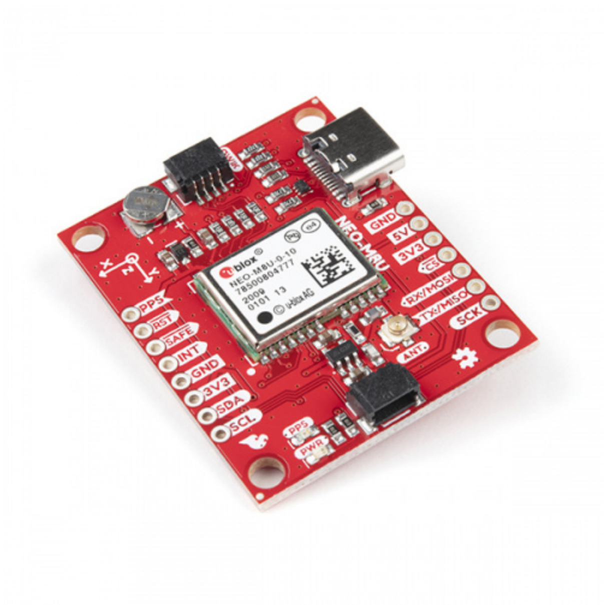
# SparkFun GPS Dead Reckoning NEO-M8U Hookup Guide

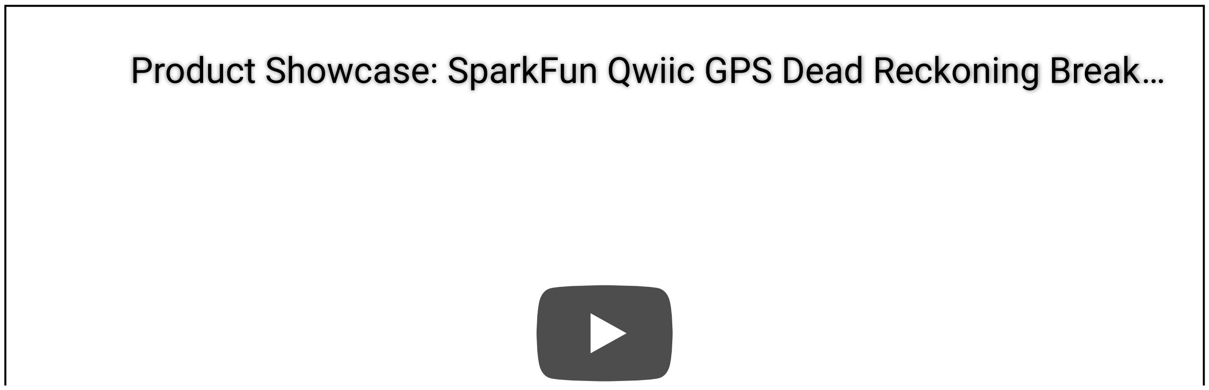
## Introduction

The u-blox NEO-M8U is a powerful GPS units that takes advantage of untethered dead reckoning (UDR) technology for navigation. The module provides continuous positioning for vehicles in urban environments and during complete signal loss (e.g. short tunnels and parking garages). We will quickly get you set up using the Qwiic ecosystem and Arduino so that you can start reading the output!



### SparkFun GPS Dead Reckoning Breakout - NEO-M8U (Qwiic)

GPS-16329



Product Showcase: SparkFun Qwiic GPS Dead Reckoning Break…

## Required Materials

To follow along with this tutorial, you will need the following materials. You may not need everything though depending on what you have. Add it to your cart, read through the guide, and adjust the cart as necessary.

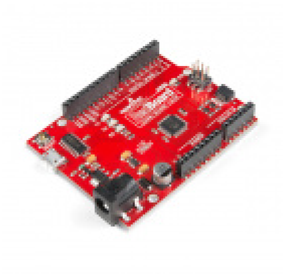
**Untethered Dead Reckoning GPS Wish List** SparkFun Wish List

| | |
|---|---|
| | USB micro-B Cable - 6 Foot<br>CAB-10215<br>USB 2.0 type A to micro USB 5-pin. This is a new, smaller connector for USB devices. Micro USB connectors are a… |
| | Qwiic Cable - 100mm<br>PRT-14427 |
| | SparkFun RedBoard Qwiic<br>DEV-15123 |
| | SparkFun GPS Dead Reckoning Breakout - NEO-M8U (Qwiic)<br>GPS-16329 |
| | Interface Cable SMA to U.FL<br>WRL-09145<br>This is a 4" connector cable that interfaces U.FL RF connectors to regular SMA connectors. **Documents:*** [Data… |

## Microcontroller

One method of connecting to the GPS is using a Qwiic cable and microcontroller. We will use the RredBoard Qwiic fo the scope of this tutorial.

USB micro-B Cable - 6 Foot
● CAB-10215



SparkFun RedBoard Qwiic
● DEV-15123

Qwiic Cable - 100mm
● PRT-14427

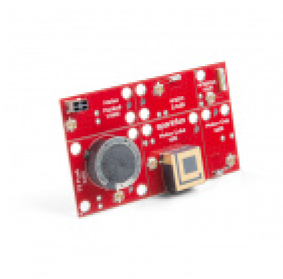
## Additional GPS Antenna Options

Below are some other GPS Antenna options. Some of the options below have an SMA connector, so make sure to get the u.FL to SMA cable if you decide to use those. Link for that is below in the GPS accessories. If you want to try different chip antennas, then try the GNSS Antenna Evalutation Board listed below and make sure to get the u.FL to u.FL connector in the accessories.

