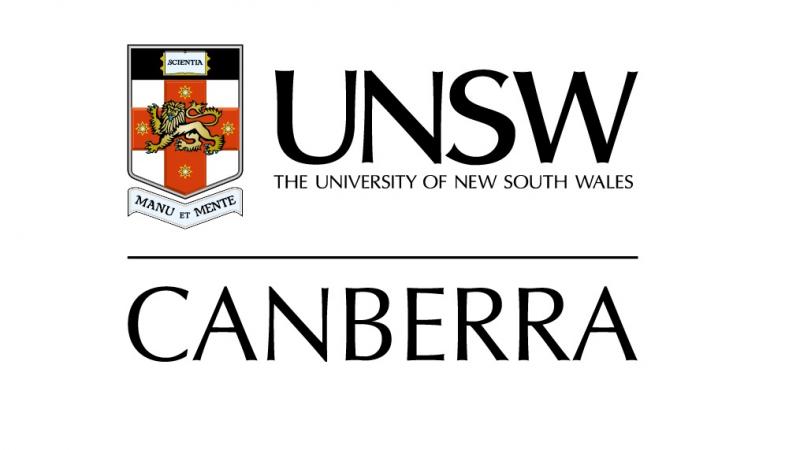
**Design of a small satellite UHF identification and TT&C radio beacon**



**Originator – Pilot Officer Travis J McKee**

**Supervisors – Dr Edwin Peters and Dr Craig Benson**

**The University of New South Wales at the Australian Defence Force Academy**

Contents

[Introduction 3](#_Toc42251485)

[Quick Overview (Abstract) 3](#_Toc42251486)

[Background 3](#_Toc42251487)

[Aim 4](#_Toc42251488)

[Requirement 4](#_Toc42251489)

[System Overview 6](#_Toc42251490)

[Solar power generation subsystem 8](#_Toc42251491)

[Power Budget 8](#_Toc42251492)

[Power regulation 8](#_Toc42251493)

[Computer sub-system 8](#_Toc42251494)

[Radio sub-system 8](#_Toc42251495)

[Total current requirements 8](#_Toc42251496)

[Full radio beacon cycle checks (1 Tx only) 9](#_Toc42251497)

[Power storage and regulation subsystem 10](#_Toc42251498)

[Computer processing subsystem 11](#_Toc42251499)

[Microprocessor 11](#_Toc42251500)

[Radio communications subsystem 12](#_Toc42251501)

[Data format 12](#_Toc42251502)

[Dropped/missing packets 12](#_Toc42251503)

[Communications Link Budget 13](#_Toc42251504)

[Ground receiver station subsystem 19](#_Toc42251505)

[Theory of Operation 19](#_Toc42251506)

[Initial configuration of the ground receiving station 19](#_Toc42251507)

[ground receiving program development 20](#_Toc42251508)

[Initial timing testing 21](#_Toc42251509)

[Sources of error in measuring the time/distance 21](#_Toc42251510)

[Timing Resolution (4µS micros() steps) 21](#_Toc42251511)

[Oscillator Resolution 22](#_Toc42251512)

[Oscillator Drift 23](#_Toc42251513)

[PPS Signal Tolerance 23](#_Toc42251514)

[RFM96 module processing time (preamble and message) 23](#_Toc42251515)

[Accuracy of the GPS position 23](#_Toc42251516)

[System Software 25](#_Toc42251517)

[Future work and system extensions 26](#_Toc42251518)

# Introduction

## Quick Overview (Abstract)

The reduced cost of designing, manufacturing, launching and operating small satellites has seen a significant increase in the number of objects deployed into the low earth orbit space environment. Small satellites operated by organizations with little space experience has resulted in a large failure rate, increasing the number of space debris. These two conditions have resulted in a greater reliance on expensive space monitoring equipment to maintain space situational awareness.

The aim of this study is to provide a concept design for a self-sustained satellite radio beacon that can send a unique identification and telemetry data, independent of any satellite system failures, to multiple dispersed ground stations which allows for tracking of the satellites position through trilateration. The satellite radio beacon system allows for: 1) the collection of telemetry data for fault analysis which could decrease the current satellite failure rate; and, 2) a radio signal for tracking using cost-effective ground stations irrespective of satellite system failures to reduce the reliance on resource-expensive ground monitoring equipment. The study will utilize an agile, iterative design approach in which each component of the small satellite UHF radio beacon system (satellite radio beacon, communications link and ground receiving station) will be designed, tested and verified in sequential order.

This study has produced a solderless breadboard prototype design for the satellite radio beacon that is capable of self-sustained operation in a ground environment. The communications link, using a LoRa radio module, is capable of the reliable transfer of data for the distances expected for a small satellite operational mission in a low earth orbit. The ground receiving station has proven the capability to receive the identification data, telemetry data and transmit a command for satellite control. Further investigation to reduce the uncertainty in the time measurement techniques by the ground receiving station is required to create an accurate tracking capability which would allow the continued development of the satellite radio beacon system.

## Background

The reduced cost of designing, manufacturing and launching a small satellite has led to an increased number of small satellites being launched into the Low Earth Orbit (LEO) space environment. The ability to ride-share a launch vehicle has reduced the cost of launching a small satellite into a LEO orbit but has led to a reduction in space situational awareness (SSA) due to upwards of 100 small satellites being launched from the same launch vehicle in a small-time frame. This single platform, multiple satellite launch produces difficulties in identifying an individual satellite amongst the cluster from the ground monitoring stations which can result in problems with creating a communication link with the individual satellite. The standardisation of small satellite manufacture has reduced the cost of production allowing government, educational and commercial entities to create and produce small satellite designs to meet their own requirements. The development and design process for each satellite has resulted in a 55% failure rate for academic institutions and a 23% failure rate for commercial industry with the cause of failures being unknown or reduced to a small number (5-10) of possible causes. The inability to control or track a space object and provide full SSA within the LEO area increases the risk of a collision for all users of the LEO space environment with a collision between two objects in space resulting in a larger number of smaller space debris objects leading to an exponential risk of further collisions between objects, which is known as the Kessler Syndrome. Providing a communication sub-system that is independent of all other systems on a satellite will allow the satellite to be identified and tracked regardless of the failure of any other satellite systems. The purpose of this project will be to design a self-sufficient, independent communication system that can transmit identification and basic telemetry data to multiple ground stations that can track the satellite by its radio frequency (RF) signal. This allows identification of an individual satellite during a single platform, multiple satellite launch providing greater SSA and reducing the risk of collisions with space debris for satellite and pass through operations in the LEO environment. The beacon will provide a UHF communications system that can provide identification and telemetry data for satellite on-orbit fault finding to facilitate determination of causes of failure to reduce the chance of failure for future launches and operations. This system will have the capability to be extended to include a ground monitoring system capable of tracking the small satellites and to provide an alternative communications system that can have limited control of the other satellite systems to offer a redundant system to correct on-orbit failures that prevent operation of the satellites primary communications systems.

