Sensing for Science

Level 5

This material is based upon work supported in part by the National Science Foundation EPSCoR Cooperative Agreement OIA-1757351. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

GPS modules

GPS modules are devices available to us that includes a GPS antenna and receiver. This device is able to receive data from the GPS satellites and calculate the current location of the module. The GPS module is then send us NEMA data to our Arduino board.

We will be using the Beitina Bk-180 GPS. This is a small yet accurate module. It is very easy to hook up and run on our Arduino boards.

The hook up for our GPS module is very simple. The VCC connects to 5v on the Arduino. TX connects to pin 6 on the Arduino. RX will go to pin 5 on the Arduino. GND will go to GND on the Arduino.

Once the circuit is built, it is time to test it. The code we are using is simple and will use a new prebuilt library called SoftwareSerial.

This library allows us to create new serial ports using pins. Since we will need the serial port to read the data, being able to create another port to read the GPS is essential. We will also use the string function that will collect many characters of data.

```
#include <SoftwareSerial.h>

static const int RXPin = 6, TXPin = 5;
static const uint32_t GPSBaud = 9600;
SoftwareSerial ss(RXPin, TXPin);

void setup() {

ss.begin(GPSBaud);

Serial.begin(115200);

}

void loop() {

  String msg = ss.readStringUntil('\r');
  Serial.print(msg);
}
```

Including the SoftwareSerial library

Declaring some variables for the GPS. Most GPS units use a Baud rate of 9600. If you get a different module you might have to change this.

Here we are creating an object which is our new serial port for the GPS.

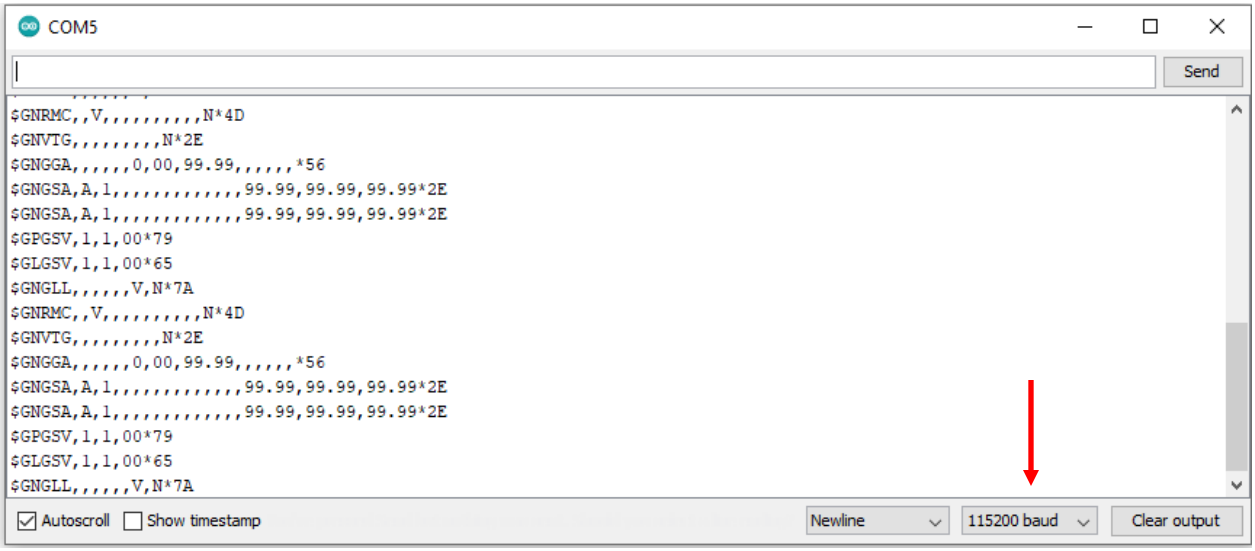
Starts the GPS unit

Start serial monitor so we can see our data. We are going to run it at 115200 to transfer all the data

