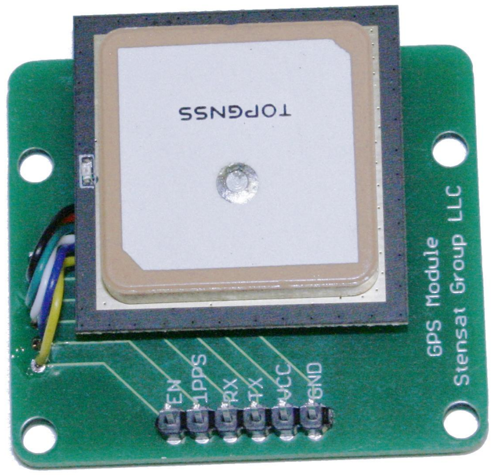
GPS Receiver Interface



Introduction

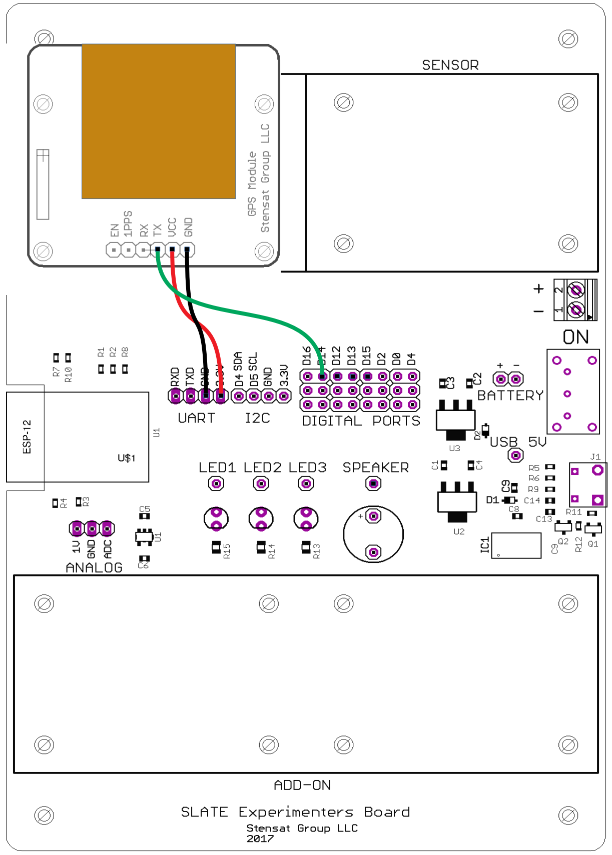
- GPS is the Global Positioning System which consists of at least 24 satellites that transmit a signal used by a receiver to triangulate and determine the receivers position on earth.
- This lesson will demonstrate how to use a GPS receiver and extract time and position data.

GPS Receiver

- The GPS receiver sends it’s data using a serial interface. Since the SLATE serial interface is connected and used by the USB adapter, a digital pin will need to be used as a serial interface.
- A software serial library will be used to use a digital pin as a serial interface.

GPS Receiver

- Install the GPS receiver module in one of the sensor positions. It can be stacked on top of another module using the included standoffs. The standoffs are aluminum so be careful to not over tighten the nuts on the screws of the standoff.
- Connect the GPS receiver GND to a GND on the SLATE.
- Connect the GPS receiver VCC to the UART 3V.
- Connect the GPS receiver TX pin to digital pin 14.



GPS Echo

- The first program will demonstrate how to use the Software Serial library. The program will echo the data from the GPS receiver to the serial monitor.
- The object gps is created as a serial interface using pins 14 as the receiver and pin 11 as the transmitter. Pin 11 will not be used but a transmit pin has to be defined.
- In the **setup()** function, the gps serial interface is configured to operate at 9600 baud.
- In the **loop()** function, as a character is received from the GPS receiver, it is sent to the serial interface to be displayed in the serial monitor.

```
#include <SoftwareSerial.h>

SoftwareSerial gps(14,11);

void setup() {
  Serial.begin(115200);
  gps.begin(9600);
}

void loop() {
  if(gps.available() > 0)
    Serial.write(gps.read());
}
```

