Citi Bike Helmet

Created by Becky Stern

https://learn.adafruit.com/citi-bike-helmet

Last updated on 2024-06-03 01:20:17 PM EDT

Table of Contents

Overview

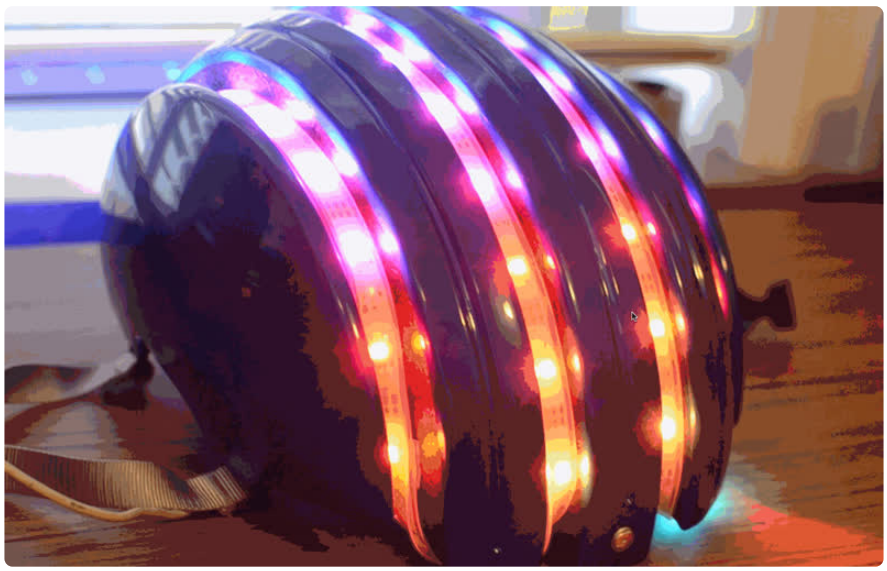
Improve your visibility with style! Mod up this bike helmet with LED strip, a FLORA GPS, and find your way to the nearest Citi Bike station with ease. We used a Carrera foldable helmet, which has grooves between the protection, perfect for NeoPixel strip.

Before you begin this project, we recommend reading the following guides:

- Getting Started with FLORA (https://adafru.it/aSZ)
- FLORA GPS (https://adafru.it/aRP)
- FLORA Accelerometer/compass (https://adafru.it/aYS)

Do not affix electronics to the outer smooth surface of your bike helmet! Helmets are designed to be smooth for your safety.

This project was created in collaboration with Tyler Cooper & Justin Cooper, with major video help from Risa Rose and JM Imbrescia.



Tools & Supplies

For this project, gather up:

- Carrera foldable helmet (https://adafru.it/cgp) (we had to order ours from the UK (https://adafru.it/cgs))
- NeoPixel strip (http://adafru.it/1376) (about 1.5m)
- FLORA main board (http://adafru.it/659)
- FLORA LSM303 accelerometer/compass (http://adafru.it/1247)
- FLORA Ultimate GPS (http://adafru.it/1059)
- Rechargeable lithium polymer battery (http://adafru.it/258) and charger (http://adafru.it/1304)
- coincell battery (http://adafru.it/654) and holder (http://adafru.it/653) (optional to improve GPS fix time)
- Needle and clear thread
- Scissors

Any entry level 'all-in-one' soldering iron that you might find at your local hardware store should work. As with most things in life, you get what you pay for.
Upgrading to a higher end soldering iron setup, like the Hakko FX-888 that we stock in our store (http://adafru.it/180), will make soldering fun and easy.

Do not use a "ColdHeat" soldering iron! They are not suitable for delicate electronics work and can damage the Flora (see here (https://adafru.it/aOo)).

Click here to buy our entry level adjustable 30W 110V soldering iron. (http://adafru.it/180)

Click here to upgrade to a Genuine Hakko FX-888 adjustable temperature soldering iron. (http://adafru.it/303)

Learn how to solder with tons of tutorials! (https://adafru.it/aTk)

You will want rosin core, 60/40 solder. Good solder is a good thing. Bad solder leads to bridging and cold solder joints which can be tough to find.

Click here to buy a spool of leaded solder (recommended for beginners). (http://adafru.it/145)

Click here to buy a spool of lead-free solder. (http://adafru.it/734)

You will need a good quality basic multimeter that can measure voltage and continuity.

Click here to buy a basic multimeter. (http://adafru.it/71)

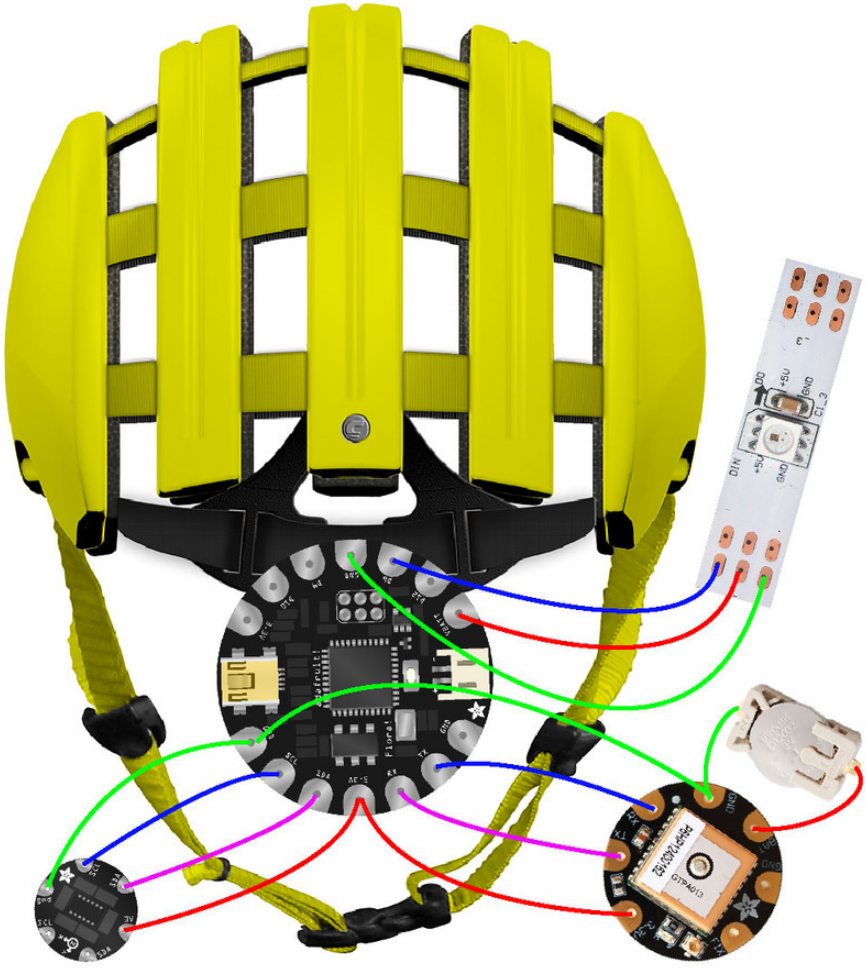
Click here to buy a top of the line multimeter. (http://adafru.it/308)

Click here to buy a pocket multimeter. (http://adafru.it/850)

Don't forget to learn how to use your multimeter too! (https://adafru.it/aOy)

Don't forget your wire strippers (http://adafru.it/527), pliers (http://adafru.it/146), and flush snips (http://adafru.it/152)!

