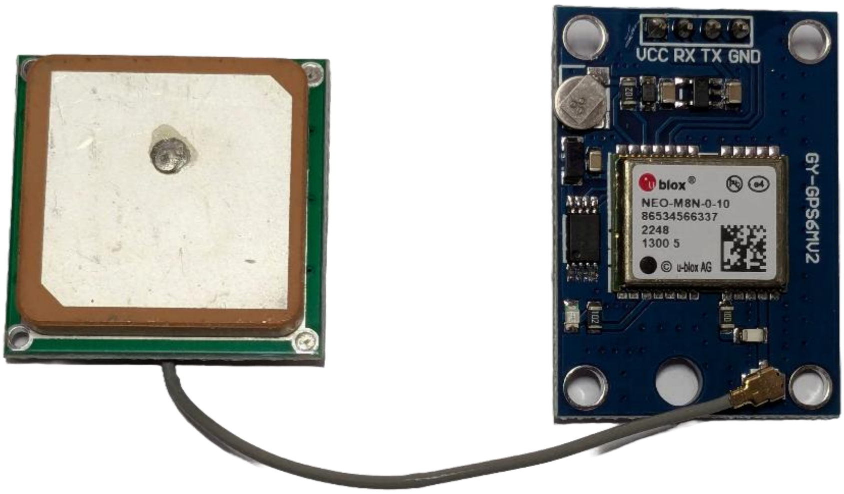
Interfacing GY-NEO-M8N GPS Module with Arduino



Introduction

The NEO-M8N is a GPS module that can be plugged into your Arduino board. It uses satellites to find your location and then sends that information to your Arduino. This lets you add location features to your projects, like a GPS tracker or a robot that navigates based on its position.

Overall, compared to the previous model NEO-6M, the NEO-M8N is a superior choice for applications that require higher accuracy, faster updates, lower power consumption, or support for multiple satellite constellations. Since NEO-M8N is backward compatible with NEO-6M, same breakout board or PCB layout might be used by manufacturers.

Key Advantages of the Neo-M8N GPS Module Over the Neo-6M:

- **Improved Accuracy:** The Neo-M8N offers a higher accuracy of 2 meters compared to the Neo-6M's 2.5 meters.
- **Faster Update Rate:** The Neo-M8N boasts a faster update rate of 10 Hz, allowing for more frequent position updates than the Neo-6M's 1 Hz.
- **Lower Power Consumption:** The Neo-M8N is designed to be more power-efficient, making it suitable for battery-powered applications.
- **Multiple Constellation Support:** The Neo-M8N can track GPS, GLONASS, Galileo, and BeiDou satellites, providing better coverage and performance in various locations.
- **Newer Technology:** As a more recent product, the Neo-M8N may benefit from advancements in GPS technology and offer additional features or capabilities.

Specifications

Model	Ublox NEO-M8N
Receiver Type	72-channel Ublox M8 engine
Input Supply Voltage (VDC)	3~5 V
Main Chip	Ublox NEO-M8N
Sensitivity (dBm)	156 dBm
Position Accuracy (Meter)	2 - 2.5m
Acceleration (g)	≤ 4 g
Navigation Update Rate	5 - 10 Hz
Operating Temperature Range (°C)	-45 to 105 °C
Tracking Sensitivity (dBm)	165 – 167 dBm
Co-Ordinate System	WGS-84
Capture Time	0.1s Average
Average Cold Start Time (s)	≤ 29 s
Warm / Hot Start Time (s)	1 s
Boot Time (s)	15 s
Maximum Altitude (m)	50,000 m
Dimensions (mm)	
Antenna	26 x 26 x 9* mm
Module	26 x 36 x 4 mm
Weight (gm):	17g

Objective

Interfacing the GPS Module with the Arduino *(or with any compatible microcontroller)* and getting the GPS data.

