20140817 - Arduino GPS Demo

There are quite a few different types of GPS units that will work with the Arduino. GPS units generally speak to another device using a RS232 interface, this make interface very easy. Of course for the Arduino you'll need to convert RS232 levels to TTL levels, but in todays world that is quite easy to do. Most modern GPS that have Serial out are at TTL already. Most of the GPS units that are made for vehicles don't have RS232 out, unless it's a very old unit, or a special unit (some trucker GPS still have RS232 out for logging software reasons, and Some aviation GPS still have RS232 out for the same reason) Garmin uptil about 2005 made a unit that would interface with ham radio and talk APRS, AVMaps (a very expensive GPS unit) still makes a GPS that was designed with APRS (ham radio) in mind. So you can still find them. Most manufactures have removed the serial output because of "costs" and really not many people use or need that feature.
Hand Held units are a little bit of a different story, a lot of those units will have serial out - though it may not be a standard type of interface, or require a special cable.
Some Marine GPS, Some Trucker GPS, and Some Aviation units still have serial out, mostly used now for logging software, and not really for mapping anymore.
Marine GPS units are waterproof, and generally speaking have features that boaters would want like - "set and drift" and "man-over-board"
In order to have a GPS talk to the Arduino it needs to have serial out and be able to talk with a NMEA statement - more about that later.
So here is what I want to cover in this update:

Some background into what is GPS, how a GPS unit works, and some basic NMEA statements.

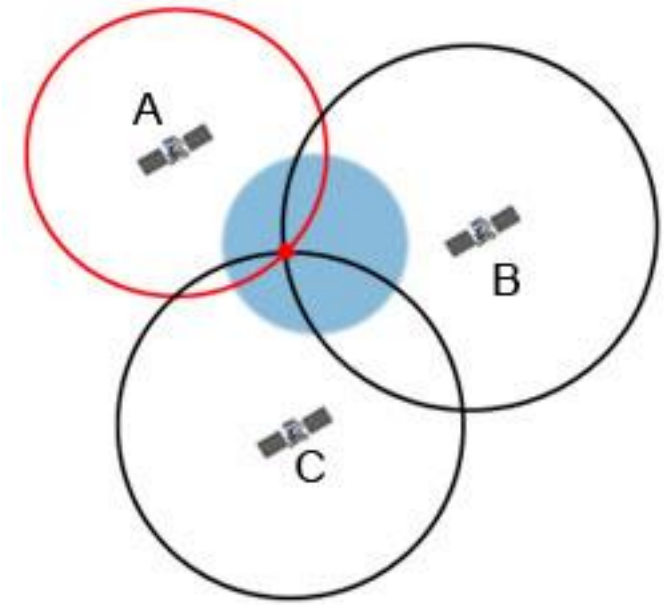
1) What is GPS - GPS stands for GLobal Positioning System, it is a satellite-based navigation system made up of a network 24 to 30 satellites placed into orbit by the U.S. Department of Defense. Originally intended for military applications, but in the 1980s, the government made the system available for civilian use.
The satellites are at an altitude of 20,000km (about 12400 miles)
No matter where you are on the planet at least 4 satellites are "visible" at any one time.

GPS was designed in 1973, and became fully operational in 1995.

There are other "GPS" systems - The Russians have "GLONASS" Global Navigation Satellite System, and the European Union has "Galileo Positioning System", India has a regional navigation satellite system. And lastly China has Beidou Navigation Satellite System.

GPS was originally embraced by the Marine community, later the aviation, and the general public at large. (My own opinion - Some of the best GPS units are still some of the old original Marine units)

2) How does GPS work - GPS works using a method of navigation called triangulation. Each Satellite transmits its “position”, and other information, at a precise time interval. And at a precise frequency. The GPS receiver, knows how often it should receive this information, it also knows what the starting frequency is. It can figure out the frequency changes (known as doppler shift) and can figure out how long it took for that change to happen. As long as it can get at least 3 Satellites it can get a fix, the more Satellites it gets the better the fix will be.



