NEO 6M gps module

GPS receivers work by figuring out how far they are from a number of satellites. They are pre-programmed to know where the GPS satellites are at any given time.

The satellites transmit information about their position and the current time in the form of radio signals towards the Earth. These signals identify the satellites and tell the receiver where they are located.

The receiver then calculates how far away each satellite is by figuring out how long it took for the signals to arrive. Once it has information on how far away at least three satellites are and where they are in space, it can pinpoint your location on Earth.

This process is known as Trilateration.

NEO-6M GPS Chip

At the heart of the module is a NEO-6M GPS chip from u-blox.

It can track up to 22 satellites on 50 channels and achieves the industry’s highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current.

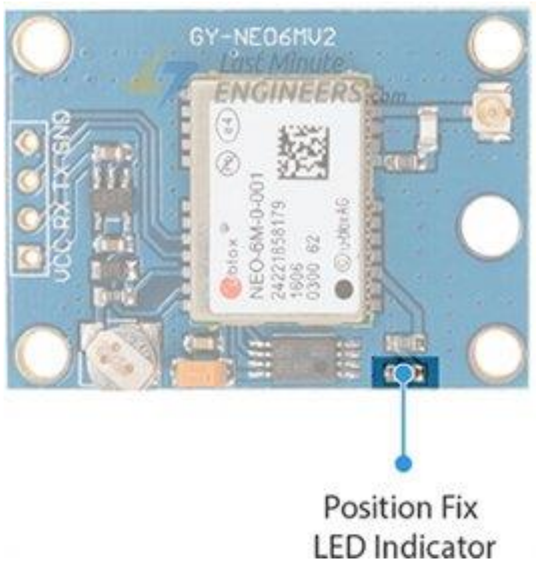
Unlike other GPS modules, it can do up to **5 location updates a second with 2.5m Horizontal position accuracy**. The u-blox 6 positioning engine also boasts a **Time-To-First-Fix (TTFF) of under 1 second**.

One of the best features the chip provides is **Power Save Mode(PSM)**. It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically **reduces** power consumption of the module **to just 11mA**.

Here are complete specifications:

Receiver Type	50 channels, GPS L1(1575.42Mhz)
Horizontal Position Accuracy	2.5m
Navigation Update Rate	1HZ (5Hz maximum)
Capture Time	Cool start: 27sHot start: 1s
Navigation Sensitivity	-161dBm
Communication Protocol	NMEA, UBX Binary, RTCM
Serial Baud Rate	4800-230400 (default 9600)
Operating Temperature	-40°C ~ 85°C
Operating Voltage	2.7V ~ 3.6V
Operating Current	45mA
TXD/RXD Impedance	510Ω

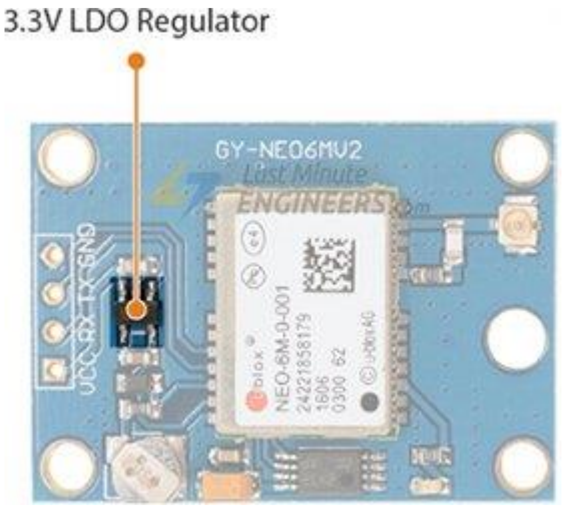
Position Fix LED Indicator



There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It'll blink at various rates depending on what state it's in:

- No Blinking – It's searching for satellites.
- Blink every 1s – Position Fix is found(The module can see enough satellites).

