signal[2], one indirect method of measuring depletions is to observe variations between the L1 and L2 GPS signals (at $1575.42\,\mathrm{MHz}$ and $1227.60\,\mathrm{MHz}$, respectively). From this it is possible to infer the slant Total Electron Content (sTEC), that is the electron density along the path of the signal, which is significantly reduced inside a plasma bubble (Magdaleno et al., 2012).

[2]In a similar way that diffraction of light is dependent on frequency.

From C/NOFS' data, the upward ion velocity inside a plasma bubble is typically between 200 and $300\,\mathrm{m\,s^{-1}}$, and both the growth and decay times of plasma bubbles is greater than 3.3 hours, which is consistent with the growth time of approximately 4 hours from simulations (Huang et al., 2011). Combining the C/NOFS data with data from the Challenging Minisatellite Payload (CHAMP) and Defense Meteorological Satellite Program (DMSP) satellites, de La Beaujardière et al. (2009) found that the depletions cover approximately 14 degrees of longitude and 50 degrees of latitude, with depletions occurring more often, and to a greater extent, in the America-Africa and India-Indonesia longitude sectors. de La Beaujardière et al. (2009) also report that C/NOFS repeatedly observed deep plasma depletions close to the crossing of the E-layer terminator, as well as unexpected depletions at dawn.

From air-glow images, Takahashi et al. (2001) observed that plasma bubbles frequently show a branched structure from the main stem, with the bifurcation most likely being caused by vertically modulated electric fields within the bubble, causing the plasma to drift from the centre of the bubble laterally. However, other methods of bifurcation are possible, with more data being required (Takahashi et al., 2001).

Magdaleno et al. (2012) found, from GPS based observations, that during years of high solar activity (2000/01), plasma bubbles were largely found at the equator; during medium solar activity (2004/05), plasma bubbles were largely located at $\pm 10^{\circ}$ magnetic latitude; and during low solar activity (2008), the plasma bubbles were located at $\pm 14^{\circ}$ magnetic latitude. Additionally, the occurrence of plasma bubbles was slightly higher in the northern hemisphere than southern, and the number of occurrences was lowest in May to August. Magdaleno et al. (2012) describe a typical life of a plasma bubble, with generation occurring at approximately 7 pm local time, followed by two hours of rising. The bubbles then peak for approximately an hour before gradually decreasing until sunrise.

Discussion

Both Takahashi et al. (2001), observing plasma bubble bifurcation, and de La Beaujardière et al. (2009), observing unexpected depletions at dawn, illustrate the fact that plasma bubbles are not fully understood. This is largely due to a lack of experimental data on plasma bubbles, rather than a technological or theoretical boundary, with Saylor et al. (2007) describing the data as 'sparse' and 'very undersampled'. Use of indirect methods, such as air-glow and GPS scintillations, provide reasonably high temporal resolution over a given area, but lack spatial resolution. Additionally, the reliance on ionospheric models to calculate certain properties of the plasma, such as the drift velocities (Magdaleno et al., 2012), is not ideal. The use of direct measurements results in significantly better spatial resolution, but due to the limited number of satellites, the overall temporal resolution is low - hence why de La Beaujardière et al. (2009) combined data from several satellites.

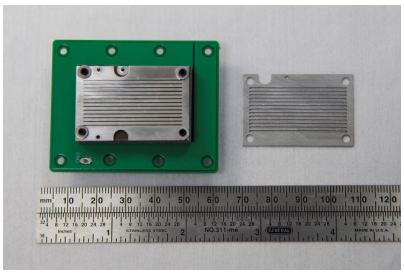
For significant gains to be made in the understanding of plasma bubbles, high spatial and temporal resolution measurements of the ionospheric plasma are required. With high spatial resolution being very difficult to achieve through indirect measurements, due to the large number of observing stations needed, the only practical method of achieving the required resolution is through the use of large scale *in-situ* measurements. Whilst these measurements could be achieved through the use of many traditional-sized scientific satellites, similar in size to C/NOFS for example, the cost of such a programme would be prohibitive. The use of very small satellites, those below $1\,\mathrm{kg}$ (such as pico- and femto-satellites), operating as a sensor network, however, would have a significantly lower cost, whilst still providing the necessary resolution. For this to be feasible, it is necessary that a small enough sensor exists, that is capable of preforming the required measurements with enough resolution. One such sensor is described in the following section (1.2.2).

As plasma bubble generation is confined to the equatorial region, *in-situ* observations of bubble formation are best observed in an equatorial orbit, with observation of bubble evolution requiring an inclined orbit. With the majority of plasma bubbles remaining within 50 degrees of latitude, centred on the equator, this only needs to be a relatively small inclination, with the orbit being near-equatorial. Additionally, an elliptical orbit, with the perigee towards the local sunset, would allow the evolution of the plasma bubbles to be observed to a certain extent. A polar orbit would be the least useful of the low Earth orbits, as only a small part of each orbit would be within the observation region. However, a polar orbit would allow both sides of the equator to be observed by a single satellite, although this is unlikely to be a useful attribute.

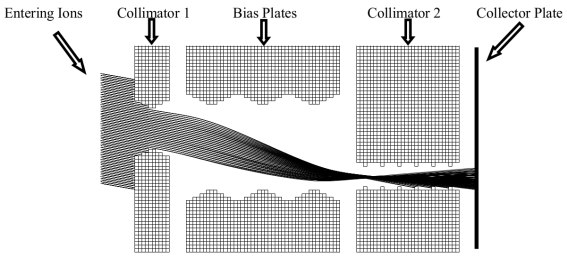
1.2.2 MESA sensor

There are many plasma sensors that can be used in space, however many of them are quite large for pico-satellites[3]. The miniaturised electrostatic analyser (MESA) sensor is a compact electron and ion spectrometer designed by the United States Air Force Academy (USAFA), consisting of an array of analysers, each having a volume of $1.5\,\mathrm{cm}^3$, with an array of 1920 analysers resulting in a sensor size of $5 \times 5 \times 0.5\,\mathrm{cm}^3$ (Balthazor et al., 2009; Enloe et al., 2003; Krause et al., 2005). Each sensor consists of three plates, containing etched holes, that are separated by insulating spacers, that act as an S-bend for incident particles. Each plate has a certain potential applied to it, so that the sensor acts as a band pass for a certain energy of either electrons or ions, with the particles being detected as a current on a biased collecting plate, this is shown schematically in figure 1.2b (Enloe et al., 2003). As the sensor is an array of analysers, any arbitrary size can be used which can reduce the volume and mass to that required, at the cost of decreas-

[3]That is, taking up the majority of a 1U cubesat.



(a) A prototype sensor (Balthazor, 2013)



(b) Simulation of ion behaviour in the sensor (Barnhart, 2008)

Figure 1.2: MESA sensor

ing the signal to noise ratio. For example, a $100 \times 20 \times 2.54\,\mathrm{mm}^3$ sensor has a mass of approximately $45\,\mathrm{g}$ (Balthazor, 2013).

Due to the geometry of the sensor, the analysers' aperture has to be within $\pm 4°$ of the satellite's ram direction (Balthazor et al., 2009), a constraint that is potentially quite difficult to achieve on smaller than pico-satellites. Additionally, to detect ions, the MESA sensor requires a magnetic field of less than $20\,\mu\mathrm{T}$, which means that it cannot be used on satellites with fixed permanent magnets for attitude control, without being significantly redesigned (Balthazor, 2013).

Another small plasma sensor, and the only available alternative to the MESA sensor, is the Conceptual and Tiny Spectrometer (CATS) designed at University College London, which has a fixed size of $2 \times 2 \times 1\,\mathrm{cm}^3$ (Bedington et al., 2011). It has a structure that is reminiscent of the hemispherical analysers used in many spectroscopy applications, with the sensor consisting of two concentric domes at the same potential. Electrons enter horizontally through the top of the dome and are deflected through 90 degrees to detectors at the domes' base. The detectors were originally designed to be micro-channel plates (MCPs), with more recent prototypes using CCDs (Bedington et al., 2011). Both MCPs and CCDs used in this way will degrade with time, with the degradation dependant on the environment, potentially limiting the lifetime of such sensors. However, for short lifetime missions, this degradation is unlikely to significantly affect the data collection, with the radiation susceptibility of CCDs being a much more likely source of error. This, with the fixed sensor size, are limitations that the MESA sensor does not have, without the CATS sensor having any particular advantages over the MESA sensor. Additionally, as the CATS sensor has a more complicated structure, it is likely to cost more to manufacture than the MESA sensor, which could significantly increase the cost of a large sensor network. Therefore, based on the grounds of flexibility and cost, the MESA sensor is the most suitable for the pico-/femto-satellite mission envisaged by this research.

As the MESA and CATS sensors are the only two available small plasma sensors, they represent the state of the art.

1.3 Satellites

Satellites are largely categorised by their wet mass, that is their mass at launch, as it is the main factor that affects the launch costs. A system of SI-style prefixes are frequently used, where micro-satellites are those between 10 and $100\,\mathrm{kg}$, nano-satellites are between 1 and $10\,\mathrm{kg}$, pico-satellites are between $100\,\mathrm{g}$ and $1\,\mathrm{kg}$, and femto-satellites are between 10 and $100\,\mathrm{g}$. Additionally, the category of mini, or small, satellites is sometimes used for wet masses between 100 and $500\,\mathrm{kg}$.

With both the cost and size of technology decreasing, smaller satellites are now able to perform more meaningful tasks. It is because of this, that the cubesat standard was created by the California Polytechnic State University in 2001, to provide a standardised form factor and deployment system, that reduces the launch costs. The standard cubesat form is a 1 litre cube, measuring $10 \times 10 \times 10\,\mathrm{cm}^3$, with a maximum wet mass of $1.33\,\mathrm{kg}$ and is referred to as a 1U (one unit) cubesat. Larger cubesats are also in the standard, with the 2U being $20 \times 10 \times 10\,\mathrm{cm}^3$ and less than $2.7\,\mathrm{kg}$, and the 3U being $30 \times 10 \times 10\,\mathrm{cm}^3$ and less than $4\,\mathrm{kg}$ - effectively combining multiple 1U cubesats along a single axis. The 3U cubesat is the largest available as it is the largest cubesat that can fit into the cubesat deployment system, the P-POD.

As satellites are complex systems, they are often broken down into subsystems, and whilst there are variations in the subsystem naming and contents, they can generally be defined as:

AOCS Attitude and Orbit Control System - responsible for the determination and control of the satellite's attitude and orbital position, through passive or active control methods.

Comms Communications system - typically responsible for the communication with the ground station, including telecommand, although can be with other satellites.

OBDH On-Board Data Handling - the main processing behind the satellite, implementing the telecommand and controlling the other subsystems depending on the satellite state. As the name suggests, it is also responsible for buffering the data between the subsystems and the ground station.

Payload The subsystem that performs the mission specific task, such as data collection in scientific satellites. Typically, the payload will consume a large proportion of the power budget.

EPS Electrical Power System - responsible for providing a reliable power supply to the satellite, including the steering of moveable solar panels.

Thermal Thermal system - responsible for maintaining the satellite's temperature within certain limits, for example, by enabling heaters.

In the following section, an overview of the current state of the art for pico- and nano-satellites is given, with a particular emphasis on the OBDH subsystem.

1.3.1 Pico- and nano-satellites

In a review of post-1997 pico- and nano-satellite missions, Bouwmeester and Guo (2010) highlight that all the satellites were either secondary or tertiary payloads of the launch vehicle. This is a trend that has continued to the present day, partly helped by the cubesat standard, but is mainly due to the large costs of launching a satellite as the launch vehicle's primary payload.

AOCS

Bouwmeester and Guo (2010) found that only 9% of the reviewed satellites had orbital control, of which, the majority were testing new propulsion technologies, with the orbital control system being a payload rather than a subsystem. However, the vast majority, 80%, used attitude control, of which, half used passive control methods, that is hysteresis rods and static magnets, with active control largely being provided by magnetorquers, with momentum based control methods being used infrequently. The majority of the reviewed satellites used the simplest attitude control solution possible, with the main aim of the attitude control being to reduce the rate of the satellite's spin, to increase the reliability of the communication and power generation (Bouwmeester and Guo, 2010). However, some 15% of the reviewed satellites used attitude control to point an instrument, however, the achieved accuracy was significantly less than that of larger satellites. Bouwmeester and Guo (2010) also note that 16% of the reviewed satellites used GPS receivers, which were likely to be used for logging of orbital kinematics rather than accurate timing; however the authors do not discuss how successfully they were used and any errors or problems that occurred.

For orbit control, several types of electric and chemical propulsion systems have been suggested for pico- and nano-satellites, including plasma and arc thrusters. However, only the chemical propellent sulphur hexafluoride (SF_6), used in a cold gas thruster, has been flight tested (Selva and Krejci, 2012). Despite this, current technology is capable of active formation flying required to maintain a constellation, although passive formation flying, using controlled drag, has also been suggested for constellations (Selva and Krejci, 2012).

The design of the attitude control is typically made as simple as possible, unless a mission requirement states otherwise. The lack of orbit control is likely due to the added complexities combined with the short mission lifetimes, as the extra management required is likely to outweigh any potential benefits.

Comms

The majority of reviewed satellites used UHF for ground station communications, with VHF and S-band often being used as secondary downlinks, however, S-band uplinks were infrequently used Bouwmeester and Guo (2010). With a UHF link having an approximate maximum bit rate of $512\,\mathrm{kb/s}$, low resolution VGA pictures (approximately $2\,\mathrm{Mb}$) and hyper-spectral data (approximately $256\,\mathrm{Mb}$) become difficult to transmit and infeasible for a scientific payload (Selva and Krejci, 2012). Additionally, this maximum bit rate is a lot higher than that typically used in cubesats, with $9.6\,\mathrm{kb/s}$ being common for UHF and $256\,\mathrm{kb/s}$ common for S-band (Selva and Krejci, 2012). This, therefore, limits the type of payloads that can be used. However, the use of low data rates, simplistic designs and the preference for UHF, is likely due to the limited power available and the limited tracking abilities of the satellites, as well as to increase the reliability of the subsystem.

EPS

Bouwmeester and Guo (2010) found that 87% of the reviewed satellites used solar cells to generate power, with the majority of these using higher efficiency gallium arsenide (GaAs) cells. A small number of those reviewed, 16%, used deployable solar cells to increase the power generated. However, the average power available of the reviewed satellites was less than $7\,\mathrm{W}$, with the most common power conversion method being the least-efficient direct energy transfer (DET), with only 7% using the significantly more efficient maximum power point tracking (MPPT). Bouwmeester and Guo (2010) highlight that only one of the reviewed satellites did not use batteries, Delfi-C^3; with the most common battery technology being lithium ion and lithium polymer, most likely due to them not suffering memory effects, making them ideal for this type of satellite, where the battery is not likely to be fully cycled. Bouwmeester and Guo (2010) describe three different technologies used to convert power from the solar cells: DET, where the power is taken at a fixed voltage; peak power tracking (PPT), where the voltage is changed to increase the power converted; and MPPT, where the voltage is changed so that the maximum available power can be converted. Of these, PPT is used infrequently as it can have problems with current surges that would be typical for a rotating satellite (where the solar cells are mounted on the outside of the spacecraft). MPPT is only used in 7%, due to it being the most complicated; with DET being used the most often as it is the simplest, despite it being the least efficient and requiring the degradation of the solar cells to be considered in the satellite's power budget.

Thermal

The vast majority of cubesat missions use passive thermal control (Bouwmeester and Guo, 2010), with the exception of heaters in the batteries, as radiators are impractical on

satellites with such limited surface area. For a Sun-synchronous orbit, the satellite's internal temperature typically varies between -15 and $45\,^{\circ}\mathrm{C}$ (Selva and Krejci, 2012). This additionally makes it difficult for some Earth observation technology, such as photodiodes, that work optimally when cooled. However, the temperature range is within typical industrial specifications, allowing off-the-shelf components to be used without the need to consider special thermal management.

OBDH

None of the satellites reviewed by Bouwmeester and Guo (2010) used on-board computers with radiation hardened components, with commercial off-the-shelf (COTS) components being used instead. This was most likely done due to the prohibitive costs of radiation hardened processors and due to COTS microcontrollers having similar processing power to the older space-rated processors, whilst consuming significantly less power. The effects of radiation were mitigated on some of the satellites by the use of redundancy. Bouwmeester and Guo (2010) notes that the most commonly used interface for subsystems was I^2C, which offers no bus-based protection for errors; however, both CAN and USB have been used in pico- and nano-satellites.

As the Bouwmeester and Guo (2010) survey is a few years old, a survey of the cubesats in orbit was conducted. For 36 of these satellites, information about their OBDH system was available and is shown in table 1.1. It was found that the majority used either a 16 or 32 bit architecture for the main processor, with only 7 using 8 bit (19%); with the ARM and TI's MSP430 architectures being the most popular architectures, with 28% each. The operating frequencies varied quite widely, with the lowest being $1\,\mathrm{MHz}$ and the highest being $400\,\mathrm{MHz}$. Although this is not a measure of performance, due to architectural differences, it does illustrate the wide range of processors found in cubesats.

The First-MOVE satellite, launched in November 2013 (Technical University of Munich, 2014), took an interesting approach to reliability, using a single COTS OBC combined with a 'hard commanding unit' (Czech et al., 2010). The hard commanding unit was designed so that it had the ability to reset the OBC via an uplink command, whilst being a separate unit to the transceiver, instead monitoring its output for a reset pattern. This is an interesting and peculiar design decision, as the use of pattern matching by the hard commanding unit appears to be an unnecessary complication. For system reliability, the OBC's software was stored in both flash memory and magnetic RAM (FRAM), to avoid having to use expensive radiation hardened memory, as FRAM is not susceptible to SEL and SEB events (Czech et al., 2010). However, the use of a voting system, between the FRAM and the two flash memories, increases the chance of memory corruption as errors in both of the flash memories can corrupt the less susceptible FRAM. A more radiation resilient solution is to use a single FRAM with checksums, which is a little counter-intuitive as radiation redundancy typically involves duplicating hardware. This relies on the FRAM

Table 1.1: Survey of processor architectures used in cubesats.

Name	Architecture	Number of bits	Frequency (MHz)	Reference
AAU cubesat	C161	16	10	University of Aalborg (2013a)
AAUSAT II	ARM	32	40	University of Aalborg (2013b)
AAUSAT 3	ARM	32	40	University of Aalborg (2013c)
AubieSat-1	ATmega	8	16	Auburn University (2012); Wersigner (2013)
BEESAT-1		32	60	Technische Universitat Berlin (2013a)
BEESAT-2		32	60	Technische Universitat Berlin (2013b)
CanX-1	ARM	32	40	University of Toronto (2013a)
CanX-2	ARM	32	15	University of Toronto (2013b)
CAPE1	PIC	8	40	University of Louisiana (2013)
Compass-1	8051	8	100	Aachen University of Applied Sciences (2013)
CSSWE	MSP430	16	16	University of Colorado (2013a)
Cute 1	H8	8	16	Tokyo Institute of Technology (2013a)
Cute-1.7 + APD II	ARM	32	400	Tokyo Institute of Technology (2013b)
Delfi-C3	MSP430	16	8	Delft University of Technology (2013)
DTUsat-1	ARM	32	16	Technical University of Denmark (2013)
E-ST@R	MSP430	16		Politecnico di Torino (2013)
ESTCube	ARM	32	72	University of Tartu (2013)
F-1	PIC	8		FPT University (2013)
FITSAT-1 (NIWAKA)	PIC	8	20	Tanaka (2013)
GOLIAT	MSP430	16		Romanian Space Agency (2013)
HERMES	MSP430	16	40	University of Colorado (2013b)
ITUpSAT1	MSP430	16	16	Istanbul Technical University (2013)
M-Cubed	ARM	32	400	University of Michigan (2013a)
NCUBE	PIC	8	8	University of Oslo (2013)
OSSI-1	MSP430	16	1	Hojun (2013)
QuakeSat	x86	32	100	Stanford University (2013)
RAX-1	MSP430	16		University of Michigan (2013b)
RAX-2	MSP430	16		University of Michigan (2013b)
STRaND 1	ARM	32	40	University of Surrey (2013)
Swiss Cube	ARM	32	33	Ecole Polytechnique Federale De Lausanne (2013)
Tisat-1	PIC	16		University of Applied Sciences of Southern Switzerland (2013)
UNICUBESAT GG	MSP430	16		La Sapienza University of Rome (2013)
UWE-1	H8	16		University of Würzburg (2013)

only being susceptible to SEUs, with an event either upsetting all transfers or only part of the transfer, both of which can be detected and corrected for using checksums. To increase reliability, 3 separate power buses were used inside First-MOVE's OBDH, all of which have latch-up detection (Czech et al., 2010). However, no details are given on how these buses were used and whether this increased the reliability and robustness of the subsystem, or whether electronically controlled power to the different components could give an equal reliability with less complexity. An end of mission statement was published on the 21st December 2013 by the First-MOVE team stating that whilst they have been able to remotely reset the satellite, presumably using their hard commanding unit, the on-board computer was not completing its boot sequence, with the most likely reason being corruption of the program memory (Amateur Radio station PE0SAT, 2016; Technical University of Munich, 2014).

de Jong et al. (2008) describes how the team that designed Delfi-C^3 changed the design for its successor, Delfi-N3XT, which allows a rare insight into the problems faced during development. Delfi-C^3 had a battery-less design based on an MSP430 that was under-clocked to $1\,\mathrm{MHz}$ (from $8\,\mathrm{MHz}$, presumably to reduce power consumption. Each subsystem had its own processing provided by a PIC microcontroller, running at $31\,\mathrm{kHz}$, which acted as decentralised redundancy, as each subsystem monitored the main on-board computer's activity. Due to the different frequencies used, integration issues limited the performance to effectively half that of the slowest processor. This limitation is rather unusual, as often complicated systems consist of many processors running at a range of frequencies, therefore, it is likely that this limitation was due to the particular implementation of decentralised redundancy as well as the communication bus. Many improvements were made for the successor, such as the use of a higher data rate S-band downlink and an active 3-axis attitude control system. However, one of the most interesting changes was the removal of decentralised redundancy in preference of having a single redundant node based in the radio system. Additionally, the successor uses a multi-master I^2C bus, with the masters being the on-board computer, an MSP430, and the radio subsystem, a PIC. To prevent the bus from becoming locked, such as the babbling idiot condition, I^2C buffers (P82B96) were placed on each node, which consumes approximately an extra $10\,\mathrm{mW}$ per node, but is a rather simple and elegant solution. The successor additionally included a real time clock (RTC), as the initial design, partly due to it being battery-less, produced different data frames with the same identifier, making it near impossible to assemble the received data in the transmitted order (de Jong et al., 2008).

From de Jong et al. (2008) there are a few important points to take note of. The integration issues with the original satellite are most likely due to a lack of testing in the design phase, a poor decentralised redundancy implementation and possibly making the satellite too heavily dependent on the communications bus, to such an extent that it became a bottle neck in the design. It is interesting that de Jong et al. decided that the

original design needed such a high level of redundancy. It is highly probable that this was excessive, considering the mission parameters, and a simpler, less redundant design would have not caused the same integration problems. The use of I^2C buffers as bus guardians, in the design of the successor, is a very simple and elegant solution to maintain a common bus when devices fail. From this, it is important to realise that the system needs to be thoroughly tested from the beginning so that potential integration issues are found and resolved early in the design. It is also important to design for the flight environment, so that the design has an adequate, but not excessive, amount of redundancy. It is quite surprising that the issue of unique identifiers for communications was not thought of before the original design was completed, especially considering its battery-less design.

Discussion

Throughout the review by Bouwmeester and Guo (2010), there is a common theme that pico- and nano-satellite missions often use the simplest solution possible, rather than complicating the design. This is most likely due to cubesats, which make up a large proportion of the missions reviewed, being seen as either educational platforms, where the main purpose is to give students experience, or as testing platforms for new hardware, where the device to be tested is the main interest, with the support hardware sacrificing functionality to increase reliability. Bouwmeester and Guo (2010) cite, and Selva and Krejci (2012) agree, that the main bottleneck in pico- and nano-satellite designs is the attitude control, with the lack of accuracy and dynamic control being the main limiting factor. This is both a limitation for Earth observation, as well as for communication with a ground station, with highly directional communication being infeasible, limiting the communication to lower frequencies and data rates.

Although it is fairly easy to pinpoint the state of the art for each subsystem, cubesats are not designed with each subsystem being the best possible, they are designed to be the best fit for the mission requirements, which is a common theme amongst satellites.

1.3.2 Femto-satellites

There have currently been very few attempts at using femto-satellites. This is most likely due to their small size limiting their applications, in terms of both payloads and communication. One notable attempt was the KickSat satellite, that was launched in April 2014 (Manchester, 2014a). This was a project to launch 128 sprite satellites from a 3U cubesat, with each sprite measuring $35 \times 35 \times 3\,\mathrm{mm}^3$ and weighing $5\,\mathrm{g}$. The project was funded through the crowd-funding website Kickstarter, with the aim being to lower the cost of satellites so that anyone could launch one (Manchester, 2011). Removing the marketing spiel, the aim of the project can be considered to be public engagement and this is reflected in the simplistic design of the sprites. Each sprite had a Texas Instruments CC430

SoC, which combines an $8\,\mathrm{MHz}$ MSP430 processor with a $10\,\mathrm{mW}$ UHF transceiver, a gyroscope and a magnetometer. The sprites were designed to have an unregulated power supply (i.e. directly from the solar cells) and to be battery-less, with Manchester et al. (2013) stating that batteries would not be able to survive the cold temperatures during eclipse. In addition to this, the sprites were designed to have no attitude control, either active or passive, and instead were designed to rely on the spin of the launching cubesat for stabilisation and Sun orientation.

Overall, the design of the sprites is quite reasonable, considering the size limitations. The SoC's processor is more than capable of performing the required tasks, however, the radio has a lower than ideal transmit power. In particular, Manchester et al. (2013) estimates that the signal to noise ratio at ground level is around $2\,\mathrm{dB}$ below the noise floor. To allow the signal to be received, CDMA is used - however, the number of PRN codes used is less than the total number of sprites. So that the sprites' communications do not interfere with each other, they are designed to only transmit data 5% of the time - which is approximately 137 seconds of their 91 minute orbit. At the radio's $50\,\mathrm{b/s}$ data rate[4], each sprite can transmit, at most, 6850 bits per orbit, which is approximately $856\,\mathrm{B}$. This is a severe limitation of the sprite design, limiting their usefulness as a satellite platform. The transmit power limitation could be, at least partially, mitigated by the inclusion of a battery - allowing short transmissions at a higher power. The conclusion that batteries should be omitted is a curious one. The use of battery heaters in space is so widely adopted that using a battery without a heater, or some kind of thermal control system, would be beyond negligent. Whilst it could be argued that using a heater on such a small satellite would consume too much power, the size of the battery, to fit within the physical constraints, would be small enough that any required heating would also be small and fit well within the approximate power budget stated in Manchester et al. (2013). The lack of attitude control is to be expected, considering the size of the sprites, however, the sensor payload could be made significantly more useful with the addition of an accelerometer. The gyroscope and magnetometer are able to provide information on the separation of the sprites from the launching cubesat, but the sensor payload could be significantly improved by the addition of an accelerometer. Manchester et al. (2013) estimate the sprite's life, i.e. the time before re-entry, as being between 5 and 27 days, which is a large range. The addition of an accelerometer would allow the effect of the drag on the sprites to be measured, providing additional information that would allow better life predictions for future flights.

Unfortunately, the design of the sprites could not be tested during the 2014 flight, due to a few failings with the design of the launching cubesat. The cubesat's uplink required a minimum of $8\,\mathrm{V}$ to operate, with only around $6.5\,\mathrm{V}$ being seen in orbit (Manchester, 2014c). This meant that a command to launch the sprites could not be sent to the cubesat.

[4]The data rate is actually $100\,\mathrm{b/s}$, however, half the bits are reserved for parity.

However, the cubesat was designed with a backup for the launch, in case an uplink could not be established. This relied on a 16 day timer being run on the cubesat's main processor. Unfortunately, the processor was reset - with a likely cause being radiation - and with it, the launch timer (Manchester, 2014c). This meant that the time of the sprite launch was delayed until after the cubesat had re-entered, and so the sprites were not launched (Manchester, 2014b).

Both of these can be considered to be failings in the design of the cubesat. No information on the design of the cubesat is available, but the voltage requirement gives the impression that the cubesat's main power source was not regulated. With the uplink being a vital part of a satellite, if not one of the most important parts, it is only logical to ensure that it can always be powered, with any differences in voltage ranges being satisfied with regulators. It is highly unlikely that the KickSat launcher suffered from inadequate power levels, as telemetry was being broadcast, so the issues with the uplink can only be a design oversight. The resetting of the launch timer was due to the cubesat's system clock being reset by a processor reset (caused by a watchdog) (Manchester, 2014c). This means that the system clock was only stored in volatile memory, which seems to be careless at best. A common practice, for clocks that lack external synchronisation, is for them to periodically store their value, so that if they are reset, they only lose a relatively small amount of time. In a radiation environment, to avoid corruption of this value, a common method would be for it to be stored in several different places - e.g. in different pages of flash memory. These failings in the design, highlight the need for a well designed mission, with a thorough testing plan, to ensure that no oversights are made, even when utilising low cost hardware and short-life, disposable satellites.

There are very few femto-satellites described in the literature, however, one is the PCBsat introduced by Barnhart et al. (2009), along with suggestions of what distributed satellite missions could be used for (terrestrial and space weather monitoring, distress beacon monitoring and atmospheric composition measuring). The design is based on a 9 by $9.5\,\mathrm{cm}$, 4-layer PCB, using an 8 bit microcontroller (an Atmel Mega128L) as the on-board computer. Power is provided by hobby grade ($15\,\%$ efficient) solar cells and is stored in a $645\,\mathrm{mAh}$ Li-ion camera battery, giving approximately 6 hours of battery life from a full charge. The PCBsat does not have the capability of communication with a ground station, instead a $60\,\mathrm{mW}$ ZigBee RF module is used to transmit data to other nodes and a larger node with ground station communication capabilities (i.e. a cubesat). The attitude is determined by magnetometers and Sun sensors, with a single axis magnetorquer being used to allow some degree of attitude control. Additionally, a commercial GPS receiver is included to determine the satellite's orbit position.

The original design was meant to be a proof of concept rather than a flight model, however, the use of COTS components was designed to make the PCBsat cheap to manufacture, rather than being convenient for the proof of concept model. Unfortunately, there

are several aspects of the PCBsat's design that prevent it from being used in space - one of these is the GPS receiver. Commercial GPS receivers have a typical operational ceiling, due to export restrictions, of $18\,\mathrm{km}$. This means that GPS receiver would be incapable of functioning in space, however, this initial concept is something that could be developed further into a fully working satellite.

Barnhart et al. (2007) furthers the idea of creating a space sensor network using multiple PCBsats and suggests three possible measurements such a network could make. These are measurements of the day-side mid-latitude trough, detecting Joule-heating sources and measuring ionospheric plasma bubbles. Barnhart et al. go into more details of such a network, stating that the only commercial constellation with cross-links is the IRIDIUM constellation, that formation flying is complex and so far has only been used in experimental tests, and that a node separation of $10\,\mathrm{cm}$ would require sampling of any sensors to occur every $10\,\mu\mathrm{s}$.

1.4 Satellite constellations and swarms

Using multiple satellites for the same mission, or task, is the obvious, and often the only, solution for communication and global navigation systems, largely due to the necessity of global coverage. However, this is not the only use of satellite constellations. For many scientific missions, the use of multiple satellites can be a requirement of the scientific payload or can increase the data's scientific value, either through higher spatial and temporal resolution, or through multiple measurements using different apparatus. There are certain types of scientific missions that are more suitable for constellations, with these typically being Earth observation based. Here we make a distinction between a satellite constellation and a satellite swarm, with a satellite constellation being a number of satellites where the distribution is largely fixed (that is, where the satellite separation is maintained) and a satellite swarm being where the distribution is largely free, with satellite dispersion being allowed to occur.

Navigational satellites will only be discussed broadly in this section, as they are discussed in depth in section 1.5. Despite there being many different global navigation satellite systems (GNSS), they all work on the same principle of placing highly accurate clocks at known reference points. With the receiver calculating its position from the difference between several clocks[5]. There is, therefore, no need for GNSS satellites to have inter-satellite links.

[5] At least four.

1.4.1 Satellite constellations

The two satellite constellation GRACE, launched in 2002, is designed to measure the Earth's gravitational field by studying the perturbations of the satellite separation (Tapley et al., 2004). To do this, identical satellites were launched into a near-circular orbit at approximately $500\,\mathrm{km}$ and carry precise positional measurement systems - a GPS receiver and an inter-satellite microwave link - as well as a high precision accelerometer. Variations in the gravitational field are detected by one of the satellites accelerating, with respect to the other, approximately $220\,\mathrm{km}$ away. As these accelerations are very small, multiple systems are used to perform the measurement. The GPS receiver uses both the L1 and L2 frequencies, has a precision of $7\,\mathrm{mm}$ and is used mainly for time tagging the data and coarse position information. A K-band microwave link, with a precision better than $10\,\mu\mathrm{m}$, is used as the main apparatus to determine the satellite separation. The high precision accelerometer, with a precision of $10^{-11}\,\mathrm{g}$, is used to remove the effects of any non-gravitational forces (Tapley et al., 2004). The inter-satellite link is not a conventional communications link, with the communication being limited to the phase of the signal.

ESA are developing a technology demonstration for formation flying called PROBA-3, which will consist of a two satellite constellation with a combined mass of between 500 and $600\,\mathrm{kg}$ (Sephton et al., 2008), that is scheduled for launch in 2017 (ESA, 2013). The proposed method of formation flight is to use an occulted solar chronograph - the lead satellite, the occulter, will have a $1.5\,\mathrm{m}$ diameter occulting disk, that casts a shadow onto the following satellite, the chronograph. The chronograph, following the occulter at between 150 and $250\,\mathrm{m}$, will adjust its orbit so that the occulting disk's shadow is in the same place on its optical sensor (Sephton et al., 2008), effectively producing a limited one-way communications link. This method is particularly useful for satellites that directly follow each other, with a small separation, and so is well suited to imaging based Earth observation tasks. However, due to the required size of the occulting disk, this method is not particularly well suited for pico or femto-satellites - a $10\,\mathrm{cm}$ disc, producing the same solid angle, would have a separation of between 10 and $17\,\mathrm{m}$, with an optical system fifteen times larger than that of the PROBA-3 satellite required to work at the same separation distances - see appendix A for details. This practically limits the separation distance for occulting pico-satellites to below $1\,\mathrm{km}$.

ESA launched a constellation project, consisting of three satellites, to measure the Earth's magnetic field in November 2013, called Swarm (Merayo et al., 2008; ESA, 2016). Each satellite has a mass of $468\,\mathrm{kg}$ and was launched into a near polar orbit, with one satellite at an altitude of $530\,\mathrm{km}$ and two at $460\,\mathrm{km}$ (Haagmans et al., 2012). The higher altitude satellite is designed to have a slightly different inclination, by $0.6°$, so that it will cross the lower satellites' orbit at a right angle during the third year after launch (Haagmans et al., 2012). With the aim of the project being precise measurements of the

Earth's magnetic field, both in spatial and temporal terms. Each satellite uses three star trackers, each having an accuracy of better than 1 second of arc, as well as a GPS receiver and a retro-reflector (Merayo et al., 2008; Haagmans et al., 2012). Unlike the PROBA-3 mission, the satellites of the swarm constellation do not have any inter-dependencies, as they are designed to work separately.

The GANDER constellation was a mission concept by the Surrey Space Centre[6], consisting of 12 micro-satellites in a Sun-synchronous orbit to perform sea monitoring (Zheng, 1999). The satellites were designed to measure the relative sea level using radar altimetry, which would then be broadcast on UHF for marine use - to aid ships in avoiding difficult weather that could cause them damage. To perform this in a beneficial way, multiple satellites would be required. However, the satellite constellation was never put into orbit, possibly due to its high estimated cost of US$ 50 million. Each satellite of the constellation was designed to be less than $100\,\mathrm{kg}$ and $60 \times 60 \times 80\,\mathrm{cm}^3$, which partly explains the high cost, and designed to work independently, with no kind of inter-satellite communication.

1.4.2 Inter-satellite communication

There has only been one commercial satellite constellation to date that utilises inter-satellite communication - the Iridium communications constellation (Barnhart et al., 2007). The Iridium constellation consists of 66 satellites in 6 orbital planes at an altitude of approximately $780\,\mathrm{km}$, with each satellite weighing $680\,\mathrm{kg}$ (Fossa et al., 1998). An integral part of the Iridium constellation is the satellite cross-links, which is used to pass the handling of phone calls from one satellite to another, to avoid the phone calls being dropped, as each satellite is only visible for approximately 9 minutes, due to the satellites being in LEO. These inter-satellite links use steerable antennas and operate in the K band, providing a data rate of $25\,\mathrm{Mbps}$ (Fossa et al., 1998).

The lack of inter-satellite links among constellations is likely due to a lack of requirement. For many commercial satellites, such as those that perform satellite imaging, inter-satellite links are not beneficial. For those where inter-satellite links could potentially be beneficial, the requirement can often be designed out. In the case of the Iridium constellation, the inter-satellite links are required due to the satellites being in LEO and the user terminals only being able to connect to one satellite at a time. If, however, the user terminals were designed so that they could communicate, at least partially, with a second satellite, then the inter-satellite link would no longer be required. When the Iridium constellation was designed, in the early 1990s, this addition to the user terminal would have considerably increased its size and cost. However, with the constant evolution of technology, this high level of integration is becoming more practical and also has the ad-

[6]Part of the University of Surrey.

ditionally benefit of not only reducing the complexity of the satellites, but also allowing the user terminal to select the best satellite to change to, using a metric that can be easily updated. Therefore, it is highly likely that if the Iridium satellite was being designed in the present day, it would not use inter-satellite links.