## Aim

The aim of this project is to produce a design and ground tested prototype for a self-sustained communication platform that is independent of the existing satellite systems and can provide satellite identification, basic satellite telemetry and control data through a periodic radio frequency beacon transmission that is capable of being tracked utilizing multiple geographically dispersed ground receiving stations. The communications beacon will be able to operate as a separate system with its own independent power source, computer processor and radio providing information to the ground station regardless of a failure in any other system on the satellite for the length of a typical LEO small satellite mission (between 8-36 months). The primary focus of this design will be to provide an individual satellite with a unique RF identification for ground station monitoring equipment to a identify a satellite after a single platform, multiple satellite launch and provide a mechanism in which space monitoring organisations can perform SSA analysis of the LEO environment without using resource expensive radar and optical monitoring equipment. There is scope to use the radio beacon to provide basic telemetry data to allow for investigation of possible causes of failure for on-orbit post-failure analysis and rectification. This system can be further enhanced to include a receiving function that allows an alternative pathway for executing a control command if primary communications with the main satellite system is lost. Multiple ground receiving stations can be used to estimate the position of the satellite by analyzing the time of arrival and frequency differences to provide full a tracking capability for the operators or space monitoring organisation.

## Requirement

The standardisation of small satellite design, construction and launch has reduced the cost of manufacturing and launching a small satellite into a low earth orbit which has led to the number of small satellite missions increasing from 20 missions in 2011 to 322 in 2018 with the number estimated to increase to 300 missions in 2019 and between 2000 to 2800 missions to be launched in the next 5 years. A small minority of these planed missions will be delivered to LEO using dedicated launch vessels but most of the satellite are expected to utilize the current rideshare or piggyback launch vessels. The Indian Space Research Organisation released 104 small satellites into LEO during a 12-minute cluster release from a single launch vessel on the 15th of February 2017, which provided a demonstration of the ability for a cluster launch but also presented the problem of identifying a singular satellite within the launch cluster. Early identification and communication with the satellite and the ability to provide multiple pathways to correct system error or faults provides a good determination of mission success.

The 55% and 23% failure rate of the previous small satellite missions can be contributed to some common faults such as communications systems failure, power system failure, Bus system or payload interface failures, deployable component failure, failure of components (due to quality, inadequate documentation, limited testing) or system failures due to delays in launch (system degradation). When the satellite fails on launch or in orbit then the exact determination of the cause of failure is difficult to ascertain as the communication link with the ground station is often non-functional preventing satellite telemetry data from being received. The provision of a system that provides telemetry and satellite state data to the ground station irrespective of a failure of any satellite sub-system provides information that can be used to either determine the cause of failure or help reduce the number of possible causes of failure. When the cause of failure is identified or the number of possible causes of failure is reduced then this information can be used to reduce the incidents of full or partial failure in future small satellite missions reducing the failure rate of small satellites.

If the cause of failure is diagnosed whilst the satellite is on mission from the provided telemetry data, then there remains the possibility that an on-orbit rectification could be carried out and the mission can be completed or partially completed. The on-orbit rectification will require some sort of communication path to the satellite to be able to carry out any command to bring the affected system back to a serviceable state. If an emergency communication system has not been provided with the satellite system, then this radio beacon could be utilized to provide minor control of the satellite systems via an alternative communications link.

# System Overview

The system will be designed based on a self-contained radio beacon attached to the satellite transmitting a several data packets on a low power UHF signal to multiple ground receiving stations that contain a satellite identification and simple satellite telemetry information. The multiple ground receiving stations will send the data to a central server in which the time of arrival at each station will be used to calculate the satellite position.

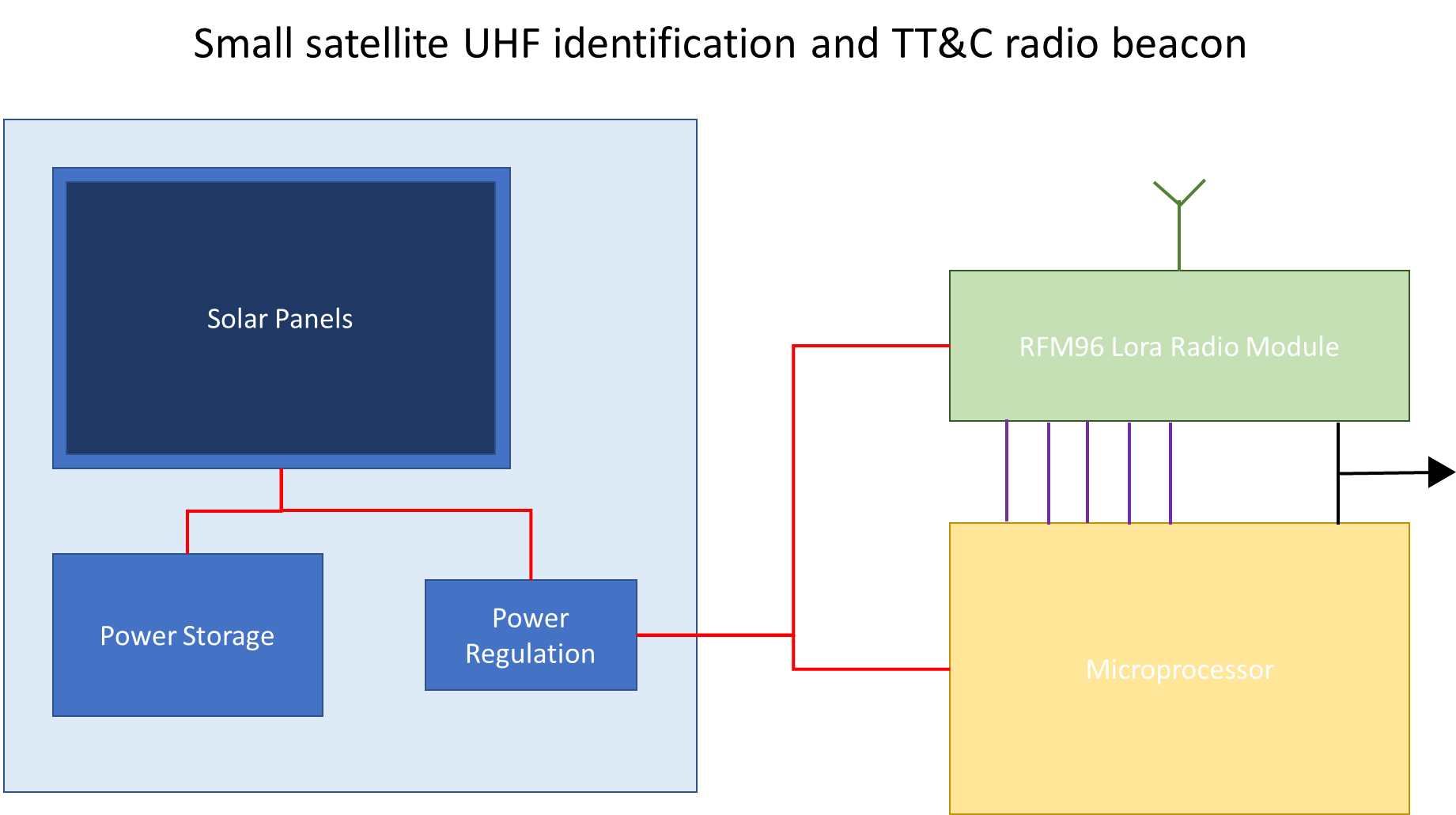


Figure 1 - Radio Beacon system breakdown

The radio beacon will be broken up into a power system (solar generation, regulation and storage), a computer system that will be controlled by a microcontroller (Arduino Pro mini) and the radio system (RFM96 LoRa radio) as shown in Figure 1. The Systems can utilise using a Adafruit 32U4 with RFM96 LoRa radio board, Can be broken up to the separate components on a breadboard or be soldered onto a red PCB board that can be ordered from diycon.nl or through EBAY.

