06/28/2023
Wesley Caoba

Setting up an RTK Base and Rover System

Overview

The ZED-F9P is able to operate in two modes. One as a base and one as a receiver. In the base mode, the ZED-F9P assumes that it is not moving. This allows it to watch satellites moving overhead and compute a correction value for the data it recieves. This correction value is in the RTCM format.
In the "rover" or receiver mode, the zed-f9p is calculating its own position based on the satellite signals it is receiving. It is able to take in RTCM correction data from various sources that are operating as a base.

Objective

Create a base station that transmits RTCM data over LoRa. Use the RTCM data on a rover device to track GPS locations.

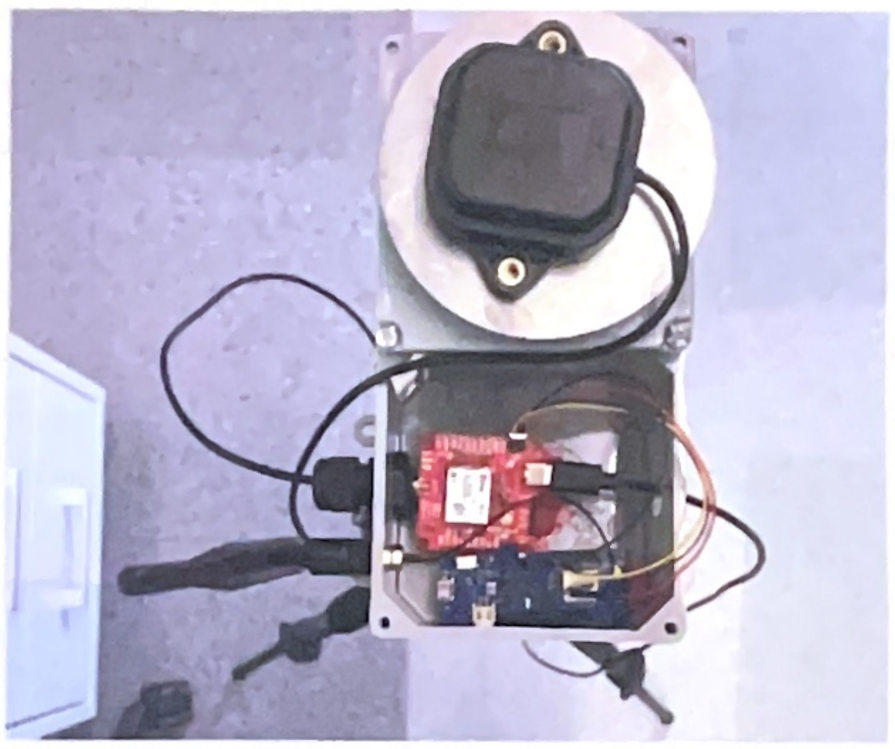


Figure 1
Our base station. This is a ZED-F9P connected to a 1310 via I2C. It sits in a junction box on top of a tripod. The USB connection is used to initiate a Survey-in with U-center on a laptop.

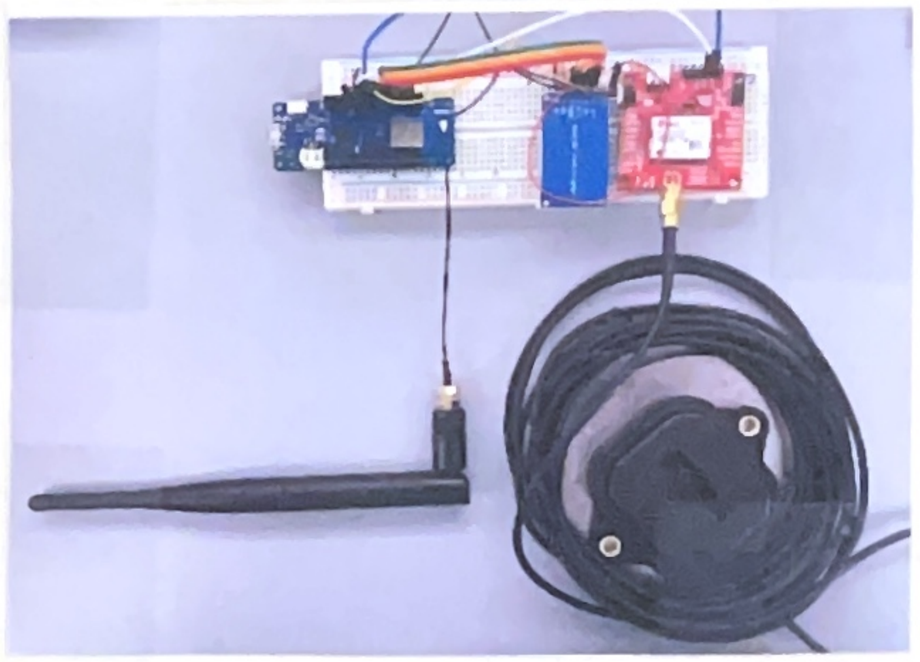


Figure 2
our "Rover" GPS tracker.

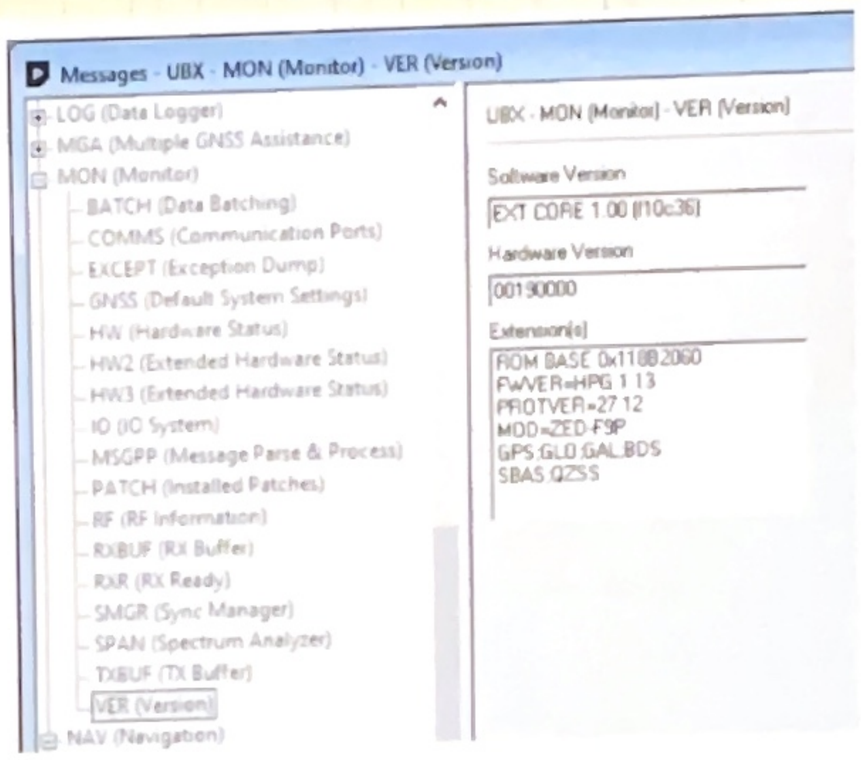
The hardware serial is connected to the UART2 of the zed-FP9. The SD card module is using SPI

