SANJIL K C

20BCE1855

**Interface NEO-6M GPS Module with Arduino For Checking Our Module**

In this tutorial, we will learn about NEO-6M GPS module and how to interface it with Arduino to obtain GPS parameters such as latitude, longitude, altitude date, time, speed, satellites, etc. We will learn how GPS works and the overview of the NEO-6M GPS module with an introduction, pinout, and specifications. After that, we will learn to interface a NEO-6M GPS Module module with Arduino.

For demonstration purposes, we will see two Arduino example sketches. In the first sketch, we will simply read the GPS data obtained from NEO-6M GPS. This will be in the form of NMEA sentences that are not easy to understand. To make the GPS data readable, we will use the TinyGPS++ library in the second sketch. In second example sketch, we will show GPS location coordinates on the serial monitor of the Arduino IDE. The location will be displayed in the form of longitude and latitude. Additionally, the location’s altitude and date/time will also be displayed.

## GPS Introduction

The Global Positioning System (GPS) is a satellite-based navigation system that consists of 24 orbiting satellites, each of which makes two circuits around the Earth every 24 hours. These satellites transmit three bits of information – the satellite’s number, its position in space, and the time the information is sent. These signals are picked up by the GPS receiver, which uses this information to calculate the distance between it and the GPS satellites. With signals from three or more satellites, a GPS receiver can triangulate its location on the ground (i.e., longitude and latitude) from the known position of the satellites. With four or more satellites, a GPS receiver can determine a 3D position (i.e., latitude, longitude, and elevation).

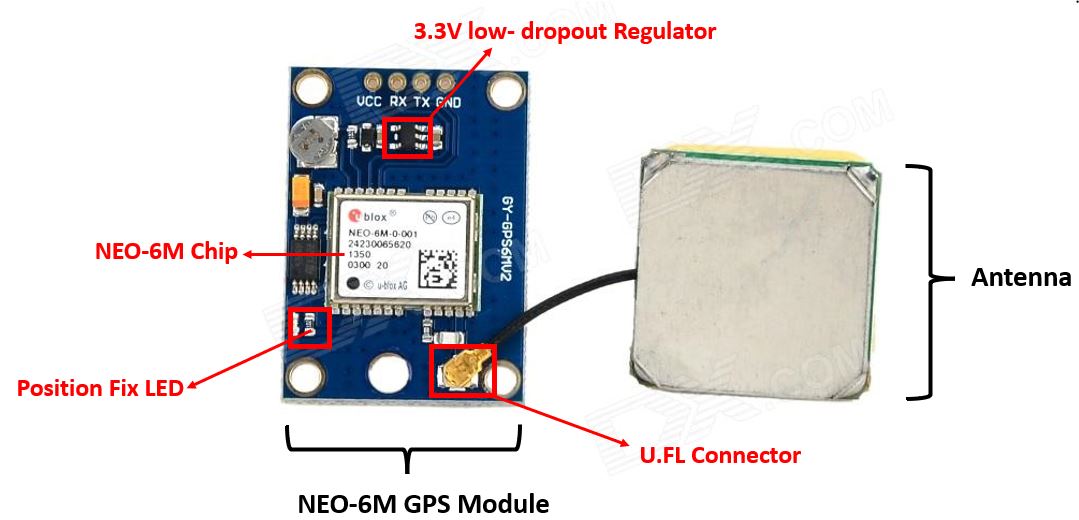
In addition, a GPS receiver can provide data on your speed and direction of travel. Anyone with a GPS receiver can access the system. Because GPS provides real-time, three-dimensional positioning, navigation, and timing 24 hours a day, 7 days a week, all over the world, it is used in numerous applications, including GIS data collection, surveying, and mapping.

## NEO-6M GPS Module Introduction

The NEO-6M GPS module is a GPS receiver that can locate all locations on Earth as it is able to track approximately 22 satellites. It consists of a high-performance u-blox 6 positioning engine. Measuring 16 x 12.2 x 2.4 mm, its compact architecture along with its low power consumption makes it a good choice for IoT projects. Overall it is a good cost-effective GPS receiver.