Imagine you are standing somewhere on Earth with three satellites in the sky above you. If you know how far away you are from satellite A, then you know you must be located somewhere on the red circle. If you do the same for satellites B and C, you can work out your location by seeing where the three circles intersect. This is just what your GPS receiver does, although it uses overlapping spheres rather than circles.
The more satellites there are above the horizon the more accurately your GPS unit can determine where you are.

The GPS receivers also take into account the “Theory of Relativity” which is not so much a Theory anymore. But in short “General Relativity” states clocks on satellites will run faster than a clock on Earth. while “Special Relativity” states that the clocks on Satellites are moving relative to a clock on Earth they will appear to run slower.

This youtube video explains it pretty good - he even touchs on (but doesn't come right out and say that it uses the doppler effect)
http://youtu.be/3zRlbboMvb0

Most of this information was found on this site:
http://www.physics.org/article-questions.asp?id=55

I don't know if it's just me, or if these site just don't think people will understand the doppler effect. Maybe it's because I had some of this training in ET-A school, when we were using the old satellite system, in the early 90s before GPS was complete.
3) Basic information about the NMEA Statements - or a little on how the receivers work.
So what is NMEA or "National Marine Electronics Association" - NMEA is the output string from a GPS unit, it contains all the data received, the calculation data (in some cases), the predicted accuracy of the fix (in some cases), the Fix itself, way point information (in some cases), and a lot of other data.

Each manufacturer complies with at very least the basic NMEA sentences. And most expand on the basic NMEA with their own proprietary sentences. (Garmin being the worse offender) Even thou every GPS receiver has NMEA sentences, it doesn't mean that every manufacturer gives (or has) a RS232 interface for users to have access to that information (see above).

At the time of this writting there are at least 3 different versions of NMEA sentences. And a few hundred proprietary sentences (some are documented while others are not).

Depending on how old your GPS unit is will depend on how you connect to it, most you'll find will be 4800 baud 8N1, while very old units maybe 1200 baud, and some of the new Garmins are 9600 baud (Thou most of those can be switched for older equipment).

NMEA consists of sentences, the first word of which, called a data type, defines the interpretation of the rest of the sentence. Each Data type would have its own unique interpretation and is defined in the NMEA standard.

Here are a couple of sites that explain the sentences in detail:
http://www.gpsinformation.org/dale/nmea.htm
http://aprs.gids.nl/nmea/

The one sentence we are interested in is:

$GPGGA - Global Positioning System Fix Data

$GPGGA

Global Positioning System Fix Data

Name	Example Data	Description
Sentence Identifier	$GPGGA	Global Positioning System Fix Data
Time	170834	17:08:34 Z
Latitude	4124.8963, N	41d 24.8963' N or 41d 24' 54" N
Longitude	08151.6838, W	81d 51.6838' W or 81d 51' 41" W
Fix Quality: - 0 = Invalid - 1 = GPS fix - 2 = DGPS fix	1	Data is from a GPS fix
Number of Satellites	05	5 Satellites are in view
Horizontal Dilution of Precision (HDOP)	1.5	Relative accuracy of horizontal position
Altitude	280.2, M	280.2 meters above mean sea level
Height of geoid above WGS84 ellipsoid	-34.0, M	-34.0 meters
Time since last DGPS update	blank	No last update
DGPS reference station id	blank	No station id
Checksum	*75	Used by program to check for transmission errors

Courtesy of Brian McClure, N8PQI.

Global Positioning System Fix Data. Time, position and fix related data for a GPS receiver.