GPS/GNSS Magnetic Mount Antenna - 3m (SMA)
● GPS-14986

GPS/GNSS Embedded Antenna - 1m (SMA)
● GPS-14987

GPS Embedded Antenna SMA
● GPS-00177



SparkFun GNSS Chip Antenna Evaluation Board
● GPS-15247

## GPS Antenna Accessories

Interface Cable SMA to U.FL
● WRL-09145

GPS Antenna Ground Plate
◒ GPS-15004

U.FL to U.FL Mini Coax Cable - 200mm
● WRL-15114

Other Qwiic Cable Accessories

SparkFun Qwiic Cable Kit
● KIT-15081

Qwiic Cable - 100mm
● PRT-14427

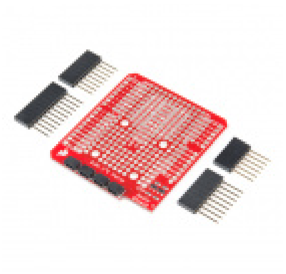
Qwiic Cable - 200mm

Qwiic Cable - 50mm

● PRT-14428

● PRT-14426

**Heads up!** If you are using the RedBoard **without** a Qwiic connector, we recommend getting the Qwiic Shield for Arduino.



SparkFun Qwiic Shield for Arduino
● DEV-14352

## Suggested Reading

If you aren't familiar with the Qwiic system, we recommend reading here for an overview.

*Qwiic Connect System*

We would also recommend taking a look at the following tutorials if you aren't familiar with them.

GPS Basics
The Global Positioning System (GPS) is an engineering marvel that we all have access to for a relatively low cost and no subscription fee. With the correct hardware and minimal effort, you can determine your position and time almost anywhere on the globe.

Serial Peripheral Interface (SPI)
SPI is commonly used to connect microcontrollers to peripherals such as sensors, shift registers, and SD cards.

I2C

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

How to Work with Jumper Pads and PCB Traces

Handling PCB jumper pads and traces is an essential skill. Learn how to cut a PCB trace, add a solder jumper between pads to reroute connections, and repair a trace with the green wire method if a trace is damaged.

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.

Three Quick Tips About Using U.FL

Quick tips regarding how to connect, protect, and disconnect U.FL connectors.

## What is Dead Reckoning?

Dead Reckoning is the process of determining current position by combining previously determined positional data with speed and heading. This process can also be applied to determine *future* positions as well! The NEO-M8U uses what is called **Untethered** Dead Reckoning (UDR) which calculates speed and heading (amongst many other points of data) through the use of an internal **inertial measurement unit** (IMU). The addition of an IMU allows the M8U to produce more accurate readings in between GNSS data refreshes!

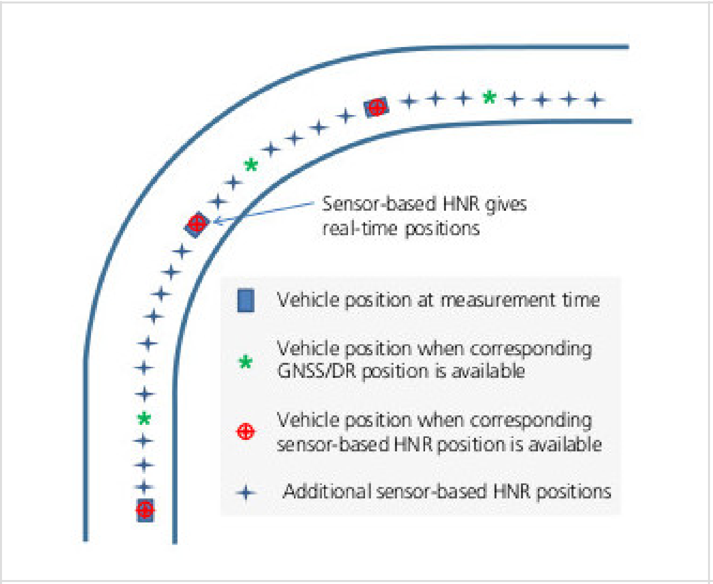
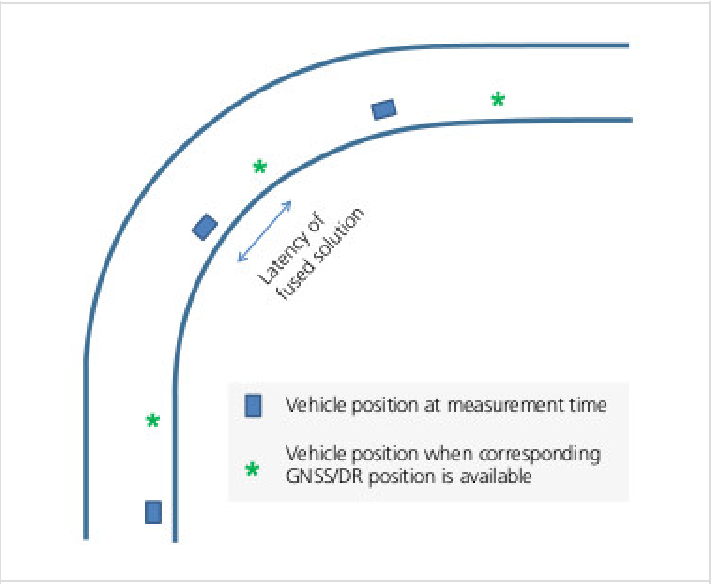
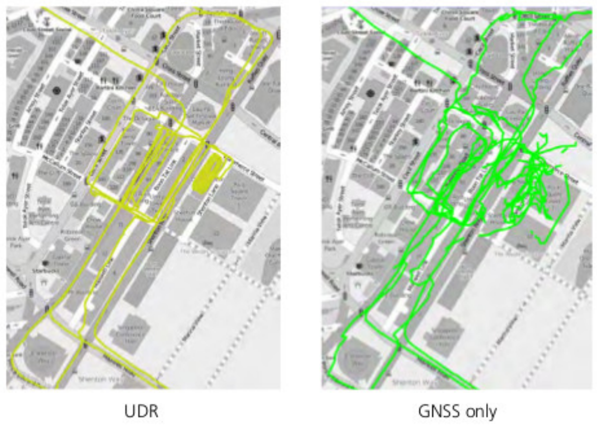
| | |
|---|---|
|  Sensor-based HNR gives real-time positions<br>Vehicle position at measurement time<br>Vehicle position when corresponding GNSS/DR position is available<br>Vehicle position when corresponding sensor-based HNR position is available<br>Additional sensor-based HNR positions |  Latency of fused solution<br>Vehicle position at measurement time<br>Vehicle position when corresponding GNSS/DR position is available |
| *Illustration of Using an IMU with GNSS Data* | *Illustration of Using only GNSS Data Only* |

*Illustration of Refresh Rate to Accurately Determine Position. Image Courtesy of u-blox from the UDR Whitepaper.*

In addition, the module can also give accurate and useful GNSS data in areas where satellite connections are difficult to maintain: areas like the dense urban environments of major cities, long tunnels, parking garages, any large UFO's that may descend from the sky, etc.