Of the scientific satellite constellations, those that use inter-satellite links do so as part of their measurement apparatus, rather than using them as an alternative to, or to complement, a ground station link. However, a need for inter-satellite links is introduced if a distributed system is used - whether this is sharing tasks between satellite nodes, or providing a data relay system. In the case of a data relay system, there would be very little, if any, computation of the received data occurring on the receiving node; however, with distributed computing the computational processing of the received data is typically large, with a tendency for the transferred data to be as small as possible, as to increase the computational efficiency. Horst and Noble (2011) present a market-based task allocation, designed for large numbers of heterogeneous satellites in a dynamic network. Whilst their market-based allocation is particularly relevant to heterogeneous nodes, where each message can only be received by a small subset of the total number of nodes, their model of a dynamic network is applicable to many distributed satellite systems. Their model is Keplerian based, uses orbital parameters derived from a single reference orbit, randomly perturbed for each satellite, and does not consider non-Keplerian contributions, such as non-Newtonian forces (e.g. drag), solar and terrestrial pressure, and variations in the Earth's gravitational potential. These assumptions are stated as valid, as only one orbit, which is less than 100 minutes, is considered. This model is used by Horst and Noble (2011) to analyse the efficiency of their task allocation algorithm, however similar models would be necessary to understand the connectivity of any distributed satellite system.

1.4.3 Small satellite constellations

Despite the lack of small satellite constellations, there are certain types of Earth observation, such as gravity measurements, that could significantly benefit from multiple satellites in regular orbits to improve and supplement existing data (Selva and Krejci, 2012). Thomsen et al. (2008) investigate the feasibility of using a constellation of miniature satellites to measure the Earth's magnetic field. The authors set the requirements of better than $1\,\mathrm{ms}$ clock accuracy, with inter-satellite time synchronisation better than $1\,\mathrm{ms}$, better than 1° attitude determination and better than $500\,\mathrm{m}$ orbit determination. Both the timing and orbital determination requirements could be met by using GPS, whilst the attitude determination requirement could be met with commercial Sun sensors and, during eclipse, magnetometers. Thomsen et al. (2008) suggest using magnetorquers for attitude control and controlled drag for constellation control, through the use of controlled flaps on the side of the satellites. For flaps that double the drag area of the satellite, their simulations

show that during a solar maximum at an altitude of $550\,\mathrm{km}$, the flaps would affect the orbit by approximately $20\,\mathrm{m}$ per orbit, however during solar minimum, at an altitude of $450\,\mathrm{km}$, the flaps would affect the orbit by approximately $2.7\,\mathrm{m}$ per orbit (Thomsen et al., 2008). This highlights the effect the solar cycle has on a satellite's drag and suggests that controlling a satellite formation using drag is significantly more difficult during solar minima.

1.5 GNSS

For Earth and atmospheric measurements, among many others, to have any scientific value the exact location and time of the measurement needs to be known. For many missions, this is achieved using global navigation satellite systems, such as GPS.

Global Navigation Satellite System (GNSS) is the generic name of any satellite based navigation system that is designed to provide global coverage. The most known of these is the American GPS, however, there are also the Russian GLONASS, the Chinese COMPASS and the EU's Galileo, which is due for completion in 2019 (ESA, 2011). All GNSS use the same principle for navigation - providing high precision clocks that transmit their current time and position, so that a receiver can determine its location by inferring the distance between itself and the clocks by measuring the delay in the received transmissions. Each of these systems use different methods for signal encoding and transmission, with them operating on different frequencies and using different modulation methods. However, there are many similarities between them, with the most similar being GPS and Galileo. The GLONASS and COMPASS constellations are used much less than GPS, as they do not provide the same global coverage and have a history of not being as reliable, with the Russian GLONASS spending many years with the constellation only providing partial coverage, with only a quarter of the designed number of satellites being operational, due to a lack of funding (Hofmann-Wellenhof et al., 2008).

GNSS is divided into three different segments - space, control and user. The space segment consists of the orbiting satellites, both those which are active and inactive; the control segment consists of the management of the space segment, changing which satellites are active based on constellation specific criteria and updating the satellite's data; and the user segment consists of the GNSS receivers, receiving the broadcasts from the space segment (Hofmann-Wellenhof et al., 2008). The interaction between the user and space segments are the most important for locating the user, as the control segment, whilst being vitally important to the accuracy of the GNSS and the continued functionality of the constellation, is hidden from the user.

To describe a non-constrained position in a three dimensional space, three coordinates are required and are usually measured from a fixed point in the reference frame. If three unique points in the space are known, then any position in the space can be described

by the distances between these points and the position, through the use of trilateration. Using this, a three dimensional location system could consist of three retro-reflectors, being optical or radio in nature, at three different locations. Then any position in the space could be found by measuring the time of flight of a signal to complete a round trip from a certain position to the three retro-reflectors. Whilst this method would be effective, it has certain drawbacks that make it impractical for a GNSS. The requirement to transmit the signal would require the transmitted signal to be very high powered, for large distances, due to the signal's dispersion. An additional drawback is the pointing requirement, which would make it increasing difficult, with distance, to accurately aim the transmitted signal onto the retro-reflectors. If we replace the three retro-reflectors with transmitters, which encode the time the signal was sent onto the signal, then the time of flight of the signal from the transmitter can be calculated at the receiving position. This, however, has the drawback that each of the three transmitters and the receiving position require an accurate clock - of atomic precision - with all of the clocks being synchronised. This drawback can be simplified for the receiving position, by using four transmitting unique points rather than three, then the receiver can use a significantly less accurate clock, as the fourth transmitting point provides enough information for the receiver to correct for its clock inaccuracies. This is the principle used in GNSS, however, in practice it is not as simple.

As each of the transmitters are on satellites orbiting the Earth, their exact position at a given time is not known. This means that this information has to also be encoded onto the transmitted signal. However, as the satellites are moving, their location is described using orbital elements that are statistically based on the satellite's previous motion, adding additional errors. For at least four of the satellites to be visible at any given point on the Earth's surface, many more satellites are required - GPS uses 24 operational satellites in six orbital planes, so that between 4 and 8 satellites can be typically seen (Hofmann-Wellenhof et al., 2008; El-Rabbany, 2002); with Galileo being designed to use 27 operational satellites in three orbital planes, so that at least 6 satellites are visible (Hofmann-Wellenhof et al., 2008); and GLONASS using 24 operational satellites also in three orbital planes, with at least 5 satellites being visible from over $99\,\%$ of the Earth's surface (Hofmann-Wellenhof et al., 2008). For accurate positioning, there are additional complications that affect the transmitted signals that have to be considered, such as multipath propagation, ionospheric and tropospheric refraction, and relativistic effects (Xu, 2003).

1.5.1 GNSS signals

To minimise the effect of interference, the transmissions from GNSS satellites use spread spectrum techniques, in particular direct-sequence spread spectrum (DSSS), with the majority using code division multiple access (CDMA). This is where a pseudo-random noise (PRN) code, unique to each satellite, is used to spread the transmitted data over a wide

shared bandwidth; with each PRN code chosen to be orthogonal to each other (i.e. so that there is little correlation between the codes used in the system). The signal is then decoded by the receiver knowing the PRN codes and applying them successively to the received signal. GLONASS is the exception to this, using frequency-division multiple access (FDMA), where the bandwidth is divided into 15 channels that are shared between the 24 satellites; however, the use of CDMA for GLONASS is being researched, to improve compatibility with other GNSS (Inside GNSS, 2008).

GPS satellites were originally designed to broadcast two BPSK modulated signals on two carrier frequencies, one at $1575.42\,\mathrm{MHz}$ and one at $1227.60\,\mathrm{MHz}$, referred to as L1 and L2 respectively. The L1 signal consists of a coarse acquisition code, referred to as C/A, and a precision code that is intended for military use, referred to as P(Y) to denote that it is encrypted with a secret Y code, with the L2 signal consisting of only the P(Y) code. The C/A code uses a 1023 chip long PRN code broadcast at a chip rate of $1.023\,\mathrm{Mcps}$. This results in a code that repeates every millisecond and a chip range - the distance the signal travels during a chip - of $293\,\mathrm{m}$. The P(Y) code uses a 6.1871×10^{12} chip long PRN code broadcast at a chip rate of $10.23\,\mathrm{Mcps}$, resulting in a chip range of $29.3\,\mathrm{m}$ and a code that repeats every week (Kaplan and Hegarty, 2006). The length of the P(Y) code means that it cannot be directly acquired, without an accurate clock estimate or knowledge of the satellite's position, and so requires the C/A code to be acquired first (Hofmann-Wellenhof et al., 2008). Both the C/A and P(Y) codes contain the same navigational message that is transmitted at $50\,\mathrm{b/s}$. The navigational message is split into 5 subframes of 300 bits - the first subframe contains the GPS week number, the satellite's accuracy and clock corrections, with the second and third frames containing the satellite's ephemeris, which is used to find the satellite's position. The fourth and fifth subframes contain 25 pages each and describe the GPS constellation, with almanac and health data for all 32 possible satellites as well as information to correct for ionospheric effects. The almanac contains coarse orbital elements for all of the GPS satellites, to assist in satellite acquisition, with the ephemeris containing more accurate orbital elements to enable precise location, which are typically valid for 4 hours (Kaplan and Hegarty, 2006).

As of 2005, supplemental signals and codes have been added to the broadcasts of GPS satellites in what is referred to as GPS modernisation (Kaplan and Hegarty, 2006; Hofmann-Wellenhof et al., 2008). The first addition, added in 2005, was the military, or M, code. It is designed to be the successor to the P(Y) code, providing better accuracy, an improved resistance to signal jamming as well as being directly acquirable, that is, not requiring the use of any additional signals. Like the P(Y) code, it is encrypted, but unlike the legacy GPS signals, it does not use BPSK modulation, instead it uses binary offset carrier (BOC) modulation (Hofmann-Wellenhof et al., 2008).

The second addition, starting in 2010 (National Coordination Office, 2013), was the introduction of a civil signal to the L2 carrier, referred to as L2C, and the introduction

of a civil safety-of-life signal to the new L5 carrier ($1176.45\,\mathrm{MHz}$), referred to as L5C, with both the L2C and L5C signals using BPSK modulation (Hofmann-Wellenhof et al., 2008). The L2C signal consists of two balanced codes, both of which have a chip rate of $511.5\,\mathrm{kcps}$, but are time multiplexed on to the L2 carrier to create a $1.023\,\mathrm{Mcps}$ code (Hofmann-Wellenhof et al., 2008). The first is a moderate code, referred to as L2CM, whose length is $10\,230$ chips, and the second is a long code, referred to as L2CL, whose length is $767\,250$ chips (Kaplan and Hegarty, 2006). The L2CM code is modulated by a 50 baud data stream, containing a $25\,\mathrm{b/s}$ navigational message and a forward error correction (FEC) code, with the L2CL code being left as a pilot channel and so containing no data. The use of a lower data rate navigational message, than the legacy signals, is designed to assist demodulation in difficult environments, where the signal is attenuated to a greater extent, whilst the use of a data-less L2CL code is designed to make the tracking of the signal more robust (Kaplan and Hegarty, 2006). The L5 signal contains in-phase and quadra-phase signal components that use different PRN codes, with lengths of 8190 and $10\,230$ chips, and a chip rate of $10.23\,\mathrm{Mcps}$. The in-phase component is modulated with a navigational message, where as the quadra-phase component is used as a pilot channel, drawing similarities to the L2C signal (Hofmann-Wellenhof et al., 2008). The navigational message is added as a $50\,\mathrm{b/s}$ stream and also, like the L2C signal, is combined with an FEC resulting in a 100 baud data stream (Kaplan and Hegarty, 2006). Both phases are modulated with Neuman-Hoffman synchronisation codes that have a chip rate of $1\,\mathrm{kcps}$ and are 10 chips, for the in-phase signal, and 20 chips, for the quadra-phase signal, long (Hofmann-Wellenhof et al., 2008). These synchronisation codes are designed to reduce narrow-band interference and cross-correlation, whilst allowing for a more robust symbol/bit synchronisation (Hofmann-Wellenhof et al., 2008).

The third addition to GPS is the introduction of a fourth civilian signal to the L1 carrier, referred to as L1C (Hofmann-Wellenhof et al., 2008), with the first satellite intended to be launched in 2017 (Gunter's Space Page, 2016). The L1C signal is designed to be a common signal between GPS and Galileo and will use multiplexed binary offset carrier (MBOC) modulation, with the signal consisting of both a data and a pilot channel, like the L2C and L5 signals (Hofmann-Wellenhof et al., 2008).

As these additions will be completed in stages, complete support for the modernised signals across all 24 operational GPS satellites is expected to be completed in 2018 for L2C, 2021 for L5 and 2026 for L1C (National Coordination Office, 2013).

The Galileo signals are categorised into 4 services - the open service (OS), intended for general use; the commercial service (CS), intended for users that require a guaranteed quality of service, for example in asset management; the safety-of-life service (SoL), intended for aeronautical and maritime users who require a guaranteed quality of service as well as integrity information, to indicate the user of faults with the system (in particular the satellites); and the public regulated service (PRS), intended for the use of European

nations for civil protection, national security and law enforcement. Both the commercial and public regulated services are encrypted to enable controlled access, with little information made public regarding the public regulated service (Hofmann-Wellenhof et al., 2008). Galileo uses four carrier frequencies - E1 at $1575.42\,\mathrm{MHz}$, which is the same as GPS L1; E6 at $1278.75\,\mathrm{MHz}$; E5a at $1176.45\,\mathrm{MHz}$, which is the same as GPS L5; and E5b at $1207.14\,\mathrm{MHz}$. Both the E1 and E6 signals have three components (denoted as A, B and C), whereas the E5a and E5b signals contain two components (I and Q), with the C and Q components designed to be pilot codes (Hofmann-Wellenhof et al., 2008). E1-A and E6-A are used solely for the PRS, with E6-B and C being used solely for the CS. As these are both encrypted, we shall not discuss them further. E1-A, E1-B, E5b-I and Q are shared between OS, CS and SoL, with E5a-I and Q being shared between OS and CS (Hofmann-Wellenhof et al., 2008). As these signals are shared between multiple services, the entire signal is not encrypted, as it is with the E6 and GPS P(Y) signal, instead only certain data fields in the navigation message are encrypted.

Galileo, unlike GPS, uses a tiered code sequence, where a primary and secondary code are added together, using modulo-2 addition, to produce the final code. The E1-B and C signals have a primary code length of 4092 chips, and a secondary length of 1 and 25 chips, respectively; with the E5a and E5b signals having a primary code length of $10\,230$ chips, and a secondary code length of 100 chips for the Q pilot signals, 20 chips for E5a-I and 4 chips for E5b-I. This is designed to increase the robustness of the signal, as the overall code is large, whilst still allowing a short acquisition time, due to the short repetitive period (Hofmann-Wellenhof et al., 2008). The E1-A and B signals have a chipping rate of $1.023\,\mathrm{Mcps}$ and use MBOC modulation; whilst the E5a and E5b signals have a chipping rate of $10.23\,\mathrm{Mcps}$ and use BPSK modulation.

1.5.2 GNSS receivers

The first space-bourne GPS receiver was flown on NASA's Landsat-4 (Montenbruck et al., 2008), which was launched in July 1982 (NASA, 2013), since then, GNSS receivers have been included on a wide range of satellites, including cubesats (Bouwmeester and Guo, 2010). GPS receivers are available in both single and dual frequency configurations - single frequency receivers use the commercial L1 signal, whilst dual frequency receivers additionally use the L2 signal. As the P(Y) code is encrypted and only available for military applications, dual frequency receivers use signal processing techniques that do not rely on being able to decrypt the signal. The first is known as codeless processing, where it is assumed that the same P(Y) signal is broadcast on both L1 and L2 (Kaplan and Hegarty, 2006), whereas the second technique, known as semi-codeless processing, relies on the encryption rate being approximately $\frac{1}{20}^{\text{th}}$ of the P(Y) chip rate (Kaplan and Hegarty, 2006; Woo, 2000). Both of these techniques increase the accuracy of the receiver

as the P(Y) chipping rate is higher, allowing the range from the satellites to be calculated to a higher accuracy, and the use of two different frequencies allows the receiver to compensate for the ionospheric delay in the signals to a better extent than with the navigation message based correction, decreasing the uncertainty in the position. The accuracy of a receiver is largely dependant on the environment, as well as the receiver itself; however, in LEO, a single frequency GPS receiver can have a spatial accuracy of approximately $10\,\mathrm{m}$, with a timing accuracy better than $1\,\mu\mathrm{s}$; with a dual frequency receiver achieving a spatial accuracy of less than $1\,\mathrm{m}$ and approximately $5\,\mathrm{cm}$, when post-processing is used (Montenbruck et al., 2008). This basic order of magnitude increase in accuracy is due to the order of magnitude difference in the chipping rates of the C/A and P(Y) codes.

Commercial receivers

A typical GNSS receiver, as found in most mobile devices (such as mobile phones and tablets), is not capable of being used in space for several reasons. As the satellites, and possibly the receiver, are moving, the signals detected by the receiver are Doppler shifted. This results in the receiver having to scan a two dimensional search space to acquire the satellites, one consisting of the possible Doppler shifts and the other being the satellite PRN codes[7]. As the majority of GNSS receivers are designed to work on Earth, the Doppler shifts are optimised to the speeds the receiver is likely to observe, as an excessively increased search space would only increase the time required to acquire the satellites. This means that the typical speeds in LEO of around $7800\,\mathrm{m\,s^{-1}}$, will produce Doppler shifts that are significantly greater than those the receiver searches for - just as an example, a Boeing 747 cruises at $248\,\mathrm{m\,s^{-1}}$ and an Airbus A380 at $262\,\mathrm{m\,s^{-1}}$ (Boeing, 2013; Airbus, 2017b) - therefore, the receiver will be unable to acquire the satellites.

The second reason is a regulatory one - as GNSS receivers that can work in space, could be used in ballistic missiles. Due to this, export restrictions are placed on GNSS receivers by different countries. The European Union has the 'EU dual-use regulations' that regulates the export of goods that can be used in both civil and military applications and applies to all EU countries, which requires any GNSS receiving equipment that is designed, or modified, to provide navigation information at speeds greater than $600\,\mathrm{m\,s^{-1}}$, or designed, or modified, for use on space vehicles, unmanned aerial vehicles or sounding rockets, to have an export licence (Council of the European Union, 2009). The US government has similar regulations, called ITAR (International Traffic in Arms Regulations), which classifies any GNSS receiver that is capable of functioning above an altitude of $60\,000\,\mathrm{ft}$ ($18.3\,\mathrm{km}$) and a velocity greater than $1000\,\mathrm{knots}$ ($514\,\mathrm{m\,s^{-1}}$, $1151\,\mathrm{mph}$) as munitions, which require an export licence (Electronic Code of Federal Regulations, 2013). To simplify the import and export of devices to different markets, most GNSS receiver

[7]Which, in itself, is a two dimensional search space, as each satellite has a unique code and the codes are delayed due to the receiver's position.

Table 1.2: Space-rated commercial GNSS receivers.

Name	Type	TID (kRad)	Power (W)	Dimensions (mm^3)	Mass (g)
SSBV GPS receiver[a]	12 ch L1	>10	<1	50x20x5	<30
SSTL SGR-05U[b]	12 ch L1	>10	0.8	70x40x15	52
SSTL SGR-05P[c]	12 ch L1	>10	1	105x65x12	60
SSTL SGR-07[d]	12 ch L1	>10	1.6	120x78x48	450
SSTL SGR-10[e]	24 ch L1	>10	5.5	160x160x50	950
SSTL SGR-20[f]	24 ch L1	>10	5.5	195x162x48	950
DLR Phoenix-S[g]	12 ch L1	15	0.9		20
SpaceQuest GPS-12-V1[h]			1.25	100x70x25	20
ASTRA GAMMA[i]	40 ch L1 & L2		5	102x95x32	200
General Dynamics Viceroy-4[j]	18 ch L1		7	152x132x43	1100
General Dynamics Explorer[k]	12 ch L1		7	160x132x43	1200
RUAG innovative GNSS[l]	24 ch L1 & L2	>20	<8	300x240x50	1300
Thales Alenia Space TopStar[g]	16 ch L1	>30	1.5		1500
General Dynamics Monarch[m]	12 ch L1 & L2	100	25	205x200x140	3720
Airbus MosaicGNSS[n]	8 ch L1	100	10	272x284x92	3900
Thales Alenia Space Tensor[g]	9 ch L1	100	15		4000

[a] CubeSatShop (2013). [b] SSTL (2013b). [c] SSTL (2013a).
[d] SSTL (2013c). [e] SSTL (2013d). [f] SSTL (2013e).
[g] Montenbruck and D'Amico (2013). [h] SpaceQuest (2013).
[i] ASTRA (2015). [j] General Dynamics Mission Systems (2013c).
[k] General Dynamics Mission Systems (2013a). [l] RUAG (2013).
[m] General Dynamics Mission Systems (2013b). [n] Airbus (2017a).

manufactures limit their receivers to the US government's requirements, which prevents the receiver from being used in satellites.

Additionally, terrestrial GNSS receivers do not have any radiation protection and so have an increased chance of failure when exposed to space conditions. To meet these environmental requirements, specialist space-rated GNSS receivers have been designed and manufactured by several companies, with a non-exhaustive list shown in table 1.2. The majority of space-rated GNSS receivers, as shown in table 1.2, are designed for large spacecraft, with few solutions existing for pico-satellites, such as cubesats, and none existing for femto-satellites. Both the SSTL SGR-05U and the SSBV GPS receivers could be used on a cubesat platform, however, their power consumption is only really suitable for cubesats larger than 1U, or with deployable solar panels, as the average power budget for a 1U cubesat is $1\,\mathrm{W}$ (Montenbruck et al., 2008). Additionally, the unit cost of all the receivers is quite high (ranging from approximately £10k to £165k), due to the specialisation and the limited market of space-rated GNSS receivers (Montenbruck et al., 2008); and is high enough to limit their use in a pico-satellite constellation or swarm, as the cost of each node would be prohibitive.

Scientific receivers

As GNSS is a requirement for many scientific missions, either for precise orbit determination or as a scientific instrument, custom receivers have been made by different institutions. One notable receiver is ESA's AGGA-4, which is designed to be versatile with support for 36 single, or 18 dual, frequency channels and the GPS, Galileo and COMPASS constellations (Guasch et al., 2010; Rosello et al., 2012). The receiver is designed as an ASIC, without an RF front-end, using 6 million gates and manufactured using a $180\,\mathrm{nm}$ technology. It is packaged into a 352 pin MQFP and has many different interfaces including UART, SPI, GPIO, SpaceWire and Mil-Std-1553 (Rosello et al., 2012). AGGA-4 is a progression of a previous ESA design, the AGGA-2, with the improvements being an embedded LEON2-FT processor (with FPU), on-chip 128-point FFT and CRC modules, and 5 complex correlators per channel, compared with 3 per channel of the AGGA-2 receiver (Rosello et al., 2012). As the AGGA-4 is designed primarily for ESA's use on large spacecraft, it is unsuitable for use on a cubesat due to power constraints. Although the exact power consumption is not available, a presentation on the AGGA-2 receiver (Hollreiser, 2001), states that the power consumption for 4 dual channels is between 1.2 and $3\,\mathrm{W}$, and that the AGGA-2 is used in the Astrium MosaicGNSS receiver (see table 1.2), which has a maximum power of $10\,\mathrm{W}$. As the AGGA-4 has more gates than the AGGA-2, albeit using a small manufacturing technology, and contains an embedded processor, it is likely to consume more power than its predecessor. It is, therefore, unlikely that it will fit into the power constraints of a cubesat.

Grondin et al. (2010) present a GNSS receiver designed for micro-satellites in LEO, in particular two satellites, MicroSCOPE, that was launched in April 2016, and TARANIS, that is due for launch in 2018 (ESA, 2016a,b; CNES, 2017). Although it is only suitable for micro-satellites and larger, due to its volume ($180 \times 140 \times 40\,\mathrm{mm}^3$), mass (less than $900\,\mathrm{g}$) and power consumption (less than $5\,\mathrm{W}$, with a peak of less than $8\,\mathrm{W}$), it is particularly well described. It is designed to have two operating modes, the first is a normal mode where it operates continuously and the second is a fractionated mode, where it minimises power by only working in short bursts. The fractionated mode decreases the power by approximately a factor of five (to approximately $1\,\mathrm{W}$), but has the disadvantage of also decreasing the accuracy, which, unfortunately, is not quantified by Grondin et al. (2010). The receiver uses an FPGA and a DSP for processing the signal, with the FPGA being used for filtering, IF compensation and signal tracking, and the DSP for signal acquisition (using FFT) and calculating the navigation solution. However, when the receiver is operating in fractionated mode, the FPGA does not do any signal tracking - effectively the signal is captured and then post-processed, by the DSP, to find the navigation solution. The idea of a fractionated mode, as a method of reducing the required power, is quite logical and, for a lot of scientific missions, could be used as the main operating mode of the

receiver, especially if the dilution of accuracy could be reduced by using accelerometer data or externally obtained orbit data, either in real time or by using post-processing.

The receiver has a similar performance to typical terrestrial receivers, with a spatial accuracy of $15\,\mathrm{m}$ and a timing accuracy of less than $5\,\mu\mathrm{s}$, when in its normal mode. It has a cold-start time of approximately 120 seconds, which is reduced to approximately 5 seconds when aided, presumably by providing the ephemeris or almanac data. Despite the use of COTS components, the receiver has a TID of $10\,\mathrm{kRad}$, which is possible as the FPGA and DSP were chosen for their radiation robustness. SEUs are mitigated by the memory being periodically scrubbed and safety circuitry is used to protect against latch-up conditions. Although the safety circuitry is not described, it is likely to be a current limiting device, which interrupts the power when an over-current condition is detected. It is also unfortunate that Grondin et al. (2010) do not state whether the FPGA and DSP used in the receiver design are space-rated or standard commercial grade parts, although it is likely that these are not space-rated, as space-rated FPGAs typically have higher TIDs[8].

Tang et al. (2012) present a low power GPS baseband processor that is designed to be implemented in hardware, with transistor level simulations showing a power consumption of less than $1.5\,\mathrm{mW}$ for 6 channels, whilst having an accuracy of less than $4\,\mathrm{m}$, when simulated against ideal signals. Although the baseband processor does not include an RF front-end, with the authors quoting that $10\,\mathrm{mW}$ front-ends have been designed, and requires further processing with a DSP, or microcontroller, to produce a navigation solution, the design is very promising. It is quite possible that, using this design, a complete GPS receiver could be built that would have a power consumption under $200\,\mathrm{mW}$, that would be well suited for femto and pico-satellites. Unfortunately, the design is not radiation-hardened, has not been tested in hardware and can only use the GPS constellation. Additionally, Tang et al. (2012) do not state how long it takes for the first fix to be obtained or what the maximum number of channels for their design is. To achieve a low power, Tang et al. (2012) use a methodology of reusing components, for example, each channel in their design uses mathematical functions for the tracking loop and accumulator calculations, but these functions are only required at the end of each loop (that is every millisecond), so each mathematical block is shared by up to 6 channels. This methodology of component reuse, could be used in many embedded applications to reduce power consumption.

Previously flown on pico-satellites

Many cubesats have been flown with GPS receivers (Bouwmeester and Guo, 2010), however, due to the size and power consumption of many available space-rated commercial GPS receivers (see table 1.2), some cubesat missions have used COTS GPS receivers

[8]For example the Xilinx Virtex-4QV has a TID of $300\,\mathrm{kRad}$ and the Virtex-5QV has a TID of greater than $1\,\mathrm{MRad}$ (Xilinx Inc., 2010b, 2011d).

modified for use in space. The CanX-2 cubesat used the NovAtel OEM4-G2L receiver with the altitude and velocity restrictions removed, however, the tropospheric correction was not removed, causing an error of between 10 and $20\,\mathrm{m}$, resulting in an overall positionally accuracy of approximately $30\,\mathrm{m}$ (Spangelo et al., 2013). The OEM4-G2L measures $60 \times 100 \times 16\,\mathrm{mm}^3$, has a mass of $56\,\mathrm{g}$ and a typical power consumption of $1.6\,\mathrm{W}$, which is large enough to require a cubesat that is larger than 1U (NovAtel Inc., 2006a). The Radio Aurora eXplorer (RAX) mission used a NovAtel OEMV-1-L1 receiver on each of the two satellites, with the altitude and velocity limits, and, learning from CanX-2, the tropospheric corrections, removed. They additionally extended the Doppler window in the receiver, to account for the higher velocities of satellites (Spangelo et al., 2013). The OEMV-1-L1 is smaller than the OEM4-G2L, at $46 \times 71 \times 13\,\mathrm{mm}^3$, has less than half its mass, at $21.5\,\mathrm{g}$ and has a lower power consumption at $1\,\mathrm{W}$ (NovAtel Inc., 2011). The cost of the OEM4-G2L is not known, but Spangelo et al. (2013) state that the cost of the GPS receivers, used in the RAX mission, was less than $3,000. Additionally, Spangelo et al. (2013) cite the GPS receiver as the cause for both of their satellites failing after a month, for the first satellite, and two weeks, for the second satellite. The exact cause of this failure was deemed to be unexpected electrical interference produced by the GPS receiver that caused the power supply system to fail (Spangelo et al., 2013), highlighting the potential problems of using a black-box system.

There are many potential problems with using a commercial GPS receiver not designed for space, besides the lack of radiation tolerance. The first is the difference between the environments, requiring modification of the receiver to allow for a higher altitude, a larger Doppler range and to remove corrections that are not applicable (e.g. tropospheric corrections). The second potential problem is that these modifications have to be performed to the receiver's firmware, with the modifications, due to the propriety nature of the firmware, being subject to the manufacturer's discretion. This is potentially a problem as the manufacturer can, at any point in time, withdraw the ability to modify the firmware. Additionally, these modifications are likely to be outside the manufacturer's specification, with the receiver operating in a non-tested state. These problems mean that the power consumption is likely to be more, that the receiver has a larger error in the calculated navigation solution, interferes with other components or behaves unexpectedly. As the length of the mission, from first conception to flight, can span the majority of a decade, or longer, it is ideal for the components used in the satellites to be available several years later. Often, with commercial products, the product's lifetime is a lot shorter than this - an example of this is the OEMV-1-L1, which was flown on the RAX satellites in October 2010 and November 2011, that, as of October 2012, is no longer being manufactured (Spangelo et al., 2013; NovAtel Inc., 2012). The OEMV-1-L1 first became available in April 2006 (NovAtel Inc., 2006b), therefore, it was available for a total of 6 years and 3 months.

Challenges for GNSS receivers

Montenbruck et al. (2008) state that there are four key challenges that have to be met regarding GPS receivers - miniaturisation, increased accuracy and robustness, support of new signals, and advanced science applications. Additionally, there is a challenge of minimising the power consumption so that such receivers can be used on 1U cubesats and femto-satellites. For many satellite applications, the start up time of the GNSS receiver can be problematic, with space-rated GPS receivers typically having a cold start time of between 10 and 15 minutes (Montenbruck et al., 2008). This can be reduced by providing the almanac through the satellite's uplink, but even with this, GNSS receivers do not offer instant-on positioning. This is improved with the new safety-of-life signals in both Galileo and GPS, which could potentially reduce the overall power consumption of a fractionated GNSS receiver.

1.6 Summary

The highly dynamic nature of ionospheric plasma depletions requires both high temporal and spatial resolution from ionospheric measurements to further the understanding of how they form, develop and dissipate. The most cost effective way of achieving this is through the use of a large number of low-cost satellites, with the use of femto-satellites being very attractive. The sensor technology to measure the ionospheric plasma on such a small satellite is available, however, only a small number of femto-satellites have ever been flown. Pico- and nano-satellites have been flown in significantly higher numbers, with the majority designed as testing platforms for a particular new piece of technology. Whilst they frequently lack attitude and orbit control systems, they make use of COTS components to reduce costs and allow them to perform more complicated tasks. Conventional attitude and orbit control are unlikely to be suitable for femto-satellites, due to size and power constraints. Constellations of larger commercial and scientific satellites exist, however, inter-satellite links are frequently designed out where possible, to reduce the complexity of the constellation. In a constellation, or swarm, of smaller satellites, which are dispersed over a smaller volume, the use of inter-satellite communication could be both easier, through the use of COTS components, and a requirement of the mission. Overall, the current main limitation in using femto-satellites for high temporal and spatial resolution ionospheric measurements is not in their cost, ability to collect the data, or to get the data to a ground station, but in their ability to know their own location with a high enough precision to make the measurements meaningful.

Chapter 2

Mission concept

The University of Leicester has proposed a concept for a combined pico- and femto-satellite mission, consisting of a single pico-satellite, a 3U cubesat, and a single femto-satellite, a PCBsat (Vladimirova et al., 2011). The design of the PCBsat is based on the initial design of Barnhart et al. (2009) and would form part of the cubesat's structure at launch. During flight, the cubesat would deploy the PCBsat on a tether, so that the PCBsat could be fully tested whilst guaranteeing it to be in communication range of the cubesat. At a later point in the mission, the tether would be cut to test how the satellites separate, the communication over a longer distance and so that the PCBsat can test its scientific payload. The overall aim of this mission is to perform an in-orbit test of the technologies required to create a space weather monitoring satellite swarm, whilst providing useful data from additional payloads on the cubesat (which are outside the scope of this work). The eventual aim is a significantly larger mission - a satellite swarm mainly consisting of PCBsats, that perform data collection, with the data being relayed to a smaller number of cubesats, dispersed among the PCBsats, which provide a ground link. Therefore, the PCBsat needs to be designed so little, if any, changes are needed between the two missions. The original design of the PCBsat was a proof of concept, to show that a femto-satellite could be used, and is, unfortunately, not flight ready.

This chapter discusses the original design and ways to modernise it. In section 2.1.1 a power budget is constructed, with section 2.2 discussing possible location techniques that could be used.

2.1 PCBsat design

The original PCBsat, designed by Barnhart et al. (2009) and hence will be referred to as the 'Barnhart PCBsat', has already been discussed in section 1.3.2, however, it is now briefly described to illustrate the proposed changes. It was designed to use COTS components to reduce manufacturing costs, with the effects of the increased exposure to ionising

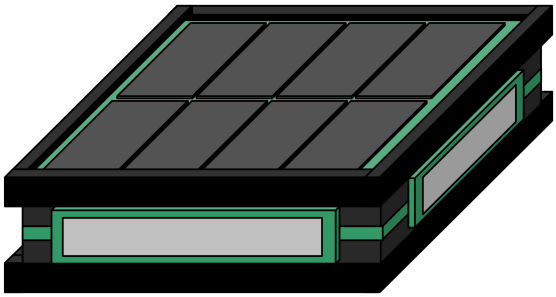


Figure 2.1: Concept drawing of the PCBsat (Barnhart, 2008).

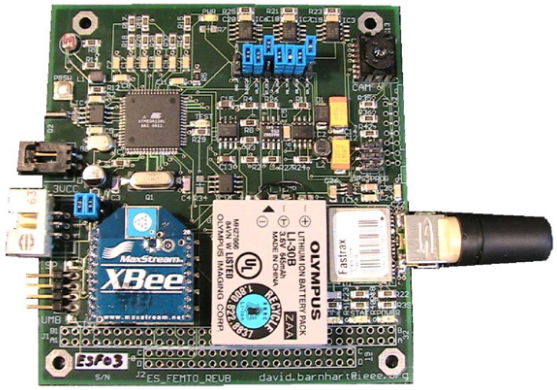


Figure 2.2: The prototype version of the Barnhart PCBsat (Barnhart et al., 2009).

radiation, common in satellite orbits, largely being disregarded, due to the short mission life.

The Barnhart design, shown in figure 2.2, is based around an 8 bit microcontroller (Atmel ATmega128L), serial flash memory (16 Mb) and a $900\,\mathrm{MHz}$ mesh-network radio. It has a main payload of the MESA sensor, with a commercial GPS receiver and a VGA CMOS camera. For housekeeping, there is a real time clock and temperature sensor, and there is a Sun sensor for simple attitude determination. Power is provided by single junction GaAs solar cells on the two largest faces and a $2322\,\mathrm{mWh}$ ($645\,\mathrm{mAh}$ at $3.6\,\mathrm{V}$) Li-ion battery (Barnhart, 2008).

Whilst the Barnhart design was fully functional on the ground, certain parts of it would not be able to function in space, others might cause problems and some can be updated so that they are better suited for a satellite. The first major problem is the use of a commercial, non-space-rated GPS receiver. As explained in section 1.5, non-space-rated commercial GPS, and GNSS, receivers are limited to only work below a certain altitude ($18\,\mathrm{km}$) and a certain velocity (less than $600\,\mathrm{m\,s^{-1}}$), and even with these limitations removed, the majority are unlikely to work due to a significantly increased Doppler shift of the received signals. This is a severe problem with the original design, as it means that location information could not be obtained once the satellite was in orbit. Additionally, the use of a $900\,\mathrm{MHz}$ radio could cause problems, as it is only part of the license-free ISM

band in some countries, not worldwide, which might restrict its use in space. Additionally, the solar cells used in the original design have a fairly low efficiency compared to that of commonly used triple-junction GaAs cells, and so could be upgraded to increase the flexibility of the design.

