Measuring Orientation adapted for the SeaPerch

Devan Anderson
Department of Mechanical Engineering
(Undergraduate)
Brigham Young University
Provo, UT 84604

In order for any pilot to effectively navigate his or her vehicle or ROV he must know the orientation of the vehicle he is trying to navigate. Research and development was done to create a digital compass that could be used on board the SeaPerch ROV. The instrument uses a magnetometer and arduino board as the main hardware components for this project. Program code was developed to calibrate the instrument and output the calculated heading to a laptop computer. Testing was performed to verify the accuracy and precision of the instrument. After a series of tests it was determined that when used correctly the instrument's accuracy is ±4.5° of the true heading. Large inaccuracies were found when the compass instrument was calibrated incorrectly, or when the instrument was significantly tilted. These cases where large inaccuracies occurred are not common if the instrument is used correctly. The accuracy of the built instrument is satisfactory for application to the SeaPerch.

Introduction

Position and orientation are two crucial pieces of information one must always know to be able to effectively pilot the SeaPerch ROV. Thus far the pilot has always had to have a clear visual of the SeaPerch in order to determine the position and orientation of the ROV. In murky or dark conditions the pilot may or may not always be able to see the SeaPerch, especially if the aquatic vehicle is taken on excursions outside of pools or water tanks. Imagine if the SeaPerch needed to go under a ledge or a space where it was no longer visible. The operator would be able to guess at the position of the SeaPerch but determining the orientation would be much more difficult making it nearly impossible to maneuver the vehicle back out.

Conventionally 2-D headings (N,E,S,W) have been used to determine orientation traveling on earth's surface. This description of orientation is commonly referred to as a heading. The SeaPerch is designed in such a way so as to remain level, therefore it is safe to assume that the orientation in the altitudinal direction is unchanging and parallel to earth's surface. A traditional heading using cardinal directions is an appealing way to inform the pilot about the orientation of the SeaPerch.

The proposed device will use a magnetometer to measure the magnetic field in two perpendicular axes, parallel to the ground. Using a calibration process and basic trigonometric functions we will use the raw measurements of the magnetic field,that we

obtain from the magnetometer, to calculate a heading and output that to a laptop display. The heading will update itself continuously so that the pilot may know the orientation of the ROV always. The device must be waterproofed to keep it from getting wet while onboard the SeaPerch.

Goal

The goal of the project is to continuously provide the pilot with an accurate heading in reference to magnetic north. The goal is to achieve a heading that is accurate within ±5°.

Methods and Materials

Overview

The HMC5883L 3-axis digital compass module (magnetometer), arduino circuit board, and arduino breadboard shield are the primary parts we need to make our instrument, for a complete list of parts and estimated prices see Appendix B. The compass module is wired to the arduino through the breadboard shield as shown in Figure 1.

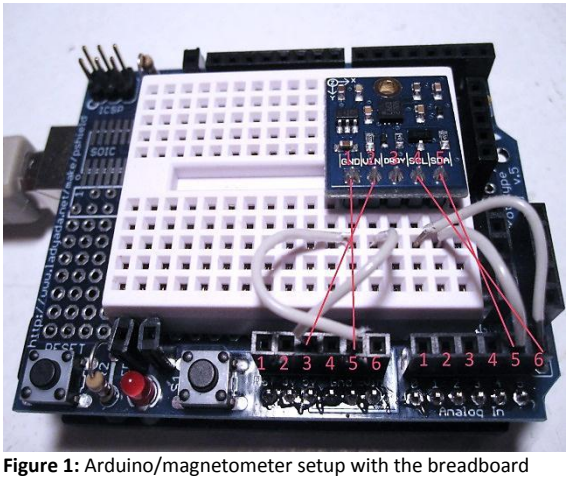

Figure 1: Arduino/magnetometer setup with the breadboard shield mounted on top of the arduino circuit board. The magnetometer and arduino breadboard are interface as seen above.

The Arduino board is programmed to ask the compass module for a measurement of the magnetic field. The magnetometer sends a the measured values in I2C, a low level programming language commonly used in microcontrollers, to the arduino. The arduino interprets the signal and the program derives the heading and outputs it to the Serial monitor on your laptop.