### Hardware Overview

Let us learn a little bit about its hardware. To obtain GPS readings, we have to use the NEO-6M GPS module with an antenna. The antenna is firmly attached to the module via the U.FL connector. This connector is found on the GPS module.



#### NEO-6M GPS Chip

In the middle of the GPS module, you can find the NEO-6M chip. This is responsible for tracking up to 22 satellites and any location on the Earth on several channels. Due to its highly sensitive tracking nature, it makes the NEO-6M module a popular GPS tracker.

Some key features of NEO-6M chip include:

* High sensitivity for tracking
* Low supply current (~45mA)
* Is able to track 5 locations per second with an accuracy of 2.5m (horizontal).
* Comes equipped with PSM also known as Power Saving Mode. This mode causes very less power consumption by turning the module ON/OFF according to the need.
* Great use as GPS trackers in smart watches due to very low power consumption (~11mA)

#### Position Fix LED Indicator

Moving ahead, the module comes with a position fix LED indicator. This LED indicates through its blinking effect whether the module is searching for satellites or has already found them. If the LED blinks after every second, then it indicates that the position fix is found. However, if the LED does not blink then the module is still searching for the satellites.

#### 3.3V low-dropout Regulator

The module also comes equipped with a 3.3V LDO regulator (MIC5205). This provides an efficient linear voltage regulation with ultralow-noise output and very low dropout voltage. Additionally, the module is can also tolerate 5V easily so programming it with Arduino is very convenient.

### Specifications

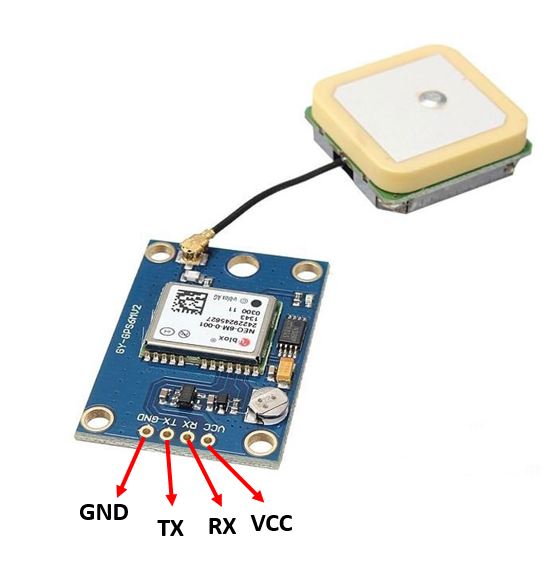
|  |  |
| --- | --- |
| Type | GPS |
| Supply | 2.7 V-3.6 V |
| Operating Current | 45mA |
| Operating Temperature | -40°C ~ 85°C |
| Horizontal Position Accuracy | 2.5m |
| Communication Protocol | NMEA, UBX Binary, RTCM |
| Features | RTC Crystal and External Interrupt/Wake up |
| Interface | UART, SPI, USB and DDC |

The table below shows some specifications of the NEO-6M module.

### Pinout of NEO 6M Module

The diagram below shows the pinout of the NEO 6M module. It consists of 4 pins named GND, TX, RX, and VCC.

|  |  |
| --- | --- |
| GND | This is the ground pin that will be connected with the ground of the Arduino UNO. |
| TX | This is the transmission pin used for serial communication. |
| RX | This is the receiver pin used for serial communication. |
| VCC | This is the VCC pin used to power up the GPS module. Connect it with the 5V of the Arduino UNO board. |