74 | Setting up an RTK Base and Rover system | 06/28/2023 Wesley Cooper

To configure the base station, I followed a guide from sparkfun's website. Ive included the peices that I found useful.

Quick note: You will see the terms *GPS* and *GNSS* used interchangeably throughout this tutorial. GNSS, or Global Navigation Satellite System, is the collective term for all GPS (USA), GLONASS (Russia), BeiDou (China), and Galileo (EU) satellites. GPS was the predominant constellation up to about 2017. After this time enough BeiDou satellites were functional and enough GLONASS satellites were transmitting additional bands (L2, etc) that advanced receivers are now designed to receive signals from a variety of constellations rather than *just* GPS. This is a very good thing. With more satellites and more frequencies we can improve accuracies greatly. Said differently, you don't own a *GPS* receiver, you own a *GNSS* receiver. Congrats!

Please make sure both base and rover receivers have the most up to date firmware from u-blox. As of writing, there has been a new v1.30 firmware release. u-blox provides a simple to use upgrade tool inside of u-center. You can find the latest firmware for all u-blox receivers on their website under the documentation and resources tab. u-blox has been making some good improvements for the ZED-F9P from v1.12, to v1.13, to v1.30. We can now receive from EGNOS!



Firmware version 1.13

To check your firmware, open the messages window. Shrink the NMEA branch by double clicking on 'NMEA'. Expand the UBX branch and look for 'MON' or monitor. Expand MON and look for VER (last on the list). FWVER will indicate your current firmware version.

06/28/2023
Wesley Cooke

Setting up an RTK Base and Rover System

75

Setting up a Temporary Base

A basic GNSS antenna on a tripod

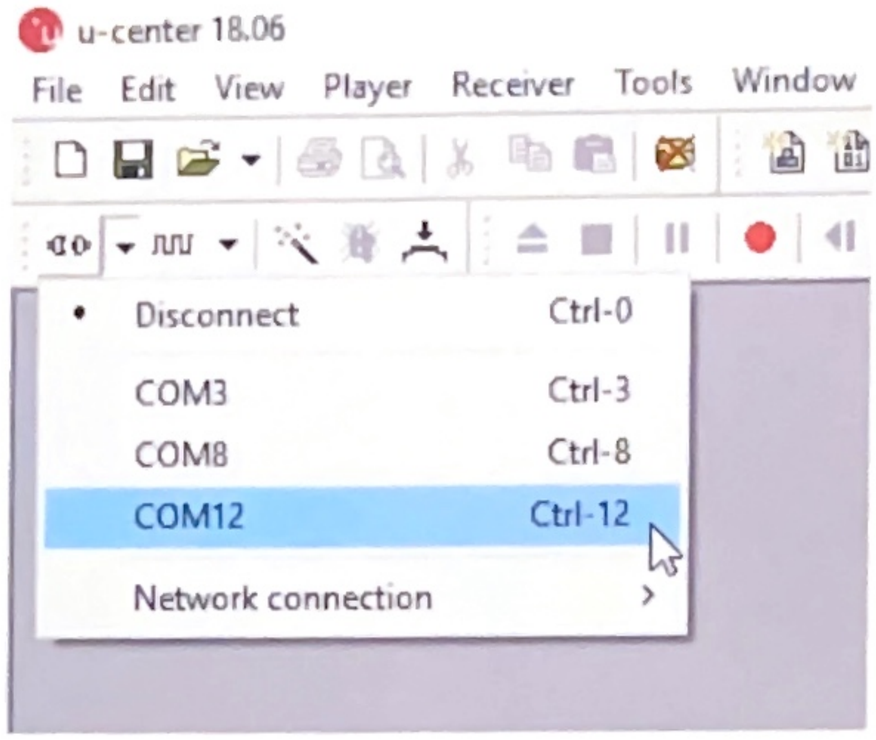
The first step in a base/rover setup is setting up a base station. There are a few ways to set up a base station: a temporary base station is faster but less precise. A static base station takes more time to set up and configure but provides the greatest precision. There is a third option and that is getting your correction data from a public feed. That option is covered in this tutorial and so we will not cover public correction data in this tutorial.

We've covered setting up a temporary base station in a previous tutorial but we will go a bit more in depth here. Additionally, this tutorial will cover the settings required for the ZED-F9P (rather than the previous generation NEO-M8P).

