With a source of GPS coordinates and a way to tell which direction we are facing, we can now combine that information to figure out how to get from one location to the next.

Heading Control.H:

```
#ifndef HEADINGCONTROL_H
#define HEADINGCONTROL_H

struct GPSPair
{
    // Lat1 and Lon1 can be viewed as the "current position".
    // Lat2 and Lon2 can be veiwed as the "target position".
    long lat1_L; // Lattitude and longigutes
    long lon1_L; // The _L indicates they are
    long lat2_L; // long data types. Not
    long lon2_L; // in decimal form yet.
};

class HeadingControl
{
    public:
        double GetDistanceBetween(GPSPair &someGPSPoints);
        double GetTargetHeading(GPSPair &someGPSPoints);
        double GetHeadingError(double CurrentHeading, double TargetHeading);
        HeadingControl();
};

#endif
```

Notice that we care about 3 functionalities:

What is the distance between a pair of GPS coords?
What heading do we need to go from pair 1 to pair 2?
What is the difference in our current and target heading?

Writing a Heading Control Library

HeadingControl.cpp:

```cpp
#include "HeadingControl.h"
#include <math.h>
#include <Arduino.h>

HeadingControl::HeadingControl(){}

double HeadingControl::GetDistanceBetween(GPSPair &someGPSPoints)
{
  // returns distance in meters between two positions, both specified
  // as signed decimal-degrees latitude and longitude. Uses great-circle
  // distance computation for hypothetical sphere of radius 6372795 meters.
  // Because Earth is no exact sphere, rounding errors may be up to 0.5%.
  // Courtesy of Maarten Lamers

  // Convert lat and long to degrees
  double lat1   =  (double)someGPSPoints.lat1_L  / 10000000.0;
  double long1  =  (double)someGPSPoints.lon1_L  / 10000000.0;
  double lat2   =  (double)someGPSPoints.lat2_L  / 10000000.0;
  double long2  =  (double)someGPSPoints.lon2_L  / 10000000.0;
  double delta  = radians(long1-long2);
  double sdlong = sin(delta);
  double cdlong = cos(delta);
  lat1          = radians(lat1);
  lat2          = radians(lat2);
  double slat1  = sin(lat1);
  double clat1  = cos(lat1);
  double slat2  = sin(lat2);
  double clat2  = cos(lat2);
  delta         = (clat1 * slat2) - (slat1 * clat2 * cdlong);
  delta         = sq(delta);
  delta        += sq(clat2 * sdlong);
  delta         =  sqrt(delta);
  double denom  = (slat1 * slat2) + (clat1 * clat2 * cdlong);
  delta         = atan2(delta, denom);
  return delta * 6372795;
}

double HeadingControl::GetTargetHeading(GPSPair &someGPSPoints)
{
  // returns course in degrees (North=0, West=270) from position 1 to position 2,
  // both specified as signed decimal-degrees latitude and longitude.
  // Because Earth is no exact sphere, calculated course may be off by a tiny fraction.
  // Courtesy of Maarten Lamers

  // Convert lat and long to degrees
  double lat1  =  (double)someGPSPoints.lat1_L  / 10000000.0;
  double long1 =  (double)someGPSPoints.lon1_L  / 10000000.0;
  double lat2  =  (double)someGPSPoints.lat2_L  / 10000000.0;
  double long2 =  (double)someGPSPoints.lon2_L  / 10000000.0;
  double dlon  = radians(long2-long1);
  lat1         = radians(lat1);
  lat2         = radians(lat2);
  double a1    = sin(dlon) * cos(lat2);
  double a2    = sin(lat1) * cos(lat2) * cos(dlon);
  a2           = cos(lat1) * sin(lat2) - a2;
  a2           = atan2(a1, a2);

  if (a2 < 0.0)
  {
    a2 += TWO_PI;
  }
  return degrees(a2);
}
```

```cpp
double HeadingControl::GetHeadingError(double aHeading, double aTargetHeading)
{
    // Computes the difference between aHeading and aTargetHeading.
    // Returns a value between -1 and 1 representing how away the two headings are.

    double error = aTargetHeading - aHeading;

    // Give the shortest turn distance from -180 to 180.
    if (error >= 180)
    {
        error -= 360;
    }
    else if (error <= -180)
    {
        error += 360;
    }

    // Return error number between -1 and 1.
    return error/180.0;
}
```

Sample Driver Code:

```cpp
#include "src/HeadingControl/HeadingControl.h"
#include "src/GPS/gps.h"
#include "src/BNO/BNO.h"

// Objects
HeadingControl myHeadingControl = HeadingControl();
GPS            myGPS            = GPS();
BNO            myBNO            = BNO();

// Data Structs
GPSData        myGPSData;
GPSPair        myGPSPair;

// Set a destination here using a LONG variable.
// This is inbetween the garages at the Pi Lab.
long destinationLat  =  334672860;
long destinationLon =  -819910550;

// Variables we want to get from heading control
double distanceBetween;
double targetHeading;
double headingError;

int bnoResetPin = 6;
double currentHeading;
```

Writing a Heading Control Library

Dec 2023
Working code

```
11:36:49.119 -> Distance: 61.79
11:36:49.119 -> TargetHeading: 210.26
11:36:49.119 -> CurrentHeading: 213.59
11:36:49.119 -> Error: -0.02
11:36:49.119 ->
11:36:50.129 -> Distance: 61.54
11:36:50.129 -> TargetHeading: 210.25
11:36:50.129 -> CurrentHeading: 213.59
11:36:50.129 -> Error: -0.02
11:36:50.129 ->
11:36:51.118 -> Distance: 61.35
11:36:51.118 -> TargetHeading: 210.22
11:36:51.118 -> CurrentHeading: 213.59
11:36:51.118 -> Error: -0.02
11:36:51.118 ->
```

```ino
void setup()
{
  // Serial monitor for output.
  Serial.begin(9600);
  Serial.println("GetObsessed!");

  // Connect the GPS to Serial1.
  // This is RX0 and TX0 on the giga board.
  Serial1.begin(38400);
  if(!myGPS.setupGPS(Serial1))
  {
    // Freeze if setup fails
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("GPS Failed to begin.");
    while(1);
  }

  // Connect the BNO to Serial2
  Serial2.begin(115200);
  if(!myBNO.setupBNO(&Serial2, bnoResetPin))
  {
    // Freeze if BNO failed
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("BNO Failed to start");
    while(1);
  }

  // Set the "destination" to point 2 in the data structure.
  myGPSPair.lat2_L = destinationLat;
  myGPSPair.lon2_L = destinationLon;

}
```

```ino
void loop()
{
  // Get an update from the GPS.
  myGPS.GetData(myGPSData);

  // Set the current position to point 1 in the data structure.
  myGPSPair.lat1_L = myGPSData.lattitude;
  myGPSPair.lon1_L = myGPSData.longitude;

  // Get the pieces of information that we care about.
  distanceBetween = myHeadingControl.GetDistanceBetween(myGPSPair);
  targetHeading   = myHeadingControl.GetTargetHeading(myGPSPair);
  currentHeading  = myBNO.GetHeading();
  headingError    = myHeadingControl.GetHeadingError(currentHeading, targetHeading);

  // Print the information
  Serial.print("Distance: ");        Serial.println(distanceBetween);
  Serial.print("TargetHeading: ");   Serial.println(targetHeading);
  Serial.print("CurrentHeading: ");  Serial.println(currentHeading);
  Serial.print("Error: ");           Serial.println(headingError);     Serial.println();
}
```