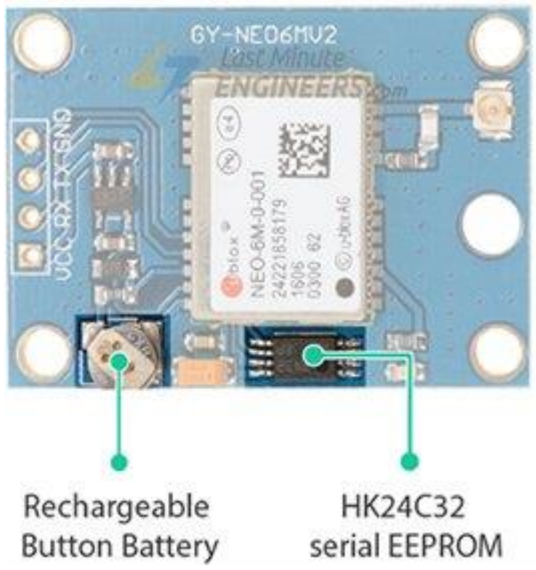
3.3V LDO Regulator



The operating voltage of the NEO-6M chip is from 2.7 to 3.6V. But the good news is that, the module comes with MIC5205 ultra-low dropout 3V3 regulator from MICREL.

The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.

Battery & EEPROM



The module is equipped with an HK24C32 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I2C.

The module also contains a rechargeable button battery which acts as a super-capacitor.

An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data(GNSS orbit data) and module configuration. But it's not meant for permanent data storage.

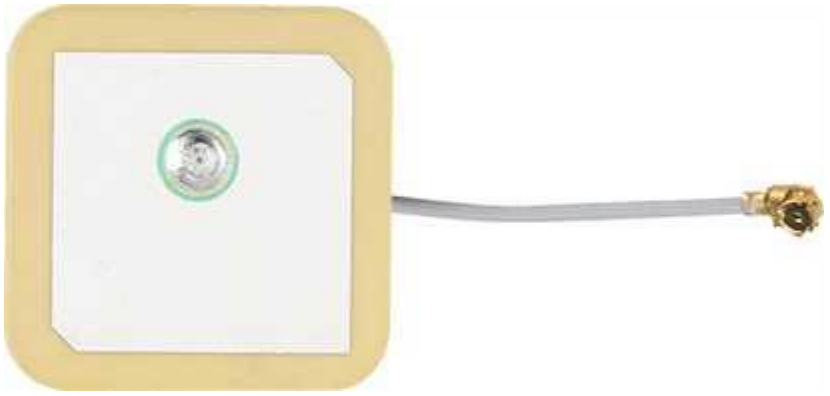
As the battery retains clock and last position, time to first fix (TTFF) significantly **reduces to 1s**. This allows much faster position locks.

Without the battery the GPS always cold-start so the initial GPS lock takes more time.

The battery is automatically charged when power is applied and maintains data for up to two weeks without power.

Antenna

An antenna is required to use the module for any kind of communication. So, the module comes with a patch antenna having **-161 dBm sensitivity**.

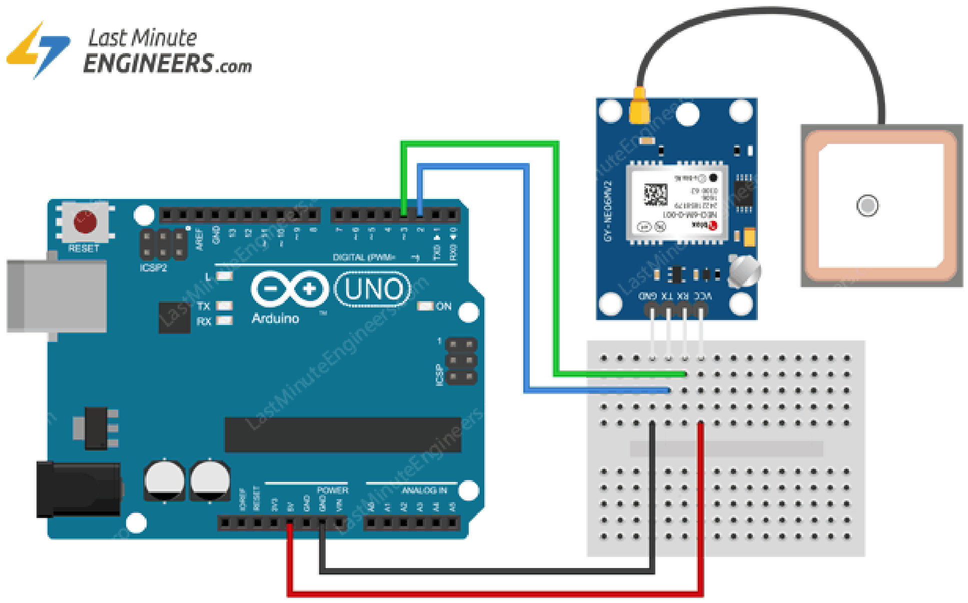


Wiring NEO-6M GPS module with Arduino UNO

We start by connecting the patch antenna to the U.FL connector. Threading the U.FL cable through one of the mounting holes will pave a way for a more robust connection.