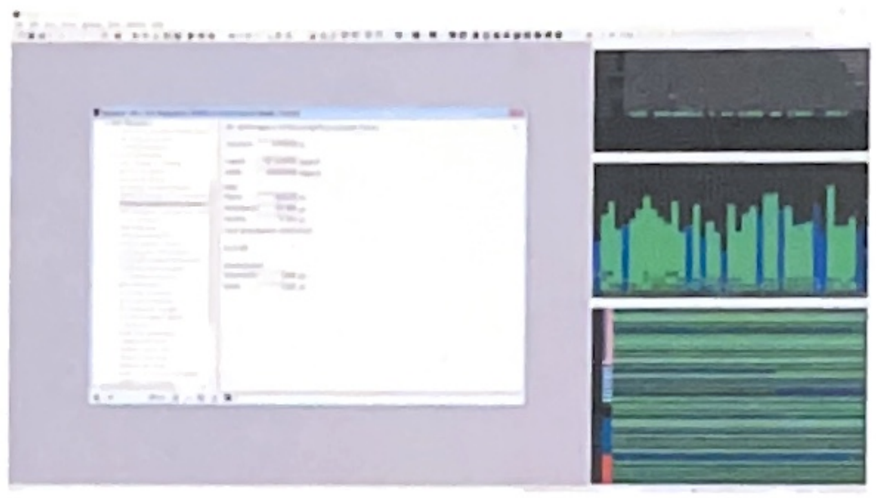
Above is the u-blox antenna attached to the ground plate, on top of a tripod, with SMA cable running indoors to the receiver. This is a great setup for experimenting and for short trips to the field. The purpose of a GNSS RTK base is to not move. Once you tell a receiver that it is a base station (ie, not moving) it will begin to look at the satellites zooming overhead and calculate its position. As the environment changes the signals from the GNSS (Global Navigation Satellite System - the collective term for GPS, GLONASS, Beidou, Galileo satellites) network change and morph. Because the base knows it is motionless, it can determine the disturbances in the ionosphere and troposphere (as well as other error sources) and begin to calculate the values needed to correct the location that the satellites are reporting to the actual location of the base station. These values are unsurprisingly called 'correction values' and are encoded into a format called RTCM. (RTCM stands for Radio Technical Commission for Maritime but is just a name for a standard akin to "802.11". Don't worry about it.). You will often see the term RTCM3 being used; this is simply correction data being transmitted using version 3 of the RTCM format.

Note: The accuracy of the base's location will be reflected in the accuracy of the rover's position. Said differently, if the base is incorrectly recorded 1m away from its actual location than all the rover readings will be exactly 1m wrong. It's really important to locate the antenna with a clear view of the sky and to obtain an accurate base station location.

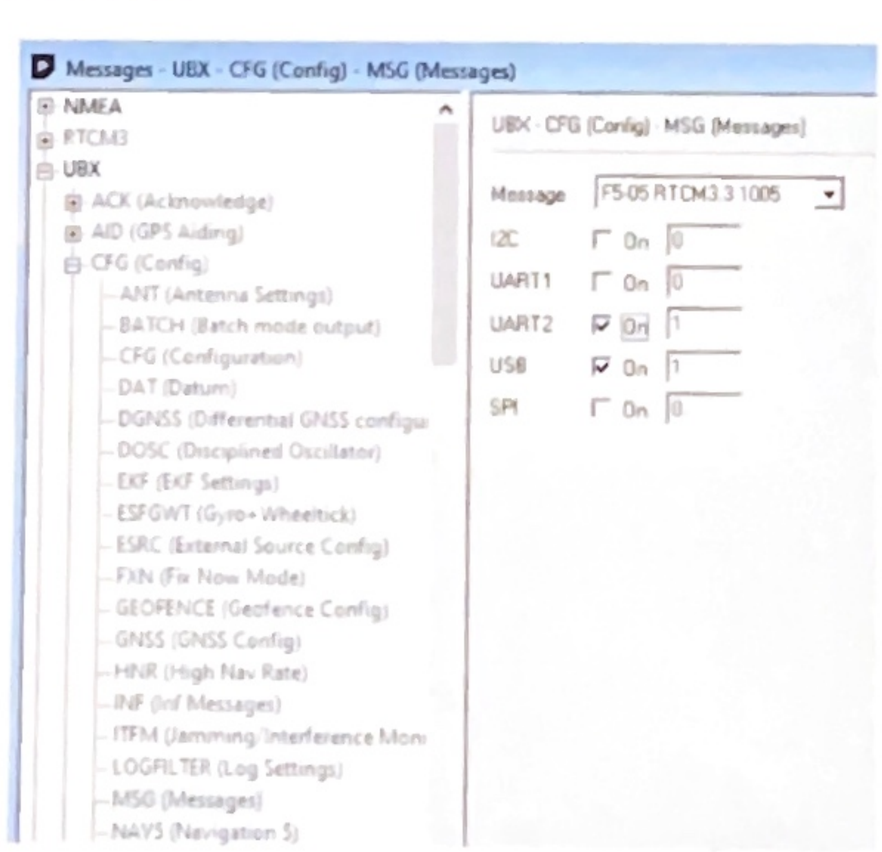
Open u-center and connect to the corresponding COM port that Windows created when the board was connected.



Having Problems With COM Ports? Please checkout our tutorial on u-center specifically. It will walk you through driver install and determining which COM port is the receiver.



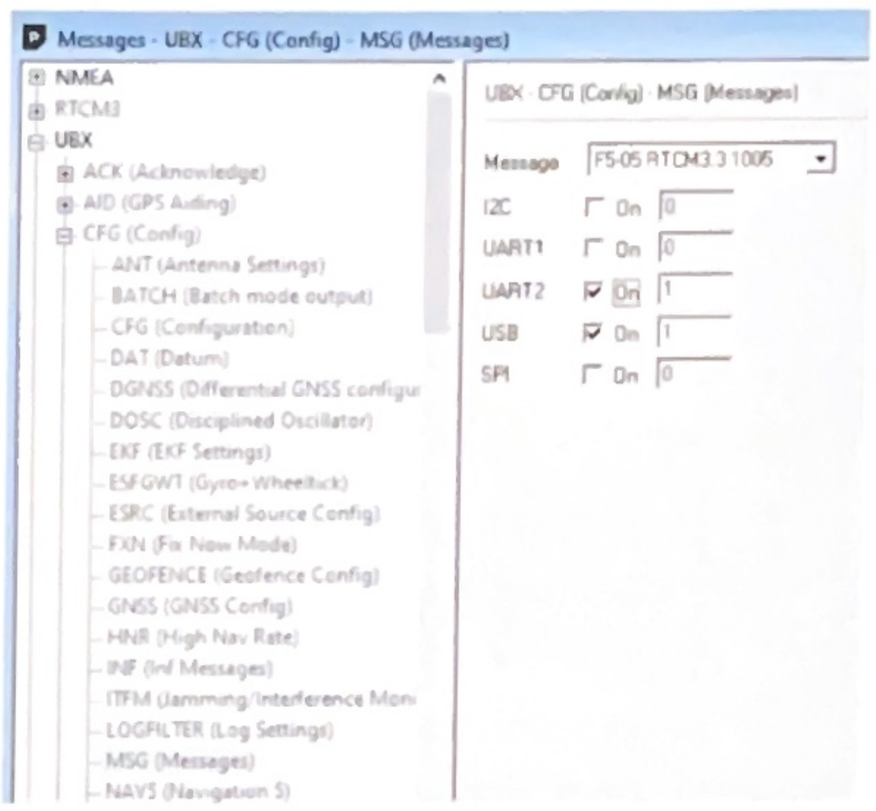
You should begin to see things change and data flowing through. Above shows the ZED-F9P running for a few minutes obtaining a heap of satellites and achieving 0.9m horizontal accuracy. Let's open the Messages window and begin configuring the module.



