The things I did not get to discuss this morning for the ground receive station, the errors that I have identified in measuring time/distance

### The 4µs resolution of the Arduino built-in timer – micros()

This one was solved by using the counting the clock cycles using the Timer1, but there is a small tolerance in measuring using clock cycles (see Note 1), further testing may be required

1. The resolution of the processor external oscillator – 16MHz results in a 62.5ns clock cycle

This one is stated to show there is a resolution in measuring with clock cycle, but I also discuss using the Due for a higher resolution if required (the final tolerance is too large). No testing to be done for this error

1. The number of clock cycles taken to carry out an interrupt

This one was discussed as the ISR process is identical for each ground station, then the number of clk cycles carried out for each ISR is the same which would result in no change in the time differential of arrival. No testing to be done for this error

1. Oscillator drift due to temperature, tolerances and other sources of error

This one is solved by using the PPS signal averaged over 3 one second periods using 4 PPS signals to get the instantaneous frequency of each ground station which determines the length of each clock cycle to measure the time. This compensates for any oscillator or environmental induced errors. This may suffer the same drift in measure clock cycles as error 1) (see Note 1), further testing may be required

1. Tolerance of the GNSS PPS signal – 30ns

Acceptable tolerance, No testing to be done for this error

1. Accuracy of the ground station GNSS position – between 2-10m

Acceptable tolerance, No testing to be done for this error

1. The time taken for the LoRa module built-in software to carry out integrity checks (time between when the signal is received and when it is made available by the LoRa module) – unknown

I still haven’t figured out how to measure this one accurately (I have it done to a error of milliseconds which is too large). I am going to investigate other libraries that can drive the RFM96 module that has the CAD mode properly implemented where I get a interrupt output from the LoRa mode when activity is detected and when RXDone is flagged. I am also still looking through the semtech datasheet to see if they have some tolerances, and I am trying to get in touch with their support if they have any testing, further testing may be required

**NOTE 1**

I performed a test where the number of clock cycles between PPS pulses are measured to see the amount of difference from one pulse to the next. I have included the code I used to get the number of clock cycles between pulse in Appendix A and the method of determining the difference from one pulse period to the next with histogram and graph in Appendix B.

The results show that when I take a large sample (9758 samples), then there are 180 clock count difference that are greater than 4 or -4, see Figure 1 and Figure 2. These results tell me that when you are counting clocks cycles then if the Arduino is carrying out some other instruction then the number of clock cycles in error will be fair amount (much greater than 5). If it is operating as per normal, then the clock counter will have a tolerance of ±4 clock cycles for 98.2% of the time. There are 260 differences greater than ±3 cycles (tolerance of ±3 97.3% of the time) and 893 differences greater than ±2 cycles (90.8%, this confidence level is not good enough). I have included the histogram below as well in figure 3 with the information that get from MATLAB for a confidence interval of 99% being…

total\_stddev = 10.8625, total\_mean = -0.0463, total\_samples = 9757

H = 0, P = 1, 99%CI = -0.3296, 0.2369, 95%CI = -0.2619, 0.1692

I have tried to get in touch with Lessa Sidhu to see if she can help with the statistics side of it but had no response yet. I am unsure what is the best way to use this tolerance in counting the clock cycles.

Do I need to include this tolerance in counting clock cycles every measurement I make for the time stamps?

Do I need to factor in this error when I determine the instantaneous frequency? And are the statistics different as the sample size will be 3 measurements?

\*I have also included a cool graph in Figure 4 which shows the decline in clock cycles as the temperature goes down and in particular at the end when the sun goes done and it gets really cold\*



Figure 1 - clock cycle count difference between 2 PPS signals



Figure 2 - clock cycle count difference between 2 PPS signals - Zoomed in



Figure 3 - Histogram for the clock cycle difference between PPS pulses\



Figure 4 - Number of clock cycles measured between each PPS pulse

volatile unsigned long pulseCount = 0, count = 0;

volatile unsigned long pulseNum,clockNum;

volatile unsigned long diffTime = 0;

volatile byte state = LOW;

void setup() {

Serial.begin(115200);

noInterrupts();

TCCR1A = 0;// set entire TCCR1A register to 0

TCCR1B = 0;// same for TCCR1B

TCNT1 = 0; //initialise counter value to 0

OCR1A = 65535; //sets the value

TCCR1B |= (1<< WGM12); //turn on CTC mode

TCCR1B |= (1<<CS10); //prescaler set to 1 (62.5nS resolution)

TIMSK1 |= (1 << OCIE1A); //sets up the interupt for timer1

interrupts();

attachInterrupt(digitalPinToInterrupt(3), stamp, RISING);

}

void loop() {

if (state == HIGH)

{

Serial.print((pulseNum\*65535)+clockNum); Serial.print(",");

state = LOW;

}

}

void stamp()

{

noInterrupts();

clockNum = TCNT1;

pulseNum = pulseCount;

state = HIGH;

pulseCount = 0;

TCNT1 = 0;

interrupts();

}

ISR(TIMER1\_COMPA\_vect){//perform interupt when timer1 hits 65535 or 4,096us (top of register)

pulseCount++;

}

clear;clc; close all;

load data1\_small\_sample.mat

load data1\_large\_sample.mat

edge = [-50:50];

N=length(data1\_small\_sample);

timing\_diff1 = zeros(N-1,1);

for x = 1:N-1

diff = data1\_small\_sample(x+1) - data1\_small\_sample(x);

timing\_diff1(x,1)=diff;

end

N=length(data1\_large\_sample);

timing\_diff2 = zeros(N-1,1);

for x = 1:N-1

diff = data1\_large\_sample(x+1) - data1\_large\_sample(x);

timing\_diff2(x,1)=diff;

end

figure

histogram(timing\_diff1,edge)

title('histogram - small sample size (2692)','fontSize',18);

total\_stddev = std(timing\_diff1)

total\_mean = mean(timing\_diff1)

total\_samples = length(timing\_diff1)

Z\_star = 2.5758

low\_CI = total\_mean-(Z\_star\*(total\_stddev/sqrt(total\_samples)))

high\_CI = total\_mean+(Z\_star\*(total\_stddev/sqrt(total\_samples)))

[H,P,CI] = ztest(timing\_diff1,total\_mean,total\_stddev,0.01)

figure

histogram(timing\_diff2,edge)

title('histogram - large sample size (9758)','fontSize',18);

total\_stddev = std(timing\_diff2)

total\_mean = mean(timing\_diff2)

total\_samples = length(timing\_diff2)

Z\_star = 2.5758

low\_CI = total\_mean-(Z\_star\*(total\_stddev/sqrt(total\_samples)))

high\_CI = total\_mean+(Z\_star\*(total\_stddev/sqrt(total\_samples)))

[H,P,CI] = ztest(timing\_diff2,total\_mean,total\_stddev,0.05)

count1 = 0;

for x = 1:length(timing\_diff1)

if (timing\_diff1(x) > 5 || timing\_diff2(x) < -5)

count1 = count1+1;

end

end

count2 = 0;

for x = 1:length(timing\_diff2)

if (timing\_diff2(x) > 3 || timing\_diff2(x) < -3)

count2 = count2+1;

end

end