Microcontroller Differential GPS to Subtract Signal Delay Due to Ambient Free Electrons in the Ionosphere

Diana Swanson
Senior Project, Physics Department. California Polytechnic State University
Winter and Spring 2019
Advisor: Dr. Tom Bensky

Purpose

The goal of this project is to create a Global Positioning System (GPS) receiver that is more precise than one GPS receiver on its own. The technique is to take the difference between a GPS receiver's measured position and its actual position, then use radio frequency (RF) communication to send that differential value to another microcontroller GPS receiver. This differential value will be added to the measured second location to get a more accurate position for the second GPS receiver, thus creating a differential GPS.

Introduction

Global Positioning System receivers are becoming increasingly common in daily life as smartphones and navigating using a GPS is becoming more widespread. The error in reported location by a GPS receiver is often on the order of meters. The time delay influencing the signal between the satellites and the receiver is most due to the ionosphere, the region of free electrons in our atmosphere about 50 km – 10,000 km above the earth. GPS systems use triangulation and a series of signals to determine the location of each receiver. Ground stations can determine the location of the receiver by tracking the orbits of GPS satellites using radio frequency signals, the GPS receiver receives synchronized orbital and time data sent out from at least four satellites. Based on the time the receiver senses the signal, the distance away from the satellites of the receiver can be calculated due to finite speed of information and the known sent and received times, and use triangulation using at least four satellites to determine the receiver location in latitude, longitude, height and time, see Figure 1.

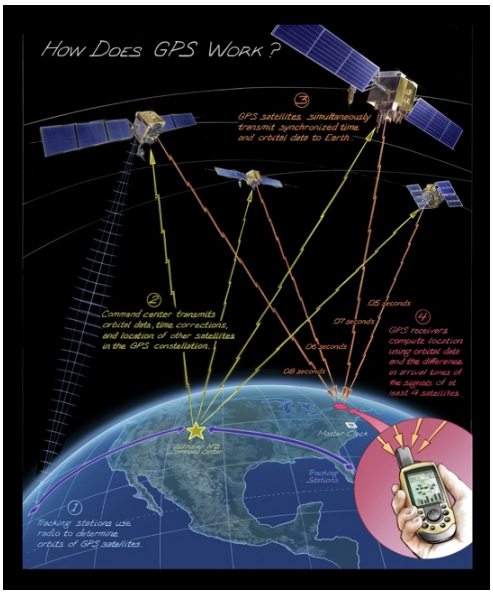

Figure 1: Simple GPS signal map

One of the main reasons for GPS location error is due to differing propagation delay times in the signal between the satellites and the receiver on the ground due to the ionosphere. The ionosphere is a region of ionized plasma with a varying density of ambient free electrons, (described by the total electron content, TEC). The larger the TEC, the larger the delay of the GPS signal through the ionosphere. The ionosphere TEC is known to vary heavily based on the sun's location, because the UV radiation coming from the sun is what ionizes the atmospheric gas molecules. When the earth is at perihelion the TEC is largest versus the smallest at aphelion (The Ionosphere Effect, 2018).

GPS operate at two common frequencies for satellites to send the signal to the ground. That frequency band is reserved for the GPS systems and transmissions are not allowed to use it. It is even possible for nearby electronics that are operating at a frequency that is a harmonic of the GPS frequency to interfere with the signal. For now the cause of the noise in the data is not of concern, the goal is to subtract it from the measured location of the second GPS receiver, thus creating a differential GPS.

Methods

The differential GPS works using two microcontroller GPS units. The first GPS receiver unit, called GPS_A for convenience, is placed at a known location. By placing it at a known location the measured location from the GPS_A unit can be subtracted from the true location to find the difference in the reported location and the actual. The difference of the measured from the actual location for GPS_A is sent to the microcontroller hooked up with GPS_B, the second GPS receiver unit, using radio frequencies with a transmitter and a receiver. The particular RF frequency used to transmit the data between the two microcontrollers was 434 MHz using a transmitter and receiver pair from SparkFun (GeckoStudios). The

microcontroller for GPS_B receives the value of the difference between GPS_A measured location and GPS_A actual location and adds it to the GPS_B measured location to output the true location of GPS_B. The assumption of this method is that the two GPS recievers are close enough to each other that they experience the same, or similar, signal delay and therefore the difference found between GPS_A and its actual location is the same difference GPS_B is experiencing.

The known location of GPS_A was found by taking the average of the measured location of the GPS unit in a constant location where it will stay over a period of one day. Assuming the interference creating the noise is random, this average of the data as the true location was sufficient for this purpose, the average latitude was 35.2901 degrees North, and longitude was -120.6281 degrees West for the location of GPS_A in San Luis Obispo, CA.