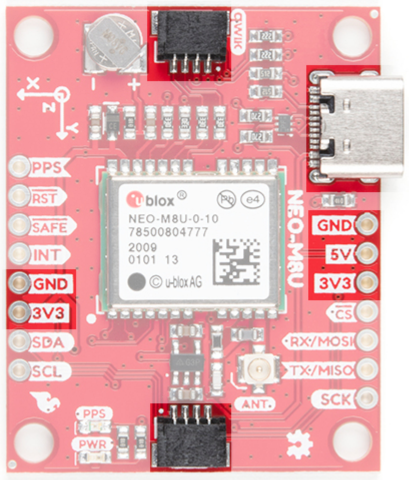


*Untethered Dead Reckoning vs GNSS Only Comparison in an Urban Canyon. Image Courtesy of u-blox from the UDR Whitepaper.*

## Hardware Overview

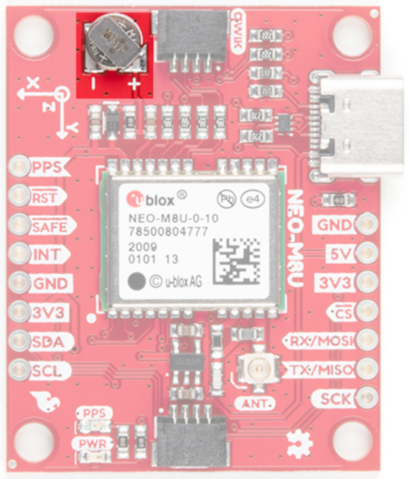
### Power

Power for this board is **3.3V** and we have provided multiple power options. This first and most obvious is the **USB-C connector**. Secondly, are the **Qwiic Connectors** on the top and bottom of the board. Thirdly, there is a **5V pin** on the PTH header along the side of the board that is regulated down to **3.3V**. Make sure that power your provide to this pin does *not* exceed 6 volts. Finally, just below the 5V pin is a **3.3V** pin that should only be provided a clean 3.3V power signal.



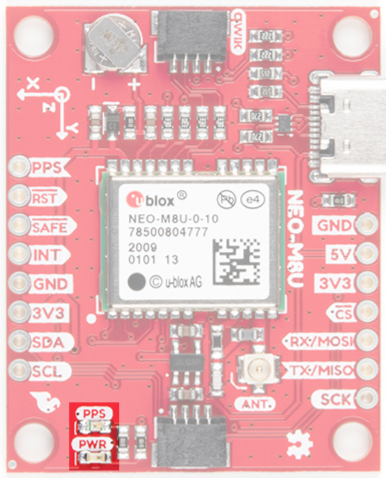
## Battery

The small metal disk in the upper left corner is a small lithium battery. This battery does not provide power to the IC like the 3.3V system does, but to relevant systems *inside* the IC that allow for a quick reconnection to satellites. The time to first fix will about **~26 seconds**, but after it has a lock, that battery will allow for about a **1.5 second** time to first fix. This is known as a **hot start** and lasts for four hours after the board is powered down. The battery provides over a years worth of power to the backup system and charges slowly when the board is powered. To charge it to full, leave your module plugged in for 48 hours.



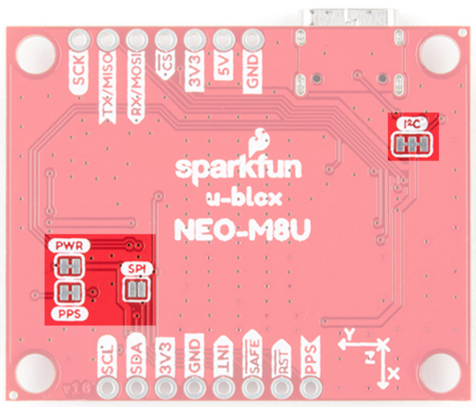
## LEDs

There's is a red power LED just to the left of the bottom Qwiic connector and near the board's edge to indicate that the board is powered. There is another LED just above the power LED labeled `PPS` that is connected to the *Pulse Per Second* line. When connected to a satellite, this line generates a pulse that is synchronized with a GPS or UTC time grid. By default, you'll see one pulse a second.



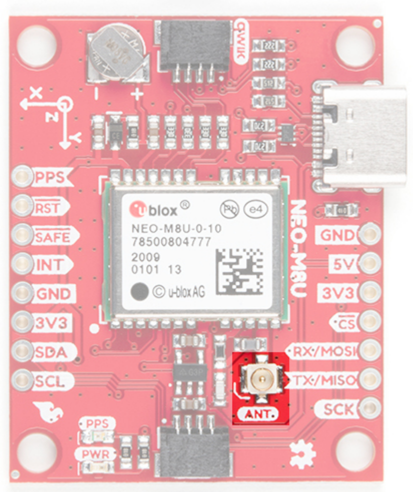
## Jumpers

There are four jumpers on the underside of the product, each labeled with its function. At the upper right of the picture is a three way jumper labeled `I²C` that connects two pull-up resistors to the I$^2$C data lines. If you have many devices on your I$^2$C data lines, then you may consider cutting these. On the left side of the board is a jumper labeled `PWR`. If you cut this trace it will disconnect the **Power** LED. Just below is the `PPS` jumper that when cut disconnects the **PPS** LED. Finally, there's a jumper labeled `SPI` which enables the SPI data bus thus disabling the UART functions on those lines. For more information, check out our tutorial on working with jumper pads and PCB traces.