Wiring Diagram

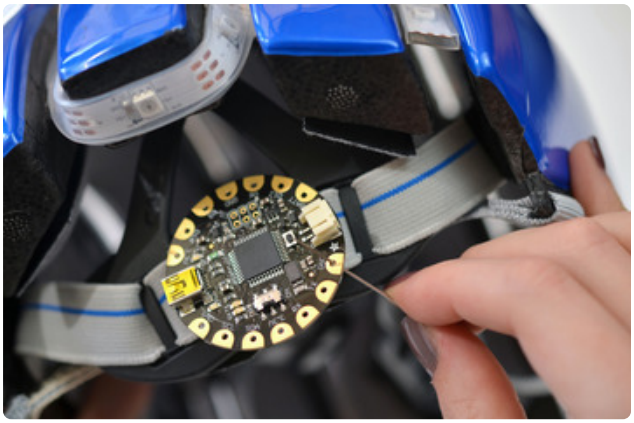


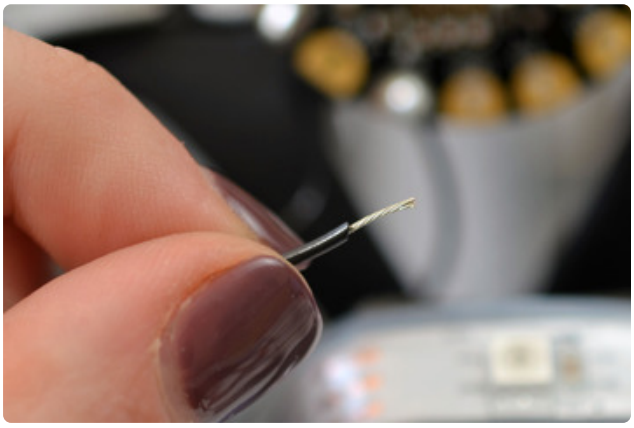
Build Circuit

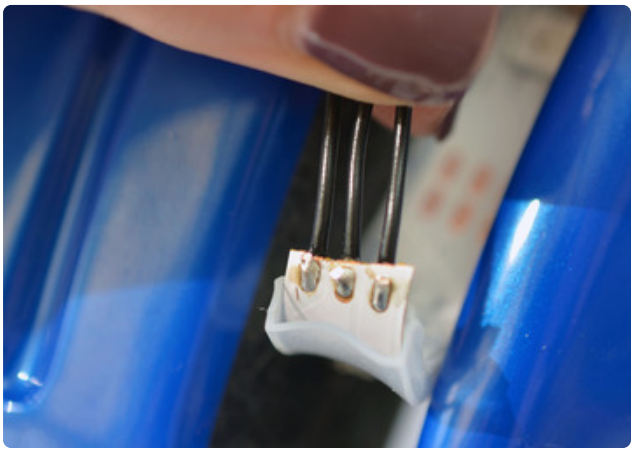
Wrap a long piece of NeoPixel strip throughout the grooves in the Carrera "foldable" helmet (https://adafru.it/cgp). Start at one back edge, shifting the strip so that four LEDs are visible to the rider along the front brim.

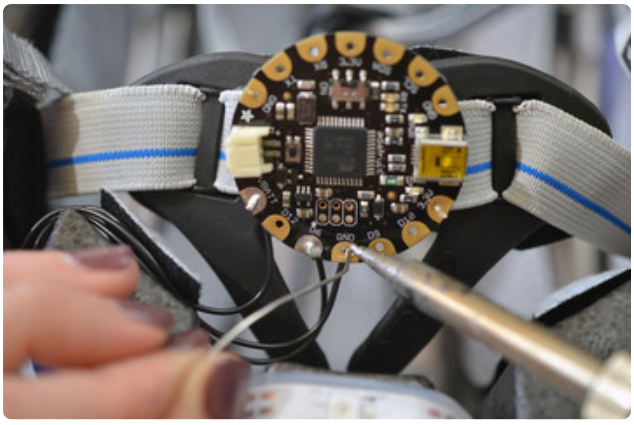
Cut the strip to length at the opposite back edge and tack the LED strip to the nylon webbing in the grooves of the helmet with a needle and clear thread. We didn't affix to every piece of webbing, just in enough spots to secure the LED strip.

Do not affix electronics to the outer smooth surface of your bike helmet! Helmets are designed to be smooth for your safety.





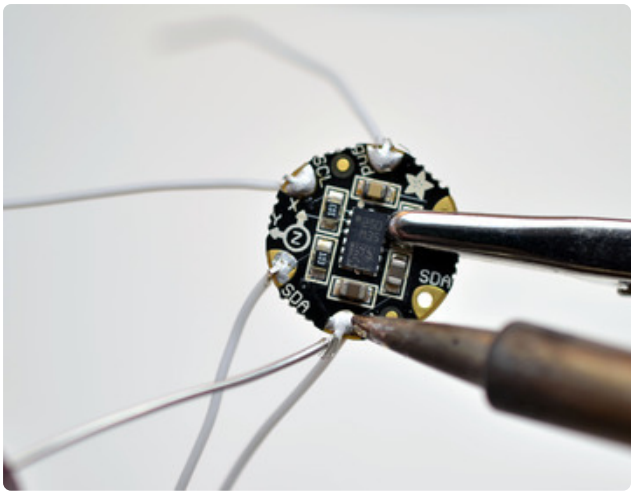


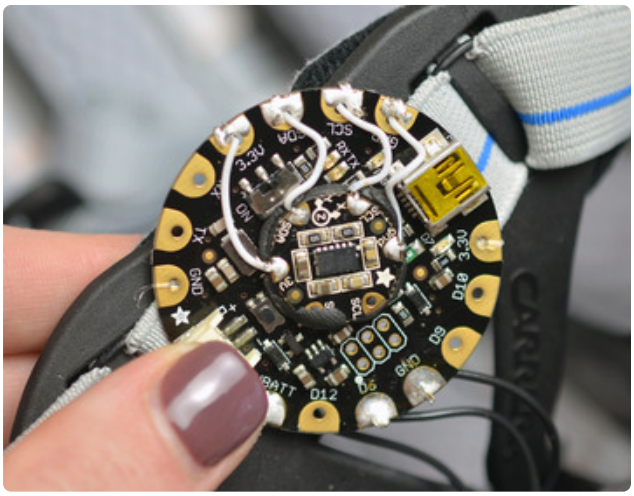


Using a needle and more clear thread, affix the FLORA main board to the back head brace of the helmet through two unused pins (we used the 3.3v pad next to the USB port and the GND pad next to the JST connector).

Strip and tin three wires, then solder them to the input side of the LED strip.

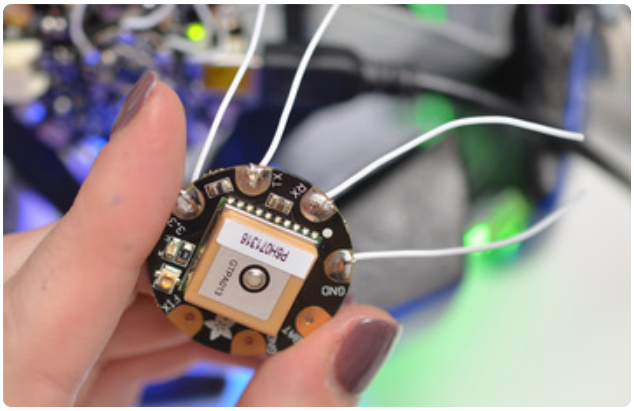
Cut to length, strip and solder these wires to VBATT, D6, and GND, referring to the wiring diagram on the previous page if necessary.