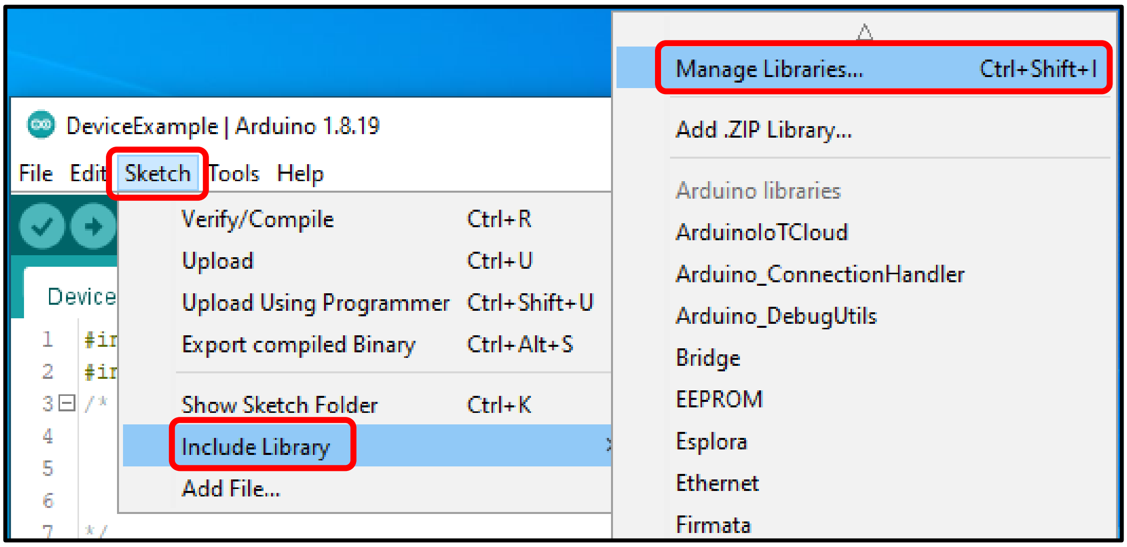
List of items needed

1. NEO-M8N GPS Module
2. Arduino UNO (any compatible MCUs)
3. Jumper Wires
4. Breadboard

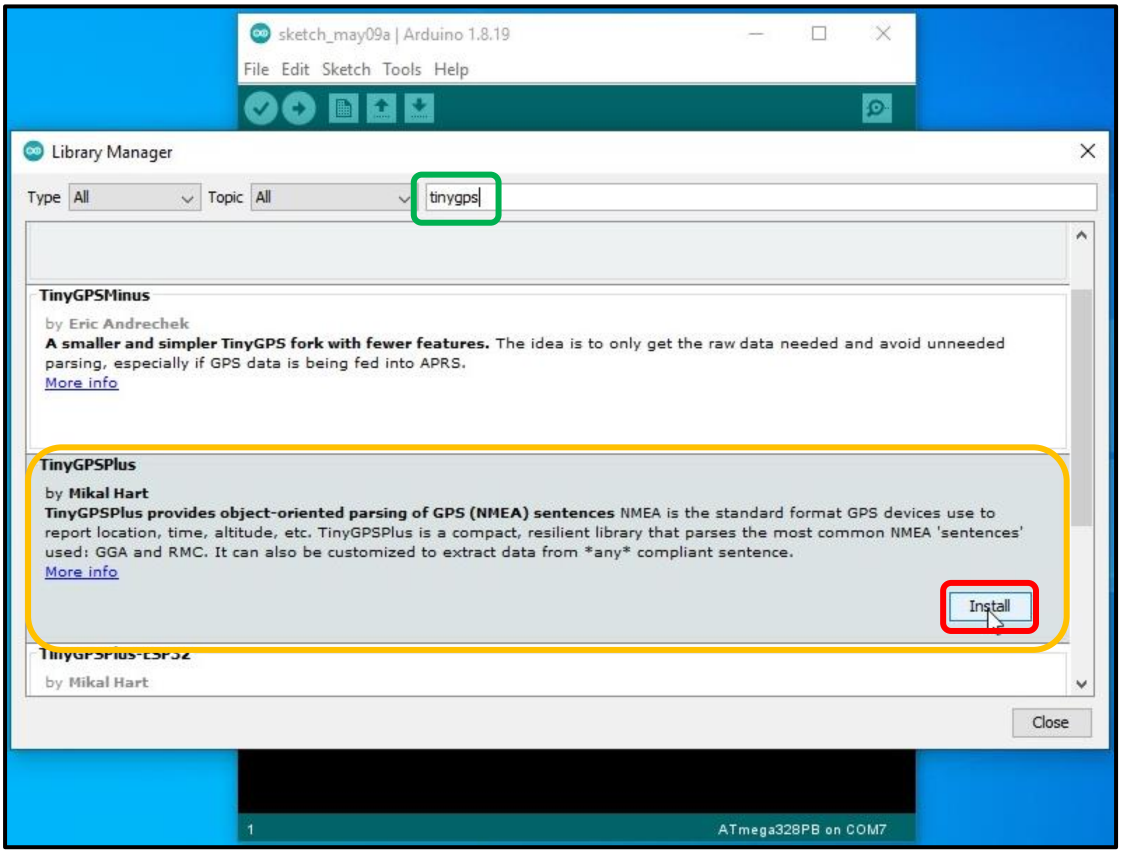
Procedures

A. Installing the Arduino library.

1. Launch **Arduino IDE**, navigate to the **toolbar** and select the **Sketch > Include Library > Manage Libraries…**.


DeviceExample | Arduino 1.8.19
File Edit Sketch Tools Help
Device
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
ArduinoIoTCloud
Arduino_ConnectionHandler
Arduino_DebugUtils
Bridge
EEPROM
Esplora
Ethernet
Firmata