First, close the NMEA branch to get it out of the way. Open the UBX branch and then the CFG configuration branch. The number of configuration options can be overwhelming at first but over time they can become quite

helpful. We are looking for **MSG**.

Take this moment and ask yourself how you will be transmitting the correction data from the base to the rover. The correction data, or RTCM, varies in size but is approximately 1 to 2k bytes per second. Tiny compared to downloading a song or video, but just large enough that LoRa and smaller packet radios may not be able to handle it. We recommend using a Serial Telemetry Radio. Running at 915MHz, these radios can hit ~300m out of the box at 100mW with a 2km/500mW option available as well. The radios act as a transparent serial passthrough at 57600bps (out of the box). We've found them to be just about perfect for Base to Rover RTCM communication with the only limitation being distance. If you need miles of range, you will need to get your HAM radio license and crank the power. But if you're just getting started with RTK, these radios are great!



We are going to assume a telemetry radio will be attached to UART2 of the ZED-F9P. From u-center Messages window, enable the following messages for both UART2 and USB. *

- RTCM3.3 1005
- RTCM3.3 1074
- RTCM3.3 1084
- RTCM3.3 1094
- RTCM3.3 1124
- RTCM3.3 1230 x 5 (Enable message every 5 seconds)

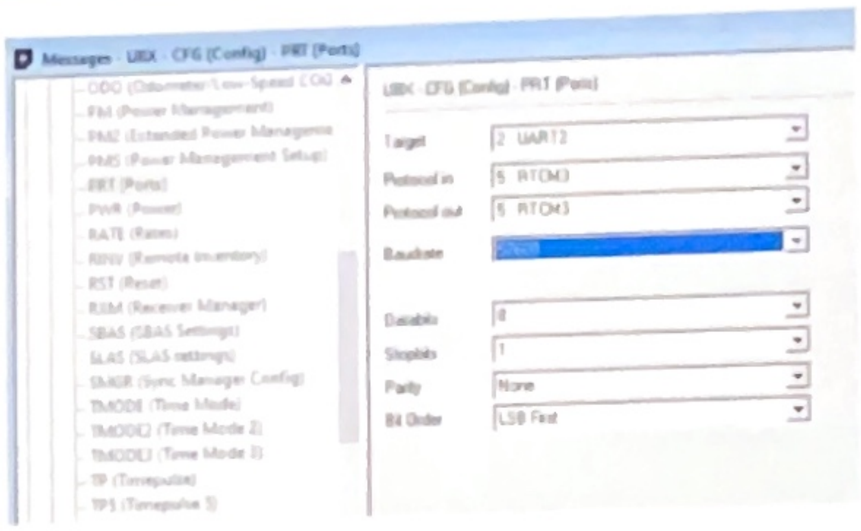
As you set each setting, you will need to press the 'Send' button. It doesn't hurt to press the 'Poll' button to verify the setting has stuck.

We recommend enabling these messages for both USB and UART2 as shown in the photo. This will tell the module to begin transmitting just these RTCM sentences (message types) once the module has completed its survey. UART2 is best used for sending serial RTCM in/out. USB is enabled so that we will be able to see and verify that RTCM messages are actually being output by the module.

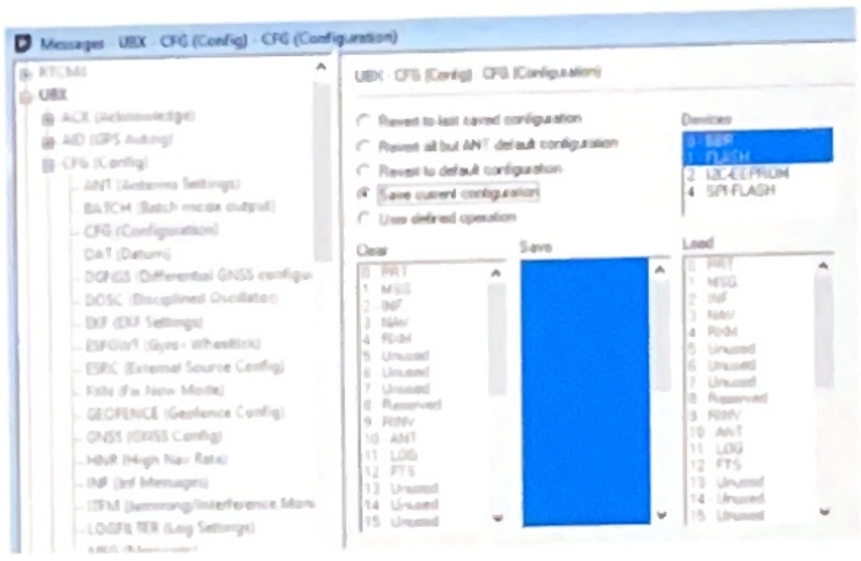
* Instead of using UART 2, I enabled the messages on I2C. This means the messages will appear in the I2C traffic. It was more convenient to wire this way.

