soft s

**GeoCache Hunt**

# OBJECTIVES

The primary objective for these labs is to provide you hands-on learning experiences programming the Arduino microcontroller to control a variety of hardware components supplied with the [SparkFun Inventors Kit (SIK)](https://www.sparkfun.com/products/12060).

The objective of this project is to work as a team programming the SparkFun RedBoard, Adafruit GPS Shield and Adafruit NeoPixel to perform a GeoCache Hunt (similar to the worldwide [GeoCaching](https://www.geocaching.com/) community). The teams will be given sufficient classroom time for project coding and debugging. Then on Finals Day, each team will be given latitude and longitude locations of four survey flags located on campus to retrieve. The teams will then leave to locate and collect their assigned survey flags, return to the classroom, and display their plotted course using Google "My Maps", and demonstrate how the NeoPixel was used to direct you to the target.

**NOTE**: Quizzes are tailored to test your hardware and programming knowledge gained executing this lab.

**NOTE**: Read all information provided you for this project before engaging a Lab Specialists or Course director with clarifying questions.

# INSTRUCTIONS

Please refer to the "AtmelStudioInstructions.doc" within the "Resource.zip" for help creating Atmel Studio library or executable projects. Be sure to review the [SparkFun SIK Guide](https://cdn.sparkfun.com/datasheets/Kits/RedBoard_SIK_3.2.pdf) and [SparkFun SIK Guide Code](http://www.sparkfun.com/sikcode) for identifying circuit construction and programming techniques utilized in this lab. The key to successfully completing any assignment is to first understand what you are to do conceptually. It is imperative that you carefully study all the relative materials supplied for this lab before starting to code.

Each team must brainstorm their solution, divvy up and perform the work, integrate and test the solution, then perform the GeoCache Hunt on Finals Day.

## HARDWARE IMAGE



## STEP-BY-STEP

You will be supplied with a GPS Shield, Secure Digital Card, 9 Volt Rechargeable Battery and 9V Battery Cable for this project. You may use your Laptop, One Sheeld, Cell Phone or Android Tablet for development and debugging of your project code only. You will use the NeoPixel as your visual display for locating the targets. The 9 volt battery and cable will power the RedBoard and shields during the GeoCache Hunt. You are also provided a skeleton "GeoCache " executable source code which periodically retrieves the required "GPRMC" string message delivered by GPS board.

**NOTE**: You will responsible for returning the GPS Shield, Secure Digital Card, 9 Volt Rechargeable Battery and 9V Battery Cable upon completion of the GeoCache Hunt on Day 11. The cost of these devices is about $60.00 total, which would be applied to your tuition costs if lost. Also, at the end of the GeoCache Hunt, you will return the flags you retrieved.