2. Once **Library Manager** window appeared, search for **'TinyGPSPlus'** and click on **Install**.

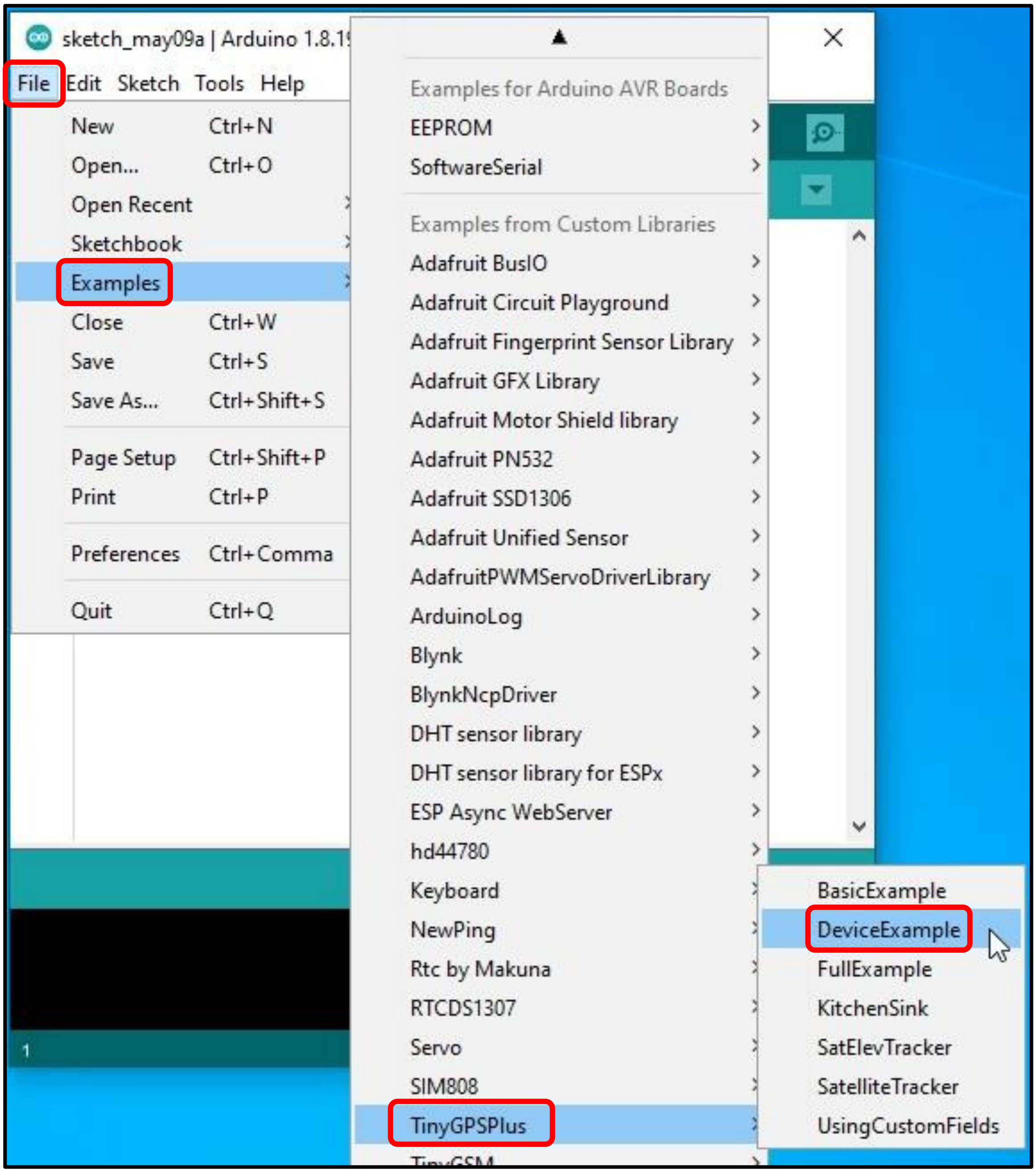


TinyGPSMinus
by **Eric Andrechek**
A smaller and simpler TinyGPS fork with fewer features. The idea is to only get the raw data needed and avoid unneeded parsing, especially if GPS data is being fed into APRS.
More info

TinyGPSPlus
by **Mikal Hart**
TinyGPSPlus provides object-oriented parsing of GPS (NMEA) sentences NMEA is the standard format GPS devices use to report location, time, altitude, etc. TinyGPSPlus is a compact, resilient library that parses the most common NMEA 'sentences' used: GGA and RMC. It can also be customized to extract data from *any* compliant sentence.
More info

