# GPS-RTK2 Hookup Guide

Introduction

The SparkFun GPS-RTK2 and GPS-RTK-SMA raises the bar for high-precision GPS. Utilizing the latest ZED-F9P module from u-blox the RTK2 and RTK-SMA is capable of 10mm 3 dimensional accuracy. Yes, you read that right, these boards can output your X, Y, and Z location that is roughly the width of your fingernail. With great power comes a few requirements: high precision GPS requires a clear view of the sky (sorry, no indoor location) and a stream of correction data from an RTCM source. We'll get into this more in a later section but as long as you have two ZED-F9P breakout boards, or access to an online correction source, your ZED-F9P can output lat, long, and altitude with centimeter grade accuracy.
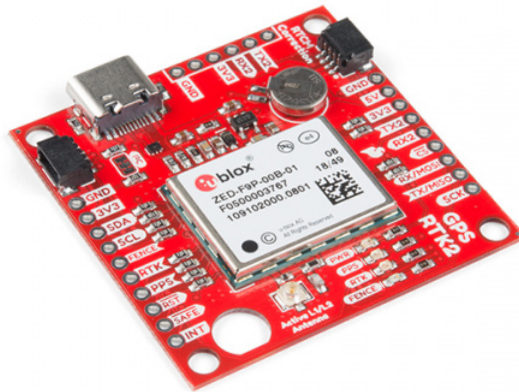
SparkFun GPS-RTK2 Board - ZED-F9P (Qwiic)
◉ GPS-15136
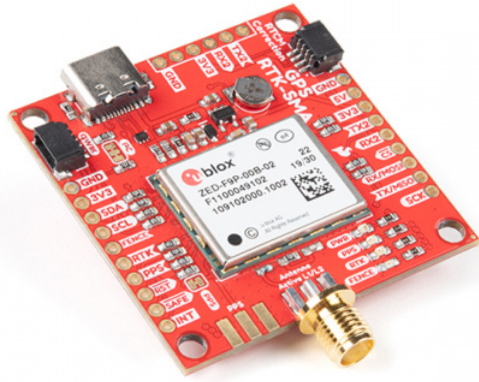
SparkFun GPS-RTK-SMA Breakout - ZED-F9P (Qwiic)
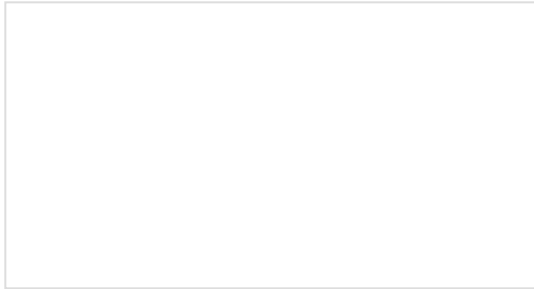
◉ GPS-16481

Product Showcase: SparkFun GPS-RTK2 Board

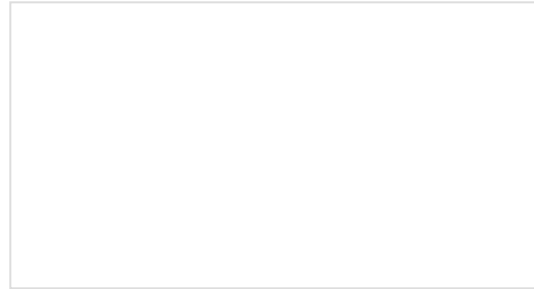Product Showcase: SparkFun Qwiic GPS-RTK-SMA Breakout

Suggested Reading

Before getting started, be sure to checkout our What is GPS RTK? tutorial and if you want to pre-read a bit have a look at our Getting Started with U-Center.
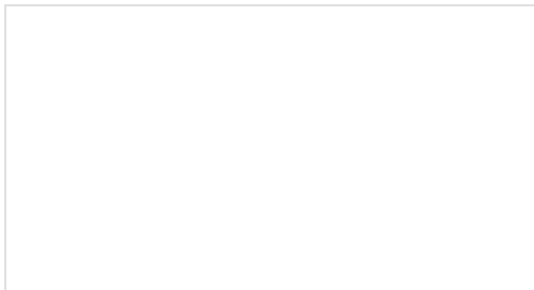

I2C
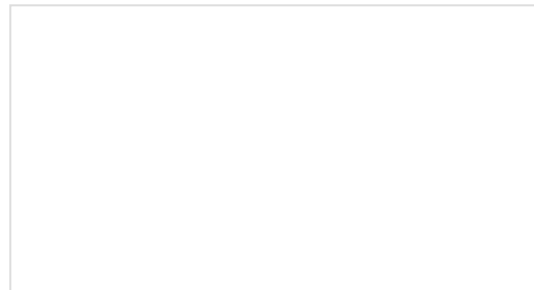An introduction to I2C, one of the main embedded communications protocols in use today.

Serial Basic Hookup Guide
Get connected quickly with this Serial to USB adapter.

What is GPS RTK?
Learn about the latest generation of GPS and GNSS receivers to get 2.5cm positional accuracy!

Getting Started with U-Center for u-blox
Learn the tips and tricks to use the u-blox software tool to configure your GPS receiver.

Hardware Overview

One of the key differentiators between the ZED-F9P and almost all other low-cost RTK solutions is the ZED-F9P is capable of receiving both L1 and L2 bands.
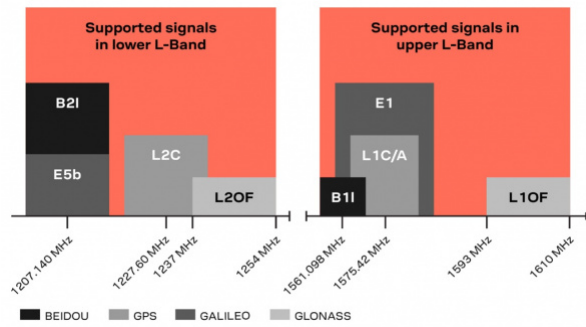
Communication Ports

The ZED-F9P is unique in that it has *five* communication ports which are all active simultaneously. You can read NMEA data over I^{2}C while you send configuration commands over the UART and vice/versa. The only limit is that the SPI pins are mapped onto the I^{2}C and UART pins so it's either SPI or I^{2}C+UART. The USB port is available at all times.

SparkFun GPS-RTK2 Board	*SparkFun GPS-RTK-SMA Board*
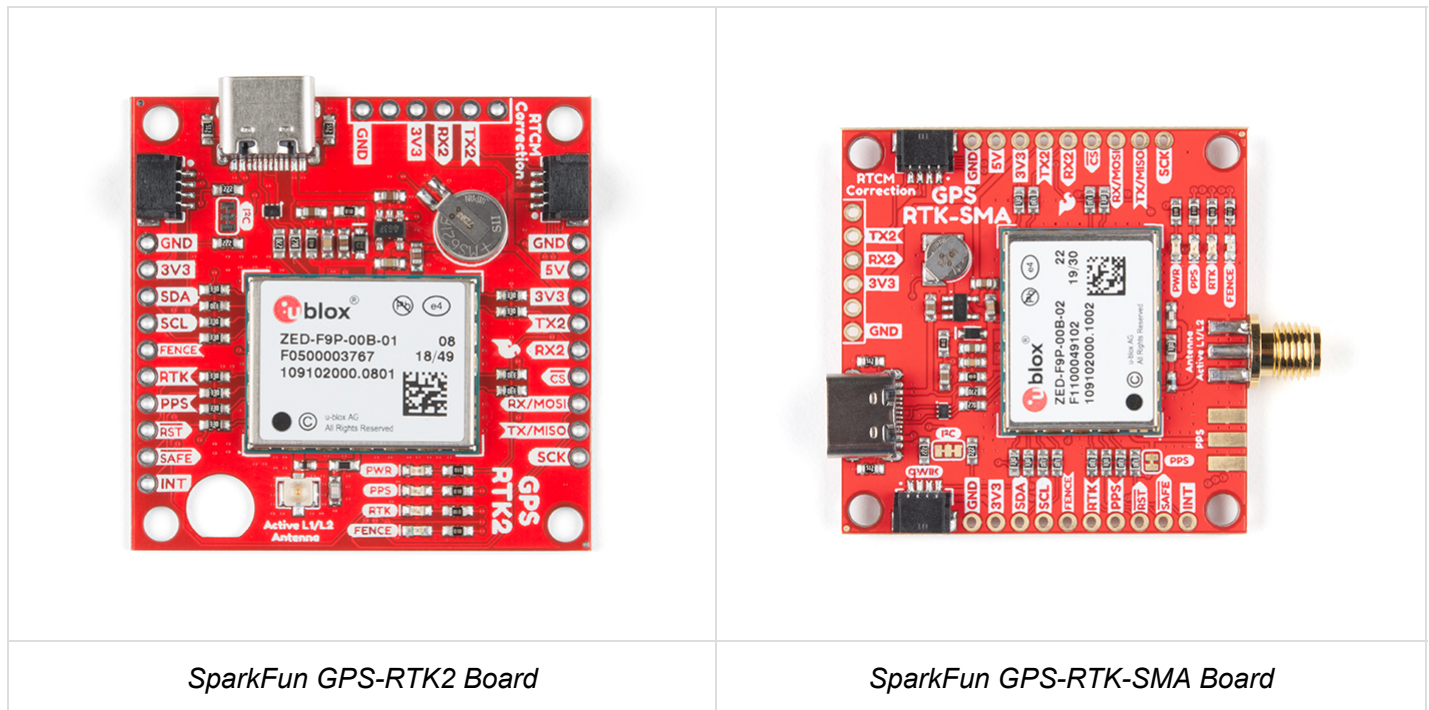
USB