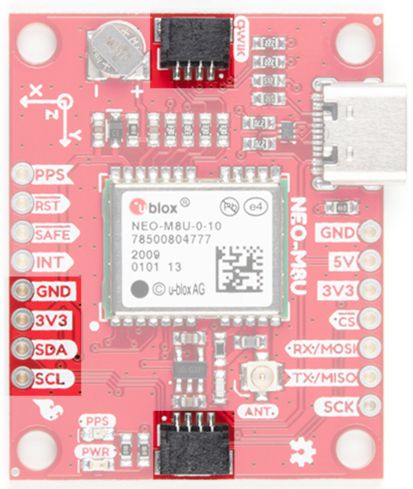
## U.FL Connector

The SparkFun GPS NEO-M8U has a u.FL connector in which you can connect a patch antenna.



### Qwiic and I$^2$C

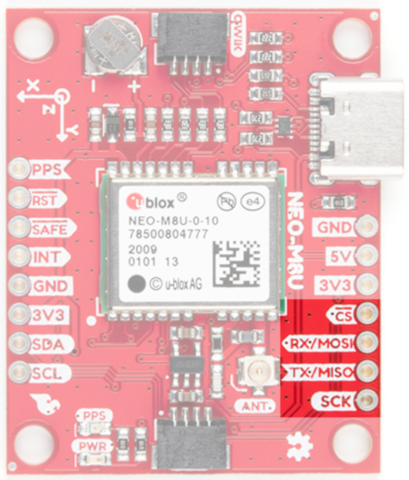
There are two pins labeled `SDA` and `SCL` which indicates the I$^2$C data lines. Similarly, you can use either of the Qwiic connectors to provide power and utilize I$^2$C. The Qwiic ecosystem is made for fast prototyping by removing the need for soldering. All you need to do is plug a Qwiic cable into the Qwiic connector and voila!



The only I$^2$C address for this and all u-Blox GPS products is **0x42**, though each can have their address changed through software.

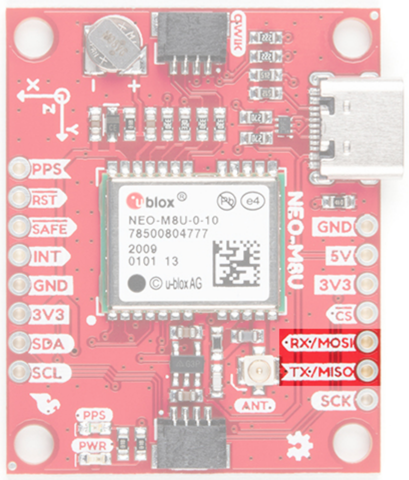
### SPI

There are four pins on the right most header that are labeled with their corresponding SPI functionality. As mentioned in the jumpers section, you'll need to close the `SPI` jumper on the underside to enable SPI.



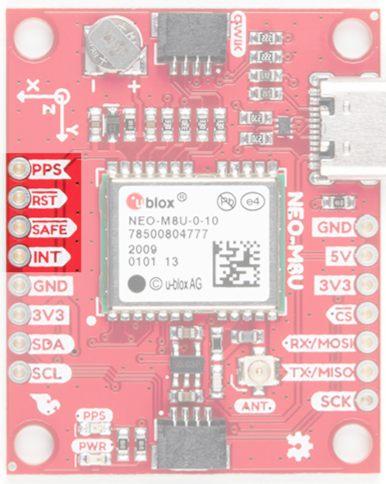
## UART

There are two pins on the right most header labeled for their UART functionality.



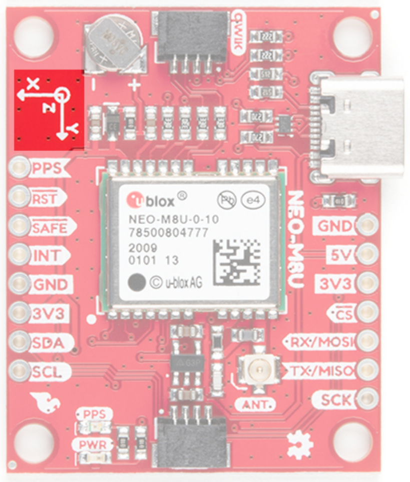
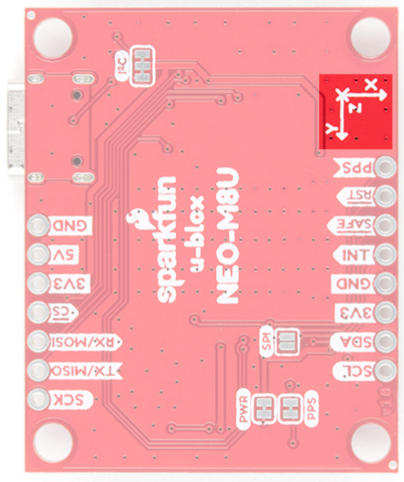
## Broken Out Pins

There are four other pins broken out: Pulse per second ( `PPS` ), Reset ( `RST` ), Safeboot ( `SAFE` ), and finally the interrupt pin ( `INT` ). The first pin `PPS` , outputs pulse trains synchronized with the GPS or UTC time grid. The signal defaults to once per second but is configurable over a wide range. Read the **u-blox Receiver Protocol Specification** in the Resources and Going Further tab for more information. The reset pin resets the chip. The next pin, `SAFE` is used to start up the IC in safe boot mode, this could be useful if you somehow manage to corrupt the module's Flash memory. The final pin `INT` can be used to wake the chip from power save mode.