## **Wiring a NEO-6M GPS Module to an Arduino**

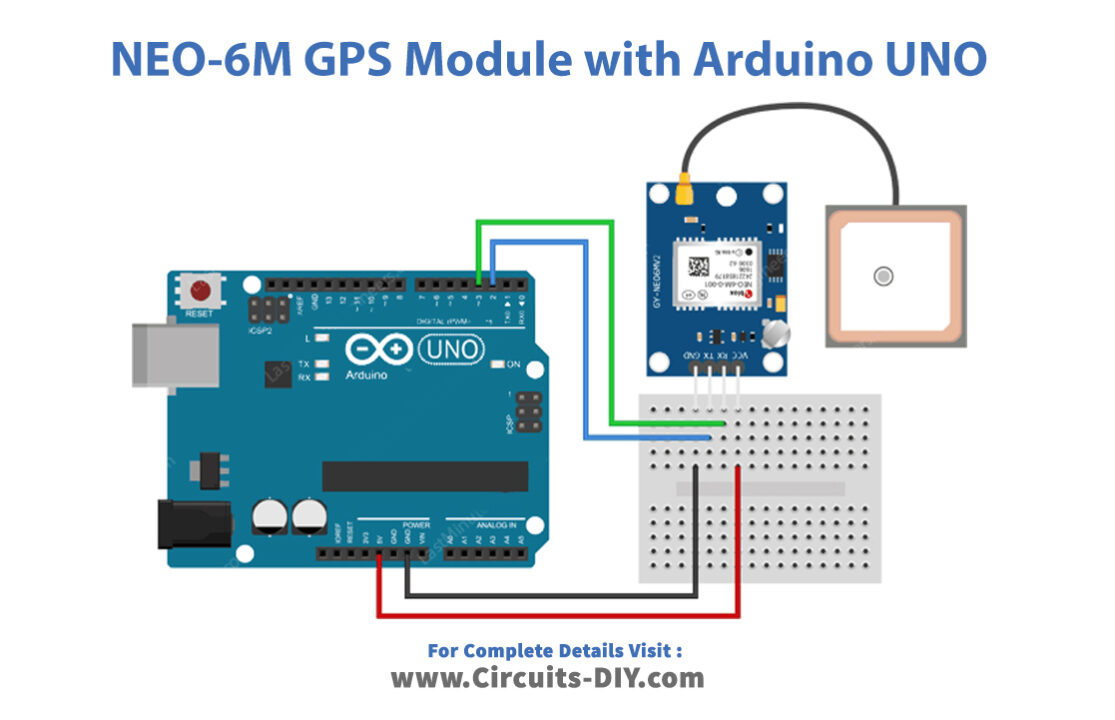
Now that we know everything about the module, we can start connecting it to our Arduino.

Begin by connecting the patch antenna to the U.FL connector. You can thread the U.FL cable through one of the mounting holes.

The module usually comes with unsoldered header pins. So you will need to solder them first.

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

Finally connect the Tx and Rx pins on the module to digital pins #2 and #3 respectively.



Wiring NEO-6M GPS Module to Arduino UNO

Code:

// TARP PROJECT PET SITTER

#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");