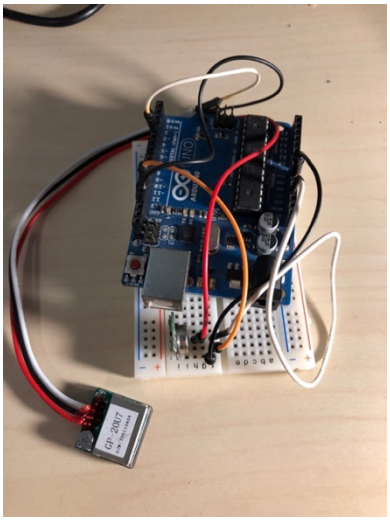

Figure 2: Arduino A with RF Transmitter

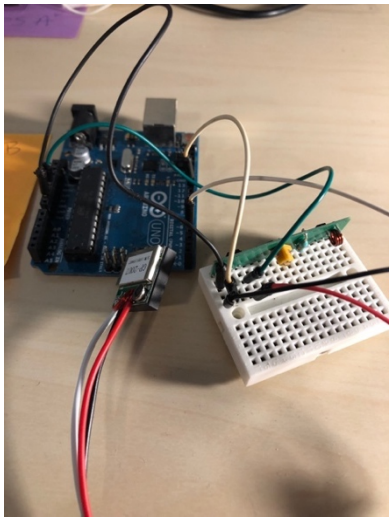

Figure 3: Arduino B with RF Reciever

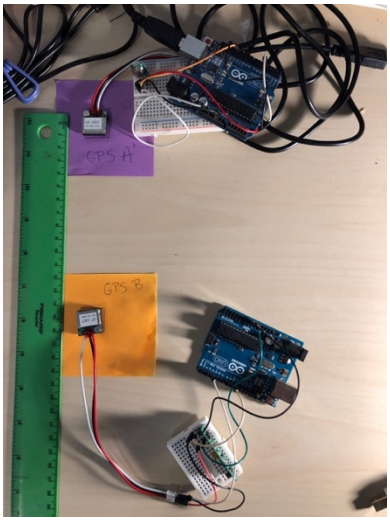

Figure 4: GPS_A at known location and GPS_B setup 15 cm apart

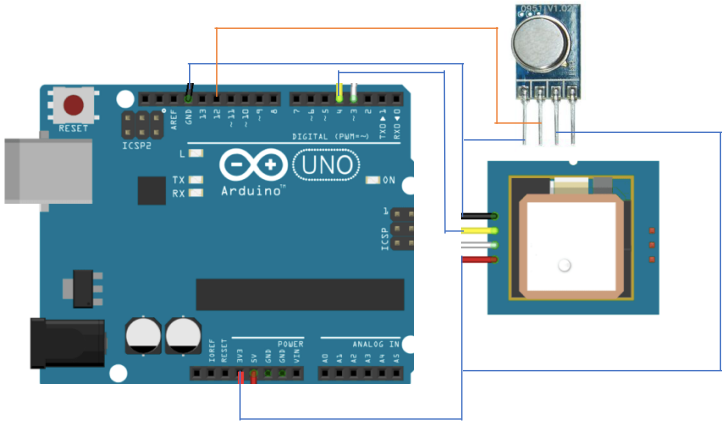

Diagram 1: Arduino A, GPS and RF transmitter

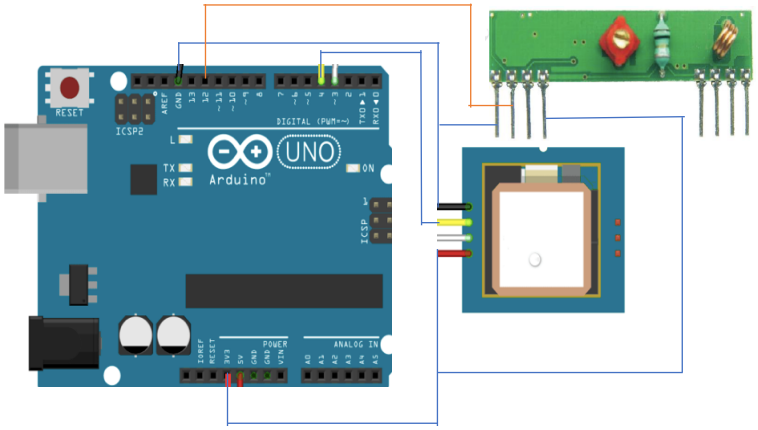

Diagram 2: Arduino B, GPS and RF receiver

Technical Details of the Code and Hardware setup

Diving into the code, a package of three scripts was created to work simultaneously to interact with the Arduino microcontrollers, GPS units, RF transmitter and receiver link, and plotting and analyzing the data. The first script was in Arduino language, and was uploaded to Arduino A that had the GPS_A unit and RF transmitter connected to its ports. This script was used to collect the GPS_A location and time data, subtract that from the hard-coded known location of GPS_A and send that to Arduino B connected up with GPS_B and the RF receiver. This script was labelled `transmitter_code.` Three Arduino libraries available from the open source Arduino devices community were used. The first library was TinyGPS++ (M. H.,2019), and was used to collect the GPS data from GPS_A. The TinyGPS++ library has some built in functions to read the output of the GPS units and save them as useful variables such as latitude, longitude, date and time. The GPS units output the raw data in a NMEA sentence that is very common for GPS units, and this library is able to read that. Using the TinyGPS++ library, this was edited by changing the pins the Arduino receives the GPS data from and the way it prints out the data to the serial monitor which will be explained later. Most of the data is outputted to the serial monitor so the SoftwareSerial library was used to allow this. The other important function of the `transmitter_code` script was to transmit the data from GPS_A to Arduino B. To do this the RH_ASK library from RadioHead (M.M, 2019) was used. This library was modified to create a struct of the data to send to Arduino B so multiple variables of data could be sent. This allowed the location, time, date and difference of GPS_A location from it's actual location to be sent to the B Arduino. After assigning the GPS_A data to the variables in the structure, the data was converted to bits and sent using the RF transmitter using a `driver.send` function.