The radio beacon system will operate by the beacon operating on a variable length of time cyclic program in which the system will collect data, transmit the data, wait for a command for a small period then enter a power down mode. Each cycle will last between 4-6 minutes based upon the orbital parameters of a small satellite mission in low earth orbit. An example of s simple beacon cycle is detailed in Figure 1 for the initial development and testing of the system, but noting that it will change depending on the final mission parameters.

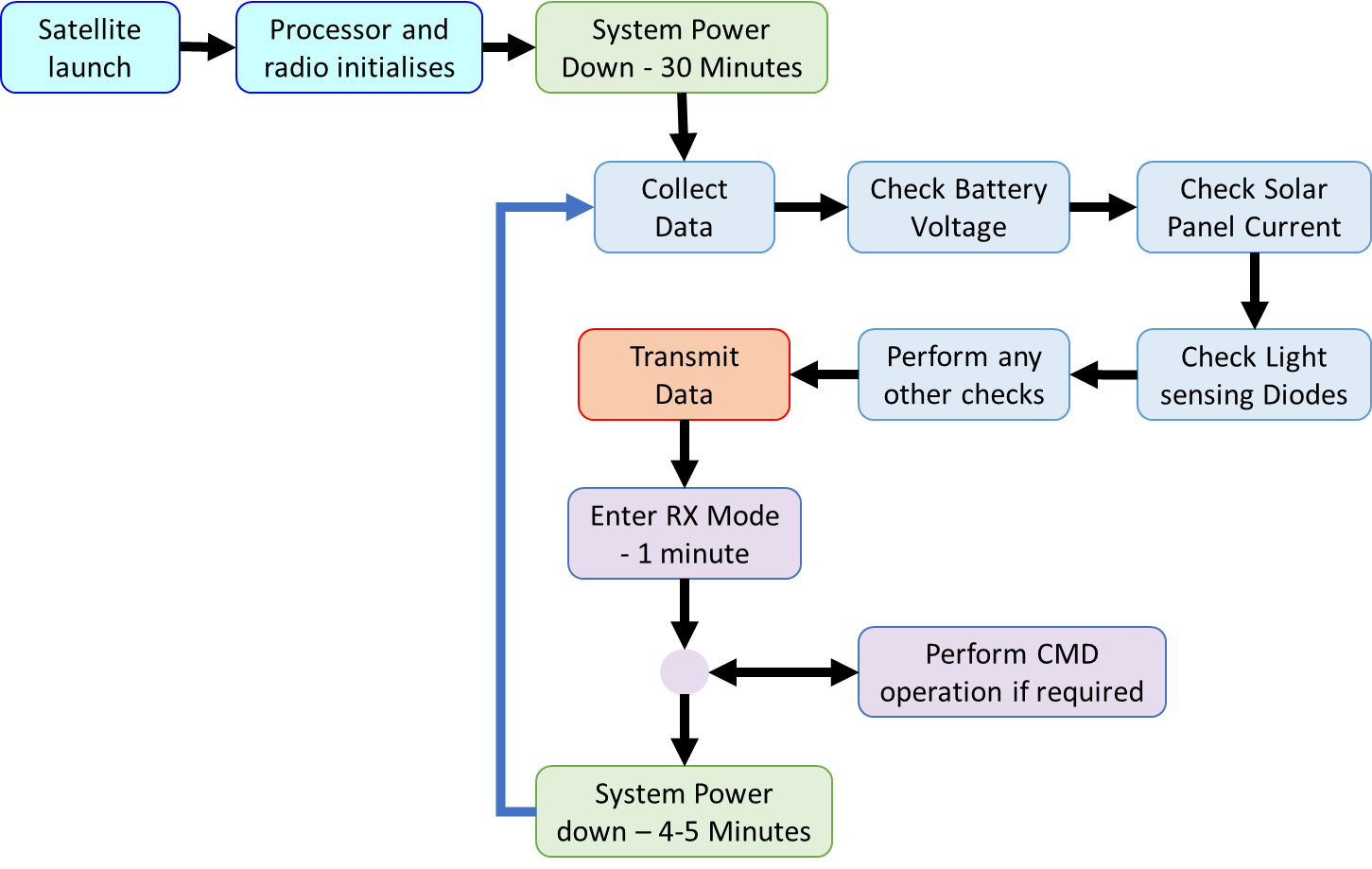


Figure 2 - Simple beacon cycle

The timing of the beacon cycle (4-6 minutes) is based upon LEO sun-synchronous with an inclination of 98 and an altitude of 600km. This results in an orbital period of approx. 90 minutes, a view window of 8-10 minutes and a slant range of between 1000-2000kms. The view window will allow at least transmission to occur with 2 transmission in that window being more likely.

# Solar power generation subsystem

The largest constraint on the solar power generation system will be the amount of space available on the outside of the satellite for the solar panels and the amount of power required by the computer and radio sub system. The major types of solar panels available are Gallium-Arsenide (GaAs) or a silicon-based (Si) with the GaAs being more efficient (producing higher power values for surface area) but it also comes with a higher purchase cost. A full investigation into a space worthy solar power generation system is yet to be carried out but will be carried out in the future.

The initial ground testing for the solar power generation system was carried out using a 0.5W monocrystalline Silicon solar panel from Seeed studio. The solar panels are 55x70mm and perform at a conversion rate of up to 17% producing approx. 5V at 100mA in full sun per panel. An investigation into the 1W (80x100mm) Seeed solar panels that produce 5V at 170mA will be carried out in the future.

## Power Budget

The power required by each sub-system will be tested before selecting a suitable solar power generation system.

### Power regulation

The 3.3V Arduino Pro Mini (APM) utilises a Low-Dropout voltage regulator (MIC5205 from microchip) with the datasheet show a quiescence current of 5µA, but testing of the system shows a consumption of approximately 5mA for regulation depending on the difference of input-output voltage but some current is also consumed by the onboard LEDs. When the LEDs are removed from the APM module then the regulation current is expected to drop below 1mA, but it maybe worthwhile investigating alternative LDO or utilising a buck converter if the power budget starts to get excessive.