GPS NMEA Messages

- Save the program on the previous page as gps_echo.
- Run the program and open the serial monitor. Observe the different sentences coming from the GPS receiver.
- Notice all the GPS sentences start with **$GN**. After those two characters are three characters that define the type of sentence. The sentence of interest for now is **$GNGGA**. This sentence provides time, position, altitude, quality of measurement, number of satellites and other information.
- Also, notice the data in each message is separated by commas. This is the delimeter that will be used to extract the information.
- For information of all the different sentences, go to: http://aprs.gids.nl/nmea/

GPS Receiver

- the $GNGGA sentence consists of the following:
- Time is formatted as hhmmss.ss.
- Latitude is formatted so the first two digits is degrees and the of the digits including the decimal point in minutes.
- Longitude is formatted with the first three digits in degrees and the remaining digitals minutes.

Name	Example	Description
Time	170834.00	Time in hhmmss.ss
Latitude	4124.8963	Latitude in degrees
N/S	N	North or South
Longitude	08151.6838	Longitude in degrees
E/W	W	East or West
Fix Quality	1	0 – invalid, 1 – GPS, 2 – DGPS
No of Sats	12	Number of satellites being used
HDOP	1.1	Horizontal position accuracy
Altitude	83.2	Altitude in meters
Height of GEOID	-34	Height of geoid above WGS84 ellipsoid
Time last DGPS	0	Ignore
DGPS Station	0	Ignore
Checksum	40	Checksum

GPS Receiver

- Fix quality indicates if enough satellites are being tracked to correctly calculate the position.
- The No. of satellites indicates how many satellites are being used to calculate the position. More is generally better.
- HDOP is horizontal dilution of position. It is an indicator how how accurate the position is being calculated. Lower value is better.

Name	Example	Description
Time	170834.00	Time in hhmmss.ss
Latitude	4124.8963	Latitude in degrees
N/S	N	North or South
Longitude	08151.6838	Longitude in degrees
E/W	W	East or West
Fix Quality	1	0 – invalid, 1 – GPS, 2 – DGPS
No of Sats	12	Number of satellites being used
HDOP	1.1	Horizontal position accuracy
Altitude	83.2	Altitude in meters
Height of GEOID	-34	Height of geoid above WGS84 ellipsoid
Time last DGPS	0	Ignore
DGPS Station	0	Ignore
Checksum	40	Checksum

GPS Receiver

- Altitude is indicated in meters. You will notice altitude can vary significantly. It is the least accurate of the position data.
- The height of the GEOID can be ignored for now. For more information on it, go to https://www.esri.com/news/arcuser/0703/geoid1of3.html
- Differential GPS is not used so the last two data items can be ignored.

Name	Example	Description
Time	170834.00	Time in hhmmss.ss
Latitude	4124.8963	Latitude in degrees
N/S	N	North or South
Longitude	08151.6838	Longitude in degrees
E/W	W	East or West
Fix Quality	1	0 – invalid, 1 – GPS, 2 – DGPS
No of Sats	12	Number of satellites being used
HDOP	1.1	Horizontal position accuracy
Altitude	83.2	Altitude in meters
Height of GEOID	-34	Height of geoid above WGS84 ellipsoid
Time last DGPS	0	Ignore
DGPS Station	0	Ignore
Checksum	40	Checksum

GPS Receiver

- The following code will parse the $GNGGA sentence.
- To organize the data better, a data structure will be created. A data structure is a way to organize variables into a group. The structure **gngga** is defined. No variable are created at this time. It will be created later.
- The software serial interface is created and called gps just as before. The same pins will be used.

```
#include <SoftwareSerial.h>
#include <string.h>

struct gngga {
  char ttag[11];
  char latitude[12];
  char northsouth;
  char longitude[12];
  char eastwest;
  float altitude;
  int sats;
  float hdop;
};

SoftwareSerial gps(14,11);
```

Beginning of Program

GPS Receiver

- This function parses the **$GNGGA** sentence. The first argument is the pointer to the sentence to be parsed. The second argument is the pointer to the structure.
- A pointer *token is created. The function **strtok()** is used to split the string into multiple pieces based on the delimiter.
- The first argument is the string that is passed. The second argument is the delimiter character.

```
void getPosition(char *b,struct gngga *ptr) {
  char *token;
  token = strtok(b,",");
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->ttag,token);
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->latitude,token);
  token = strtok(NULL,",");
  ptr->northsouth = token[0];
  token = strtok(NULL,",");
  strcpy(ptr->longitude,token);
  token = strtok(NULL,",");
  ptr->eastwest = token[0];
  token = strtok(NULL,",");
  token = strtok(NULL,",");
  ptr->sats = atoi(token);
  token = strtok(NULL,",");
  ptr->hdop = atof(token);
  token = strtok(NULL,",");
  ptr->altitude = atof(token);
}
```