  }

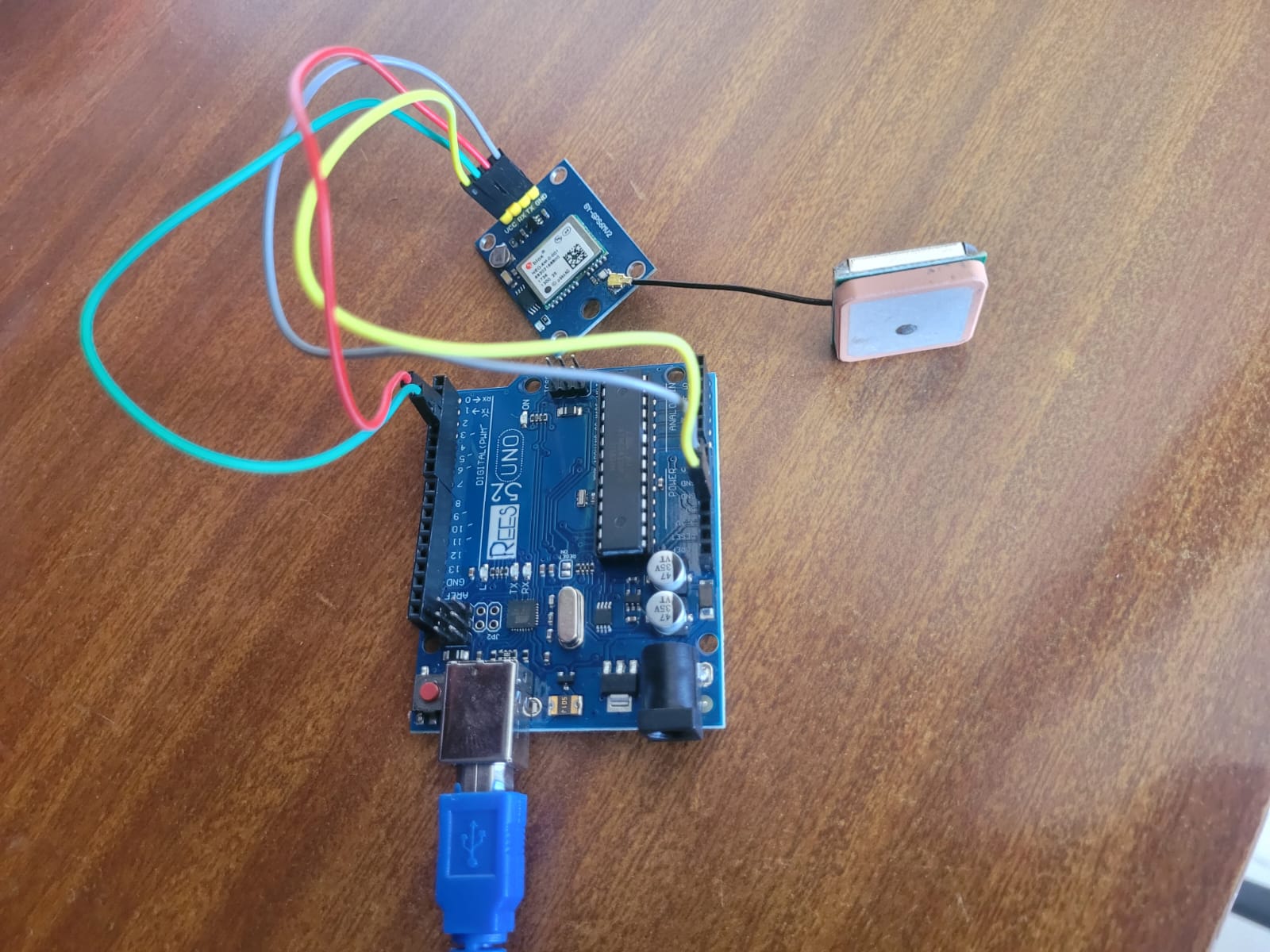
  Serial.println();