Now, connect Tx and Rx pin on module to digital pin#? and #?(PWM pin) respectively on Arduino; as we'll be using software serial to talk to the module.

Next, connect VCC pin to the 5V pin on the arduino and GND to ground.



Arduino Code – Reading GPS Data

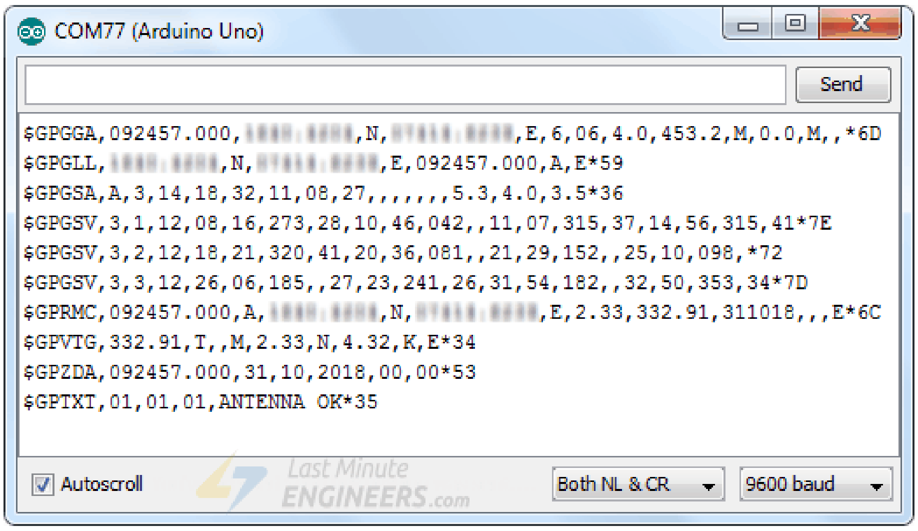
The best thing about any GPS receiver is that they start spitting out data as soon as you turn them ON.

The best way to test this data is to use Arduino as USB to TTL Converter. Following program does just that.

```
#include <SoftwareSerial.h>

// Choose two Arduino pins to use for software serial

int RXPin = 2;

int TXPin = 3;

//Default baud of NEO-6M is 9600

int GPSBaud = 9600;

// Create a software serial port called "gpsSerial"

SoftwareSerial gpsSerial(RXPin, TXPin);

void setup()

{
  // Start the Arduino hardware serial port at 9600 baud
  Serial.begin(9600);
  // Start the software serial port at the GPS's default baud
  gpsSerial.begin(GPSBaud);
}
void loop()
{
  // Displays information when new sentence is available.
  while (gpsSerial.available() > 0)
    Serial.write(gpsSerial.read());
}
```

Ths above program is used to just read the data from the module.

Upload the program and open up the serial monitor from the Arduino IDE. Remember to select 9600 baud. You should see text like the following:



```
$GPGGA,092457.000,[redacted],N,[redacted],E,6,06,4.0,453.2,M,0.0,M,,*6D
$GPGLL,[redacted],N,[redacted],E,092457.000,A,E*59
$GPGSA,A,3,14,18,32,11,08,27,,,,,,,5.3,4.0,3.5*36
$GPGSV,3,1,12,08,16,273,28,10,46,042,,11,07,315,37,14,56,315,41*7E
$GPGSV,3,2,12,18,21,320,41,20,36,081,,21,29,152,,25,10,098,*72
$GPGSV,3,3,12,26,06,185,,27,23,241,26,31,54,182,,32,50,353,34*7D
$GPRMC,092457.000,A,[redacted],N,[redacted],E,2.33,332.91,311018,,,E*6C
$GPVTG,332.91,T,,M,2.33,N,4.32,K,E*34
$GPZDA,092457.000,31,10,2018,00,00*53
$GPTXT,01,01,01,ANTENNA OK*35
```

Autoscroll: ✓ · Both NL & CR · 9600 baud

The data you are getting over a serial interface are actually NMEA sentences.

NMEA is an acronym for the *National Marine Electronics Association*. This is a standard message format for Nearly all GPS receivers.