78 Setting up an RTK Base and Rover System

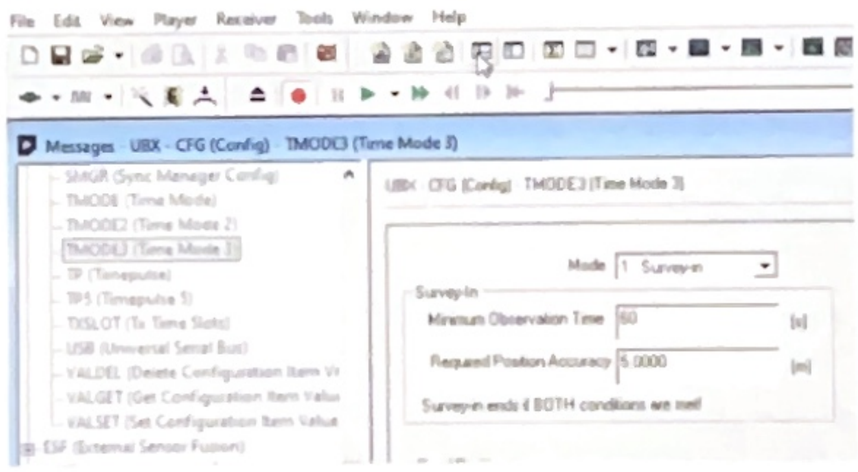
06/28/2023
Wesley Cook



Next, set UART2 to the appropriate baud rate. Scroll down to the 'PRT' or port settings. Select UART2. By default it is 38400bps and the radios expect 57600bps. Increase the speed. Remember to press 'Send'.

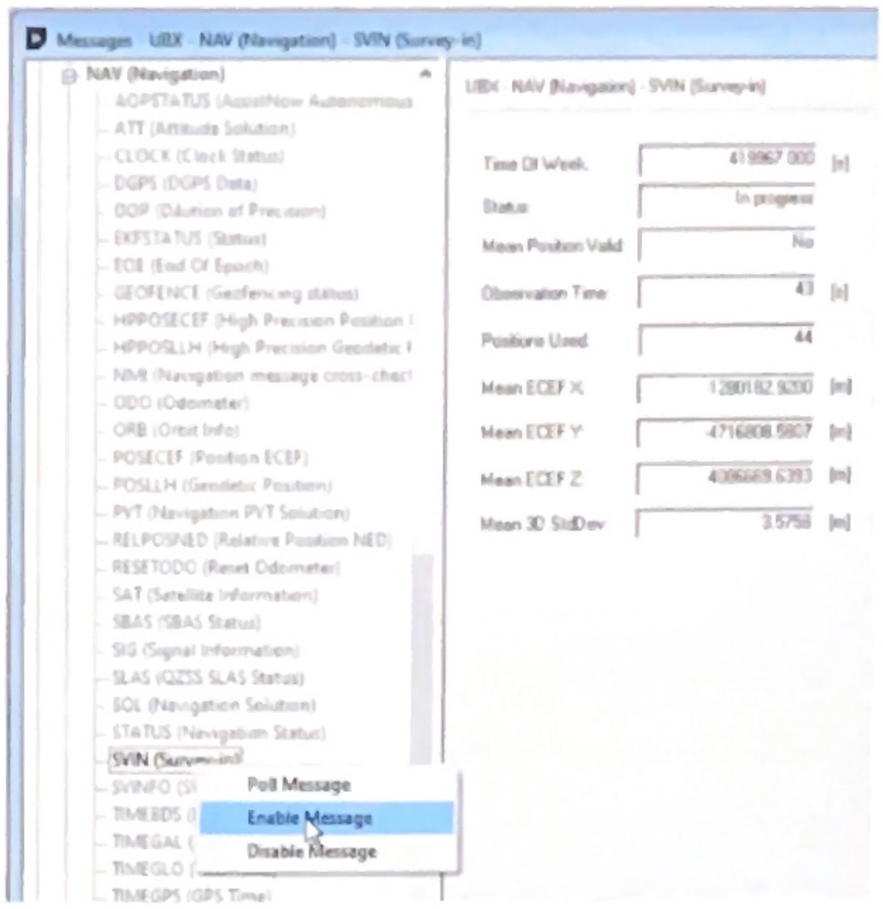


Now scroll up to the CFG or config option. This menu allows us to save the current settings. When you plug in your module again, you won't have to re-do all these settings. *BBR* is battery backed RAM. There is a small battery on the SparkFun GPS-RTK boards that should maintain the settings for over a week. The *Flash* setting records the settings to flash memory as well so if the battery loses power, flash will maintain the settings (also called non-volatile memory or NVM).

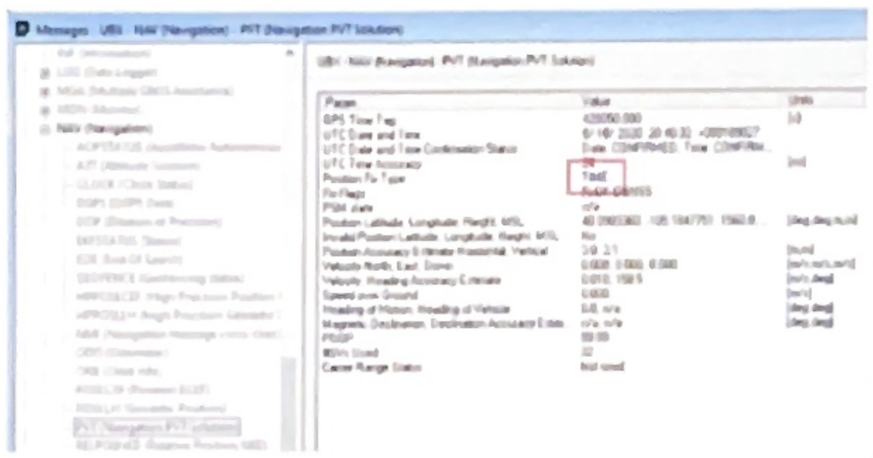


The last thing to do is to initiate a survey in. Navigate to the TMODE3 menu, set the mode to '1 - Survey-In', set the observation time to 60s, and required accuracy to 5m. These are the settings that u-blox recommends. Press 'Send' and sit back. It can take many minutes for the module to obtain enough fixes to have a standard deviation less than 5m.