### Computer sub-system

The APM module utilises an ATMEGA328P module for the processing and the testing carried show that the microprocessor utilises approx. 40mA while carrying out intensive computations when collecting the data, approx. 10mA when using the radio to transmit or receive and 1mA when put into a power saving mode.

### Radio sub-system

When the RFM96 module is transmitting at 5dBm then it consumes approx. 70mA and consumes approx. 10mA when in receive mode. When the satellite ID packet (2 Bytes) is being transmitted, the time on-air will be 170mS per radio packet and when the telemetry data is being transmitted the time on-air will be 495mS for 32 Bytes and 870mS. These values were obtained using the LoRa calculator obtained from HopeRF for a RFM96 radio at 437MHz with a spreading factor of 12, Bandwidth of 500kHz, preamble of 8 and a 4/8 coding rate.

The energy for each radio packet is to be determined and will be used to calculate the required energy in the capacitors to support a transmission.

## Total current requirements

The estimated maximum current requirement for the whole system during each phase of the software cycle based on a transmit packet of 50 Bytes is…

Data Collection – 45mA

Transmit data – 88mA

Receive – Waiting for command – 26mA

Receive – Performing command operation – 56mA

Low power mode (power saving) – 7mA

## Full radio beacon cycle checks (1 Tx only)

The initial testing for the solar panel checks were carried out using an abbreviated form of the radio beacon cycle, these tests will focus on the carrying out 1 transmission containing 50 Bytes of data using the expected beacon cycle as shown in Figure 2. The tests were carried out on a clear and sunny day at 34°C with the solar panel facing perpendicular to the sun. The results show that one SEEED solar panel was able to support Beacon operation and transmission of the radio up to a TX power of 15dBm. When the TX power was increased to maximum power (23dBm) then the system (APM and radio module) would reset after the data was transmitted.

To support operation of the transmission, x5 2200uF electrolytic capacitors (2.2mF) were added in parallel with the power source to provide enough energy when the radio transmits. The estimated current consumption during transmission is 88mA and the transmit cycles takes approx. 100mS which results in a system energy consumption of 0.03 Joules. The 5 capacitors are at 3.3V potential and has a capacitance of 11mF which results in 0.06 joules being stored in the capacitors. When the test was carried out again, beacon operations with a transmit power of 23dBm was able to be supported by a single solar power and a capacitance of 11mF.

The next step is repeat the test using a number of surface mount capacitors to check if 23dBm transmit power during normal beacon operation can be sustained and how much of the cycle or how many of the cycles can be support when the solar panels are covered.

I also need to investigate a larger storage system to support operation in low-level or no sun operations as well as investigating more efficient solar panels.

# Power storage and regulation subsystem

Need to conduct further tests on storing energy in capacitors including how many transmissions can be stored in capacitors (i.e. how long the radio can operate without sun)

Need to talk about the regulators that I am thinking of using (no testing) with the data verified by data sheets

Need to investigate super/ultra capacitors that can be used in space and order a generic/cheap option that can store energy and conduct some tests.

<http://www.home-automation-community.com/arduino-low-power-how-to-run-atmega328p-for-a-year-on-coin-cell-battery/>

# Computer processing subsystem

## Microprocessor

The microprocessor chosen for this application is the ATMEGA328P that is utilised on the Arduino Pro Mini module, but other items that could be used are the Teensy based microprocessors or the ATMEGA32U4 processor used in the Adafruit feather module. The APM module with the 328P microprocessor was chosen as the user interface (Arduino IDE) is a simple to use program for inexperienced users and has many resources available on the internet but it can also be programmed using a C++ program such as Microsoft visual, Visual Source Code (VSC), eclipse or codeblocks for the experienced user to free up memory onboard the module. The 328P processor also has a proven history of being used will lower clock speeds and lower voltages for low power applications as well as having a large amount of resources available for reference on the internet.

The initial testing of the system will be carried out using the Arduino IDE to show the proof of concept on the ground, but the full testing and operation will be carried out using a C++ program. The details of the software programming will be detailed in the software section. The 328P processor contains 32K Bytes of flash memory, 1K Bytes of EEPROM and 2K Bytes of RAM available for use with the Arduino IDE bootloader taking up to 2KB of the flash memory.

The APM module used was the version that operates on 3.3V and contains an external oscillator for the processor that operates at 8MHz. The 328P processor contain a 16MHz internal processor but the speed is lowered to 8MHz to decrease the required operating current. The speed of the processor can be lowered as the space segment will not require a large amount of processing, but the ground segment can be operated at 5V and 16MHz processing speed as it does not have the same power constraints. The RFM96 LoRa radio module and the 328P processor are currently being tested at 3.3V, but the voltage can be lowered to a minimum of 1.8V operating voltage as per the datasheet but it will also require the bootloader and programming file to be updated for the Arduino IDE to use the module. This will require an investigation into determining the new value for the fuses in the 328P processors which will be completed after the into testing of the communication link is completed. More information about power saving for the 328P microprocessor can be found at <http://www.gammon.com.au/power>.

# Radio communications subsystem

## Data format

The data packets will be broken up into 2 elements, the first containing the satellite ID number and the second containing satellite telemetry information. The satellite identification will be represented by 4 hexadecimal characters which allows 65,536 combinations for the address which will a large enough of combinations to cover all predicted number of LEO satellite launches. Each satellite ID will consist of 2 Bytes representing 4 hexadecimal characters (0xffff) and will be repeated 8 times in the transmission in 4 packets of data transmitted in 4 radio transmissions, i.e. [0xffff 0xffff] [0xffff 0xffff] [0xffff 0xffff] [0xffff 0xffff]. These packets will then be preceded with one radio packet containing the 32 or 64 Bytes of satellite telemetry data with the transmission looking like…

[0xffff 0xffff] [0xffff 0xffff] [0xffff 0xffff] [0xffff 0xffff] [32-64 Bytes of telemetry data]

The satellite ID will be sent eight times to increases the Bit energy of the ID data to minimise Bit errors and maximise the likelihood of receiving the satellite ID. The ground receiving station will be able to process each bit of the hexadecimal data using a majority polling to provide an additional error checking process.

## Dropped/missing packets

The initial testing of the RFM95 radio module indicated that some packets of information were not received by the ground station module. The testing was carried out by sending 10,000 radio packets containing one randomly generated Byte (a Hexadecimal character) in 100 batches each containing 100 packets. The radio was initiated using the default settings with the coding rate changed between 4/5 and 4/8 with the results showing that the number of packets that were not received decreasing as the coding rate increases (as expected). When the coding rate is at 4/8 then the number of packets dropped by the receiver was 8 in every 10,000 packets approximately. When the coding rate is dropped to 4/5 then the rate increased to between 20-30 dropped packets per 10,000 packets sent. The LoRa radio receiver drops the incoming data due to multiple reasons with the two main reasons for not reading the data is if there is an error in the message preamble (*ValidHeader*) or if the cyclic redundancy check (CRC) fails due to an error in the message (*PayloadCRCError*).