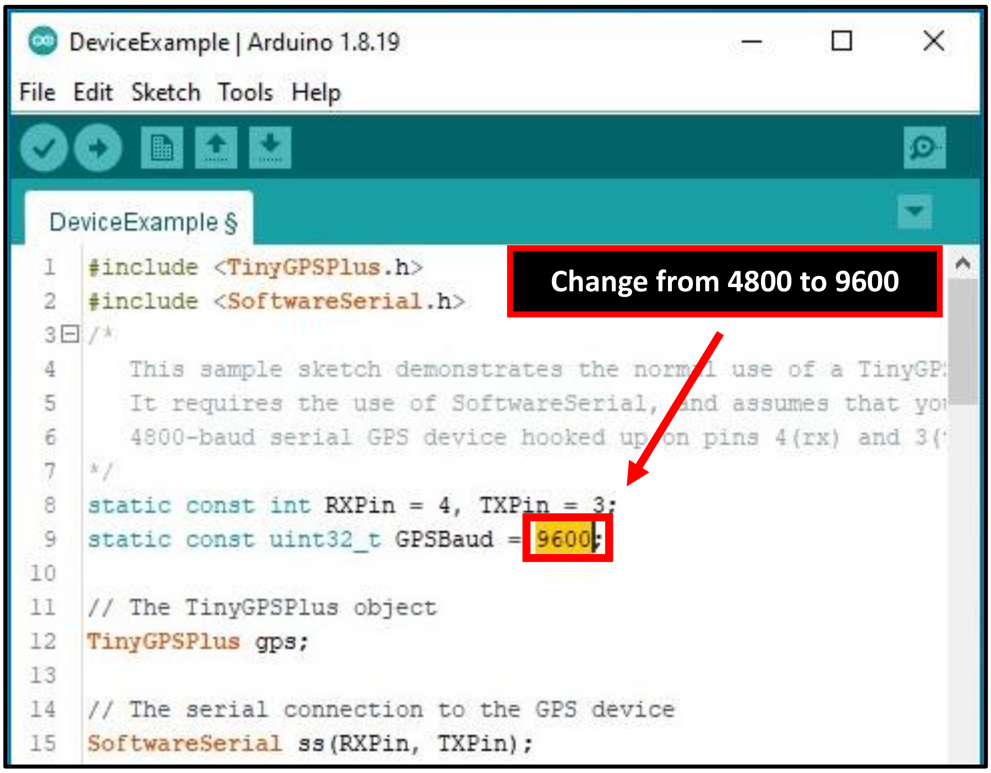
TinyGPSPlus-ESP32
by **Mikal Hart**

3. Once library downloaded & installed open the code sample code provided. From toolbar, **Examples > TinyGPSPlus > DeviceExample**.



4. The code will utilize the **Digital Pin 3 and 4** to receive and send data between both Arduino and the GPS Module.

5. **Change** the GPS baud rate from **4800** to **9600**. By default, the **NEO-M8N** GPS Module run at **9600** bps from factory.



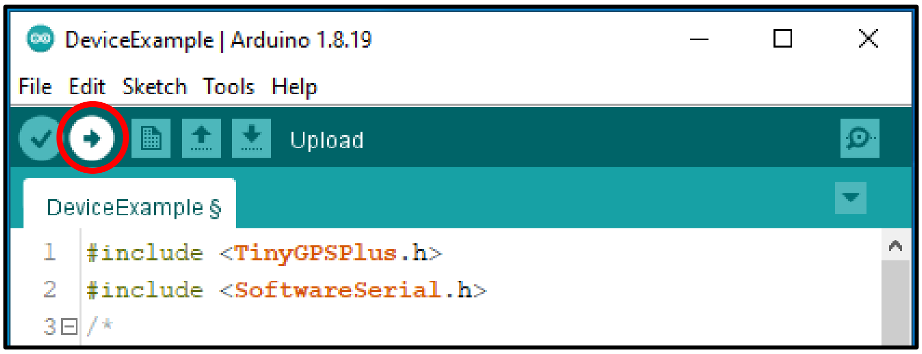