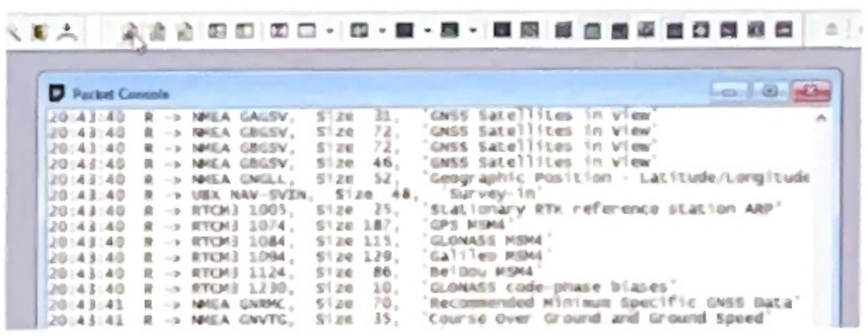
Note: You *can* enter the observation time and required accuracy values *then* save the current settings to BBR/Flash. This will effectively tell your module to survey-in at every power-on. This can be handy in many field deployments where u-center is not available but use it with caution. I have been very confused a few times when my device stopped updating its position because I had left it in survey-in mode and didn't realize it.



To monitor the status of the survey, scroll down, exiting the CFG section and enter the NAV section. Right click on 'SVIN' and click on *Enable Message*. This will tell the module to send the status of this register every second. Once the module has gotten 60s of data *and* the mean standard deviation is less than 5m then the survey will report 'Complete!'. This module's lat/long/height will no longer change because it has been fixed.



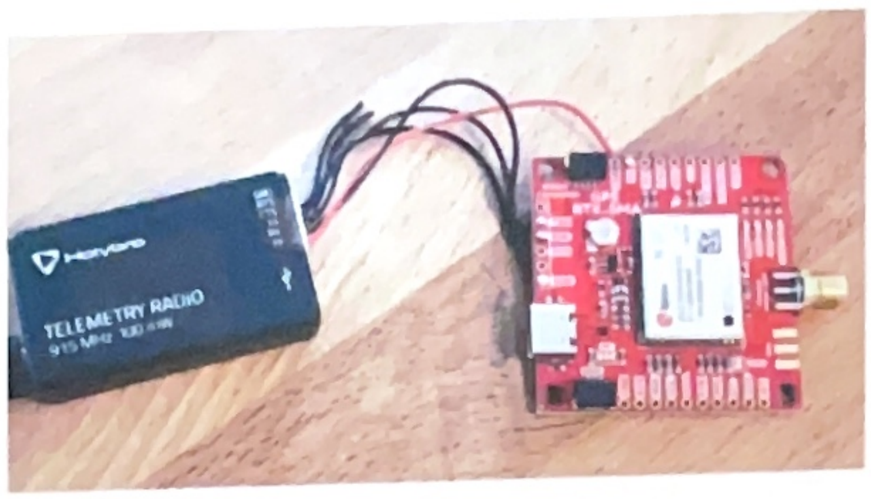
You can verify the survey in is complete by checking the PVT message. If the Fix Type is 'TIME' then you know you have successfully setup a base. Congrats!



Furthermore, we can view the RTCM messages. Open the packet console. If the RTCM messages have been enabled *and* the survey-in is complete, we should see the RTCM messages scroll by. **Note:** The RTCM messages are not visible ASCII like NMEA sentences, they are encoded bytes of data. You can't 'see' them in the Text Console but they are shown in the Packet Console.

What about fast update rates? The ZED-F9P is capable of outputting up to 30Hz fix rates. This is an astounding amount of math. "Isn't moar, better?"™ Not really for RTK. The base is transmitting correction values once per second. We could increase the output of the module to 4 or 10Hz, but because the conditions between you and the satellites are not changing that quickly, it really only bogs down your radio link.

80 | Setting up an RTK Base and Rover system | 06/28/2023 Wesley Cooke



The last thing we need to do is attach our radio. Cut the 6 pin JST-GH cable that came with the Serial Radio Telemetry kit in half. Plug the cable into one radio and note where the TX, RX, 5V and GND pins fall. **Go slow**. I wired TX/RX backwards and caused myself an hour's worth of grief. I recommend stripping one wire at a time as this makes it clear which wire I'm working on. Solder the wires as follows:

- Radio 5V - RTK 5V
- Radio TX - RTK RX2
- Radio RX - RTK TX2
- Radio GND - RTK GND

For the advanced readers of this tutorial, go ahead and wire solder the second radio cable to your rover module in *exactly* the same way.

Shown above, I wired the radio to the 6-pin UART2 header. This space is normally used for the 6-pin connection to a Bluetooth module so if you'd like to leave those pins open (for future potential Bluetooth) you can also solder the radio to the TX2/RX2 pins on the side of the GPS-RTK board. We wired the radio to 5V instead of 3.3V on the 6-pin header. We have seen the radios work at 3.3V but a high voltage should get you the maximum transmit distance.

Now is a good time to mark your module with a sharpie or a 'BASE' label so you know which module is which.

Instead of doing this, we write a LoRa program for the 1310 to forward the RTCM packets.