Using the LoRa datasheet, it was determined that the CRC error check is able to be turned off using the Radiohead library function setPayloadCRC() which is used in the receiver only. This function sets the receiver *RegModemConfig – RxPayloadCrcOn* such that the CRC information is or is not extracted from the received packet header. When the CRC check was turned on or off then there was not a significant change in the number of radio packets that were dropped. The same test as above (100 batches of 100 radio packets) was carried out but where each Byte that was received by the LoRa receiver was compared against the originally generated random data. The results show that no Bytes received by the Lora radio contained any errors which indicate that the CRC checks is carried out regardless of the setting the CRC check on or off and that the packets being dropped are due to an invalid header in the preamble. Further testing was carried out where each radio packet was increased to contain 10 Hexadecimal Bytes, with each batch contain 10 radio packets and 1,000 batches sent in total. The results of this testing corresponded with the previously obtained results in which the determining factor of the dropped packets was inconclusive and no Bytes with errors were processed by the receiver.

The functions txGood(), rxGood() and rxBad() from the RHGenericDriver class library were used to determine if the transmitting radio fails to send data or if the receiving radio fails to process the data due to errors in the preamble or data. The txGood() function returns the number of packets successfully transmitted by maintaining a running total. The rxGood() function returns a running total of the number of good packets received and the rxBad() function returns a running total of the number of bad received packets which were rejected and not delivered to the application (APM module). For the data to be passed from the FIFO register to the APM module then the following flags are checked in the *RegIrqFlags* register…

* *ValidHeader*
* *PayloadCrcError*
* *RxDone*
* *RxTimeout*

If any of the flags are set to on, then the received packet is dumped from the FIFO register and the Arduino module does not receive the sent data. To test this, the 1 byte per radio packet, 100 packets per batch and 100 batches test program was used where a running total of the good/bad received packets and good transmitted packets are monitored. When the SetPayloadCRC() is set to true then 10,000 good packets were transmitted, with 9915 packets received with 85 packets not being seen by the Arduino module. Of the 85 dropped packets, 79 were dropped due to an error flag in the *RegIrqFlags* register and 6 not being received at all due to a failure to detect the preamble (preamble CRC check). When the SetPayloadCRC() is set to false then number of dropped packets decreases to 12 with 3 packets received with an error flag and 9 not received at all (preamble failure).

When investigating the LoRa radio datasheet, it was determined that when the radio operates in LoRa mode then the radio registers/address can not be changed to prevent the FIFO from clearing if the radio receives a packet containing a CRC with an error. If the radio is operated in FSK/OOK mode then the CrcAutoClearOff register can be set but this is not available in the LoRa mode. It was decided at this point that it was not worth pursuing the ability to turn the CRC check on or off and accepting that there will be some packets loss (typically 0.001%) which is acceptable. To ensure that we receive some usable data the satellite identification data will be transmitted in multiple packets so that the loss of 1 to 3 packets for every 4 sent will still allow the data to be useful and usable in this application.

Review testing document and provide further elaboration…. possibly

## Communications Link Budget

The basic equation used for the communications link budget is…

Where,

The transmitter power used in the application is 5dBm which equates to -25dB.

The gain of the receiver and transmitter antenna will be based upon a ½ wave dipole antenna or a spring antenna which typically has a gain of 2dB (32dBm). The value for the receiving antenna will change as a higher gain antenna can be used on the ground.

The antenna coupling loss for the receiver and transmitter will be estimated to be -4dB (34dBm) for each antenna. The value for coupling loss will change dependant on the type of antenna used and the PCB placement.

The distance that the radio will be required to operate over is between 1000-2000kms slant range from the ground station to the satellite which equates to a free space path loss of 145-151dB. The equation used for the free space path loss is…

Where,

The exact values for FSPL for varying distances are…

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Distance (km)** | 1000 | 1090 | 1220 | 1370 | 1540 | 1730 | 1930 | 2000 |
| **FSPL (dB)** | 145.2 | 146 | 147 | 148 | 149 | 150 | 151 | 151.3 |

The maximum receive sensitivity of the LoRa radio (RFM96) is -111 to -148dBm, which requires that further gains be found in the coding gain, increasing the spreading factor, higher antenna gains, increasing the transmitter power, lower data rates, modulation and more. Other losses that will also need to be considered like doppler effect, satellite tumbling, polarisation, modulation, atmospheric scintillation and more.

The Maximum attenuation available for testing is 167dB which is equal to approximately 12,500kms. The testing will be carried out using a minimum of 145dB FSPL (1000km) to ensure the system will work for the designed mission but will aim to achieve an operating distance of 2000km which equates to 151dB FSPL.

To test and prove that the radio can operate theoretically over the distance, a signal is passed through a series of attenuators and cables that represent the desired distances. The cables used for the testing is RG-174/U coaxial cable which has an attenuation of 0.62dB/m (obtained from multiple datasheets). The program used for the testing is the generic “Hello world” program obtained from the Adafruit website which utilises the Radiohead library for the RFM95 module (<https://learn.adafruit.com/adafruit-feather-32u4-radio-with-lora-radio-module/using-the-rfm-9x-radio>). The radio is initiated using the RadioHead (0) default settings which set the modem config to Bw125Cr45Sf128 which is designed for medium range and has the settings…

Bandwidth = 125kHz

Coding Rate = 4/5

Spreading factor = 8 (128 chips/symbol)

Resultant data rate is 3125bps and time-on-air is 57.86 mS (obtained using the Lora calculator)



In order to get longer range for the radio, then the modem config can be changed to the modem settings Bw31\_25Cr48Sf512 or Bw125Cr48Sf4096 where the settings are….

**Bw31\_25Cr48Sf512**

Bandwidth = 31.25kHz

Coding Rate = 4/8

Spreading factor = 10 (512 chips/symbol)

Resultant data rate is 152.59 bps and time-on-air is 1122.3 mS



**Bw125Cr48Sf4096**

Bandwidth = 125kHz

Coding Rate = 4/8

Spreading factor = 12 (4096 chips/symbol)

Resultant data rate is 183.11 bps and time-on-air is 860.16 mS



**Distance test using 5dBm and standard modem settings (Bw125Cr45Sf128)**



Figure 3 - RFM95 distance test set-up - 5dBm transmit power

The test setup used for testing the maximum transmit distance when transmitting at 5dBm is detailed in Figure 3 with the A attenuator being adjusted to fine tune the total attenuation. The results from the testing are…

When A = 4dB, total attenuation is 129.06 and all packets are received

When A = 5dB, total attenuation is 130.06 and all packets are received

When A = 7dB, total attenuation is 132.06 and no packets are received

The results show that the radio can operate without losing any packets when the total attenuation in the transmit path is 130.05 which represents a FSPL distance of 175kms.