Change from 4800 to 9600

```
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>
/*
   This sample sketch demonstrates the normal use of a TinyGP
   It requires the use of SoftwareSerial, and assumes that yo
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(
*/
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPSPlus object
TinyGPSPlus gps;

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

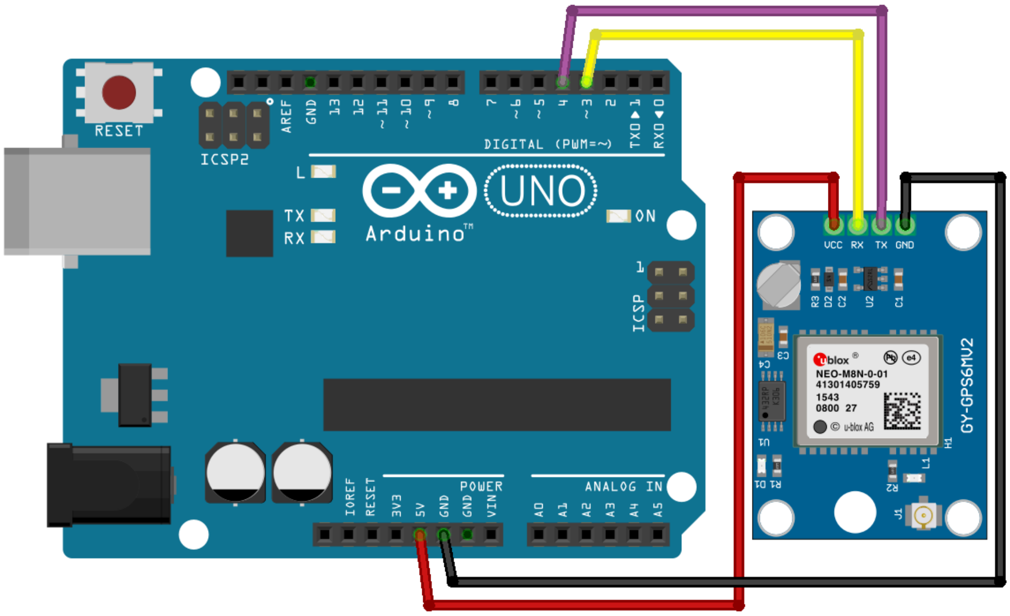
6. Once code correctly configured **Upload** the code to **Arduino UNO** board.