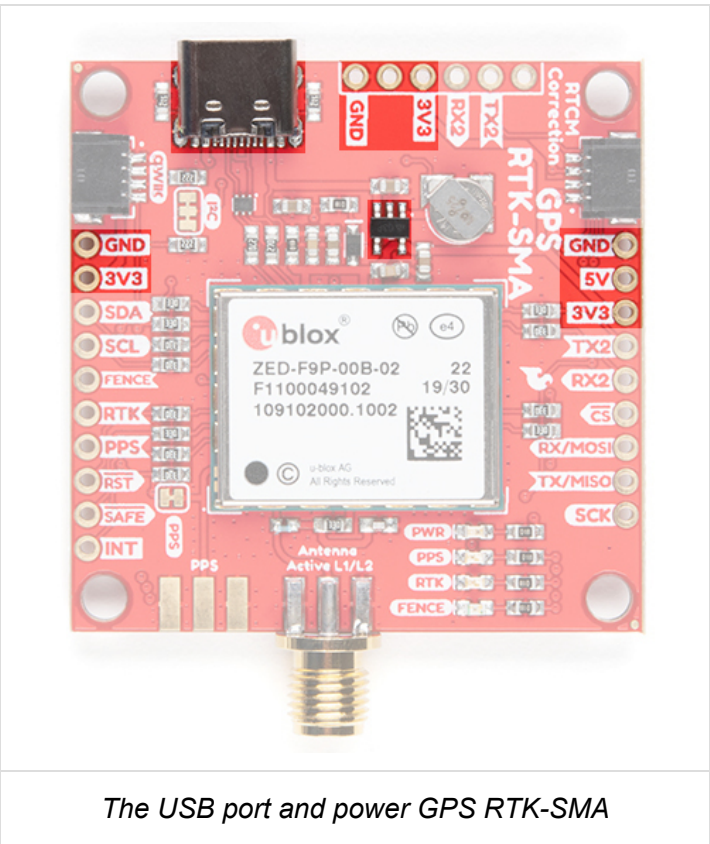The USB C connector makes it easy to connect the ZED-F9P to u-center for configuration and quick viewing of NMEA sentences. It is also possible to connect a Raspberry Pi or other single board computer over USB. The ZED-F9P enumerates as a serial COM port and it is a separate serial port from the UART interface. See Getting Started with U-Center for more information about getting the USB port to be a serial COM port.
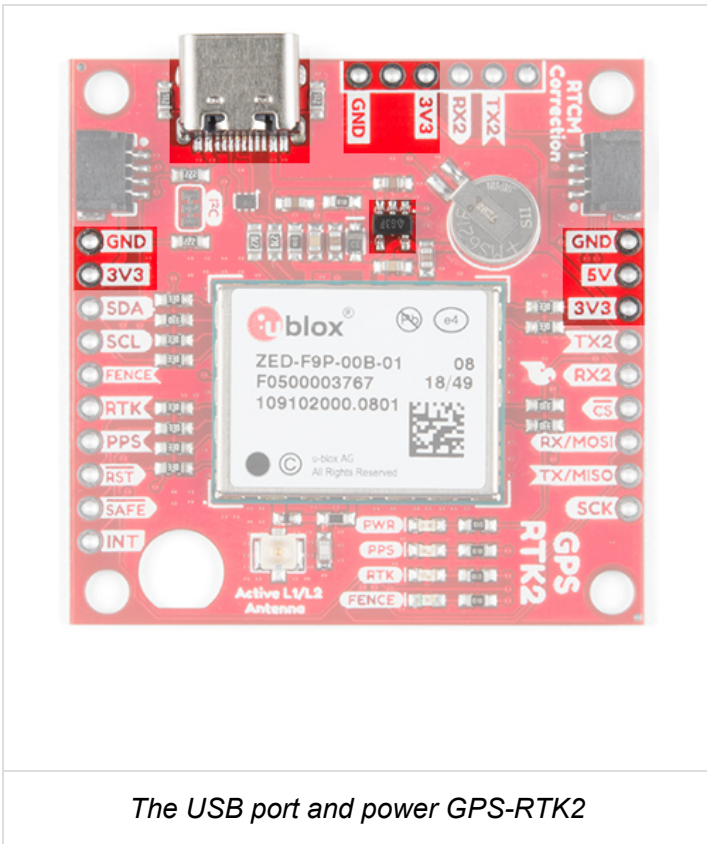
| The USB port and power GPS-RTK2 | The USB port and power GPS RTK-SMA |

A 3.3V regulator is provided to regulate the 5V USB down to **3.3V** the module requires. External 5V can be applied or a direct feed of 3.3V can be provided. Note that if you're provide the board with 3.3V directly it should be a clean supply with minimal noise (less than 50mV VPP ripple is ideal for precision locating).

The 3.3V regulator is capable of sourcing 600mA from a 5V input and the USB C connection is capable of sourcing 2A.

I^2C (a.k.a DDC)

The u-blox ZED-F9P has a "DDC" port which is really just an I^2C port (without all the fuss of trademark issues). These pins are shared with the SPI pins. By default, the I^2C pins are enabled. Be sure the DSEL jumper on the rear of the board is open. The GPS-RTK2 and GPS-RTK-SMA from SparkFun also includes two Qwiic connectors to make daisy chaining this GPS receiver with a large variety of I^2C devices. Checkout Qwiic for your next project.
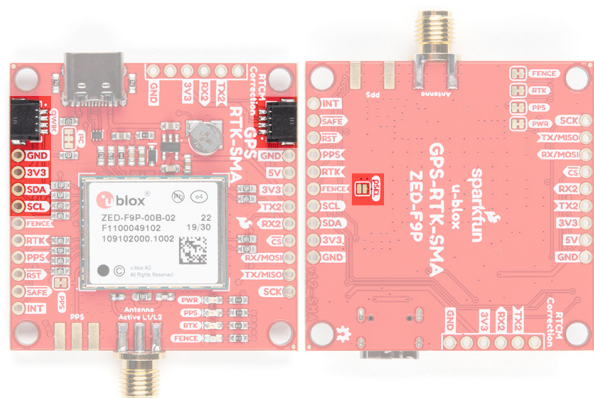
All features are accessible over the I^{2}C ports including reading NMEA sentences, sending UBX configuration strings, piping RTCM data into the module, etc. We've written an extensive Arduino library showing how to configure most aspects of the ZED-F9P making I^{2}C our preferred communication method on the ZED. You can get the library through the Arduino library manager by searching '**SparkFun Ublox**'. Checkout the SparkFun U-blox Library section for more information.

UART/Serial

The classic serial pins are available on the ZED-F9P but are shared with the SPI pins. By default, the UART pins are enabled. Be sure the DSEL jumper on the rear of the board is open.

- TX/MISO = TX out from ZED-F9P
- RX/MOSI = RX into ZED-F9P
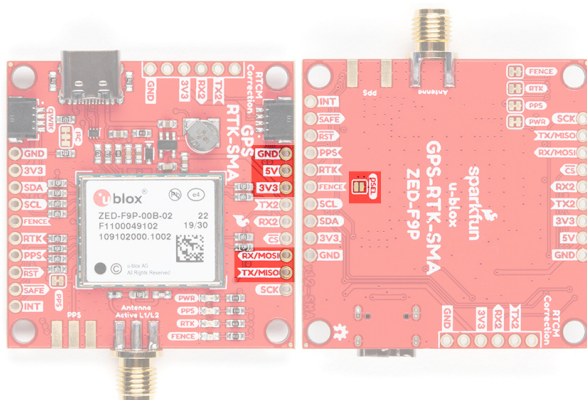
Serial pins on SparkFun ZED-F9P breakout boards highlighted

There is a second serial port (UART2) available on the ZED-F9P that is primarily used for RTCM3 correction data. By default, this port will automatically receive and parse incoming RTCM3 strings enabling RTK mode on the board. In addition to the TX2/RX2 pins we have added an additional '**RTCM Correction**' port where we arranged the pins to match the industry standard serial connection (aka the 'FTDI' pinout). This pinout is compatible with our Bluetooth Mate and Serial Basic so you can send RTCM correction data from a cell phone or computer. Note that RTCM3 data can also be sent over I^{2}C, UART1, SPI, or USB if desired.
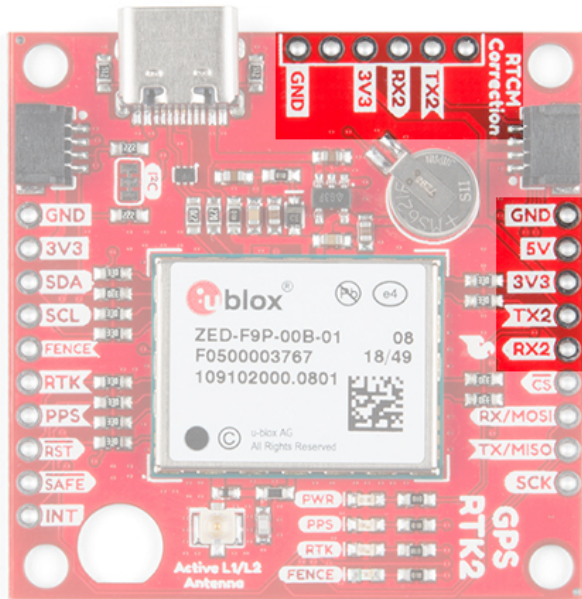
UART2 and RTCM Correction Port on the GPS-RTK2	*UART2 and RTCM Correction Port on the GPS-RTK-SMA*
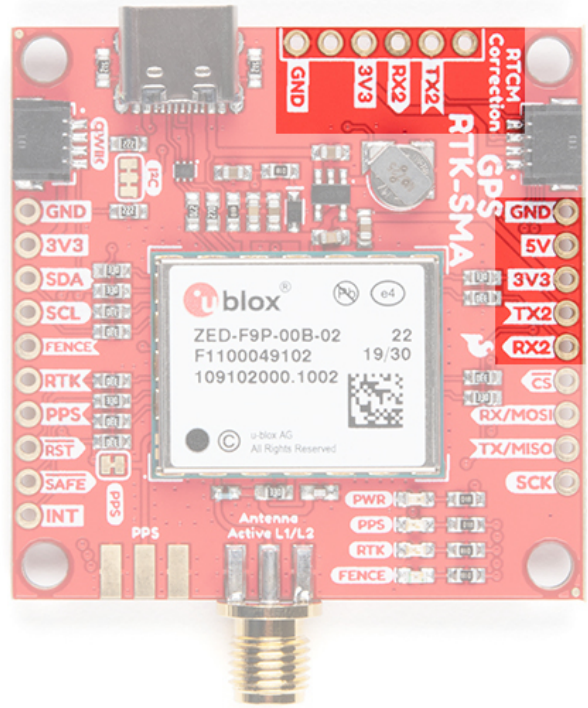
The RTCM correction port (UART2) defaults to **38400bps** serial but can be configured via software commands (checkout our Arduino library) or over USB using u-center. Keep in mind our Bluetooth Mate defaults to **115200bps**. If you plan to use Bluetooth for correction data (we found it to be easiest), we recommend you increase this port speed to 115200bps using u-center. Additionally, but less often needed, the UART2 can be configured for NMEA output. In general, we don't use UART2 for anything but RTCM correction data, so we recommend leaving the in/out protocols as RTCM.
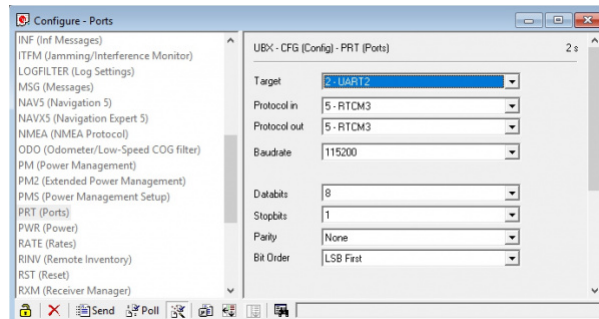
If you've got the ZED-F9P setup for base station mode (also called *survey-in* mode) the UART2 will *output* RTCM3 correction data. This means you can connect a radio or wired link to UART2 and the board will automatically send *just* RTCM bytes over the link (no NMEA data taking up bandwidth). Here is an example using the RTK2 but you can also use the RTK-SMA using the same setup.
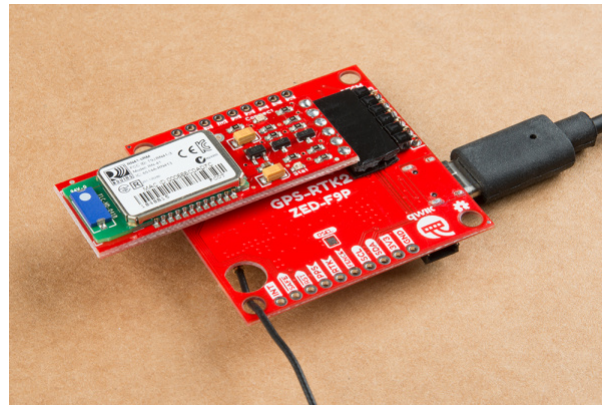
Base station setup to send RTCM bytes out over Bluetooth