2.1.1 Modernised PCBsat

In the years since the Barnhart PCBsat was designed, technology has progressed in many ways. One of these is the fairly recent availability of ferroelectric RAM (FRAM, or sometimes FeRAM) based microcontrollers. FRAM is a non-volatile storage that stores data by using a ferroelectric film - a material that can be polarised, like a permanent magnet (with similar hysteresis), using an electric field (rather than a magnetic field). As the data is stored as a crystal polarisation, it is substantially less susceptible to the effects of ionising radiation as flash memory is - which relies on a charge being stored on a floating gate. This essentially means that FRAM microcontrollers are still susceptible to single event upsets (SEUs), but the microcontroller's program and data memory are practically not, allowing their use in a satellite to increase the reliability over that of non-radiation hardened microcontroller.

One FRAM based microcontroller is the TI MSP430FR5738. It is a 16 bit microcontroller that operates at up to $24\,\mathrm{MHz}$, which is three times faster than the ATmega128L, whilst having a maximum power consumption of $12.9\,\mathrm{mW}$, compared to the $55\,\mathrm{mW}$ of the ATmega128L (Texas Instruments, 2011; Atmel Corporation, 2011). Whilst it is not fair to compare the clock frequencies of microcontrollers with different architectures, it is possible to say that the MSP430FR5738 is likely to perform significantly more efficiently, with its 16 bit architecture and lower power consumption. Additionally, the MSP430FR5738 includes a real time clock, which simplifies the overall design and also decreases the power consumption.

The preliminary design for the modernised PCBsat includes an accelerometer as a secondary payload rather than a camera. Whilst camera images from satellites are very useful for publicity, they add very little scientifically to this mission[1]. The Barnhart design envisaged the camera being used to capture the dispersion of the PCBsats through images taken at random times (Barnhart, 2008). However, there is little chance that the captured images would show the other PCBsats, as, under the best conditions, the camera of one PCBsat would be looking at one of the smallest faces on the rear of a leading PCBsat. As a consequence, the majority of images would show very little and so would require a considerable amount of processing on the PCBsats to determine which images were useful, to avoid transmitting useless data. Although an accelerometer would not provide

[1] However, a camera on the cubesat could be useful to help determine the state of the tether and the deployment of the PCBsat.

Table 2.1: Time averaged power generation for a variety of solar array configurations. Figures calculated assuming a solar irradiance of $135.3\,\mathrm{mW/cm}$ (Spectrolab Inc., 2008).

Solar cell		Estimated time-averaged power[a] (mW)			
Type	Area (cm^2) / Efficiency	56	64	81	100
Generic					
Silicon	11.0%	417	476	603	744
GaAs triple junction	28.0%	1061	1212	1534	1894
Spectrolab (GaAs)					
ITJ[b]	26.8%	1015	1160	1469	1813
UTJ[c]	28.3%	1072	1225	1551	1915
XTJ[d]	29.5%	1118	1277	1617	1996

[a] When illuminated. [b] Spectrolab Inc. (2008). [c] Spectrolab Inc. (2010a).
[d] Spectrolab Inc. (2010b).

information on the dispersion of the PCBsats, without post-processing, it would allow the effects of atmospheric drag on the PCBsat to be measured, which would be useful for future missions. The presence of a GPS receiver in the Barnhart design, and some kind of location technology in the modernised design, would allow the dispersion to be observed, providing that the PCBsats and the communication network functioned correctly.

Many space-rated receivers are available (see section 1.5), however, they are typically too large, in both size, mass and power consumption, for the PCBsat. To be able to judge what location technologies can be used on the PCBsat, it is necessary to construct a power budget.

Power budget

Due to the dimensions of the PCBsat, the only practical place for the solar cells is the two side faces, that measure 10 by $10\,\mathrm{cm}^2$. For a first order approximation, we consider the PCBsat to not have any attitude control, instead it is assumed that it is tumbling at a constant rate. Whilst this is unlikely to be true, it provides a good approximation of the illumination characteristics. At a given distance from the PCBsat, each length of the side face will vary sinusoidally, with the rotation, from its true length to zero. Therefore, the visible area of the side face will vary like a squared sinusoidal. With the time average of a squared sinusoidal being $1/2$, the effect of the PCBsat's rotation can be approximated as reducing the power output by half of that of a static PCBsat, with the incident radiation normal to the solar panels. From this, it is possible to calculate the average solar power for different solar arrays, which is shown in table 2.1. The four areas chosen are $56\,\mathrm{cm}^2$, the area of solar cells in the original PCBsat design (Barnhart, 2008), and 64, 81 and $100\,\mathrm{cm}^2$, respectively 8×8, 9×9 and $10 \times 10\,\mathrm{cm}^2$. Although a solar array with an area

Table 2.2: Solar power generation considering orbital eclipse. The minimum and maximum are based on the estimated power of 1 and 1.5 W, respectively.

Satellite altitude (km)	Orbit illuminated (%)	Average power (mW)	
		Minimum	Maximum
320	60.4	603.6	905.3
500	62.6	626.1	939.1
700	64.6	646.2	969.3

Table 2.3: Available power for different orbit altitudes, with required battery capacity.

Solar power (mW)	Power stored (%)	Altitude (km)	Orbit eclipsed (%)	Power (mW)		Minimum battery capacity (mWh)
				Lit	Eclipsed	
1000	50	320	39.6	500.0	609.0	365.1
		500	37.4	500.0	669.7	394.1
		700	35.4	500.0	730.6	424.7
1500	45	320	39.6	825.0	822.1	492.9
		500	37.4	825.0	904.1	532.0
		700	35.4	825.0	986.3	573.4

of $100\,\mathrm{cm}^2$ is not practical, it is included as the physical maximum. Disregarding the silicon solar cells, due to their low efficiency, the estimated power output is between 1 and $1.5\,\mathrm{W}$.

To calculate the power available, it is necessary to know how much of each orbit is spent in eclipse. There are a few methods of doing this, with one being simple trigonometry (see appendix B) which results in the eclipse angle being,

$$\theta = 2\arcsin\left(\frac{r_e}{r_e + r_s} - \frac{r_{sun} - r_e}{d}\right),$$

where r_e is the Earth's radius, r_s is the altitude of the satellite, r_{sun} is the radius of the Sun and d is the average distance between the Earth and the Sun. This can be easily used to calculate the amount of the orbit that is spent in eclipse. This is done in table 2.2, where the average power for several orbital altitudes is shown. This average power is not particularly useful for constructing a power budget, as it is averaged over an orbit, however, it illustrates how sensitive the orbital illumination is to altitude.

This orbital illumination information can be used to calculate how much available power there will be during both the lit and eclipsed parts of the orbit. This is shown in table 2.3, using the same likely orbits for space weather monitoring as table 2.2. The table uses a chosen percentage of the estimated power output of the solar cells ($1\,\mathrm{W}$ and $1.5\,\mathrm{W}$) to be stored for use during eclipse, with the minimum battery capacity to do this being shown. The percentage of the power stored was chosen to approximately equalise the lit and eclipsed power, with the power during eclipse considering a battery charge-discharge efficiency of $80\,\%$. Additional calculations, that have been omitted, have verified that the

charging rates of 500 and $675\,\mathrm{mW}$, implied by table 2.3, are within the charging limits of small li-ion batteries, that would be suitable for the PCBsat's form.

From table 2.3, the PCBsat would have between 500 and $825\,\mathrm{mW}$, when lit, and between 609 and $822\,\mathrm{mW}$, when in eclipse, for the lowest altitude considered[2]. Therefore, a conservative power budget should be based on maximum power targets of $500\,\mathrm{mW}$, when lit, and $600\,\mathrm{mW}$, when eclipsed. The preliminary power budget for the PCBsat is shown in table 2.4. As the PCBsat would operate differently depending on whether it is eclipsed or not - as the data collection from the MESA sensor is only performed during eclipse - the power budget is split into two. It should be noted that this power budget does not consider either the Sun sensors or magnetorquers, and uses estimates for the MESA sensor and radio, with the radio being intentionally excessive.

With such a restrictive power budget, none of the available GPS, or GNSS, receivers are suitable for the PCBsat. There is, therefore, a need to find, or create, an alternative to the commercially available location techniques that fits within the PCBsat's power constraint of less than $360\,\mathrm{mW}$.

2.2 Potential location techniques

There are many potential methods to locate an orbiting satellite, which can be classified into two categories - prediction and measurements. Predictive methods use orbital models to determine a satellite's position at a certain time, whilst measurement methods calculate the distance from known reference points - practically, this means ranging. For a swarm of satellites, Earth based ranging, that is bouncing a laser off of a target on the satellite, becomes impractical due to the number of satellites and would rely on predictive methods to calculate the location between measurements. The small size of pico- and femto-satellites, would make this significantly more difficult as the retro-reflecting target would be a lot smaller. Additionally, ranging using an Earth facing camera on the satellites could be problematic due to cloud cover and requiring a definitively shaped land mass, and would also not be particularly useful for measuring space weather as the position would have to measured at night, where the land masses are harder to distinguish and identify, or would rely on predictive measurements from daytime measurements.

2.2.1 Prediction

Predictive methods largely revolve around the use of the two line elements (TLEs) produced by NORAD, an example of which is shown in figure 2.3. These are orbital elements that are calculated for each satellite, or orbiting body, and are made available for public use. Because of this, they only have a positional accuracy of approximately $1\,\mathrm{km}$

[2]This is considered, as the PCBsats should be able to function fully as they de-orbit.

Table 2.4: Preliminary power budget.

Device	Operating mode	Maximum power (mW)	Lit		Eclipsed	
			Duty (%)	Power (mW)	Duty (%)	Power (mW)
Microcontroller (MSP430FR5738)	Active mode	12.9000	60	7.7400	60	7.7400
	Low power mode	1.3650	40	0.5460	40	0.5460
Accelerometer (ADXL345)	>100 Hz	0.3500	50	0.1750	50	0.1750
	Standby	0.0003	50	0.0001	50	0.0001
Magnetometer (MAG3110)	80 Hz	2.1600	50	1.0800	50	1.0800
	Standby	0.0048	50	0.0024	50	0.0024
Micro SD card (Sandisk)	Read/write (<25 MHz)	330.0000	10	33.0000	10	33.0000
	Standby	0.8250	0	0.0000	0	0.0000
	External power switch	0.0066	100	0.0066	100	0.0066
MESA sensor (estimate)		200.0000	0	0.0000	100	200.0000
Radio (estimate)	Receive	300.0000	100	300.0000	0	0.0000
	Transmit	2,000.0000	5	100.0000	0	0.0000
		Total		442.5501		242.5501
		Target		500.0000		600.0000
		Available		57.4498		357.4498

```
ISS (ZARYA)
1 25544U 98067A   16226.93779845  .00005281  00000-0  84287-4 0  9998
2 25544  51.6448 129.8623 0002079 130.1779 315.5703 15.55001256 13971
```

Figure 2.3: A TLE for the International Space Station (ISS). The first line contains the satellite's name. The first line of data contains (in order from left to right), the line number, the satellite number, the international designator, the epoch of the TLE, the first and second derivatives of the mean motion, the BSTAR drag term, the ephemeris type (which is always 0) and an element number (which is incremented for each TLE). The second line of data also contains the line and satellite numbers, but then has the orbit inclination, the right ascension of the ascending node, the eccentricity, the argument of perigee, the mean anomaly, the mean motion and the revolution number at epoch.

at epoch (Vallado et al., 2006a). To calculate a satellite's position at a time before or after the epoch, a propagator has to be used, with SGP4 being the only propagator that the TLEs are designed to work with (Vallado et al., 2006a). SGP4 was first published in 1980 and considers the atmospheric effects on a satellite through a ballistic drag coefficient, however, this coefficient frequently changes for satellites and it is fairly common for it to become negative at some epochs, which is unrealistic, to account for inaccuracies in the propagation model. Using SGP4, the accuracy of a satellite's position decreases by approximately 1 to $3\,\mathrm{km}$ per day (Vallado et al., 2006a,b). This, combined with the TLEs only being published for a satellite if its positional accuracy meets certain unknown criteria, results in a positional accuracy of a satellite that can be severely inaccurate for the purpose of space weather monitoring.

As the TLEs are only accurate to approximately $1\,\mathrm{km}$ at epoch, any propagators that use them also have this level of inaccuracy, making them less than ideal for the measurement of space weather phenomena, prior to considering the inaccuracies of each individual model.

Additionally, to be able to calculate the satellite's position, each satellite would have to record the time of each measurement with a reasonable degree of accuracy, which requires an accurate time source. A typical watch crystal has a frequency stability of approximately $\pm 20\,\mathrm{ppm}$ (ABRACON Corporation, 2010), which, excluding temperature effects, will drift by 1.73 seconds every 24 hours. To maintain a clock accuracy of $10\,\mathrm{ms}$, which at orbital velocities is around $100\,\mathrm{m}$, the clock would have to be synchronised with a high precision clock at least once every 8.3 minutes. Higher accuracy crystals are available that have a frequency stability of $\pm 0.5\,\mathrm{ppm}$ (IQD Frequency Products, 2012), however, to maintain the same clock accuracy, this would have to be synchronised at least once every 330 minutes, which is approximately every 5.5 hours. In addition to this, these higher stability crystal are significantly larger than a standard watch crystal, they consume

more power and cost approximately 10 times more; however, they are still feasible to use on a femto-satellite, such as the PCBsat.

The choice of the synchronisation clock is also very important to the accuracy of the clocks on the satellites. If each of the cubesats were to have high stability clocks, perhaps relying on the reception of GNSS signals, then the communication network would have to be able to relay the time to all of the PCBsat nodes within a few milliseconds. This relaying would have the potential problem that the PCBsat nodes would not be synchronised with each other, hindering data analysis through ambiguity. Terrestrial time broadcasts, commonly used in radio controlled clocks, could be utilised, but unfortunately, there is no worldwide standard, with different countries using different frequencies and encodings. This would result in a large number of protocols having to be supported by each satellite to ensure an accurate clock, whilst there would be no guarantee that the time broadcasts could be received in space under all circumstances.

Satellite time broadcasts, such as those from GNSS, could be used to ensure a high accuracy clock. However, as this would require each node to have a GNSS receiver, it would be illogical to use predictive methods for location, as a more accurate position would already be known.

2.2.2 Satellite ranging

As seen is section 1.4, it is possible to use satellite ranging to locate satellites. However, the use of a microwave link, such as that used in the GRACE satellites, or the occulted solar chronograph, used in the PROBA-3 mission, requires a fixed satellite separation in a predetermined geometry. With the intention of the PCBsats to disperse over the mission lifetime, these methods are not suitable. It is possible to design an optical or radio based ranging system, however, these are not particularly practical.

Firstly, an optical based system, such as three evenly spaced lights mounted on each face of the PCBsat, would allow the distance to be calculated by analysing captured images from other nodes. However, this would require that each PCBsat has a camera on each face as well as having the necessary processing power to analyse the images. This is likely to require too much power and be too large for the PCBsat, as well as not providing any method of synchronising the clocks between nodes. Additionally, the positions of the PCBsats would only be known relative to each other, requiring the cubesats to be used to locate the swarm absolutely.

Secondly, an RF based system would either require time of flight signals, essentially RF echoes, or for the nodes, at least the cubesat nodes, to have very accurate and precise clocks. Time of flight signals have the issue that they have to be transmitted by the PCBsats over a potentially large distance to several nodes, which could consume a large amount of power and become increasingly difficult to manage when there are a large

number of nodes. Time of flight signals would also not provide a way to perform clock synchronisation. If, instead, accurate and precise clocks were used on the cubesat nodes, then effectively a custom GNSS would be created. This has the issue that it would be less accurate than current GNSS and would require knowing the exact location of each of the cubesats, essentially resulting in a GNSS that would be less reliable than current systems. The only advantage to doing this would be if the PCBsat nodes were able to consume less power using a custom system, and this lower power consumption out-weighed the loss in accuracy, however, this is highly unlikely.

2.2.3 Summary

Predictive methods do not have the required accuracy for measuring space weather phenomena, where measurements require a positional accuracy of at least $100\,\mathrm{m}$, due to the methods relying on TLE data that is, at best, only accurate to $1\,\mathrm{km}$. Earth based and optical satellite ranging are impractical for the small size of the PCBsats and the potentially large number of nodes. With any RF based satellite ranging methods resembling a less accurate version of current GNSS solutions. Therefore, the only practical method of finding the precise location of nodes in a dispersive swarm is to use current GNSS.

Chapter 3

GNSS receiver design

As described in section 1.5, GNSS utilise very similar signals, frequently with the same encoding methods. This results in a typical architecture for a GNSS receiver, that varies very little. This typical architecture, as shown in figure 3.1, consists of five distinct parts. The first is the RF front-end, that down-converts the received signal from the antenna to an intermediate frequency - from the order of gigahertz to tens of megahertz. The front-end then digitises the signal, with 1 to 3 bits being typically used, although some software defined radios will use 8 bits[1]. We will refer to this digitised data from the front-end as being the baseband data, which is fed into multiple tracking channels. Each tracking channel despreads the received signal with one of the satellite's PRN codes, by tracking the frequency offset and code delay of the signal, and, if the signal is present, will output a decoded data stream, alongside information such as the satellite's pseudorange (the perceived distance between the satellite and the receiver). This data stream is then processed by the navigation processor individually, to determine the satellite's position and apply any necessary corrections, and together with the data from other tracking channels to calculate the location of the receiver (referred to as the navigation solution). The navigation processor will then output the location of the receiver at a set rate and using a predefined protocol.

[1]However, it should be noted, that this level of quantisation is excessive for GNSS signals.

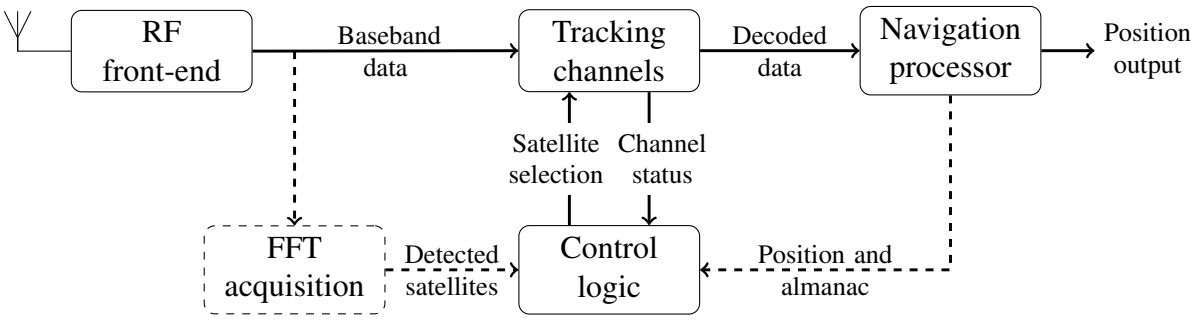


Figure 3.1: Typical GNSS architecture, optional elements are shown with dashed lines.

To manage the satellite tracking, the receiver has some control logic that determines which satellites are assigned to the tracking channels. The complexity of this varies, from assigning new satellites to tracking channels that have been unable to detect a signal, in a round-robin fashion, to using the position of the receiver with estimates of the satellite locations (the almanac) to choose potentially visible satellites. One method of assisting the control logic, is the addition of an FFT acquisition unit. This provides a method of determining which satellites are visible and estimating the required tracking parameters without any prior knowledge.

The PRN codes in a system are designed to be orthogonal, that is, they are chosen so that they cross-correlate poorly. When used in the time domain, the PRN codes will only correlate strongly when they are aligned, however, in the frequency domain, auto-correlation can be used, where two unaligned signals are used to produce a correlation peak that is delayed by the unaligned offset. The tracking channels can be used to search for a signal by iterating through the possible offsets, which, due to the large search space, can take a considerable amount of time. An FFT acquisition unit, however, can determine which satellites are visible in a smaller amount of time, but there are certain drawbacks to using the method. The main drawback is that the produced tracking parameters (the signal's frequency offset and code delay) are only estimates, with more accurate estimates requiring more data, memory and a larger processing time. In addition to this, the signal to noise ratio (SNR) required for an FFT based acquisition is higher than that of using the tracking channels to perform a slower time domain correlation. This means that some visible satellites will not be acquirable, whilst they are present, and trackable, in the signal. This is why an FFT acquisition unit is optional - it is not a required part of the receiver's architecture, but its addition can be advantageous.

3.1 Distributed receiver concept

The architecture in figure 3.1 implies that all the processes are concurrent, whilst this is true for most receivers, it is not true for all. The only requirement a GNSS receiver has is that the tracking of satellites is concurrent, so that the distance from at least four satellites is known at a given time, to calculate the receiver's position. It is possible to store the baseband data as it leaves the RF front-end. This is referred to as post-processing, as it allows the acquisition and tracking to performed at a later time and also repeatedly, which has several advantages.

The first is that a receiver can track a higher number of satellites, by using more tracking channels. Any receiver has a maximum number of channels, which is less than the number of transmitting satellites in a GNSS. Whilst it is not possible to see all the satellites at once, it is possible for more to be visible than the receiver can concurrently track. When post-processing is used, the tracking channels do not have to run at the same

time, they just have to run so that they are concurrent with the data (i.e. the timing is based on the current sample number rather than a clock), which allows the same channel to be used for tracking multiple satellites, resulting in the possibility of tracking more satellites. In addition to this, the tracking channels, as they do not have to run in real-time, can be designed to have higher precision and greater accuracy. The second advantage is that the receiver can acquire and track satellites that it would not otherwise be able to. Due to the signal type, the SNR required for tracking is lower than that required for acquisition. In a post-processing receiver, when the SNR of a satellite changes, from low to high, it is possible to use the tracking parameters used during the higher SNR period to allow the receiver to acquire and track the satellites in the low SNR period. Whilst this does require multiple iterations of the receiver, it can be beneficial. The third advantage is that the positional accuracy can be increased when using post-processing. Part of this is fairly obvious: if you use more satellites, you can more accurately pinpoint the location of the receiver; however, it is also possible to more accurately know the location of the satellites. The navigation message from each satellite (the data that the tracking channels decode), contains an estimate of the satellite's position in terms of orbital elements (the ephemeris). These orbital elements are predictions and have a certain amount of error associated with them. Higher accuracy ephemerides are produced by the International GNSS Service (IGS), by utilising a range of ground based monitoring stations. By using the IGS data, the positional accuracy of GPS satellites can be increased from the navigation message's $100\,\mathrm{cm}$ to approximately $2.5\,\mathrm{cm}$, with the timing accuracy increasing from a standard deviation of $2.5\,\mathrm{ns}$ to approximately $20\,\mathrm{ps}$ (International GNSS Service, 2014). However, it can take the IGS up to 18 days to produce these improved accuracy ephemerides, as they use data collected over periods of weeks, from different sources (including laser ranging), to model the movement of the satellites, whilst trying to minimise any sources of measurement error.

Whilst post-processing has many advantages, it has some significant disadvantages. The primary of these is the amount of data storage that is required to capture the baseband data. The smallest possible data rate is 1 bit at $1.023\,\mathrm{MHz}$ (the chipping rate), which is around $125\,\mathrm{kB/s}$ or $439\,\mathrm{MB/h}$. However, this makes tracking difficult and will produce poor quality navigation solutions. A more practical sample rate is around $5\,\mathrm{MHz}$, which results in a data rate of around $610\,\mathrm{kB/s}$ or $2\,\mathrm{GB/h}$. In addition to this, post-processing cannot produce any information about the receiver's position until after the fact. These two main disadvantages, limit the use of post-processing to laboratory use and niche, short duration applications.

There is another point, in a typical GNSS receiver, where the data can be stored, which has currently not been investigated - where the data leaves the tracking channels (see figure 3.2). This, like post-processing, has certain advantages and disadvantages - namely the ability to use the IGS ephemerides and the receiver not knowing its position - whilst

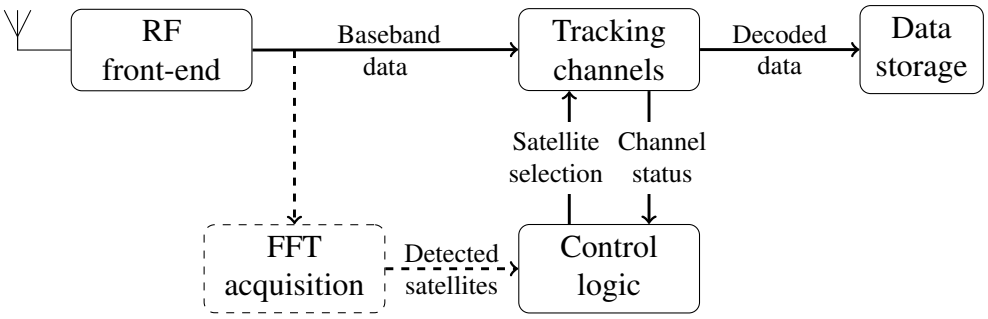


Figure 3.2: An alternative post-processing architecture, where the decoded data, rather than the baseband data, is stored.

not requiring the large amount of data storage. In particular, it is possible that it could be beneficial for it to be used in two applications. The first is in a power constrained environment. The navigation processor calculates the position of the receiver by calculating the positions of the satellites, applying corrections and solving at least 4 simultaneous equations (which is typically done through matrix inversion). This, in particular the last step, is quite computationally complex and can consume a significant amount of power. If the decoded data stream, from the tracking channels, is stored, there is no need for a navigation processor to be part of the receiver's design, reducing the receiver's power consumption. The second application is a hybrid between a typical receiver and one that stores the decoded data. This would allow a receiver to be able to know its position in real-time, like a typical receiver, whilst storing the decoded data to be used later with the more accurate ephemerides, which could be useful in a variety of different applications.

For a power constrained design, such as the PCBsat, the first application is very attractive and could be integrated into the mission's design. In particular, it could be used in a distributed receiver design where the receivers on the PCBsat are designed without a navigation processor and transmit their collected scientific data with the unprocessed position data (see figure 3.3). Then the receiving cubesats, which provide a ground-link for the PCBsats, process the receiver's data to produce a navigation solution before transmitting the scientific data to the ground station. This, from the point of view of the ground

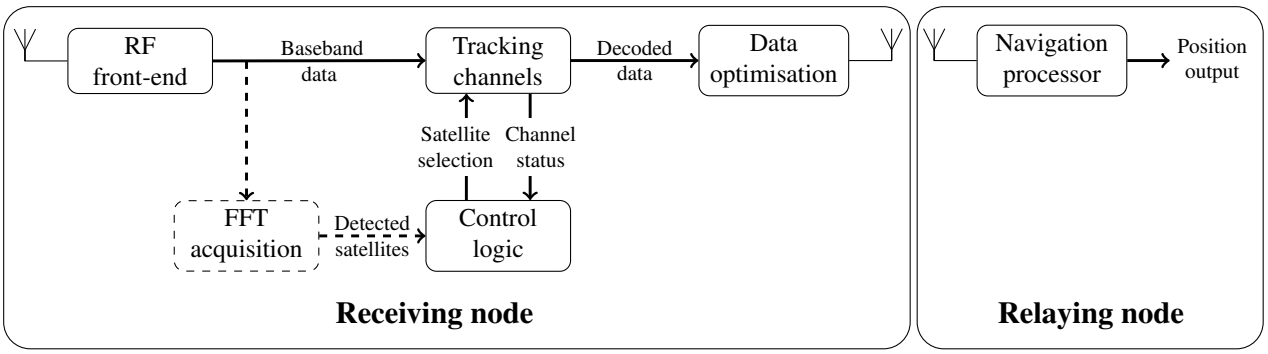


Figure 3.3: Architecture of a distributed GNSS receiver, with the navigation processor being in a different location to the GNSS antenna.

station, would be as if the PCBsats were calculating and transmitting their own positions, like they were using a typical GNSS receiver; whilst reducing the power consumption of the PCBsats (by placing the computational burden on the cubesats, which have a more accommodating power budget). However, as the PCBsats have to transmit more data, there is a trade-off between the power saving made by the receiver and the increase in power of the communications subsystem. To determine if this design could reduce the power consumption of the PCBsats, a feasibility study was conducted.

3.2 Distributed receiver feasibility study

Whether the distributed receiver design reduces the power consumption depends on the power consumption of the navigation processor and the power consumption of the radio transmitter. Both of these will be quantified as the amount of energy required per position, as they are both discrete processes.

The power consumption of the navigation processor was estimated by mimicking the main computational burden - the matrix inversion - as a full navigation processor, at the time of the feasibility study, was not available. There are many ways to do this and singular value decomposition (SVD) was chosen as it is capable of finding an approximate solution to a singular, or numerically close to singular, matrix (Press et al., 2007). A singular matrix is a matrix that cannot be inverted and occurs, in square matrices, when multiple rows are degenerative - i.e. they contain the same information. Whilst this is not likely to happen in a GNSS receiver - as no two satellites will be in the same position - it is possible for a singular matrix to occur because of conflicting information, for example, caused by differences in the signal's propagation velocity.

A typical receiver will track around 6 satellites at a time - 4 is the minimum required to calculate the receiver's position, with around 8 being the highest number of satellites visible at one time. It is often the case that one or two of the 8 visible have an SNR that is too low for a receiver to acquire and track, so 6 was chosen to cover the typical case. As the positions of the satellites and their pseudoranges was not available, 6 points were chosen at the approximate altitude of GNSS satellites (in particular, GPS satellites) with the pseudoranges being manually calculated from a position on the Earth's surface. Whilst this did not include any propagation delays or errors, and so was idealistic, it was more than adequate for testing the SVD routine and estimating the required energy.

The calculation of the receiver's position can be performed on a wide range of processors, so three likely candidates were chosen, based on the design of the PCBsat. The first was a low performance, 16 bit microcontroller: Texas Instruments' MSP430FR5739[2]. The second and third choices were both 32 bit ARM Cortex-M microcontrollers but

[2]The MSP430FR5739 is very similar to the MSP430FR5738, with the only differences being in the available peripherals and that the '39 is available as part of a development board.

Table 3.1: Summary of the 3 microcontrollers used for the navigation processor estimates.

	Performance		
	Low	Medium	High
Microcontroller	MSP430FR5739[a]	STM32L152[b]	STM32F407[c]
Architecture	16 bit MSP430	32 bit Cortex-M3	32 bit Cortex-M4
FPU	No	No	Yes
Operating frequency (MHz)	24	32	168
DMIPS	24 (estimated)	33.3	210
Maximum power (mW)	12.9	54.3	392.4

[a] Texas Instruments (2011). [b] STMicro Electronics (2013a). [c] STMicro Electronics (2013b).

from different series - the second was a Cortex-M3, designed for low power use: ST's STM32L152; with the third being a Cortex-M4, which is an Cortex-M3 with a floating point unit (FPU) and DSP extensions: ST's STM32F407. These three microcontrollers, summarised in table 3.1, cover the range of processing that a PCBsat, or any femto-satellites, is likely to be capable of. The DMIPS row of table 3.1 is a considerably more useful metric than the operating frequency, as it is a standardised benchmark that shows the performance of the processor.

Most modern processors have low powered sleep modes that stop the processor's clock, which are used instead of continuously running the processor when there is nothing for it to do. This can reduce the power consumption of the processor quite drastically, but, if the processor is only running at full power for small periods of time (i.e. milliseconds), can make measuring the power consumption difficult. An alternative method, which was used, is to measure the power consumption whilst running the processor continuously doing the required task - the matrix inversion - and measure how long it takes to perform each task; then the power consumption of the sleep mode is measured. With the task run time known, the amount of sleep time is also known and so an overall power consumption can be calculated.

When a processor wakes up from its sleeping state, there is a certain amount of time required before it can start processing as normal, which is due to its internal clocks stabilising and its interfaces being re-initialised. This means that this method of calculating the power consumption will be an underestimate, as the processor will spend some time consuming more than its sleep power, which is not considered in the task processing time. However, this amount of time is very small and significantly smaller than the task processing time, resulting in the underestimation being very small. For example, the typical wakeup times for the microcontrollers used are 78 µs, 360 ns and 1 µs (Texas Instruments, 2011; STMicro Electronics, 2013b,a).

The active power consumption for the three microcontrollers, shown in table 3.2, was taken whilst running the SVD task in a loop, with an output on the microcontroller being

Table 3.2: Microcontroller power consumption and task length for the SVD (matrix inversion) task. Both the active and sleep powers, and the task length are averages over 100 measurements.

Microcontroller		Power consumption		
		Low	Medium	High
Active power (mW)		14.82 ± 0.03	44.69 ± 0.03	121.58 ± 0.03
Sleep power (mW)		2.0082 ± 0.0003	16.151 ± 0.002	36.539 ± 0.003
Task length (ms)		434 ± 4	55.2 ± 0.6	10.84 ± 0.03
Maximum task frequency (Hz)		2	18	92
Task energy (mJ)	10 Hz		3.19 ± 0.03	4.58 ± 0.01
	4 Hz		5.61 ± 0.04	10.06 ± 0.03
	2 Hz	6.56 ± 0.07	9.65 ± 0.08	19.19 ± 0.05

toggled after each calculation. The frequency of the output pin was then used to calculate the task length. In table 3.2, the maximum frequency of the task is shown, which is number of navigation solutions that can be calculated per second. The minimum rate of navigation solutions required, due to the changes in the orbit being quite small, is 4 Hz. This results in a distance of approximately 2 km between each position measurement, for a satellite in low Earth orbit. Due to the complexity of the calculation, the lowest performance of the three microcontrollers is not capable of this rate, and so its maximum rate, 2 Hz, is shown in the table. In addition to this, the highest likely rate, 10 Hz, is also shown in the table, which would result in a distance between position measurements of approximately 800 m. The task energies increase when the rate decreases and this is to be expected, as the energy includes when the processor is sleeping between tasks. For example, the energy for the 4 Hz rate consists of the task length at the active power and then the sleep power for 250 ms minus the task length. The energy per task will, therefore, be smallest when the processor spends no time sleeping, however, this will increase the total power consumption of the processor.

The second part of the feasibility study requires the amount of extra data needed to calculate the navigation solution to be known. Each GNSS uses a different format for the navigation message, so the GPS constellation was chosen to produce these numbers. However, the differences in the data sizes for different GNSS is going to be very small, as they all implement the same level of functionality, to similar accuracy levels.

A GPS navigation message is 1500 bits long and is transmitted in 5 frames, each 300 bits long. Each of these frames is further divided into 10 words of 30 bits. The contents of each word depends on its position in the frame and what frame it is in, but the last 6 bits of each word is used for parity - resulting in 20 % of the navigation message being used for parity. Whilst this is high, it is needed because of the signal transmission method. However, the communication between the nodes in the distributed receiver do not need this level of error checking as, unlike GNSS, the message can be retransmitted

Table 3.3: Size of the data transmission for the distributed receiver design.

Data caching	Number of used satellites	Data required per position (bits)
Yes	4	328
	6	476
	8	624
No	4	2744
	6	4100
	8	5456

if it is incorrectly received. In addition to the reduction in the number of parity bits, the entirety of the 5th frame and the majority of the 4th frame are not needed as they contain the almanac. The almanac is the approximate orbits of the satellites, that are valid for around a month, and are used for predicting which satellites should be visible to a receiver. However, as the receiver on the PCBsats do not know their own location, the almanac is of no use to them. Frame 4 also contains the correction information for the ionospheric model, which is useful for the receiver. But, like the majority of the first 3 frames, which contain the satellite's ephemeris and clock information, this data does not change very frequently - approximately once every 4 hours. It would therefore be quite redundant to send this common data repeatedly alongside each position measurement. This data is also common amongst all receivers tracking the satellite and so only needs to be stored by one of the receivers. In the envisaged mission, the relaying cubesats will also have GNSS receivers on them and, as they will be in a similar location to the PCBsats, it is highly likely that the same satellites will be visible to both the cubesats and PCBsats. Therefore, the receivers on the cubesats could decode and store this common data, without the PCBsats having to transmit it to them, further reducing the amount of data that has to be transmitted. If a cubesat did not already have the ephemeris and clock data for a given satellite, then the PCBsat could transmit it once to the cubesat. However, this would not be strictly necessary, as this data could be supplemented by that provided by the IGS.

This caching approach reduces the required navigation data for each satellite a receiver tracks to the satellite's week number and accuracy, and an issue of data, that is used, with the week number, to uniquely identify the ephemeris and clock data. This is 24 bits per satellite, with the full non-cached data being 628 bits, which is a drastic decrease from the original 1500 bits. In addition to this, the time of reception, relative to the other satellite signals, is required, with a generous estimate of this being 50 bits. This results in 74 bits being required per satellite for the distributed receiver. The total number of bits required for a position measurement is therefore a multiple of this, with an additional 32 bits. These bits are used to indicate which satellites were used by the receiver, so that the

Table 3.4: Comparison of the three radio transceivers considered.

	Digi XBee-PRO ZB[a]	Atmel AT86RF212B[b]	Amber Wireless AMB8636[c]
Frequency	2.4 GHz	868 MHz	868 MHz
Data rate (kbps)	250	20	50
Transmit power (dBm)	10	10	27
Receiver sensitivity (dBm)	-102	-110	-123
Transmitting power consumption (mW)	738	79.5	1650
Approximate estimated range (km)	2.48	10.9	34.3

[a] Digi International Inc. (2014) [b] Atmel Corporation (2014)
[c] AMBER wireless GmbH (2014)

navigation processor can calculate the position correctly[3]. The total number of bits for both the cached and non-cached receiver are shown in table 3.3.