```
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>
/*
```

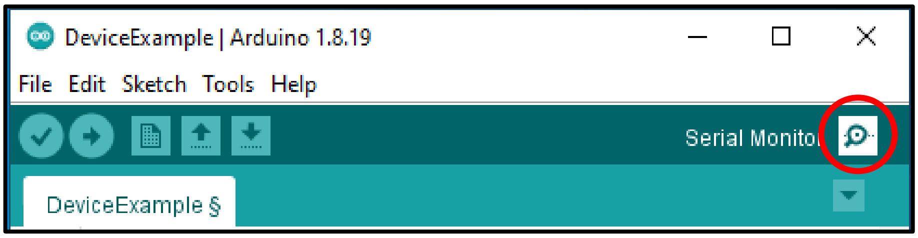
B. Wiring Setup.

Arduino UNO R3		GY-GPSV3-NEO-M8N
5V		VCC
GND		GND
D4 (as Receiver)	←	TX
D3 (as Transmitter)	→	RX



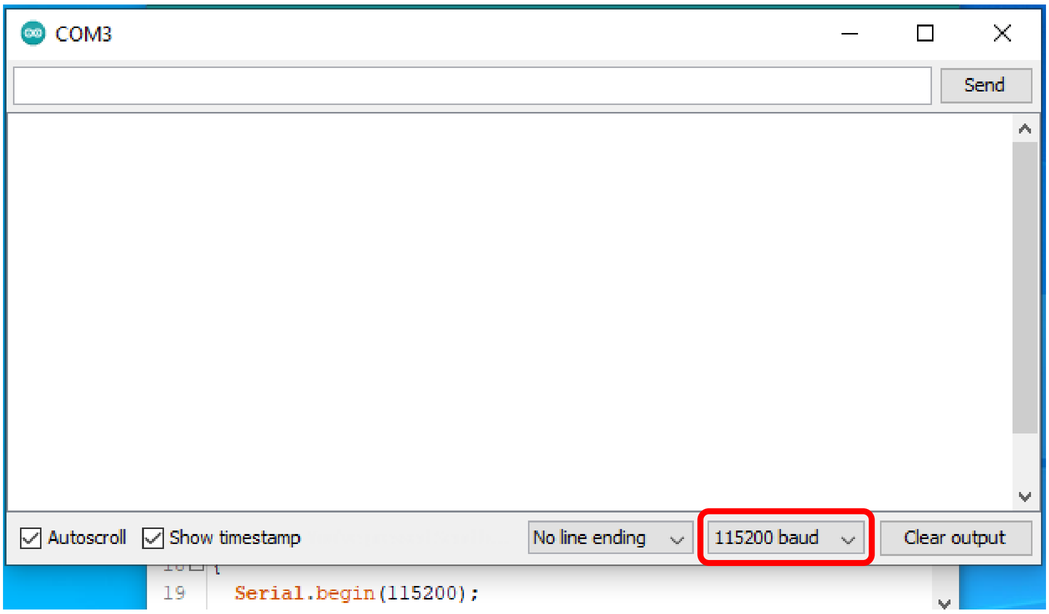
C. Check the Result.

1. After code uploaded & both boards *(Arduino UNO and GPS Module)* connected, open the **Serial Monitor** to check the acquired GPS data.



2. Make sure to set the **Serial Baud Rate** same as in the code which is **115200 baud**. (*Do not get confused with GPS Baud Rate this is for displaying data from Arduino*)

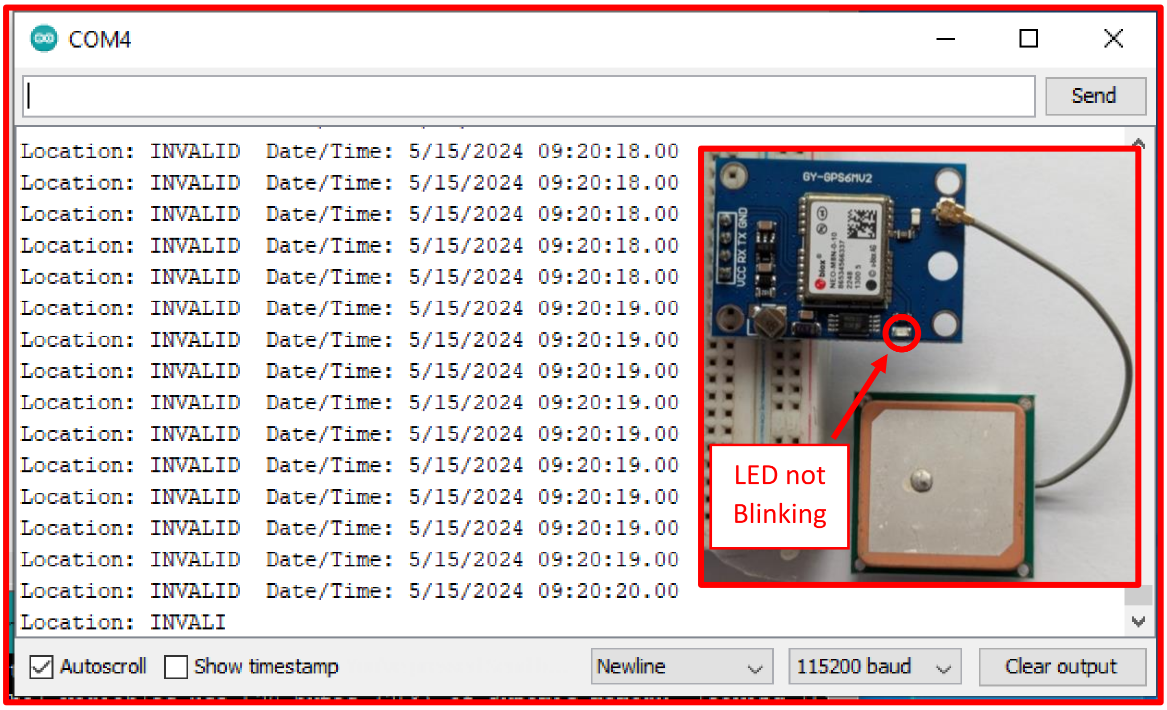
   (Arduino UNO -> **GPS NEO-M8N = 9600 bps**)
   (Arduino UNO -> **Serial Monitor = 115200 bps**)



☑ Autoscroll ☑ Show timestamp No line ending 115200 baud Clear output