SPI

The ZED-F9P can also be configured for SPI communication. By default, the SPI port is disabled. To enable SPI close the DSEL jumper on the rear of the board. Closing this jumper will disable the UART1 and I^{2}C interfaces (UART2 will continue to operate as normal).
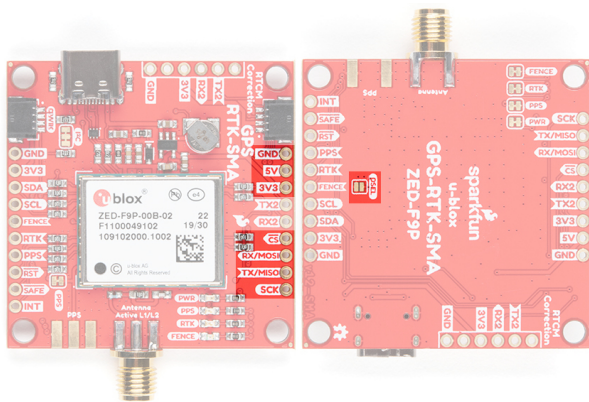
The SPI pins highlighted on the SparkFun RTK2 and RTK-SMA

Control Pins

The control pins are highlighted below.

| *Highlighted control pins of the SparkFun GPS-RTK2* | Highlighted control pins of the SparkFun GPS-RTK-SMA |

These pins are used for various extra control of the ZED-F9P:

- **FENCE:** Geofence output pin. Configured with U-Center. Will go high or low when a geofence is setup. Useful for triggering alarms and actions when the module exits a programmed perimeter.
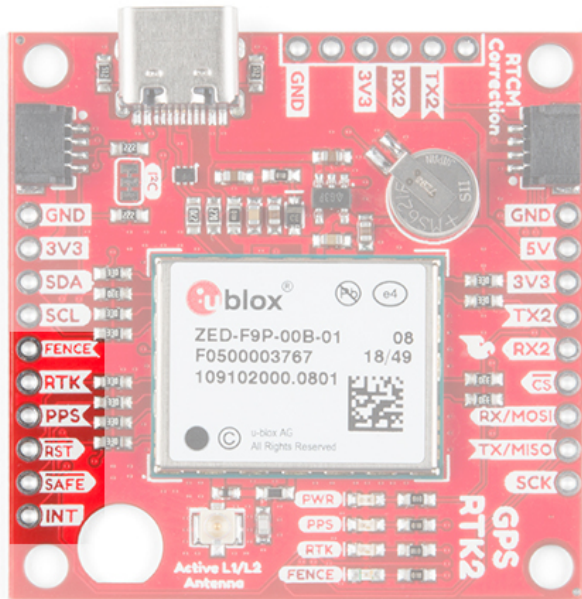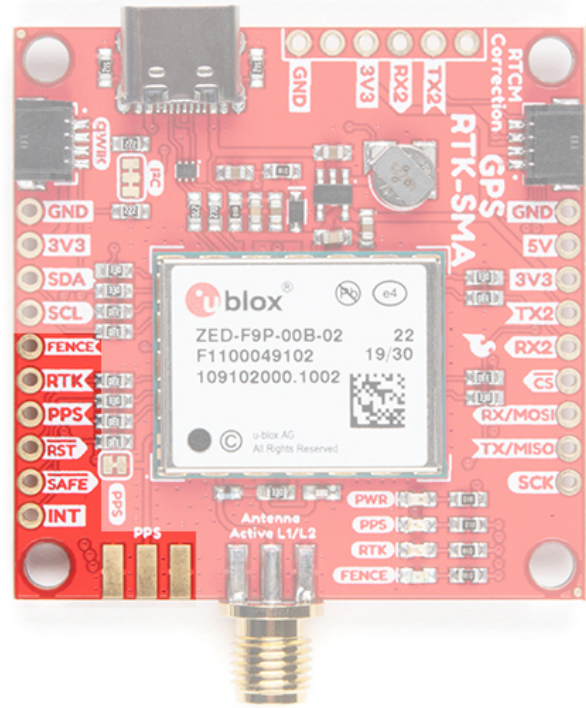- **RTK:** Real Time Kinematic output pin. Remains high when module is in normal GPS mode. Begins blinking when RTCM corrections are received and module enters RTK float mode. Goes low when module enters RTK fixed mode and begins outputting cm-level accurate locations.
- **PPS:** Pulse-per-second output pin. Begins blinking at 1Hz when module gets basic GPS/GNSS position lock.
- **RST:** Reset input pin. Pull this line low to reset the module.
- **SAFE:** Safeboot input pin. This is required for firmware updates to the module and generally should not be used or connected.
- **INT:** Interrupt input/output pin. Can be configured using U-Center to bring the module out of deep sleep or to output an interrupt for various module states.

Note: For those that need to connect a SMA connector to the PPS pin, the RTK-SMA includes a footprint so that you can manually solder the connector to the board for your application. The PPS output is helpful as a clock source correction when synchronizing equipment (not 1 Hz but in many MHz). The PPS output can be configured to output a very accurate clock which scientists use to correct less accurate, but much faster clocks. To configure, you can use the u-center to adjust The NEO-F9P's setting under **View** > **Conviguration View** > **TP (TimePulse)**.

SMA Connector

⊙ WRL-00593

Antenna

The ZED-F9P requires a good quality GPS or GNSS (preferred) antenna. A U.FL connector is provided. Note: U.FL connectors are rated for only a few mating cycles (about 30) so we recommend you set it and forget it. You may need to secure the u.FL to SMA cable depending on your application. Otherwise, you could use the RTK-SMA version.
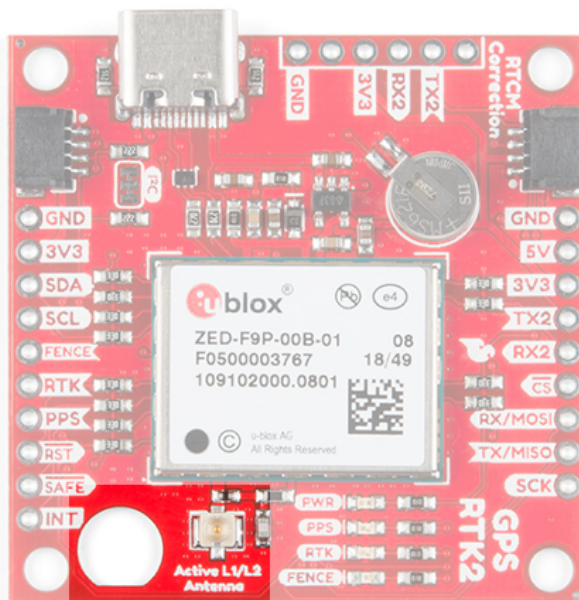
| U.FL antenna connector and SMA cut-out on the GPS RTK2 | SMA antenna connector on the GPS-SMA |
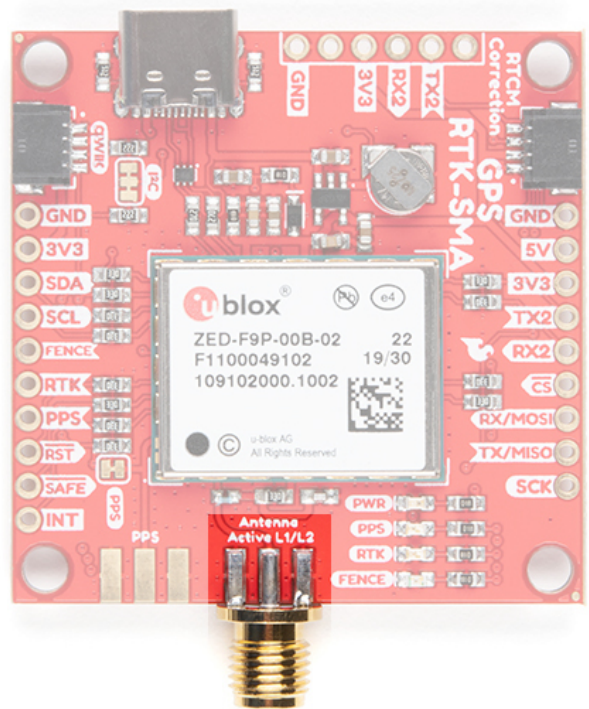
A cutout for the SMA bulkhead is available for those who want an extra sturdy connection. We recommended installing the SMA into the board only when the board is mounted in an enclosure. Otherwise, the cable runs the risk of being damaged when compressed (for example, students carrying the board loose in a backpack). ac
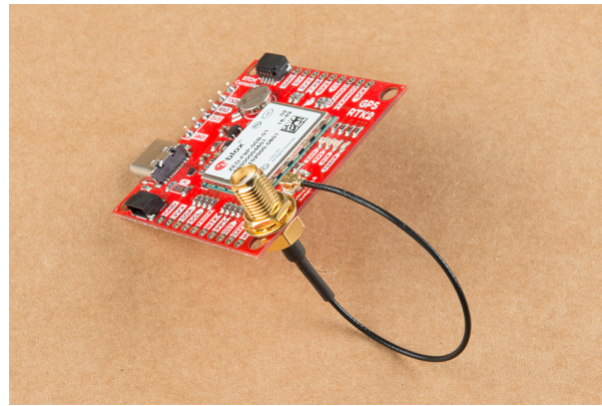
A U.FL to SMA cable threaded through the mounting hole provides a robust connection that is also easy to disconnect at the SMA connection if needed. Check below in **Connecting an Antenna** for more information.