eg2. $--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx

hhmmss.ss = UTC of position

llll.ll = latitude of position

a = N or S

yyyyy.yy = Longitude of position

a = E or W
x = GPS Quality indicator (0=no fix, 1=GPS fix, 2=Dif. GPS fix)
xx = number of satellites in use
x.x = horizontal dilution of precision
x.x = Antenna altitude above mean-sea-level
M = units of antenna altitude, meters
x.x = Geoidal separation
M = units of geoidal separation, meters
x.x = Age of Differential GPS data (seconds)
xxxx = Differential reference station ID
eg3. $GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
1 = UTC of Position
2 = Latitude
3 = N or S
4 = Longitude
5 = E or W
6 = GPS quality indicator (0=invalid; 1=GPS fix; 2=Diff. GPS fix)
7 = Number of satellites in use [not those in view]
8 = Horizontal dilution of position
9 = Antenna altitude above/below mean sea level (geoid)
10 = Meters (Antenna height unit)
11 = Geoidal separation (Diff. between WGS-84 earth ellipsoid and mean sea level. -=geoid is below WGS-84 ellipsoid)
12 = Meters (Units of geoidal separation)
13 = Age in seconds since last update from diff. reference station
14 = Diff. reference station ID#
15 = Checksum

This statement will give us the latitude and longitude and if the GPS believes it's a valid fix or not. It will also tell you how many satellites, and some other useful information.

We are also interested in this sentence:

$GPRMC

Recommended minimum specific GPS/Transit data
eg1. $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
eg2. $GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

225446 Time of fix 22:54:46 UTC
A Navigation receiver warning A = OK, V = warning
4916.45,N Latitude 49 deg. 16.45 min North
12311.12,W Longitude 123 deg. 11.12 min West
000.5 Speed over ground, Knots
054.7 Course Made Good, True
191194 Date of fix 19 November 1994
020.3,E Magnetic variation 20.3 deg East
*68 mandatory checksum

eg3. $GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70
1 2 3 4 5 6 7 8 9 10 11 12

1 220516 Time Stamp
2 A validity - A-ok, V-invalid
3 5133.82 current Latitude
4 N North/South
5 00042.24 current Longitude
6 W East/West
7 173.8 Speed in knots
8 231.8 True course
9 130694 Date Stamp
10 004.2 Variation
11 W East/West
12 *70 checksum

eg4. $GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh
1 = UTC of position fix
2 = Data status (V=navigation receiver warning)
3 = Latitude of fix
4 = N or S
5 = Longitude of fix
6 = E or W
7 = Speed over ground in knots
8 = Track made good in degrees True
9 = UT date
10 = Magnetic variation degrees (Easterly var. subtracts from true course)

11 = E or W
12 = Checksum

So now that we have a background on what is going on, and a little understanding of how it works lets look at the code.
http://playground.arduino.cc/Tutorials/GPS

So I am using a very old GPS puck, and it only outputs 4 NMEA sentences, it's not what this sketch is looking for (well other then the $GPGGA sentence), but I am able to get a fix.