```
Serial.begin(115200);
```

GPS Location does not fix

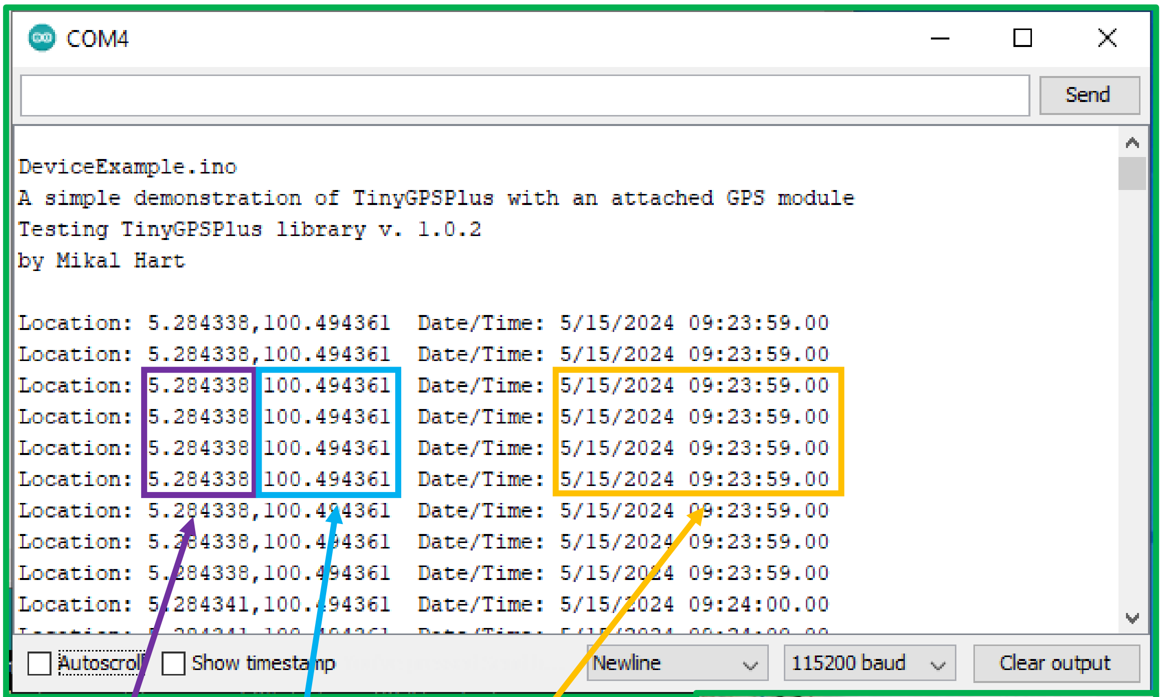