The instrument itself is not waterproof and must be placed in a waterproof container like a Tupperware (make sure you test the container before you put it underwater). It is important that buoyancy is considered to keep the instrument from floating. Make sure that the Tupperware weighs slightly more than the water it will displace by putting bags of dirt or rocks overtop the instrument in the Tupperware, otherwise it will float. (Serway, 285)

Magnetism

The earth has two magnetic poles one in the north and one in the south, this produces a magnetic field which envelopes the surface of the earth. Think of the field as flowing from the north pole to the south pole, this flowing is commonly described and/or illustrated as magnetic field lines. Magnetic field lines penetrate water allowing our instrument to work underwater the same as on land. (Serway, 809)

The compass module uses a 3 axis magnetometer from Honeywell (HMC5883L) which has a digital resolution of 0.7 - 4.35 milligauss (milligauss is a unit of magnetic field strength). The resolution of this instrument is sufficient for our application because the earth's magnetic field ranges between 250-650 milligauss, much larger than our instruments

resolution. The magnetometer has three different sensors that measure magnetic field strength in three orthogonal directions as seen in figure 2. These three directions are referred to as the x, y, and z components. (Honeywell, 2)

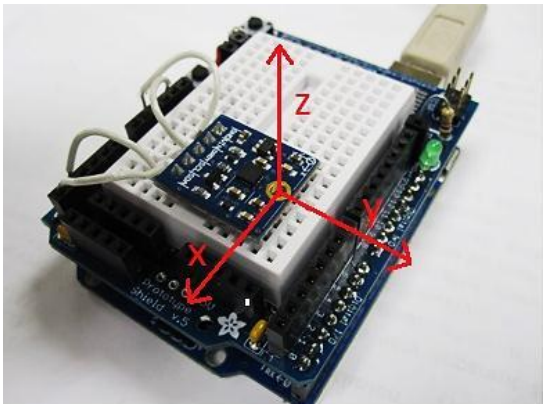

Figure 2: x, y, & z are the common direction vectors we use to describe 3-D space, these are the x, y, and z directions with respect to the magnetometer.

The sum of these three components can describe any point in 3-D space. For this project only the x and y components are needed because a heading is a 2-D direction on a flat plane, such as earth's surface.

Program Methodology

The big picture of the program is to take the measured x and y components of the magnetic field and manipulate them so that when plotted on a graph they will create the end point of a unit vector (a unit vector is a vector that is one unit in length) from the origin (0, 0). With this unit vector we can easily use the arctangent function to find the heading in degrees. If the program works as it should the manipulated x and y data should create a circle of radius one. This is referred to as a unit circle because any line (vector) drawn from the center of the circle (origin) to the outside of the circle is one unit in length. See Figure 3.

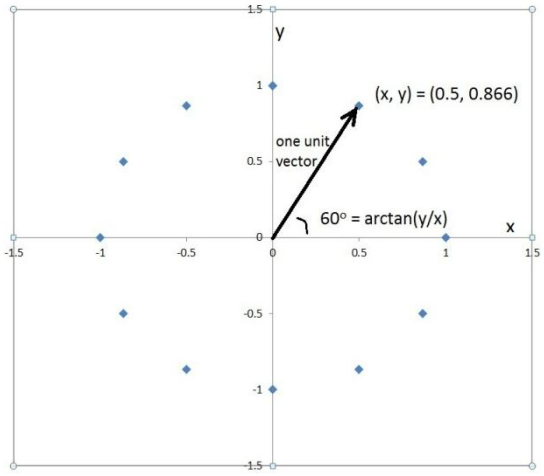

Figure 3: The unit circle is the ideal plot that our data will give. Each point should be one unit in length from the origin (0, 0). This line can be referred to as a unit vector. The heading can be calculated with the arctangent function.

The whole program is imbedded in one continuous loop and contains two if statements that divide the program into two separate parts: calibration, and post-calibration. During calibration the program iterates every three milliseconds. The user is to rotate the compass approximately 720° over the course of the one minute calibration phase. The compass stores and holds on to the largest and the smallest values it measures in the x, y, and z direction: x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, and z_{min}. These maximum and minimum values are used to create the following calibration factors.

$$x_{range} = \frac{x_{max} - x_{min}}{2}$$

$$x_{offset} = x_{range} + x_{min}$$

x_{offset} is used to center the range of data about zero. X_{range} represents the difference from zero to x_{max} once the data is centered about zero by the offset factor. This process is repeated for y values.