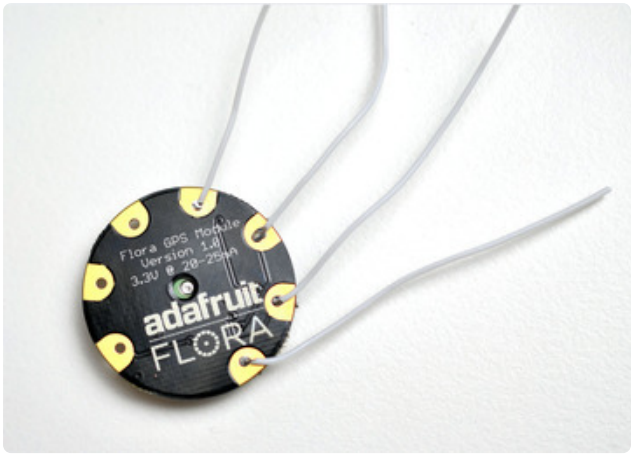


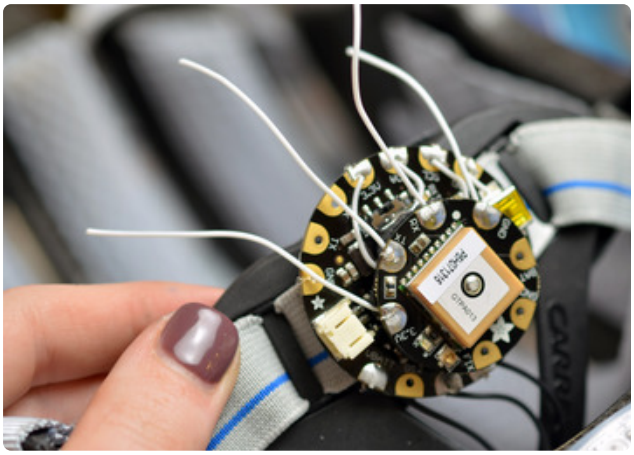


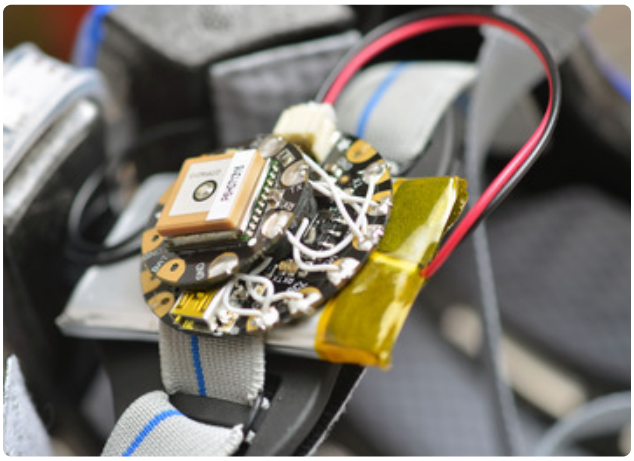
Solder small flexible wires to the LSM303 accelerometer/compass module, then apply a small piece of Sugru to cover the back of the module. Stick it in the middle of the FLORA main board, making sure not to cover the on/off switch or reset button. We are making a circuit sandwich!

Solder the wires to 3.3v, SCL, SDA, and GND on FLORA, referring again to the wiring diagram.









Repeat the process above with the GPS module, connecting corresponding TX/RX pads to FLORA (remember that TX goes to RX and RX goes to TX).

Sugru insulates the back of the GPS module from the LSM303 and FLORA main board, and also provides a semipermanent sticky situation. The silicone is not quite an adhesive, though it will remain affixed unless you choose to carefully peel it off.

The lithium polymer battery slides behind the elastic of the head brace. We sewed a small fabric pouch around the battery, which in turn was stitched to the elastic.

Program it

The Citi Bike Helmet code takes the FLORA to the limit. The code is used to drive LEDs, the FLORA GPS module, and the FLORA compass/accelerometer module. It looks at a long list of all Citi bike sharing stations in NYC (over 300!) and determines which coordinate is closest to you. It then uses the GPS module, and the compass module to navigate you there. The code is located in this GitHub repo (https://adafru.it/19By). Click the button below to download the code.

Download the Citi Bike Helmet Arduino Sketch

https://adafru.it/19Bz

This code example requires the following Arduino libraries, located in the Arduino Library Manager and in GitHub as follows:

- Adafruit GPS library which can be downloaded from https://github.com/adafruit/Adafruit-GPS-Library (https://adafru.it/aMm)
- Adafruit NeoPixel library which can be downloaded from https://github.com/adafruit/Adafruit_NeoPixel (https://adafru.it/aZU)
- Pololu LSM303 library which can be downloaded from https://github.com/pololu/LSM303 (https://adafru.it/cgq)