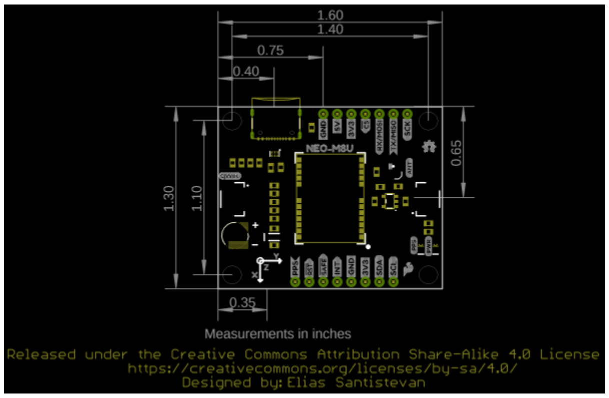
## 3D IMU Orientation and Reference

For easy reference, we've documented the IMU's vectors with 3D Cartesian coordinate axes on the top and bottom side of the board. Make sure to orient and mount the board correctly so that the NEO-M8U can accurately calculate navigation information. This is explained in detail in the Dead Reckoning Overview. Remember, it's all relative.

| | |
|---|---|
|  |  |
| *Top View with the the Axis for Reference* | *Bottom View with the the Axis for Reference* |

## Board Dimension

Overall, the board is 1.30"x1.60".


1.60
1.40
0.75
0.40
NEO-M8U
0.65
1.30
1.10
0.35
Measurements in inches
Released under the Creative Commons Attribution Share-Alike 4.0 License
https://creativecommons.org/licenses/by-sa/4.0/
Designed by: Elias Santistevan

## GPS Capabilities

The SparkFun NEO-M8U is able to connect to up to three different GNSS constellations at a time. Below are the listed capabilities of the GPS unit taken from the datasheet when connecting to different GNSS constellations.

| Constellations | | GPS+GLO | GPSL | GLO | BDS | GAL |
|---|---|---|---|---|---|---|
| **Horizontal Position Accuracy** | Autonomus | 2.5m | 2.5m | 4.0m | 3.0m | To Be Confirmed |
| | with SBAS | 1.5m | 1.5m | | | |
| **Max Navigation Update Rate** | PVT | 25Hz | 25Hz | 25Hz | 25Hz | 25Hz |
| **Time-To-First-Fix** | Cold Start | 24s | 25s | 26s | 28s | 29s |
| | Hot Start | 2s | 2s | 2s | 2s | 2s |
| **Sensitivity** | Tracking and Navigation | -160dBm | -160dBm | -160dBm | -160dBm | -154dBm |
| | Reacquisition | -160dBm | -159dBm | -156dBm | -155dBm | -152dBm |
| | Cold Start | -148dBm | -147dBm | -145dBm | -143dBm | -133dBm |
| | Hot Start | -157dBm | -156dBm | -155dBm | -155dBm | -151dBm |
| **Velocity Accuracy** | | 0.05m/s | 0.05m/s | 0.05m/s | 0.05m/s | 0.05m/s |
| **Heading Accuracy** | | 1deg | 1deg | 1deg | 1deg | 1deg |

## Dead Reckoning Overview

As mentioned in the "What is Dead Reckoning?" section, the u-blox M8U module has an internal inertial measurement unit or **IMU** for short. The IMU calculates position based on the last GNSS refresh and its own movement data points. To use the SparkFun GPS Dead Reckoning Board, there are a few guidelines to orienting and mounting the module to a vehicle outlined in the u-blox ReceiverDescrProtSpec Datasheet.

## Orientation for the SparkFun Dead Reckoning

The SparkFun Dead Reckoning adheres to two particular frames of reference: one frame of reference for the car and the second a geodetic frame of reference anchoring it to the globe. The latter, known as the **local level frame** uses the following as its' axes:

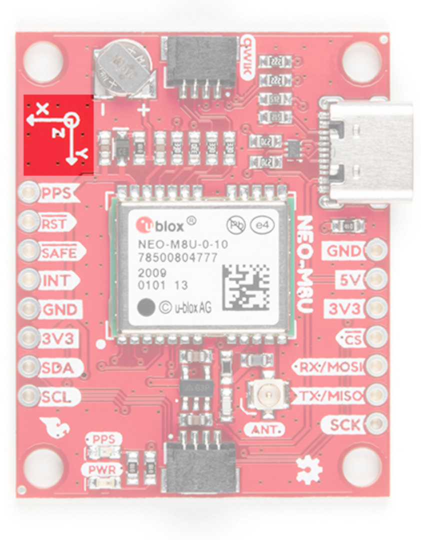
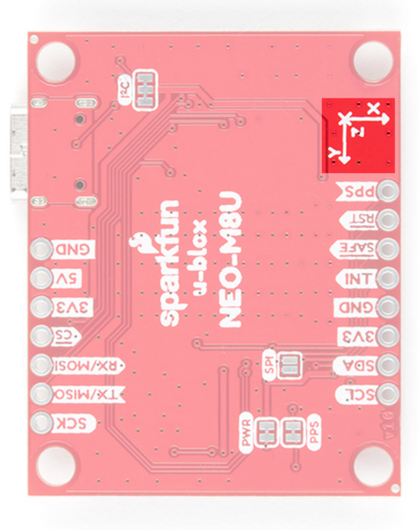
- **X-axis** points to the **North**
- **Y-axis** points to the **East**
- **Z-axis** uses the right hand system by pointing **down**.

This frame will be referred to by its acronym **NED** (North-East-Down) in the image below.

The second frame of references is the **Body-Frame** reference and uses the following as *its'* axes.

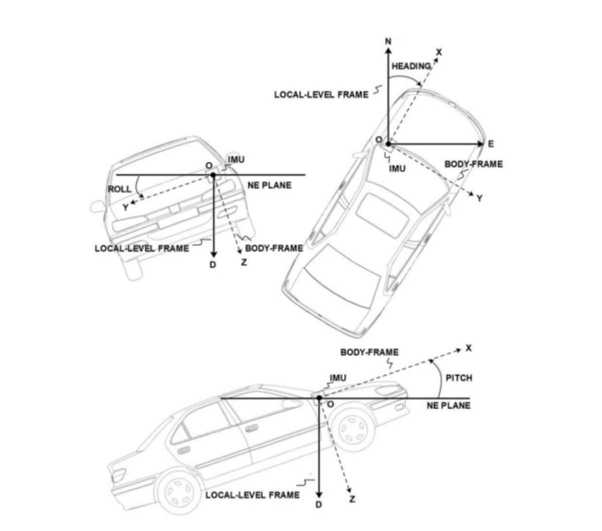
- **X-axis** points to the **front** of the vehicle
- **Y-axis** points to the **right** of the vehicle
- **Z-axis** uses the right hand system by pointing **down**.