  Serial.println();

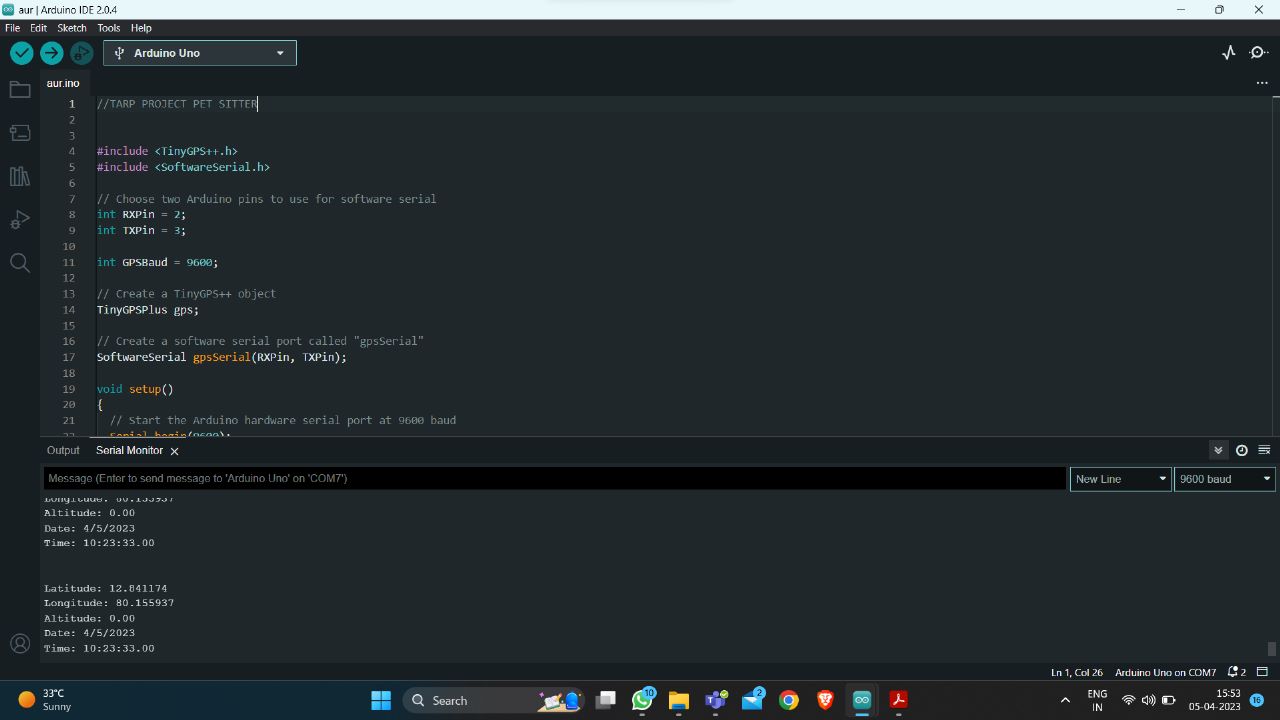
  delay(1000);