The NMEA standard is formatted in lines of data called sentences. Each sentence is comma separated to make it easier to parse by computers and microcontrollers.

These NMEA sentences are sent out at an interval called the update rate.

NEO-6M GPS module updates this information once per second(1Hz frequency) by default. But you can configure it for up to 5 updates per second(5Hz frequency).

Parsing NMEA Sentences

There are many sentences in the NMEA standard, the most common ones are:

- $GPRMC (Global Positioning Recommended Minimum Coordinates) provides the time, date, latitude, longitude, altitude and estimated velocity.
- $GPGGA sentence provides essential fix data which provide 3D location and accuracy data.

Let’s take an example of $GPRMC NMEA sentence from a GPS receiver.

$GPRMC, 123519, A, 4807.038, N, 01131.000, E,022.4, 084.4, 230394, 003.1, W*6A

$	Every NMEA sentence starts with $ character.
GPRMC	Global Positioning Recommended Minimum Coordinates
123519	Current time in UTC – 12:35:19
A	Status A=active or V=Void.
4807.038,N	Latitude 48 deg 07.038’ N
01131.000,E	Longitude 11 deg 31.000’ E
022.4	Speed over the ground in knots
084.4	Track angle in degrees True

220318	Current Date – 22rd of March 2018
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

Let’s take an example of $GPGGA NMEA sentence.

```
$GPGGA, 123519, 4807.038, N, 01131.000, E, 1, 08, 0.9, 545.4, M, 46.9, M, , *47
```

$	Starting of NMEA sentence.
GPGGA	Global Positioning System Fix Data
123519	Current time in UTC – 12:35:19
4807.038,N	Latitude 48 deg 07.038’ N
01131.000,E	Longitude 11 deg 31.000’ E
1	GPS fix
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude in Meters (above mean sea level)
46.9,M	Height of geoid (mean sea level)
(empty field)	Time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	The checksum data, always begins with *

Sounds tedious? TinyGPS++ library to the rescue..

Often for our projects to understand the information we have received, we need to parse NMEA sentences into useful information. To simplify our work, we have a library called TinyGPS++ library.

This library does a lot of heavy lifting required for receiving data from GPS modules, such as reading and extracting useful data in the background. So, we don't need to worry about icky parsing work.

The following test sketch will print the location information(Latitude, Longitude & Altitude) and UTC(Date & Time) on the serial monitor.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
// Choose two Arduino pins to use for software serial
int RXPin = 2;
int TXPin = 3;
int GPSBaud = 9600;
// Create a TinyGPS++ object
TinyGPSPlus gps;
// Create a software serial port called "gpsSerial"
SoftwareSerial gpsSerial(RXPin, TXPin);
void setup()
{
  // Start the Arduino hardware serial port at 9600 baud
  Serial.begin(9600);
  // Start the software serial port at the GPS's default baud
  gpsSerial.begin(GPSBaud);
}
```

```
void loop()
{
 // This sketch displays information every time a new sentence is correctly encoded.
 while (gpsSerial.available() > 0)
  if (gps.encode(gpsSerial.read()))
   displayInfo();
 // If 5000 milliseconds pass and there are no characters coming in
 // over the software serial port, show a "No GPS detected" error
 if (millis() > 5000 && gps.charsProcessed() < 10)
 {
  Serial.println("No GPS detected");
  while(true);
 }
}
void displayInfo()
{
 if (gps.location.isValid())
 {
  Serial.print("Latitude: ");
  Serial.println(gps.location.lat(), 6);
  Serial.print("Longitude: ");
  Serial.println(gps.location.lng(), 6);
  Serial.print("Altitude: ");
  Serial.println(gps.altitude.meters());
 }
 else
```

```
{
  Serial.println("Location: Not Available");
}
Serial.print("Date: ");
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print("/");
  Serial.print(gps.date.day());
  Serial.print("/");
  Serial.println(gps.date.year());
}
else
{
  Serial.println("Not Available");
}
Serial.print("Time: ");
if (gps.time.isValid())
{
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(":");
  if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(":");
  if (gps.time.second() < 10) Serial.print(F("0"));
```

```
    Serial.print(gps.time.second());
    Serial.print(".");
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.println(gps.time.centisecond());
  }
  else
  {
    Serial.println("Not Available");
  }
  Serial.println();
  Serial.println();
  delay(1000);
}
```