soft s

**GeoCache Hunt**

# OBJECTIVES

The primary objective of these labs are 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).

Be sure to review the following resources to assist in understanding the components utilized, circuit construction and coding techniques for this lab:

[Arduino Language Reference Guide](http://arduino.cc/en/Reference/HomePage)

[Arduino Library Reference Guide](https://www.arduino.cc/en/Reference/Libraries)

[SparkFun SIK Guide & Code](https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/all)

The key to successfully completing any assignment is to thoroughly review the instructions in order to understand what is to be done conceptually, and then begin constructing the circuit and writing the code.

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

# INSTRUCTIONS

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. The teams will then leave to locate and capture 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. 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.

**NOTE:** You are to create a new Visual Studio Solution/Project for each lab assignment. You do this by opening Visual Studio, click on “vMicro->New Arduino Project”, enter the lab project name, and then click on “OK”.

Now proceed by assembling the circuit and following the coding instructions.

## HARDWARE IMAGE

## 

The above image demonstrates the configuration used during execution of the actual GeoCache Hunt. The NeoPixel must be plugged into the breadboard and only GND, +5V and pin 6 connected.

**NOTE**: The pins of the button on the left side are actually in the gully between the left and right side of the breadboard.

## 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 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" char string message delivered by GPS board.

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.

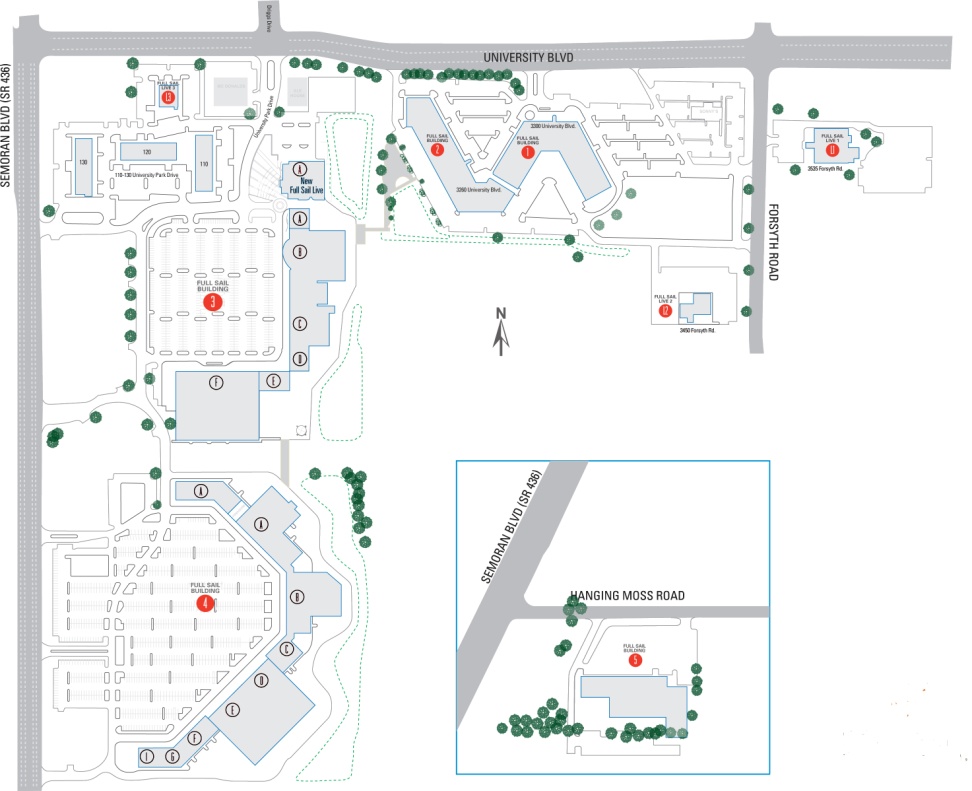
**NOTE**: You are not permitted to use the Sparkfun Kit LCD or SERVO components in this project.

The GPRMC char 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.

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. Use LEDs on the NeoPixel to indicate which one of four flags is selected (0, 1, 2 or 3).