**NOTE**: You are not permitted to use the LCD or SERVO in this project.

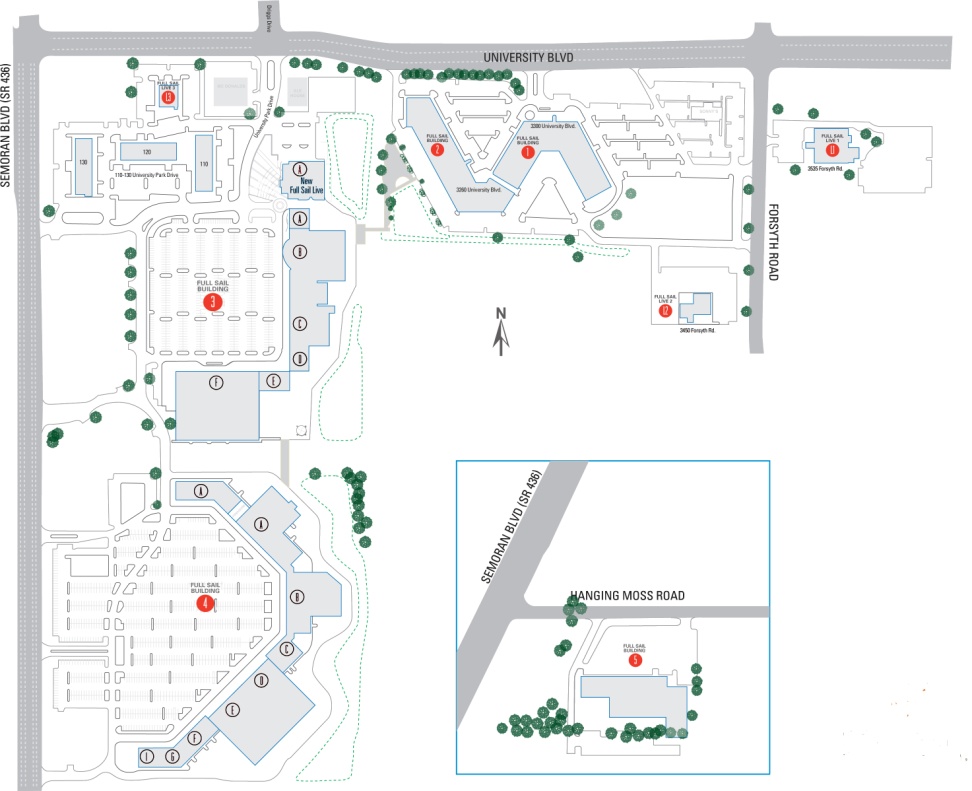
The GPRMC string message provides a comma separated set of parameters available from the GPS device. Depending on the number of satellites that have been acquired, the GPS will provide your current position within 3 to 10 feet. A brief description of the GPRMC message is given within the source code of the skeleton project, and can also be found described in the manufacturers GPS specifications document located in the "Resources\GPSTracker\\*" folder.

**NOTE**: You should consider using sprintf(), dtostrf(), strtok() and strtod() for GPS string message parameter parsing and converting between strings and floats. There are also methods provided by the String Class ( eg: "String(float, 6).c\_str()" ) to convert floats and other data types to strings.

Your code must be complete before GeoCache Day 11. On GeoCache day you will be given latitude and longitude coordinates of 4 flags located on campus, which you will retrieve in the order given you. You will enter these coordinates into your program and then compile your code. You must implement a push button method for selecting the flag you are hunting. You may use the NeoPixel to indicate which flag is selected (1, 2, 3 or 4).

**NOTE**: You must not use digital pins 0, 1, 6, 7, 8, 10, 11, 12, 13 for implementing your button. These digital pins are being used by GPS, SecureDigital and NeoPixel.

**NOTE**: All flags are located on Full Sail Campus within 20 feet of cement paths or driveways (excluding buildings 5, 11 and 12 areas). All flags are outdoors. Depending on the angle in which you approach the flag, you may need to be standing within feet of the flag to see it visually, and don't be surprised if you find a flag on a second floor building walkway. Your GeoCache Hunt should not take more than an hour, and your trip will be about 1.7 miles of walking all together... unless of course, you get really lost!!! You **DO NOT** cross any major roads such as University Boulevard, Semoran Boulevard or Forsyth Road, or enter any dense forests, climb any boulders, or wade through any ponds or streams !!



There are 5 parameters you must parse out of the GPRMC message and use in your calculations (Latitude, N/S Latitude Indicator, Longitude, E/W Longitude Indicator, North Course Over Ground). The GPS latitude and longitude coordinates are given in Degrees Minutes format ("DDMM.MMMM"). These coordinates must be converted to Decimal Degrees format ("DDD.DDDD") for use in calculating bearing and distance to the target. You must create a function to do this. Be sure to include the hemisphere indicator in your calculations.

With the latitude and longitude and course over ground parameters supplied by the GPS Shield and the selected GeoCache target coordinates, your program must calculate the distance in feet and bearing in degrees from your GPS position to the GeoCache target position, and then display this information on the NeoPixel LED's in a manner that allows you to locate the target flag. Targets can be hidden visually, requiring you be standing on top of them to see it. Targets can also be over 1500 feet from your current position. It is highly recommended that the NeoPixel be designed to display a target distance between 1 and 1500 feet. You should be able to find distance and bearing calculations by performing Google searches containing combinations of the following key words: "Arduino GPS Robot RMC Waypoint Distance Bearing". Examples follow:

<http://www.gpsinformation.org/dale/nav.htm>

<http://www.csgnetwork.com/gpscoordconv.html>

<http://www.movable-type.co.uk/scripts/latlong.html>

<http://www.rapidtables.com/convert/number/degrees-minutes-seconds-to-degrees.htm>

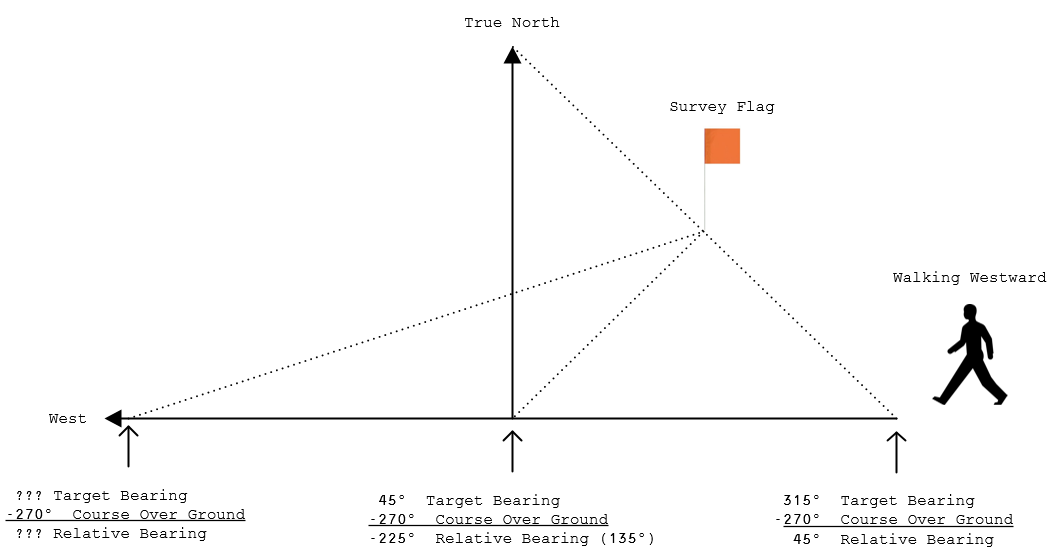
<http://www.rapidtables.com/convert/number/degrees-to-degrees-minutes-seconds.htm>

**NOTE**: The GPS Shield requires the SoftwareSerial library to be added to your project, which is provided in COASolutions. The GPS Shield has a switch that must be set to the "Soft Serial" position.

**NOTE**: The GPS Shield provides the "Course Over Ground" and "Magnetic Variation" parameters. True north and magnetic north do not provide exactly the same bearings. Magnetic north is what a compass provides and true north is what a GPS provides as "Course Over Ground". Your code will use the true north "Course Over Ground" bearing. However, if you added "Magnetic Variation" to the "Course Over Ground", you would obtain the magnetic north compass bearing.

**NOTE**: The GPS Shield has 3 LEDs. The Green **PWR** LED tells you that there is a good 5V power supply. If this isn't on, there's a serious problem with the power supply, and perhaps the battery has died or is dying. The Yellow **L13** (SD card access) LED is connected to digital pin 13, this is handy for telling when the Arduino is bootloading and will also flicker whenever the SD card is accessed. The Red **FIX** LED is connected to the GPS's fix output. When this is turning on/off once a second it **does not** have a fix. When it blinks once every 15 seconds, the GPS has a fix. The first time you use the GPS Shield, it may take up to 5 minutes to acquire a fix and provide a position.

Both the GPS device and calcBearing() function provides bearings relative to true north. Your NeoPixel will display your bearing relative to the selected target. If you are heading straight towards the selected target, the relative bearing displayed on your NeoPixel will be 0 degrees. The following diagram demonstrates relative bearings.



**NOTE**: A negative relative bearing can be adjusted to a positive relative bearing by adding 360 to it. For example, a relative bearing of -225⁰ is equivalent to a relative bearing of +135⁰ (-225⁰+360⁰=135⁰). Similarly, and bearing greater than or equal to 360⁰, can be adjusted by subtracting 360⁰. The relative bearing is what you want to display on your NeoPixel. A relative bearing of 0⁰ means you are heading directly towards your target.