Learn how to install libraries here (https://adafru.it/aYM).

```
// SPDX-FileCopyrightText: 2019 Becky Stern for Adafruit Industries
// SPDX-FileCopyrightText: 2019 Justin Cooper for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// Code for the Adafruit FLora Citi Bike Helmet Tutorial
// https://learn.adafruit.com/citi-bike-helmet
// Becky Stern and Justin Cooper, Adafruit.com
//
// This code shows how to listen to the GPS module in an interrupt
// which allows the program to have more 'freedom' - just parse
// when a new NMEA sentence is available! Then access data when
// desired.
//
// Tested and works great with the Adafruit Flora GPS module
//    ------> https://adafruit.com/products/1059
// Pick one up today at the Adafruit electronics shop
// and help support open source hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <LSM303.h>
#include <Adafruit_NeoPixel.h>

LSM303 compass;
```

```
#define LAT_LON_SIZE 311

const float lat_lon[LAT_LON_SIZE][2] PROGMEM = {
  {40.767272, -73.993928},
  {40.719115, -74.006666},
  {40.711174, -74.000165},
  {40.683826, -73.976323},
  {40.741776, -74.001497},
  {40.696089, -73.978034},
  {40.686767, -73.959281},
  {40.731724, -74.006744},
  {40.727102, -74.002970},
  {40.714255, -73.981308},
  {40.692395, -73.993379},
  {40.698398, -73.980689},
  {40.716250, -74.009105},
  {40.715421, -74.011219},
  {40.720873, -73.980857},
  {40.721815, -73.997203},
  {40.714739, -74.009106},
  {40.690892, -73.996123},
  {40.729170, -73.998102},
  {40.753230, -73.970325},
  {40.748900, -73.976048},
  {40.739713, -73.994564},
  {40.760646, -73.984426},
  {40.738176, -73.977386},
  {40.709056, -74.010433},
  {40.743349, -74.006817},
  {40.700378, -73.995480},
  {40.702771, -73.993836},
  {40.690284, -73.987071},
  {40.737815, -73.999946},
  {40.711463, -74.005524},
  {40.741951, -74.008030},
  {40.727434, -73.993790},
  {40.695976, -73.990148},
  {40.692462, -73.989639},
  {40.728418, -73.987139},
  {40.730473, -73.986723},
  {40.736196, -74.008592},
  {40.691965, -73.981301},
  {40.689810, -73.974931},
  {40.697883, -73.973503},
  {40.688663, -73.980518},
  {40.691960, -73.965368},
  {40.693270, -73.977038},
  {40.735353, -74.004830},
  {40.721853, -74.007717},
  {40.718709, -74.009000},
  {40.724560, -73.995652},
  {40.723179, -73.994800},
  {40.732263, -73.998522},
  {40.735439, -73.994539},
  {40.735324, -73.998004},
  {40.719392, -74.002472},
  {40.689407, -73.968854},
  {40.701221, -74.012342},
  {40.703651, -74.011677},
  {40.694748, -73.983624},
  {40.691782, -73.973729},
  {40.717290, -73.996375},
  {40.707064, -74.007318},
  {40.722293, -73.991475},
  {40.723683, -73.975748},
  {40.750977, -73.987654},
  {40.719105, -73.999733},
  {40.693082, -73.971789},
```

```
  {40.685281, -73.978058},
  {40.686918, -73.976682},
  {40.686500, -73.965633},
  {40.717487, -74.010455},
  {40.697665, -73.984764},
  {40.699869, -73.982719},
  {40.733319, -73.995101},
  {40.708272, -73.968341},
  {40.734545, -73.990741},
  {40.684568, -73.958810},
  {40.760202, -73.964784},
  {40.713452, -73.983985},
  {40.730286, -73.990764},
  {40.730493, -73.995721},
  {40.714066, -73.992939},
  {40.714130, -73.997046},
  {40.686832, -73.979677},
  {40.728984, -73.990518},
  {40.722174, -73.983687},
  {40.720828, -73.977931},
  {40.723627, -73.999496},
  {40.704633, -74.013617},
  {40.760957, -73.967244},
  {40.708235, -74.005300},
  {40.714274, -73.989900},
  {40.713079, -73.998511},
  {40.714978, -74.013012},
  {40.689269, -73.989128},
  {40.717227, -73.988020},
  {40.722632, -73.988873},
  {40.696102, -73.967510},
  {40.694246, -73.992159},
  {40.703553, -74.006702},
  {40.709559, -74.006536},
  {40.724537, -73.981854},
  {40.753201, -73.977987},
  {40.713361, -74.009376},
  {40.717439, -74.005834},
  {40.699917, -73.989717},
  {40.696192, -73.991218},
  {40.692361, -73.986317},
  {40.689888, -73.981013},
  {40.736245, -73.984737},
  {40.729538, -73.984267},
  {40.715337, -74.016583},
  {40.724055, -74.009659},
  {40.720434, -74.010206},
  {40.714504, -74.005627},
  {40.711731, -73.991930},
  {40.712199, -73.979481},
  {40.742387, -73.997262},
  {40.729039, -73.994046},
  {40.730477, -73.999060},
  {40.703799, -74.008386},
  {40.725806, -73.974224},
  {40.712690, -73.987763},
  {40.717821, -73.976289},
  {40.717399, -73.980165},
  {40.697940, -73.969868},
  {40.685144, -73.953809},
  {40.736494, -73.997043},
  {40.736528, -74.006180},
  {40.728738, -74.007488},
  {40.724909, -74.001547},
  {40.718502, -73.983298},
  {40.715595, -73.987029},
  {40.705309, -74.006125},
  {40.763406, -73.977224},
  {40.685395, -73.974314},
```

```
  {40.693631, -73.962235},
  {40.716021, -73.999743},
  {40.716226, -73.982612},
  {40.732617, -73.991580},
  {40.732915, -74.007113},
  {40.755102, -73.974986},
  {40.707179, -74.008873},
  {40.716058, -73.991907},
  {40.751726, -73.987535},
  {40.708346, -74.017134},
  {40.689004, -73.960238},
  {40.682231, -73.961458},
  {40.693261, -73.968896},
  {40.758280, -73.970694},
  {40.730385, -74.002149},
  {40.732241, -74.000263},
  {40.694528, -73.958089},
  {40.693317, -73.953819},
  {40.726794, -73.996950},
  {40.708621, -74.007221},
  {40.722437, -74.005664},
  {40.734011, -74.002938},
  {40.734926, -73.992005},
  {40.736251, -74.000836},
  {40.683178, -73.965964},
  {40.714948, -74.002344},
  {40.712732, -74.004607},
  {40.710445, -73.965250},
  {40.692215, -73.984284},
  {40.697601, -73.993445},
  {40.695078, -73.987247},
  {40.722992, -73.979954},
  {40.725213, -73.977687},
  {40.688070, -73.984106},
  {40.680342, -73.955768},
  {40.684157, -73.969222},
  {40.691651, -73.999978},
  {40.688515, -73.964762},
  {40.719260, -73.981780},
  {40.720195, -73.989978},
  {40.740343, -73.989551},
  {40.725028, -73.990696},
  {40.740582, -74.005508},
  {40.739323, -74.008119},
  {40.695128, -73.995950},
  {40.700101, -73.991043},
  {40.710762, -73.994003},
  {40.720664, -73.985179},
  {40.722280, -73.976687},
  {40.715815, -73.994223},
  {40.702818, -73.987657},
  {40.704717, -74.009260},
  {40.687534, -73.972651},
  {40.712912, -74.010202},
  {40.702240, -73.982578},
  {40.695807, -73.973555},
  {40.687644, -73.969689},
  {40.695733, -73.971296},
  {40.770513, -73.988038},
  {40.765849, -73.986905},
  {40.717548, -74.013220},
  {40.702515, -74.014270},
  {40.724677, -73.987834},
  {40.701485, -73.986569},
  {40.688646, -73.982634},
  {40.726217, -73.983798},
  {40.729553, -73.980572},
  {40.743174, -74.003664},
  {40.741739, -73.994155},
```

```
  {40.682165, -73.953990},
  {40.680983, -73.950047},
  {40.727791, -73.985649},
  {40.726280, -73.989780},
  {40.752554, -73.972826},
  {40.756019, -73.967446},
  {40.746841, -73.994458},
  {40.708530, -73.964089},
  {40.742354, -73.989150},
  {40.727407, -73.981420},
  {40.744876, -73.995298},
  {40.763707, -73.985161},
  {40.756603, -73.997900},
  {40.764618, -73.987894},
  {40.762272, -73.987882},
  {40.744751, -73.999153},
  {40.754557, -73.965929},
  {40.750019, -73.969053},
  {40.759710, -73.974023},
  {40.766953, -73.981693},
  {40.748061, -74.007231},
  {40.745227, -74.007979},
  {40.712858, -73.965902},
  {40.735876, -73.982050},
  {40.746919, -74.004518},
  {40.742065, -74.004431},
  {40.759345, -73.967596},
  {40.755135, -73.986580},