Once the calibration phase is complete the calibration factors do not change. The program begins to iterate every five seconds.

The adjusted x and y values used to calculate the heading are computed as follows.

$$x_{center} = x - x_{offset}$$

$$x_{vector} = \frac{x_{center}}{x_{range}}$$

x_{vector} and y_{vector} represent the x and y components of the hoped for unit vector. The program then goes into one of four if-statements depending on the sign (- or +) of x_{vector} and y_{vector}. These four if-statements represent the four quadrants of the unit circle. Finally the heading is determined and converted to degrees.

$$heading = \tan^{-1}\left(\frac{y_{vector}}{x_{vector}}\right) * \left(\frac{180°}{\pi}\right)$$

Experimental Setup

Six tests will be executed to ensure the accuracy and dependability of the instrument. The arduino will be rebooted between each test so that the instrument will recalibrate. For each test the instrument will be aligned with true magnetic north, and will be rotated in increments of 10°. The raw and adjusted x and y components will be recorded as well as the output heading and true heading. Of the six tests, four will verify that the instrument operates correctly under normal conditions. Two of the tests will test how the instrument performs if it is not level or if it has been improperly calibrated.

True magnetic north will be found with an engineering analog compass and verified with a standard digital compass. A 360° protractor will be fastened to the table with the 360° tick mark pointing towards magnetic north. See Figure 4.

Figure 4: The analog compass orients the protractor so that 360° points north. This gives us an accurate heading with which to compare the instrument's heading.

Tests 1 and 2 are performed using a 4000 sample calibration phase. Tests 3 and 4 use a 6000 sample calibration phase, it is expected that will improve the calibration factors thereby improving the accuracy of the instrument. In Test 5 the instrument is calibrated on a level surface and then tilted 15° about the x-axis. It is then tested at this angle. In Test 6 the compass is only rotated 180° during the calibration phase instead of the usual 720°. Accuracy and precision are expected to decrease dramatically for tests 5 and 6.

Results

The data from the tests were analyzed and are summarized in tables and graphs found in this section. Error between the true heading and the instrument's measured heading was important; as well as the change in x and y from pre-calibration to post-calibration.

The average difference in degrees between the true heading and the instrument's heading is shown in the average error column in Table 1.

Table 1: The average number of degrees that the output heading differed from the true heading on the protractor. (All negatives were changed to positives in order to get the magnitude of error)

Test	Average Error (deg.)
Test 1	4.09
Test 2	4.37
Test 3	4.17
Test 4	5.54
Test 5	32.5
Test 6	17.7

The x and y component data for Tests 1-4 is graphed in Figures 5 and 6. Figure 5 shows all of the non-adjusted x and y values. Figure 6 shows the x and y components after they have been adjusted by the calibration factors.

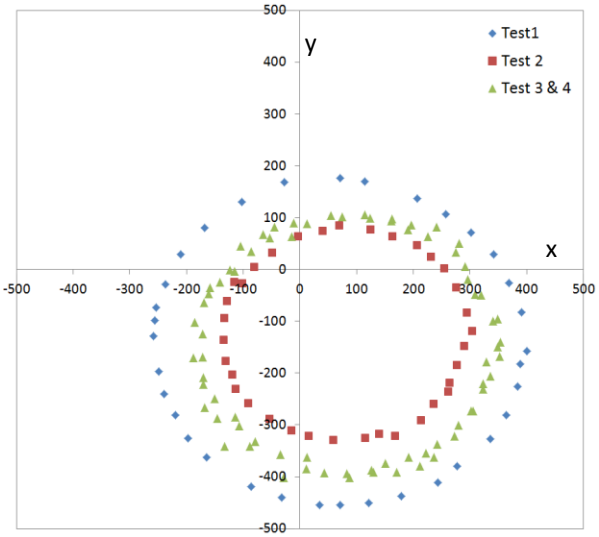

Figure 5: Non-adjusted x and y values (x, y) for Tests 1-4, pre-calibration. The data appears very off-center and slightly elliptical in shape.