Low-cost magnetic GPS/GNSS antennas can be used (checkout the ublox white paper) but a 4" / 10cm metal disc is required to be placed under the antenna as a metal ground plane.

LEDs

The board includes four status LEDs as indicated in the image below.

Status LEDs on the GPS-RTK2	Status LEDs on the GPS-SMA
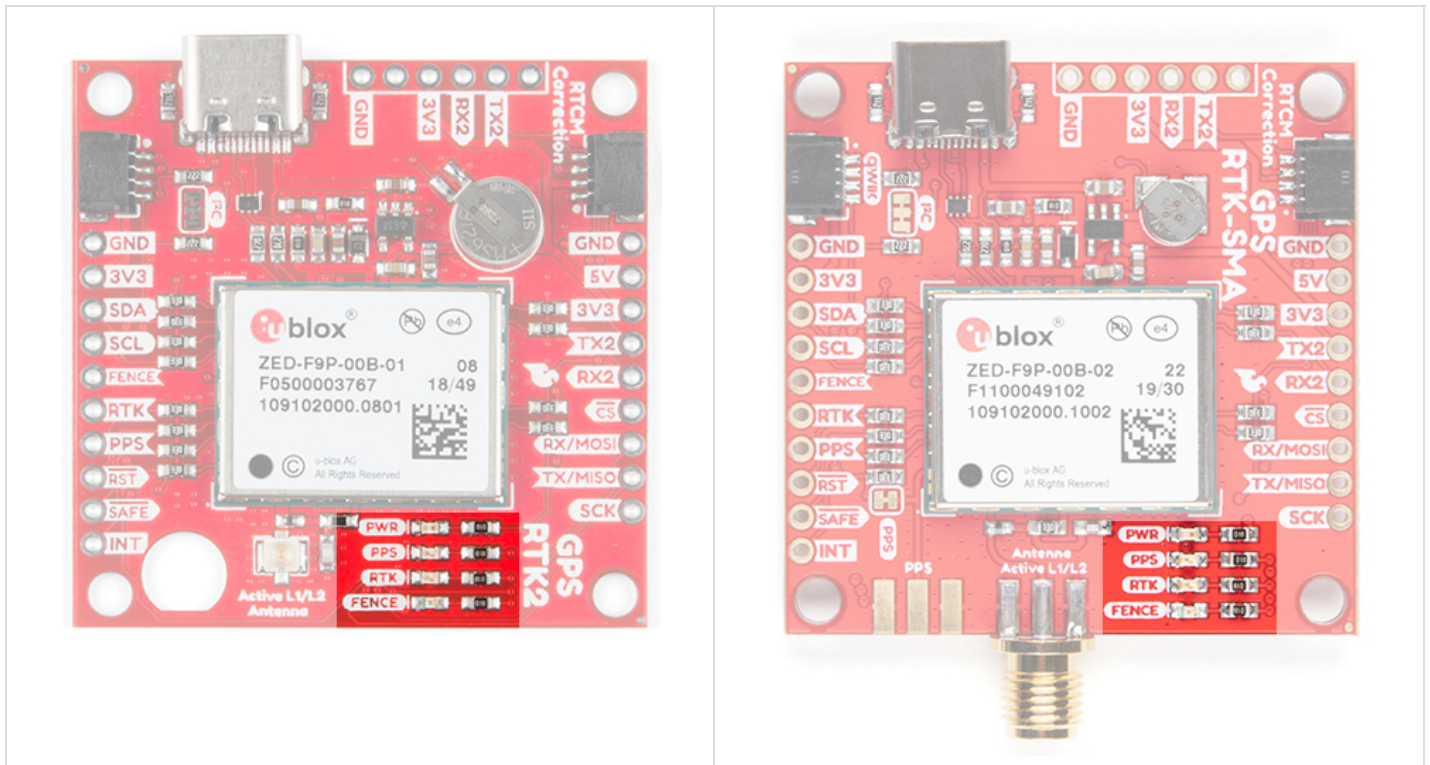
- **PWR:** The power LED will illuminate when 3.3V is activated either over USB or via the Qwiic bus.
- **PPS:** The pulse per second LED will illuminate each second once a position lock has been achieved.
- **RTK:** The RTK LED will be illuminated constantly upon power up. Once RTCM data has been successfully received it will begin to blink. This is a good way to see if the ZED-F9P is getting RTCM from various sources. Once an RTK fix is obtained, the LED will turn off.
- **FENCE:** The FENCE LED can be configured to turn on/off for geofencing applications.

Jumpers

There are five jumpers used to configure the GPS-RTK2.

User jumpers on the SparkFun RTK2 and RTK-SMA

Closing **DSEL** with solder enables the SPI interface and disables the UART and I^{2}C interfaces. USB will still function.

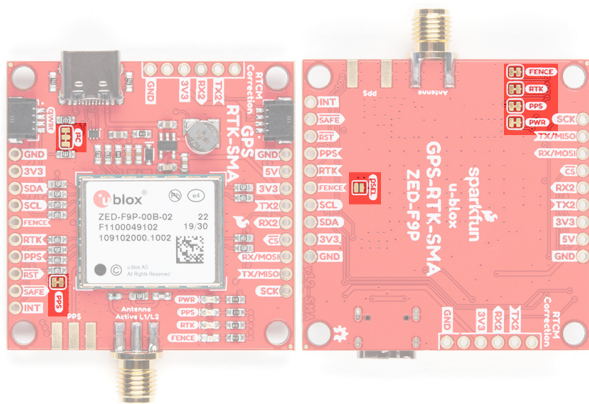Cutting the **I^{2}C** jumper will remove the 2.2k Ohm resistors from the I^{2}C bus. If you have many devices on your I^{2}C bus you may want to remove these jumpers. Not sure how to cut a jumper? Read here!

Cutting the **JP1**, **JP2**, **JP3** jumpers on the RTK2 will disconnect of the various status LEDs from their associated pins. These have been labeled on the RTK-SMA version. We have included a jumper for the **PPS** on the top and bottom of the RTK-SMA.

Backup Battery

The MS621FE rechargeable battery maintains the battery backed RAM (BBR) on the GNSS module. This allows for much faster position locks (a.k.a. hot start). The BBR is also used for module configuration retention. The battery is automatically trickle charged when power is applied and should maintain settings and GNSS orbit data for up to two weeks without power.
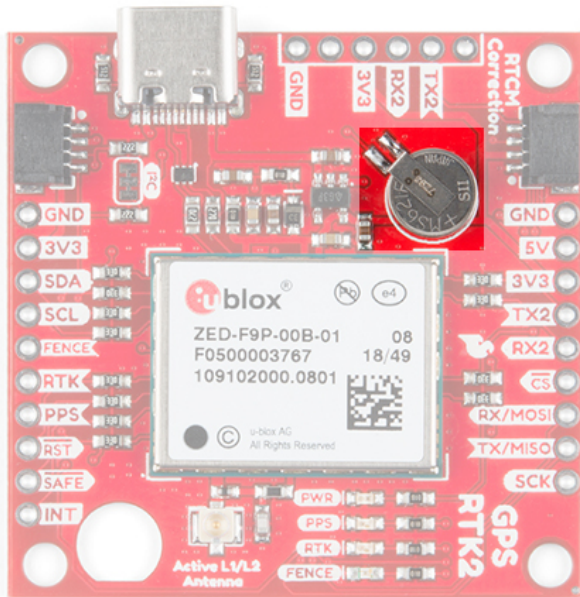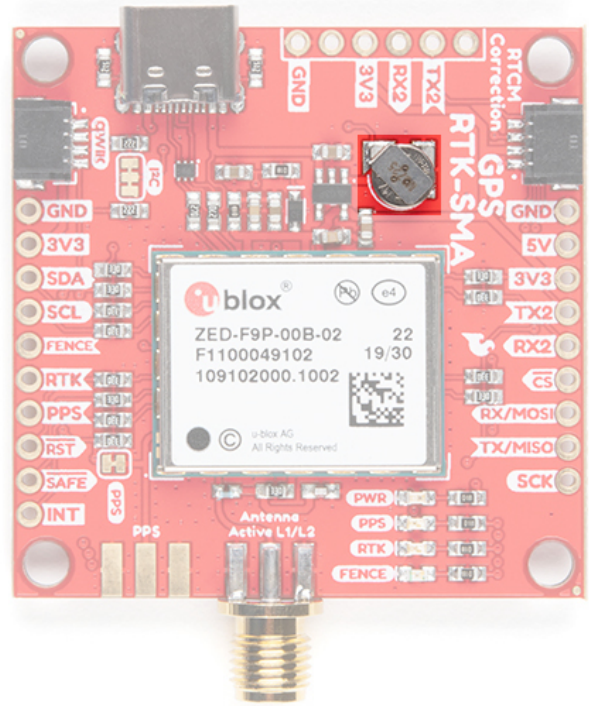
| The backup battery on the SparkFun RTK2 | The backup battery on the SparkFun RTK-SMA |