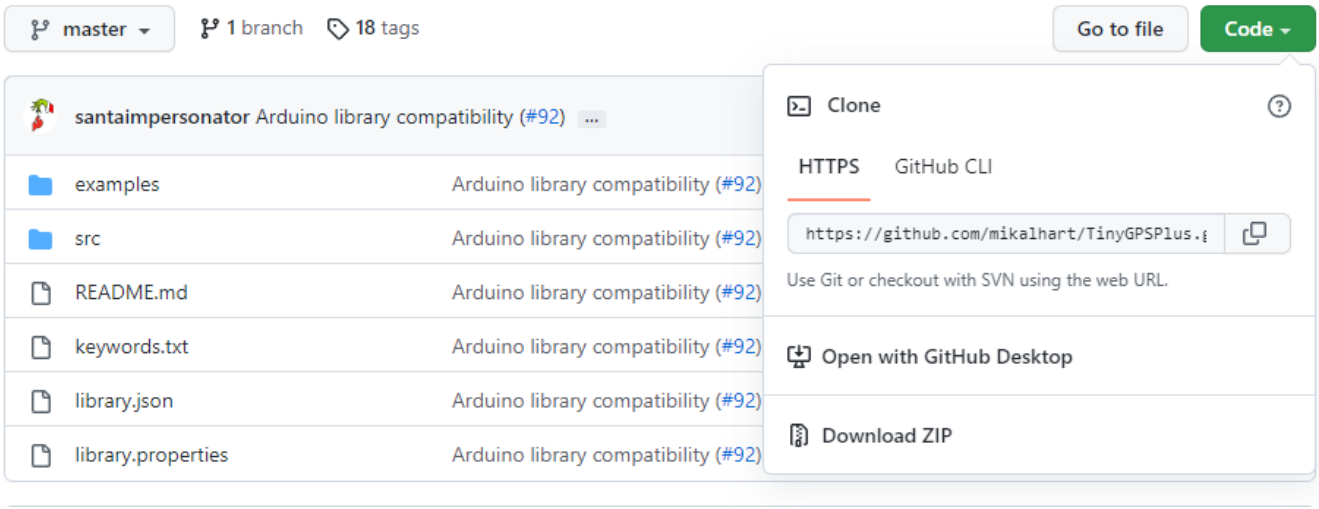
This code will read and collect the characters of data from the GPS until it gets to a return.

Prints the string data to the serial port

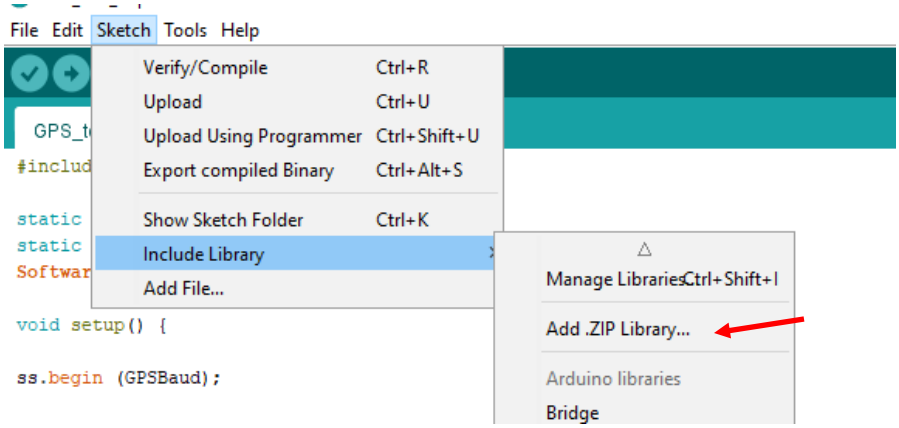
Upload the code and open the serial monitor. Change the baud rate to 115200 and watch the data come in. You might get trash at first, but after a while if you have sight to satellites will get useful data. Buildings and trees can block satellites. The lines of code you are seeing in the monitor is the NEMA data.