You can reference the **Body-Frame** axes directly on the SparkFun Dead Reckoning NEO-M8U silk in the corner closest to the battery.

| | |
|---|---|
|  |  |
| *Top View with the the Axis for Reference* | *Bottom View with the the Axis for Reference* |

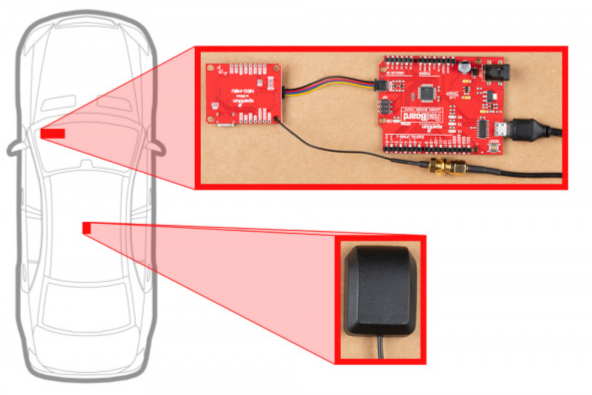
## Vehicle Attitude

The transformation of the vehicle within these two frames are given as **heading**, **pitch**, and **roll**. In the datasheet these three angles are called the **vehicle's attitude**. Below is an image that illustrates how all of these elements fit together.



## Mounting the SparkFun Dead Reckoning NEO-M8U

The only guideline here is that the SparkFun Dead Reckoning is stable within 2 degrees, and of course that the X-axis points towards the front of the car as mentioned above.



In the image above the SparkFun Dead reckoning is seen in the front, driver's side of the car and it may be tempting to think that this is also a necessary requirement. However, it can be mounted anywhere within the vehicle (or RC-car, or boat). Keep in mind that the **pitch** and **roll** is relative to the SparkFun Dead Reckoning's position.

A slightly more detailed description of these frames can be found on page 106 of the u-blox Receiver Description Datasheet.

## Calibration

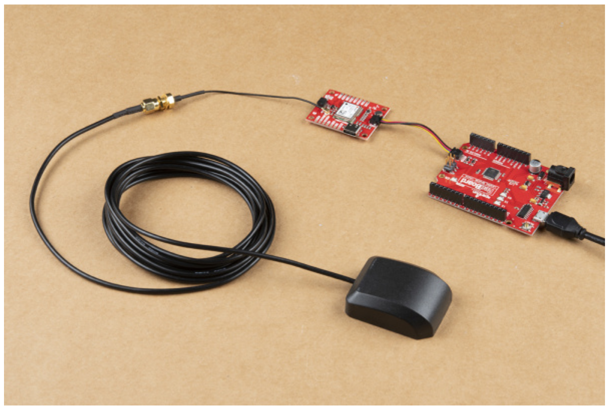
After you've mounted the SparkFun Dead Reckoning M8U, there is still a calibration phase to complete that must satisfy the following movements:

- First, the car needs to be stopped with the engine turned on.
- Secondly, the car must do left and right hand turns.
- Lastly, the car must reach a speed over **30 km/h**.

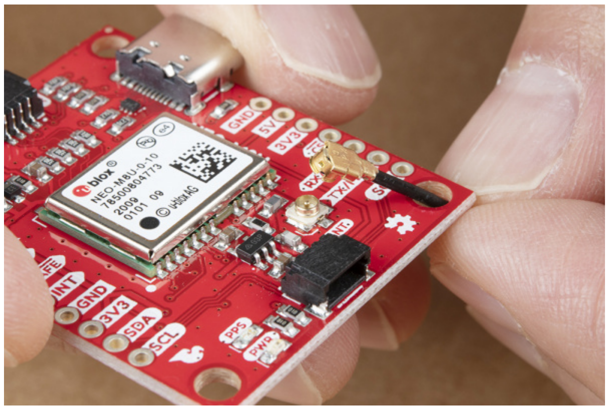
In SparkFun's u-blox Arduino library, SparkFun has included an Example (shown below), that prints out the module's calibration status.

## Hardware Assembly

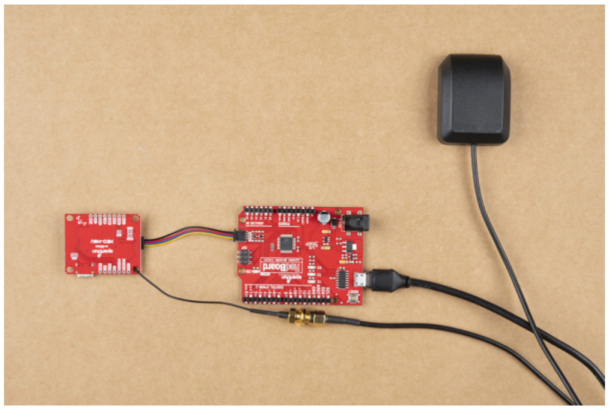
For this example, I used a RedBoard Qwiic and associated USB cable. Connecting the boards with Qwiic cable, the assembly is very simple. Plug a Qwiic cable between the RedBoard and SparkFun NEO-M9U. Then plugged in one of our patch antennas to the u.FL connector. If you need tips on plugging in the U.FL connector, then check out our U.FL tutorial. If you're going to be soldering to the through hole pins for I$^2$C functionality, then just attach lines to power, ground, and the I$^2$C data lines to a microcontroller of your choice. Your setup should look similar to the image below.