For a non-distributed receiver, the size of the position data is fixed. Whilst most commercial GNSS receivers use a text format for their position output, a binary output is significantly more efficient. Using Cartesian coordinates, rather than longitude, latitude and altitude, single precision floating points - which are 32 bits in size - will give metre-level precision, which is an order of magnitude better than the typical accuracy of a single frequency GNSS receiver. In addition to this, the time and satellites used are needed, both of which are 32 bits long. When calculating a navigation solution, the arrangement of the satellites affects the precision of the solution. There are a common set of values, the dilution of precision, that are used to describe this, with 4 being needed - for the horizontal, vertical, positional and time dilutions. 16 bits each for these is adequate, bringing the total number of bits required for a navigation solution to 224.

Different radios require different amounts of energy to send each bit, depending on the transmission method and the signal power. Other than the network being mesh-based, the radio has not been defined, so three possible radio transceivers are considered, and these are shown in table 3.4. To determine the feasibility of the distributed receiver, we need to find the break-even point - where the non-distributed and distributed receivers are equal in their energy consumption. This is fairly straightforward, with the break-even point being when,

$$E_c = \Delta N E_b,$$

where E_c is the energy required to calculate the navigation solution, E_b is the energy required to transmit a bit of data and ΔN is the difference between the number of bits that the distributed and non-distributed receivers transmit (with the difference being positive for a larger number of distributed bits). These break-even points are shown in figure 3.4 as

[3] With the data always being ordered from lowest satellite number to highest.

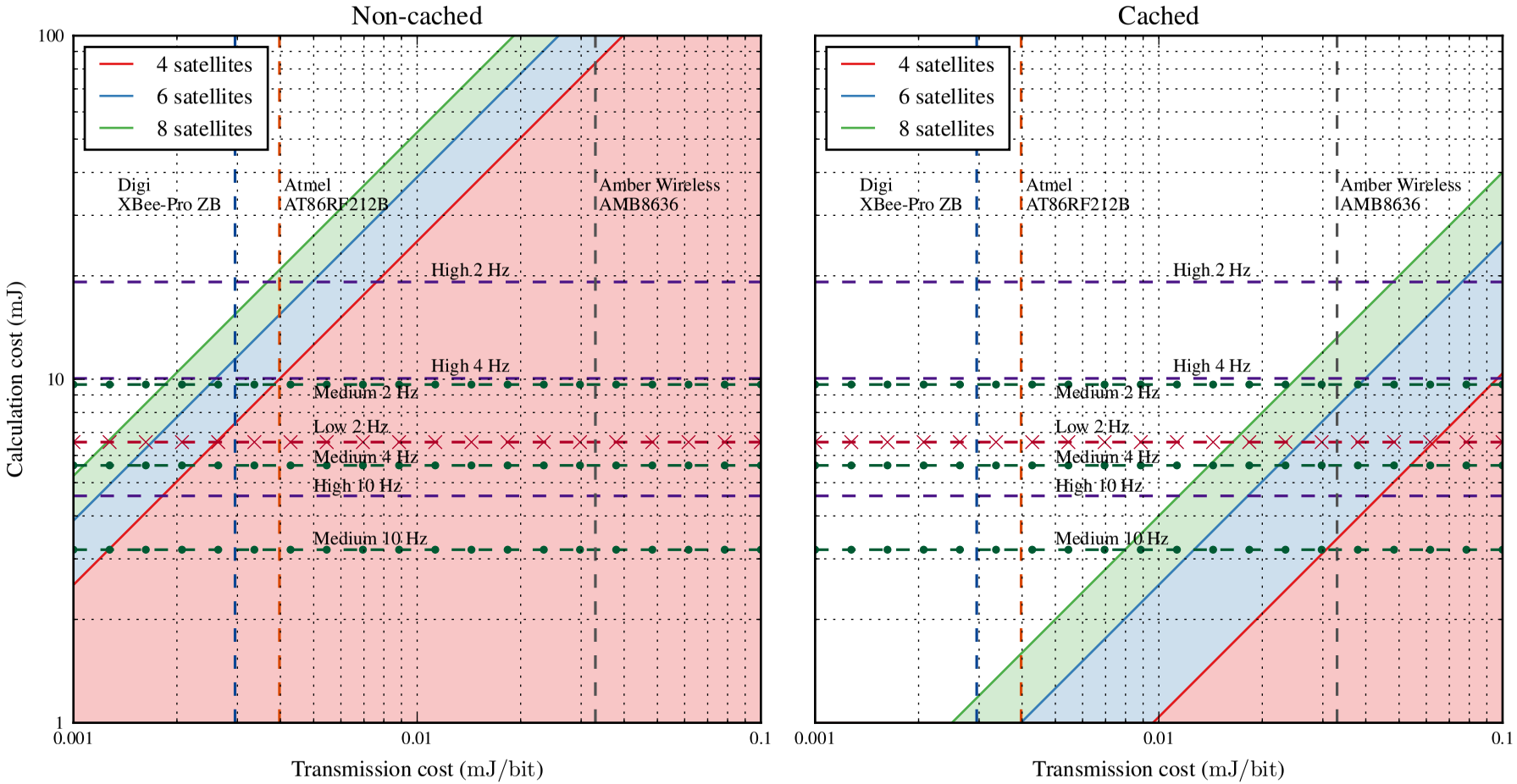


Figure 3.4: Energy comparison of the distributed and non-distributed designs. The solid lines show the break-even points, with the shaded area not being energetically viable. Intersections of the vertical lines (dashed) and the horizontal lines (dot-dashed) indicate whether the combination of a transceiver and a navigation processor pair is viable.

solid lines. Figure 3.4 uses an array of lines - horizontal for the estimates of the navigation processors and vertical for the radio transceivers - with their intersections representing the different configurations. The intersections which occur in the shaded regions (below the solid lines) are where the configuration is not energetically viable - i.e. the distributed design uses more energy than the non-distributed design.

When caching is not used (the left hand side of the figure), only the Digi radio transceiver has viable configurations, however, these are of questionable worth. For example, when used with the high microcontroller at $2\,\mathrm{Hz}$, the distributed design is viable for up to 8 satellites, however, at the same rate it is not viable for the low microcontroller and only for 4 satellites for the medium microcontroller. Whilst there might be some situations where it is not be possible to choose the lower performing microcontrollers, from a pure receiver perspective it would be reckless to choose a less efficient solution to make the distributed design viable. This results in the non-caching version of the distributed receiver almost always being infeasible, due to the large increase in the amount of data that has to be transmitted.

The right hand side of figure 3.4, shows that when data caching is used, with either the Digi or Atmel transceivers, the distributed design reduces the energy consumption. However, with the Amber Wireless radio transceiver, a similar situation to the non-cached case is seen, where the distributed design is feasible for some configurations, whose microcontrollers are less efficient.

In conclusion, the distributed receiver design is feasible if data caching and an appropriate radio transceiver is used. Whilst the transceivers in table 3.4 represent the available range rather than the most optimum, it does suggest that the distributed design is only feasible for node separations under $10.9\,\mathrm{km}$. Despite this being quite small for satellite separations, it fits in well with the size of plasma bubbles and so this limitation is unlikely to negatively impact the intended mission.

3.3 GNSS receiver prototype

There are two options available for the RF front-end - either a commercial IC can be used or a custom solution can be made. The custom solution has certain advantages, the main being that the intermediate frequency and the number of bits the signal is digitised to can be easily chosen. However, it comes at the cost of extra complexity and an increase in power consumption - as one chip would be replaced with many to do the same job. Unfortunately there are not many commercial ICs available that implement only the RF front-end. With the design of a GNSS receiver being quite complex, especially when considering extra features which are frequently used in terrestrial receivers, the vast majority of GNSS receivers have become single chip solutions (i.e. the front-end is integrated into the receiver). In addition to this, the pressure on miniaturisation in the mobile phone in-

dustry has led to many companies integrating their GNSS receivers into system on chips (SoCs) - where the peripheral devices are placed into the same package as the processor - such as Qualcomm Technologies Inc. (2016)[4].

Whilst there have been several commercially available RF front-end ICs in the past, there is currently only one available - Maxim Integrated's MAX2769 (Maxim Integrated, 2010). However, this IC is quite adequate and is capable of being used for the most popular GNSS constellations - GPS, GLONASS and Galileo.

Its digitised output utilises 5 lines - 4 for data and one for the clock. This makes it quite difficult for it to be captured by a microcontroller, as few communication interfaces use 4 bit parallel data and those which do cannot be manipulated so that they only sample the data. The only method of capturing this data would be to use the general purpose I/Os, which would use the clock line to generate interrupts. Whilst this would work, guaranteeing a low and predictable amount of jitter would not be an easy task, and would be made significantly more difficult by the addition of an appropriate interface to store or transfer the data (e.g. flash memory, SD cards, USB or Ethernet); therefore, an FPGA was used.

As the FPGA's requirements were not initially known at the start of the prototyping, an available FPGA development board was used, which had a Xilinx Spartan-3 FPGA. Whilst this FPGA is not particularly suited to DSP, it is more than capable of correctly sampling the front-end's data output. However, the development board lacks any high data rate peripherals that could be used to store or transfer the data, with the best option being a UART/serial interface. The limited block RAM available in the FPGA was used as a FIFO to store $5\,\mathrm{ms}$ of baseband data, which then took approximately $800\,\mathrm{ms}$ to transfer to a computer over a UART link.

In addition to this, due to the RF front-end IC being in a surface mounted package and requiring a non-trivial circuit, the initial prototyping used an RF front-end evaluation board, shown in figure 3.5. This had several drawbacks. The first was that the board required 3 power supplies (3.3, -5 and $5\,\mathrm{V}$) and was controlled, through the use of an add-on board, using a desktop computer. The second was that the board was primarily designed to test and characterise the analogue functionality of the IC, rather than use it for digital output. But, the board could be modified - through the removal of soldered links - so that it could be used for digital output. The third was the location of the digital output pins, which required the use of approximately $20\,\mathrm{cm}$ wires to connect to the FPGA board. This prevented the set up from being taken outside, where the satellite signals have a greater SNR.

This affected the development of the prototype in two ways. First of all, the $5\,\mathrm{ms}$ of data was enough to detect the visible satellites, but was not enough to perform any tracking. With the GPS navigation message being broadcast at $50\,\mathrm{bps}$, $5\,\mathrm{ms}$ of data holds

[4]It should be noted that Qualcomm is one of the market leaders in mobile device SoCs.

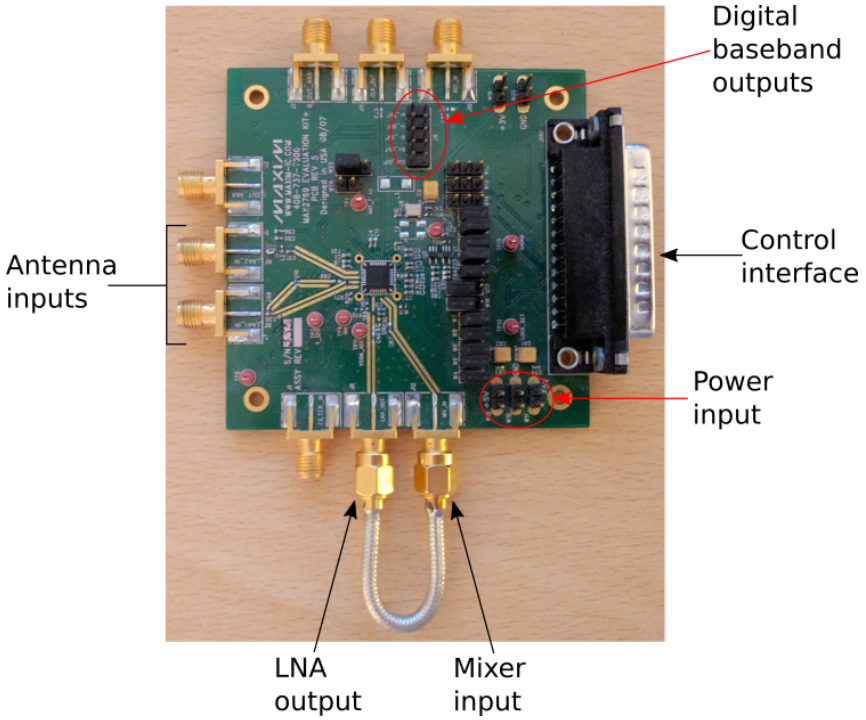


Figure 3.5: The RF front-end (MAX2769) evaluation board. The control interface connects to a computer interfacing board, which is not shown.

approximately a quarter of a bit. Due to the 800 ms gap between captures, the data could not be joined. The second was that the signal was significantly attenuated by the lab's building[5], resulting in only one satellite being typically viewable at a time (both in the captured signal and with commercial receivers). When added to the set up's requirements - i.e. computer for control and receiving of data, multiple power supplies and multiple static sensitive boards connected to each other - it was impractical to move the set up outside, or to a place with a better signal. Overall, the initial set up was not particularly well suited, so external baseband data was sought.

There is very little baseband data that is publicly available, which is most likely due to the large data size and limited application, however, there are some sources. One of the best is the SetiQuest project (which is funded by the SETI institute), who acquire signals from many known sources, including spacecraft and satellites, by using radio telescopes (SetiQuest, 2015). This means that their GPS data consists of a strong signal from a single satellite, with very little signal, if any, from other satellites, as the telescope has a limited field of view and was only directed towards one target. This restricts the use of the data, as a navigation solution cannot be calculated, however, both signal acquisition and tracking are possible.

[5]It should be noted that this was not a feature of this building, most GNSS signals are difficult, if not impossible to receive within most buildings.

3.3.1 Python prototypes

To enable a prototype to be constructed in the shortest possible time, a high level scripting language (Python) was used. This allowed for rapid development of the algorithms for both acquisition and tracking, as the specifics of the low level implementation did not have to be considered, however, it had the disadvantage of sacrificing performance.

As the data available only contained GPS satellites, the FFT acquisition program was designed to only detect GPS signals. FFT based acquisition utilises the orthogonal nature of the PRN codes to determine which satellites are visible, using the process of auto-correlation. This is a significantly faster technique than detecting visible satellites using correlation, which requires that each code delay is individually tried, but has the downside of being less precise.

The program first loads the PRN codes, at the correct sample rate, and then calculates the FFT of each code. The baseband data is then loaded and an FFT of it is also calculated. Rather than the program demodulating the signal in the time domain, which would involve multiplying each sample with a generated sine wave, the modulation is removed in the frequency domain, by rotating the FFT's bins. The number of bins to rotate by is calculated using,

$$r = \left\lfloor (f_c + f_D) \frac{N}{f_s} \right\rfloor,$$

where r is the number of bins to rotate by, f_c and f_D are the carrier and Doppler frequencies, N is the number of samples and f_s is the sample rate. The demodulated signal is then used to produce the correlation power spectrum,

$$A(\omega) = F(\omega) P^*(\omega),$$

where $F(\omega)$ is the demodulated signal and $P(\omega)$ is FFT of the PRN code. The inverse FFT of $A(\omega)$ is then calculated, squared and searched for peaks. This procedure is then repeated for all of the required Doppler shifts, with the highest peak across all Doppler shifts being stored and then outputted. This is then repeated for all of the PRN codes.

```
# calculate FFT of baseband data
f_data=fft.fft(data)
# iterate through PRNs 1 to 32
for prn in range(1, 33):
    # set best values to small numbers
    best_freq=-1e20
    best_peak=-1e20
    best_snr=-1e20
    best_delay=0

    # iterate through frequency shifts
```

```
    for freq in np.arange(doppler_min, doppler_max, freq_step):
        # calculate shift
        shift=int((freq + centre_freq)*num_codes/1e3)
        # shift FFT'd baseband data / demodulate the baseband data
        data_s=np.append(f_data[shift:], f_data[:shift])

        # calculate inverse FFT of shifted baseband data multiplied
            with the FFT conjugate of the PRN code
        result=fft.ifft(data_s*f_ca_conj[prn])
        # take the absolute value and square it
        result=np.square(np.abs(result))

        # find the peak value
        t_peak=np.max(result)
        # calculate SNR metric - peak signal over average
        t_snr=t_peak/np.mean(result)

        # if peak value is large than current best value
        if t_peak>best_peak:
            # store it as the best value with the relevant information
            best_peak=t_peak
            best_snr=t_snr
            best_freq=freq
            # this line finds the array index of where the peak value
                is
            best_delay=np.where(result==t_peak)[0][0]
    # output the best peak found
    print(prn, best_freq, best_peak, best_snr)
```

Listing 3.1: Example of Python based FFT acquisition.

The main part of the program is shown in listing 3.1, where the PRN and data loading has been omitted for brevity. This particularly shows off the advantage of using Python, as the produced code is remarkably close to pseudocode, or the most idealistic way of writing it.

With this approach, the smallest change in the Doppler frequency is the width of a frequency bin (the resolution of the FFT). It could be argued that a greater Doppler resolution could be achieved through demodulating the signal in the time domain, however, the limitation of the FFT's resolution will still be the limiting factor, resulting in a limited, at best, increase in the resolution.

The tracking program processes the data in a loop, with the processing initially being simplified. The frequency / carrier tracking was performed with a Costas PLL, with a very simple loop filter, with code removal being achieved through a matched filter. Whilst a matched filter does the job very well, it is not compatible with calculating a navigation solution, as it does not produce a code delay. However, it did allow for different error

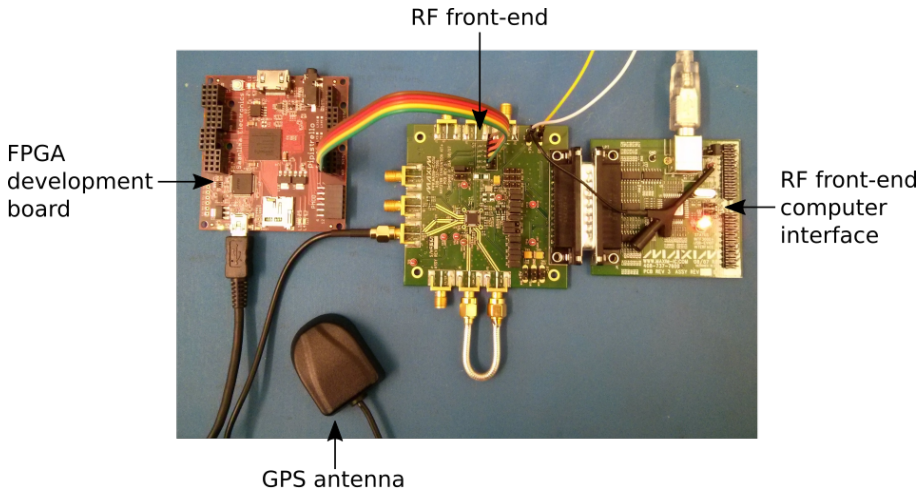


Figure 3.6: Spartan 6 FPGA development board connected to the RF front-end evaluation board.

discriminators to be tested in the Costas PLL, without having to be concerned with the code tracking. As the SNR in the SetiQuest data is very high, that is significantly higher than that normally seen by a GNSS receiver, the tracking of the signal is not very difficult. This is advantageous, as it allows a sub-optimal implementation to correctly track and decode the satellite's signal, however, it is a little unrealistic.

Shortly after this initial prototype was completed, an FPGA development board that better fitted the requirements was acquired. This utilised a Spartan 6 FPGA (LX45), a USB FIFO interface and enough I/O pins (available through connectors) to meet the expected requirements. The FPGA has a few advantages over the Spartan 3, such as lower power, higher logic density and efficiency, and more block RAM, but most importantly has DSP slices (Xilinx Inc., 2011b). These are hard logic elements that implement a 36 bit (18×18 bit) multiplier, with a pre- and post-adder, that are particularly useful for DSP designs (as multipliers consume a large amount of logic) (Xilinx Inc., 2014).

The FPGA's acquisition logic was re-written to use the USB FIFO interface, which allowed the samples to be both captured and transferred in real-time. However, two additionally problems occurred - latency and interference. Whilst the USB FIFO could transfer the data to the computer at the required bandwidth, the USB architecture means that the host (the computer) has to poll the device (the USB FIFO) to see if it is requesting to send data and then the hosts initiates the transfer (Axelson, 2015). This results in a certain amount of latency that is dependant on the software running on the computer, which, due to the multitasking inherent in modern computer design, meant that some samples were occasionally lost due to the FPGA's buffer overflowing. This happened in an unpredictable way and could not be mitigated by modifying the software running on the computer (including memory mapped IO, process priorities and processor shielding - limiting a processor, in a multiprocessor system, to only one task or process). In addition to this, it became apparent that the digital interface, between the RF front-end and the

FPGA, was being affected by interference. The RF front-end was connected to the FPGA using $2.54\,\mathrm{mm}$ wire jumpers, which were approximately $20\,\mathrm{cm}$ in length, see figure 3.6. Whilst this type of parallel cable should be more than capable of carrying signals at the baseband's sampling rate, $16.368\,\mathrm{MHz}$, movement of the cable or near the cable, resulted in significant changes in the signal's power spectral density. The exact cause of this is not known, it could have been due to the changes made to the RF front-end board or, more likely, due to the way the front-end produced its output. The RF front-end's output is not at a single frequency, but instead is produced at two frequencies, one above the sample rate and one below (namely, $16.67\,\mathrm{MHz}$ and $15.80\,\mathrm{MHz}$), with the number at each averaging to the sampling rate. This adds additional frequency components to the signals, that could have affected the signal, through reflection or coupling. Attempts were made to reduce the interference, but these were not particularly successful, with the fundamental problem being the use of cabling to interface the two boards.

With these problems being combined with the practical difficulties of taking the set up outside, mainly computing and power, it became obvious that additional tools would have to be developed.

Chapter 4

Required tools

The required tools, to allow the development of the GNSS receiver, that are discussed in this chapter are:

Data storage A form of data storage that can be used to store the RF baseband data, that can be easily interfaced to by hardware.

Baseband recorder A tool that uses the data storage to record the digital baseband being produced by the RF front-end.

Baseband reader Similar to the baseband recorder, but instead designed to playback the baseband data as if it was being produced by the RF front-end.

Custom RF front-end board A custom designed PCB for the RF front-end which connects directly to the FPGA development board, to reduce any interference and to remove the requirement of any undesirable external power supplies.

GNSS signal simulator A GNSS specific signal generator used to test the receiver design with a known input.

4.1 Data storage

To be able to record the RF baseband data, an interface with a fairly fast data rate (around $8.2\,\mathrm{MB/s}$) and a low latency is needed. The data rate is that of the RF front-end and the small latency is required due to the limited buffering that can be performed in the FPGA (due to the available RAM).

The fastest data rate of the USB interface on the FPGA board (an FTDI FT2232H) is approximately $10\,\mathrm{MB/s}$. However, the latency, combined with the interface's small internal buffers, is large enough to cause the buffer in the FPGA to overflow. It is, additionally, unfavourable to require the baseband recorder to be connected to a computer

during data collection, as it limits the flexibility of the data collection. Therefore, an alternative, non-volatile storage solution was sought.

With the high data rate, the size of the storage medium has to be quite large. For example, 10 minutes of baseband data requires approximately $4.6\,\mathrm{GB}$ of storage. Whilst there are many different types of non-volatile storage mediums, many have complex interfaces or are only available in capacities that are considerably smaller than required. For example, hard drives, either SATA or PATA, use a fairly complex command system, whilst also having less than desirable physical interfaces - high speed differential signalling for SATA and a large number of data and control lines for PATA. USB hard drives, whilst still using differential signalling, operate at a lower speed making them easier to physically interface with. However, the USB protocol consists of many layers and is quite complex to implement (Axelson, 2015). Many microcontroller manufacturers (particularly for ARM Cortex-M based microcontrollers), offer USB software stacks to aid in the interfacing of their products with USB devices. Whilst a microcontroller could be used, there are many drawbacks and potential issues in doing so. For example, the microcontroller is likely to be a bottleneck in the system, as it would have to buffer the data internally. With microcontrollers only having small amounts of RAM and the software stacks being designed primarily for functionality, rather than performance, it is likely that there would be latency issues due to the high data rate required. There are also issues with sampling the baseband data, with the primary concern being that it is difficult to ensure that a microcontroller samples an input in a deterministic way - that is, without any jitter. This is especially the case when a large software stack has to be used, as there is a non-trivial amount of processing occupying the processor, and any jitter would be exacerbated by the high data throughput. There is also an issue that the baseband data has to be captured fairly precisely on the data clock edge as the data lines are not held constant by the RF front-end. This is something that is difficult to guarantee on a microcontroller due to jitter, but could be solved by adding external logic.

At the opposite end of the spectrum, there are commercially available flash memories that use serial interfaces, typically SPI or I^2C. Whilst they are considerably easier to interface with and control, the largest typical size is around 1 to $4\,\mathrm{Gb}$ (which is 128 to $512\,\mathrm{MB}$) and their write speeds are lower than required. As an example, a Micron $4\,\mathrm{Gb}$ serial NAND flash memory has a typical page write speed of $220\,\mu\mathrm{s}$ (around $9.3\,\mathrm{MB/s}$) (Micron Technology Inc., 2016). But as the writes can not be pipelined, this has to be combined with the time it takes to transfer the data, which, at $50\,\mathrm{MHz}$ (the fastest the memory supports), takes around $42.4\,\mu\mathrm{s}$, resulting in an overall typical write speed of around $7.8\,\mathrm{MB/s}$, below that required.

An alternative option would be to use parallel flash memory, which have faster write speeds and larger capacities - currently up to $2\,\mathrm{Tb}$ - but they also have more complicated interfaces. In addition to this, write-levelling techniques have to be used with flash

memory, due to the limited number of times flash memory can be written, and have the disadvantage, alongside serial flash memory, that they require either the same hardware, or additional hardware, to be able to transfer the data to a computer. These are, for the design of a fairly simple data recorder, unnecessary complications. One particularly nice solution is to use SD cards, which are essentially a proprietary serial interface coupled to parallel flash memory. They also have the advantages of being available in a range of sizes, are fairly simple to interface to, and, as they are the *de-facto* standard for digital image storage, have adapters for computers that are readily available.

4.2 SD card controller

SD cards, including micro and mini SD cards[1], utilise a serial bus that can be operated in two modes - as a standard SPI bus and as a proprietary SD bus.

The SPI bus consists of a clock, a chip select and two data lines - one for each device's data output (master to slave and slave to master). The proprietary SD bus consists of a clock, a bi-directional command line and 4 data lines - either a single data line or all 4 can be used depending on the card's configuration.

There are several main differences between the two buses. First of all the SPI bus is full-duplex and the SD bus is half-duplex - this, however, makes very little difference to how the card is controlled, as the protocol is half-duplex / command and response based. Secondly, the SD bus uses CRCs to ensure that both commands and data are transferred correctly, whereas the SPI bus does not[2]. This inevitably makes the SD bus more complicated to implement. And thirdly, the card responds to commands in a different fashion. In SPI bus mode, the responses from the card are transferred at the request of the master, with any errors being directly reported. In the SD bus mode, the card sends a response within a set time limit, with errors being indicated by the card not responding to the command and a flag being set in the card's status register. This further complicates the implementation of the SD bus, as the master has to detect these time-out conditions.

Overall, an SD bus master is more complicated than a SPI bus master. However, there are certain advantages of the SD bus. For example, the SD bus supports a higher clock frequency and additionally, with the 4 data lines, allows a nibble to be transferred on each clock, resulting in an increased data rate.

Whilst there are a small number of SD card interfaces available, none of them were suitable for the application. For example, Edvardsson (2016) and Czerski and Clayton (2013) are designed to use the Wishbone bus and are primarily meant to be controlled by a processor, with the processor instructing the controller to send the requested commands

[1] Which are just smaller physical formats of SD cards.

[2] Technically, the first command issued over the SPI bus enables the SPI mode of the card and disables the card's CRC checking. But as the first command is always the same, the CRC for it is constant.

and data. To use these, would require the use of a soft-core processor and more than likely interrupts. Whilst this does add complexity to the design, it would also add a non-deterministic latency. Instead a design was sought where the initialisation and both the commands and their responses were hidden from the interface of the controller. Presenting a simpler interface, consisting of an address, a read/write flag and a data buffer, whilst complicating the controller's internal logic, simplifies the commanding logic. Allowing the controller to be fully tested and then used as a module, requiring less time to test the main design, which would otherwise have to reimplement the same logic - such as the card initialisation. Unfortunately, the only available open source SD card controller that fits this description also used the Wishbone interface and had a maximum card size of $2\,\mathrm{GB}$ (Fielding, 2014). Whilst it might have been possible to extend the design and remove the Wishbone requirement, the amount of time it would require to become familiar with the controller's structure and design, and then make the changes, would likely be similar to that of writing a controller from scratch. In addition to the open source controllers already mentioned, there are a number of commercial SD card interfaces available. However, like the open source designs, the majority of these are designed to be used with processors, with some offering DMA functionality, and so would require a soft-core processor. However, the overriding factor of commercial interfaces was the cost, with all of them being too expensive to consider.

Due to the lower complexity, a SPI bus based SD card controller was designed. However, whilst testing the controller with physical SD cards, it became apparent that it would not be suitable for the data recording application. Whilst the SPI bus was theoretically capable of the data rates required, it appeared that the SD card was the bottleneck, limiting the data rate by taking a longer than anticipated time to write the data, once the data had been transferred across the bus. Although not documented publicly, it seems that an SD card operating in SPI mode will significantly limit its performance. The initial tests used a class 2 microSD card, where the effective write speed was limited to approximately $120\,\mathrm{kB/s}$. This was considerably below the class 2's minimum serial write speed[3] of $2\,\mathrm{MB/s}$. Following this, other microSD cards were tested, from a range of manufacturers and a range of higher class ratings, but all showed similar write speeds. This, is therefore not an issue with a particular make or model, but a systematic change in the functionality of many, if not all, SD cards. Unfortunately, due to this not being publicly documented, it is only possible to speculate on why the cards impose this restriction. Possible reasons include backwards compatibility, in terms of power consumption, with the preceding MMC cards (which only operated in SPI mode); and potentially a more commercial reason. The SD card specification is closed and licensed, however, since 2007 a simplified version of the specification has been available (SD Association, 2007). It is possible that the performance is limited when using the SPI bus to prevent manufacturers from using the

[3]The SD card class system refers to the serial write speed in $\mathrm{MB/s}$.

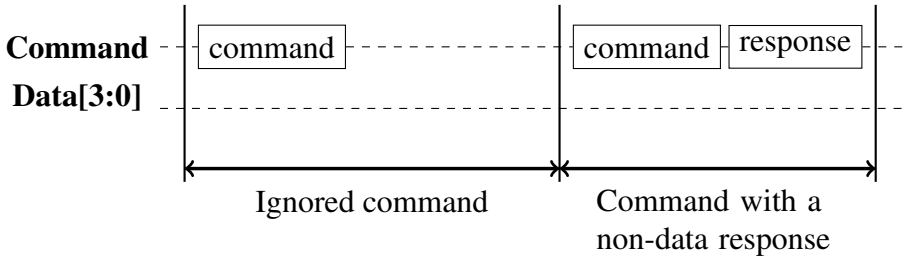


Figure 4.1: Diagram of the two ways that an SD card can respond to a command.

legacy interface for new products, that will likely require higher data throughput, forcing them to use the licensed interface. Of course, this is just speculation, but it would make commercial sense for such a restriction to be enforced.

With the data rate being significantly less than expected, the only option was to change the SPI bus based SD card controller to use the full SD bus, which is described below. The structure of both the SPI and SD bus versions of the SD controller are remarkably similar - they consist largely of a single finite state machine, that performs the card initialisation and the subsequent read and write commands, a 512 byte RAM interface, that is used to hold the current transaction's data, and the bus interface (including timing generator). It would therefore be superfluous to describe both of these, so, as the SD bus version is the one that was used, it is the version that is detailed below.

4.2.1 SD card bus

The SD card bus is an amalgamation of topologies - during normal operation it is a single master (the controller), multiple slave (the SD cards) topology, with each individual card being referred to by an address. However, the cards are assigned their address during initialisation and so the specification requires that each card has its own command line whilst initialising. There are advantages to this bus topology, namely that the same data can be written to multiple SD cards concurrently, however, the requirement of independent command lines during initialisation essentially reduces the topology to a single master, single slave bus. It is because of this, that the SD card bus is most frequently used in a single master, single slave topology.

The bus consists of a clock, generated by the controller, a command line and 4 data lines. Commands are issued by the controller on the command line and, if the command is successfully recognised by the card, a response is returned on the command line. If the command is not recognised, either because it fails CRC checking, the card does not understand the command or the card ignores the command (for example if the card is in the wrong state for the command), no response will be issued by the card. Instead, a flag is set in the card's status register. Figure 4.1 shows a high-level diagram of how a card can respond to a command. As the card has the ability to not respond to a command, the card

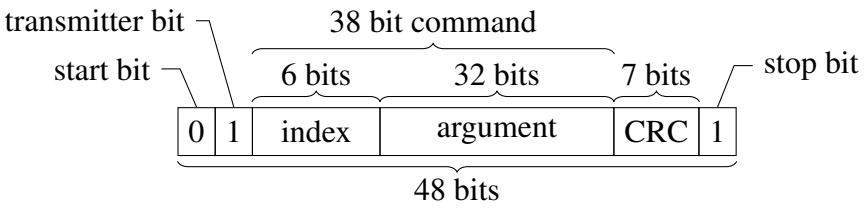
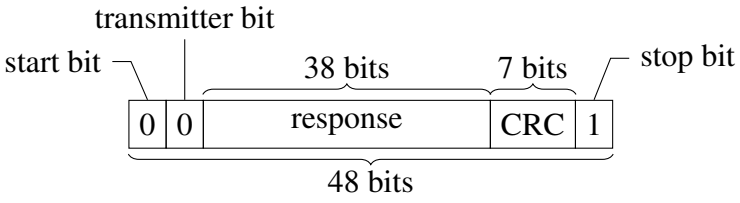


Figure 4.2: SD bus command format.

R1, R3 & R6 response



R2 response

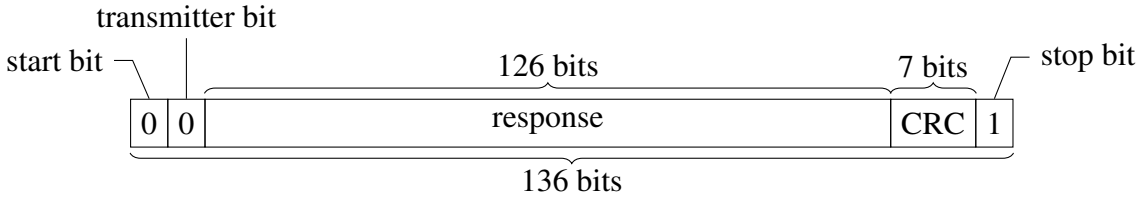


Figure 4.3: SD bus response formats.

has to produce a response within a set time limit. During the initialisation, this time limit is set to 1 second (SD Association, 2007), however, no figure is given for the time limit during normal operation. Tests have shown that a time limit value of $10\,\mathrm{ms}$ is appropriate, as most cards respond in less than $1\,\mathrm{ms}$.

All SD commands have a fixed size and format - they are 48 bits long and contain a 7 bit CRC. The first and last bits are start and stop bits, removing the command line from its idle state and returning it, respectively. The second bit is referred to as the transmitter bit and indicates if the transfer is a command or response, for commands this is 1. The remaining 38 bits are split into a command index and an argument, as shown in figure 4.2. The command's index is a big-endian (most significant bit first) encoded integer, which specifies the command number. The content of the command's argument varies between commands, but when a command does not require an argument it is filled with stuff bits so that the command is always 48 bits long. The 7 bit CRC is calculated on the first 5 bytes of the command - that is from the start bit to the last bit of the argument, inclusively.

An SD card can respond in two main formats - a 48 bit response and a 136 bit response. All of these have the transmitter bit set to 0, to indicate that they are responses from the card. The 48 bit response is used in three different forms, however, they all share the same overall format as the command format (figure 4.2). In the R1 and R6 formats, the

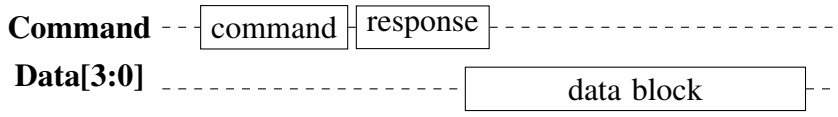


Figure 4.4: Diagram of a single data transaction, following a successful command and response.

command's argument is replaced with the card's status and a combined card address and status, respectively. The R3 format is used for returning the card's operating conditions register to the controller and is only used during initialisation, it has the equivalent of the command's index and CRC set high (i.e. each bit is set to 1). The remaining response, R2, is used for the card's identification and specific data registers, both of which are 126 bits long[4]. The equivalent of the command's index - the 6 bits following the transmitter bit - is set high, like in the R3 format, with the equivalent of the command's argument being extended to 120 bits. The 7 bit CRC is classed as part of the registers and is only calculated for the 120 bits, which are equivalent to the command's argument.

Data is transferred on the bus after a correctly received, and understood, data command, with the delay between the command's response and the start of the data being variable. When the controller is sending the data, no time limit for the delay is enforced by the card[5]. However, when the controller is reading the data, the card responds within 100 times its typical read access time[6] or $100\,\mathrm{ms}$, whichever is smaller (SD Association, 2007).