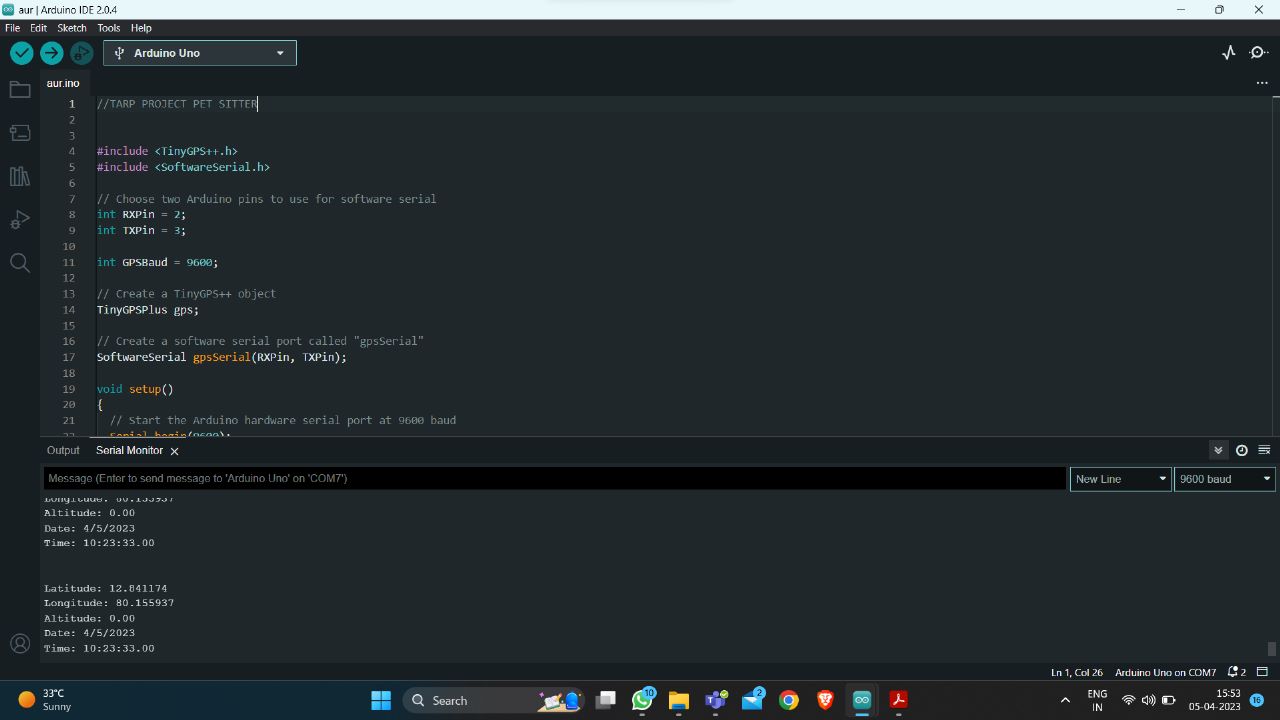
}

Hardware Photo



Output





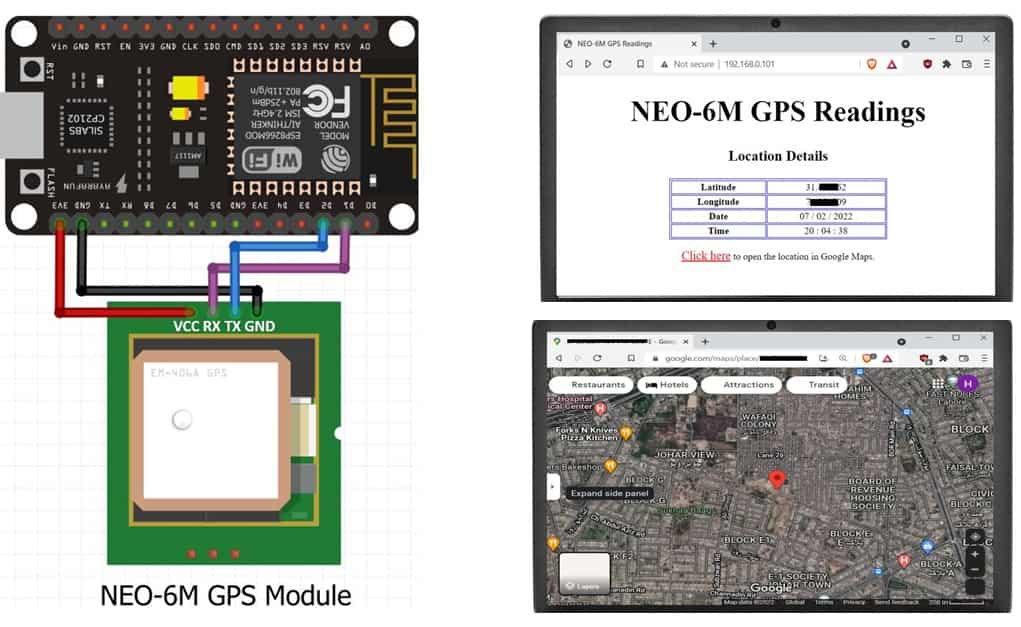
GPS interfacing with nodeMCU to track location for PET SITTER

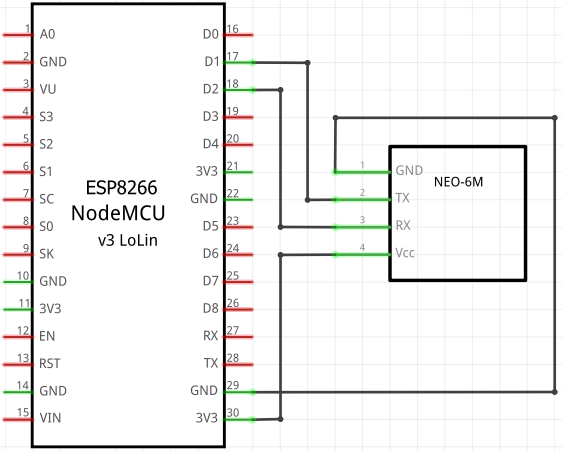
NodeMCU is ESP8266 based development board. It features ESP-12E as its processing core. It is a 32bit MCU. It has 14 GPIO pins, single channel 10 bit integrated ADC. It supports UART, I2C, SPI communication. It is 3.3V compatible, it cannot handle 5V. .

**The connections between NodeMCU and GPS module**

|  |  |
| --- | --- |
| **NodeMCU** | **GPS module** |
| 3V3 | VCC |
| GND | GND |
| D1 (GPIO5) | RX |
| D2 (GPIO4) | TX |

Below is **the circuit Diagram of connecting GPS with NodeMCU:**





GPS module takes some time to capture location details once it is powered on. NodeMCU starts webserver and waits for a client to get connected to the webserver. Once client is connected to the webserver, NodeMCU sends location details to connected client. The location details are displayed in a simple webpage designed using HTML.