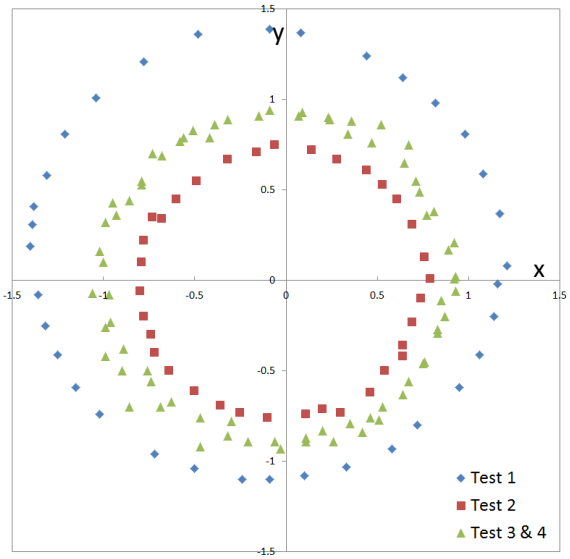

Figure 6: Adjusted (post-calibrated) x and y values (x_{vector}, y_{vector}) for Tests 1-4. The data appears centered and circular in shape.

Notice in Figure 4 and 5 how the data varies between tests; with each new calibration the radius of the data changes. Also notice that Test 3 and 4 (higher number of samples during calibration) are much more closely calibrated to the target values -1 and 1.

The adjusted x and y values are graphed in Figure 7. It is visually apparent from Figure 7 that the data does not produce smooth circles like the data from our other four tests; these two data sets generate very large errors compared to Tests 1-4, see Figure 8 and Table 1.

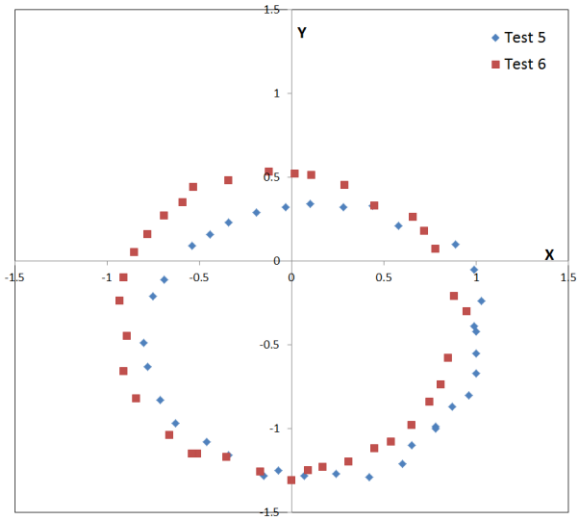

Figure 7: Adjusted (post-calibration) x and y values (x_{vector}, y_{vector}) for Tests 5 and 6. The data produces rough, off-center circles.

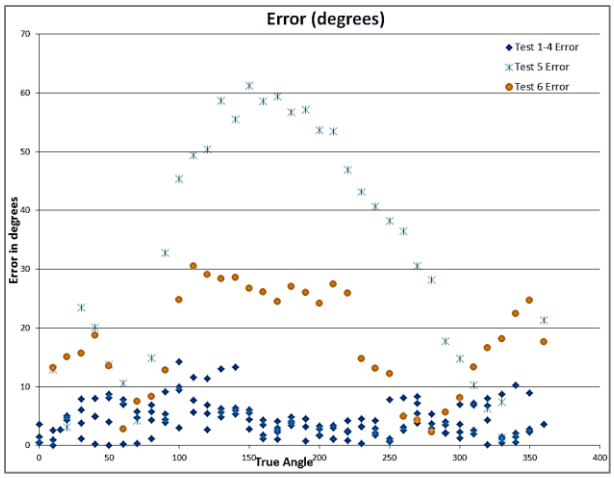

Figure 8: Absolute error at each measurement of Tests 1-6. The x–axis is the true heading read off of the protractor. The y-axis is the difference between the true heading and measured heading in degrees.