**Distance test using 10dBm and standard modem settings (Bw125Cr45Sf128)**



Figure 4 - RFM95 DISTANCE TEST SET-UP - 10DBM TRANSMIT POWER

The test setup used for testing the maximum transmit distance when transmitting at 10dBm is detailed in Figure 4 with the A attenuator being adjusted to fine tune the total attenuation. The results from the testing are…

When A = 4dB, total attenuation is 137.06 and all packets are received

When A = 5dB, total attenuation is 138.06 and the packets are inconsistent (every 4th packet RXed)

The results show that the radio can operate without losing any packets when the total attenuation in the transmit path is 137.06 which represents a FSPL distance of 390kms.

**Distance test using 15dBm and standard modem settings (Bw125Cr45Sf128)**



Figure 5 - RFM95 DISTANCE TEST SET-UP - 15DBM TRANSMIT POWER

The test setup used for testing the maximum transmit distance when transmitting at 15dBm is detailed in Figure 5 with the A attenuator being adjusted to fine tune the total attenuation. The results from the testing are…

When A = 9dB, total attenuation is 141.06 and all packets are received

When A = 10dB, total attenuation is 142.06 and no packets are received

The results show that the radio can operate without losing any packets when the total attenuation in the transmit path is 141.06 which represents a FSPL distance of 620kms.

**Distance test using 20dBm and standard modem settings (Bw125Cr45Sf128)**



Figure 6 - RFM95 DISTANCE TEST SET-UP - 20DBM TRANSMIT POWER

The test setup used for testing the maximum transmit distance when transmitting at 20dBm is detailed in Figure 6 with the A attenuator being adjusted to fine tune the total attenuation. The results from the testing are…

When A = 13dB, total attenuation is 145.06 and all packets are received

When A = 15dB, total attenuation is 147.06 and the packets are inconsistent (every 2nd packet RXed)

The results show that the radio can operate without losing any packets when the total attenuation in the transmit path is 145.06 which represents a FSPL distance of 980kms.

**Distance test using 5dBm and long-range modem settings (Bw31\_25Cr48Sf512)**

****

Figure 7 - RFM95 DISTANCE TEST SET-UP - 5DBM Transmit power with alternative modem settings



Table 1 - Measured FSPL attenuations and distances for reliable Data transmission

The test setup used is detailed in Figure 7 but the settings in the modem were changed to the Bw31\_25Cr48Sf512 modem settings (*///< Bw = 31.25 kHz, Cr = 4/8, Sf = 512chips/symbol, CRC on. Slow+long range*) and using the same TX Powers with the results detailed in Table 1.

**Distance test using 5dBm and long-range modem settings (Bw125Cr48Sf4096)**

The test setup used is detailed in Figure 7 but the settings in the modem were changed to the Bw125Cr48Sf4096modem settings (*///< Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on. Slow+long range*) and using the same TX powers with the results detailed in Table 1.

\*Further investigation into the optimal settings can be carried out to determine the best settings taking into account power, bandwidth and time-on-air restrictions imposed by using a unlicensed frequency.

# Ground receiver station subsystem

## Theory of Operation

The ground receiving station will have 2 purposes, the first is to receive the satellite identification and telemetry data and the second is to provide a precise time of arrival for tracking purposes.

The Satellite identification data is the satellite address represented as 4 hexadecimal characters that uniquely identify the satellite the signal originates from. This requires that a database of all addresses that have been assigned to a unique satellite be kept and maintained to ensure the integrity of the system. There are a possible 65,536 addresses using the 4 hexadecimal characters, which would allow for a minimum of 22 years of operation (using the nano-microsatellite forecast figures) without a clash in addresses. The telemetry data that is collected by the ground receiving station will be available to be read by any operator, but the contents of the radio packet will not be meaningful with knowing the structure of the data in the radio packet. This ensures that the operator of each individual satellite has control over the protection of that data, so if they would like to share the structure and meaning of the data in the radio packet then they can choose to do so. It will encouraged to provide the details of the telemetry data to allow the amateur operator of the ground receiving station to collect the data and send to a centralised storage and processing facility to allow inclusiveness amongst the amateur, commercial, academic and governmental organisations in the LEO space operations.

The precise Time Of Arrival (TOA) and satellite address will be recorded by each ground receiving station for each signal sent by a satellite beacon to allow triangulation of the signals using Time Difference Of Arrival (TDOA). Using the TDOA of a beacon signal being received at 4 ground receiving stations will allow calculations to be carried out that will identify the 3-Dimensional position of the satellite. The information required to perform the calculation will be the Latitude, longitude and altitude of each ground receiving station (this can be obtained using GPS) and the time of arrival of the signal at each beacon (ideally, down to nano second precision where 1nS represents a distance of 0.3ms and 100nS represents a distance of 30ms). The data that is sent from the beacon is broken up into 5 separate packets (4 identification packets and 1 telemetry packets) with each packet allowing a separate time stamp to be provided for time of arrival calculations (averaging will help with precision).

## Initial configuration of the ground receiving station

The initial configuration of the ground RX station is shown in Figure 8 and was chosen as the components are readily available and are affordable for the amateur electronic enthusiasts.

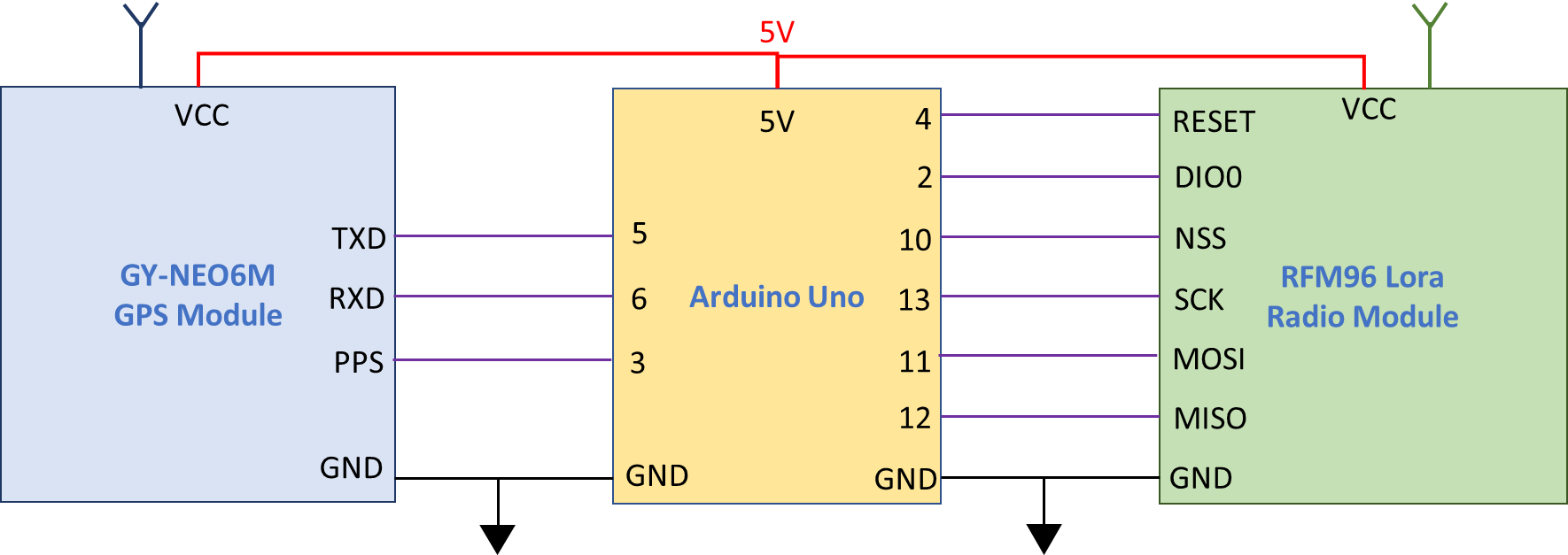


Figure 8 - Block Diagram of the initial configuration of the ground receiving station