**Steps:**

1 Connect the circuit as shown in the schematic.

2 Upload the code after changing ssid and password.

3 Open serial monitor in Arduino IDE and note down IP address of the webserver.

4 Open any Browser and enter the IP address of the webserver.

5 It will display Location details, date, time and Google maps link.

**Code :**

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

TinyGPSPlus gps;  // The TinyGPS++ object

SoftwareSerial ss(4, 5); // The serial connection to the GPS device

const char\* ssid = "sanjil";

const char\* password = "sanjil12345";

float latitude , longitude;

int year , month , date, hour , minute , second;

String date\_str , time\_str , lat\_str , lng\_str;

int pm;

WiFiServer server(80);

void setup()

{

  Serial.begin(115200);

  ss.begin(9600);

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED)

  {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  server.begin();

  Serial.println("Server started");

  // Print the IP address

  Serial.println(WiFi.localIP());

}

void loop()

{

  while (ss.available() > 0)

    if (gps.encode(ss.read()))

    {

      if (gps.location.isValid())

      {

        latitude = gps.location.lat();

        lat\_str = String(latitude , 6);

        longitude = gps.location.lng();

        lng\_str = String(longitude , 6);

      }

      if (gps.date.isValid())

      {

        date\_str = "";

        date = gps.date.day();

        month = gps.date.month();

        year = gps.date.year();

        if (date < 10)

          date\_str = '0';

        date\_str += String(date);

        date\_str += " / ";

        if (month < 10)

          date\_str += '0';

        date\_str += String(month);

        date\_str += " / ";

        if (year < 10)

          date\_str += '0';

        date\_str += String(year);

      }

      if (gps.time.isValid())

      {

        time\_str = "";

        hour = gps.time.hour();

        minute = gps.time.minute();

        second = gps.time.second();

        minute = (minute + 30);

        if (minute > 59)

        {

          minute = minute - 60;

          hour = hour + 1;

        }

        hour = (hour + 5) ;

        if (hour > 23)

          hour = hour - 24;

        if (hour >= 12)

          pm = 1;

        else

          pm = 0;

        hour = hour % 12;

        if (hour < 10)

          time\_str = '0';

        time\_str += String(hour);

        time\_str += " : ";

        if (minute < 10)

          time\_str += '0';

        time\_str += String(minute);

        time\_str += " : ";

        if (second < 10)

          time\_str += '0';

        time\_str += String(second);

        if (pm == 1)

          time\_str += " PM ";

        else

          time\_str += " AM ";

      }

    }

  // Check if a client has connected

  WiFiClient client = server.available();

  if (!client)

  {

    return;

  }

  // Prepare the response

  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html> <html> <head> <title>TARP PROJECT PET SITTER</title> <style>";

  s += "a:link {background-color: YELLOW;text-decoration: none;}";

  s += "table, th, td {border: 1px solid black;} </style> </head> <body> <h1  style=";

  s += "font-size:300%;";

  s += " ALIGN=CENTER>TARP PROJECT PET SITTER</h1>";

  s += "<p ALIGN=CENTER style=""font-size:150%;""";

  s += "> <b>Location Details</b></p> <table ALIGN=CENTER style=";

  s += "width:50%";

  s += "> <tr> <th>Latitude</th>";

  s += "<td ALIGN=CENTER >";

  s += lat\_str;

  s += "</td> </tr> <tr> <th>Longitude</th> <td ALIGN=CENTER >";

  s += lng\_str;

  s += "</td> </tr> <tr>  <th>Date</th> <td ALIGN=CENTER >";

  s += date\_str;

  s += "</td></tr> <tr> <th>Time</th> <td ALIGN=CENTER >";

  s += time\_str;

  s += "</td>  </tr> </table> ";

  if (gps.location.isValid())

  {

     s += "<p align=center><a style=""color:RED;font-size:125%;"" href=""http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=";

    s += lat\_str;

    s += "+";

    s += lng\_str;

    s += """ target=""\_top"">Click here!</a> To check the location in Google maps.</p>";