Discussion and Conclusion

Looking at the graphs and tables we can quickly draw some conclusions about the data. By looking at the average error in Table 1 and looking at the data in Figure 6 we see that the general spread of the data doesn't affect the accuracy because Tests 1-4 produced about the same average error. By comparing the error data in Table 1 to Figure 6 and 7 we see that concentricity about the center makes the biggest difference in accuracy. This makes sense because the angle is calculated about the origin. The data from Test 5 and Test 6 make it apparent that if the compass is not calibrated correctly or if the compass is not calibrated to the correct plane, the errors will be huge and the output heading worthless. Because the SeaPerch is designed to remain fairly level inaccuracies due to tilt should be minimal and short.

This instrument could be improved by using an accelerometer to measure changes in tilt; this could be integrated with the magnetometer to compensate for tilt. Another nice improvement would be to know when the SeaPerch is too tilted for the instrument to give an accurate heading. This could be implemented by programming the arduino to output a warning to the laptop screen when the z component measures outside of a predetermined range. This should work because the z axis should remain constant throughout operation it always should point up (Figure 2).

From the Tests 1-4 we can conclude that the compass operates within ±5° of accuracy when it is level and calibrated correctly. Because the resolution of our instrument is very high and our data follows expected trends I think it a safe assumption that a lot of the noise and inaccuracies in our data were due to measurement error. The resolution of the protractor is about ± 1°, and then the alignment accuracy was probably added ±2° to that. Either way the accuracy of the created instrument will meet the needs of any SeaPerch enthusiast, enriching the experience by giving them an accurate heading in real time.

Acknowledgements

I would like to thank Tony Guntharp for the use of a snippet of his code used to get the raw x, y, and z values from the magnetometer. I would also like to thank Annie Xie for her expertise in helping me write this paper. Finally, a special thanks to Dr. Tadd Truscott for his mentorship and hard work to get the grant money to make this project possible.

References

Serway, Raymond A., Chris Vuille, and Jerry S. Faughn. "Buoyant Forces and Archimede's Principle." *College Physics*. 8th ed. Vol. 1. Belmont: Brooks/Cole, 2008. 284-288. Print.

Etter, Delores M., and Jeanine A. Ingber. *Engineering Problem Solving with C++*. 2nd ed. Upper Saddle River: Pearson Prentice Hall, 2008. Print.

Serway, Raymond A., Jewitt, John W. "Magnetic Fields and Forces." *Physics for Scientists and Engineers with Modern Physics*. 7th ed. Belmont: Harris, 2008. 808-810. Print.

Figliola, Richard S., and Donald E. Beasley. *Theory and Design for Mechanical Measurements*. 5th ed. N.p.: John Wiley & Sons, 2011. Print.

Honeywell International. *3-Axis Digital Compass IC HMC5883L*. Digital image. N.p., Feb. 2010. Web. 16 Mar. 2013. <http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf>.

"Arduino - Reference." *Arduino - Reference*. N.p., n.d. Web. 1 Apr. 2013. <http://arduino.cc/en/Reference/HomePage>.