**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.

All flags will be located on Full Sail Campus, 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!!! **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, 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.DDDDDD") for use in calculating bearing and distance to the target. You must create a function to do this. The GPS Shield provides N/S and E/W parameters to indicate the corresponding coordinate’s hemisphere (N/S or E/W). Once you have converted Degrees Minutes to Decimal Degrees, you must negate the resulting coordinate (coordinate \* -1.0) if its corresponding indicator is S or W hemisphere.

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 great-circle distance in feet and great-circle 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.

All flags are outdoors and located on Full Sail Campus, which can reach up to 2500 feet across diagonally. Depending on the angle in which you approach the flag, you may need to be standing within a few feet of the flag to see it visually. You DO NOT cross any major roads such as University Boulevard, Semoran Boulevard or Forsyth Road, or enter any dense forests, or wade through any ponds or streams!! There may be more than one flag located at the target position. Pick up only ONE flag at the target position. Your NeoPixel should display distances of 1 to 2500 feet.

You can find distance and bearing calculations by researching at the Full Sail library or performing Google searches containing combinations of the following key words: "Arduino GPS Robot RMC Waypoint Distance Bearing". You may also find help searching books on Safari using the same key words. The 3rd link below provides **JavaScript** examples that can be similarly coded in ‘C’ for calculating the great-circle distance and target bearing.

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

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

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

<https://en.wikipedia.org/wiki/Geographic_coordinate_system>

<https://mynasadata.larc.nasa.gov/latitudelongitude-finder/>

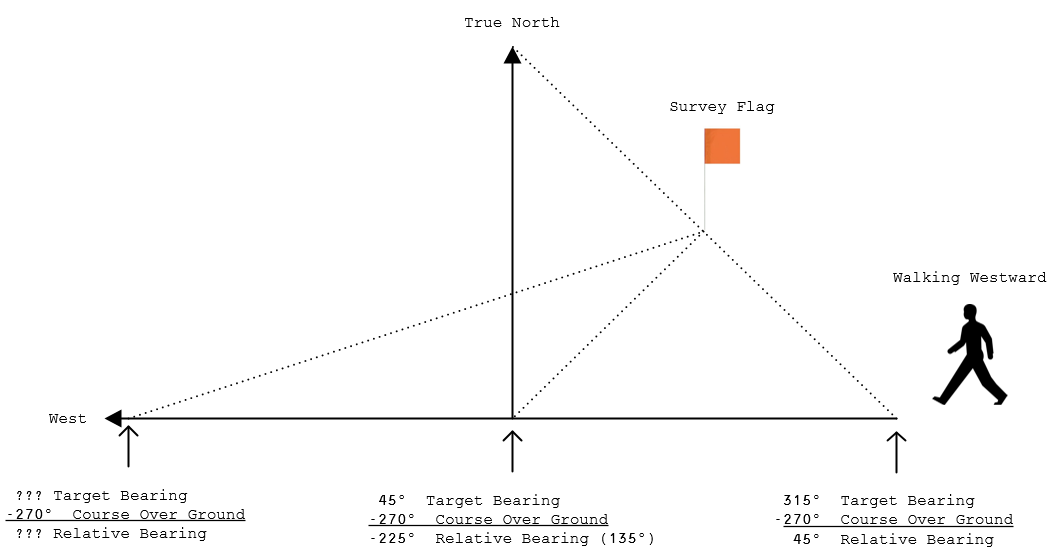
**NOTE**: The GPS Shield requires the Visual Micro built-in “SoftwareSerial” library to be added to your project. The GPS Shield has a switch that must be set to the "Soft Serial" position.

**NOTE**: There are macros provided in Arduino.h to convert degrees to radians and radians to degrees.

The GPS Shield provides the "Course Over Ground" and "Magnetic Variation" parameters. True north and magnetic north are not exactly the same. Magnetic North is what a magnetic compass provides. True North is what the GPS provides, and is what you will use in your code.

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 in reference to true north. Your NeoPixel will display your Target Relative Bearing towards the selected target. If you are heading straight towards the selected target, the Target Relative Bearing displayed on your NeoPixel will be 0 degrees. Note that the atan() used to calculate target bearing returns values between -180 to 180 relative to true north.