  }

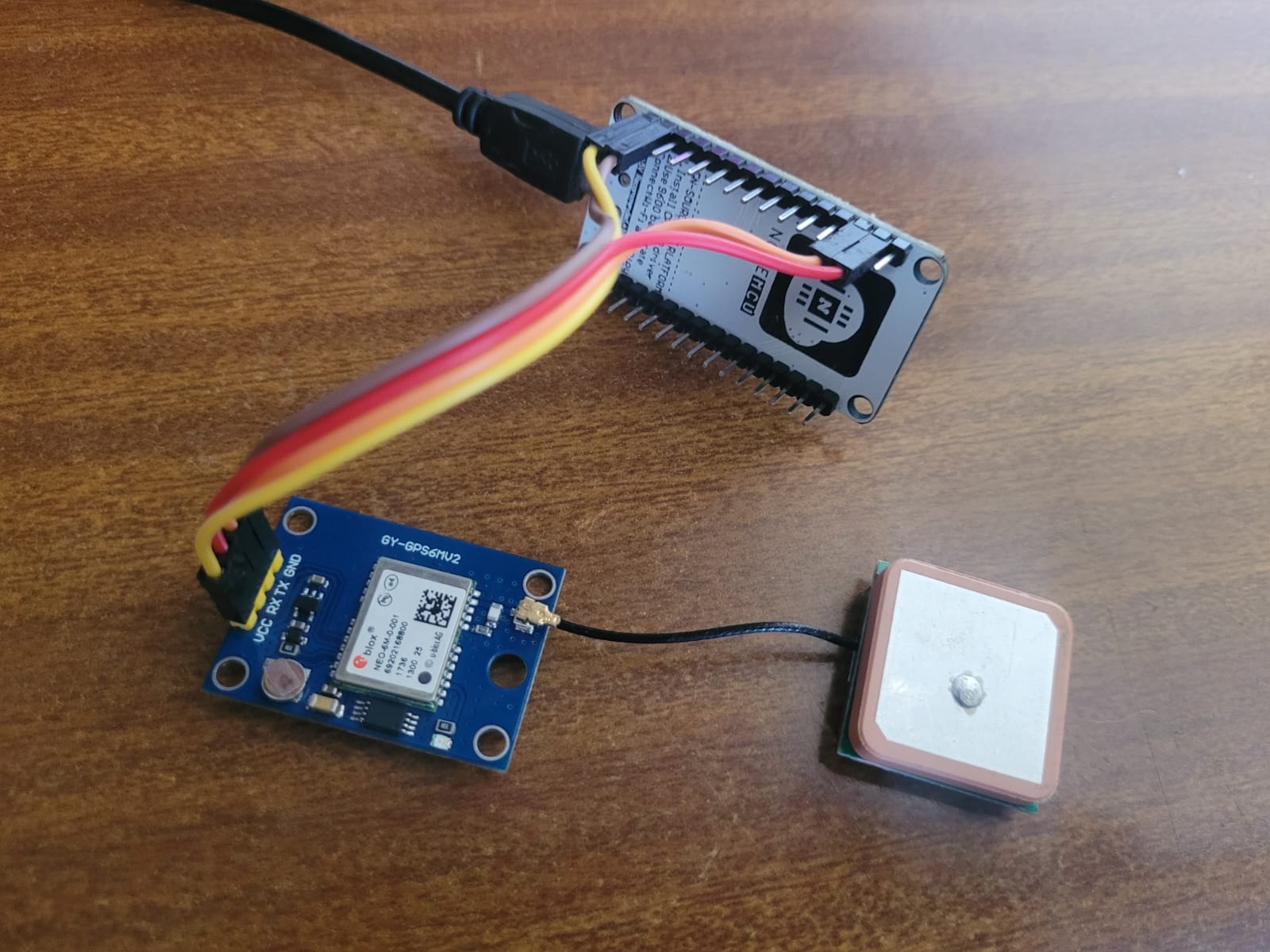
  s += "</body> </html> \n";

  client.print(s);

  delay(100);

}

Hardware photo





Output