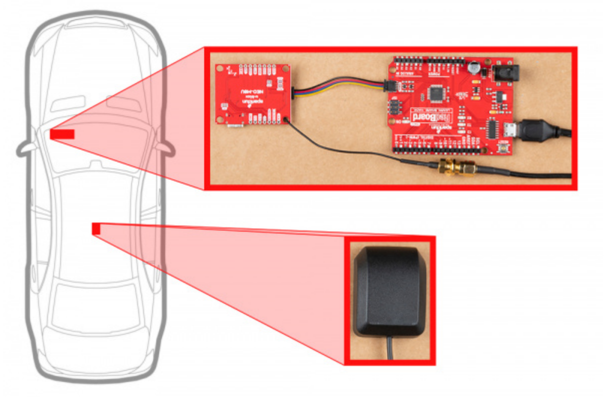
For secure connections, you may want to thread the U.FL cable through a mounting hole before connecting. Adding tape or some hot glue will provide some strain relief to prevent the cable from disconnecting.



When using the NEO-M8U, you will want to orient the board according to the guidelines explained earlier. Below is a top-down view with the board pointing down.



Make sure to secure the board above a vehicle's dashboard using some tape or sticky tack when prototyping and testing. For best signal reception, it is suggested to guide the antenna from the inside of the car and through a window before attaching the GPS on top of a car.



## 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.

All of our u-blox based GPS boards share the same library: this board, their predeccesors and the higher precision u-blox cousins. 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 to manually install.:

**SPARKFUN U-BLOX ARDUINO LIBRARY (ZIP)**

There are 13 example sketches provided to get you up and receiving messages from space. The examples listed below highlight the *additional* capabilities of the SparkFun Dead Reckoning NEO-M8U. For the scope of this tutorial, we will not focus on the basic GPS polling sketches as shown in the other u-blox hookup guides.

**Note:** Example 2 uses the '**MicroNMEA**' library by **Steve Marple**. Make sure to install the library as well by searching for it in the Arduino library manager. You could also grab the zip here from the GitHub repository to manually install.

**MICRONMEA ARDUINO LIBRARY (ZIP)**

## Example 1 - Calibrate Sensor

**Heads up!** Make sure stay focused when driving and obey all traffic laws when driving. Ensure that you are not distracted while operating a vehicle and are aware of your surroundings. For each example, make sure to plan your route accordingly to achieve the maneuvers. If you need, grab a friend and adult as you ride in the passenger seat. After you attempt these movements, park your car in a safe location and turn your engine off before checking the status or monitoring the readings!

Now that the SparkFun Dead Reckoning is mounted and oriented correctly with regards to the vehicle, it's time to calibrate the sensor. To do this, a few movements with the vehicle must be done all while maintaining good GNSS reception.

- First, the car needs to be stopped with the engine turned on.
- Secondly, the car must do left and right hand turns.
- Lastly, the car must reach a speed over **30 km/h**.

For the first example (located in **File Examples > SparkFun Ublox Arduino Library** > **Dead Reckoning** > **Example1_calibrateSensor**), the **calibration** status will be printed to the Arduino's serial monitor to indicate when calibration is ongoing and when it has completed.

In the first code block, there's a call to `myGPS.setI2COUtput(COM_TYPE_UBX)` towards the bottom which turns off NMEA noise.

```
void setup()
{
  Serial.begin(115200);
  while (!Serial); //Wait for user to open terminal
  Serial.println("SparkFun Ublox Example");

  Wire.begin();

  if (myGPS.begin() == false) //Connect to the Ublox module using Wire port
  {
    Serial.println(F("Ublox GPS not detected at default I2C address. Please check wiring. Freezi
ng."));
    while (1);
  }

  myGPS.setI2COutput(COM_TYPE_UBX); //Set the I2C port to output UBX only (turn off NMEA noise)
}
```

The second code block below is where the heart of the sketch lives. The code pings the module to get its "ESF Info" (External Sensor Fusion Info), to see what Fusion Mode has been achieved. A *zero* indicates calibration while a *one* indicates **calibrated**.

```
void loop()
{

  if (myGPS.getEsfInfo()){
    Serial.print("Fusion Mode: ");
    Serial.println(myGPS.imuMeas.fusionMode);
    if (myGPS.imuMeas.fusionMode == 1)
      Serial.println("Sensor is calibrated!");
  }

  delay(250);
}
```

If you have not already, select your Board (in this case the **Arduino Uno**), and associated COM port. Upload the code to the board and and set the serial monitor to **115200 baud.** Perform those fancy maneuvers (while obeying the traffic laws) before parking your car in a safe location. Then turn your engine off before checking the status! You should see a message indicating that the NEO-M8U is calibrated. If you do not, try driving around with the board once again!

## Example 2 - IMU Data

After you have your sensor calibrated (see example 1), you can now poll the internal IMU to see what data is being fed to the GNSS calculations. Open the second example (located in **File Examples > SparkFun Ublox Arduino Library** > **Dead Reckoning** > **Example2_getIMUData**) to follow along! First, the sketch checks to see that the board is calibrated before it attempts to read IMU data.

```
void setup()
{
  Serial.begin(115200);
  while (!Serial); //Wait for user to open terminal
  Serial.println("SparkFun Ublox Example");

  Wire.begin();

  if (myGPS.begin() == false) //Connect to the Ublox module using Wire port
  {
    Serial.println(F("Ublox GPS not detected at default I2C address. Please check wiring. Freezi
ng."));
    while (1);
  }

  myGPS.setI2COutput(COM_TYPE_UBX); //Set the I2C port to output UBX only (turn off NMEA noise)

  if (myGPS.getEsfInfo()){

    Serial.print("Fusion Mode: ");
    Serial.println(myGPS.imuMeas.fusionMode);

    if (myGPS.imuMeas.fusionMode == 1){
      Serial.println("Fusion Mode is Initialized!");
        }
        else {
      Serial.println("Fusion Mode is either disabled or not initialized - Freezing!");
            Serial.println("Please see Example 1 description at top for more information.");
        }
  }
}
```

Next, the sketch grabs that IMU data which is stored in a **struct** called `imuMeas` .

```
void loop()
{

  if (myGPS.getEsfIns())
  {
    Serial.print("X: ");
    Serial.println(myGPS.imuMeas.xAngRate);
    Serial.print("Y: ");
    Serial.println(myGPS.imuMeas.yAngRate);
    Serial.print("Z: ");
    Serial.println(myGPS.imuMeas.zAngRate);
    Serial.print("X Acceleration: ");
    Serial.println(myGPS.imuMeas.xAccel);
    Serial.print("Y Acceleration: ");
    Serial.println(myGPS.imuMeas.yAccel);
    Serial.print("Z Acceleration: ");
    Serial.println(myGPS.imuMeas.zAccel);
        // These values also have "validity checks" that can be provided by the
        // ublox library, add "Vald" to values: e.g. xAngRateVald or xAccelVald.
  }

  delay(250);
}
```

If you have not already, select your Board (in this case the **Arduino Uno**), and associated COM port. Upload the code to the board and set the serial monitor to **115200 baud**. This may be a good time to bring a friend along to drive if you decide to actively monitor the output. Otherwise, check out the data after taking the board for a stroll. Try driving around as the board senses the car's movement. Then park in a safe location with the engine turned off before inspecting the data.