Arduino B was the unit that was hooked up with the RF receiver and GPS_B. The code was again in Arduino language and called `receiver_code_3_formatlab,` and was very similar to the transmitter code. It used the TinyGPS++ library and collected the GPS_B location in the same way GPS_A data was collected in the transmitter_code. The data was outputted to the serial monitor in a CSV format so that data the data could be analyzed and plotted in a Matlab. The receiver code also defined the same variables in a structure named `myData`, exactly mirroring the transmitter code so the labels of the data in both scripts are the same. The library RH_ASK was used to receive the data by using the library built function `RH_ASK_MAX_MESSAGE_LEN,` to allow the memory to know the length of the data it was receiving was, and then to read in that data. The final goal of this project was to find the actual location of GPS_B. The code has the GPS_B reported location and was receiving the differential latitude and longitude values that GPS_A was experiencing. The differential values were added to the respective latitude and longitude values of GPS_B reported location values to produce the final_latitude, and final_longitude values of GPS_B.

The experimental setup consisted of Arduino A plugged into one serial port of a MacBook Air, with the GPS hooked into the 3.3 V power supply in the Arduino, ground, and port 4, and the RF transmitter hooked up to the power, ground, and port 12 for the data signal, see Figure 2. When data was printed to the serial monitor using the `transmitter_code,` the data was being written to the serial monitor it was hooked up to. Arduino B was plugged into a different serial port, and the GPS was hooked up to power, ground and data pin 4, and the RF receiver was hooked up to power, ground and pin 12 for data transmission, see Figure 3.

With the hardware plugged in and connections made, and the respective Arduino codes uploaded to the devices, the data then needed to be read and analyzed. The data was analyzed using a script in Matlab which opened the serial ports on the computer for which the Arduinos were plugged into and read in the data that was being printed there by the uploaded code on the Arduinos. This is why the format of the data that was printed to the serial monitor was so important. The Matlab script named `real_arduino_serial_output.m` used a loop to read in the CSV data from the serial monitor so the user can specify how many data points to read in, and in the Arduino codes, the user can decide the delay time between readings.

Once the data is read into Matlab and defined as the variable `data`, the serial port was closed to maintain efficiency. This sacrificed completely real time data, but sped up the process significantly rather than keeping the serial port open continuously. The data variable is a vector that was then spliced based on the commas that were written between the values sent to the serial port by the Arduino code. Once the data was spliced it was stored as latitude, longitude, date, and time variables that could then be manipulated and plotted in Matlab, (see plots in the results section). The script was split into three

sections, one to read the serial port from Arduino A and plot the position of GPS_A without the differential being subtracted off, one to read in the serial port from Arduino B and plot the position of GPS_B after the differential was subtracted, and one to calculate the distance between the GPS units after the differential values were subtracted from GPS_B location.

Results

The initial plots showed a lot of structure in the GPS_A signal. When the GPS receiver was at a constant location the measured location varied on the order of tens to thousands of meters. All of the data was taken in North America, mostly in San Luis Obispo, California and the units of the latitude and longitude measurements are in degrees north and west respectively. As seen in the initial plots of GPS_A data the measured location of the GPS unit varies a lot. Figures 7 and 9 (and Appendix A) were all taken while the GPS was in one location in a time frame of less than 10 minutes. The pattern, or lack thereof, of the measured location is intriguing. Some of the plots seem to indicate very complex dynamics in the ionosphere that are creating signal delays in the GPS signal and therefore location measurement errors. Other plots seem to show the GPS location changing based on a step function or changing a lot in one dimension but staying much more constant in the other. While the goal of this project is to subtract out this variation, not caring what it is, this data is very interesting, and physical descriptions of the dynamics and variables at play should be studied further.

The plots of GPS_A location varied a lot, but the goal is to pinpoint the location of GPS_B. The results were mixed in this regard. The final location of GPS_B often still varied, but sometimes the range of the known location was smaller than the variation from GPS_A on the order of tens of meters. To understand if the final location of GPS_B was correct, the distance between the two GPS receivers location was calculated and converted into meters. When GPS_A was at its known location on a desk, the location of GPS_B relative to GPS_A on the desk was measured. As seen in Figures 8 and 10, the distance in meters between the devices was measured to be on an order of tens of meters in the latitude direction and hundreds of meters in the longitude direction. This was when the receivers were about 15 centimeters apart on the desk stationary during each trial. This means the differential GPS was not working as expected and the precision was still on the order of meters, not centimeters as hoped for.