Connecting an Antenna

U.FL connectors are very good but they are a designed to be implemented inside a small embedded application like a laptop. Exposing a U.FL connector to the wild risks it getting damaged. To prevent damaging the U.FL connection on the GPS-RTK2 we recommend threading the U.FL cable through the stand-off hole, then attach the U.FL connectors. This will provide a great stress relief for the antenna connection. Now attach your SMA antenna of choice.
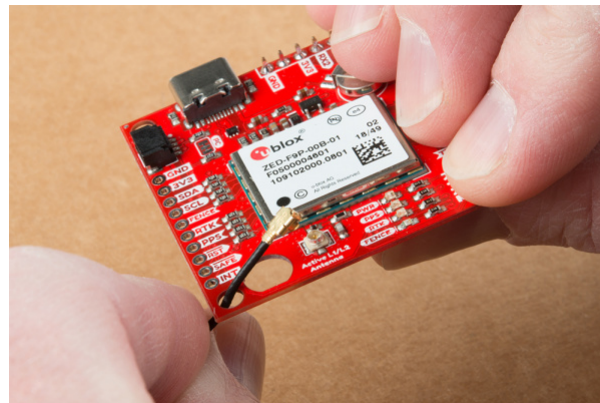
⚠ **Be Careful!** U.FL connectors are easily damaged. Make sure the connectors are aligned, flush face to face (not at an angle), then press down using a rigid blunt edge such as the edge of a PCB or point of a small flat head screwdriver. For more information checkout our tutorial Three Quick Tips About Using U.FL.
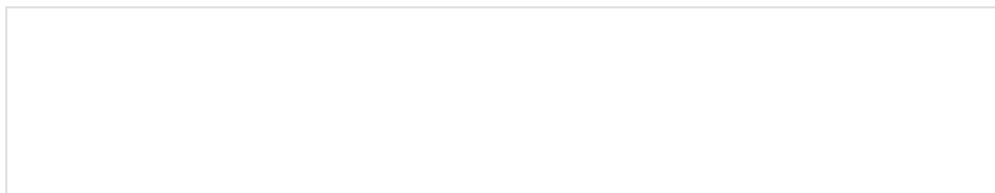
Three Quick Tips About Using U.FL
Quick tips regarding how to connect, protect, and disconnect U.FL connectors.

Additionally, a bulkhead cut-out is provided to screw the SMA onto the PCB if desired.
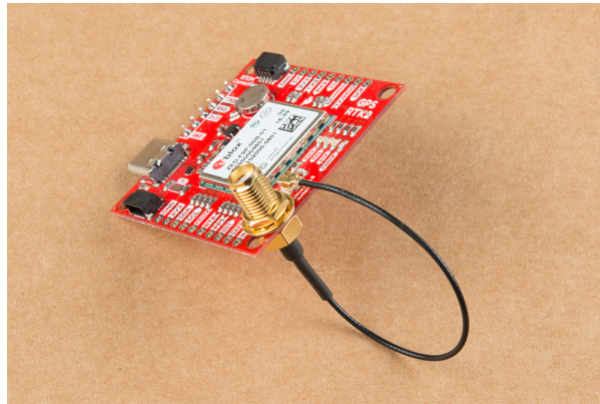
While this method decreases stress from the U.FL connector it is only recommended when the board has been permanently mounted. If the board is not mounted, the cable on the U.FL cable is susceptible to being kinked causing impedance changes that may decrease reception quality.

If you are using the the RTK-SMA, there is no need to worry about this since there is a SMA connector soldered on the board providing a sturdy connection.
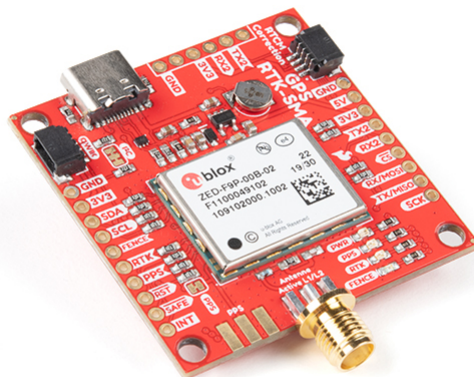
If you're indoors you *must* run a SMA extension cable long enough to locate the antenna where it has a clear view of the sky. That means no trees, buildings, walls, vehicles, or concrete metally things between the antenna and the sky. Be sure to mount the antenna on a 4"/10cm metal ground plate to increase reception.

Board Dimensions

The overall board dimensions for each board is ~1.70" x ~1.70" (~43.18mm x ~43.18mm). Keep in mind the connectors are not flush with the board, which adds additional length to the board. Make sure to take that into account when mounting and placing the board in an enclosure.
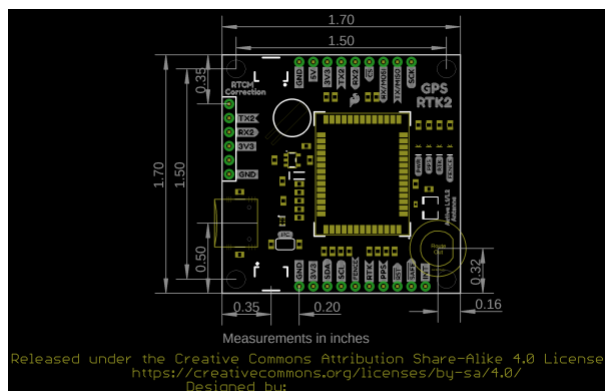
Board Dimensions for RTK2	*Board Dimensions for RTK-SMA*
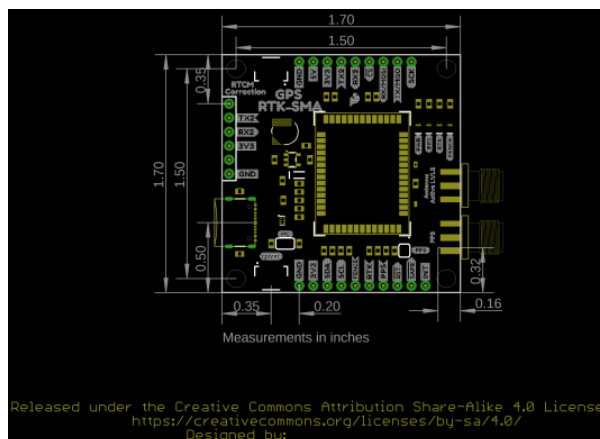
Connecting the ZED-F9P to a Correction Source

Before you go out into the field it's good to understand how to get RTCM data and how to pipe it to the GPS-RTK2 and GPS-RTK-SMA. We recommend you read Connecting a Correction Source section of the original GPS-RTK tutorial. This will give you the basics of how to get a UNAVCO account and how to identify a Mount Point within 10km of where your ZED-F9P rover will be used. This section builds upon these concepts.
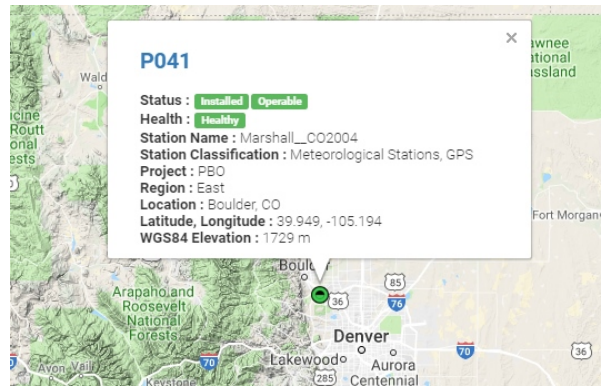
GPS RTK Hookup Guide: Connecting a Correction Source

For this example, we will show how to get correction data from the UNAVCO network and pull that data in using the Android app called NTRIP Client. The correction data will then be transmitted from the app over Bluetooth to the ZED-F9P using the SparkFun Bluetooth Mate.

Required Materials

- 1x GPS-RTK2 or GPS-RTK-SMA
- 1x GPS or GNSS Antenna
- 1x Metal Plate of 4" or larger
- 1x SMA extension cable (if needed to get a clear view of the sky)
- 1x USB C 2.0 Cable
- 1x Bluetooth Mate
- 1x male and female header pair

GNSS antenna sitting on a metal ground plate elevated with clear view of the sky

Now setup your GPS receiver such that you can work from your desk but have the antenna outdoors with a clear view of the sky.

Required Software

- Credentials with a free RTCM provider such as UNAVCO
- U-Center
- Get the NTRIP By Lefebure app from Google Play. There seem to be NTRIP apps for iOS but we have not been able to verify any one app in particular. If you have a favorite, please let us know.

First, we need to attach the Bluetooth Module to the ZED-F9P's breakout board. Solder a female header to the Bluetooth Mate so that it hangs off the end.

On the GPS-RTK2 or GPS_RTK-SMA board we recommend soldering the right-angle male header underneath the board. This will allow the Bluetooth module to be succinctly tucked under the board.

When attaching the Bluetooth Mate to board be sure to align the pins so that the **GND** indicator align on both boards. Once Bluetooth has been installed attach your GNSS antenna and connect the board over USB. This will power the board and the Bluetooth Mate.
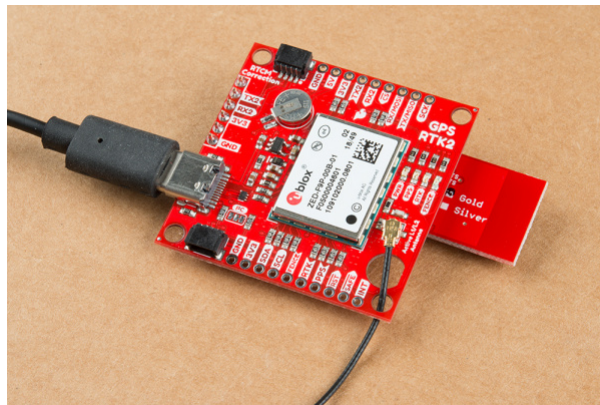
Where the male and female headers go is personal preference. For example, here are two Bluetooth Mates; one with male headers, one with female.