The RFM96 Lora radio is mounted onto a breakout PCB which allows for direct integration with a standard breadboard (2.54mm hole-to-hole spacing), which can be made by any PCB manufacturer (PCBWay, PCBs.io) using files from GITHUB (<https://github.com/attexx/rfm9x_breakout_board>). The antenna used for the initial testing was a simple SMA 433MHz omni-directional antenna with a 2dBi gain. A higher gain alternative is to be investigated for operations (non-testing purposes).

The GPS unit used on the breadboard is the GY-NEO6MV2 U-blox GPS receiver which was purchased from ebay for a cost of approx. $15 (<https://www.ebay.com.au/itm/AU-Ublox-NEO-GY-GPS-6M-V2-Flight-Module-GY-NEO6MV2-3V-5V-passive-antenna/172134476783?hash=item281403abef:g:02sAAOSwi-9ZisoG&frcectupt=true>). This model was selected as it contained a built-in passive antenna that has a gain of 28dBi which is sufficient to capture a usable GPS signal if the ground receiving station is located outside. This GPS unit has a provision to solder a alternative SMA plug onto the board which allows for a second antenna to be fixed to the GPS chip. This allows a better antenna (the one used for testing has a gain of 40dBi) which allows the ground station to be utilised indoors. This unit also contains a breakout pin for obtaining the GPS PPS signal which is a low-to-high pulse that repeats every second with a tolerance of 10nS. This signal, in conjunction with the UTC provided by the GPS signal will be used to synchronise the timing between the individual ground receiving stations.

The Arduino unit selected for the initial testing and development of the software was the Arduino Uno as it has a higher processing power and more pins for use over the Arduino Pro mini. The only problem with this unit is that the oscillator used is a ceramic oscillator (CST CE 16MOV 53-RO) which has a large tolerance for frequency stability and tolerance. This results in a drift of 0.8% or ±3000ppm (parts per million) which equates to a drift of 128,000 clock cycles over one second.

One alternative board to use will be the Arduino Mega2560 which has slightly better processing performance, similar pin layouts but contains a crystal oscillator which suffers less drift than a ceramic oscillator. The frequency drift of the crystal oscillator used in the Mega2560 is ±70ppm which results in a tolerance of 1120 clock cycles over 1 second.

A second alternative would be to utilise a teensy3.2 which contains a 72MHz processor with more memory (64k RAM) or a teensy3.5 which has a 120MHz processor and 192k RAM.

## ground receiving program development

The initial development of the ground receiving software cycle was used to get a baseline program that allows for reception of the LoRa signal, recording a LoRa signal time stamp, obtaining time stamp of GPS PPS signal, obtaining GPS data (ground station lat/long/alt and the UTC value) and displaying these on the serial monitor. The program ran such that when the Arduino Uno is powered then the LoRa and GPS modules are initialised, and the system waits for a valid signal from the LoRa unit. When a valid signal is received by the LoRa radio (contains valid header and packet CRC checks, Continuous Reception operating mode – RFM96 data sheet, page 35-6), then a time stamp is taken and saved in the Arduino RAM with the contents of the packet is displayed on serial monitor. When the fifth packet is received (5th time stamp taken, and telemetry data displayed) then the ground receiver waits for the PPS signal from the GPS unit which triggers an interrupt sequence. During this sequence a time stamp is immediately taken which allows for the time between the PPS signal (which is representative of the UTC time with a second resolution as received the last packet of GPS data) and the time stamp of each of the 5 packets of data. There will be a delay in time between receiving the PPS signal, creating an interrupt and measuring the current micros() time stamp, but this delay will be the same for each ground unit as the computer processes are the same, with only the delay in each clock frequency needing to be considered for errors in measurement.

## Initial timing testing

The first test carried out was to see if the PPS signal from the GPS unit is a valid signal that can be used to signpost breaks in time every second for the Arduino Uno. Each time the signal transitions from low to high, an interrupt is carried out by the microcontroller where a time stamp is captured using the micros() function in Arduino. When the next PPS signal arrives then a 2nd time stamp is captured in which the difference represents the time between PPS signals as determined by the microcontroller clock (obtained from measuring the frequency of the oscillator). The results show that the there was a time difference of -252 or -256 microseconds every time the PPS signal was received which shows that as room temperature (approx. 25\*C) the time drifted by 252-256µS which equates to a drift of approx. 3906 clock cycles. This clock drift is comparable to a crystal oscillator and if the drift is measured directly after the time stamps are taken then the drift caused by local conditions for that ground station can be compensated for. This testing also showed that the maximum resolution that can be obtained using the micros() function is 4mS which equates to a distance tolerance of 1.2kms which is too large for this application. This requires an alternative method to measuring time using the Arduino Uno (i.e. directly reading the clock registers, etc) to increase the resolution of the timing and to decrease the error in measuring the distance from the ground station to the beacon.

## Sources of error in measuring the time/distance

The possible main sources of error in measuring the time down to an acceptable degree of accuracy are..

* Resolution of the micros() function timing – 4uS
* Resolution of the oscillator – 16MHz equates to a clock cycle every 62.5nS (or 18.75ms distance travelled by an EM wave).
* Drift in Oscillator frequency due to tolerances, temperature, and other sources of error.
* Tolerance in the PPS signal (typically 30nS)
* The time taken for the LoRa module software to initially process the preamble (valid header CRC and *RXDone* interrupt is set) before the radio packet is checked for integrity.
* The time taken for the packet information (the message) to be checked for integrity (*PayloadCrcError* is checked).
* Accuracy of GPS latitude/longitude/altitude measurement (typically within 10ms but is dependent on GPS signal strength and number of GPS satellites obtained)

### Timing Resolution (4µS micros() steps)

When the in-built Arduino micros() function is used to capture the time, the measurement is made in 4µS increments which equates to a distance measuring error of 1.2kms which is too large for this application. If the Timing clock of the ATMEGA328P can be accessed, then theoretically the resolution of timing measurement could be reduced to 62.5nS which has a distance error of 18.75ms. The Timer1 register in the ATEMEGA328P is a 32Bit counter which has a resolution of 62.5nS for the values held in the TCNT1 counter (number of clock pulses) which can hold 65535 values (or counts up to 4096mS) if the pre-scaler is set to 1 or 0x01 in the TCCR1B register. The Timer1 can be set-up such that once the TCNT1 counter overflows (reaches 65536 clock pulses) then an interrupt can be setup to count the number of overflows and the TCNT1 resets back to 0 to begin again. To set this up then the CTC mode must be turned on in the TCCR1B register (*TCCR1B |= (1<< WGM12);*) and the interrupt must be turned on (*TIMSK1 |= (1 << OCIE1A);*). In this way the total number of clock pulse that have occurred since the Timer1 has been reset can be counted and multiplied by the frequency of the oscillator to determine the amount of time that has passed with a resolution of 62.5nS.