8.1 System Setting up an RTK Base and Rover

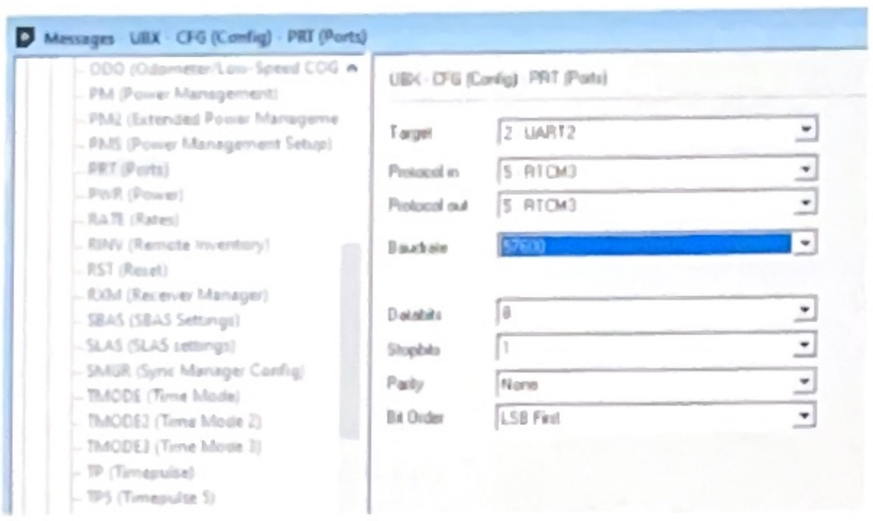
```ino
#include <Wire.h> //Needed for I2C to GNSS
#include <SparkFun_u-blox_GNSS_v3.h> //http://librarymanager/All#SparkFun_u-blox_GNSS_v3
#include <LoRa.h>;

// ZED-F9P object
SFE_UBLOX_GNSS myGNSS;

int byteIndex = 0;
uint8_t myBytes[500]; // Indexes 0 - 499

void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // Start the LoRa module
  if(!LoRa.begin(915E6))
  {
    digitalWrite(LED_BUILTIN, HIGH);
    while(1);
  }

  // Start I2C
  Wire.begin();
  Wire.setClock(400000);

  // Make sure we are connected to the ZED-F9P
  if (myGNSS.begin() == false)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    while(1);
  }
}

void loop()
{
  myGNSS.checkUblox();
  delay(250);
}
void DevUBLOXGNSS::processRTCM(uint8_t incoming)
{
  // Store the incoming bytes until we have enough
  // to make some lora packets

  myBytes[byteIndex] = incoming ; // 499 is the last
  byteIndex++;                    // 499++ is 500

  // If we have filled indexes 0-499.
  // Then we need to make some LoRa packets.

  if (byteIndex == 500)
  {
    // LoRa packets are limited to 255 bytes.

    // First packet. Indexes 0-249
    LoRa.beginPacket();
    for(int i=0; i<250; i++)
    {
      LoRa.write(myBytes[i]);
    }
    LoRa.endPacket();

    // Second Packet. Indexes 250-499
    LoRa.beginPacket();
    for(int i=250; i<500; i++)
    {
      LoRa.write(myBytes[i]);
    }
    LoRa.endPacket();

    // Start storing new bytes at the beginning of the array
    byteIndex = 0;
  }
}
```

08/28/2023 writing code

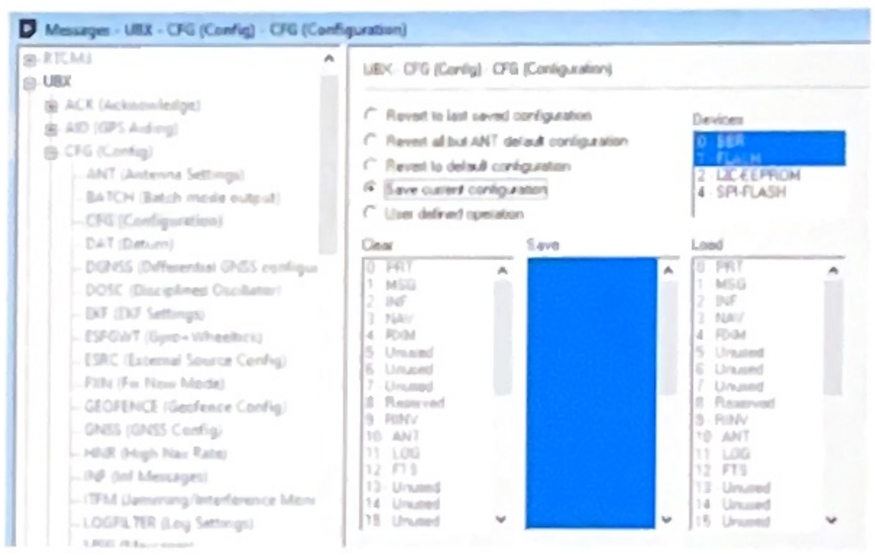
Rover Setup

We're going to assume you've got a base station setup, either as a temporary base or with ECEF coordinates (we'll go into that advanced topic later). u-blox has some good documents on how to set up the ZED-F9P in rover mode but it's surprisingly easy. The rover simply needs to be fed RTCM corrections and it will enter RTK Float, then RTK Fix.

Open u-center and connect to the rover module. Multiple instances of u-center can be started if you have multiple modules connected to one computer.

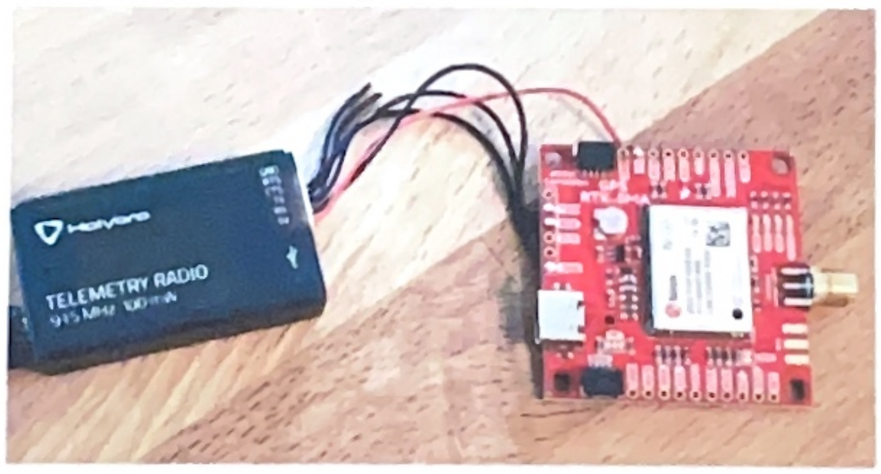


Navigate to the CFG / PRT message and configure UART2. We want to set UART2 to 57600bps to match the base and the telemetry radio kit we will be using.



Now save these port settings so they get loaded at each power-on.

Note: If you ever run into problems with the configuration of a module you can always select 'Revert to default configuration' and hit send. This should give your module a good brainwash returning it to the default factory settings. This command will also reset the BBR/Flash settings.