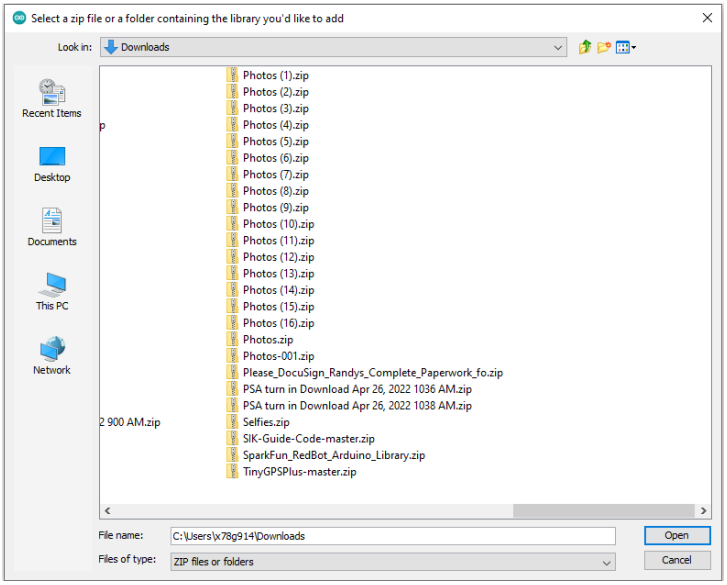
Now that we know the GPS module works, lets turn the NEMA data into something we can easily understand. We are going to use a custom made library that is not available for install via the Arduino library. The library we need can be found here at the following link: https://github.com/mikalhart/TinyGPSPlus. On the webpage is a green code button. Click it and click “download ZIP”.



When the download is finished, open Arduino and go to sketch->include library->import library.



Find the downloaded zip file and open it. The TinyGPSPlus library is now available for us to use in our code.



We are now ready for start writing the code.

```
#include <TinyGPS++.h>
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>

String gpstext;
static const int RXPin = 6, TXPin = 5;
static const uint32_t GPSBaud = 9600;
SoftwareSerial ss(RXPin, TXPin);
TinyGPSPlus gps;

void setup() {

ss.begin(GPSBaud);

Serial.begin(115200);

}

void loop() {

  while (ss.available() > 0)
    if (gps.encode(ss.read()))

      // See if we have a complete GPS data string
      if (displayInfo() != "0")
      {

        // Get GPS string
        gpstext = displayInfo();
Serial.println(gpstext);
}}
```

The libraries we will need.

Declaring variables

Creating two objects: SS and GPS

Our setup code to start the serial monitor and GPS

Our loop code is fairly simple. When the SS serial is available, read the gps and get data from the displayIfno() function and print the gpstext from displayInfor(). What is the displayInfo?

```
String displayInfo()
{
  // Define empty string to hold output
  gps.encode(ss.read());
  String gpsdata = "";

  // Get latitude and longitude
  if (gps.location.isValid())
  {
    gpsdata = String(gps.location.lat(), 6);
    gpsdata += (",");
    gpsdata += String(gps.location.lng(), 6);
    gpsdata += (",");
  }
  else
  {
    return "0";
  }

  // Get Date
  if (gps.date.isValid())
  {
    gpsdata += String(gps.date.year());
    gpsdata += ("-");
    if (gps.date.month() < 10) gpsdata += ("0");
    gpsdata += String(gps.date.month());
    gpsdata += ("-");
    if (gps.date.day() < 10) gpsdata += ("0");
    gpsdata += String(gps.date.day());
  }
  else
  {
    return "0";
  }
  // Space between date and time
  gpsdata += (" ");

  // Get time
  if (gps.time.isValid())
  {
    if (gps.time.hour() < 10) gpsdata += ("0");
    gpsdata += String(gps.time.hour());
    gpsdata += (":");
    if (gps.time.minute() < 10) gpsdata += ("0");
    gpsdata += String(gps.time.minute());
    gpsdata += (":");
    if (gps.time.second() < 10) gpsdata += ("0");
    gpsdata += String(gps.time.second());
  }
  else
  {
    return "0";
  }

  // Return completed string
  return gpsdata;
}
```

This is the displayInfo function. It is outside of the VOID setup and loop. Sense we desire to this function to give us information we will use String (data with many characters) instead of VOID (nothing returned).