Soldering a female header to a Bluetooth Mate makes it easier to add Bluetooth to boards that have a 'FTDI' style connection like our OpenScale, Arduino Pro, or Simultaneous RFID Reader. Whereas, soldering a male header to the Bluetooth Mate makes it much easier to use in a breadboard. It's really up to you!

The Bluetooth Mate defaults to **115200bps** whereas the ZED-F9P is expecting serial over UART2 to be **38400bps**. To fix this we need to open u-center and change the port settings for UART2. If you haven't already, be sure to checkout the tutorial Getting Started with U-Center to get your bearings.

Open the Configure window and navigate to the **PRT (Ports)** section. Drop down the target to UART2 and set the baud rate to **115200**. Finally, click on the 'Send' button.

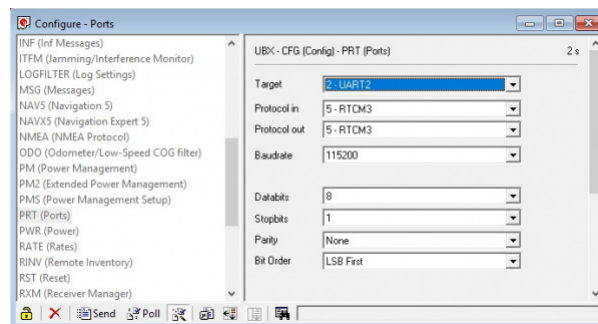By this time you should have a valid 3D GPS lock with ~1.5m accuracy. It's about to get a *lot* better.

We are going to assume you've read the original RTK tutorial and obtained your UNAVCO credentials including the following:

- Username
- Password
- IP Address for UNAVCO (69.44.86.36 at time of writing)
- Caster Port (2101 at time of writing)
- Data Stream a.k.a. Mount Point ('P041_RTCM3' if you want the one near Boulder, CO - but you should really find one nearest your rover location)

The Bluetooth Mate should be powered up. From your phone, discover the Bluetooth Mate and pair with it. The module used in this tutorial was discovered as **RNBT-E0DC** where *E0DC* is the last four characters of the MAC address of the module and should be unique to your module.

Once you have your UNAVCO credentials and you've paired with the Bluetooth module open the NTRIP client.
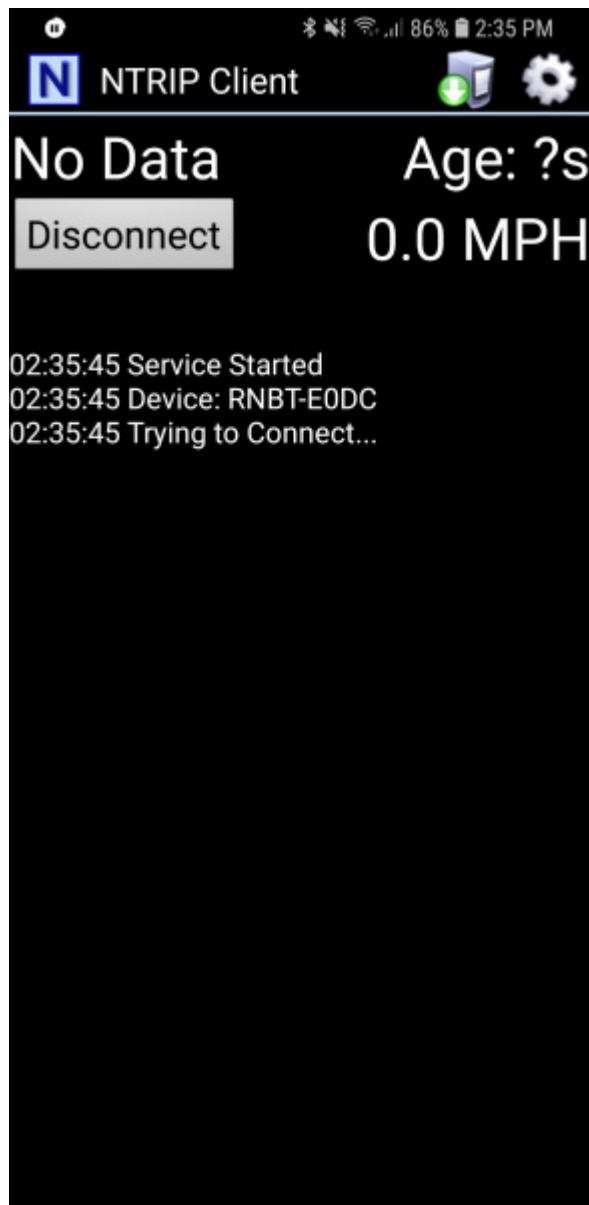
From the home screen, click on the gear in upper right corner then *Receiver Settings*.
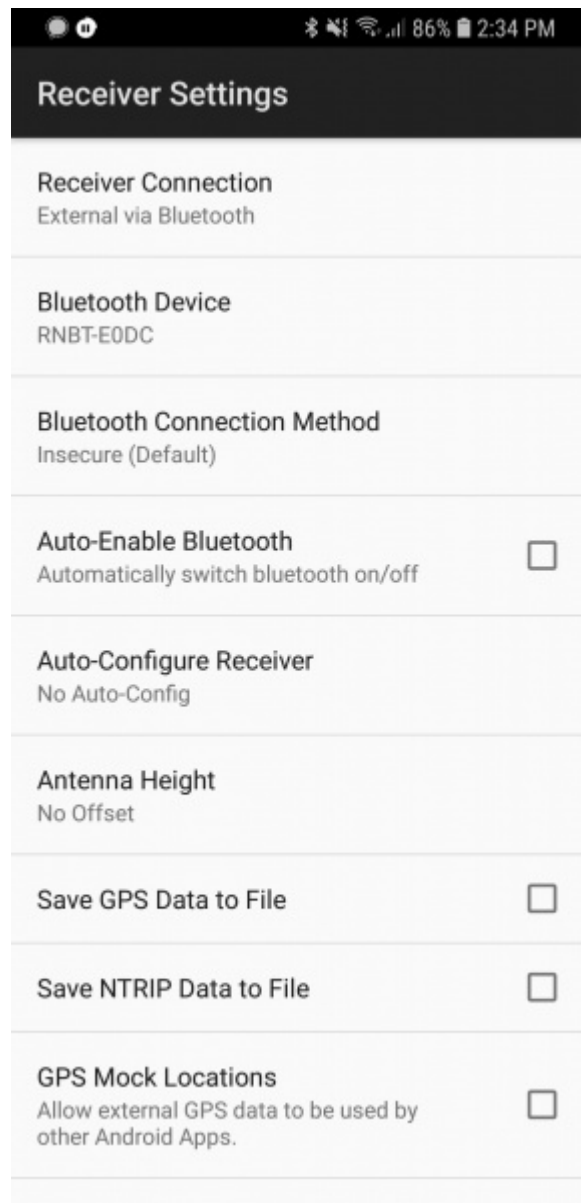
Verify that the *Receiver Connection* is set to Bluetooth then select *Bluetooth Device* and select the Bluetooth module you just paired with. Next, open NTRIP settings and enter your credentials including mounting point (a.k.a. *Data Stream*).
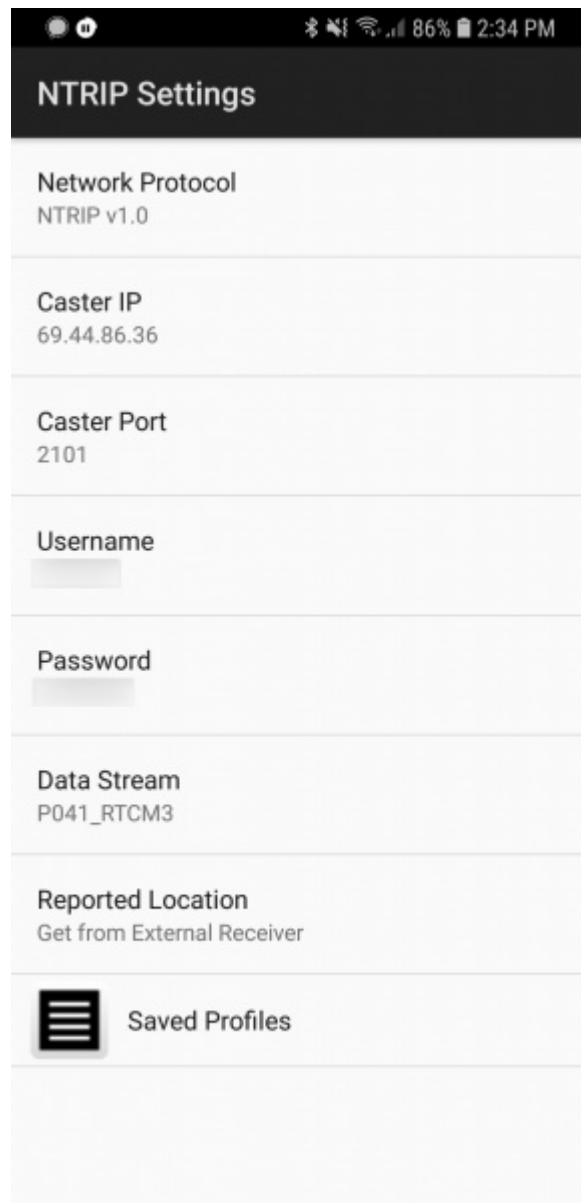
This example demonstrates how to obtain correction data from UNAVCO's servers but you could similarly setup your own base station using another ZED-F9P and RTKLIB to broadcast the correction data. This NTRIP app would connect to your RTKLIB based server giving you some amazing flexibility (the base station could be anywhere there's a laptop and Wifi within 10km of your rover).