It is recommended that you add a potentiometer or a photoresistor to your circuitry to control the brightness of the NeoPixel LEDs. Particularly if you use all 40 LEDs in your display. The 9V Rechargeable Battery has a limited amount of power it can deliver. If you are using all of the NeoPixel LEDs at full brightness, the battery life will be shortened and could stop powering your hardware in the middle of your GeoCache Hunt. A symptom of low battery voltage is that the NeoPixel LEDs will stop changing.

The getGPSMessage() function within the code provided for this project, returns a message once a second. Every time you receive a GPS message, and after parsing required data from the GPS message, you will write your current GPS Coordinates and distance to current target to the Secure Digital Card. You must write this data in the following format: "longitude,latitude,distance". Following is an example of a few lines of coordinate and distance data written to the SD card:

-81.303940,28.594528,894.576840

-81.303940,28.594526,893.881900

-81.303940,28.594526,893.881900

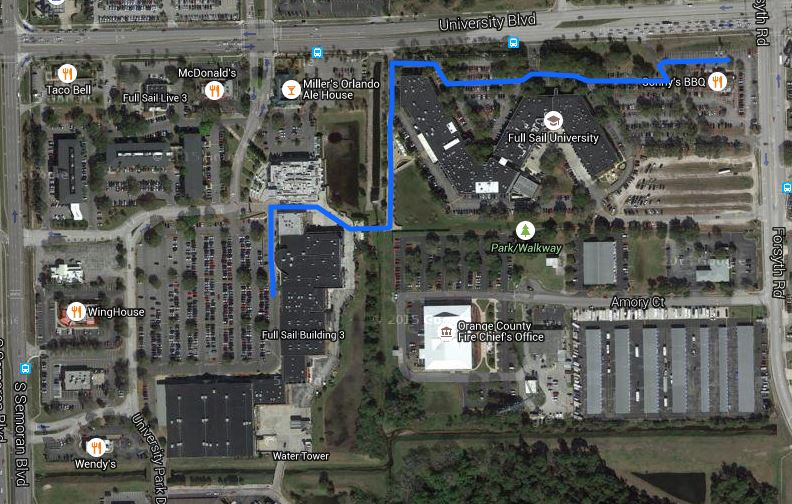
-81.303940,28.594526,893.881900

-81.303947,28.594530,895.155090

-81.303955,28.594524,892.959350

Be sure to write these parameters to the SD Card with 6 decimal places to the right of the decimal point (xxx.yyyyyy). You will create a sequentially number file name "MyFileNN.txt" on the SD Card each time the GeoCache program code is started, where NN is the file number beginning with "00" and maximum of "99". In other words, you will not append or rewrite an existing file, but rather create a new file each time the program code is started. Closing an opened file is not required. Upon returning to the classroom, you will cut and past the recorded information of your trip into the GeoCacheTeamX.kml file between the "coordinates> **insert data here** </coordinates" tags. This KML file is then imported to "[Google MyMaps](https://www.google.com/maps/d/u/0/)" to display your trip.

**EXAMPLE KML FILE (GeoCacheTeamX.kml)**



The assignment download includes a libLABLibrary.a object file and corresponding LABLibrary.h header file. The header defines functions within the library object for converting degrees/minutes to decimal degrees and for calculating distance and bearing between two coordinates. There is also a function to calculate a target coordinate given a starting coordinate, bearing and distance. You may add this library to test with and compare results of the functions you must write. You add the libLABLibrary.a library object to your project like other libraries, accept that you add the library by selecting the library file using the library browsing button.