  {40.743954, -73.991448},
  {40.683124, -73.978951},
  {40.765265, -73.981923},
  {40.763440, -73.982681},
  {40.743453, -74.000040},
  {40.712868, -73.956981},
  {40.745712, -73.981948},
  {40.721100, -73.991925},
  {40.745167, -73.986830},
  {40.735242, -73.987585},
  {40.743943, -73.979660},
  {40.756405, -73.990026},
  {40.760300, -73.998842},
  {40.760192, -73.991255},
  {40.766696, -73.990617},
  {40.712604, -73.962644},
  {40.739355, -73.999317},
  {40.732232, -73.988899},
  {40.755002, -73.980144},
  {40.750380, -73.983389},
  {40.746200, -73.988557},
  {40.733142, -73.975738},
  {40.756458, -73.993722},
  {40.750663, -74.001768},
  {40.751575, -73.994190},
  {40.740963, -73.986022},
  {40.750199, -73.990930},
  {40.756800, -73.982911},
  {40.747348, -73.997235},
  {40.762698, -73.993012},
  {40.737261, -73.992389},
  {40.737049, -73.990092},
  {40.748548, -73.988084},
  {40.762288, -73.983361},
  {40.744219, -73.971212},
  {40.738274, -73.987519},
  {40.732218, -73.981655},
  {40.749012, -73.988483},
  {40.739126, -73.979737},
  {40.763413, -73.996674},
  {40.745497, -74.001971},
```

```
  {40.760659, -73.980420},
  {40.729386, -73.977724},
  {40.750072, -73.998392},
  {40.768254, -73.988639},
  {40.760875, -74.002776},
  {40.760094, -73.994618},
  {40.752068, -73.967843},
  {40.751492, -73.977988},
  {40.747803, -73.973441},
  {40.751884, -73.977701},
  {40.759922, -73.976485},
  {40.750449, -73.994810},
  {40.757147, -73.972078},
  {40.754665, -73.991381},
  {40.755273, -73.983169},
  {40.755941, -74.002116},
  {40.747659, -73.984907},
  {40.743155, -73.974347},
  {40.742909, -73.977060},
  {40.757569, -73.990985},
  {40.771522, -73.990541},
  {40.718939, -73.992662},
  {40.710451, -73.960876},
  {40.752996, -73.987216},
  {40.702550, -74.012723},
  {40.741443, -73.975360},
  {40.740258, -73.984092},
  {40.757952, -73.977876},
  {40.715348, -73.960241},
  {40.741472, -73.983209},
  {40.736502, -73.978094},
  {40.744449, -73.983035},
  {40.702550, -73.989402},
  {40.698920, -73.973329},
  {40.716887, -73.963198},
  {40.734160, -73.980242},
  {40.725500, -74.004451},
  {40.705311, -73.971000},
  {40.765909, -73.976341}
};

// Connect the GPS Power pin to 5V
// Connect the GPS Ground pin to ground
// If using software serial (sketch example default):
//   Connect the GPS TX (transmit) pin to Digital 8
//   Connect the GPS RX (receive) pin to Digital 7
// If using hardware serial:
//   Connect the GPS TX (transmit) pin to Arduino RX1 (Digital 0)
//   Connect the GPS RX (receive) pin to matching TX1 (Digital 1)

// If using software serial, keep these lines enabled
// (you can change the pin numbers to match your wiring):
//SoftwareSerial mySerial(8, 7);
//Adafruit_GPS GPS(&mySerial);

// If using hardware serial, comment
// out the above two lines and enable these two lines instead:
Adafruit_GPS GPS(&Serial1);
HardwareSerial mySerial = Serial1;

// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences
#define GPSECHO false

Adafruit_NeoPixel strip = Adafruit_NeoPixel(45, 6, NEO_GRB + NEO_KHZ800);
int FarRight = 9;
int CenterRight = 10;
int CenterLeft = 34;
int FarLeft = 35;
```

```
int HeadsUp[] = {35, 34, 10, 9};
int RightStrip[] = {8, 7, 6, 5, 4, 3, 2, 1, 0};
int RightCenterStrip[] = {11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21};
int LeftCenterStrip[] = {33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23};
int LeftStrip[] = {36, 37, 38, 39, 40, 41, 42, 43, 44};
int counter = 0;

// this keeps track of whether we're using the interrupt
// off by default!
boolean usingInterrupt = false;

void setup()
{
  // connect at 115200 so we can read the GPS fast enough and echo without dropping
chars
  // also spit it out
  Serial.begin(115200);
  delay(5000);

  Serial.println("initialize");
  //test to find the closest location from the list, just plug in a lat lon from
the above Array
  //Serial.print("Closest Test Loc: ");
  //Serial.println(find_closest_location(40.726378, -74.005437));

  Wire.begin();
  // 9600 NMEA is the default baud rate for Adafruit MTK GPS's- some use 4800
  GPS.begin(9600);

  // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data)
including altitude
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  // uncomment this line to turn on only the "minimum recommended" data
  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  // For parsing data, we don't suggest using anything but either RMC only or
RMC+GGA since
  // the parser doesn't care about other sentences at this time

  // Set the update rate
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz update rate
  // For the parsing code to work nicely and have time to sort thru the data, and
  // print it out we don't suggest using anything higher than 1 Hz
  compass.init();
  compass.enableDefault();
  // Calibration values. Use the Calibrate example program to get the values for
  // your compass.
  compass.m_min.x = -581; compass.m_min.y = -731; compass.m_min.z = -1097;
  compass.m_max.x = +615; compass.m_max.y = +470; compass.m_max.z = 505;
  delay(1000);
  // Ask for firmware version
  Serial1.println(PMTK_Q_RELEASE);

  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}



uint32_t timer = millis();
void loop()                     // run over and over again
{
  // read data from the GPS in the 'main loop'
  char c = GPS.read();
  // if you want to debug, this is a good time to do it!
  if (GPSECHO)
      if (c) Serial.print(c);

  // if a sentence is received, we can check the checksum, parse it...
  if (GPS.newNMEAreceived()) {
```

```
    // a tricky thing here is if we print the NMEA sentence, or data
    // we end up not listening and catching other sentences!
    // so be very wary if using OUTPUT_ALLDATA and trytng to print out data
    //Serial.println(GPS.lastNMEA());   // this also sets the newNMEAreceived()
flag to false

    if (!GPS.parse(GPS.lastNMEA()))   // this also sets the newNMEAreceived() flag
to false
      return;  // we can fail to parse a sentence in which case we should just wait
for another
  }

  // if millis() or timer wraps around, we'll just reset it
  if (timer > millis())  timer = millis();

  // every 300 milliseconds, update the heading/distance indicators
  if (millis() - timer > 300) {
    timer = millis(); // reset the timer

    //Serial.print("\nTime: ");
    //Serial.print(GPS.hour, DEC); Serial.print(':');
    //Serial.print(GPS.minute, DEC); Serial.print(':');
    //Serial.print(GPS.seconds, DEC); Serial.print('.');
    //Serial.println(GPS.milliseconds);
    //Serial.print("Date: ");
    //Serial.print(GPS.day, DEC); Serial.print('/');
    //Serial.print(GPS.month, DEC); Serial.print("/20");
    //Serial.println(GPS.year, DEC);
    //Serial.print("Fix: "); Serial.print((int)GPS.fix);
    //Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
    if (GPS.fix) {
      Serial.print("GPS FIX");
      //Serial.print("Location: ");
      //Serial.print(GPS.latitude, 2); Serial.print(GPS.lat);
      //Serial.print(", ");
      //Serial.print(GPS.longitude, 2); Serial.println(GPS.lon);

      float fLat = decimalDegrees(GPS.latitude, GPS.lat);
      float fLon = decimalDegrees(GPS.longitude, GPS.lon);

      int closest_loc = find_closest_location(fLat, fLon);