A negative Target Relative Bearing can be adjusted to a positive Target Relative Bearing by adding 360 to it. For example, a Relative Target Bearing of -225⁰ is equivalent to a Relative Target Bearing of +135⁰ (-225⁰+360⁰=135⁰). Similarly, and Relative Target Bearing greater than or equal to 360⁰, can be adjusted by subtracting 360⁰. The Relative Target Bearing calculation results in the range 0 >= N° < 360, and is what is displayed on your NeoPixel. A Relative Target Bearing of 0⁰ means you are heading directly towards your target, 90°means turn right, 270° turn left, 180°turn around!

It is recommended that you add a potentiometer or a photo resistor to your circuitry to control the brightness of the NeoPixel LEDs. This is particularly true 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 a line once a second to the SD card containing the received GPS coordinates and calculated target distance in feet using the following format: "longitude,latitude,bearing.distance”. Note that the coordinates are written in decimal degrees format. Following is an example of a few lines of data written to the SD card:

-81.343254,28.781599,8.370

-81.343254,28.781591,16.367

-81.343254,28.781572,15.361

-81.343254,28.781555,348.354

-81.343254,28.781544,353.350

-81.343254,28.781530,355.346

-81.343254,28.781527,27.344

-81.343254,28.781519,68.341

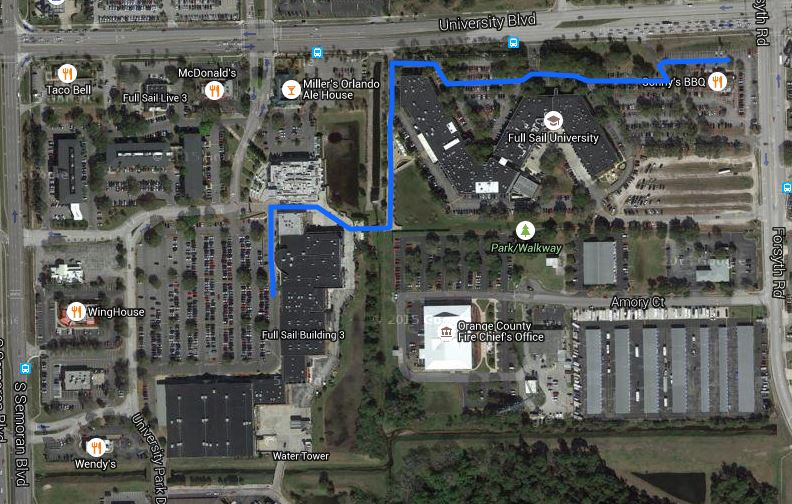
-81.343239,28.781519,80.341

Be sure to write the longitude and latitude float parameters to the SD Card with 6 decimal places to the right of the decimal point (xxx.yyyyyy). The third parameter is formated like a float, but is actually a concatenation of the integer values of Target Relative Bearing and Target Distance displayed on NeoPixel. The value to the left of the decimal point is the integer value of the Target Relative Bearing, and the value to the right of the decimal point is the integer value of the Target Distance.

You will create a sequentially numbered file name "MyFileNN.txt" on the SD Card each time the GeoCache program code is started, where N is the file number beginning with "0" 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. Once the correct new file has been opened, you keep it open for writing. You never close the file in your program. Upon returning to the classroom, you will cut and paste recorded file information of your trip into the GeoCacheTeamX.kml file between the coordinate tags ("<coordinates> **insert data here** </coordinates>"). This KML file is then imported to "[Google MyMaps](https://www.google.com/maps/d/u/0/)" to display your trip.

**NOTE**: A comment in KML files are denoted as follows: **<!-- this is a comment -->**

**EXAMPLE of KML file on Google My Maps (GeoCacheTeamX.kml):**



No doubt, you will have to compile your project in "**Release**" mode in order to optimize your data and program size 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 program memory space utilization below 94%, and your data memory space utilization below 75%. If you go beyond these recommended ranges, you run the risk of the program stack and data memory getting corrupted, causing weird and unexplained program execution. The compiler “Minimum Memory Usage” ONLY INCLUDES global data. It does not include data allocated dynamically on the stack or within the heap.