Ok. You ready? This is the fun part. Return to the main NTRIP window and click **Connect**. The app will connect to the Bluetooth module. Once connected, it will then connect to your NTRIP source. Once data is flowing you will see the number of bytes increase every second.
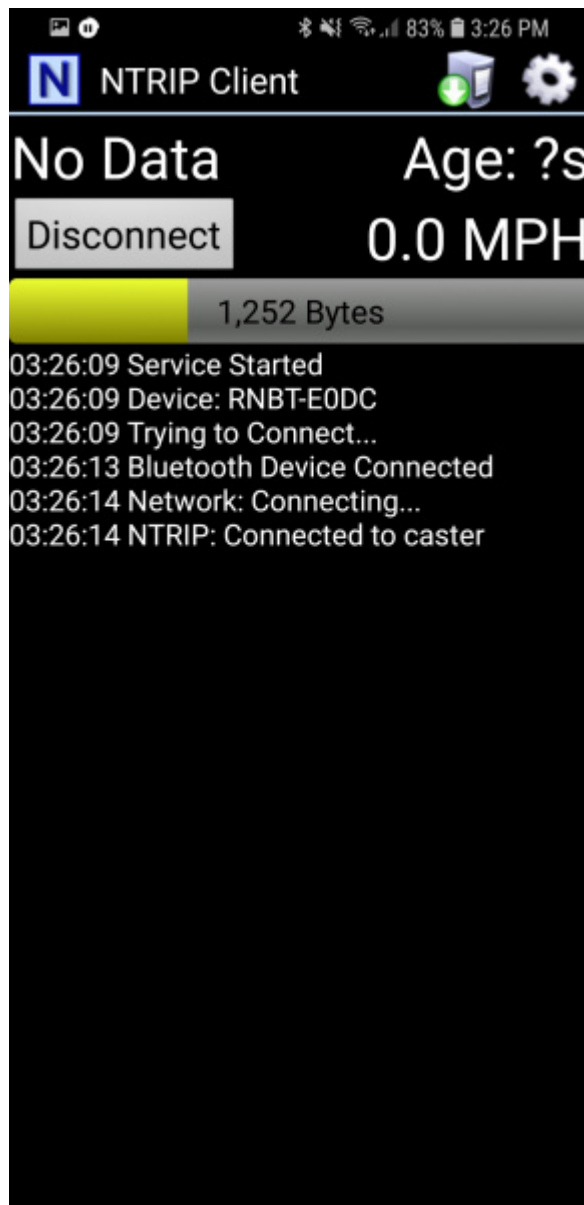
Within a few seconds you should see the **RTK** LED on the ZED-F9P's breakout board turn off. This indicates you have an RTK fix. To verify this, open u-center on your computer. The first thing to notice is that **Fix Mode** in the left hand black window has changed from *3D* to *3D/DGNSS/FIXED*.
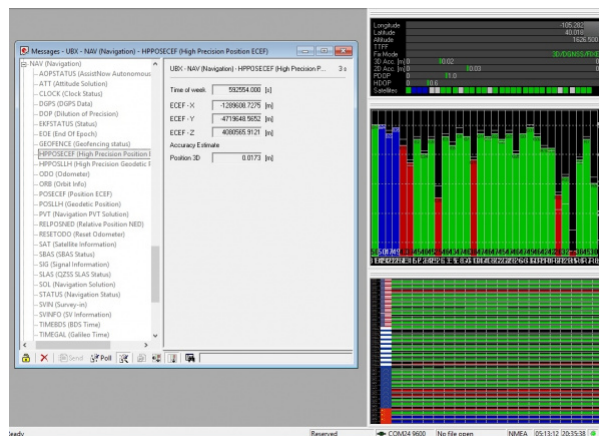
Navigate to the UBX-NAV-HPPOSECEF message. This will show you a high-precision 3D accuracy estimate. We were able to achieve 17mm accuracy using a low-cost GNSS antenna with a metal plate ground plane *and* we were over 10km from the correction station.

Congrats! You now know where you are within the diameter of a dime!

SparkFun U-blox Library

> **Note:** This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

The SparkFun Ublox Arduino library enables the reading of all positional datums as well as sending binary UBX configuration commands over I^{2}C. This is helpful for configuring advanced modules like the ZED-F9P but also the NEO-M8P-2, SAM-M8Q and any other Ublox module that use the Ublox binary protocol.

The SparkFun U-blox Arduino library can be downloaded with the Arduino library manager by searching **'SparkFun Ublox'** or you can grab the zip here from the GitHub repository:

<div align="center">

SPARKFUN U-BLOX ARDUINO LIBRARY (ZIP)

</div>

Once you have the library installed checkout the various examples.

- **Example1:** Read NMEA sentences over I^{2}C using Ublox module SAM-M8Q, NEO-M8P, etc
- **Example2:** Parse NMEA sentences using MicroNMEA library. This example also demonstrates how to overwrite the `processNMEA` function so that you can direct the incoming NMEA characters from the Ublox module to any library, display, radio, etc that you prefer.
- **Example3:** Get latitude, longitude, altitude, and satellites in view (SIV). This example also demonstrates how to turn off NMEA messages being sent out of the I^{2}C port. You'll still see NMEA on UART1 and USB, but not on I^{2}C. Using only UBX binary messages helps reduce I^{2}C traffic and is a *much* lighter weight protocol.
- **Example4:** Displays what type of a fix you have the two most common being none and a full 3D fix. This sketch also shows how to find out if you have an RTK fix and what type (floating vs. fixed).
- **Example5:** Shows how to get the current speed, heading, and dilution of precision.
- **Example6:** Demonstrates how to increase the output rate from the default 1 per second to many per second; up to 30Hz on some modules!
- **Example7:** Older modules like the SAM-M8Q utilize an older protocol (version 18) whereas the newer modules like the ZED-F9P depricate some commands using the latest protocol (version 27). This sketch shows how to query the module to get the protocol version.
- **Example8:** Ublox modules use I^{2}C address 0x42 but this is configurable via software. This sketch will allow you to change the module's I^{2}C address.
- **Example9:** Altitude is not a simple measurement. This sketch shows how to get both the ellipsoid based altitude and the MSL (mean sea level) based altitude readings.
- **Example10:** Sometimes you just need to do a hard reset of the hardware. This sketch shows how to set your Ublox module back to factory default settings.
- **NEO-M8P**
  - **NEO-M8P Example1:** Send UBX binary commands to enable RTCM sentences on U-blox NEO-M8P-2 module. This example is one of the steps required to setup the NEO-M8P as a base station. For more information have a look at the Ublox manual for setting up an RTK link.
  - **NEO-M8P Example2:** This example extends the previous example sending all the commands to the NEO-M8P-2 to have it operate as a base. Additionally the `processRTCM` function is exposed. This allows the user to overwrite the function to direct the RTCM bytes to whatever connection the user would like (radio, serial, etc).

- - **NEO-M8P Example3:** This is the same example as NEO-M8P's Example2. However, the data is sent to a serial LCD via I^{2}C.
  - **ZED-F9P**
    - **ZED-F9P Example1:** This module is capable of high precision solutions. This sketch shows how to inspect the accuracy of the solution. It's fun to watch our location accuracy drop into the millimeter scale.
    - **ZED-F9P Example2:** The ZED-F9P uses a new Ublox configuration system of VALGET/VALSET/VALDEL. This sketch demonstrates the basics of these methods.
    - **ZED-F9P Example3:** Setting up the ZED-F9P as a base station and outputting RTCM data.
    - **ZED-F9P Example4:** This is the same example as ZED-F9P's Example3. However, the data is sent to a serial LCD via I^{2}C.

This SparkFun Ublox library really focuses on I^{2}C because it's faster than serial and supports daisy-chaining. The library also uses the UBX protocol because it requires far less overhead than NMEA parsing and does not have the precision limitations that NMEA has.

## Setting the ZED-F9P as a Correction Source

If you're located further than 20km from a correction station you can create your own station using the ZED-F9P. Ublox provides a setup guide within the ZED-F9P Integration Manual showing the various settings needed via U-Center. We'll be covering how to setup the ZED-F9P using I^{2}C commands only. This will enable a headless (computerless) configuration of a base station that outputs RTCM correction data.

Before getting started we recommend you configure the module using U-Center. Checkout our tutorial on using U-Center then read section **3.5.8 Base Station Configuration** of the Ublox Integration Manual for getting the ZED-F9P configured for RTK using U-Center. Once you've been successful controlling the module in the comfort of your lab using U-Center, then consider heading outdoors.

For this exercise we'll be using the following parts:

- 1x SparkFun GPS-RTK2 Board or RTK-SMA Board
- 1x U.FL to SMA Cable for the RTK2
- 1x SparkFun BlackBoard makes I^{2}C easy
- 1x USB C 2.0 Cable if you need one
- 1x Two Qwiic Cables
- 1x 20x4 SerLCD with Qwiic Adapter soldered on
- 1x Antenna GNSS 3-5V Magnetic Mount
- 1x GPS Antenna Ground Plate
- 1x 20+ft SMA extension can be handy when first experimenting with base stations so you can sit indoors with a laptop and analyze the output of the GPS-RTK
- 1x A standard camera tripod

The ZED-F9P can be configured using Serial, SPI, or I^{2}C. We're fans of the daisy chain-ability of I^{2}C so we'll be focusing on the Qwiic system. For this exercise we'll be connecting the an LCD and GPS-RTK2 to a BlackBoard using two Qwiic cables. You can also use the RTK-SMA as an alternative.
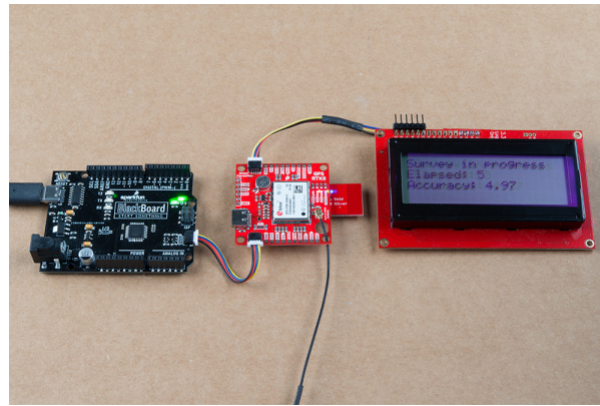
For the antenna, you'll need a clear view of the sky. The better your antenna position the better your accuracy and performance of the system. We designed the GPS Antenna Ground Plate to make this setup easy. The plate has a ¼" threaded hole that threads directly onto a camera tripod. The plate thickness was chosen to be thick enough so that the threaded screw is flush with the plate so it won't interfere with the antenna. Not sure why we're using a ground plate? Read the Ublox white paper on using low-cost GNSS antennas with RTK. Mount your magnetic mount antenna and run the SMA cable to the U.FL to SMA cable to the GPS-RTK2 board. If you have the RTK-SMA, you just need to run the SMA cable to the board's connector.