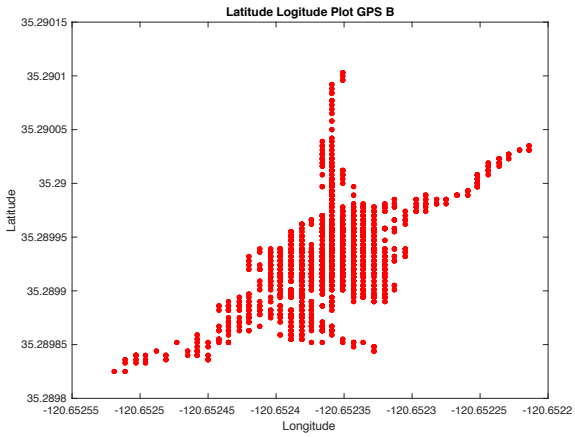

Figure 5: GPS_B location on 20190531, 10000 data points

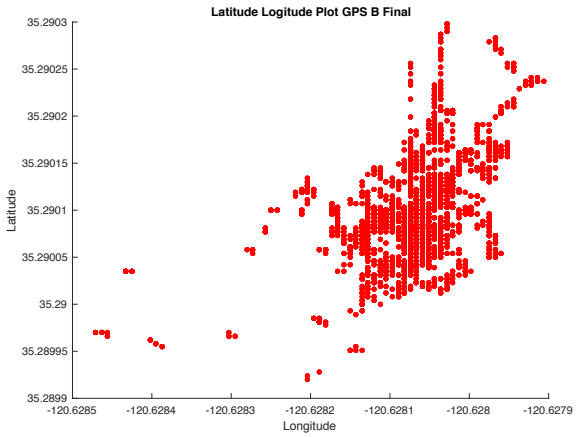

Figure 6: GPS_B final location on 20190531, 10000 data points

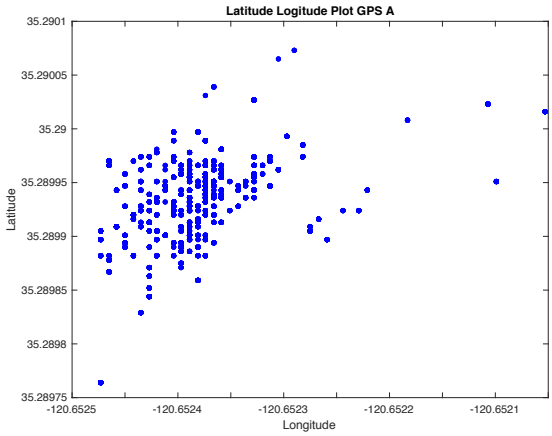

Figure 7: GPS_A location on 20190531, 10000 data points

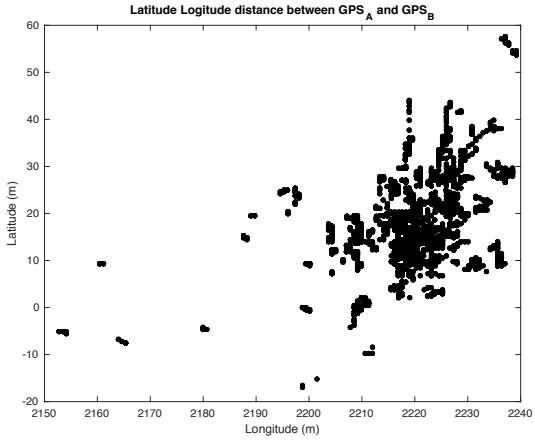

Figure 8: Difference between GPSes on 20190531, 10000 data points

One reason for the error in these measurements and lack of precision is due to the delay time in sending the signal using the RF link. The differential GPS needs to subtract the differential value in real time from the location of GPS_B, but there was a delay of about 1s for the differential values to be sent to the GPS_B Arduino so the differential value from the second before was being subtracted off the GPS_B location. The one second delay was expected to be negligible, but that may not be the case. Further research and trial with the data is needed to see if changing the method of subtracting off the differential value is necessary.