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

Program size: 24,170 bytes (used 75% of a 32,256 byte maximum) (12.24 secs)

Minimum Memory Usage: 1372 bytes (67% of a 2048 byte maximum)

**NOTE:** An Internet search for “ARDUINO PROGMEM” will provide methods for storing and reading “const” arrays/variables into the 32k Program Memory Space in order to reduce usage in the 2k Program Data Space.

**WARNING**: Using the String Class is **NOT RECOMMENDED**, since it allocates/deallocates memory from program data memory heap leaving memory fragmented and not usable.

Data Memory can be easily consumed with the GeoCache project, and will require conservative allocation of global and local variables. The String class allocates data in the heap between the global and stack of the 2048 bytes of data memory. You have no view of the stack or heap allocation during compile time, since they are used dynamically. The heap is very inefficient, quickly getting defragmented and consumed, eventually leaving you with no memory to allocate String class data. In addition, the compiler does not provide an estimate of stack or heap data usage, leaving the possibility of stack, heap or global data corruption during run time. Consider using strtok(), sprintf(), strtod(), dtostrf(), modf() and fmod(), or using the Serial or SD print() or println() overload functions. Click [this link](https://learn.adafruit.com/memories-of-an-arduino) to learn how to identify memory problems and to most effectively utilize memory.

**NOTE:** char strings must be NULL terminated!

When working with char string arrays, make sure you allocate an array with room for a NULL terminator at the end. “Hello World” needs a char array of 12 bytes, since the last byte in the array must be ZERO (same as ‘\0’). If you don’t do this, then strange things will happen!

**NOTE**: Do not mix floats and integers in calculations, or the fractional component may truncate to zero (3/2 = 1 (using integers), 3.0/2.0 = 1.5 (using floats)).

The GeoCache project is a team development effort, and will require sharing code amongst the team members. Since VisualMicro stores absolute file paths within the solution file, sharing the solution with the team may not work on their machine. To avoid this problem, it is recommended that the source code (\*.ino file) be shared instead.

# RUBRIC (Total 100 Points)

|  |  |
| --- | --- |
| Points | Description |
| 10 | Converting Degree Minutes to Decimal Degrees format. |
| 15 | Calculating target distance in feet from current GPS position. |
| 15 | Calculating relative target bearing from current GPS course over ground. |
| 10 | Displaying target distance in feet from current GPS position. |
| 10 | Displaying relative target bearing in degrees on NeoPixel. |
| 5 | Selecting and displaying target index on NeoPixel. |
| 10 | Correctly storing current GPS position, target direction/distance to SD Card file(s), and creating Google MyMaps trip. |
| 15 | Capturing correct flags, giving brief team presentation, submitting KML file with project that has no compiler warnings (warnings emitted from Arduino libraries are excluded). |
| 10 | Course Directors evaluation of team cooperation, effectiveness and on schedule. Simply speaking, if your project is not ready by start of finals day, anticipate loosing up to 10 points. |

# SIDEKICK SUBMISSIONS

Your Visual Micro “Solution” folder must be zipped up and submitted (turned-in) to SideKick within the date/time window defined on SideKick. Do not delete anything from the solution folder. Labs do not require being checked off by the Lab Specialist or Course Director before submission. However, if you complete your lab before the lab period ends, then you may request check-off and leave early. Labs submitted late (after the date/time window defined on SideKick) will receive a zero grade, unless otherwise instructed by the course director.

# COPYRIGHT & TRADEMARK NOTICES

* Arduino® and Arduino UNO® are registered trademarks of [Arduino LLC](http://www.arduino.cc/).
* Visual Micro® is a registered trademark of [Visual Micro Limited](http://www.visualmicro.com/page/Arduino-Visual-Studio-Downloads.aspx).
* Visual Studio® is a registered trademark of [Microsoft Corporation](https://www.microsoft.com/en-us/).
* Atmel® and AVR® are registered trademarks of [Atmel Corporation](http://www.atmel.com/).
* 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/).
* 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.