

In this project we are using the Arduino,

Adafruit Ultimate GPS, Virtuabotix SD card reader, and a Battery Clip to power the arduino during your mobile jaunts. You will need a 9V battery. If you are going to be doing a lot of mobile work, it is

probably worth it to invest in a good 9v Rechargeable battery system. Adafruit Ultimate GPS

When you get your gear gathered up, you will be ready to get this project going. First, hook the GPS

and Card Reader up as we did in Lesson 23. We will start with the software that we used in Lesson 23. This software correctly reads data from the GPS and then logs the data into two data files on the SD card. The problem, though, we did not really do any fine tuning of the data file we were creating. Displaying Data from Adafruit Ultimate GPS as track on Google Earth

In this lesson we want to create a data file that we can display as a track in Google Earth. In order for this to work, we have to save the data in the manner a KML file wants to see. We have to save the data as:

Decimal Longitude, Decimal latitude, Altitude

and it works to not use a line ending, but put a single white space after the Altitude. That is, delimit with commas as shown above, but then use one white space to delimit between successive lines of data.

Lesson 24 explained in great detail how to interpret the NMEA sentences. The challenge here is that the Arduino is very poor at parsing strings. The NMEA sentence would be easy to parse in Python, because Python is so good at easily manipulating strings. Since it is tedious to manipulate strings in Arduino, we are going to try and parse the data using numbers and math, not strings. To explain this, I

will give an example for Latitude, and Longitude will work in a similar manner. The latitude data we

get from the parsed data from the Adafruit GPS looks like this:

3051.8007 From lesson 24, we know that this represents 30 degrees and 51.8007 minutes. In order to parse this

in Arduino using just numbers I do the following. First create three variables deg, degWhole, and degDec. The variable deg will hold the final answer, degWhole holds the whole part of the number, and degDec holds the part to the right of the decimal. Adding degWhole to degDec gives you deg, your final answer. So, let's assume $x = 3051.8007$, our Latitude from the GPS.

Arduino

```
degWhole = float(int(x/100));
```

Dividing by 100 moves the decimal to the left two spots, and taking the int removes anything to the right of the decimal. Then converting back to float gives us a round number with no values to the right of the decimal. For 3051.8007 it turns it into 30.0, the whole number part of degrees. Now to get the fractional part, we need to take minutes and divide by 60. This will always be a number less than one. To get fractional part of the latitude in degrees, we do the following:

Arduino

```
degDec=(x-degWhole*100)/60;
```

We have to multiply degWhole by 100 to get the decimal moved back to the right by 2. Then we subtract that number from the original latitude, and we are left with minutes. Then divide by 60 and we have converted minutes to a fraction of a degree. We now have the whole part and the fractional part of the answer, so we just add those together:

Arduino

```
deg=degWhole + degDec;
```

We now have a nice decimal number, deg, which is the decimal representation of our Latitude. We have one more thing to deal with. If the GPS is reporting 'N', leave the number positive. If the GPS is

reporting 'S' (that we are in the Southern Hemisphere), you need to make your latitude number negative. Similarly, when you are doing your longitude, you need to make the longitude negative if

you are in the Western Hemisphere.

Finally, when you write the file, Google Earth will want you to store the data as:

Longitude, Latitude, altitude

It is curious that this is reverse from what you would do if you were entering coordinates into the Google Earth search bar. That wants Latitude first.