Read the SS serial data and encode it. Start adding to the string

If the GPS gives a valid location, then add the latitude and longitude to the string up to 6 characters each. Otherwise, return a 0.

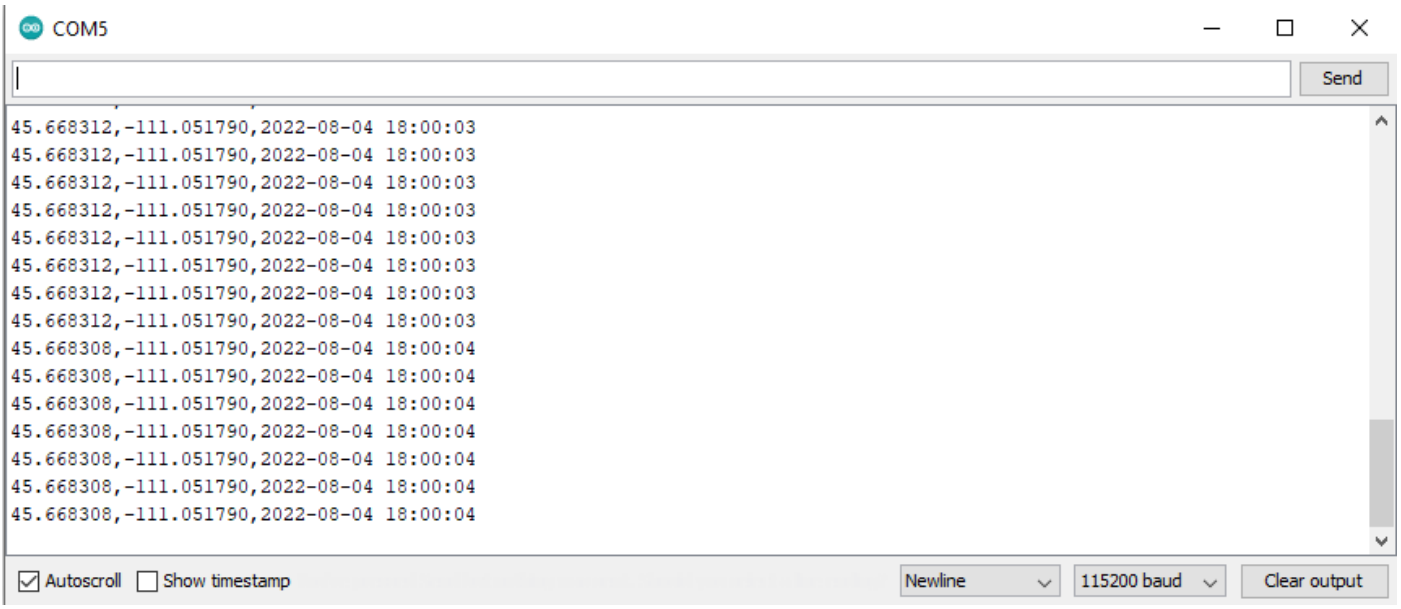
If we have a valid date from the GPS, add it to the string. This date will be in GST. Otherwise return a 0

IF we have a valid time from the GPS, add it to the string. This time will be in GST. Otherwise return a 0.

If the GPS is not reading send a 0

Return the String back to the loop.

Lets upload the code and see if it works. When the upload is complete open the serial monitor and see what the data looks like. It might take a moment for data to start coming in. If after a while you still not getting a location try moving outside (laptops make this easy). This data can be copied to the notepad and saved. If you import this data into excel or google maps you can see how accurate the module is.



Sources:

Building a GPS System - SparkFun Electronics. (n.d.). Www.sparkfun.com. https://www.sparkfun.com/gps

Workshop, DroneBot. “Using GPS Modules with Arduino & Raspberry Pi.” DroneBot Workshop, 26 June 2021, dronebotworkshop.com/using-gps-modules/. Accessed 5 Aug. 2022.