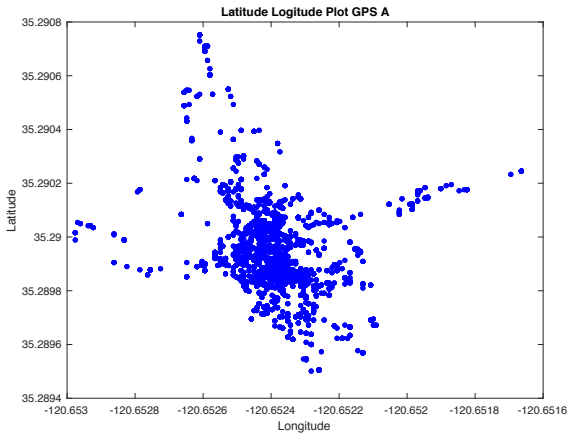

Figure 9: GPS_A location on 20190531, 10000 data points

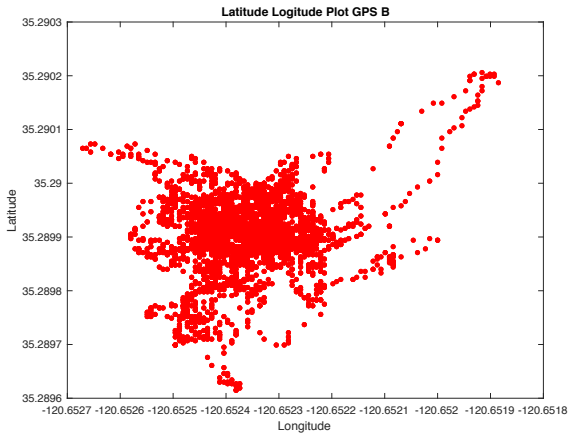

Figure 10: GPS_B location on 20190531, 10000 data points

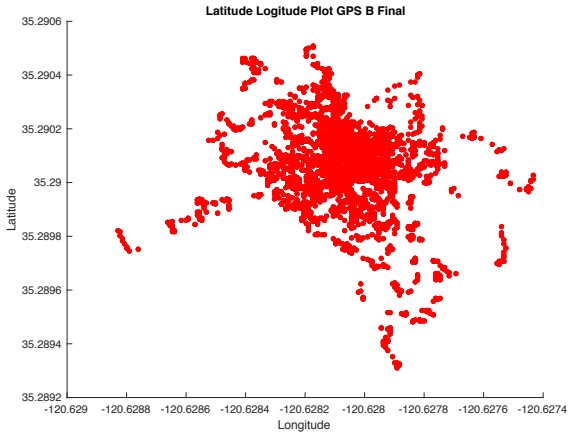

Figure 11: GPS_B final location on 20190531, 10000 data points

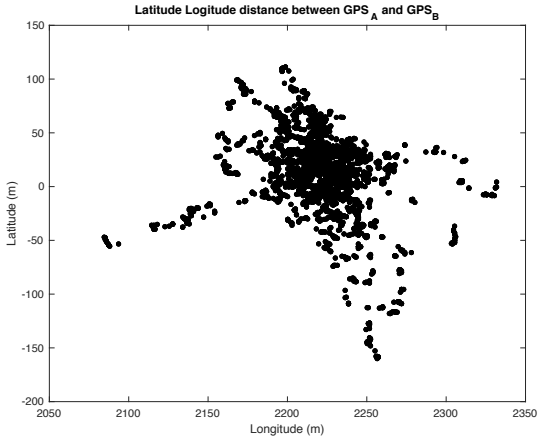

Figure 12: Difference between GPSes on 20190531

Another reason for the error was well illustrated through the data run of 100,000 points of continuous data without moving either GPS receiver. To collect 10,000 data points takes about 15 minutes, to get 100,000 data points the GPS units were left in the same location for almost a full day. The plots showed the variation in the measured location of GPS_A to be a very different pattern than the variation in the measured location of GPS_B. This is an intriguing result as the GPS receiver units that are roughly 15 cm apart on a desk are receiving signals from satellites that are sending signals though the same columns of the ionosphere assuming they are contacting the same satellites. Another interesting part of this data collection run was that the distance between the GPS_A and GPS_B measured data was much more precise as see in Figure 17 than the calculated distance between GPS_A and the final location of GPS_B, Figure 16. Looking at the data it seemed that adding the differential value was skewing the data too far, almost over correcting. The most likely cause of this error is the true location of GPS_A was not known

well enough and was too high of a value in the longitudinal direction. This caused the final GPS_B calculated location to be farther to the west than it actually was.

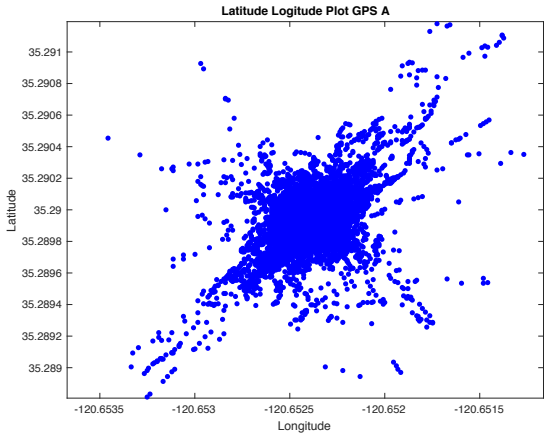

Figure 13: GPS_A location on 20190601, 100000 data points

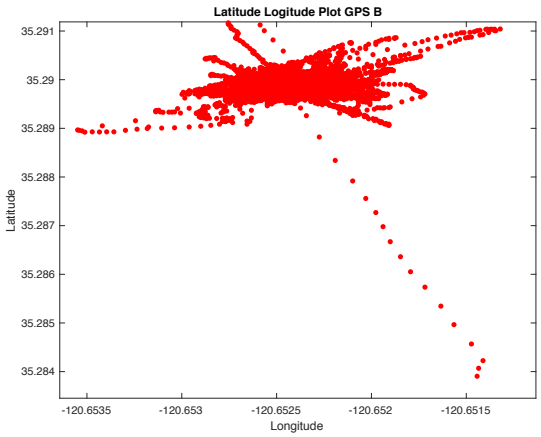

Figure 14: GPS_B location on 20190601, 100000 data points

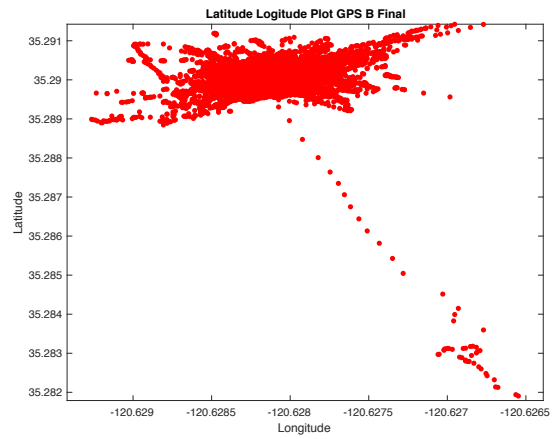

Figure 15: GPS_B final location on 20190601, 100000 data points

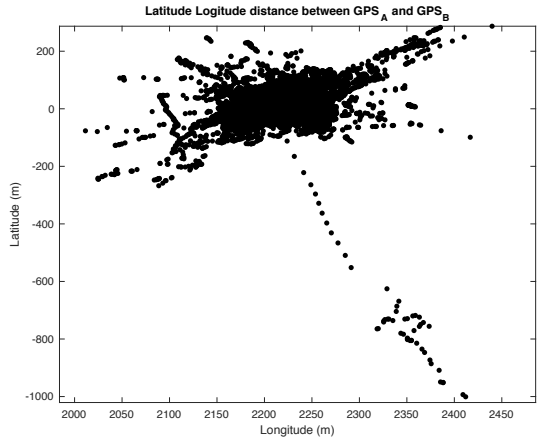

Figure 16: Difference between GPSes on 20190601

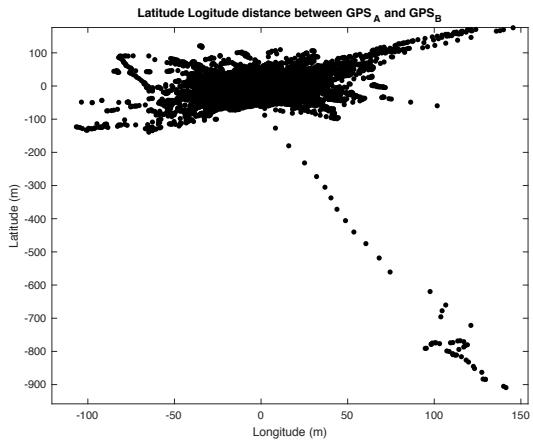

Figure 17: Difference between GPS_A and GPS_B raw data on 20190601

Conclusion

The differential GPS was able to collect data using GPS_A, calculate a differential value from the true location, transmit that value using an RF link to a second Arduino that gathered location data for GPS_B and add the differential value back to determine it's true location. The true location of GPS_A receiver was found by taking data for about a day and averaged as a true location of the receiver, this could have been more precise by using multiple days and more data to determine the true location, or using multiple GPS devices to reduce any bias due to the devices used.