Appendix A: Arduino Code

Below is the code used to operate the SeaPerch digital compass. There are a lot of print commands spread throughout the program which were used to check proper functionality of the variables as I went. If you want the variable output to the screen delete the comments at the front of the line of code (//) if you don't want it output to the screen add comments(//) in front of that line of code.

```
#include <Wire.h>

////////////////////////////Tony's Code//////////////////////

/*
  Compass Module 3-Axis HMC5883L

  Read the Compass Module 3-Axis HMC5883L and prints them over the serial connection to the computer.

  The circuit:
   * SDA (Data) output of compass to analog pin 4
   * SCL (Clock) output of compass to analog pin 5
   * +V of accelerometer to +5V
   * GND of accelerometer to ground

  created 29 Nov 2012
  by Tony Guntharp

 */

#define Addr 0x1E            // 7-bit address of HMC5883 compass

void setup() {
  Serial.begin(9600);
  delay(100);               // Power up delay
  Wire.begin();

  // Set operating mode to continuous
  Wire.beginTransmission(Addr);
  Wire.write(byte(0x02));
  Wire.write(byte(0x00));
  Wire.endTransmission();
}
int calibrate = 1;
static double bigx, smallx, bigy, smally, bigz = -400, smallz;
static double avgx, avgy, offsetx, offsety;

void loop() {
```

```
 double x, y, z, heading;


 // Initiate communications with compass
 Wire.beginTransmission(Addr);
 Wire.write(byte(0x03));       // Send request to X MSB register
 Wire.endTransmission();

 Wire.requestFrom(Addr, 6);    // Request 6 bytes; 2 bytes per axis
 if(Wire.available() <=6) {    // If 6 bytes available
   x = Wire.read() << 8 | Wire.read();
   z = Wire.read() << 8 | Wire.read();
   y = Wire.read() << 8 | Wire.read();
 }
 ////////////////////////////My Code//////////////////////////

 if(calibrate==1){
  Serial.println("Calibrating, turn in circles slowly for two minutes, keep compass level");
 }
////////////////Calibration if statement/////////////////////
 if(calibrate<6000){

 if(x>bigx){
  bigx=x;
 }
 if(x<smallx){
  smallx=x;
 }
 if(y>bigy){
  bigy=y;
 }
 if(y<smally){
  smally=y;
 }
 if(z>bigz){
  bigz=z;
 }
 if(z<smallz){
  smallz=z;
 }

////////////////////Create and store Calibration factors///////////
 avgx= (bigx-smallx)/2;
 avgy= (bigy-smally)/2;
 offsetx= avgx+ smallx;
 offsety= avgy+ smally;
 delay(3);
/////////////////////////////////////////////
```

```
}
if(calibrate==6000){
  Serial.println("Calibration complete");
}

//////////////////////Post-Calibration/////////////
 if(calibrate>6000){
     // Print raw values
  /*Serial.print("X=");
  Serial.print(x);
  Serial.print(", Y=");
  Serial.print(y);
  Serial.print(", Z=");
  Serial.println(z);
   Serial.print("bigX=");
  Serial.print(bigx);
  Serial.print(", smallX=");
  Serial.print(smallx);
  Serial.print(", bigY=");
  Serial.print(bigy);
   Serial.print("smallY=");
  Serial.print(smally);
  Serial.print(", bigz=");
  Serial.print(bigz);
  Serial.print(", smallz=");
  Serial.println(smallz);
  Serial.print("offsetx: ");
  Serial.println(offsetx);
  Serial.print("avgx: ");
  Serial.println(avgx);
  Serial.print("offsety: ");
  Serial.println(offsety);
  Serial.print("avgy: ");
  Serial.println(avgy);*/
  Serial.print(x);
  Serial.print(" ");
  Serial.print(y);
  Serial.print(" ");
  Serial.print(z);
  Serial.print(" ");

//////////////////////////////////////
//////////////Applying calibration factors////////////
  x=x-offsetx;
  y=y-offsety;
  x=x/avgx;
  y=y/avgx;
/////////////////////////////////////////
```

```
  /*Serial.print("X=");
  Serial.print(x);
  Serial.print(", Y=");
  Serial.print(y);
  Serial.print(", Z=");
  Serial.println(z);*/

//////////////////////Four if statements that determine which quadrant you go into////////
  if(x>0&&y>0){//0 to 90
    heading = atan(y/x)*180/3.1415;
  }
  if(x>0&&y<0){//270 to 360
    heading = 360- atan(abs(y)/x)*180/3.1415;
  }
  if(x<0&&y<0){//180 to 270
    heading =  180 + atan(abs(y)/abs(x))*180/3.1415;
  }
  if(x<0&&y>0){//90 to 180
    heading = 180 - atan(y/abs(x))*180/3.1415;
  }

 /*
  Serial.print("X=");
  Serial.print(x);
  Serial.print(", Y=");
  Serial.print(y);
  Serial.print(", Z=");
  Serial.println(z);

  Serial.print("               Heading ");
  Serial.print(heading);
  Serial.println(" degrees");*/
  Serial.println(heading);
  delay(5000);

}
  calibrate=calibrate+1;////////////////////increments counter at end of loop
}
```

Appendix B: Bill of Materials

Description	Cost
HMC5883-L 3-axis digital compass module from parallax	$30
Standard Arduino UNO	$30
Breadboard shield or breadboard	$15-$1
Fifteen foot A-B USB cable	$5
Tupperware (size of Arduino and breadboard	$2
Hot glue gun plus glue	$5
Total	$87-$73