## Example 4 - Vehicle Dynamics

What happened to Example 3? It's been skipped over because its used primarily as a diagnostic sketch. What sensors are currently being used, are they functioning correctly, are the measurements being listed as bad or non-existent? Example 3 helps diagnose these various issues. Lets move ahead to the fourth example in the library (located in **File Examples > SparkFun Ublox Arduino Library** > **Dead Reckoning** > **Example4_vehicleDynamics**)

The **vehicle attitude** is a termed coined by u-blox that encompasses three measurements: **vehicle pitch**, **vehicle roll**, and **vehicle heading**. Much like the other example sketches, this one checks to make sure that the SparkFun Dead Reckoning NEO-M8U has been calibrated before pulling data.

```
if (myGPS.getEsfInfo()){

    Serial.print("Fusion Mode: ");
    Serial.println(myGPS.imuMeas.fusionMode);

    if (myGPS.imuMeas.fusionMode == 1){
      Serial.println("Fusion Mode is Initialized!");
        }
        else {
      Serial.println("Fusion Mode is either disabled or not initialized - Freezing!");
            Serial.println("Please see Example 1 description at top for more information.");
        }
  }
}
```

If the SparkFun Dead Reckoning NEO-M8U has indeed been calibrated, then it gets the relevant information by calling `myGPS.getVehAtt()` . As in Example 2, the data is stored within a *struct* called **vehAtt**.

```
void loop()
{
        myGPS.getVehAtt(); // Give the sensor you want to check on.
        Serial.print("Roll: ");
        Serial.println(myGPS.vehAtt.roll);
        Serial.print("Pitch: ");
        Serial.println(myGPS.vehAtt.pitch);
        Serial.print("Heading: ");
        Serial.println(myGPS.vehAtt.heading);
        Serial.print("Roll Accuracy: ");
        Serial.println(myGPS.vehAtt.accRoll);
        Serial.print("Pitch Accuracy: ");
        Serial.println(myGPS.vehAtt.accPitch);
        Serial.print("Heading Accuracy: ");
        Serial.println(myGPS.vehAtt.accHeading);

        delay(250);
}
```

If you have not already, select your Board (in this case the **Arduino Uno**), and associated COM port. Upload the code to the board and set the serial monitor to **115200 baud**. This may be a good time to bring a friend along to drive if you decide to actively monitor the output. Otherwise, check out the data after taking the board for a stroll. Try driving around as the board senses the car's movement. Then park in a safe location with the engine turned off before inspecting the data.

## Resources and Going Further

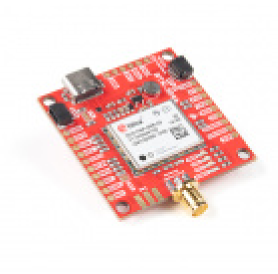
Now that you've successfully got your GPS receiver up and running, it's time to incorporate it into your own project! For more information, check out the resources below:

- **SparkFun u-Blox NEO-M8U with Chip Antenna**
  - Schematic (PDF)
  - Eagle Files (ZIP)
  - Board Dimensions (PNG)

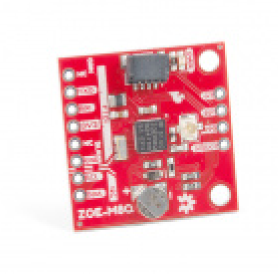
- **u-blox NEO-M8U Module Documentation and Resources**
    - NEO-M8U Datasheet (PDF)
    - Protocol Summary (PDF)
    - Integration Manual (PDF)
    - u-blox Protocol Specification (PDF)
    - UDR Whitepaper (PDF)
    - u-center Software
- **GitHub**
    - SparkFun u-blox Arduino Library
    - Product Repo
    - SFE Product Showcase

Are you looking for a GPS receivers? Check out the following GPS boards from the SparkFun catalog!

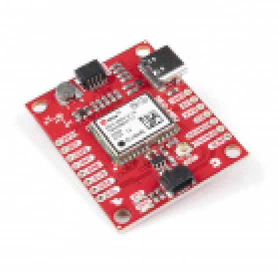
SparkFun GPS-RTK2 Board - ZED-F9P (Qwiic)
● GPS-15136



SparkFun GPS-RTK-SMA Breakout - ZED-F9P (Qwiic)
● GPS-16481



SparkFun GPS Breakout - ZOE-M8Q (Qwiic)
● GPS-15193



SparkFun GPS Dead Reckoning Breakout - NEO-M8U (Qwiic)
● GPS-16329

Need some inspiration for your next project? Check out some of these related tutorials:

### GPS Basics

The Global Positioning System (GPS) is an engineering marvel that we all have access to for a relatively low cost and no subscription fee. With the correct hardware and minimal effort, you can determine your position and time almost anywhere on the globe.

### GPS Logger Shield Hookup Guide

How to assemble and hookup the SparkFun GPS Logger Shield. Never lose track of your Arduino again!

### HX1 APRS Transmitter Hookup Guide

The HX1 is a low-power amateur radio transmitter that can be used to send data to the Automatic Packet Reporting System (APRS) network.

### GNSS Chip Antenna Hookup Guide

You've always wanted to experiment with those tiny GPS antennas. Now you can!