In any event, with the parsing and formatting described above, you can create a file that is almost ready to read into Google Earth. The code below will create your coordinates in the correct format for KML, which Google Earth will read. Please watch the video for a complete description of the code.

```
#include <SD.h> //Load SD card library
```

```
#include <SPI.h> //Load SPI Library
```

```
#include <Adafruit_GPS.h> //Install the adafruit GPS library
```

```
#include <SoftwareSerial.h> //Load the Software Serial library
```

```
SoftwareSerial mySerial(3,2); //Initialize the Software Serial port
```

```
Adafruit_GPS GPS(&mySerial); //Create the GPS Object
```

```
String NMEA1; //Variable for first NMEA sentence
```

```
String NMEA2; //Variable for second NMEA sentence
```

```
char c; //to read characters coming from the GPS
```

```
float deg; //Will hold position data in simple degree format
```

```
float degWhole; //Variable for the whole part of position
```

```
float degDec; //Variable for the decimal part of degree
```

```
int chipSelect = 4; //chipSelect pin for the SD card Reader
```

```
File mySensorData; //Data object you will write your sensor data to
```

```
void setup() {
```

```
  Serial.begin(115200); //Turn on serial monitor
```

```
  GPS.begin(9600); //Turn on GPS at 9600 baud
```

```
  GPS.sendCommand("$PGCMD,33,0*6D"); //Turn off antenna update nuisance data
```

```
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA); //Request RMC and GGA Sentences only
```

```
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); //Set update rate to 1 hz
```

```
  delay(1000);
```

```
  pinMode(10, OUTPUT); //Must declare 10 an output and reserve it to keep SD card happy
```

```
  SD.begin(chipSelect); //Initialize the SD card reader
```

```
  if (SD.exists("NMEA.txt")) { //Delete old data files to start fresh
```

```
    SD.remove("NMEA.txt");
```

```
  }
```

```
  if (SD.exists("GPSTData.txt")) { //Delete old data files to start fresh
```

```
    SD.remove("GPSTData.txt");
```

```
  }
```

```
}
```

```
void loop() {
```

```

readGPS();

if(GPS.fix==1) { //Only save data if we have a fix
  mySensorData = SD.open("NMEA.txt", FILE_WRITE); //Open file on SD card for writing
  mySensorData.println(NMEA1); //Write first NMEA to SD card
  mySensorData.println(NMEA2); //Write Second NMEA to SD card
  mySensorData.close(); //Close the file

  mySensorData = SD.open("GPSData.txt", FILE_WRITE);

  degWhole=float(int(GPS.longitude/100)); //gives me the whole degree part of Longitude
  degDec = (GPS.longitude - degWhole*100)/60; //give me fractional part of longitude
  deg = degWhole + degDec; //Gives complete correct decimal form of Longitude degrees
  if (GPS.lon=='W') { //If you are in Western Hemisphere, longitude degrees should be negative
    deg= (-1)*deg;
  }
  mySensorData.print(deg,4); //writing decimal degree longitude value to SD card
  mySensorData.print(","); //write comma to SD card

  degWhole=float(int(GPS.latitude/100)); //gives me the whole degree part of latitude
  degDec = (GPS.latitude - degWhole*100)/60; //give me fractional part of latitude
  deg = degWhole + degDec; //Gives complete correct decimal form of latitude degrees
  if (GPS.lat=='S') { //If you are in Southern hemisphere latitude should be negative
    deg= (-1)*deg;
  }
  mySensorData.print(deg,4); //writing decimal degree longitude value to SD card
  mySensorData.print(","); //write comma to SD card

  mySensorData.print(GPS.altitude); //write altitude to file
  mySensorData.print(" "); //format with one white space to delimit data sets

  mySensorData.close();
}

}

void readGPS() {

  clearGPS();
  while(!GPS.newNMEAreceived()) { //Loop until you have a good NMEA sentence
    c=GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //Parse that last good NMEA sentence
  NMEA1=GPS.lastNMEA();

  while(!GPS.newNMEAreceived()) { //Loop until you have a good NMEA sentence
    c=GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //Parse that last good NMEA sentence
  NMEA2=GPS.lastNMEA();
  Serial.println(NMEA1);
  Serial.println(NMEA2);
}

```

```

Serial.println("");
}

void clearGPS() { //Clear old and corrupt data from serial port
  while(!GPS.newNMEAreceived()) { //Loop until you have a good NMEA sentence
    c=GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //Parse that last good NMEA sentence

  while(!GPS.newNMEAreceived()) { //Loop until you have a good NMEA sentence
    c=GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //Parse that last good NMEA sentence
  while(!GPS.newNMEAreceived()) { //Loop until you have a good NMEA sentence
    c=GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //Parse that last good NMEA sentence
}

```

The nal thing we have to do is to put a “wrapper” around the coordinates to turn the coordinates into a .kml le Google Earth will like. I just do this manually. I open the text le on the SD card created

by the code above, and then just paste it into this KML wrapper, and save the le with a .kml extension. The KML wrapper is as follows:

```

C
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Style id="yellowPoly">
<LineStyle>
<color>7f00ffff</color>
<width>4</width>
</LineStyle>
<PolyStyle>
<color>7f00ff00</color>
</PolyStyle>
</Style>
<Placemark><styleUrl>#yellowPoly</styleUrl>
<LineString>
<extrude>1</extrude>
<tessellate>1</tessellate><altitudeMode>absolute</altitudeMode>
<coordinates>

</coordinates>
</LineString></Placemark>

</Document></kml>

```

Just paste your coordinate data in the le above between `<coordinates>` and `</coordinates>`, save with a .kml extension, and you should be able to open it with Google Earth and see your track.
Good
luck!