The Arduino Code

```
/*

Example code for connecting a Parallax GPS module to the Arduino

Igor Gonzalez Martin. 05-04-2007
 igor.gonzalez.martin@gmail.com

English translation by djmatic 19-05-2007

Listen for the $GPRMC string and extract the GPS location data from this.
 Display the result in the Arduino's serial monitor.

*/

#include <string.h>
 #include <ctype.h>

int ledPin = 13;                // LED test pin
 int rxPin = 0;                 // RX PIN
 int txPin = 1;                 // TX TX
 int byteGPS=-1;
 char linea[300] = "";
 char comandoGPR[7] = "$GPRMC";
 int cont=0;
 int bien=0;
 int conta=0;
 int indices[13];
```

```
void setup() {
  pinMode(ledPin, OUTPUT);       // Initialize LED pin
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  Serial.begin(4800);
  for (int i=0;i<300;i++){       // Initialize a buffer for received data
    linea[i]=' ';
  }
 }

void loop() {
  digitalWrite(ledPin, HIGH);
  byteGPS=Serial.read();         // Read a byte of the serial port
  if (byteGPS == -1) {           // See if the port is empty yet
    delay(100);
  } else {
    // note: there is a potential buffer overflow here!
    linea[conta]=byteGPS;        // If there is serial port data, it is put in the buffer
    conta++;
    Serial.print(byteGPS, BYTE);
    if (byteGPS==13){            // If the received byte is = to 13, end of transmission
      // note: the actual end of transmission is <CR><LF> (i.e. 0x13 0x10)
      digitalWrite(ledPin, LOW);
      cont=0;
      bien=0;
      // The following for loop starts at 1, because this code is clowny and the first byte is
the <LF> (0x10) from the previous transmission.
      for (int i=1;i<7;i++){     // Verifies if the received command starts with $GPR
        if (linea[i]==comandoGPR[i-1]){
          bien++;
        }
      }
      if(bien==6){               // If yes, continue and process the data
        for (int i=0;i<300;i++){
          if (linea[i]==','){    // check for the position of the  "," separator
            // note: again, there is a potential buffer overflow here!
            indices[cont]=i;
            cont++;
          }
          if (linea[i]=='*'){    // ... and the "*"
```

```
          indices[12]=i;
          cont++;
        }
      }
      Serial.println("");      // ... and write to the serial port
      Serial.println("");
      Serial.println("---------------");
      for (int i=0;i<12;i++){
        switch(i){
          case 0 :Serial.print("Time in UTC (HhMmSs): ");break;
          case 1 :Serial.print("Status (A=OK,V=KO): ");break;
          case 2 :Serial.print("Latitude: ");break;
          case 3 :Serial.print("Direction (N/S): ");break;
          case 4 :Serial.print("Longitude: ");break;
          case 5 :Serial.print("Direction (E/W): ");break;
          case 6 :Serial.print("Velocity in knots: ");break;
          case 7 :Serial.print("Heading in degrees: ");break;
          case 8 :Serial.print("Date UTC (DdMmAa): ");break;
          case 9 :Serial.print("Magnetic degrees: ");break;
          case 10 :Serial.print("(E/W): ");break;
          case 11 :Serial.print("Mode: ");break;
          case 12 :Serial.print("Checksum: ");break;
        }
        for (int j=indices[i];j<(indices[i+1]-1);j++){
          Serial.print(linea[j+1]);
        }
        Serial.println("");
      }
      Serial.println("---------------");
    }
    conta=0;                  // Reset the buffer
    for (int i=0;i<300;i++){    //
      linea[i]=' ';
    }
  }
 }
}
```

The output of the code should look something like this:

...
 ...

$GPGGA,154653,4428.2011,N,00440.5161,W,0,00,,-00044.7,M,051.6,M,,*6C
 $GPGSA,A,1,,,,,,,,,,,,,,*1E
 $GPGSV,3,1,10,02,50,290,003,10,25,24,045,35,27,56,145,00,,,,,,,*78
 $GPRMC,154653,V,4428.2011,N,00440.5161,W,000.5,342.8,050407,,,N*7F

 Time in UTC (HhMmSs): 154653
 Status (A=OK,V=KO): V
 Latitude: 4428.2011
 Direction (N/S): N
 Longitude: 00440.5161
 Direction (E/W): W
 Speed in knots: 000.5
 Direction in degrees: 342.8
 Date in UTC (DdMmAa): 050407
 Magnetic variation:
 Variation (E/W):
 Mode: A

...
 ...

MY Output looks like this:

1036808287739067724448494448444849444844505044544448494849444449484844
4449484844449484844444948521336718071716544444444444448464844956485 6
4952445446544487425469131036808287739067724448504448444850444844505044
5444485044844495644544448504448444850444844485048484444504848444450 48
5213367180717165444444444444484448484444444444444425454131036718071831
6544654449444444444444444444444444444846484844846484844846484842484 8
1310367180827767444851534848504486445157504846535057484478444856525051
4654504851448744484648484844484648444956485649524454465444874254681 3

Time in UTC (HhMmSs): 035002
Status (A=OK,V=KO): V

Latitude: 3920.5290
Direction (N/S): N
Longitude: 08423.6203
Direction (E/W): W
Velocity in knots: 0.000
Heading in degrees: 0.0
Date UTC (DdMmAa): 180814
Magnetic degrees: 6.6
(E/W):
Mode: $GPRMC,035002,V,3920.5290,N,08423.6203,W,0.000,0.0,180814,6.6,W

I think I am getting the very long string because my GPS isn't sending the sentences that the sketch expects to see.