```
Location: INVALID  Date/Time: 5/15/2024 09:20:18.00
Location: INVALID  Date/Time: 5/15/2024 09:20:18.00
Location: INVALID  Date/Time: 5/15/2024 09:20:18.00
Location: INVALID  Date/Time: 5/15/2024 09:20:18.00
Location: INVALID  Date/Time: 5/15/2024 09:20:18.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:19.00
Location: INVALID  Date/Time: 5/15/2024 09:20:20.00
Location: INVALI
```

LED not Blinking

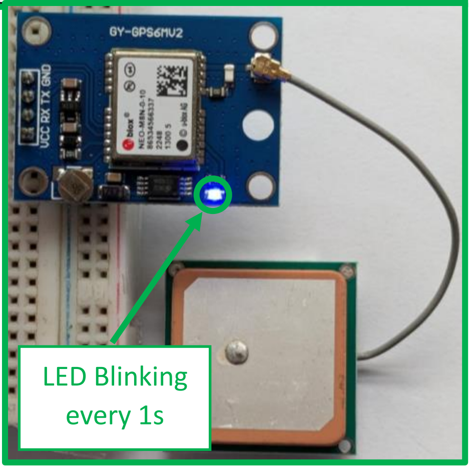
☑ Autoscroll □ Show timestamp Newline 115200 baud Clear output

GPS Location Fixed



```
DeviceExample.ino
A simple demonstration of TinyGPSPlus with an attached GPS module
Testing TinyGPSPlus library v. 1.0.2
by Mikal Hart

Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284338,100.494361  Date/Time: 5/15/2024 09:23:59.00
Location: 5.284341,100.494361  Date/Time: 5/15/2024 09:24:00.00
```



Troubleshooting *(What to do if module is not working properly?)*

1. **Check** the **wiring**, make sure the **Receiver, Rx** and **Transmitter, Tx** connection is correct.
2. Make sure the **Baud rate is correct**. **(9600 bps for NEO-M8N and 6M)**
3. **Check** the **Antenna U.FL connector**, make sure the connection is **solid**.
4. **Test** the GPS on the **open space area (outdoor)**. GPS Signal might be **weaker** when tested **Indoor**. (Alternatively, user can upgrade their antenna)
5. **Power** the Arduino or GPS Module with a Sufficient Power Adapter / Supply or Battery **(5v)**.
6. **Let the module powered by Arduino for a few minutes** if module isn't used for a long period of times. This module equipped with a small battery to act as capacitor to store configuration data. (**normally** that **battery discharged let it charge first**)
7. **Test and Debug** the Module **with** manufacturer provided program **U-Center.**