GPS Receiver

- In the GNGGA sentence, the time is not always populated so it needs to be checked if there is a value. The if() statement checks if time is available. If not the function returns immediately.
- If the GPS receiver has not run long enough to determine the position, the latitude and longitude are not filled in the sentence. The **if()** statement checks if the latitude has been filled in the sentence. If the latitude is not available, the length is one. This is an indication that the GPS receiver has not been determined.

```c
void getPosition(char *b,struct gngga *ptr) {
  char *token;
  token = strtok(b,",");
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->ttag,token);
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->latitude,token);
  token = strtok(NULL,",");
  ptr->northsouth = token[0];
  token = strtok(NULL,",");
  strcpy(ptr->longitude,token);
  token = strtok(NULL,",");
  ptr->eastwest = token[0];
  token = strtok(NULL,",");
  token = strtok(NULL,",");
  ptr->sats = atoi(token);
  token = strtok(NULL,",");
  ptr->hdop = atof(token);
  token = strtok(NULL,",");
  ptr->altitude = atof(token);
}
```

GPS Receiver

- the **setup()** function configures the serial interface and the GPS serial interface.
- In the **loop()**, character array **buf** is declared which will hold the sentence received from the GPS receiver. The structure **pos** is declared. This allocates memory to hold the group of variables.
- The gps serial interface is checked for any available data. If data is available, the code inside the brackets is executed.

```
void setup() {
  Serial.begin(115200);
  gps.begin(9600);
}

void loop() {
  char buf[128];
  struct gngga pos;
  if(gps.available() > 0) {
    bzero(buf,128);
    gps.readBytesUntil('\n',buf,128);
    if(!strncmp("$GNGGA",buf,6)) {
      getPosition(buf,&pos);
      Serial.println(pos.ttag);
      Serial.println(pos.latitude);
      Serial.println(pos.longitude);
      Serial.println(pos.altitude,1);
      Serial.println(pos.sats);
      Serial.println(pos.hdop,2);
    }
  }
}
```

GPS Receiver

- The first thing done is the buf array is cleared by setting all the elements to zero using the **bzero()** function. This is to delete the old sentence so no old data gets interpreted as new data by accident. The **readBytesUntil()** function does not clear the array before using it.
- The sentence is read from the GPS receiver. The terminating character is the linefeed.
- strncmp() compares the first six bytes of the array. If the string in quotes equals the first six bytes of the array, the result is zero. That is why the (!) invert symbol is used in front.

```
void setup() {
  Serial.begin(115200);
  gps.begin(9600);
}

void loop() {
  char buf[128];
  struct gngga pos;
  if(gps.available() > 0) {
    bzero(buf,128);
    gps.readBytesUntil('\n',buf,128);
    if(!strncmp("$GNGGA",buf,6)) {
      getPosition(buf,&pos);
      Serial.println(pos.ttag);
      Serial.println(pos.latitude);
      Serial.println(pos.longitude);
      Serial.println(pos.altitude,1);
      Serial.println(pos.sats);
      Serial.println(pos.hdop,2);
    }
  }
}
```

GPS Receiver

- Next, the position information is extracted from the sentence by calling the **getPosition()** function. The address of the buf array and the structure are passed to the function. This is called passing arguments by reference.
- Lastly, the information is displayed to the serial monitor.
- Try the program.

```
void setup() {
  Serial.begin(115200);
  gps.begin(9600);
}

void loop() {
  char buf[128];
  struct gngga pos;
  if(gps.available() > 0) {
    bzero(buf,128);
    gps.readBytesUntil('\n',buf,128);
    if(!strncmp("$GNGGA",buf,6)) {
      getPosition(buf,&pos);
      Serial.println(pos.ttag);
      Serial.println(pos.latitude);
      Serial.println(pos.longitude);
      Serial.println(pos.altitude,1);
      Serial.println(pos.sats);
      Serial.println(pos.hdop,2);
    }
  }
}
```

GPS Receiver

- At this point, the GNGGA sentence was parsed with the position data extracted and put into another character array. This section will show how to convert the latitude and longitude into fractional degrees that could be used for plotting or navigation.
- Two functions are added, one to convert latitude and the other to convert longitude. The functions will separate the degrees from the minutes and convert the degrees and minutes into separate floating point variables and then combined for a complete coordinate.