      float targetLat = pgm_read_float(&lat_lon[closest_loc][0]);
      float targetLon = pgm_read_float(&lat_lon[closest_loc][1]);

      //Serial.print("Speed (knots): "); Serial.println(GPS.speed);
      //Serial.print("Angle: "); Serial.println(GPS.angle);
      //Serial.print("Altitude: "); Serial.println(GPS.altitude);
      //Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
      compass.read();
      int heading = compass.heading((LSM303::vector<int16_t>){0,-1,0});
      Serial.print("Heading: ");
      Serial.println(heading);
      if ((calc_bearing(fLat, fLon, targetLat, targetLon) - heading) > 0) {
        headingDirection(calc_bearing(fLat, fLon, targetLat, targetLon)-heading);
       }
      else {
        headingDirection(calc_bearing(fLat, fLon, targetLat, targetLon)-
heading+360);
      }

      //headingDistance((double)calc_dist(fLat, fLon, targetLat, targetLon));
      //Serial.print("Distance Remaining:"); Serial.println((double)calc_dist(fLat,
fLon, targetLat, targetLon));
    }
  }
}

int calc_bearing(float flat1, float flon1, float flat2, float flon2)
{
```

```
  float calc;
  float bear_calc;

  float x = 69.1 * (flat2 - flat1);
  float y = 69.1 * (flon2 - flon1) * cos(flat1/57.3);

  calc=atan2(y,x);

  bear_calc= degrees(calc);

  if(bear_calc<=1){
    bear_calc=360+bear_calc;
  }
  return bear_calc;
}

void headingDirection(float heading)
{
  //Use this part of the code to determine which way you need to go.
  //Remember: this is not the direction you are heading, it is the direction to the
destination (north = forward).
  if ((heading > 348.75)||(heading < 11.25)) {
    Serial.println("  N");
    //Serial.println("Forward");
    GoForward(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 11.25)&&(heading < 33.75)) {
    Serial.println("NNE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 33.75)&&(heading < 56.25)) {
    Serial.println(" NE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 56.25)&&(heading < 78.75)) {
    Serial.println("ENE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 78.75)&&(heading < 101.25)) {
    Serial.println("  E");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 101.25)&&(heading < 123.75)) {
    Serial.println("ESE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 123.75)&&(heading < 146.25)) {
    Serial.println(" SE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 146.25)&&(heading < 168.75)) {
    Serial.println("SSE");
    //Serial.println("Go Right");
    TurnRight(strip.Color(16, 247, 206), 200);
  }
```

```
  if ((heading >= 168.75)&&(heading < 191.25)) {
    Serial.println("  S");
    //Serial.println("Turn Around");
    TurnAround(strip.Color(247, 16, 70), 200);
  }

  if ((heading >= 191.25)&&(heading < 213.75)) {
    Serial.println("SSW");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 213.75)&&(heading < 236.25)) {
    Serial.println(" SW");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 236.25)&&(heading < 258.75)) {
    Serial.println("WSW");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 258.75)&&(heading < 281.25)) {
    Serial.println("  W");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 281.25)&&(heading < 303.75)) {
    Serial.println("WNW");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 303.75)&&(heading < 326.25)) {
    Serial.println(" NW");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }

  if ((heading >= 326.25)&&(heading < 348.75)) {
    Serial.println("NWN");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(16, 247, 206), 200);
  }
}

unsigned long calc_dist(float flat1, float flon1, float flat2, float flon2)
{
  float dist_calc=0;
  float dist_calc2=0;
  float diflat=0;
  float diflon=0;

  diflat=radians(flat2-flat1);
  flat1=radians(flat1);
  flat2=radians(flat2);
  diflon=radians((flon2)-(flon1));

  dist_calc = (sin(diflat/2.0)*sin(diflat/2.0));
  dist_calc2= cos(flat1);
  dist_calc2*=cos(flat2);
  dist_calc2*=sin(diflon/2.0);
  dist_calc2*=sin(diflon/2.0);
  dist_calc +=dist_calc2;

  dist_calc=(2*atan2(sqrt(dist_calc),sqrt(1.0-dist_calc)));
```

```
  dist_calc*=6371000.0; //Converting to meters
  return dist_calc;
}

//returns the location in the lat_lon array of the closest lat lon to the current
location
int find_closest_location(float current_lat, float current_lon)
{
  int closest = 0;
  signed long minDistance = -1;
  signed long tempDistance;
  for (int i=0; i < LAT_LON_SIZE; i++) {
    float target_lat = pgm_read_float(&lat_lon[i][0]);
    float target_lon = pgm_read_float(&lat_lon[i][1]);

    tempDistance = calc_dist(current_lat, current_lon, target_lat, target_lon);

    /*
    Serial.print("current_lat: ");
    Serial.println(current_lat, 6);
    Serial.print("current_lon: ");
    Serial.println(current_lon, 6);
    Serial.print("target_lat: ");
    Serial.println(target_lat, 6);
    Serial.print("target_lon: ");
    Serial.println(target_lon, 6);

    Serial.print("tempDistance: ");
    Serial.println(tempDistance);
    Serial.print("Array Loc: ");
    Serial.println(i);
    */

    if ((minDistance > tempDistance) || (minDistance == -1)) {
      minDistance = tempDistance;
      closest = i;

    }

  }
  return closest;
}

// Convert NMEA coordinate to decimal degrees
float decimalDegrees(float nmeaCoord, char dir) {
  uint16_t wholeDegrees = 0.01*nmeaCoord;
  int modifier = 1;

  if (dir == 'W' || dir == 'S') {
    modifier = -1;
  }

  return (wholeDegrees + (nmeaCoord - 100.0*wholeDegrees)/60.0) * modifier;
}

void TurnLeft (uint32_t c, uint8_t wait) {
  strip.setPixelColor(CenterRight, 0);
  strip.setPixelColor(CenterLeft, 0);
  strip.setPixelColor(FarLeft, c);
      strip.show();
      delay(wait);
  strip.setPixelColor(FarLeft, 0);
  strip.show();
      delay(wait);
}

void TurnRight (uint32_t c, uint8_t wait) {
  strip.setPixelColor(CenterRight, 0);
```

```
  strip.setPixelColor(CenterLeft, 0);
  strip.setPixelColor(FarRight, c);
      strip.show();
      delay(wait);
  strip.setPixelColor(FarRight, 0);
  strip.show();
      delay(wait);
}

void TurnAround (uint32_t c, uint8_t wait) {
  strip.setPixelColor(CenterRight, c);
  strip.setPixelColor(CenterLeft, c);
      strip.show();
      delay(wait);
  strip.setPixelColor(CenterRight, 0);
  strip.setPixelColor(CenterLeft, 0);
  strip.show();
      delay(wait);

}

void GoForward (uint32_t c, uint8_t wait) {
  strip.setPixelColor(CenterRight, c);
  strip.setPixelColor(CenterLeft, c);
      strip.show();

      delay(wait);
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i;

  //for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< 10; i++) {
      strip.setPixelColor(RightStrip[i], Wheel(((i * 256 / 10) + counter) & 255));
      strip.setPixelColor(RightCenterStrip[i], Wheel(((i * 256 / 10) + counter) &
255));
      strip.setPixelColor(LeftCenterStrip[i], Wheel(((i * 256 / 10) + counter) &
255));
      strip.setPixelColor(LeftStrip[i], Wheel(((i * 256 / 10) + counter) & 255));
    }
    counter = counter+25;
    if (counter >256*5) {counter = 0;}
    strip.show();
    delay(wait);
  //}
}


// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Right near the top of the sketch, you are going to see a long list of GPS coordinates that look like this:

```
float lat_lon[LAT_LON_SIZE][2] PROGMEM = {
  {40.767272, -73.993928},
  {40.719115, -74.006666},
  {40.711174, -74.000165},
```

These coordinates were copied and pasted using a bit of node.js (learn more here (https://adafru.it/cgr)). This also means that if you live in a city other than NYC that has a bike sharing program, you can use our code to get your own list of coordinates. If you live in NYC, all of the current bike share stations are loaded into the sketch.

The next piece of important code is this:

```
// Change the first # to match the # of LEDs in your strip -- we have 45 LEDs
Adafruit_NeoPixel strip = Adafruit_NeoPixel(45, 6, NEO_GRB + NEO_KHZ800);

int FarRight = 9;
int CenterRight = 10;
int CenterLeft = 34;
int FarLeft = 35;
int HeadsUp[] = {35, 34, 10, 9};
int RightStrip[] = {8, 7, 6, 5, 4, 3, 2, 1, 0};
int RightCenterStrip[] = {11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21};
int LeftCenterStrip[] = {33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23};
int LeftStrip[] = {36, 37, 38, 39, 40, 41, 42, 43, 44};
int counter = 0;
```

This is where we set up which LEDs on our helmet are used. This will likely need to be updated to fit your own bike helmet project. Learn more about how the FLORA NeoPixels work here (https://adafru.it/c9J).

In the setup function, you will find a section that deals with calibrating the FLORA compass module.

```
// Calibration values. Use the Calibrate example program to get the values for
// your compass.
  compass.m_min.x = -581; compass.m_min.y = -731; compass.m_min.z = -1097;
  compass.m_max.x = +615; compass.m_max.y = +470; compass.m_max.z = 505;
```

Included with the Pololu LSM303 library is a little sketch that will get you these values. Run the sketch and view the serial monitor. Then tilt your helmet/sensor in every possible direction. When done, dump the values in the serial monitor into the above bit of code (replacing our values).

Thats about all you need to know to get started. Upload the code to your FLORA, and you are ready to hit the road.

To test out your animations inside, try the following code that reacts to just the compass:

```
#include &lt;Adafruit_NeoPixel.h&gt;
#include &lt;LSM303.h&gt;
// Test code for Adafruit Flora GPS modules
//
// This code shows how to listen to the GPS module in an interrupt
// which allows the program to have more 'freedom' - just parse
// when a new NMEA sentence is available! Then access data when
// desired.
//
// Tested and works great with the Adafruit Flora GPS module
//    ------&gt; http://adafruit.com/products/1059
// Pick one up today at the Adafruit electronics shop
// and help support open source hardware &amp; software! -ada

Adafruit_NeoPixel strip = Adafruit_NeoPixel(45, 6, NEO_GRB + NEO_KHZ800);
int FarRight = 9;
int CenterRight = 10;
int CenterLeft = 34;
int FarLeft = 35;
int HeadsUp[] = {35, 34, 10, 9};
int RightStrip[] = {46, 8, 7, 6, 5, 4, 3, 2, 1, 0, 46};
int RightCenterStrip[] = {11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21};
int LeftCenterStrip[] = {33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23};
int LeftStrip[] = {46, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46};
int counter = 256*5;

#include &lt;Wire.h&gt;
#include &lt;LSM303.h&gt;

LSM303 compass;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  compass.init();
  compass.enableDefault();

  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

  // Calibration values. Use the Calibrate example program to get the values for
  // your compass. M min X: -561 Y: -679 Z: -558 M max X: 232 Y: 109 Z: 224
  compass.m_min.x = -561; compass.m_min.y = -679; compass.m_min.z = -558;
  compass.m_max.x = 232; compass.m_max.y = 109; compass.m_max.z = 224;
}

void loop() {
  compass.read();
  int heading = compass.heading((LSM303::vector){0,-1,0});

  //Use this part of the code to determine which way you need to go.
  if ((heading &gt; 348.75)||(heading &lt; 11.25)) {
    Serial.println("  N");
    //Serial.println("Forward");
    GoForward(strip.Color(0, 51, 20), strip.Color(255, 255, 0), 200);

  }

  if ((heading &gt;= 11.25)&amp;&amp;(heading &lt; 33.75)) {
    Serial.println("NNE");
    //Serial.println("Go Left");
    GoForward(strip.Color(0, 51, 10), strip.Color(255, 255, 0), 200);
  }

  if ((heading &gt;= 33.75)&amp;&amp;(heading &lt; 56.25)) {
    Serial.println(" NE");
    //Serial.println("Go Left");
    TurnLeft(strip.Color(11, 79, 0), strip.Color(255, 255, 0), 200);
```

```
}

if ((heading &gt;= 56.25)&amp;&amp;(heading &lt; 78.75)) {
  Serial.println("ENE");
  //Serial.println("Go Left");
  TurnLeft(strip.Color(39, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 78.75)&amp;&amp;(heading &lt; 101.25)) {
  Serial.println("  E");
  //Serial.println("Go Left");
  TurnLeft(strip.Color(74, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 101.25)&amp;&amp;(heading &lt; 123.75)) {
  Serial.println("ESE");
  //Serial.println("Go Left");
  TurnLeft(strip.Color(74, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 123.75)&amp;&amp;(heading &lt; 146.25)) {
  Serial.println(" SE");
  //Serial.println("Go Left");
  TurnLeft(strip.Color(79, 61, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 146.25)&amp;&amp;(heading &lt; 168.75)) {
  Serial.println("SSE");
  //Serial.println("Go Left");
  TurnAround(strip.Color(79, 61, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 168.75)&amp;&amp;(heading &lt; 191.25)) {
  Serial.println("  S");
  //Serial.println("Turn Around");
  TurnAround(strip.Color(79, 32, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 191.25)&amp;&amp;(heading &lt; 213.75)) {
  Serial.println("SSW");
  //Serial.println("Go Right");
  TurnAround(strip.Color(79, 61, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 213.75)&amp;&amp;(heading &lt; 236.25)) {
  Serial.println(" SW");
  //Serial.println("Go Right");
  TurnRight(strip.Color(79, 61, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 236.25)&amp;&amp;(heading &lt; 258.75)) {
  Serial.println("WSW");
  //Serial.println("Go Right");
  TurnRight(strip.Color(74, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 258.75)&amp;&amp;(heading &lt; 281.25)) {
  Serial.println("  W");
  //Serial.println("Go Right");
  TurnRight(strip.Color(74, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 281.25)&amp;&amp;(heading &lt; 303.75)) {
  Serial.println("WNW");
  //Serial.println("Go Right");
TurnRight(strip.Color(39, 79, 0), strip.Color(255, 255, 0), 200);
}

if ((heading &gt;= 303.75)&amp;&amp;(heading &lt; 326.25)) {
```

```
    Serial.println(" NW");
    //Serial.println("Go Right");
    TurnRight(strip.Color(11, 79, 0), strip.Color(255, 255, 0), 200);
  }

  if ((heading &gt;= 326.25)&amp;&amp;(heading &lt; 348.75)) {
    Serial.println("NWN");
    //Serial.println("Go Right");
    GoForward(strip.Color(0, 51, 10), strip.Color(255, 255, 0), 200);
  }
}