**\*\*Important Note – When the Arduino Uno is initiated using the Arduino IDE, then the init() function for the microcontroller is carried out in the background where all the pins are initially set-up for PWM operation. This causes the Timer1 to revert to an 8-Bit timer where the low register (TCNT1L) holds the 8-Bit value, but the high register (TCNT1H) is turned off. Before using the Timer1 it must be reset to revert it back to a 16-Bit timer which can be done by resetting the TCCR1A and TCCR1B registers to 0 or 0x00 (*TCCR1A=0;* and *TCCR1B-=0;*)\*\***

**\*\*Important Note – If using the Radiohead library to drive the LoRa radio, then the RH\_ASK.c and RH\_ASK.hpp utilise the Timer1 counter. The Arduino compiler does not work with this conflict and as these files are not required in this application then they should be removed from the radio library\*\***

To use the time function in this application, we can reset the Timer1 register (TCNT1) when the first LoRa signal arrives, then we can mark the time between the first signal against the next 4 LoRa signals. After the timestamps of all 5 Lora signals are captured then we can wait for the next PPS signal from the GPS to mark the final time stamp and get the current GPS Universal time Coordinated (UTC) time stamp which has a 1 second resolution. We can then subtract each LoRa timestamp from the GPS UTC time stamp to get the time of arrival with a 62.5nS resolution.

All the ground receive stations will receive the GPS PPS signal and the correspond UTC time simultaneously which will synchronise all the times between the ground stations. The time-of-arrival will be dependant on the number of clock cycles that have occurred since the LoRa signal has arrived and the PPS signal. The two main factors that may have an error in measurement between the measurements will the different drift in oscillators frequencies (which will be discussed shortly) and the way that the microprocessor is coded to measure the PPS signal and microprocessor clock cycles. If all the ground stations execute the measurement of the PPS signal and microprocessor clock cycles in the same way using the same code, then there will be no difference in the measurements.

To ensure there is no variances, a Uno was used to measure the amount of clock cycles between PPS signals which in theory there should be no or very little difference between one pulse to the next. Over time there may be an overall drift due to factors such as a difference in temperature effecting the oscillator, etc. To measure if there is a variance in the number of clock cycles between PPS signal then clocks cycles between each PPS pulse will be measured over a large period to allow a large sample of data. The difference in the number of cycles between the previous and next is determined to find the immediate average differences in clock cycles. This will allow an error in reading the measurements to be determined to factor in the distance calculation.

\*measure difference in clock cycle measurements using the GPS PPS Signal\*

### Oscillator Resolution

The Arduino Uno contains an 16MHz oscillator which results in a clock cycle occurring every 62.5nS which is the maximum resolution that could be achieved when measuring time. This results a resolution of 18.75ms when using the measured time to calculate the distance from the ground station to the satellite. The only way to improve this will be to either install a oscillator with a larger frequency, which would be a non-preferred option as it would require specialist equipment to perform this operation and would cause the ground station to be unable to be made by an amateur operator. A more preferred option would be to investigate other existing Arduino boards that contain a higher oscillator frequency with the following boards being research and found to be acceptable to be using in this application…

* Arduino Due ($80) – 84MHz = 12nS clock cycles or 3.6ms distance resolution
* Arduino M0 (PRO) ($50) – 48MHz = 20.8nS or 6.22ms \*may be difficult to get one
* Arduino Zero ($45) – 48MHz = 20.8nS or 6.22ms
* Arduino MKRZero ($60) – 48MHz = 20.8nS or 6.22ms \*will need to check if enough pins)

### Oscillator Drift

The Arduino UNO contains a 16MHz ceramic resonator (CSTCE16M0V53-R0) which has a frequency tolerance of 0.5% and a frequency stability of 0.3%. This results in the oscillator possibly having a frequency drift of 128kHz due to temperature, aging or tolerance in manufacture and the tolerance will not be the same for each oscillator in each ground station. This means that the exact frequency of the oscillator at the time the measurement for each ground station must be known in order to determine the exact length of time for each clock cycle.

To GPS PPS signal occurs every one second with a tolerance of 10nS which can then be used to measure the number of clock cycles in a second. The number of Timer1 clock cycles that occur between PPS pulses will determine the instantaneous frequency of the oscillator and the time of each clock cycle. To help reduce errors in determining the oscillator frequency, then the number of clock cycles per PPS pulse should be done over multiple PPS pulse cycles and averaged. This will be implemented in the ground station after all the time stamps of the LoRa radio and GPS PPS are taken. When the GPS PPS time stamp is taken then the clock pulse are counted for the next 3 GPS PPS pulse, summed together and then divided by 3 to determine the instantaneous frequency of the oscillator. This value is then used to determine the exact time of one clock pulse which is then used to determine the amount of time that has elapsed between each LoRa timestamp and the GPS PPS timestamp (using UTC time).

### PPS Signal Tolerance

The GPS module used in the ground station utilises the u-blox NEO-6M GPS module which has a PPS time tolerance of ±10nS that equates to an added distance measuring error of 3ms.

\*need to find reference to data sheet\*

### RFM96 module processing time (preamble and message)

\*need to document what I have done so far\*

### Accuracy of the GPS position

The data sheet for the u-blox NEO-6M states that the accuracy of the GPS positioning is 2.5ms, my own tests utilising the u-center to monitor the position shows that the positioning tolerance is larger than 2.5ms but never greater than 10ms. The biggest determinant of the tolerance in position is the antenna that is used and the weather of the day. The external antenna used can lock onto more GPS satellites than the built-in antenna which results in less positional error, while the using the built-in antenna or a cheap patch antenna could produce an error of up to 30ms as less GPS satellites are able to be locked onto.

If there is a lot of bad weather present in the local environment such as clouds or rains, then less GPS satellites can be picked up result in a greater positional tolerance. The only way to counter this error due to weather is to use the best antenna available for the GPS module.

# System Software

Start to investigate using Arduino Vs using a C++ program

Include sample code with comments

# Future work and system extensions

Solar power panels

Power storage

Reduction of voltage supply to 1.8V

Full PCB design for space segment

Full PCB design for ground receiving station

Tracking system for multiple geographically dispersed ground receiving stations