The SD bus has four data lines which, as shown in figure 4.5, can be used in two different ways - either as a wide bus, where all four lines are utilised, or as a narrow bus, where only the first line is used, with the others remaining idle (i.e. high). Like both the commands and responses, the start and end of a data transfers are indicated by a start and stop bit, respectively. The data transfer is also protected by a 16 bit CRC, opposed to the command and response's 7 bit CRC, and is calculated on a line basis - this means that in the wide bus mode, the controller has to calculate 4 separate checksums in parallel. The data is transferred across the bus most significant bit first (i.e. big-endian), which, when using the narrow bus, is done sequentially. In the wide bus mode, a nibble is sent at a time across the four data lines (as shown in figure 4.5), which quadruples the bus' throughput for a given clock speed. The number of bytes in a transfer is determined by the card's settings, but, for the vast majority of uses, it is set to 512.

When the controller writes data to the card, the transfer of each data block is followed by a status response from the card, indicating if the block was accepted. Figure 4.6 shows the three responses: a success response and the two possible error responses, for either

[4]Technically, these registers are 127 bits long, with the least significant bit being indefinitely reserved. It is this bit that is omitted when the card responds with the register.

[5]As the card does not impose any time limit functionality.

[6]Which is available via the card's specific data register.

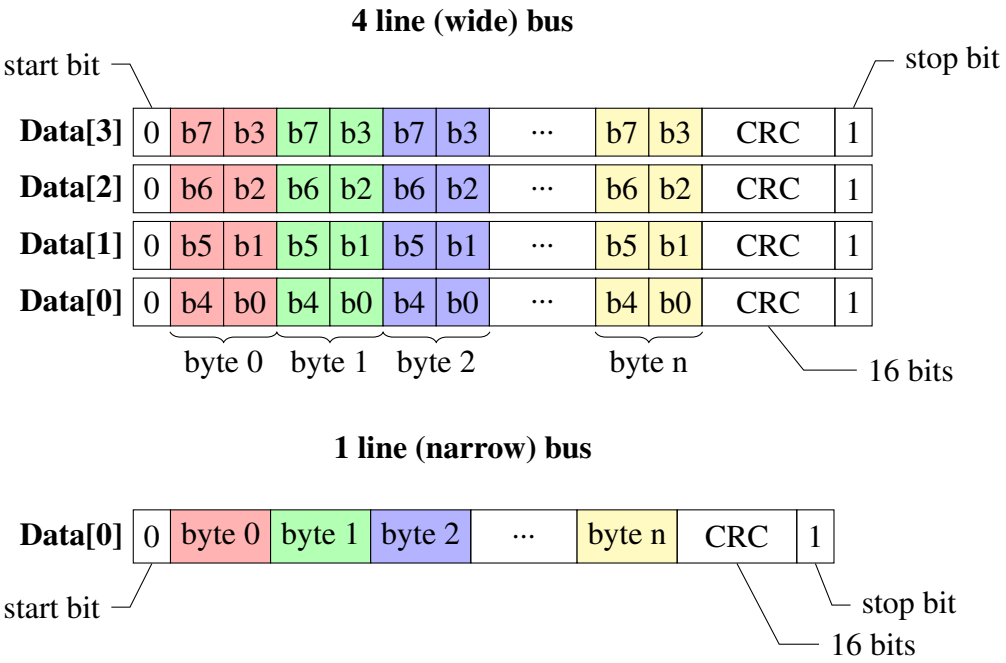


Figure 4.5: SD bus data formats for the two bus width modes.

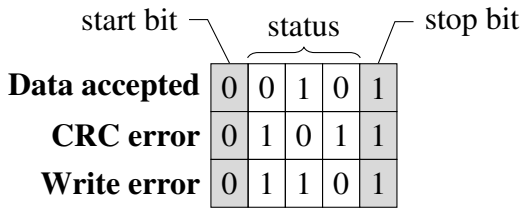


Figure 4.6: SD bus data write responses.

a CRC or write error. After the stop bit of the write response, the SD card will hold the first data line (`Data[0]`) low whilst the card is performing the write, preventing further transfers, and will return the line to its idle state when the card is ready to accept new data. This means that the write error response is largely misnamed, as when the card sends this response it does not know if the write was successful. Instead, this should be thought more of a mis-configured write - such as writing past the end of the card's memory space or to an un-writeable region. The true status of a write can only be found by checking the card's status register once the data lines have returned to their idle state.

In addition to single data transactions, as shown in figure 4.4, there are multiple data transactions, which come in two varieties - counted and uncounted. An uncounted multiple data transaction consists of multiple data blocks, separated by a variable spacing, until either an error occurs (e.g. reaching the end of the card) or a stop command is issued. A counted multiple data transaction consists of a set number of data blocks, with the number of blocks being sent in a command prior to the data transfer command. It is worth noting that the number set in a counted transaction is the maximum number of data blocks and the controller can stop the transaction, in the same way as it would for an uncounted transaction, prior to this number being reached[7]. The data responses, described above, are particularly useful for multiple block writes, as they allow any errors to be detected sooner, and in a simpler manner, than otherwise possible.

When write commands are issued, the card will wait indefinitely for the data block. However, when read commands are issued, the card will respond as fast as it can. When reading a single block of data, this is unlikely to cause a problem, but when reading multiple blocks it is possible for the card to produce data at a higher rate than the controller can handle - for example, if the controller is also processing the data. In this case, it is possible to stop the bus' clock signal, essentially pausing the current transactions. Whilst the clock can be stopped at any point, it is recommended that the clock is stopped in between transfers (i.e. after a response or data block) and that the controller issues 8 clocks prior to stopping the clock.

4.2.2 Design

The SD card controller is designed to be simple to interface to, consisting of a 512 bit RAM interface, used for the data transfers, and a small number of input signals and status outputs, such as enable, read/write, start address, block count and data continue, and idle, error and card version information.

The design of the SD card controller is divided into several sub-modules, which are shown in figure 4.7. The three simplest modules are the clock strobe, the clock controller

[7]The main advantage of counted transactions is that the SD card is aware of, and can optimise for, the amount of data that is being sent or requested. This improves performance as the card can perform erases and data buffering on the total transaction size, rather than on a block by block basis.

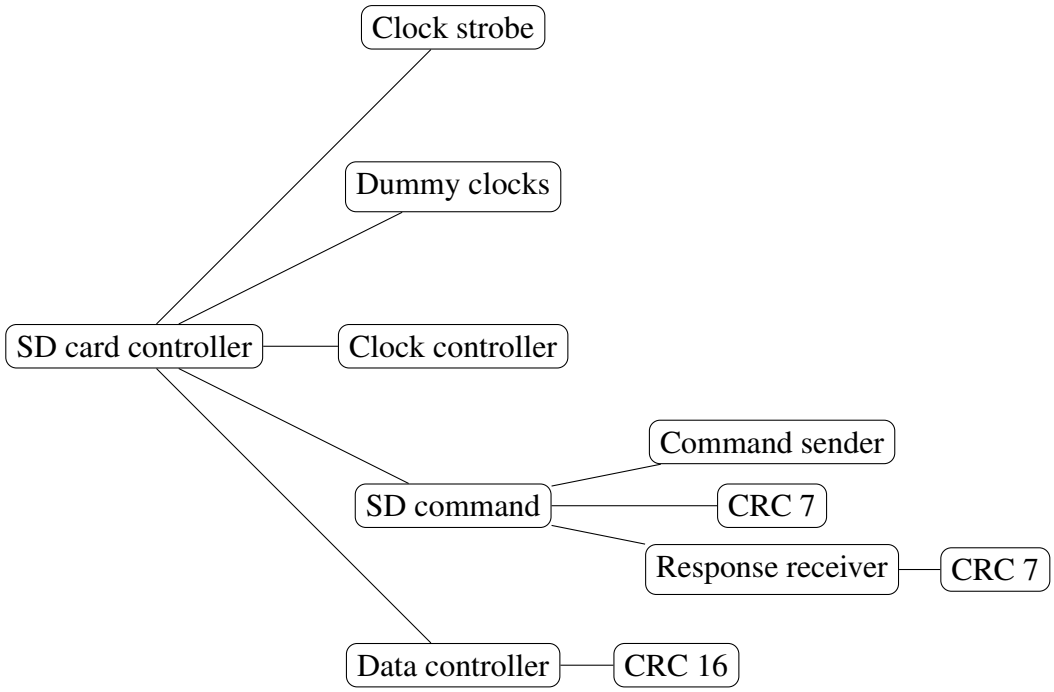


Figure 4.7: SD card controller sub-module hierarchy.

and the dummy clocks. The clock strobe, when enabled, pulses an output signal (for one clock cycle) at a specified divided rate. This is used by the clock controller to generate the bus' clock and as a gating signal for many parts of the controller, rather than controller running at a lower clock frequency. In addition to producing the bus' clock, the clock controller also ensures that the required 8 clocks are issued before the bus is stopped. The dummy clocks module exists for a very special purpose - prior to starting the initialisation procedure, the SD card standard requires that a set number of clocks are issued to the card. These are used so that the card can pre-initialise itself. Whilst this could be done by the controller issuing a command, where all of the bits are high, it adds extra complexity, as both the start bit and CRC generation need to be modified to ensure all bits are high, or the command output needs to be multiplexed, resulting in logic that would not produce an output in this circumstance. Instead, the dummy clocks module issues a set number of clocks whilst holding both the command and data lines high. This module is then disabled and the multiplexer is switched to the relevant other modules for initialisation and normal operation.

The SD command module is responsible for issuing commands and controlling the reception of the response, it does this by using two main modules - the command sender and the response receiver. The command sender writes a byte at a time to the command line of the bus. It does this, as opposed to accepting and sending the full 6 byte command at a time, so that the SD command module can calculate the CRC of the command on-the-fly, allowing a smaller time between the command being issued and the command being seen on the bus. Whilst there are alternative methods of implementing this, so that the

entire command is written instead of each byte, this method is the simplest. The response receiver consists of a 128 bit register and a time out counter. When the module is enabled, it will count the number of rising edges of the bus clock whilst waiting for the response's start bit. If this counter exceeds the module's time out input, then the module raises an error and returns to its idle state. If the start bit is detected before this, then the response is shifted into the 128 bit register. The length of the response is determined by the module's input and changes the number of times new data from the command line is shifted into the register. If CRC checking is enabled, the CRC sub-module is enabled after each byte has been shifted in.

The data controller is the logic that connects the data lines of the SD bus with the controller's RAM interface and, as performance was a major concern, only supports the wide bus mode. This means that the data is transferred over the bus a nibble at a time. The design of the data controller is, therefore, fairly simple. When reading data, it waits for the start bits, then repeatedly combines the nibbles to form bytes, which are then written using the RAM interface. At the end of the data block, the CRC values and the presence of stop bits are checked, with an error being raised if they do not match or are missing, respectively. For writes, the process is largely the same, with the exception of the direction of the data. The only difference is what happens after the stop bits, where the data controller will wait for the write response and, if it is received correctly, until the data lines become idle. As the data controller operates on individual data blocks, it is unaware if the transfer is part of a multiple transfer transaction.

Both of the CRC generators that are used in the controller have wider inputs than are typically expected. For the 7 bit CRC, the generator processes a byte at a time and for the 16 bit CRC generator a two bit input is used. This is done to match the byte based design of the command sending (for the 7 bit CRC) and the nibble to byte conversion that is done for the data lines[8]. This is done to simplify the CRC generation, with the generators being created using the parallel CRC generator by Stavinov (2010).

Initialisation

The initialisation procedure for an SD card consists of many steps, which are, in part, due to the evolution of the standard. In particular, during initialisation the controller has to determine if the card is an MMC or SD card and, if it is an SD card, which version it is and whether it is locked.

As the SD card controller was intended to be used with larger capacity cards, that is SDHC and SDXC cards, the controller was not designed to be used with card versions below 2.0. In addition to this, the controller was not designed to be used with locked cards

[8]When operating in the wide / 4 bit data mode, a nibble is sent at a time (see figure 4.5), but bytes are used both inside the controller and in the RAM interface. So the data controller handles the top nibble (bits 7 to 4) then the bottom nibble (bits 3 to 0), making two bits the best input size for the 16 bit CRC.

(cards that are password protected). In both of these cases, the controller will produce an error. The controller's initialisation process is shown in figure 4.8, with the start occurring after a card has been detected and the end being where the controller enters its idle state - i.e. waiting for a read, write or erase command. The flow chart, for clarity, assumes that a correct response is received for each of the commands. If a response is not received within the time limit, the controller will produce an error. There is an exception to this - command 0 is used to force any SD card on the bus into its idle state. As it is possible to have multiple cards on the same bus, SD cards are designed to not respond to this command. In addition to this, command 8 is only recognised by version 2.0 or higher SD cards, and will also be ignored if the SD card's voltage is out of range. However, as the controller does not support cards with versions below 2.0 and an SD card is not usable outside its voltage window, the lack of a response for command 8 is considered to be an error by the controller.

To maintain backwards compatibility, the initialisation has to be performed at a lower clock speed than what is typically used - $400\,\mathrm{kHz}$ instead of $25\,\mathrm{MHz}$. From the point of view of the card, its initialisation is completed when the clock speed is increased in figure 4.8. However, as the controller can only communicate with a single card, it has to select the card, which places the card in a state where it can accept read, write and erase commands, and it needs to enable the wide bus mode before it can enter its idle state, which is the final part of the controller's initialisation procedure.

Reading and writing

Other than the direction of the data transfer, the process of reading and writing is almost identical. For a single read transaction, the controller issues a read command and checks its response, it then enables the data controller for the receiving of the data. When the data has been received, the controller checks that the data's CRC is a match before returning to its idle state. In the case of a single write transaction, the difference is that the controller does not check the data's CRC, but instead checks that the data has been correctly acknowledged by the card - through the data write response - and then checks that the card has correctly written the data. This is done by waiting for the data lines to become idle and then by checking the card's response to command 13 (send status).

Multiple transactions are done by essentially repeatedly enabling the data controller, with the exception that an additional command is used on the last data transfer to instruct the card to stop sending or receiving data. In between the multiple transfers, the controller stops the bus and waits for the interfacing logic to signal that it should transfer the next block of data and when to end the transaction. To enable better read and write performance from the SD card, a multiple data transaction can be counted (i.e. a fixed length read or write), when this is signalled to the controller by the interfacing logic, it issues a command

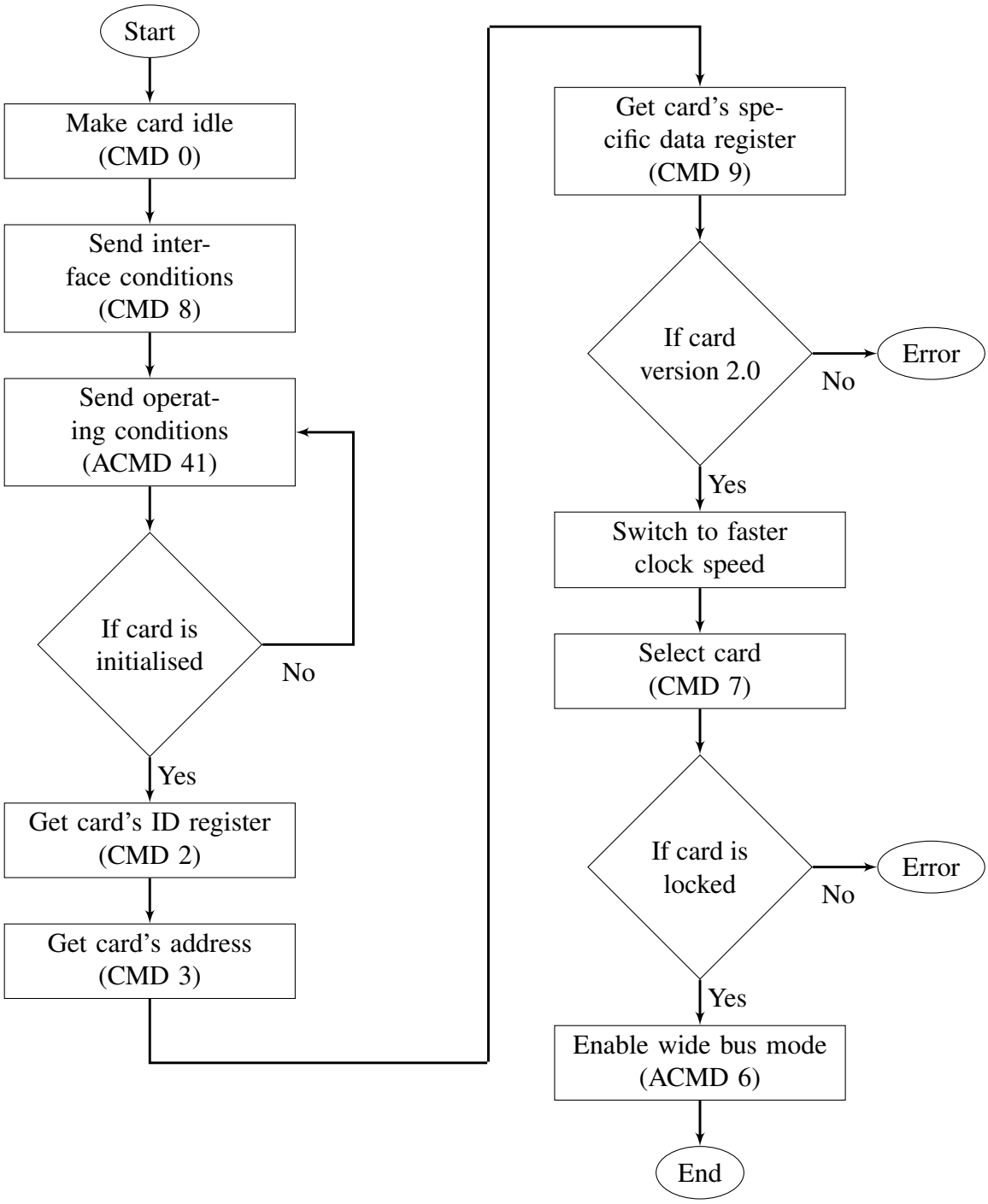


Figure 4.8: SD card controller's initialisation process.

to the card informing it of the length of the data transaction and then the data transaction continues as in the uncounted case.

Erasing

SD cards have a built-in erase command that is similar to the page erase functionality of flash memory. Three commands are required to perform an erase - one to set the start address, one to set the end address and one to perform the erase. When the erase command is issued, the controller, in a similar way to when writing data, has to wait until the data lines become idle before returning to its idle state. As the interfacing logic is primarily designed for reading and writing, where only a start address is required, the controller uses the same continue signal of its interface to separate the start and end addresses. This requires more interaction with the controller's interface, but is not a concern as erases are unlikely to be time sensitive.

4.3 Baseband recorder

The concept of the baseband recorder was quite simple, the baseband data is sampled, stored into the SD card controller's RAM buffer, then written to the SD card. However, there are additional complexities to make the recorder dependency-less and to make it capture the baseband in a reliable way.

The largest dependency of the RF front-end evaluation board was the computer connection required for configuration. The MAX2769 uses an SPI interface for configuration (Maxim Integrated, 2010), so this dependency was removed by adding an SPI master to the baseband recorder, with the configuration settings fixed during the FPGA's synthesis.

The baseband data, despite having a clock signal, was sampled asynchronously by the FPGA. This was done to avoid any clock domain crossing in the FPGA and because the MAX2769's digital output is not particularly well specified. Any metastability issues were mitigated by the baseband's clock and data signals being passed through 2 flip-flops before being used by the FPGA's logic. The data is sampled on the falling edge of the clock signal, as implied by the oscilloscope traces in Maxim Integrated (2010), with the falling edge being detected by the logic looking for a 1 to 0 transition (i.e. the logic detects that a falling edge has occurred rather than detecting the falling edge itself.).

The data buffering was initially implemented using double buffering - where data is placed into one 512 byte RAM whilst the SD controller is reading from another, with a multiplexer being used to determine which RAM is in use by which logic. Whilst this works well, there were consistent buffer overflows that occurred in a periodic manner. It was found that an SD card achieves its minimum writing / class speed on an average over tens of blocks, with occasional blocks requiring significantly longer to write than

others. It is highly likely that these longer block writes are where the card is performing an erase or a page write. For most applications this is irrelevant with the overall speed being important and not the time between successive blocks. However, these longer writes cause the baseband recorder's buffer to overflow.

Increasing the size of the buffer, by utilising block RAM based FIFOs as well as the double buffering, reduced how frequently these buffer overflows occurred, but did not remove them completely, due to the limited total size of block RAM in the FPGA. By utilising the $64\,\mathrm{MB}$ LPDDR RAM on the FPGA development board, as a very large FIFO, the buffer overflow issue was completely removed. The size of the LPDDR RAM was significantly more than what was needed, but the entirety was used as it was available. The LPDDR RAM was controlled via Xilinx's Memory Interface Generator (MIG) which allowed access to the RAM through two $64\,\mathrm{b}$ ports - one for reading and one for writing - each with an 8 deep data and command FIFO. The interfacing logic then implemented a FIFO over the LPDDR RAM. On the write side, the baseband was sampled and buffered into $64\,\mathrm{b}$ blocks which were queued directly into the MIG's interface. On the read side, available blocks were read into the SD card controller's double buffer. Whilst this might sound a little complicated, the double buffering is particularly advantageous as it prevents the SD card controller from waiting mid-transfer for data. Such waiting would significantly slow down the card's write speed.

One particular limitation with the FPGA development board is its lack of user input (i.e. buttons). With only one button available, the control of the baseband recorder's state was achieved through button presses of different durations (0.5, 1.5 and 5 seconds), with the state and the action of the button shown using LEDs. When the baseband recorder is first powered, it enters an idle state. From there the user starts the initialisation, where the recorder performs the SD card initialisation and the configuration of the RF front-end. When this completes, which is less than a second, the flashing rate of the status LED changes to indicate that the recording can begin. The user then starts the recording, which will continue recording until either the SD card becomes full, an error occurs or the user stops the recording (again through the single button). When the recording ends, the recorder enters its final state, where it can only be reset, to prevent the recorded data from being accidentally overwritten. At any point in time, the user can hold the button down for its longest duration (5 seconds) and the recorder will be fully reset.

4.4 Baseband reader

The baseband reader, unlike the recorder, does not need to utilise the LPDDR RAM on the FPGA development board, as the SD card's reading is at a more consistent data rate[9], pro-

[9] This is part of the reason why it is believed that the longer write times are due to the writing mechanism - i.e. either erasing or page writing.

viding that two conditions were met. The first was that the SD card controller performed a counted multiple read transaction, which allows the SD card to optimise its buffering, and the second was that the SD card had been erased and then written once with the data. This second condition is quite curious, as it produced a noticeable difference, and can only really be attributed to write-levelling. SD cards are known to perform write-levelling (ensuring each part of the flash memory is used a similar number of times, so one area does not fail significantly before another) and it is possible that when sequentially reading from an SD card, that has been written to many times, that the data in the card's internal memory is not sequential, resulting in the card having to frequently switch the flash page it is reading from. This frequent switching could negatively impact the read performance quite significantly. When the SD card is erased, it is possible that the write-levelling statistics are reset, or reset to a certain extent, that the data is stored sequentially in memory, allowing for faster and more consistent read speeds.

However, it is not necessarily detrimental if a given block takes longer to read as, unlike in the recorder, no data is lost. Whilst there might be some implications for a GNSS receiver design, it was considered highly likely that the receiver's design would be clocked, or more likely gated, by the baseband data, so any longer read times would not affect the receiver's performance, or indeed be noticeable by the receiver's design.

The baseband reader was designed to be integrated with other logic, rather than being standalone (like the recorder), as there would be very little point in reading the data without logic being present to process it. The reader's interface consists of a start address, the number of blocks to read, an enable input and a loop input, which allows a section of the baseband data to be repeatedly read. There is also an idle and an error output. When removed from reset, the baseband reader initialises the SD card and then issues a read command, requiring that the start address and number of blocks inputs to be set to valid values before removing the module from reset. The enable input will allow playback of the baseband and will pause any playback when the input is low. The idle output is only continuously asserted when the playback has finished (i.e. all of the requested data has been read) and the loop input is low. If looping is enabled, the idle output will pulse for 1 clock cycle when a new loop begins.

4.5 Custom RF front-end board

The RF front-end evaluation board was not ideal in two ways - the use of multiple power supplies and the need to use fairly long wires, considering the size of the board, for connecting the RF front-end to the FPGA development board. This set up was also quite large, with it being difficult to move and the connecting wires were prone to poor connections and becoming disconnected. As the MAX2769's SPI interface does not have a slave to master output (Maxim Integrated, 2010), this had the potential to cause issues

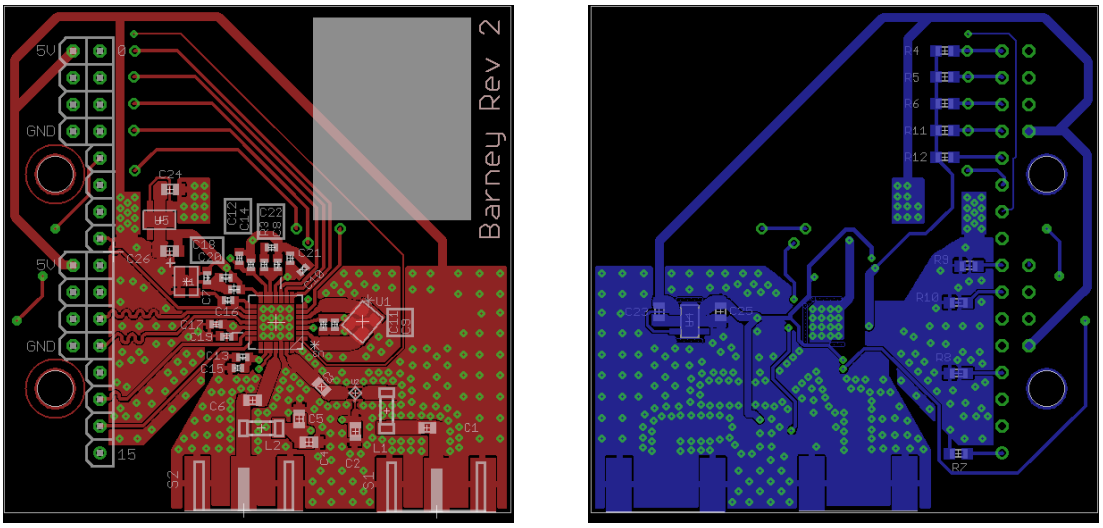
as the RF front-end's configuration could not be checked, with a bad connection causing misconfiguration.

Both of these were solved by the designing of a custom front-end board that connected directly to the FPGA development board, through a pin header. However, as GNSS signals are broadcast at frequencies in the low gigahertz, special attention has to be paid to the PCB design to ensure that it works correctly and that its performance is not degraded.

The FPGA development board provides two power supplies to its main IO connectors, referred to as wings, these are $5\,\mathrm{V}$ and $3.3\,\mathrm{V}$. The $5\,\mathrm{V}$ supply is from an external power supply (provided through a USB connector), with the $3.3\,\mathrm{V}$ supply being from a linear regulator on the FPGA development board. To ensure the best performance, the MAX2769 needs to be supplied with a low noise power supply, as any noise in the power supply will result in noise in the baseband data. So a linear regulator was chosen with a high PSRR and a very low noise output, with two regulators being used - one for the analogue portion of the MAX2769 and one for the digital. Both of these had $2.85\,\mathrm{V}$ outputs, as these are optimum for the MAX2769, and were powered from the $5\,\mathrm{V}$ supply. Whilst this $5\,\mathrm{V}$ supply is noisier, as USB supplies are typical switch-mode based, the $3.3\,\mathrm{V}$ supply does not have enough headroom for a $2.85\,\mathrm{V}$ linear regulator.

Prior to laying out the PCB design, high frequency techniques were researched to find the best way of routing traces. With the small distances between components, impedance matching is not particularly critical, however, sharp corners in the traces should be avoided. Typically, mitred corners are used in PCB design, as right angled corners can suffer from manufacturing defects during the etching process. From using the EM analysis package Sonnet Lite (Sonnet Software Inc., 2013), it became apparent that diagonal traces with shallow angles result in the best signal transfer. Using this, the impedance of traces were matched as best as possible, considering the PCB manufacturer used was targeting low cost, rather than RF applications; and any traces that required angles were routed so that they were as shallow as possible, using components to perform larger changes in direction. Figure 4.9 shows the design of the board, which measures $48 \times 48\,\mathrm{mm}^2$. The board uses a star-based ground to minimise ground loops and an excessive number of vias, to ensure that the ground planes on the top and bottom side of the board were coupled together with the smallest amount of capacitance as possible. To reduce any potential problems with the IC's digital output, the baseband data signals were length matched to within $1\,\mu\mathrm{m}$, including the trace lengths on the FPGA development board. This is why four of the traces going to the IO connector on the top of the board (left hand side of figure 4.9a) are serpentine - i.e. have repetitive 'S' curves.

When assembling the board, it became apparent that the choice of U2 was overly optimistic. U2 is a high quality low noise amplifier that is connected to the secondary input of the IC and, whilst not technically needed, would have allowed for a wider range of testing. U2 is only available in a very small, 'wafer-level' package, which measures



(a) Top (b) Bottom

Figure 4.9: CAD drawing of the custom RF front-end board.

$0.86 \times 0.86\,\mathrm{mm}^2$; as the board was assembled by hand, this proved impractical to solder and so only the primary antenna input was used on the board. Figure 4.10 shows an assembled board, with the secondary input (labelled S1) being unused. The figure also shows the use of the white box on the top silkscreen layer, to show any information about the board.

4.6 GNSS Signal simulator

There are many issues with using real GNSS data to test a receiver, with the majority revolving around additional sources of error - such as propagation errors and not knowing the exact position of the satellites. Whilst they can be used to test a receiver, there will be some ambiguity in the results. In the case of satellite receivers, there is an additional difficulty of not being able to easily acquire the data. One solution to this is to use a GNSS signal simulator, which simulates the signals that would be received by a receiver in a wide range of circumstances. We were very fortunate that Spirent Communications agreed to lend us one of their GNSS signal simulators that was configured for the GPS constellation for both terrestrial and satellite receivers. Their simulator produces in-depth log files during each simulation, which list, amongst others, the satellites being broadcast, with their position and pseudorange, and the position of the receiver. This is incredibly useful, as it allows both the acquisition, tracking and navigation calculation to be checked for accuracy.

As the simulator was only available for a limited period, it was used with the custom RF front-end board and the baseband recorder to create baseband data files that could be used repeatedly. In total, approximately $4\,\mathrm{TB}$ of baseband was collected from the

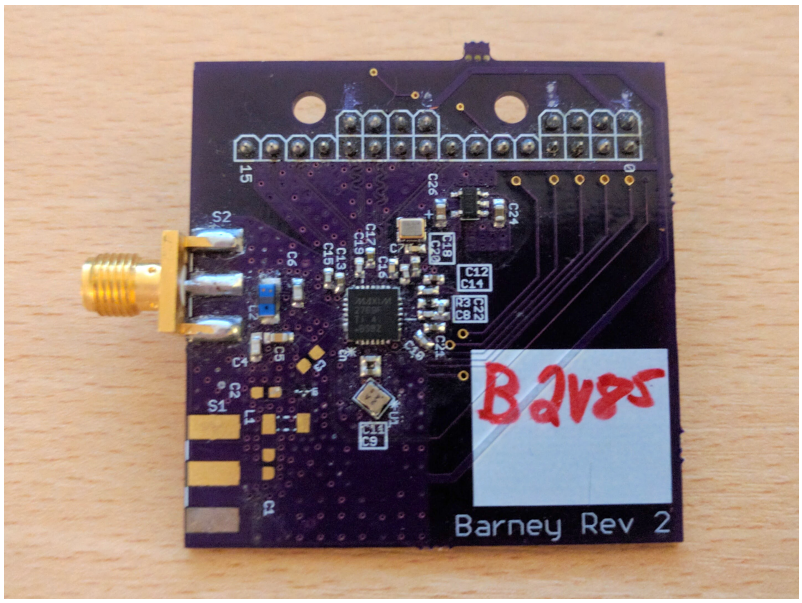


Figure 4.10: Assembled custom RF front-end board.

simulator (around 150 hours), which was recorded in blocks varying from 10 minutes to just under 3 hours (due to the size limit of the SD card). This data covers a wide range of scenarios, including static and moving receivers on the Earth's surface and a variety of satellite orbits. These were recorded with both an isotropic antenna and a simulated patch antenna. There was also additional data recorded to characterise and debug the receiver design.

Chapter 5

Software receiver

The software receiver follows on the from the prototypes written in Python (see section 3.3.1), with it being implemented in the C programming language. This was done for two reasons, the first was performance. Whilst Python code is very easy and quick to write, it tends to perform badly in data heavy tasks and it is not possible for it to have a similar performance to lower level programming languages (such as C)[1]. The second reason is that high level languages abstract the details of an implementation away, which is normally a good thing. However, considering the intention of designing a hardware version of the receiver, knowing these details, and being able to change them, is advantageous.

The software receiver can be broken down into its key components for acquisition, tracking and the navigation solver, with the data flow for the receiver shown in figure 5.1.

5.1 Acquisition

The acquisition, like in the Python prototype, is based around the FFT method. Rather than implementing the FFT, the FFTW library was used, which is one of the highest performing implementations that is available (Frigo and Johnson, 2005). It does this by adapting its functions for different architectures, with it finding the best version of its algorithms for a given system. This process can take a considerable amount of time, but does not need to be performed on each use. Instead, the optimisation process can be run once - for the required transform size and system - with the library's algorithm choice being stored into a 'wisdom' file for future use. It is also more efficient to perform an

[1] It is, however, possible to write a library for Python in a lower level language, which will result in similar performance. But essentially, this is using a low level language from a high level language, and removes any advantages of Python being easier and quicker to write.

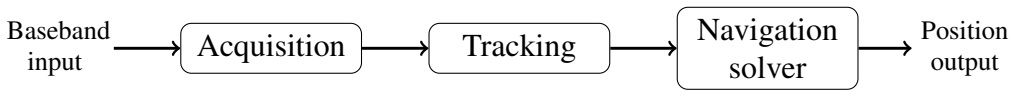


Figure 5.1: Data flow for the software receiver.

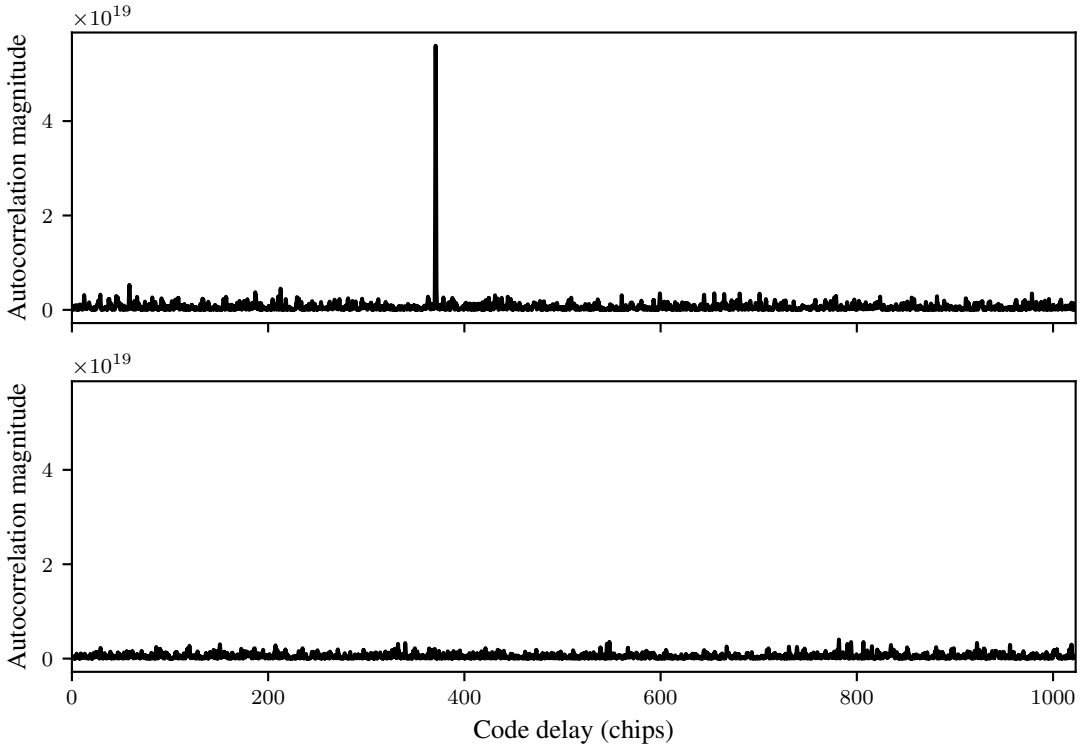


Figure 5.2: Comparison of FFT acquisitions where a satellite is visible (top) and not visible (bottom), for the same Doppler frequency, using 10 codes.

FFT whose length is 2^n, so the C implementation pads the input data with zeros. This also has the benefit of slightly increasing the FFT's resolution.

The only other notable difference is that the acquisition can be run in a comparison mode, which is believed to be novel. When a satellite is not in view of the receiver, performing an FFT acquisition will result in random peaks. However, when a satellite is visible, an FFT acquisition will produce a peak, whose amplitude is proportional to the signal to noise ratio. An example of this is shown in figure 5.2, where the top graph shows the results of an acquisition where the satellite is present, and the bottom graph showing when the satellite is not present. Both of these acquisitions were performed on the same data, and so the graphs are comparable, with the amplitude of the bottom graph being approximately equal to that of the top graph when the single peak (just below a code delay of 400) is removed. If two FFT acquisitions are done sequentially, the peak for a visible satellite should be in a similar location, with small variations in the Doppler frequency and the code delay being expected, due to the movement of the receiver and satellites.