void TurnLeft (uint32_t c, uint32_t stripe, uint8_t wait) {
  strip.setPixelColor(CenterRight, 0);
  strip.setPixelColor(CenterLeft, 0);
  strip.setPixelColor(FarLeft, c);
      strip.show();
      for(uint16_t i=0; i&lt;11; i++) {
      strip.setPixelColor(LeftCenterStrip[i], strip.Color(0, 0, 0));
      strip.setPixelColor(LeftStrip[i], strip.Color(0, 0, 0));
      strip.show();
      delay(30);
      }
      //delay(wait);
  strip.setPixelColor(FarLeft, 0);
  strip.show();
     colorWipe(stripe, 30);
      //delay(wait);
}

void TurnRight (uint32_t c, uint32_t stripe, uint8_t wait) {
  strip.setPixelColor(CenterRight, 0);
  strip.setPixelColor(CenterLeft, 0);
  strip.setPixelColor(FarRight, c);
      strip.show();
      for(uint16_t i=0; i&lt;11; i++) {
      strip.setPixelColor(RightCenterStrip[i], strip.Color(0, 0, 0));
      strip.setPixelColor(RightStrip[i], strip.Color(0, 0, 0));
      strip.show();
      delay(30);
  }
      //delay(wait);
  strip.setPixelColor(FarRight, 0);
  strip.show();
      colorWipe(stripe, 30);
      //delay(wait);
}

void TurnAround (uint32_t c, uint32_t stripe, uint8_t wait) {
  strip.setPixelColor(CenterRight, c);
  strip.setPixelColor(CenterLeft, c);
      strip.show();
      for(uint16_t i=0; i&lt;11; i++) {
      strip.setPixelColor(LeftCenterStrip[i], strip.Color(0, 0, 0));
      strip.setPixelColor(RightCenterStrip[i], strip.Color(0, 0, 0));
      strip.show();
      delay(30);
  }
      //delay(wait);
  strip.setPixelColor(CenterRight, 0);
  strip.setPixelColor(CenterLeft, 0);
  strip.show();
      colorWipe(stripe, 30);
      //delay(wait);

}

void GoForward (uint32_t c, uint32_t stripe, uint8_t wait) {
  strip.setPixelColor(CenterRight, c);
```

```
  strip.setPixelColor(CenterLeft, c);
      strip.show();
      for(uint16_t i=0; i&lt;11; i++) {
      strip.setPixelColor(LeftCenterStrip[i], strip.Color(0, 0, 0));
      strip.setPixelColor(RightCenterStrip[i], strip.Color(0, 0, 0));
      strip.show();
      delay(30);
  }
      colorWipe(stripe, 30);
      //delay(wait);
}

// Slightly different, this makes the rainbow equally distributed throughout
void colorWipe(uint32_t c, uint8_t wait) {
for(uint16_t i=0; i&lt;11; i++) {
      strip.setPixelColor(LeftCenterStrip[i], strip.Color(255, 255/i^16, 255/i^16));
      strip.setPixelColor(RightCenterStrip[i], strip.Color(255, 255/i^16, 255/
i^16));
      strip.setPixelColor(RightStrip[i], strip.Color(255, 255/i^16, 255/i^16));
      strip.setPixelColor(LeftStrip[i], strip.Color(255, 255/i^16, 255/i^16));
      strip.show();
      delay(wait);
  }
}


// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos &lt; 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos &lt; 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Wear it!

The sample code uses the LEDs at the right and left edge of the helmet's brim to signal you to turn, and uses the two LEDs in the center of your forehead to tell you

"go forward" (solid blue) or "turn around" (blinking red). Customize to your own navigation style!

Oh hey! You weren't thinking of wearing this helmet in the rain, were you? The circuit isn't waterproof! We recommend removing the battery if you have to wear it in the rain, then waiting until it's completely dry before reconnecting.

Optional: Generating Coordinates

You may need to re-generate the coordinates for the bike stations from time to time, and this page will show you how to do that.

Another reason you may want to re-generate coordinates is so you can use the bike share system in your city with the helmet! It isn't limited to the NYC Citi Bike system.

The below code will parse the wonderful citybik.es api (https://adafru.it/cgu):

```
//Change the URL to whichever city you'd like to parse.  You can then execute this
file with node.js from the command line:
//node parser.js

//The results will be a multi-dimensional array that you can just plug into the
bike helmet code for the city you live in!
//Make sure to also update the #define with the size of the array.  This is the
last number that is output after running this parser.

var request = require('request');
var BIKE_SHARE_URL = 'http://api.citybik.es/citibikenyc.json';
request(BIKE_SHARE_URL, function (error, response, body) {
  if (!error && response.statusCode == 200) {
    var comma = ",";
    var locations = JSON.parse(body);
    locations.forEach(function(l, i) {
      if (i === locations.length-1)
```

```
        comma = "";

      console.log("  {" + (l.lat / 1000000).toFixed(6) + ", " + (l.lng /
1000000).toFixed(6) + "}" + comma);
    });

    console.log(locations.length);
  }
});
```

Let's not get too far ahead of ourselves though. First, we need to install the dependencies to run that code.

To start with, you'll need node.js. It's a really easy install. Navigate to http://nodejs.org (https://adafru.it/cgt), and follow the installation instructions for your operating system (Windows, Linux, and OS X are supported).

Next, create a folder somewhere (mine is titled 'cityparser'). Then, create a file in that folder titled parser.js and copy and paste the above snippet of code into that file, and save it.

Now, open your favorite command line utility (Terminal.app, cmd.exe, etc) and navigate into the 'cityparser' folder.

Now, we need to install the one dependency that is required for the parser to run. Execute the following from within the cityparser folder:

```
npm install request
```

Now that you have request installed, you can (finally!) run the parser to generate the locations for your particular city bike share program.

Execute the following command to run the parser:

```
node parser.js
```

Great! It should have output a bunch of coordinates with a count at the end of it.

For example, these are the last few of my output:

```
  {40.715348, -73.960241},
  {40.741472, -73.983209},
  {40.736502, -73.978094},
  {40.744449, -73.983035},
  {40.702550, -73.989402},
  {40.698920, -73.973329},
  {40.716887, -73.963198},
  {40.734160, -73.980242},
  {40.725500, -74.004451},
```

```
  {40.705311, -73.971000},
  {40.765909, -73.976341}
311
```

If your output is similar to the above, you'll now want to choose the correct BIKE_SHARE_URL for your city (NYC pre-loaded).

Open the citybik.es api (https://adafru.it/cgu) page, and scroll down to the section titled "System" and "JSON". Choose your location from the dropdown, and then replace it in the parser.js file variable "BIKE_SHARE_URL" and re-run the program.

Ok, now to set up your sketch. The last number in the results is the count of locations in that bike share. Take that number and place it in the sketch for the size of the array:

```
#define LAT_LON_SIZE 311
```

Then, copy and paste all of the locations (not the number), and replace the existing locations in the Bike Helmet Sketch. It should look something like this:

```
float lat_lon[LAT_LON_SIZE][2] PROGMEM = {
  {40.767272, -73.993928},
  {40.719115, -74.006666},
  {40.711174, -74.000165},
  {40.683826, -73.976323},
  {40.702550, -73.989402},
  {40.698920, -73.973329},
  {40.716887, -73.963198},
  {40.734160, -73.980242},
  {40.725500, -74.004451},
  {40.705311, -73.971000},
  {40.765909, -73.976341}
};
```

Now, compile your sketch, upload it, and start biking!