Further work is required to finish the differential GPS, and to rigorously dive into the details of the interference the GPS receivers are experiencing, likely due to free electrons in the ionosphere. The NMEA data sent from the GPS units include the orbits of the GPS satellites. This could be used to determine the location of the satellites and therefore the path of the signal though the ionosphere. Knowing the path could lead to an understanding of how the ionosphere affects the data the GPS receiver is receiving. When the satellites are near the horizon the signal is traveling through a thicker part of the ionosphere so can lead to more signal error (The Ionosphere Effect, 2018). Further work should also draw conclusions about the physical processes leading to the dynamics of the signal.

This differential GPS method was not able to precisely determine the final location of the GPS_B receiver when it was stationary. Taking data over the span of multiple days will help create a precise true location and better differential GPS.

Appendix A: Interesting plots

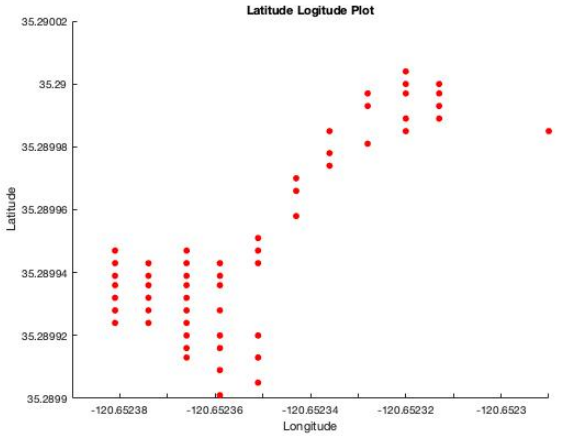

Figure 18: 20190426, 1000 data points in SLO, CA GPS_A.

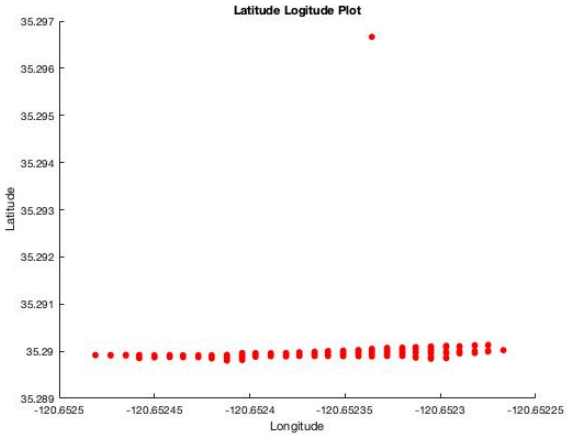

Figure 19: 20190426, 10000 data points in SLO, CA GPS_A

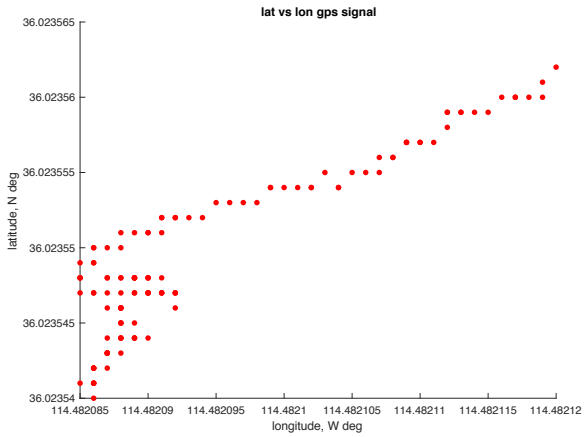

Figure 20: 20190302, 1000 data points in Boulder City, NV GPS_A.

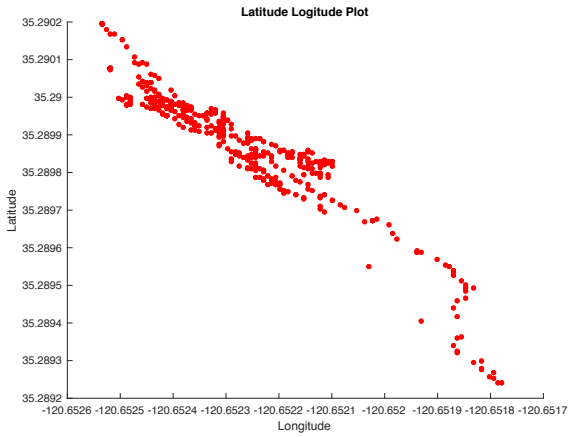

Figure 21: 20190514, 1000 data points in SLO, CA GPS_A

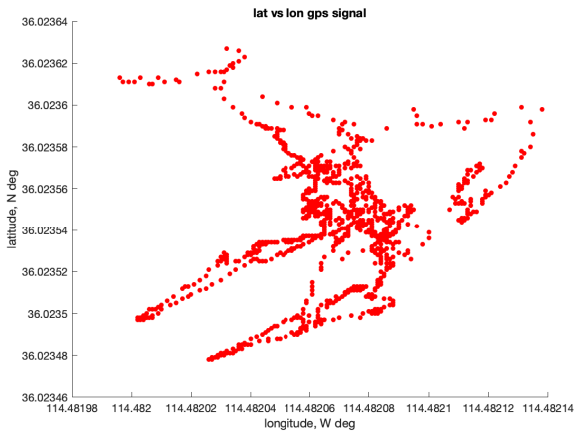

Figure 22: 20190302, 1249 data points Boulder City, NV GPS_A.