This can be used in two ways, the first is as a secondary discriminator to a typically FFT acquisition. This is where a standard FFT acquisition is performed, where the visible satellites are determined by their perceived signal strength (the size of the correlation peak), with subsequent acquisitions comparing the results to the previous results to re-

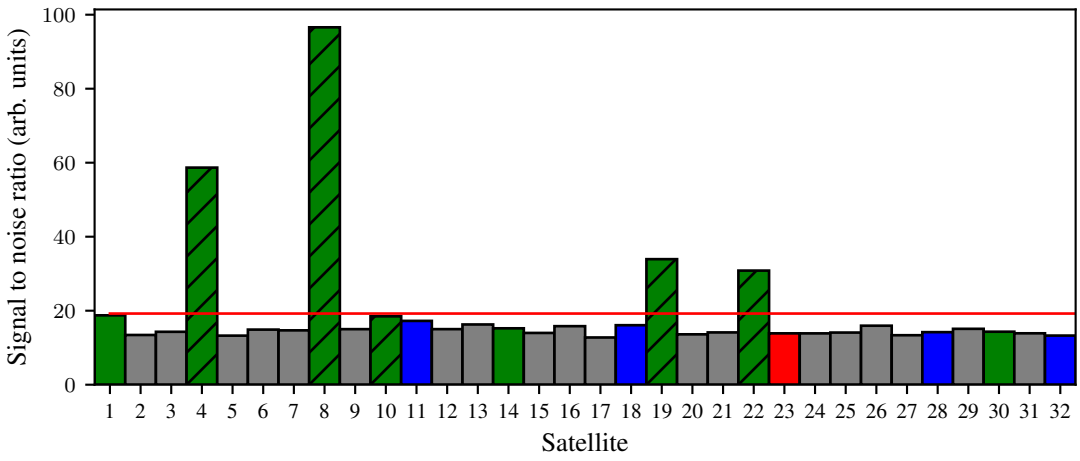


Figure 5.3: Example of using the comparison method to select visible satellites, with the acquisition using 10 codes and a spacing, for the comparison, of 1 s. See text for details.

move any satellite that is unlikely to be transmitting (e.g. large, unrealistic change in the satellite's Doppler frequency). An example of this is shown in figure 5.3, where the satellites selected from a single FFT are shown with hatching. The horizontal line shows the average correlation across the satellites, which is typically the threshold for the single FFT method. The results from the comparison are shown using colours, with green being those that were marked as visible, red being those that were marked visible but were not transmitting (false positives) and blue marking those that were transmitting but were missed by the acquisition. Grey is used to mark satellites that were not visible and were not transmitting[2]. Whilst figure 5.3, shows a false positive for satellite 23, it is able to detect satellites 1, 14 and 30 which the standard technique did not, showing the potential utility of this method.

The second way this can be utilised is through using smaller length FFTs. Typically, FFT acquisition's are done with 5 or 10 ms of baseband data, as this gives the highest FFT resolution whilst not crossing a data/bit boundary. The second approach, uses the comparison to reduce the length of the FFT required, for example, instead of performing one FFT of 10 ms of data, perform two FFTs of 5 ms of data. This allows for two FFTs, and the comparison logic, to be run in approximately the same time as a single FFT, but does have downside of reducing the FFT resolution. However, it does reduce the memory consumption by half, which is advantageous for an embedded system.

[2]The baseband data used for this was recorded from a GNSS signal simulator and so the transmitting satellites were known.

5.2 Tracking

The C implementation of the tracking is significantly more complex than the Python version. In the Python version, the carrier tracking (both the error and filtering) was updated after every sample, which is very expensive and is not possible for a hardware receiver. The C implementation uses the more common integrate and dump method, where the tracking is updated at intervals by using a summed error. In addition to this, the carrier replication is implemented as a numerically controlled oscillator (NCO). The code tracking is also implemented using an NCO, but in a DLL instead of a PLL. Both of the error discriminators for the carrier and code tracking can be easily changed, allowing a range to be used - which is advantageous, as some are significantly more computational complex than others.

Alongside the carrier and code tracking is a phase lock detector. This is based on the design of Kaplan and Hegarty (2006) and features both an optimistic and pessimistic lock output. These are used for the carrier tracking to determine the bandwidth of the loop filters - allowing a wider filter to be used when not locked, then switching to a narrower filter when locked, or close to locking, which improves the acquisition time.

To maintain precision, the C implementation uses double (64 bit) floating points for the vast majority of calculations, with the only integers being used for loop counters and for the data decoding.

The data is decoded by determining the sign of the prompt, in-phase code integrator. As it is highly unlikely that the receiver will start tracking a channel perfectly in line with the bit edges, each bit is oversampled into 20 sub-bits[3]. After 20 sub-bits have been collected, a data edge is searched for, and if one is found, the length of the next bit is modified (to either 19 or 21 sub-bits) so that the edge is closer to the start of the bit period. When the bit period is correctly aligned, the receiver is significantly better at decoding the data, as each bit has a higher amplitude. Prior to the detection of a frame, the decoder does not consider the sign of a bit's amplitude to be a specific bit value[4]. Instead, the decoder searches for the frame's preamble in both signs. When the preamble is found, the first word's parity is checked and, if it is valid, the entire frame is stored to be decoded; otherwise, the decoder continues to search for the frame's preamble. The frame decoding is not particularly complex, as it checks the parity of each word in the frame and then outputs the frame's data. However, only the first 3 frames are decoded, with frames 4 and 5 being discarded as they contain data that is not required by the receiver (i.e. the almanac).

[3] 20 was chosen as it was convenient - each bit is $20\,\mathrm{ms}$ long and the integrators were chosen to integrate for $1\,\mathrm{ms}$.

[4] The amplitude can be either ± 1, but the data can only be 0 or 1.

```
positive preamble detected at 89631168 (code offset 8539)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27003
    subframe number: 3
    C_ic: -10
    Omega_0: -837274202
    C_is: -16
    i_0: 652339396
    C_rc: 9203
    omega: -2144385594
    Omega^dot: -23493
    IODE: 36
    IDOT: -1141
positive preamble detected at 187839168 (code offset 8491)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27004
    subframe number: 4
    positive preamble detected at 286047168 (code offset 8443)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27005
    subframe number: 5
positive preamble detected at 384255168 (code offset 8395)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27006
    subframe number: 1
    week number: 842
    L2: P code on
    URA index: 0
    SV health: okay
    SV health code: 0
    NAV stream on L2 P code
    T_GD: 7
    IODC: 36
    T_oc: 10350
    a_f2: 0
    a_f1: 55
    a_f0: 84901
positive preamble detected at 482463168 (code offset 8347)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27007
    subframe number: 2
    IODE: 36
    C_rs: -2391
    Delta n: 13623
    M_0: -1135845715
    C_uc: -2122
    e: 14713626
    C_us: 2462
    sqrt(A): 2702015276
    t_oe: 10350
    curve fit = 4 hours
    AODO: 31
```

```
positive preamble detected at 580671168 (code offset 8299)
first word parity good
preamble: good
    integrity level: legacy
    TOW: 27008
    subframe number: 3
    C_ic: -10
    Omega_0: -837274202
    C_is: -16
    i_0: 652339396
    C_rc: 9203
    omega: -2144385594
    Omega^dot: -23493
    IODE: 36
    IDOT: -1141
```

Listing 5.1: 6 GPS frames decoded using the tracking program.

Listing 5.1 shows 6 frames that were decoded from a simulated GPS signal, only the time of week (TOW) and the subframe number are decoded from the discarded frames, as the TOW is important for calculating the navigation solution. In the listing, the data decoded from the subframe is inset from the left-hand side, with the variable name before the colon and its value being after. The majority of the variables are for the satellite's ephemeris, with the `a_f0`, `a_f1` and `a_f2` being clock corrections. The `IODE` and `IODC` are issues of data (for the ephemeris and clock, respectively). When a parameter of the ephemeris or clock corrections changes, the issues of data will also change - resulting in a variable which uniquely identifies the combination, within a given timespan. The three other types of lines in the listing, starting with `positive preamble detected at`, `first word parity good` and `preamble:` show the decoders state. The first says when a preamble has been detected - with the number following it being the sample number of the baseband. The second states if the first word has been received correctly. As it is possible for the preamble sequence to appear in the frame's data, this is needed to ensure that a detected preamble is actually a preamble. The last of the three is an output from the decoder, which performs a check on the preamble when decoding the entire frame.

5.3 Navigation solver

For the calculation of the navigation solution, the pseudoranges for each satellite have to be known. This is done by running all the tracking channels concurrently (at least with respect to the data) and then looking at their code delays, which frame they last decoded and the current bit (including sub-bit) that they are processing. The navigation solver, other than using SVD for calculating the pseudo-inverse of a matrix, is fairly typical of what is found in a GNSS receiver, with the exact implementation being close to that recommended in Global Positioning Systems Directorate (2014).

5.4 Fixed point model

To ensure the correctness of the hardware design, a version of the C implementation was changed to use fixed point maths. This allowed the typical number of bits required at each stage of the processing to be measured, which helped optimise the hardware design, and ensure that the tracking would perform as expected. The model also considers the latency in both the error calculation and filtering. The main advantage of the model is that it allowed parameters to be tweaked, including the choice of error discriminators, with the results being available in a useful time scale. Whereas, the simulation of the hardware required orders of magnitude more time[5].

[5]Whilst it was possible to get the data from the hardware itself, the large amount of data required to see how the tracking was performing - or more determine why it was not working - was impractical for the hardware to output.

Chapter 6

Hardware receiver

Whilst software receivers are useful for prototyping, as well as for niche applications, they require a lot of processing power and so consume a lot of electrical power, which is not compatible with femto-satellites. Hardware that is designed for a particular task consumes considerably less power than software that is emulating it. However, the cost involved in developing a custom designed integrated circuit (commonly referred to as an Application Specific Integrated Circuit or ASIC) is prohibitive - for example, $1 million for a tapeout of a processor (converting a design to transistors and physically laying them out into a chip package) is considered to be a significant reduction in cost (EE Times, 2016). An alternative to this, which is also frequently used as a stepping stone between software and ASICs, is a Field Programmable Gate Array (FPGA). FPGAs consist of look up tables (LUTs) and flip-flops that are grouped together into logic cells[1]. The contents of the LUTs and the wires in both the cell and connecting them to others, are determined by a design that is loaded into the FPGA[2]. Once the design is loaded, an FPGA acts similar to custom hardware, with some limitations - such as a higher power consumption (due to additional and unused logic) and a slower maximum operating frequency (due to delays between logic cells). It is for this reason that FPGA designs are often referred to as being 'hardware', as they are a very close approximation. In particular, the programming languages used to describe an FPGA's design (usually Verilog or VHDL) are the same that are used to design ASICs, allowing designs to transition from one to the other[3], often with only minor modifications.

[1]This is a simplified description and is similar to the first FPGAs that were commercially available. Modern FPGAs are a little more complicated, consisting of dedicated clocking resources, logic cells that are dedicated to mathematics (often referred to as DSP cells) and hard cores - specific hardware that is designed to perform a certain task (such as a RAM or communications interface, or even processors). All of these are designed to improve performance and reduce power consumption, making FPGAs more capable.

[2]Most FPGAs use RAM to hold the design, however, some use flash memory and there are also some one-time programmable FPGAs which use a fuse-based system, to disconnect certain wires.

[3]This is typically in the direction of FPGA to ASIC, however, some manufacturers have moved from ASICs to FPGAs as they allow more flexibility and a shorter development time.

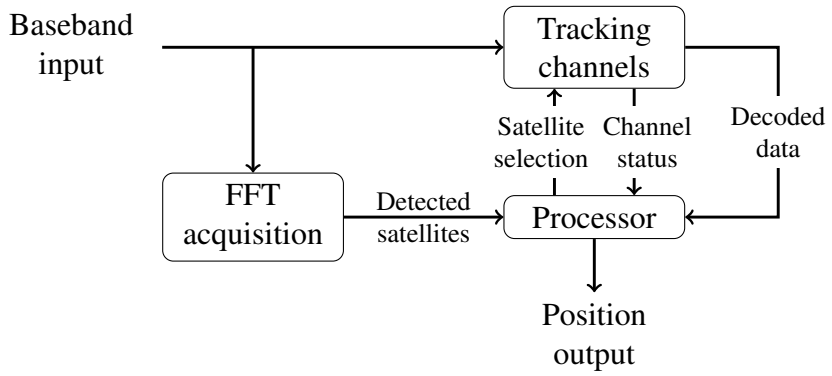


Figure 6.1: Structure of the SoC, with the processor controlling the tracking channels, rather than being directly in the data path.

The intention of the hardware receiver was for it to be modularised. So whilst it would be targeting the legacy GPS signals initially, it would be easily extendible so that it could be used with the modernised signals and other GNSS. This modularisation ranges in size, from entire channels down to different error discriminators, and even to particular implementations of filtering. The concept, and the eventual goal, is for this modularisation to enable a design to be tailored for different applications, including high accuracy and low power.

Whilst the interfaces between different elements of the tracking channels are designed for small amounts of data, the configuration of and the output of each channel uses a lot of data. As it is possible that there would be a large number of channels in certain implementations, this is an ideal candidate for a common bus design. With the channels requiring external control logic, to select which satellites to acquire, it becomes beneficial to use a system on chip (SoC) based design.

6.1 SoC design

There are many advantages to a SoC based design for a GNSS receiver, but by far the most important is the greater interoperability that is inherent in a SoC, which results in a smaller and more power efficient design.

In our GNSS receiver design, the processor is used as a controller rather than as a central part of the signal processing, as shown in figure 6.1. Therefore, the processor will spend the vast majority of its time in a low power state, only becoming active to monitor the signal tracking and to process the acquisition and tracking results. With acquisition results arriving at the processor in bunches, with only around a millisecond gap between each result, the acquisition processing has to be done as quickly as possible, whilst still remaining energy efficient.

Whilst FPGAs with hard-core processors have been available for many years, such as the Virtex-II Pro which was released in 2002 (Xilinx Inc., 2011c), it has only been

in recent years that hard-core processors have been integrated with the lower and middle range FPGAs, such as Xilinx's Zynq in 2012 (Xilinx Inc., 2016). These all use ARM's Cortex application (or A series) processors, ranging from single to quad cores, with their operating frequency between high megahertz and low gigahertz.

It is important to understand that ARM's application processors are embedded processors designed for high performance, they are typically found in smartphones, tablets and some low-end computers, as well as a range of other consumer electronics. The individual architectures vary, but floating point and memory management units, along with support for media extensions, are commonplace. With the complexity of these processors, it is recommended to use a Linux based operating system on them, to both correctly use them and to use them efficiently. With the processor in our SoC being primarily a controller, the use of such high-end processors is considerably overkill and would be a significant waste of energy.

Whilst there are certain advantages to using a full-scale operating system - such as simpler debugging and programming - there is also the large drawback that the peripherals require drivers so that they can be used by a user space program[4]. This is a significant over-complication and hindrance for our design, that is practically guaranteed to waste power without any real gain.

It is worth mentioning that there is currently one manufacturer that makes FPGAs with hard-core processors that are not application processors - Microsemi. Their SmartFusion devices have an ARM Cortex-M3 processor in addition to a large number of peripherals and the FPGA fabric. Whilst in some aspects the Cortex-M3 is beneficial, it has many peripherals that are unnecessary (such as an Ethernet MAC, ADCs and DACs) and lacks hardware DSP blocks, which would make the design consume more power and require more logic. The SmartFusion2, which was released in 2014 (after the FPGA design was started), although it suffers from a similarly large set of peripherals, at least has hardware DSP blocks as part of the FPGA fabric - but it is worth noting that these "Math Blocks" are not as flexible as the Xilinx and Altera equivalents (the DSP48A1 and the "variable precision DSP block", respectively), which would make it less favourable (Microsemi Corporation, 2016; Xilinx Inc., 2014).

With the available hard-core processors not being a viable option, this leaves soft-core processors, of which there are many, ranging in size and performance. Most FPGA manufacturers have their own soft-core processors, that are typically locked to the manufacturer's own devices. For example, Xilinx has the PicoBlaze and MicroBlaze soft pro-

[4]It is worth noting that it is possible to access peripherals in Linux without drivers (through the /dev/mem character device). However, when using this method, a bug in the controlling program can crash the operating system in a way that is very difficult to debug (somewhat similar to stack corruption, as the effects of the bug might not be seen immediately). Whereas the equivalent fault in a driver would be considerably easier to debug, with the kernel providing a stack trace, and, in some cases, a bug in the driver would not cause the operation system to crash. This essentially makes the writing of a driver easier than using the mem character device for anything but the simplest of peripherals.

cessors and Altera has the NIOS II. There are additionally open source and commercial processor IPs that are available as soft-cores. There are many advantages and disadvantages to each processor, and whilst it is impossible to compare all the available processors, we have chosen a small number which broadly cover the available range.

First of all, there are a fairly large number of 8 and 16 bit processors, with many designed for use in FPGAs. However, these are intended to be used to control processes and so are typically quite limited, in terms of functionality, ability and code size. Whilst the processor in our design is not in the signal path, it does have to process the acquisition results, requiring some calculations relating to the strength and offsets of the signals, and the tracking results, requiring a non-trivial amount of data movement and reduction. For example, Xilinx's PicoBlaze is a 16 bit processor, with an instruction memory that can hold up to 1000 instructions and 64 bytes of scratch memory. Whilst it can run at 128 MHz in a Spartan 6 FPGA, admittedly at 2 clock cycles per instruction, and can interface to external memory, it is very limited in what it can do. In a similar way, the J1 is an open source Forth processor, which is very small - less than 200 lines of Verilog and requiring 42 registers and 444 LUTs. However, the Forth language, despite its age and lack of type checking, is entirely stack based, making it difficult to program for. It additionally suffers from the same problems as the PicoBlaze, where the 16 bit data bus / word size limits its ability to perform tasks quickly.

An argument for using 32 bit, instead of 16 bit, processors that is often given is that they increase performance and decrease code size. For any moderate task, calculations that are greater than 16 bits, or potentially could be, are more efficient on a 32 bit processor. For example, adding two 32 bit numbers requires a single instruction on a 32 bit processor but at least two instructions for a 16 bit processor, providing the operands are already in the processors registers. If the required data needs to be fetched from memory, either RAM or memory mapped peripherals, then the number of instructions significantly increases for a 16 bit processor to at least 6 compared to 2 of a 32 bit processor. Whilst many 32 bit processors use 32 bit instructions, which do not offer much of a reduction in code size over 16 bit processors, which typically use 16 to 18 bit instructions. ARM processors that implement the Thumb instruction set have the advantage of mainly using 16 bit instructions, there are additionally other 32 bit processors that use 24 bit instruction sets.

In terms of power consumption, 32 bit processors can be more efficient despite their larger size, as they operate at a lower frequency and can return to a low power sleep state sooner than an 8 or 16 bit processor, due to a smaller number of instructions needed to perform the same task. The only argument that remains for using 8 or 16 bit processors is that they are simpler. However, whilst this argument might have some validity when comparing microcontrollers, where the peripherals of 8 and 16 bit devices are simpler, the argument does not apply well to the processor core, where the only change is operating

modes and interrupt behaviour, which widely vary between processors despite the data width.

The performance of a processor is not particularly tied to its operating frequency, particularly as larger processors make use of pipelining to improve performance. To rectify this, there are a range of benchmarks that can be run on a processor to determine how well it performs a task. One of these is the Dhrystone benchmark, where a score is generated by measuring how long an iteration of the Dhrystone test takes to execute. Whilst there are criticisms that the Dhrystone benchmark is not particularly realistic of typical processor use[5], it is a widely used benchmark. As the score generated by the benchmark is dependant on the processor's operating frequency, it is often converted into Dhrystone MIPS (Million Instructions Per Second) per megahertz ($\mathrm{DMIPS/MHz}$) to remove this dependency.

With the comparison being more in-depth for 32 bit processors, difficulties inevitably arise. Comparing by benchmark performance alone, whilst useful in finding the best performing processors, is not particular useful when power efficiency is a concern. In an FPGA, the power consumption can be split into two causes - static and dynamic. Static power consumption is the general power consumed by the logic, in terms of configuring it and allowing it to be used. In particular, modern FPGAs will remove the applied power to unused areas of the FPGA fabric to reduce the static power consumption. Dynamic power consumption is the power consumed by the logic being active, with the main source being the logic switching between levels. In particular, higher clock frequencies will consume considerably more dynamic power than lower clock frequencies. For a processor to be as power efficient as possible, it needs to be able to operate at a low clock speed, using as little logic as possible, whilst still being able to perform the required operations quickly. In short, we are looking for a high number of $\mathrm{DMIPS/MHz}$ whilst using as few logic elements as possible.

As different FPGAs use different logic cells - with the number and size of LUTs, and the number of registers and muxes, per logic slice varying - comparing logic use between devices and manufactures is difficult. So here we define a logic slice as containing 4 LUTs and 8 registers. This definition is chosen as it is the number of LUTs and registers in a slice for Xilinx series 6 and series 7 FPGAs (Xilinx Inc., 2010a, 2011a).

Additionally, different FPGAs are targeted at different markets, with varying costs, and so perform differently. Ultimately, higher end devices, such as Xilinx's Virtex and Altera's Stratix, will perform considerably better, with the number of logic cells used by a design decreasing and the maximum operating frequency increasing. Whilst this is not much of a concern when comparing frequency independent data, when comparing logic

[5]In particular, it can easily fit in a processor's cache and does not test floating point performance. However, these are largely irrelevant for an FPGA based processor where the use of either an instruction or data cache, or a floating point unit is unlikely.

Table 6.1: Comparison of 32 bit soft-core processors

Processor	Pipeline	Performance (DMIPS/MHz)	Logic usage			Interconnect
			LUTs	Registers	Slices	
Altera NIOS II/e	5 stage	0.15	1248	914	312	Avalon
PicoRV32	none	0.341	1353	577	339	native
OpenRISC 1200	5 stage	1	4850	2337	1213	Wishbone
ARM Cortex-M0 DS[a]	3 stage	1.033	3478	1016	870	AHB-Lite
Altera NIOS II/f	6 stage	1.13	2486	1895	622	Avalon
Aeroflex Gaisler Leon3[b]	7 stage	1.4	7983	3793	1996	AHB
Xilinx MicroBlaze	5 stage	1.44	4938	4425	1235	AXI4
Aeroflex Gaisler Leon4	7 stage	1.7	4000[c]	unknown	unknown	AHB

[a] ARM Cortex-M0 DesignStart processor (r0p0), configured with 16 interrupts and a 32 cycle multiplier. Dhrystone benchmark compiled with GCC 5.2.1 using the small-multiply option.
[b] Diligent Nexys 3 default configuration. [c] LUTs based on Stratix 3 and Virtex 5, for processor core area only.

Table 6.2: Comparison of the 32 bit soft-core processors' logic efficiency

Processor	Performance DMIPS/MHz	Logic usage Slices	Logic efficiency
Altera NIOS II/e	0.15	312	20800.0
PicoRV32	0.341	339	994.1
OpenRISC 1200	1	1213	1213.0
ARM Cortex-M0 DS [a]	1.033	870	842.2
Altera NIOS II/f	1.13	622	550.4
Aeroflex Gaisler Leon3 [b]	1.4	1996	1425.7
Xilinx MicroBlaze	1.44	1235	857.6

[a] ARM Cortex-M0 DesignStart processor (r0p0), configured with 16 interrupts and a 32 cycle multiplier. Dhrystone benchmark compiled with GCC 5.2.1 using the small-multiply option.
[b] Diligent Nexys 3 default configuration.

usage, FPGAs from different manufacturers need to be targeting a similar market, to make the comparison meaningful.

From table 6.1, the highest performing processors are the Leon3 and 4, and the MicroBlaze. Whilst the exact logic usage of the Leon4 is not available, the Leon3 and MicroBlaze have the highest logic use of the compared processors. The smallest processors are the NIOS II/e and the PicoRV32, which are also the lowest performing processors. Comparing the processors in the middle is more difficult, so to simplify the comparison, we can define the logic efficiency as the logic utilisation divided by the benchmark performance. From table 6.2, the most efficient processors are the NIOS II/f, the Cortex-M0 and the MicroBlaze.

All the processors in table 6.1, with the exception of the ARM Cortex-M0, use 32 bit instructions. The Cortex-M0 uses ARM's Thumb instruction set, where the majority

of the instructions are 16 bit, with the only 32 bit instructions being branch with link (BL), which is used to call subroutines, the synchronisation barriers (DSB, MSB and ISB) and two instructions for moving data to and from the processor's special registers (MRS and MSR). This smaller instruction size is advantageous for a lower power consumption, requiring not only less data to be stored in memory, but also fewer memory reads and therefore fewer bus transactions.

With the Cortex-M0 having a high logic efficiency, as well as it being FPGA vendor agnostic, it is the most attractive soft-core processor available. Whilst it is a commercial product, it does use the AHB interconnect, so a SoC that is designed to use it is not tied to it. It also has the advantage that the AHB interconnect is widely used and that the Thumb instruction set is likely to increase its power efficiency.

6.2 FFT acquisition

The FFT module is split into several sub-modules to separate the functionality into logical blocks, however, the main part of the calculation - the forward and inverse FFTs are performed by Xilinx's FFT IP (Xilinx Inc., 2012). This IP uses an AXI streaming interface, where the data is controlled through the use of valid and ready signals, with data being transferred when both are high. The IP is configured - i.e. changed from forward to inverse - also through an AXI interface. Due to the size of the FPGA, the largest FFT that can be performed is for 2 PRN codes in length ($32\,768$ samples), with this restriction mainly due to the limited amount of block RAM that is available. Like the software version, and unlike the Python prototype, the FFT IP uses 2^n length FFTs, with the module padding both the PRN and sample inputs.

The module consists of two RAMs, one for the PRN data and one for the sample data, and there are two main sub-modules, the loader and the unloader. The acquisition process is very similar to the software approach, with a few changes required to make it work efficiently in hardware. The FFT module, as well as the loading and unloading sub-modules, implement their functionality by using state machines.

When the FFT module is enabled, it samples a bit mask, where any high bits are satellites that should be searched for. If the bit mask is non-zero, the module configures the Xilinx IP for forward FFTs and uses the loader to load the sample data into the IP. The unloader is also enabled, with the destination being set to the sample RAM. When the loader completes, the module restarts the loading process with the first selected satellite's PRN code. This is done to increase the performance of the FFT IP, as the loading and unloading can either be concurrent or sequential. The unloader is re-enabled as soon as it becomes idle, with its destination changing to the PRN RAM. In addition to this, because the FFT-based algorithm requires one of the signals to be conjugated, the unloader conjugates the IP's output when unloading to the PRN RAM. When the unloader next becomes

idle, the FFT module has all the data ready for performing the search for the satellite, so configures the IP for inverse FFTs. The next step is to demodulate the sample data across a range of frequencies, achieved by adding an offset to the sample RAM address and implemented by repeatedly switching between a loading and unloading state. For this part of the calculation, the loader and unloader perform a larger task, initially, they act is if they were simple counters. At this point, the loader uses a complex mutliplier (which utilises the FPGA's DSP blocks) using the data stored in the two RAMs. The unloader, instead of storing the IP's output data, processes it to detect any peaks in the signal. Both of these have some latency which the module has to take into consideration. Each time after the unloader becomes idle, the required data (e.g. the satellite number, peak location and strength, and the Doppler value) are outputted from the FFT module, and, if the satellite is not the last in the bit mask, the demodulation frequency is adjusted and the search is performed on the new satellite.

The complex multiplier is an implementation of a Xilinx recommendation given in Xilinx Inc. (2014), with two versions available for use - one uses 3 DSP blocks and has a latency of 2 clocks, the other uses 4 DSP blocks and has a latency of 3 clocks. The main advantage of the 4 block version is that it can be used with clocks up to approximately $455\,\mathrm{MHz}$, where as the 3 block version has a maximum frequency of approximately $115\,\mathrm{MHz}$. With the hardware design targeting a clock frequency of $50\,\mathrm{MHz}$, both versions are likely to help timing closure, with the 4 block version not significantly aiding more than the 3 block, with its extra performance not being utilised. Therefore, to save resources and power consumption, the 3 block version was chosen.

The peak detector calculates the squared magnitude of the inverse FFT's output and stores the highest value, whilst creating an average. Unlike the complex multiplier, it does not use any hard elements in a specific way. The only notable functionality is that the average is scaled to reduce the number of bits required to store it, and the width has been further reduced for typical values, so an overflow flag has been added as it is possible for the calculated average to exceed the width of the data.

6.3 Correlator

The correlator has been designed in 6 stages, enabling the frequency of each stage to decrease with the stage number (figure 6.2). Each stage is clocked with the exception of stage 0, which uses combinational logic to map the baseband data to the correct numerical values. This is why stage 0 is not referred to as a stage. The first stage is the carrier removal, where the baseband data is multiplied with a locally generated sine wave at the carrier frequency. The sine wave is implemented as a 16 value, 3 bit lookup table, using the top 4 bits of the NCO's phase to select the required value. Each value only needs to be 3 bits due to the limited number of bits used in the baseband data, which limits the

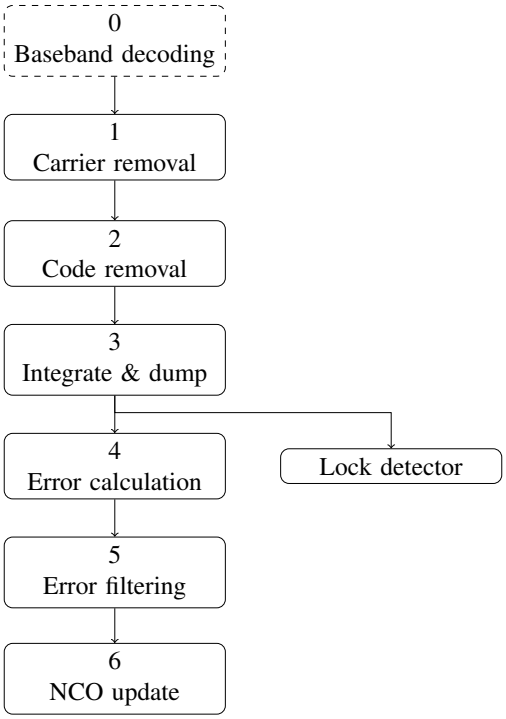


Figure 6.2: Correlator data stage structure.

required length of the table. The second stage is the code removal, which is fairly similar to the carrier removal; the code NCO is, however, different. With the carrier NCO using a lookup table, any arbitrary step can be made, however, the code NCO uses a generator to produce the code, which saves RAM usage and only requires a small amount of logic to implement. This generator can only perform one step at a time. To avoid any issues, the code NCO is updated with every sample rather than at the chipping frequency (which is once every 16 samples). This allows the code NCO to operate at up to 16 times faster than the nominal rate, with any rate higher than this being an indicator of a fault. Both the carrier and code removal work at the sample rate and work in a pipelined fashion.

The third stage is the integrate and dump, which accumulates the early, prompt and late output of the code removal for both the in-phase and quadrature signals over an integration period (which is $1\,\mathrm{ms}$). At the end of this period, the data accumulators are stored into output registers for the next stage to access. The next stage - the error calculation - as a consequence, operates once per millisecond. The dump outputs are also used by the lock detector, which performs some low pass filtering on the prompt values before comparing them to produce the lock output. The error calculation for the carrier NCO is quite simple - it uses the magnitude of the quadrature integration with the sign of the in-phase integration, which is referred to as a directed Costas (Kaplan and Hegarty, 2006). In contrast, the calculation of the code error is a little more involved. The code error requires the magnitude of the early and late components, which is typically achieved by,

$$|m_n| = \sqrt{I_n^2 + Q_n^2},$$

where n is the component and I and Q are the values from the integrators. This is problematic, as square roots are a complicated calculation to perform. Instead an estimator for the magnitude is used,

$$|m_n| \approx \max\left(|I_n|, |Q_n|\right) + \frac{1}{4}\min\left(|I_n|, |Q_n|\right),$$

this, according to Griffin (1999), is the lowest error magnitude estimator, whilst only requiring bit shifts. As the hardware used fixed not floating point maths, the hardware implements this estimation with the addition of a scaling factor, namely,

$$4\,|m_n| \approx 4\max\left(|I_n|, |Q_n|\right) + \min\left(|I_n|, |Q_n|\right).$$

This scaling factor is later removed.

With the differences in the complexity of the carrier and code error calculations, their outputs are misaligned, with the code error output becoming available 32 clocks after the carrier error. This is very useful, as it allows the same filtering hardware to be used for both the carrier and code error filtering, as the filtering takes 4 and 5 clock cycles respectively. The use of hardware for the error calculation and filtering is uncommon amongst GNSS designs. Most designs only perform up to the integration in hardware, with a processor performing the calculations. Whilst under typical terrestrial circumstances, the extra delay required by the processor to calculate the filtered value (and deal with many correlators requesting the values at the same, or similar, point in time) is negligible, the extra delay in a high velocity environment will result in the loops having to have a wider bandwidth, which has its own disadvantages. The final stage applies the filtered error to the carrier and code NCOs' steps.

6.4 Decoder

The decoder uses the same logic for decoding the navigation data from the signal as the software version. However, the decoder processes a word at a time, instead of a frame. This reduces the amount of memory required from the frame's 300 bits to the 30 bit word size. The software version decodes a frame at a time for simplicity, however, this simplicity is not present to the same extent in hardware and is not worth the extra resource cost.

In a similar way to listing 5.1, the hardware version of the receiver design was used with the same data, with its output shown in listing 6.1. This listing only shows the output of one active channel (correlator and decoder combined), purely for comparison with listing 5.1. The output format is different from the previous listing, here the point where the preamble is detected is not displayed, instead the `tick:` lines are when the

```
tick: 303594 TOW: 27003 alert: 0 enhanced: 0
  F3: IODE: 36
      Omega_0: -837274202 incl_0: 652339396 C_rc: 9203 C_ic: -10
      C_is: -16 omega: -2144385594 Omega^dot: -23493 IDOT: -1141
tick: 384339 TOW: 27004 alert: 0 enhanced: 0
tick: 465069 TOW: 27005 alert: 0 enhanced: 0
tick: 545862 TOW: 27006 alert: 0 enhanced: 0
  F1: week num: 842 IODC: 36 URA index: 0 SV health: 0 (0)
      t_oc: 10350 a_f2: 0 a_f1: 55 a_f0: 84901
tick: 626540 TOW: 27007 alert: 0 enhanced: 0
  F2: IODE: 36 curve fit = 4 hours
      M_0: -1135845715 ecc: 14713626 sqrt_A: 2702015276 Delta_n: 6811
      t_oe: 10350 C_rs: -2391 C_uc: -2122 C_us: 2462
tick: 707224 TOW: 27008 alert: 0 enhanced: 0
  F3: IODE: 36
      Omega_0: -837274202 incl_0: 652339396 C_rc: 9203 C_ic: -10
      C_is: -16 omega: -2144385594 Omega^dot: -23493 IDOT: -1141
```

Listing 6.1: 6 GPS frames decoded by the hardware receiver.

processor detects that a frame has been decoded (with the tick being the value of a clock in the processor). The inset lines show the decoded data, with the frame being indicated by `F1`, `F2` and `F3` (frames 1, 2 and 3, respectively). There are 2 `tick:` lines that are not followed by a decoded frame, these are for frames 4 and 5 which are not decoded, however, the TOW still increments when these frames have been received. The other parts of the inset lines match those in listing 5.1, showing that both the software and hardware decoders are able to decode the same data from the same input (as the baseband data for this was transferred to the hardware over the USB FIFO interface).

Correctly decoding the same data, however, does not mean that both the receivers are equivalent, as the quality of the receiver's position calculation depends on how well the receiver is able to track the satellite's carrier. Figure 6.3 shows how the three different receivers track the carrier offset, alongside the generated offset taken from the Spirent simulator's logs. Whilst the three receivers are noisier than the data in the simulator's log files, differences between the receivers cannot be seen. Figure 6.4, however, shows a smaller section of figure 6.3. In this figure, it is possible to see that the software receiver is the least noisy, with the hardware receiver having the most noise. In particular, the hardware receiver has more noise in its tracking than the fixed point model. All of the receivers have a small offset against the simulator's logs. There are several causes to this, firstly, the crystal oscillator which is used as a clock for the RF front-end is not precisely calibrated and will drift with both time and temperature. Secondly the precision in the simulator's log files is higher than what its hardware can produce, resulting in a nosier signal, and the simulator's hardware also relies on crystal oscillators, which will also drift and may not be perfectly calibrated. These explain why the software and fixed point receiver's have a similar offset. However, the hardware receiver's offset is explained by a

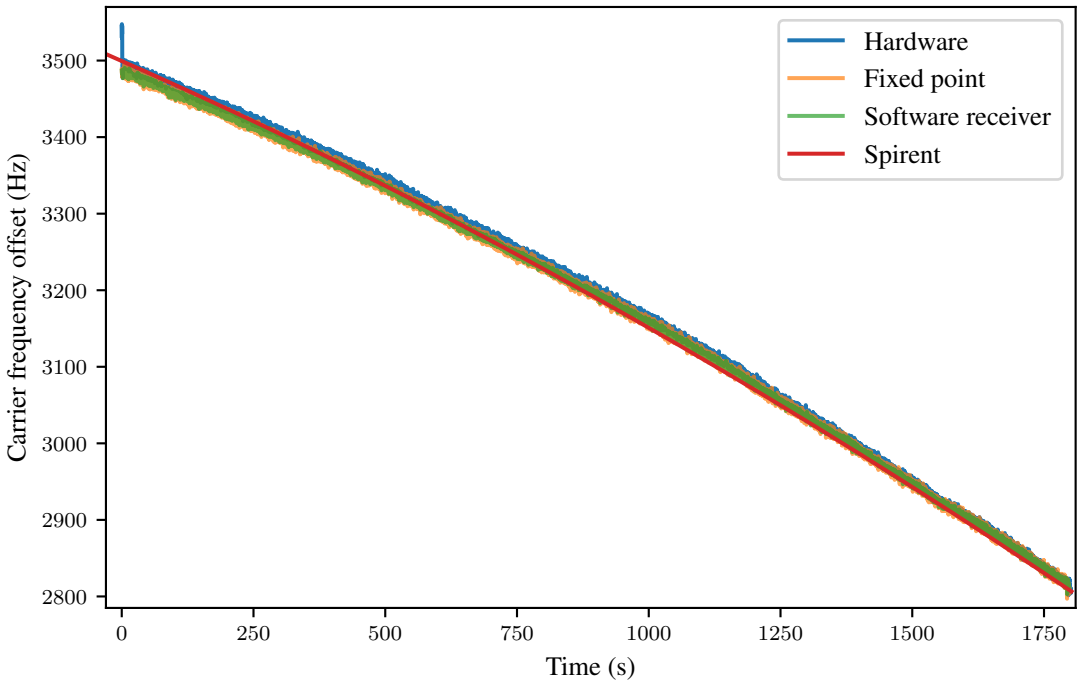


Figure 6.3: Carrier tracking of the three different receivers compared with the carrier offset being simulated by the Spirent signal simulator.