Solder the other half of the radio cable to the UART2 on the rover receiver. The setup is *exactly* the same as before with the base receiver:

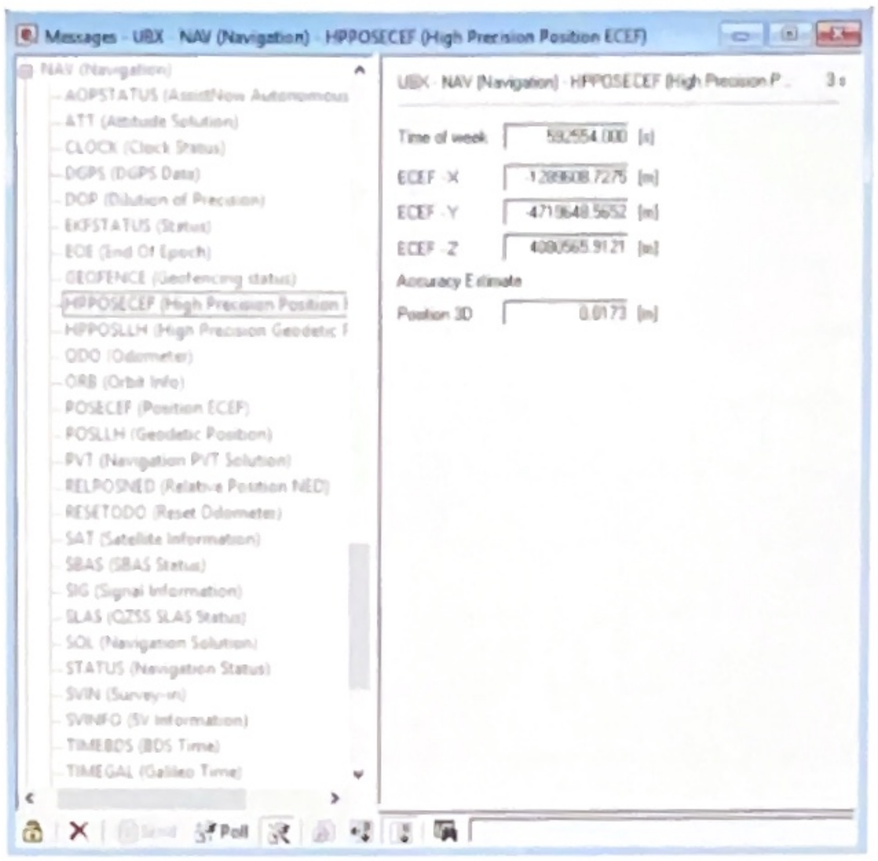
- Radio 5V - RTK 5V

* I set this to 230,400 to try to make sure LoRa would be fast enough.

- Radio TX - RTK RX2 *
- Radio RX - RTK TX2
- Radio GND - RTK GND

Time for a test! Power up the rover and base and you should see the green LED on the radios go from blinking to solid indicating the radios detect each other and are passing serial data between them. A blinking red LED on the radios indicates serial data being transmitted.

That's it! This rover module is now ready for the field. It is still recommended to have u-center handy to view the status of the rover as it receives RTCM corrections. As the rover receives RTCM it will quickly move from a 3D fix, to RTK Float, to RTK Fix.



There are a few messages to watch while the rover achieves RTK Fix. NAV/HPPOSECEF will show overall positional accuracy, NAV/HPPOSLLH will show the horizontal accuracy. NAV/PVT will show the fix type as it progresses from 3D, to RTK Float, to RTK Fixed.

PSST: We messed up. The RTK LED on the GPS-RTK boards is wired to ground instead of 3.3V. This means the RTK LED will be on in normal non-RTK mode. Once RTCM is being received and the module enters RTK Float the LED will blink. Once enough corrections are received, the RTK LED will turn off indicating RTK Fixed mode which is the best, 14mm horizontal accuracy situation.

- RTK LED On = Regular Satellite Mode
- RTK LED Blinking = RTK Float, ~<500mm accuracy
- RTK LED Off = RTK Fix, 14mm acurracy

* On our rover, we connected I2C from the Zed-F9P to I2C on the 1310. Then we connected UART2 to Serial 2.

The following code will push the RTCM messages recieved from LoRa to UART2 and track GPS locations.

Setting up an RTK Base and Rover System

06/28/2023
Wednesday
code

```ino
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <SD.h>
#include <SparkFun_u-blox_GNSS_v3.h>

// ZED-F9P Object
SFE_UBLOX_GNSS myGNSS;

long lastTime = 0;
uint8_t myBytes[249]; // One LoRa Packet
uint8_t myCount=0;
const int chipSelect = 7;

void setup()
{
  // Serial1 is connected to UART2 of the Sparkfun Board.
  Serial1.begin(230400);
  Wire.begin();

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  pinMode(chipSelect, OUTPUT);

  if (myGNSS.begin() == false)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    while(1);
  }

  myGNSS.setI2COutput(COM_TYPE_UBX);
  myGNSS.saveConfigSelective(VAL_CFG_SUBSEC_IOPORT);

  if (!LoRa.begin(915E6))
  {
    digitalWrite(LED_BUILTIN, HIGH);
    while(1);
  }

  if (!SD.begin(chipSelect))
  {
    digitalWrite(LED_BUILTIN, HIGH);
    while(1);
  }

  // If we made it this far, we are good.
  digitalWrite(LED_BUILTIN, LOW);

  File outputfile = SD.open("data.txt", FILE_WRITE);
  outputfile.println("Latitude, Longitutde, RTK, FIX, SIV");
  outputfile.close();
}
```

```
55 void loop()
56 {
57   // Try to parse the packet
58   int packetSize = LoRa.parsePacket();
59
60   // If this packet is the size we expect
61   if (packetSize == 250)
62   {
63     // Unpack it and send it out via Serial1
64     for(int i=0; i<250; i++)
65     {
66       uint8_t myByte = LoRa.read();
67       Serial1.write(myByte);
68     }
69   }
70
71   if (millis() - lastTime > 1000)
72   {
73     lastTime = millis();
74     long latitude = myGNSS.getLatitude();
75     long longitude = myGNSS.getLongitude();
76     byte fixType = myGNSS.getFixType();
77     byte RTK = myGNSS.getCarrierSolutionType();
78     byte SIV = myGNSS.getSIV();
79
80     File outputfile = SD.open("data.txt", FILE_WRITE);
81     if (outputfile)
82     {
83       outputfile.print(latitude);
84       outputfile.print(", ");
85       outputfile.print(longitude);
86       outputfile.print(", ");
87       outputfile.print(RTK);
88       outputfile.print(", ");
89       outputfile.print(fixType);
90       outputfile.print(", ");
91       outputfile.println(SIV);
92     }
93     outputfile.close();
94   }
95 }
```

Setting up an RTK Base and Rover System

04/28/2023
Wesley Cooper

Figure 3
GPS data with RTK correction data.

* This portion of the data shows amazing results. A group of students were walking toward me. I moved to the side of the sidewalk. The gps was able to track that deviation.