GPS Receiver

- The first thing to do is update the structure and add floating point variables for the latitude and longitude.
- Variables **flatitude** and **flongitude** are added to the structure.

```
#include <SoftwareSerial.h>
#include <string.h>

struct gngga {
  char ttag[11];
  char latitude[12];
  char northsouth;
  char longitude[12];
  char eastwest;
  float altitude;
  int sats;
  float hdop;
  float flatitude;
  float flongitude;
};

SoftwareSerial gps(14,11);
```

GPS Receiver

- The function **convertlat()** will convert the latitude to a floating point value.
- Three variables are created in the function. degree and minutes are the floating point variables to hold the degrees and minutes.
- character array deg will hold the degrees digits to be converted to floating point.
- **strncpy()** copies characters from the p array to deg array. Only two characters are copied. They are the degrees digits.
- The degrees and minutes character arrays are converted to floating point.
- The final result is returned by adding minutes divided by 60 to degrees.

```
float convertlat(char *p) {
  float degree;
  char deg[4];
  float minutes;
  strncpy(deg,p,2);
  deg[2] = 0;
  degree = atof(deg);
  minutes = atof(&p[2]);
  return degree + minutes/60.0;
}
```

GPS Receiver

- The function **convertlong()** will convert the longtitude to a floating point value.
- The function is the same as **convertlat()** with the only difference is 3 characters are copied for the degrees.

```
float convertlong(char *p) {
  float degree;
  char deg[4];
  float minutes;
  strncpy(deg,p,3);
  deg[3] = 0;
  degree = atof(deg);
  minutes = atof(&p[3]);
  return degree + minutes/60.0;
}
```

GPS Receiver

- Update the function **getPosition()** with the lines in bold.
- The latitude is converted to floating point. The side of the equator is checked and if the latitude is in the southern hemisphere, the latitude is made negative. This is the standard convention.
- The longitude is converted to floating point and if the coordinate is in the west, it is made negative.

```c
void getPosition(char *b,struct gngga *ptr) {
  char *token;
  token = strtok(b,",");
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->ttag,token);
  token = strtok(NULL,",");
  if(strlen(token) == 1) return;
  strcpy(ptr->latitude,token);
  token = strtok(NULL,",");
  ptr->northsouth = token[0];
  token = strtok(NULL,",");
  strcpy(ptr->longitude,token);
  token = strtok(NULL,",");
  ptr->eastwest = token[0];
  token = strtok(NULL,",");
  token = strtok(NULL,",");
  ptr->sats = atoi(token);
  token = strtok(NULL,",");
  ptr->hdop = atof(token);
  token = strtok(NULL,",");
  ptr->altitude = atof(token);
  ptr->flatitude = convertlat(ptr->latitude);
  if(ptr->northsouth == 'S') ptr->flatitude = -ptr->flatitude;
  ptr->flongitude = convertlong(ptr->longitude);
  if(ptr->eastwest == 'W') ptr->flongitude = -ptr->flongitude;
}
```

GPS Receiver

- In the loop() function, add two print statements to display the floating point version of the latitude and longitude.
- Compile the program and run it and observe the output.

```
void setup() {
  Serial.begin(115200);
  gps.begin(9600);
}

void loop() {
  char buf[128];
  struct gngga pos;
  if(gps.available() > 0) {
    bzero(buf,128);
    gps.readBytesUntil('\n',buf,128);
    if(!strncmp("$GNGGA",buf,6)) {
      getPosition(buf,&pos);
      Serial.println(pos.ttag);
      Serial.println(pos.latitude);
      Serial.println(pos.longitude);
      Serial.println(pos.altitude,1);
      Serial.println(pos.sats);
      Serial.println(pos.hdop,2);
      Serial.println(pos.flatitude,5);
      Serial.println(pos.flongitude,5);
    }
  }
}
```