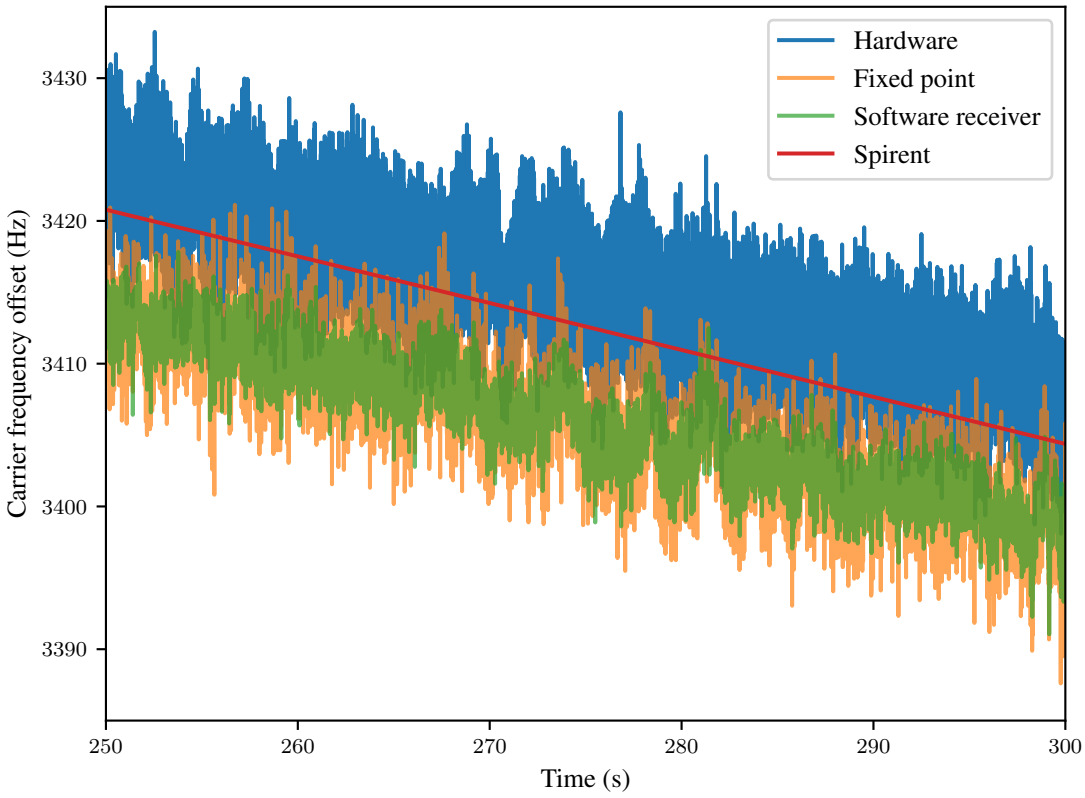


Figure 6.4: A zoomed in portion of figure 6.3, allowing the small differences in the tracking between the different receivers to be seen.

third reason - the hardware implementation of the tracking has an offset. Although these small offsets look undesirable, they will not affect the receiver's tracking performance as they are consistent or, in the case of the oscillators' drift, change only at a small rate. The main difference between the receivers, is the amount of noise they have - with the software receiver having the best quality tracking. However, these differences are quite small, with the tracking performance of the receivers being comparable.

6.5 Power consumption

To determine if the distributed design is practical, the power consumption of the receiver design was required. This was done by measuring the voltage and current of the receiver using high-end multimeters (6.5 digits for voltage, 7.5 digits for current[6]). The multimeters were chosen for several reasons. The first is that they have a higher resolution, dynamic range and accuracy than oscilloscopes, but they do have a smaller bandwidth. This is not particularly a problem, as both quantities should not be changing at a particularly high frequency. The ADC architecture of these multimeters is advantageous - they use integrating ADCs which, as they are connected to the input for the entirety of the measurement period, results in the value being the average of that measured. In another way, many short samples will average to the same value as one longer sample, where as, most ADCs are only connected to the input for a small portion of their measurement period. For measuring the power consumption, this means that a longer integration period can be used (which reduces noise), whilst the multimeters are still able to measure the average power consumption in an accurate way. As both the voltage and current have small fluctuations, the measurements were synchronised through an external trigger and carefully set trigger delay settings.

It was important to test the receiver design with known data, with data collected from the simulator chosen. However, this presents a problem - if the SD card is used, then its power consumption, as well as the power consumption of the baseband reading logic, would be included in the measurement; but a secondary board could not be used to generate the data as it would involve wires being used, which, as was previously found out, are prone to interference. The use of an SD card was the only reasonable solution, with the additional power consumption being compensated for by performing additional measurements to estimate the increase. These measurements were, the power consumption when idle, when only playing the baseband from the SD card and when performing the tracking. By knowing the increase between the idle and baseband playing power consumptions, an estimate of the increase can be made and then removed from the tracking measurement. Whilst this will not be the same power consumption as the tracking by itself, due to the

[6]The higher resolution multimeter was used for the current measurement as it has a higher dynamic range and only one 7.5 digit multimeter was available.

Table 6.3: Measured power consumption of the receiver design for the different levels of activity. The power consumptions were measured at 100 PLCs (integration time of $2\,\mathrm{s}$) and are averaged over 10 readings.

Number of channels	Power consumption (mW)		
	Idle	Baseband playing	Tracking
4	205.60 ± 0.04	332.6 ± 0.2	362.7 ± 0.2
6	216.68 ± 0.04	355.6 ± 0.2	418.5 ± 0.2
8	259.72 ± 0.05	388.4 ± 0.2	461.8 ± 0.2

Table 6.4: Calculated power consumption of the receiver, with the baseband player removed.

Number of channels	Power consumption (mW)
4	235.7 ± 0.2
6	279.6 ± 0.3
8	333.1 ± 0.3

extra logic's static power consumption, it does provide a good estimate and will be larger than that of the receiver under normal operating conditions. Table 6.3 shows the three measured power consumptions for a reasonable range of channel numbers. The difference between the baseband playing and idle power consumptions is the amount of power required to read the data from the SD card. It is this increase that needs to be removed from the tracking power consumption to determine the power consumption of the receiver and not the receiver with the SD card - which is what table 6.4 shows. When compared against the power budget of the PCBsat in table 2.4, the measured values in table 6.4 are less than the available $357\,\mathrm{mW}$ during the satellite's eclipse. From table 3.2, the navigation processor has a power consumption between approximately $22.4\,\mathrm{mW}$, at $4\,\mathrm{Hz}$, and $46\,\mathrm{mW}$, at $10\,\mathrm{Hz}$. This results in the distributed design reducing the power consumption by between $5.6\,\%$ and $13.3\,\%$.

Chapter 7

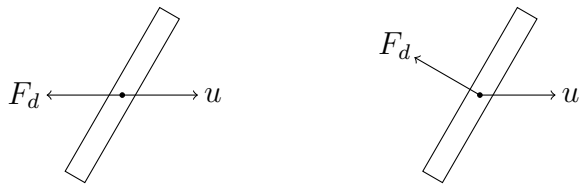
Orbital simulation

With the theoretical feasibility and the potential advantages of a distributed GNSS receiver being demonstrated, there is the question of whether a distributed receiver is practical. When the nodes are close together, the reduction in the power consumption will be more than the figures stated in section 6.5, primarily because the radio transceivers can operate at a lower transmitting power. However, as the nodes move further apart, there will come a point where they can no longer communicate with each other. Although this is a problem with the mission concept, a question that needs to be answered is: how long will it take for this to happen?

In low Earth orbit, the main dispersive force is drag. However, the vast majority of orbit simulations use ballistic drag to model the drag of satellites. Ballistic drag is the classic drag equation that is frequently taught,

$$F_d = \frac{1}{2}\rho u^2 C_D A,$$

where ρ and u are the fluid's density and relative velocity, C_D is the drag coefficient and A is the effective or projected area (NASA, 2015b). What might not be initially obvious, is that this equation does not consider torque and the drag force is always parallel to the fluid flow. When a flow is incident on a inclined surface, the drag force is parallel to the surface's normal. Figure 7.1 shows the differences between the two drags, with ballistic drag being obviously wrong. Whilst ballistic drag is a useful approximation, it is frequently used in orbit simulations as a fitting parameter, rather than having any



(a) Ballistic drag (b) Realistic drag

Figure 7.1: Diagram of the different drag effects, with u being the surface's velocity.

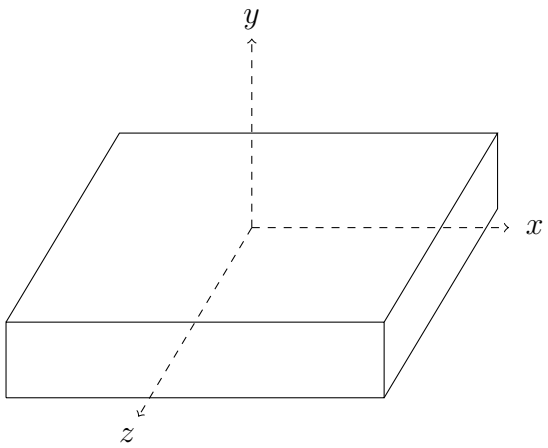


Figure 7.2: Coordinate system for orbit simulation.

physical significance (Allgeier, 2013). In particular, high-level simulations - used for precise modelling of GPS satellites - use many factors (including radiation pressure) but reduce the model to ballistic, or ballistic-like, drag to simplify its use (Allgeier, 2013). However, to correctly model drag, especially in space where the density of the atmosphere is very low, the flow must be considered to be free molecular - essentially considering individual molecules incident with the surface rather than an ideal flow.

In Barnhart (2008), the dispersion is modelled by varying the drag area by $1\,\%$. The physical basis for this is given as the random variations in the arrangement of the deployable antenna - which, whilst demonstrating the concept, does not make physical sense - as random variations, over a large time period, will be approximately equal. In addition to this, it only affects the satellites in one axis.

Previous work by Psiaki (2004), used a free molecular dynamics approach for a cubesat, but only considered rotation of the flow in a single axis, as they were interested in the aerodynamic stability of the satellite. Whilst this is a marked improvement over ballistic drag, it is not capable of modelling the dispersion. Therefore, a new model was constructed to accurately model the dispersion.

7.1 Simulation design

The derivation of the model allows the flow to rotate, with respect to the surface, in two axes. Figure 7.2, shows the coordinate system used by the simulation with the positive z direction being the satellite's ram direction. If we consider the incident face in this diagram (whose normal is in the positive z direction), any rotation of the flow round the z axis does not need to be considered on a per-surface basis for the drag, as it is equivalent to a rotation of the satellite. Utilising this symmetry, the derivation, contained in appendix

C, results in the pressure, p, and shear, τ, on each surface of,

$$
\begin{aligned}
p &= (2-\sigma')\, p_i + \sigma' p_w, \\
p_w &= \frac{\rho}{2}\sqrt{2\pi R T_w} N_i, \\
p_i &= \rho R T \left\{ \frac{u_1}{\sqrt{2\pi RT}} e^{-\frac{u_1^2}{2RT}} + \left(\frac{1}{2} + \frac{u_1^2}{2RT}\right)\left[1 + \operatorname{erf}\left(\frac{u_1}{\sqrt{2RT}}\right)\right]\right\}, \\
\tau &= \sigma \rho u_2 \left(\sqrt{\frac{RT}{2\pi}} e^{-\frac{u_1^2}{2RT}} + \frac{u_1}{2}\left[1 + \operatorname{erf}\left(\frac{u_1}{\sqrt{2RT}}\right)\right]\right),
\end{aligned}
$$

with the terms explained in the appendix. Whilst the rotation of the surfaces is important, the density and temperature of the flow are the main parameters that affect the magnitude of the drag. It is therefore important that the atmospheric model used is significantly accurate so that its inaccuracies do not affect the simulation's output. For this reason, the NRLMSISE 2000 atmospheric model was used (Picone et al., 2002), with both solar flux, magnetic intensity and polar corrections being used for specific date ranges.

Whilst commercial orbit simulators are available, with the most notable being AGI's Systems Tool Kit (STK), they are meant to be used as off-the-shelf tools for standard situations. Some do allow custom code to be used with them, however, they are not meant to be used in the way that the drag model requires. Whilst it might be possible to make a working solution, it is considerably easier, and more predictable, to implement the drag model in a custom orbit simulator.

The aim of the simulator was for it to be accurate and have a high performance, whilst being extendible and easy to control. From the first two aims, it was obvious that a low level programming language would be the most suitable as it would allow for the most control, with the C programming language being chosen. Whilst any program can be designed to be extendible, making a program easy to control can be difficult. From previous experience, simulations can involve a myriad of parameters that need to be correctly configured and this, especially with the requirement of surface positions relative to the satellite, was unlikely to be the exception. Therefore, a light-weight scripting language was embedded into the simulation program. This scripting language, Lua, is used in a wide range of applications, but excels as it can be used for simple configurations (like text files) or as a full scripting environment, whilst only having a small performance overhead[1]. From a development point of view, Lua was very useful as it allowed rapid prototyping to occur in the same environment, with the tested code being easily ported into C for a performance boost.

There are two ways to calculate a satellite's orbit - propagation and integration. Propagation uses an analytical approach to calculate the orbit, whereas, integration considers

[1]The performance overhead is not particularly in processing, which Lua, being stack based, is very fast at, but in Lua's memory management.

the forces on a satellite and then calculates the orbit through numerical integration. Propagation is considerably faster than integration, as any arbitrary time can be chosen, with the quality of integration requiring small time steps to be taken between the start and end points. However, propagation relies on an analytical solution being available that is also not too complex to implement. In addition to this, propagation is often difficult with non-conservative forces, as energy dissipation is not considered in the model. As drag is a non-conservative force and the simulator would need to be able to simulate a satellite that was constantly changing its attitude (i.e. rotating), there was no real benefit in choosing propagation, whilst integration would likely provide more accurate results - therefore integration was chosen.

Whilst two integrators were originally chosen, with the addition of the drag force, only one had a convergence that occurred at a small enough time step to be useful. These integrators were the leapfrog and a second order Runge-Kutta (Press et al., 2007). The leapfrog algorithm gets its name from calculating the position and velocity at different time steps (in a leapfrog pattern) and, despite being only first order, performs surprisingly well for conservative forces. However, when non-conservative forces are used, its performance decreases. The Runge-Kutta integrator is commonly used and performs 4 calculations per step - two trial mid-points and one trial endpoint, before calculating the end point of the step. Because of these extra points, it is significantly more capable of correctly integrating non-conservative forces, however, higher order methods can be used - at the cost of performance.

Alongside the effect of drag, the J2, J3 and J4 gravity corrections were also used. These correct a satellite's motion to include the effect of the Earth's shape. Whilst higher quality gravity models are available, the improvement over the J2, J3 and J4 corrections are very small and are much smaller than the expected drag forces, making their inclusion unnoticeable, whilst negatively affecting the simulator's performance.

The convergence of the simulation is shown in figure 7.3. This is the numerical derivative of the changes between each calculation. As the calculation converges, the line in the graph should become flat, indicating that no change is occurring between step sizes. However, this is rarely seen as the numerical precision of the calculation often affects the results. In the left hand side of the figure, for the leapfrog integrator, as the step size decreases the calculation improves. However, on the right hand side, not only are the same size changes occurring at a larger step size (indicating a shorter runtime), but the last point shows an increase in the change. This is where numerical precision is starting to impact the calculation and signifies that the 0.1 step size is the most converged for this configuration.

Figure 7.3: Drag convergence.

7.2 PCBsat dispersion

Whilst the torque experienced by a satellite is an interesting topic, this work is interested in the dispersion of the PCBsats. To model this, different angular configurations of the PCBsat were simulated, with the incident angle being fixed (i.e. torque was ignored). The results below look at only a rotation in pitch (around the x axis).

Figure 7.4 shows the overall effect of drag on the satellites by looking at the change in their radii, compared to a zero pitch angle. In particular, two different orbital altitudes are used. This shows that a $30°$ or lower pitch angle at $500\,\mathrm{km}$ altitude will result in a difference of below $9\,\mathrm{km}$ after 140 days, whereas, at $300\,\mathrm{km}$, the same separation occurs after approximately 20 days for a $30°$ pitch.

Figure 7.5 shows the position of the satellites during the first day, with the progression of time in the data being anti-clockwise from just below the $y = 0$ line, with figure 7.6 showing the first 100 days, with the progression in the same direction. Both of these figures show the large effect that the drag has on dispersion but also how the dispersion is largely in the xy plane, with significantly less dispersion occurring in the z axis. This is ideal for a multi-node constellation, as the communication and control is significantly easier if the satellites are more confined in the z axis. Figure 7.6 shows that after 100 days, at the worst combination ($0°$ pitch compared to $90°$) the separation is within $20\,\mathrm{km}$. Whilst this is a larger distance than the radio transceivers can communicate over, it is highly unlikely that this pitch would be chosen for any long period of time. Instead,

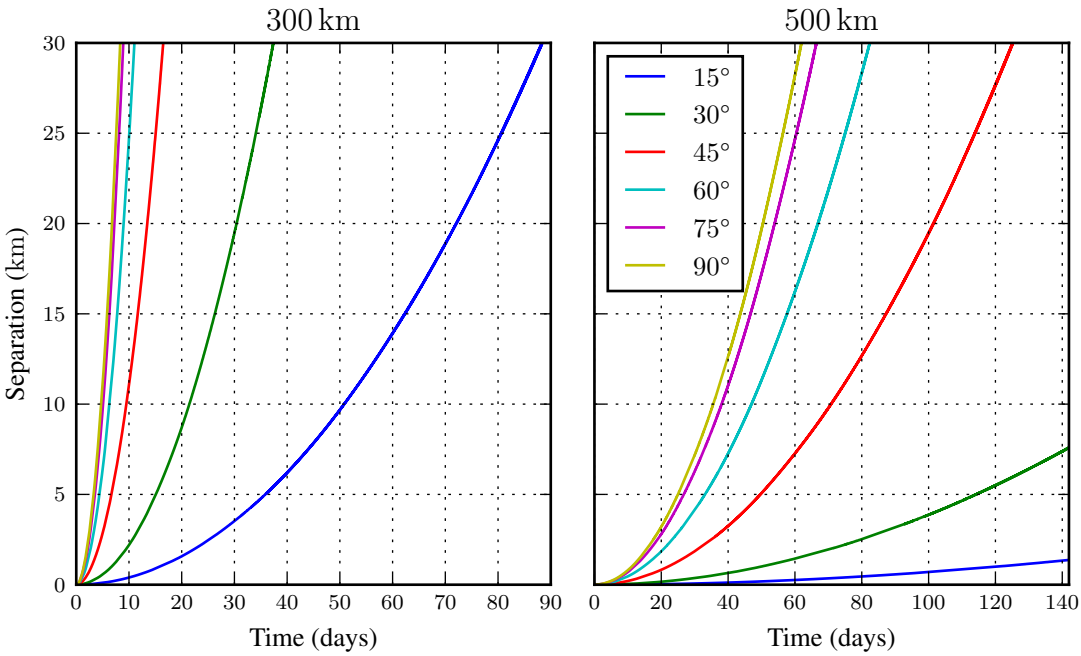


Figure 7.4: Radial separation of PCBsats at different pitch angles and altitudes.

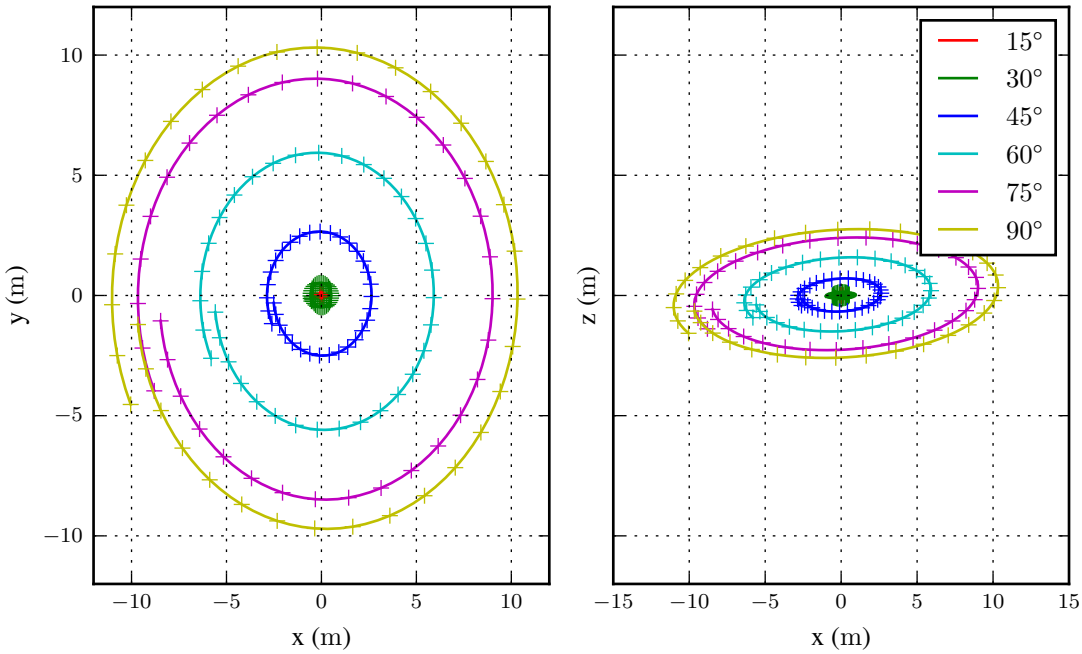


Figure 7.5: First day of orbit simulation at 500 km altitude.

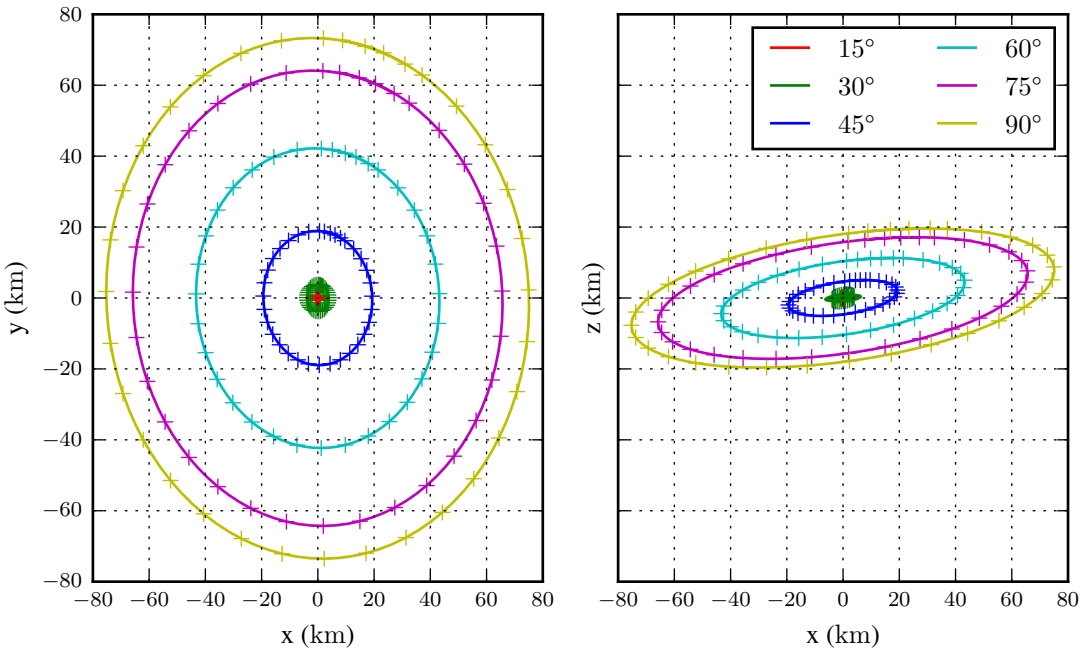


Figure 7.6: First 100 days of orbit simulation at $500\,\mathrm{km}$ altitude.

such a high pitch could be used to quickly aid the dispersion, before returning to a much smaller angle.

7.3 Summary

The results from the simulation show that dispersion in the xy plane is achievable using drag and the separation is less in the z axis. Whilst it does suggest that the concept mission has a lifetime of under 2 months at $300\,\mathrm{km}$ altitude, it also suggests that an altitude of $500\,\mathrm{km}$ could prolong the mission lifetime quite significantly. The word 'suggest' has been used here because figure 7.4 does not necessarily mean that a $300\,\mathrm{km}$ altitude mission would have a short life. These figures are comparisons with a satellite that has the smallest amount of drag possible applied to it (via orientation), so they represent the largest dispersion possible. If the satellites were allowed to change their pitch in the simulation, it would be possible for extreme satellites, in the xy plane, to become less extreme and become closer to the zero pitch satellite. However, they would still be dispersed in the z axis. This could be easily corrected for by the low pitch satellites using a high pitch for a limited time. Unfortunately, this is not a particularly easy design to demonstrate due to the runtime of the simulation.

Chapter 8

Conclusions

This works presents several novelties:

- Distributed GNSS receiver design
- Hybrid GNSS receiver design
- Open source implementation of a modular, extendible GNSS receiver
- Comparative FFT acquisitions
- Free molecular drag orbit simulation

The distributed GNSS receiver design, discussed in chapters 3 and 6, is able to reduce the power consumption by between $5.6\,\%$ and $13.3\,\%$ - which, in terms of the limited power budget of the PCBsat, and femto-satellites in general, is a substantial saving. Its novelty is split into two parts - first is the distributed aspect, where the navigation processor is moved to a different physical location, and the second is the use of a data caching and optimisation of the ephemerides, which makes the complete process energetically viable. Whilst there are technical limitations, the largest being the power efficiency of FPGAs, which improves with each product generation, the main limitation of this approach is the limited applications - as there are not many situations where a device containing a GNSS receiver does not need to know its location. However, the concept of a distributed GNSS receiver is not limited to space applications. There are many instances where remote sensing is required within the constraints of a restrictive power budget, such as on buoys in oceans and on atmospheric weather balloons, where this distributed design could prove to be as useful as it is for femto-satellites.

Following on from this, the hybrid GNSS receiver design combines a traditional receiver with the data storage and optimisation of the distributed receiver. This allows the receiver to function as a traditional receiver, where it is aware of its own location, whilst being able to store the required information to calculate its position at a later point in time, allowing it to utilise the more precise IGS ephemerides. This has a much wider range of applications than the distributed receiver design, as there are many applications,

particularly scientific, where being able to calculate a more accurate position, even with a substantial time delay, is useful.

By utilising a modular and extendible design for the GNSS receiver, individual modules can be easily exchanged, allowing the receiver to be tailored for specific usage cases. Whilst the current number of modules is small - allowing only the filters and loop discriminators to be changed - by making the design open source, it allows others to extend the design as they see fit. This is particularly advantageous for research, as it allows novelties to be built on an existing platform, rather than having to recreate the same base.

The FFT acquisition technique, described in section 5.1, is capable of detecting visible satellites that would otherwise be missed with a traditional FFT approach. This is achieved by performing multiple acquisition scans and comparing them to determine if the results are consistent with the typical movements of the broadcasting satellites and the receiver. Whilst this technique suffers from the same limitations as any FFT based acquisition - mainly the lack of precision in the results - it can be implemented alongside any existing FFT acquisition with only a small overhead.

The final novelty of this work is the derivation and implementation of the free molecular drag model - see appendix C and chapter 7. This considers a free molecular flow incident on a surface, with the flow able to rotate in all 3 axes, allowing the drag force to not only be correctly calculated but for it to exert a torque on the surface, correctly modelling drag induced rotations. This is implemented in a custom, open source orbit simulation which can be easily configured and controlled through a scripting language.

The key results of this research are the implementation of a distributed GPS receiver on an FPGA that can perform within the power budget of a femto-satellite; the distributed receiver design being not only feasible but vital in meeting the power constraints of the femto-satellite; and the distributed design, as shown by the drag simulation, being practical for the designed satellite mission and its target lifetime.

Future work

There are several areas which could be enhanced with further work. The current receiver design, whilst being modular, does not have many modules that can be exchanged and only supports the legacy GPS signals. This could be easily extended to support more modules, that are optimised for different applications, or different signals, either from different GNSS constellations or the new modernised GPS signals.

The receiver design utilises an SRAM based FPGA which is not optimised for low power usage, if the design was transitioned to an FPGA designed for low power or to an ASIC, the advantages of the distributed design would be considerably more. In addition to this, low power techniques could be investigated to reduce the power required by the design on the current FPGA.

The orbit simulation has a great potential for further work, from investigating different constellations to determining how to achieve particular flight patterns. But the greatest potential is in the simulation's ability to be used to test attitude control algorithms. This could be used to understand how a femto- or pico-satellite behaves in orbit, allowing the attitude algorithms, as well as the satellite's design, to be altered so that it performs as intended.

Appendix A

Occulting disc calculations

The PROBA-3 satellites are designed to use a $1.5\,\mathrm{m}$ occulting disc (Sephton et al., 2008), which is impractical for a pico-satellite. This section, therefore, considers a $10\,\mathrm{cm}$ occulting disc.

To maintain the same solid angle at the chronograph satellite, the angle between the centre of the occulting disc and its edge, as seen by the chronograph, must be the same. This apex angle can be described as,

$$\tan\theta = \frac{D}{2d}, \tag{A.1}$$

where D is the diameter of the occulting disc and d is the distance between the occulter and chronograph satellites. As ESA do not detail this apex angle, or the solid angle at the chronograph, we construct a ratio to calculate the separation required for the reduced size occulting disc,

$$\frac{D}{d} = \frac{D_r}{d_r},$$

where the subscript r is used for the reduced size disc. Calculating for d_r, using ESA's separation values of 150 and $250\,\mathrm{m}$, results in a separation distance for the reduced size disc of 10 and $16.67\,\mathrm{m}$, respectively.

For a larger separation distance, the solid angle will decrease and will therefore require an optical system with a greater resolving power. The angular resolution of a system can be found using the Rayleigh criterion,

$$\sin\theta = 1.22\frac{\lambda}{D_a}, \tag{A.2}$$

where θ is the angular resolution, λ is the wavelength of light and D_a is the diameter of the aperture[1]. As the minimum angle resolution is quite obviously determined by equation

[1]The factor of 1.22 in equation A.2, is for a circular aperture and is derived from the diffraction pattern of an Airy disc.

Table A.1: Minimum aperture diameter for the PROBA-3 optical system, at either end of the visible spectrum and the separation range.

d \ λ	390 nm	700 nm
150 m	95.2 µm	171 µm
250 m	159 µm	284 µm

Table A.2: Minimum aperture diameter for the reduced size occulting disc at several separation ranges at both extremes of the visible spectrum.

λ \ d	150 m	250 m	500 m	1 km	5 km	10 km
390 nm	1.43 mm	2.38 mm	4.76 mm	9.52 mm	47.6 mm	95.2 mm
700 nm	2.56 mm	4.27 mm	8.54 mm	17.1 mm	85.4 mm	171 mm

A.1, we can calculate the minimum aperture diameter as,

$$D_a = 1.22\frac{\lambda}{\sin\theta} = 1.22\frac{\lambda}{\sin\left(\arctan\frac{D}{2d}\right)}. \tag{A.3}$$

As there is very little information available of the PROBA-3 optical system, we make the assumption that it is operating within the visible part of the spectrum, that is 390 to $700\,\mathrm{nm}$. Using this assumption, and equation A.3, the PROBA-3 optical system must have an aperture that is at least 78 to $233\,\mu\mathrm{m}$ in diameter (see table A.1). This is small enough that the optical system could consist of a low pixel count CCD, without any lenses. For this, the minimum aperture diameter would be the minimum pixel size[2], with a 100 by 100 pixel CCD measuring between 7.8 and $23.3\,\mathrm{mm}$ in both width and height. However, it is highly likely that the optical system is much more complicated than this.

Using equation A.3, the minimum aperture diameter for the reduced size disc is fifteen times larger than that of the original system (see table A.2). For small separations, an optic-less system could still be used, however, for larger separations this becomes increasing difficult to accommodate and thus requiring lenses. Additionally, for pico-satellites, an optical system with an aperture of 50 to $170\,\mathrm{mm}$ would be difficult to integrate, both in terms of size and mass, resulting in the largest practical separation distance for the reduced size occulting disc of approximately $1\,\mathrm{km}$.

[2]Assuming the use of square pixels.

Appendix B

Eclipse shadow angle derivation

From simple trigonometry, it is simple to approximate that the eclipse angle will be,

$$\theta = 2 \arctan\left(\frac{r_e}{r_e + r_s}\right),$$

where r_e is the radius of the Earth and r_s is the altitude of the satellite. However, this approximation assumes that the Earth is a flat disc, with the incident radiation being normal to it. To account for a spherical Earth, the equation for the eclipse angle becomes,

$$\theta = 2 \arcsin\left(\frac{r_e}{r_e + r_s}\right). \tag{B.1}$$

This is not as obvious, but essentially moves the right angle from the Earth's centre to its surface, due to the different geometry. As the incident radiation is not normal, there is a small correction that has to be considered that will reduce the angle of eclipse. This correction is, again, simpler to understand if flat discs are considered and is shown in figure B.1,

$$\theta_c = 2 \arctan\left(\frac{r_{sun} - r_e}{d}\right),$$

where r_{sun} is the radius of the Sun and d is the average distance between the Sun and Earth, by definition, $1\,\mathrm{AU}$. This correction is the ratio of the difference in size between

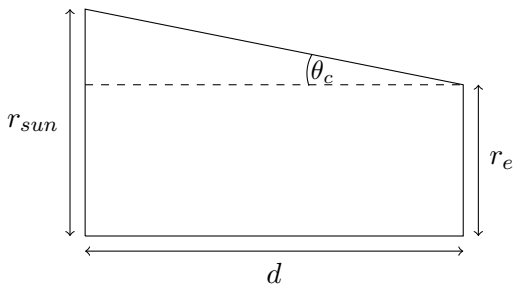


Figure B.1: Diagram showing the small correction angle that is required due to the different sizes of the Earth and Sun.

the Sun and the Earth, and the distance that separates them. Again, if a spherical Sun and Earth are considered, this becomes,

$$\theta_c = 2 \arcsin \left(\frac{r_{sun} - r_e}{d} \right), \tag{B.2}$$

and is only a small correction, at less than $5\,\mathrm{mrad}$.

The total eclipse angle is, therefore, a combination of equations B.1 and B.2,

$$\theta = 2 \arcsin \left(\frac{r_e}{r_e + r_s} - \frac{r_{sun} - r_e}{d} \right).$$

Appendix C

Derivation of drag model

To calculate the effect of drag acting on a surface in a free molecular flow, we need to consider the stress on the surface (that is, the force per unit area). This is resolved into two components, pressure - the stress normal to the surface - and shear - the stress tangential to the surface. To do this, we start from the molecular density function. Whilst this can be derived fairly easily using Maxwell Boltzmann statistics, for brevity we take it from Schaaf and Chambré (1961, p. 12),

$$f = \frac{\rho}{m\left(2\pi RT\right)^{\frac{3}{2}}} \exp\left\{-\frac{\left(v_1-u_1\right)^2+\left(v_2-u_2\right)^2+\left(v_3-u_3\right)^2}{2RT}\right\}, \tag{C.1}$$

where ρ, m and T are the density, molecular mass and temperature of the flow, respectively; R is the molar gas constant, v_n and u_n are the absolute and gas velocities in the three axes. In addition to this, we derive the number density, that is the number of particles, per area, that are incident on the surface.