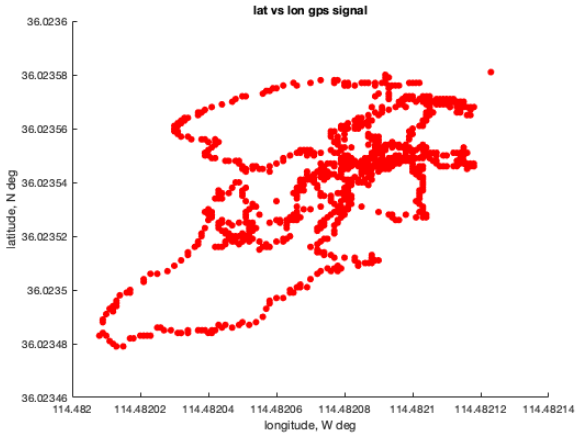

Figure 23: 20190302, 1000 data points Boulder City, NV GPS_A

Appendix B: Code

`Transmitter_code:`

```
// ask_transmitter.pde
#include <RH_ASK.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#ifdef RH_HAVE_HARDWARE_SPI
#include<RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile
#endif

RH_ASK driver;
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
```

```
TinyGPSPlus gps;
// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

//define data structure with variables to be sent
struct dataStruct{
  float la_diff ;
  float lo_diff ;
  float la ;
  float lo ;
  float hr  ;
  float minn ;
  float sec ;
  float monthh ;
  float dayy ;
  float yearr ;
  unsigned long counter;
}myData;

byte tx_buf[sizeof(myData)] = {0};

void setup()
{
  Serial.begin(9600);
  if (!driver.init())
     Serial.println("init failed");

    Serial.begin(9600);   // Debugging only
    ss.begin(GPSBaud);
   // Serial.println(F("Location | Latitude | Longitude | Lat_diff |
Long_diff | Date | Time |"));
}

void loop()
{
    // This sketch displays information every time a new sentence is
correctly encoded.
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
}

void displayInfo()
{
  if (gps.location.isValid())
  {
    ///hard coded location of GPS_A
    float latitude_avg = 35.2901 ; //35.2899 ;
    float longitude_avg = -120.6281; //-119.5207 ;

    Serial.print(gps.location.lat(),6);
    Serial.print(F(","));
```

```
    Serial.print(gps.location.lng(), 6);
    Serial.print(F(","));
    Serial.print(latitude_avg-gps.location.lat(),6);
    Serial.print(F(","));
    Serial.print(longitude_avg-gps.location.lng(),6);
    Serial.print(F(","));
    myData.la=gps.location.lat();
    myData.lo=gps.location.lng();
    myData.la_diff=(latitude_avg-gps.location.lat());
    myData.lo_diff=(longitude_avg-gps.location.lng());
  }
  else
  {
    Serial.print(F("INVALID"));
    Serial.print(F(","));
    Serial.print(F("INVALID"));
    Serial.print(F(","));
    Serial.print(F("INVALID"));
    Serial.print(F(","));
    Serial.print(F("INVALID"));
    Serial.print(F(","));
  }

 //  Date/Time:
  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
    Serial.print(gps.date.year());
    Serial.print(F(","));

    myData.yearr=gps.date.year();
    myData.dayy=gps.date.day();
    myData.monthh=gps.date.month();
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  if (gps.time.isValid())
  {
    if (gps.time.hour() < 10) Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
    Serial.print(gps.time.second());
    Serial.print(F("."));
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());

    myData.hr=gps.time.hour();
    myData.minn=gps.time.minute();
```

```
    myData.sec=gps.time.second();
  }
  else
  {
    Serial.print(F(","));
    Serial.print(F("INVALID"));
  }

  Serial.println();
///convert to bits and send data
  memcpy(tx_buf, &myData,sizeof(myData));
  byte zize=sizeof(myData);
  driver.send((uint8_t *)tx_buf, zize);
    //delay(10);
}
```