**NOTE: This libLABLibrary.a is provided to you for testing purposes only. 30 points will be subtracted from your final project grade if you use this library during Final GeoCache Day execution.**

The following example code uses the calcCoordinate() function to generate a spiral path around the tree at GEOLAT0/GEOLON0 (see GeoCache.cpp provided) in KML corrdinate format. This function can be used to generate target coordinates given starting coordinates, bearing and distance. Calling calcBearing () and calcDistance() should then product the same bearing and distance.

float flat = 0.0F, flon = 0.0F, fbng = 0.0F, fdis = 0.0F;

while(fhed < 360\*6)

{

calcCoordinate(GEOLAT0, GEOLON0, &flat, &flon, fbng, fdis);

Serial.print(flon, 6);

Serial.print(",");

Serial.print(flat, 6);

Serial.print(",");

Serial.println(fdis, 6);

fbng += 10.0F;

fdis += 10.0F;

delay(100);

}

**EXAMPLE KML FILE (GeoCacheTreeSpiral.kml)**

****

No doubt, you will have to compile COASolutions in "**Release**" mode in order to optimize your data and program sized to fit in the available program and data spaces. When you build your project, the compiler will provide a data and program utilzation summary towards the bottom of your compilers build "Output" stream. It is recommend that you keep your program memory space utilization below 30720 bytes, and your data memory space utilization below 1536 bytes. If you go beyond this recommended range, then chances are your program stack and data memory may get corrupted, causing weird and unexplained program execution.

Following is an example of the compiler memory usage output summary:

Program Memory Usage : 29088 bytes 88.8 % Full

Data Memory Usage : 1240 bytes 60.5 % Full

**NOTE**: Make sure your Atmel Studio GeoCache, ArduinoCore and all other libraries in your project are set to "-Os" optimization in the tool chain for both the 'C' and 'C++' sections!

# RUBRIC (Total 100 Points)

|  |  |
| --- | --- |
| Points | Description |
| 35 | Accurately calculating target distance and bearing from current GPS position. |
| 30 | Effectively displaying on NeoPixel target distance and bearing from current GPS position. |
| 20 | Correctly storing current GPS position and target distance to SD Card and create KML file. |
| 15 | Captures all flags, display Google MyMap trip, provide brief description of project successes & difficulties. |

**NOTE**: Partial credit may be awarded for any rubric category above.

**NOTE:** If you do not delete the library from you project but instead use it to perform the GeoCache Hunt, then there will be an automatic 30 point reduction in grade.

**NOTE**: 10 points is deducted from your total grade if the compiler emits any warnings from your code. Compiler warnings emitted from libraries you do not write are excluded (eg: ArduinoCore, OneSheeld, NeoPixel, etc).

**NOTE:** If you use code acquired and/or modified from Internet, you must attribute the author and source of code within your own code, otherwise your code must be original! The Stanford MOSS (Measurement of Software Similarity) Program is used to verify authenticity of your code. Any violations will result in a zero grade and disciplinary action.

# SIDEKICK SUBMISSIONS

You must add your Team # and names of each team member to the top of the GeoCache.cpp source file. Place your MyMaps KML file within your Atmel Studio GeoCache project folder, then ZIP up your project folder and name the ZIP file as follows: "TeamX.GeoCache.zip", where "X" is your team number. All submissions are turned in at <https://edo.fullsail.edu/sidekick/>.

**NOTE**: Failure to submit your project with Team #, team member names and KML file will result in receiving a zero for your grade.

# COPYRIGHT & TRADEMARK NOTICES

AVR® and Atmel Studio® are registered trademarks of [Atmel Corporation](http://www.atmel.com/). Arduino® and ArduinoUNO® are registered trademarks of [Arduino LLC](http://www.arduino.cc/). RedBoard®, SparkFun® and SparkFun Electronics® are registered trademarks of [SparkFun Electronics](https://www.sparkfun.com/). NeoPixel® and Adafruit® are registered trademarks of [Adafruit Industries](https://www.adafruit.com/). 1Sheeld™ is a trademark of [Integreight, Inc](http://1sheeld.com/). Fritzing™ is a trademark of [Friends of Fritzing](http://friends.fritzing.org/about/). Portions of the “[SparkFun Inventors Kit Guide](https://cdn.sparkfun.com/datasheets/Kits/RedBoard_SIK_3.2.pdf)” has been incorporated and modified herein under the [Creative Commons BY-SA license](http://creativecommons.org/licenses/by-sa/3.0/), and with permission from [SparkFun Electronics](https://www.sparkfun.com/).