These derivations use four standard integrals, which are:

$$\int_{-\infty}^{\infty} e^{-\frac{(x-b)^2}{a}} dx = \sqrt{\pi a} \tag{C.2}$$

$$\int_{-\infty}^{\infty} x e^{-\frac{(x-b)^2}{a}} dx = \sqrt{\pi a} b \tag{C.3}$$

$$\int_{0}^{\infty} x e^{-\frac{(x-b)^2}{a}} dx = \frac{a}{2} e^{-\frac{b^2}{a}} + \frac{\sqrt{\pi a} b}{2}\left[1+\operatorname{erf}\left(\frac{b}{\sqrt{a}}\right)\right] \tag{C.4}$$

$$\int_{0}^{\infty} x^2 e^{-\frac{(x-b)^2}{a}} dx = \frac{ab}{2} e^{-\frac{b^2}{a}} + \frac{\sqrt{\pi a}}{4}\left(a+2b^2\right)\left[1+\operatorname{erf}\left(\frac{b}{\sqrt{a}}\right)\right] \tag{C.5}$$

C.1 Number density

The number of particles incident on a surface is dependant on the molecular density of the flow, described by f, and the velocity of the flow normal to the surface, v_1. The density of this number is quite obviously dependant on the size of the surface,

$$\frac{N_i}{dA} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty} f v_1 dv_1 dv_2 dv_3. \tag{C.6}$$

Using equation C.1, this expands to,

$$\frac{N_i}{dA} = \frac{\rho}{m\left(2\pi RT\right)^{\frac{2}{3}}}\int_0^{\infty} v_1 e^{-\frac{(v_1-u_1)^2}{2RT}} dv_1 \int_{-\infty}^{\infty} e^{-\frac{(v_2-u_2)^2}{2RT}} dv_2 \int_{-\infty}^{\infty} e^{-\frac{(v_3-u_3)^2}{2RT}} dv_3.$$

Both of the integrals over v_2 and v_3 have the same form - that of the standard integral in equation C.2,

$$\frac{N_i}{dA} = \frac{\rho}{m\sqrt{2\pi RT}}\int_0^{\infty} v_1 e^{-\frac{(v_1-u_1)^2}{2RT}} dv_1.$$

The remaining integral is of the form of equation C.4, resulting in

$$\begin{aligned}\frac{N_i}{dA} &= \frac{\rho}{m\sqrt{2\pi RT}}\left(RTe^{-\frac{u_1^2}{2RT}} + \frac{\sqrt{2\pi RT}u_1}{2}\left[1+\operatorname{erf}\left(\frac{u_1}{\sqrt{2RT}}\right)\right]\right)\\ &= \frac{\rho}{m}\left(\sqrt{\frac{RT}{2m}}e^{-\frac{u_1^2}{2RT}} + \frac{u_1}{2}\left[1+\operatorname{erf}\left(\frac{u_1}{\sqrt{2RT}}\right)\right]\right).\end{aligned} \tag{C.7}$$

C.2 Pressure

The pressure exerted onto the surface by the flow is dependant on the momentum and number of particles incident with the surface. This makes its form not too dissimilar from that of the number density (equation C.6),

$$p_i = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty} m v_1^2 f dv_1 dv_2 dv_3,$$

where m is the mass of the molecules and v_n and f are as in equation C.6. Using the standard integrals of equation C.2 and C.5, this is reduced to,

$$p_i = \frac{\rho}{\sqrt{2\pi RT}}\int_0^{\infty} v_1^2 e^{-\frac{(v_1-u_1)^2}{2RT}} dv_1,$$

and

$$p_i = \frac{\rho}{\sqrt{2\pi RT}}\left\{RTu_1 e^{-\frac{u_1^2}{2RT}} + \frac{\sqrt{2\pi RT}}{4}\left(2RT+2u_1^2\right)\left[1+\operatorname{erf}\left(\frac{u_1}{\sqrt{2RT}}\right)\right]\right\},$$

respectively. This can be simplified to,

$$p_i = \rho RT \left\{ \frac{u_1}{\sqrt{2\pi RT}} e^{-\frac{u_1^2}{2RT}} + \left(\frac{1}{2} + \frac{u_1^2}{2RT} \right) \left[1 + \text{erf} \left(\frac{u_1}{\sqrt{2RT}} \right) \right] \right\}. \tag{C.8}$$

p_i is the pressure incident on the surface, however, it is not the total pressure felt by the surface as there are a certain amount of reflections. These reflections have a pressure of,

$$p_w = \frac{\rho}{2} \sqrt{2\pi RT_w} N_i,$$

where T_w is the temperature of the surface. The total pressure is then found through the use of a reflection coefficient, σ',

$$p = (2 - \sigma')\, p_i + \sigma' p_w, \tag{C.9}$$

with typical values for σ' being between 0.8 and 1.

C.3 Shear

The shear exerted onto the surface by the flow is effectively the same as the pressure, with the exception that the momentum of the particles is in the direction of v_2 instead of v_1,

$$\tau_i = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{0}^{\infty} m v_1 v_2 f dv_1 dv_2 dv_3.$$

This, using equation C.2 and C.3, reduces to,

$$\tau_i = \frac{\rho u_2}{\sqrt{2\pi RT}} \int_0^{\infty} v_1 e^{-\frac{(v_1 - u_1)^2}{2RT}} dv_1,$$

which, using equation C.4, reduces further to,

$$\tau_i = \frac{\rho u_2}{\sqrt{2\pi RT}} \left(RT e^{-\frac{u_1^2}{2RT}} + \frac{\sqrt{2\pi RT}}{2} u_1 \left[1 + \text{erf} \left(\frac{u_1}{\sqrt{2RT}} \right) \right] \right).$$

Simplifying and adding in the reflection coefficient results in,

$$\tau = \sigma \rho u_2 \left(\sqrt{\frac{RT}{2\pi}} e^{-\frac{u_1^2}{2RT}} + \frac{u_1}{2} \left[1 + \text{erf} \left(\frac{u_1}{\sqrt{2RT}} \right) \right] \right). \tag{C.10}$$

Appendix D

Source code

A fairly large amount of computer code and hardware design has been completed for, and is used in, this work. Rather than include it here, links are listed below to the source code repositories that contain the various parts.

SD card controller https://gitlab.com/6thimage/sd_controller

A finite state machine based SD card controller, which presents a 512 byte RAM interface for reading and writing data to the card. See section 4.2.2.

Baseband recorder https://gitlab.com/6thimage/baseband_recorder

A data storage tool designed for use on an FPGA to record the digital baseband directly from the RF front-end to an SD card. See section 4.3.

Baseband reader https://gitlab.com/6thimage/baseband_reader

A tool designed to play back the recorded baseband data (stored on an SD card). As the baseband is recorded at a frequency that cannot be directly generated by the FPGA (as the two clocks are not integer multiples of a common frequency), the reader plays the baseband data back at a faster speed - $16.667\,\mathrm{MHz}$ instead of $16.368\,\mathrm{MHz}$. See section 4.4.

Software GPS receiver https://gitlab.com/6thimage/soft_gps_c

A software implementation of a GPS receiver, written in the C programming language. This also includes the fixed point version of the receiver. See chapter 5.

GPS tracking channel https://gitlab.com/6thimage/gps_channel

Verilog implementation of a GPS tracking channel, including the frame decoder. See sections 6.3 and 6.4.

FFT acquisition module https://gitlab.com/6thimage/fft_module

Verilog implementation of the FFT acquisition module, utilising Xilinx's FFT IP. See section 6.2.

GPS SoC https://gitlab.com/6thimage/cm0_gps_soc

SoC design for the receiver, based around the AHB-Lite bus and using the ARM Cortex-M0 DS processor (which, due to its proprietary nature, is not included). See section 6.1.

Bibliography

Aachen University of Applied Sciences. COMPASS-1 CubeSat Project, 2013. http://www.raumfahrt.fh-aachen.de/compass-1/home.htm, [accessed 03/05/2017].

ABRACON Corporation. Low frequency cylindrical watch crystal, 2010. http://www.abracon.com/Resonators/AB15T.pdf, [accessed 03/05/2017].

Airbus. MosaicGNSS receiver, 2017a. https://spaceequipment.airbusdefenceandspace.com/avionics/gnss-receivers/mosaic/, [accessed 03/05/2017].

Airbus. A380 specifications, 2017b. http://www.airbus.com/aircraftfamilies/passengeraircraft/a380family/innovation/, [accessed 03/05/2017].

S. Allgeier. High Fidelity Nonconservative Force Models for Spacecraft. Seminar at the University of Leicester, Feb. 2013.

J. E. Allnutt. *Satellite-to-ground Radiowave Propagation*. The Institute of Engineering and Technology, 2nd edition, 2011. ISBN 978-1-84919-150-0.

Amateur Radio station PE0SAT. First-Move, 2016. http://www.pe0sat.vgnet.nl/satellite/cube-nano-picosats/first-move/, [accessed 04/05/2017].

AMBER wireless GmbH. 869 MHz Ultra Long Range Radio Module, 2014. https://www.amber-wireless.com/en/amb8636.html, [accessed 04/05/2017].

ASTRA. ASTRA GAMMA datasheet, 2015. http://www.astraspace.net/wp-content/uploads/2015/11/ASTRA_GAMMA-DataSheet.rev2_.pdf, [accessed 04/05/2017].

Atmel Corporation. ATmega128L - 8-bit Atmel Microcontroller with 128kBytes In-System Programmable Flash, 2011. http://www.atmel.com/Images/doc2467.pdf, [accessed 03/05/2017].

Atmel Corporation. AT86RF212B, 2014. http://www.atmel.com/devices/AT86RF212B.aspx, [accessed 05/09/2014].

Auburn University. AubieSat-1 mission is a success, July 2012. http://auburn.edu/cosam/news/articles/2012/07/aubiesat-1-mission-is-a-success.htm, [accessed 03/05/2017].

J. Axelson. *USB Complete: The Developer's Guide*. Complete Guides series. Lakeview Research, 5th edition, 2015. ISBN 978-1-931448-29-1.

R. Balthazor. MESA sensor specification and requirements. Personal communication, Jan. 2013.

R. Balthazor, M. McHarg, C. Godbold, D. Barnhart, and T. Vladimirova. Distributed space-based ionospheric multiple plasma sensor networks. In *2009 IEEE Aerospace conference*, pages 1–10, Mar. 2009. doi:10.1109/AERO.2009.4839357.

D. J. Barnhart. *Very Small Satellite Design for Space Sensor Networks*. PhD thesis, University of Surrey, June 2008.

D. J. Barnhart, T. Vladimirova, M. N. Sweeting, R. L. Balthazor, L. C. Enloe, L. H. Krause, T. J. Lawrence, M. G. Mcharg, J. C. Lyke, J. J. White, and A. M. Baker. Enabling space sensor networks with PCBSat. In *Proceedings of the 21st Annual Conference on Small Satellites*, number SSC07-IV-4, Logan Utah, USA, Aug. 2007. Utah State University.

D. J. Barnhart, T. Vladimirova, A. M. Baker, and M. N. Sweeting. A low-cost femtosatellite to enable distributed space missions. *Acta Astronautica*, 64(11-12):1123–1143, 2009. doi:10.1016/j.actaastro.2009.01.025.

R. Bedington, D. Kataria, and D. Walton. Using a CCD for the direct detection of electrons in a low energy space plasma spectrometer. In *The 9th International Conference on Position Sensitive Detectors*, Sept. 2011. doi:10.1088/1748-0221/7/01/C01079.

Boeing. Technical specifications - 747 classics, 2013. http://web.archive.org/web/20150304174244/http://www.boeing.com/boeing/commercial/747family/pf/pf_classics.page, [accessed 04/05/2017].

V. Bothmer and I. A. Daglis. *Space Weather - Physics and Effects*. Springer, Verlag, 2007.

J. Bouwmeester and J. Guo. Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, 67(7-8):854–862, 2010. doi:10.1016/j.actaastro.2010.06.004.

CNES. TARANIS, 2017. https://taranis.cnes.fr/en/TARANIS/index.htm, [accessed 04/05/2017].

Council of the European Union. Council Regulation (EC) No 428/2009, May 2009. Category 7, Item 7A105, http://trade.ec.europa.eu/doclib/docs/2009/june/tradoc_143390.pdf, [accessed 03/05/2017].

CubeSatShop. GPS receiver, 2013. http://www.cubesatshop.com/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=105&category_id=7&option=com_virtuemart&Itemid=69&vmcchk=1&Itemid=69, [accessed 28/02/2013].

M. Czech, A. Fleischner, and U. Walter. A first-MOVE in satellite development at the TU-München. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellite Missions for Earth Observation: New Developments and Trends*, pages 235–245. Springer, 1st edition, 2010. ISBN 9783642035005.

M. Czerski and J. Clayton. Wishbone SD Card Controller, Sept. 2013. http://opencores.org/project,sd_card_controller, [accessed 04/05/2017].

S. de Jong, G. T. Aalben, and J. Bouwmeester. Improved command and data handling system for the Delfi-N3XT nanosatellite. In *59th International astronautical congress, Glasgow*, 2008.

O. de La Beaujardière, J. M. Retterer, R. F. Pfaff, P. A. Roddy, C. Roth, W. J. Burke, Y. J. Su, M. C. Kelley, R. R. Ilma, G. R. Wilson, L. C. Gentile, D. E. Hunton, and D. L. Cooke. C/NOFS observations of deep plasma depletions at dawn. *Geophysical Research Letters*, 36(18):L00C06, Aug. 2009. ISSN 0094-8276. doi:10.1029/2009GL038884.

Delft University of Technology. Delfi Space, 2013. http://delfispace.nl/, [accessed 03/05/2017].

Digi International Inc. XBee ZB, 2014. http://web.archive.org/web/20150602202028/http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#specs, [accessed 04/05/2017].

Ecole Polytechnique Federale De Lausanne. Swiss Cube, 2013. http://swisscube.epfl.ch/, [accessed 03/05/2017].

A. Edvardsson. sd card controller, Mar. 2016. http://opencores.org/project,sdcard_mass_storage_controller, [accessed 04/05/2017].

EE Times. EE slashes CPU design cost, May 2016. http://www.eetimes.com/document.asp?doc_id=1330568, [accessed 04/05/2017].

A. El-Rabbany. *Introduction to GPS: the global positioning system*. Artech House, Boston, 2002. ISBN 1-58053-183-0.

Electronic Code of Federal Regulations. Part 121: The United States Munitions List, 2013. Category XV: Spacecraft Systems and Associated Equipment, http://www.ecfr.gov/cgi-bin/text-idx?node=22:1.0.1.13.58, [accessed 03/05/2017].

C. L. Enloe, L. H. Krause, R. K. Haaland, T. T. Patterson, C. E. Richardson, C. C. Lazidis, and R. G. Whiting. Miniaturized electrostatic analyzer manufactured using photolithographic etching. *Review of Scientific Instruments*, 74(3):1192–1195, 2003. doi:10.1063/1.1540715.

ESA. Galileo: Satellite launches, 2011. http://web.archive.org/web/20150702130606/http://ec.europa.eu/enterprise/policies/satnav/galileo/satellite-launches/index_en.htm, [accessed 04/05/2017].

ESA. About Proba-3, 2013. http://www.esa.int/Our_Activities/Technology/Proba_Missions/About_Proba-3, [accessed 03/05/2017].

ESA. Swarm facts and figures, 2016. http://web.archive.org/web/20160412061443/http://www.esa.int/Our_Activities/Observing_the_Earth/The_Living_Planet_Programme/Earth_Explorers/Swarm/Overview2, [accessed 04/05/2017].

ESA. MicroSCOPE, 2016a. https://directory.eoportal.org/web/eoportal/satellite-missions/m/microscope, [accessed 04/05/2017].

ESA. TARANIS (Tool for the Analysis of RAdiations from lightNIngs and Sprites), 2016b. https://directory.eoportal.org/web/eoportal/satellite-missions/t/taranis, [accessed 04/05/2017].

S. Fielding. SD/MMC Controller, Nov. 2014. http://opencores.org/project,spimaster, [accessed 04/05/2017].

C. Fossa, R. Raines, G. Gunsch, and M. Temple. An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system. In *Proceedings of the IEEE 1998 National Aerospace and Electronics Conference (NAECON)*, pages 152 –159, 1998. doi:10.1109/NAECON.1998.710110.

FPT University. FSpace laboratory, 2013. http://web.archive.org/web/20130518232536/http://fspace.edu.vn/, [accessed 04/05/2017].

M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on “Program Generation, Optimization, and Platform Adaptation”.

General Dynamics Mission Systems. Explorer GPS spaceborne receiver, 2013a. https://gdmissionsystems.com/space/space-electronics/gps-receivers/explorer/, [accessed 04/05/2017].

General Dynamics Mission Systems. Monarch GPS spaceborne receiver, 2013b. https://gdmissionsystems.com/space/space-electronics/gps-receivers/monarch/, [accessed 04/05/2017].

General Dynamics Mission Systems. Viceroy GPS spaceborne receiver, 2013c. https://gdmissionsystems.com/space/space-electronics/gps-receivers/viceroy/, [accessed 04/05/2017].

Global Positioning Systems Directorate. *Interface Specification - IS-GPS-200H*. National Coordination Office for Space-Based Positioning, Navigation, and Timing, Mar. 2014. URL http://www.gps.gov/technical/icwg/IS-GPS-200H.pdf.

G. Griffin. DSP Trick: Magnitude Estimator, Oct. 1999. https://dspguru.com/dsp/tricks/magnitude-estimator/, [accessed 04/05/2017].

I. Griffiths, T. Vladimirova, and M. Warrington. Utilising distributed processing to reduce the power consumption of multiple gnss receivers in a satellite constellation. In *65th International Astronautical Congress, Toronto*, 2014.

M. Grondin, M. Belasic, L. Ries, J.-L. Issler, P. Bataille, L. Jobey, and G. Richard. A new operational low cost GNSS software receiver for microsatellites. In *5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, Dec. 2010. doi:10.1109/NAVITEC.2010.5707991.

J. R. Guasch, P. Silvestrin, M. Aguirre, and L. Masotti. Navigation needs for ESA's Earth observation missions. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellite Missions for Earth Observation: New Developments and Trends*, pages 439–447. Springer, 1st edition, 2010. ISBN 9783642035005.

Gunter's Space Page. Gps-3 (navstar-3), July 2016. http://space.skyrocket.de/doc_sdat/navstar-3.htm, [accessed 04/05/2017].

R. Haagmans, R. Bock, and H. Rider. *Swarm: ESA's Magnetic Field Mission*. ESA, 2012. ISBN 978-92-9221-051-9. URL http://esamultimedia.esa.int/multimedia/publications/BR-302/offline/download.pdf.

B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer Wien, New York, 2008. ISBN 978-3-211-73017-1.

S. Hojun. Open Source Satellite Initiative, 2013. http://opensat.cc/, [accessed 03/05/2017].

M. Hollreiser. Advanced GPS/GLONASS ASIC (AGGA2), Mar. 2001. http://microelectronics.esa.int/presentation/AGGA2.pdf, [accessed 03/05/2017].

J. V. D. Horst and J. Noble. Task allocation in dynamic networks of satellites. In *IJCAI Workshop on Artificial Intelligence in Space*, July 2011. URL http://eprints.soton.ac.uk/272729/.

C.-S. Huang, O. de La Beaujardière, P. A. Roddy, D. E. Hunton, R. F. Pfaff, C. E. Valladares, and J. O. Ballenthin. Evolution of equatorial ionospheric plasma bubbles and formation of broad plasma depletions measured by the C/NOFS satellite during deep solar minimum. *Journal of Geophysical Research*, 116(A3):A03309, Mar. 2011. ISSN 0148-0227. doi:10.1029/2010JA015982.

Inside GNSS. Russia approves CDMA signals for GLONASS, discussing common signal design, 2008. http://www.insidegnss.com/node/648, [accessed 03/05/2017].

International GNSS Service. IGS products, 2014. http://www.igs.org/products, [accessed 04/05/2017].

IQD Frequency Products. CFPT-126 oscillator specification, 2012. http://www.iqdfrequencyproducts.com/products/details/cfpt-126-5-03.pdf, [accessed 03/05/2017].

Istanbul Technical University. ITU Space Systems Design and Testing Laboratory, 2013. http://web.archive.org/web/20121121044250/http://usl.itu.edu.tr/en/projects/itupsat1, [accessed 04/05/2017].

E. D. Kaplan and C. J. Hegarty, editors. *Understanding GPS: Principles and Applications*. Artech House, Boston, 2nd edition, 2006. ISBN 1-58053-894-0.

M. C. Kelley. *The Earth's Ionosphere: plasma physics and electrodynamics*. Academic Press, 2nd edition, 2009. ISBN 9780120884254.

L. H. Krause, C. Enloe, R. Haaland, and P. Golando. Microsatellite missions to conduct midlatitude studies of equatorial ionospheric plasma bubbles. *Advances in Space Research*, 36(12):2474–2479, 2005. doi:10.1016/j.asr.2004.03.020.

La Sapienza University of Rome. Gaussteam, 2013. http://www.gaussteam.com/, [accessed 03/05/2017].

S. Magdaleno, M. Herraiz, and B. de la Morena. Characterization of equatorial plasma depletions detected from derived GPS data in south america. *Journal of Atmospheric and Solar-Terrestrial Physics*, 74:136–144, 2012. doi:10.1016/j.jastp.2011.10.014.

Z. Manchester. KickSat – your personal spacecraft in space!, Oct. 2011. https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/description, [accessed 03/05/2017].

Z. Manchester. KickSat is alive!, Apr. 2014a. https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/posts/814463, [accessed 03/05/2017].

Z. Manchester. KickSat has reentered, May 2014b. https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/posts/843807, [accessed 03/05/2017].

Z. Manchester. KickSat's current status, May 2014c. https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/posts/831509, [accessed 03/05/2017].

Z. Manchester, M. Peck, and A. Filo. KickSat: A crowd-funded mission to demonstrate the worlds smallest spacecraft. In *Proceedings of the 27th Annual AIAA/USU Conference on Small Satellites*, 2013.

Maxim Integrated. MAX2769 Universal GPS Receiver, June 2010. http://datasheets.maximintegrated.com/en/ds/MAX2769.pdf, [accessed 03/05/2017].

J. Merayo, J. Jorgensen, E. Friis-Christensen, P. Brauer, F. Primdahl, P. Jorgensen, T. Allin, and T. Denver. The Swarm magnetometry package. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellites for Earth Observation*, pages 143–151. Springer, 1st edition, 2008.

Micron Technology Inc. MT29F4G01AAADDHC, 2016. https://www.micron.com/parts/nand-flash/serial-nand/mt29f4g01aaaddhc, [accessed 04/05/2017].

Microsemi Corporation. SmartFusion2 System-on-Chip FPGAs Product Brief, 2016. http://www.microsemi.com/document-portal/doc_download/132721-pb0115-smartfusion2-system-on-chip-fpgas-product-brief, [accessed 04/05/2017].

O. Montenbruck and S. D'Amico. *GPS Based Relative Navigation*, pages 185–223. Springer New York, New York, NY, 2013. ISBN 978-1-4614-4541-8. doi:10.1007/978-1-4614-4541-8_5. URL http://dx.doi.org/10.1007/978-1-4614-4541-8_5.

O. Montenbruck, M. Markgraf, M. Garcia-Fernandez, and A. Helm. GPS for microsatellites - status and persepectives. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellites for Earth Observation*, pages 165–174. Springer, 1st edition, 2008.

NASA. Landsat 4, 2013. https://landsat.gsfc.nasa.gov/landsat-4/, [accessed 04/05/2017].

NASA. Plunging into the ionosphere: Satellites last days improve orbital decay predictions, Dec. 2015a. https://www.nasa.gov/press-release/goddard/plunging-into-the-ionosphere-satellite-s-last-days-improve-orbital-decay-predictions, [accessed 04/05/2017].

NASA. The drag equation, May 2015b. https://www.grc.nasa.gov/www/k-12/airplane/drageq.html, [accessed 04/05/2017].

National Coordination Office. New civil signals, 2013. http://www.gps.gov/systems/gps/modernization/civilsignals/, [accessed 03/05/2017].

NovAtel Inc. OEM4-G2L, 2006a. http://www.novatel.com/assets/Documents/Papers/oem4g2l.pdf, [accessed 03/05/2017].

NovAtel Inc. NovAtel ships first production OEMV receivers, Apr. 2006b. http://web.archive.org/web/20060510030519/http://www.novatel.com/about_us/news/20060405.htm, [accessed 04/05/2017].

NovAtel Inc. OEMV-1 Series, 2011. http://www.novatel.com/assets/Documents/Papers/OEMV-1_Series.pdf, [accessed 03/05/2017].

NovAtel Inc. End of Life for the complete OEMV-1, OEMV-1G, OEMV-1GR and OEMV-2 family of receivers, May 2012. http://www.novatel.com/about-us/news-releases/news-releases-2012/end-of-life-for-the-complete-oemv-1-oemv-1g-oemv-1gr-and-oemv-2-family-of-receivers/, [accessed 03/05/2017].

J. M. Picone, A. E. Hedin, D. P. Drob, and A. C. Aikin. NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, 107(A12), Dec. 2002.

Politecnico di Torino. e-st@r, 2013. http://web.archive.org/web/20120829111838/http://areeweb.polito.it/ricerca/E-STAR/, [accessed 04/05/2017].

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007. ISBN 978-0-521-88068-8.

M. L. Psiaki. Nanosatellite Attiude Stabilization Using Passive Aerodynamics and Active Magnetic Torquing. *Journal of Guidance, Control and Dynamics*, 27(3):347–355, 2004.

Qualcomm Technologies Inc. IZat, 2016. https://www.qualcomm.com/products/izat, [accessed 04/05/2017].

Romanian Space Agency. GOLIAT, 2013. http://www.goliat.ro/, [accessed 03/05/2017].

J. Rosello, P. Silvestrin, R. Weigand, S. d'Addio, A. Garcia Rodriguez, and G. Lopez Risueno. Next generation of ESA's GNSS receivers for earth observation satellites. In *6th ESA Workshop on Satellite Navigation Technologies (NAVITEC)*, 2012. doi:10.1109/NAVITEC.2012.6423104.

RUAG. Innovative GNSS navigation receiver, 2013. http://www.ruag.com/space/Products/Digital_Electronics_for_Satellites_Launchers/Signal_Processing/Innovative_GNSS_Navigation_Receiver_, [accessed 28/02/2013].

W. W. Saylor, C. K. Smaagard, C. N. Nordby, and C. D. J. Barnhart. New scientific capabilities enabled by autonomous constellations of smallsats. In *21st Annual AIAA/USU Conference on Small Satellites*, pages SSC07–II–7, 2007.

S. A. Schaaf and P. L. Chambré. *Flow of Rarefied Gases*. Princeton University Press, Princeton, New Jersey, 1961.

SD Association. SD Host Controller Simplified Specification, Feb. 2007. https://www.sdcard.org/developers/overview/host_controller/simple_spec/Simplified_SD_Host_Controller_Spec.pdf, [accessed 04/05/2017].

D. Selva and D. Krejci. A survey and assessment of the capabilities of cubesats for Earth observation. *Acta Astronautica*, 74:50–68, 2012. doi:10.1016/j.actaastro.2011.12.014.

T. Sephton, A. Wishart, K. Strauch, and F. Teston. EO formation flying applications for small satellite missions. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellites for Earth Observation*, pages 153–162. Springer, 1st edition, 2008.

SetiQuest. SetiQuest data links, July 2015. http://setiquest.org/wiki/index.php/SetiQuest_Data_Links, [accessed 04/05/2017].

Sonnet Software Inc. Sonnet lite, 2013. http://www.sonnetsoftware.com/products/lite/, [accessed 03/05/2017].

SpaceQuest. GPS-12-V1 GPS recevier, 2013. http://web.archive.org/web/20140801211406/http://www.spacequest.com/products/GPS-12-V1.pdf, [accessed 04/05/2017].

S. C. Spangelo, M. W. Bennett, D. C. Meinzer, A. T. Klesh, J. A. Arlas, and J. W. Cutler. Design and implementation of the GPS subsystem for the Radio Aurora eXplorer. *Acta Astronautica*, 87:127–138, Feb. 2013. doi:10.1016/j.actaastro.2012.12.009.

Spectrolab Inc. 26.8% Improved Triple Junction (ITJ) Solar Cells, 2008. http://www.spectrolab.com/DataSheets/TNJCell/tnj.pdf,[accessed 03/05/2017].

Spectrolab Inc. 28.3% Ultra Triple Junction (UTJ) Solar Cells, 2010a. http://www.spectrolab.com/DataSheets/cells/PV%20UTJ%20Cell%205-20-10.pdf,[accessed 03/05/2017].

Spectrolab Inc. 29.5% NeXt Triple Junction (XTJ) Solar Cells, 2010b. http://www.spectrolab.com/DataSheets/cells/PV%20XTJ%20Cell%205-20-10.pdf,[accessed 03/05/2017].

SSTL. SGR-05P space GPS receiver, 2013a. http://www.sstl.co.uk/Products/Subsystems/Navigation/SGR-05P-Space-GPS-Receiver, [accessed 03/05/2017].

SSTL. SGR-05U space GPS receiver, 2013b. http://www.sstl.co.uk/Products/Subsystems/Navigation/SGR-05U-Space-GPS-Receiver, [accessed 03/05/2017].

SSTL. SGR-07 space GPS receiver, 2013c. http://www.sstl.co.uk/Products/Subsystems/Navigation/SGR-07-Space-GPS-Receiver, [accessed 03/05/2017].

SSTL. SGR-10 space GPS receiver, 2013d. http://www.sstl.co.uk/Products/Subsystems/Navigation/SGR-10-Space-GPS-Receiver, [accessed 03/05/2017].

SSTL. SGR-20 space GPS receiver, 2013e. http://www.sstl.co.uk/Products/Subsystems/Navigation/SGR-20-Space-GPS-Receiver, [accessed 03/05/2017].

Stanford University. QuakeSat Nano-Satellite, 2013. http://www.quakefinder.com/research/quakesat-ssite/, [accessed 03/05/2017].

E. Stavinov. A Practical Parallel CRC Generation Method. *Circuit Cellar*, (234), Jan. 2010. URL http://outputlogic.com/my-stuff/circuit-cellar-january-2010-crc.pdf.

STMicro Electronics. STM32F405xx, STM32F407xx Datasheet, June 2013a. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf, [accessed 04/05/2017].

STMicro Electronics. STM32L15xx6/8/B Datasheet, Nov. 2013b. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00277537.pdf, [accessed 04/05/2017].

H. Takahashi, M. Taylor, J. Sobral, A. Medeiros, D. Gobbi, and D. Santana. Fine structure of the ionospheric plasma bubbles observed by the OI 6300 and 5577 airglow images. *Advances in Space Research*, 27(6-7):1189–1194, 2001. doi:10.1016/S0273-1177(01)00159-4.

T. Tanaka. FITSAT, 2013. http://www.fit.ac.jp/~tanaka/fitsat.shtml, [accessed 03/05/2017].

B. Tang, S. Longfield, S. Bhave, and R. Manohar. A low power asynchronous GPS baseband processor. In *18th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2012. doi:10.1109/ASYNC.2012.20.

B. D. Tapley, S. Bettadpur, M. Watkins, and C. Reigber. The gravity recovery and climate experiment: Mission overview and early results. *Geophysical Research Letters*, 31(9), 2004. doi:10.1029/2004GL019920.

Technical University of Denmark. DTUsat-1, 2013. http://dtusat1.dtusat.dtu.dk/, [accessed 03/05/2017].

Technical University of Munich. MOVE - Munich Orbital Verification Experiment, 2014. http://www.move2space.de/First-MOVE/, [accessed 04/05/2017].

Technische Universitat Berlin. Beesat: BEESAT-1, 2013a. http://web.archive.org/web/20120316204856/http://www.raumfahrttechnik.tu-berlin.de/beesat/v-menue2/beesat-1/, [accessed 04/05/2017].

Technische Universitat Berlin. Beesat: BEESAT-2, 2013b. http://web.archive.org/web/20120818224949/http://www.raumfahrttechnik.tu-berlin.de/beesat/v-menue2/beesat-2/, [accessed 04/05/2017].

Texas Instruments. MSP430FR573X, 2011. http://www.ti.com/lit/ds/symlink/msp430fr5738.pdf, [accessed 03/05/2017].

M. Thomsen, J. Merayo, P. Brauer, S. Vennerstrom, N. Olsen, and L. Toffner-Clausen. Feasibility of a constellation of miniature satellites for performing measurements of the magnetic field of the earth. In R. Sandau, H.-P. Röser, and A. Valenzuela, editors, *Small Satellites for Earth Observation*, pages 123–132. Springer, 2008.

Tokyo Institute of Technology. CUTE-I, 2013a. http://lss.mes.titech.ac.jp/ssp/cubesat/index_e.html, [accessed 04/05/2017].

Tokyo Institute of Technology. Cute-1.7 + APD ii, 2013b. http://lss.mes.titech.ac.jp/ssp/cute1.7/index_e.html, [accessed 04/05/2017].

University of Aalborg. AAU CUBESAT - student satelitte, 2013a. http://www.space.aau.dk/cubesat/, [accessed 04/05/2017].

University of Aalborg. AAUSAT II, 2013b. http://www.space.aau.dk/aausatii/, [accessed 04/05/2017].

University of Aalborg. AAUSAT3, 2013c. http://www.space.aau.dk/aausat3/, [accessed 04/05/2017].

University of Applied Sciences of Southern Switzerland. TIsat-1, 2013. http://www.spacelab.dti.supsi.ch/tiSat1.html, [accessed 04/05/2017].

University of Colorado. Colorado Student Space Weather Experiment, 2013a. http://lasp.colorado.edu/home/csswe/, [accessed 04/05/2017].

University of Colorado. Hermes Cubesat, 2013b. http://spacegrant.colorado.edu/COSGC_Projects/co3sat/, [accessed 04/05/2017].

University of Louisiana. CAPE Wiki, 2013. http://wiki.ulcape.org/cape1:subsystems:payload, [accessed 04/05/2017].

University of Michigan. M-Cubed, 2013a. http://web.archive.org/web/20161009205553/http://umcubed.org/, [accessed 04/05/2017].

University of Michigan. Radio Aurora eXplorer, 2013b. http://web.archive.org/web/20150320152802/http://rax.engin.umich.edu/, [accessed 04/05/2017].

University of Oslo. NCUBE, 2013. http://web.archive.org/web/20090225122418/http://www.ncube.no/, [accessed 04/05/2017].

University of Surrey. STRaND-1 smartphone nanosatellite, 2013. http://www.sstl.co.uk/Missions/STRaND-1--Launched-2013/STRaND-1, [accessed 04/05/2017].

University of Tartu. ESTCube, 2013. http://www.estcube.eu/en/, [accessed 04/05/2017].

University of Toronto. CANX-1, 2013a. http://web.archive.org/web/20140320184946/http://www.utias-sfl.net/nanosatellites/CanX1/, [accessed 04/05/2017].

University of Toronto. CANX-2, 2013b. http://web.archive.org/web/20140320172405/http://www.utias-sfl.net/nanosatellites/CanX2/, [accessed 04/05/2017].

University of Würzburg. UWE-1, 2013. http://www7.informatik.uni-wuerzburg.de/en/research/space_exploration/projects/uwe_1/, [accessed 04/05/2017].

D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso. Revisiting Spacetrack Report #3. In *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2006a.

D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso. Revisiting Spacetrack Report #3, Aug. 2006b. http://celestrak.com/publications/AIAA/2006-6753/AIAA-2006-6753.pps, [accessed 04/05/2017].

T. Vladimirova, N. P. Bannister, J. Fothergill, G. W. Fraser, M. Lester, D. Wright, M. J. Pont, D. J. Barnhart, and O. Emam. Cubesat mission for space weather monitoring. In *Proceedings of the 11th Australian Space Science Conference (ASSC'11)*, pages 223–236, Sept. 2011.

J. M. Wersigner. AubieSat-1, 2013. http://www.crn2.inpe.br/conasat1/projetos_cubesat/projetos/AUBIESAT-1%20-%20Auburn%20University%20-%20Alabama%20-%20USA/AUBIESAT-1%20-%20SYS%20-%20Presentation.pdf, [accessed 04/05/2017].

K. T. Woo. Optimum semi-codeless carrier phase tracking of L2. *NAVIGATION, Journal of The Institute of Navigation*, 47(2):82–99, 2000.

Xilinx Inc. Spartan-6 FPGA Configurable Logic Block, 2010a. http://www.xilinx.com/support/documentation/user_guides/ug384.pdf, [accessed 04/05/2017].

Xilinx Inc. Space-Grade Virtex-4QV Family Overview, Apr. 2010b. http://www.xilinx.com/support/documentation/data_sheets/ds653.pdf, [accessed 04/05/2017].

Xilinx Inc. 7 Series FPGAs Configurable Logic Block, 2011a. http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf, [accessed 04/05/2017].

Xilinx Inc. Spartan-6 Family Overview, Oct. 2011b. http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf, [accessed 04/05/2017].

Xilinx Inc. Virtex-II Pro and Virtex-II Pro X Platform FPGAs Data Sheet, 2011c. http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf, [accessed 04/05/2017].

Xilinx Inc. Virtex-5QV product brief, 2011d. http://www.xilinx.com/publications/prod_mktg/virtex5qv-product-brief.pdf, [accessed 04/05/2017].

Xilinx Inc. LogiCORE IP Fast Fourier Transform v8.0, 2012. http://www.xilinx.com/support/documentation/ip_documentation/ds808_xfft.pdf, [accessed 04/05/2017].

Xilinx Inc. Spartan-6 FPGA DSP48A1 Slice User Guide, May 2014. http://www.xilinx.com/support/documentation/user_guides/ug389.pdf, [accessed 04/05/2017].

Xilinx Inc. Zynq-7000 All Programmable SoC Overview, 2016. http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf, [accessed 04/05/2017].

G. Xu. *GPS: Theory, Algorithms and Applications*. Springer, Berlin, 1st edition, 2003. ISBN 3-540-67812-3.

Y. Zheng. The GANDER microsatellite radar altimeter constellation for global sea state monitoring. In *13th AIAA/USU conference on small satellites*, pages SSC99–V–4, 1999.