Receiver Code:

```
// ask_receiver.pde

#include <RH_ASK.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#ifdef RH_HAVE_HARDWARE_SPI
#include <SPI.h> // Not actually used but needed to compile
#endif

RH_ASK driver;

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

struct dataStruct{
  float la_diff ;
  float lo_diff ;
  float la ;
  float lo ;
  float hr  ;
  float minn ;
  float sec ;
  float monthh ;
  float dayy ;
  float yearr ;
  unsigned long counter;
}myData;

void setup()
{
  Serial.begin(9600);
      if (!driver.init())
      Serial.println("init failed");
    Serial.begin(9600);    // Debugging only
    ss.begin(GPSBaud);
```

```
}

void loop()
{
////rf code
  uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
  uint8_t buflen = sizeof(buf);

  //memset (buf,0,sizeof(buf));
  memcpy(&myData, buf, sizeof(myData));

  if (driver.recv(buf, &buflen)) // Non-blocking
   {
    int i;
// Message with a good checksum received, dump it.
  memcpy(&myData, buf, sizeof(myData));
   }
    while (ss.available() > 0)
    if (gps.encode(ss.read()))
      {displayInfo();
    }
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true)
    }
}
void displayInfo()
{
  //// gps data
  if ( double(gps.time.second()) == (myData.sec+double(1)) )
  {
  if (gps.location.isValid())
  {   // Serial.println(F("Location | Latitude | Longitude | Date | Time |
Final_lat | Final_lon | GPSA lat | GPSA lon"));
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
    Serial.print(F(","));
  }
  else
  {
    Serial.print(F("INVALID_B"));
    Serial.print(F(","));
    Serial.print(F("INVALID_B"));
    Serial.print(F(","));
  }

  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
    Serial.print(gps.date.year());
    Serial.print(F(","));
  }
  else
```

```
  {
    Serial.print(F("INVALID_B"));
    Serial.print(F(","));
  }

  if (gps.time.isValid())
  {
    if (gps.time.hour() < 10) Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
   Serial.print(gps.time.second());
    Serial.print(F("."));
   if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());
    Serial.print(F(","));
  }
  else
  {

    Serial.print(F("INVALID_B"));
    Serial.print(F(","));
  }
  //delay(200);
  ////////////////////subtract off difference from raw gps data
  float final_lat = (((myData.la_diff)+gps.location.lat()));
  float final_lon = (((myData.lo_diff)+gps.location.lng()));
  //Serial.print("final Location B :");
  Serial.print(final_lat,6);
  Serial.print(",");
  Serial.print(final_lon,6);

//////////////////GPSA

    Serial.print(",");
    Serial.print(myData.la,6);
    Serial.print(",");
   Serial.println(myData.lo,6);
}
}
```

Matlab code:

```
%%
%read data from serial port
sB = serial('/dev/tty.usbmodem14101', 'BaudRate', 9600)%, 'Terminator', 'CR')

fopen(sB);
iB=1;
dataB = cell(0);
v=100000;
while(iB<=v) % Read # lines
    dataB{iB}=fgets(sB);
   iB=iB+1; % Increment the counter
```

```
end

fclose(sB); % Disconnect the serial port object from the device
delete(sB); % Delete the serial object

%split the string into matrix
for k=1:length(dataB)
        C_b(k,:)=strsplit(dataB{k},',');
end

%%%%%%%%get rid of INVALID
k1=1;
for j=1:length(C_b)
    w=C_b{j,1};
    if  w ~=  'INVALID_B'
       C_B(k1,:)=C_b(j,:);
       k1=k1+1;
    end

end
exist C_B  ;
if ans==1


%conversion from deg to centimeters at 40deg lat
deg_lat=110941; %m
deg_long=91288; %m
[m_B,n_B]=size(C_B);

for i=1:m_B

     final_la_B(i)=str2num(C_B{i,5});
     final_lo_B(i)=str2num(C_B{i,6});
     la_a(i)=str2num(C_B{i,7});
     lo_a(i)=str2num(C_B{i,8});
     y_dist_between=(final_la_B-la_a)*deg_lat;
     x_dist_between=(final_lo_B-lo_a)*deg_long;
     la_B(i)=str2num(C_B{i,1});
     lo_B(i)=str2num(C_B{i,2});

end
figure(1)
plot(final_lo_B,final_la_B,'r.', 'markersize', 14)
title("Latitude Logitude Plot GPS B Final");
xlabel('Longitude');
ylabel("Latitude");

figure(2)
plot(lo_B, la_B,'r.', 'markersize', 14)
title("Latitude Logitude Plot GPS B");
xlabel('Longitude');
ylabel("Latitude");

figure(3)
plot(lo_a, la_a, 'b.', 'markersize', 14)
title("Latitude Logitude Plot GPS A");
xlabel('Longitude');
ylabel("Latitude");
```

```
figure(4)
plot(x_dist_between,y_dist_between, 'k.', 'markersize', 14)
title("Latitude Logitude distance between GPS_A and GPS_B");
xlabel('Longitude (m)');
ylabel("Latitude (m)");

else
    r1='invaild GPS data try again'
end
```

References

1. Abdullah, M. (2003). Accurate ionospheric error correction for differential GPS. *Twelfth International Conference on Antennas and Propagation (ICAP 2003)*. doi:10.1049/cp:20030034

2. GeckoStudios. (n.d.). RF Link Transmitter - 434MHz. Retrieved from https://www.sparkfun.com/products/10534

3. Kintner, P. M., Ledvina, B. M., & Paula, E. R. (2007). GPS and ionospheric scintillations. *Space Weather,5*(9). doi:10.1029/2006sw000260

4. M. H. (2019). TinyGPS. Retrieved from http://arduiniana.org/libraries/tinygpsplus/

5. M. M. (2019). RadioHead Packet Radio library for embedded microprocessors. Retrieved from https://www.airspayce.com/mikem/arduino/RadioHead/index.html

6. Sarah Nik Zulkifli, Siti & Abdullah, Mardina & Ismail, Mahamod. (2012). Simulation of ionospheric error on GPS signals due to changes in the direction of mobile station positions in differential GPS. Annals of Geophysics. 54. 10.4401/ag-4973.

7. S. D. (2015, April 05). Send temperature using RadioHead. Retrieved from https://forum.arduino.cc/index.php?topic=313587.0

8. Time and Navigation. (n.d.). Retrieved from https://timeandnavigation.si.edu/multimediasset/how-does-gps-work proper citation to be made

9. The Ionospheric Effect. (2018). Retrieved from https://www.e-education.psu.edu/geog862/node/1715