There are only three steps to initiating a base station:

- Enable Survey-In mode for 1 minute (60 seconds)
- Enable RTCM output messages
- Being Transmitting the RTCM packets over the backhaul of choice

Be sure to grab the SparkFun Arduino Library for Ublox. You can easily install this via the library manager by searching '**SparkFun Ublox**'. Once installed click on **File** -> **Examples** -> **SparkFun_Ublox_Arduino_Library**.

The ZED-F9P subfolder houses a handful of sketches specific to its setup. *Example3* of the library demonstrates how to send the various commands to the ZED-F9P to enable Survey-In mode. Let's discuss the important bits of code.

```
response = myGPS.enableSurveyMode(60, 5.000); //Enable Survey in, 60 seconds, 5.0m
```

The library is capable of sending UBX binary commands with all necessary headers, packet length, and CRC bytes over I²C. The `enableSurveyMode(minimumTime, minimumRadius)` command does all the hard work to tell the module to go into survey mode. The module will begin to record lock data and calculate a 3D standard deviation. The survey-in process ends when both the minimum time and minimum radius are achieved. Ublox recommends 60 seconds and a radius of 5m. With a clear view of the sky, with a low cost GNSS antenna mounted to a ground plate we've seen the survey complete at 61 seconds with a radius of around 1.5m.

```
response &= myGPS.enableRTCMmessage(UBX_RTCM_1005, COM_PORT_I2C, 1); //Enable message 1005 to ou
tput through I2C port, message every second
response &= myGPS.enableRTCMmessage(UBX_RTCM_1074, COM_PORT_I2C, 1);
response &= myGPS.enableRTCMmessage(UBX_RTCM_1084, COM_PORT_I2C, 1);
response &= myGPS.enableRTCMmessage(UBX_RTCM_1094, COM_PORT_I2C, 1);
response &= myGPS.enableRTCMmessage(UBX_RTCM_1124, COM_PORT_I2C, 1);
response &= myGPS.enableRTCMmessage(UBX_RTCM_1230, COM_PORT_I2C, 10); //Enable message every 10
 seconds
```

These six lines enable the six RTCM output messages needed for a second ZED-F9P to receive correction data. Once these sentences have been enabled (and assuming a survey process is complete) the ZED-F9P base module will begin outputting RTCM data every second after the NMEA sentences (the RTCM_1230 sentence will be output once every 10 seconds). You can view an example of what this output looks like here.

The size of the RTCM correction data varies but in general it is approximately 2000 bytes every second (~2500 bytes every 10th second when 1230 is transmitted).

```
//This function gets called from the SparkFun Ublox Arduino Library.
//As each RTCM byte comes in you can specify what to do with it
//Useful for passing the RTCM correction data to a radio, Ntrip broadcaster, etc.
void SFE_UBLOX_GPS::processRTCM(uint8_t incoming)
{
  //Let's just pretty-print the HEX values for now
  if (myGPS.rtcmFrameCounter % 16 == 0) Serial.println();
  Serial.print(" ");
  if (incoming < 0x10) Serial.print("0");
  Serial.print(incoming, HEX);
}
```

If you have a 'rover' in the field in need of correction data you'll need to get the RTCM bytes to the rover. The SparkFun Ublox library automatically detects the difference between NMEA sentences and RTCM data. The `processRTCM()` function allows you to 'pipe' just the RTCM correction data to the channel of your choice. Once the base station has completed the survey and has the RTCM messages enabled, your custom `processRTCM()` function can pass each byte to any number of channels:

- A wireless system such as LoRa or Cellular
- Posting the bytes over the internet using WiFi or wired ethernet over an Ntrip caster
- Over a wired solution such as RS485

The power of the `processRTCM()` function is that it doesn't care; it presents the user with the incoming byte and is agnostic about the back channel.

> **Heads up!** We've been experimenting with various LoRa solutions and the bandwidth needed for the best RTCM (~500 bytes per second) is right at the usable byte limit for many LoRa setups. It's possible but you may need to adjust your LoRa settings to reach the throughput necessary for RTK.

What about configuring the rover? Ublox designed the ZED-F9P to automatically go into RTK mode once RTCM data is detected on any of the ports. Simply push the RTCM bytes from your back channel into one of the ports (UART, SPI, I^2C) on the rover's ZED-F9P and the location accuracy will go from meters to centimeters. The rover's NMEA messages will contain the improved Lat/Long data and you'll know where you are with mind-bending accuracy. It's a lot of fun to watch!

Can I Really Use NMEA with a High Precision GPS Receiver?

Yes! Except that NMEA sentences are right on the edge of enough precision. NMEA sentences look something like this:

```
$GNGGA,012911.00,4003.19080,N,10416.95542,W,1,12,0.75,1647.1,M,-21.3,M,,*4F
```

NMEA outputs coordinates in the ddmm.mmmmm format. So what is the weight of the least significant digit? Said differently, what is the impact of one digit change?

```
104 16.95542
```

vs

```
104 16.95543
```

If we know 1 degree of latitude is 111.3km at the equator, we can glean the change of a fraction of a minute:

- 1 degree = 60 minutes
- 1 minute = 1 degree/60 = 111.32km / 60 = 1.855km
- 1 minute = 1855m
- 0.1min = 185.5m
- 0.01min = 18.55m
- 0.001min = 1.855m
- 0.0001min = .1855m = 185.5mm
- 0.00001min = 0.0185m = 18.55mm = 1.855cm

Using the NMEA sentence, the ZED-F9P will only be able to communicate a change of ~1.5cm location change for each digit in the 5th position. This is pretty close to the 1.0cm accuracy of the module. If you want additional precision, you should consider using the UBX protocol which can output up to 8 digits of precision in dd.dddddddd format which will get you down to 1.11mm of precision! Be sure to checkout the examples in the SparkFun Ublox Arduino Library. We have various examples outputting the full 8 digits of precision over I^2C without the burden of parsing NMEA sentences.

Resources and Going Further

Ready to get hands-on with GPS?

We've got a page just for you! We'll walk you through the basics of how GPS works, the hardware needed, and project tutorials to get you started.

TAKE ME THERE!

Have fun with your new found super power: sub decimeter grade GPS! For more on the GPS-RTK2 and RTK-SMA, check out the links below:
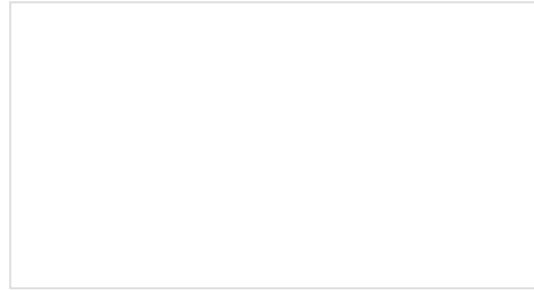
- SparkFun GPS-RTK2
  - Schematic (PDF)
  - Eagle Files (ZIP)
  - Board Dimensions
- SparkFun GPS-RTK-SMA
  - Schematic (PDF)
  - Eagle Files (ZIP)
  - Board Dimensions
- ZED-F9P Datasheet (PDF)
- ZED-F9P UBX and NMEA Protocol Manual
- ZED-F9P Integration Manual including setting up the module for base station using u-center
- ZED-F9P Product Summary
- ZED-F9P FW1.00 Release Notes
- U-blox ECCN notice
- Example RTCM output from the ZED-F9P
- GitHub
  - Arduino Library

- Product Repo
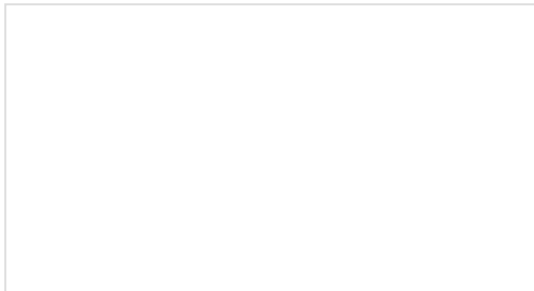   - GPS-RTK2
    - GPS-RTK-SMA
- SFE Product Showcase

Need some inspiration? Check out some of these related tutorials:
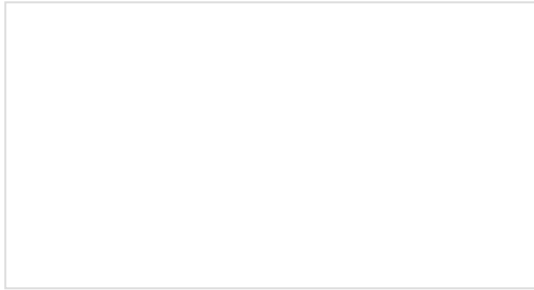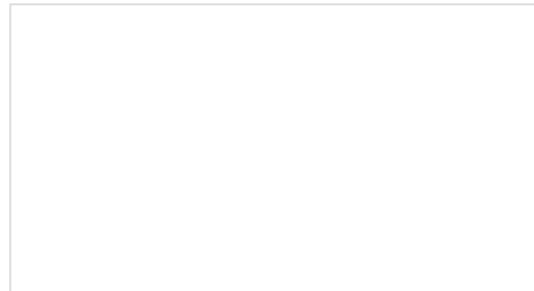
Building an Autonomous Vehicle: The Batmobile
Documenting a six-month project to race autonomous Power Wheels at the SparkFun Autonomous Vehicle Competition (AVC) in 2016.

What is GPS RTK?
Learn about the latest generation of GPS and GNSS receivers to get 2.5cm positional accuracy!

GPS-RTK Hookup Guide
Find out where you are! Use this easy hook-up guide to get up and running with the SparkFun high precision GPS-RTK NEO-M8P-2 breakout board.

Getting Started with U-Center for u-blox
Learn the tips and tricks to use the u-blox software tool to configure your GPS receiver.