

# **Department of Electrical and Computer Engineering**

## **SENIOR DESIGN PROJECT**



### **Sail Assist: Electronic Aid System**

By

Carlos G Diaz  
Jorge Iscar  
Francisco J Villegas

**Final Report  
Spring 2013**

**Advisor: Professor Reuven Lask  
Tuesday April 30, 2013**

### **Abstract**

The purpose of this report is to provide information about our senior design project; what it does, how it works, and what it is for. We have designed a system that is not similar to anything currently available on the market that helps beginner sailors to accurately learn how to handle a sailboat. Basically the system, which uses an Arduino UNO™ programmable circuit, takes in all factors that affect the manning of a sailboat and, with the use of LED indicators, provides proper instructions on the proper placement of the main sailboat controls, the tiller and sail. We believe that this project, which has proved ambitious but feasible, has the potential to become an actual consumer product in the near future.

## Table of Contents

<b>1. INTRODUCTION .....</b>	<b>4</b>
<b>1.1 Specifications.....</b>	<b>4</b>
<b>1.2 Subprojects.....</b>	<b>5</b>
<b>2. DESIGN PROCEDURE .....</b>	<b>5</b>
<b>2.1 Design Decisions.....</b>	<b>5</b>
<b>2.2 Applicable Engineering Standards.....</b>	<b>6</b>
<b>3 DESIGN DETAILS.....</b>	<b>7</b>
<b>3.1 Equations and Implementation Diagrams .....</b>	<b>8</b>
<b>4. DESIGN VERIFICATION.....</b>	<b>14</b>
<b>4.1 Testing Procedure.....</b>	<b>14</b>
<b>4.2 Results.....</b>	<b>14</b>
<b>4.3 Conclusions and Incidents .....</b>	<b>17</b>
<b>5. COST .....</b>	<b>18</b>
<b>5.1 Parts.....</b>	<b>19</b>
<b>6. CONCLUSIONS.....</b>	<b>20</b>
<b>7. Appendix.....</b>	<b>21</b>
<b>REFERENCES.....</b>	<b>25</b>

## 1. INTRODUCTION

As a sailing instructor, team member Francisco Villegas has observed a need for an onboard system to help beginner sailors control the position of their sails and tillers while out at sea. The current way on-the-water teaching is done is with an instructor beside or near them on a motorboat, yelling instructions at the usually stressed out novices. Needless to say, this method is not optimal and could use some optimization. This is the main reason we decided to build our project. The Sail Assist, as we have called it, is a system that is installed on the sailboat that helps beginners sail without the need for an instructor. Using three compass sensors that input the heading of the boat, the sail direction and the wind direction, it outputs the optimal position for the different sailboat controls. The output is sent to an array of LEDs that will provide advice on the correct way to move each component in order to sail in the desired direction.

The project was not simple. It involved many different parts and aspects that needed to be made to work together in order to finish with a working product. The hardest part of the research portion of this project was finding the appropriate sensor components that would give us the signals we needed to process in order to tell the user what would be an optimal position for the sail and the tiller. The research we did paid off and we were able to find two circuits that did exactly what we needed. The most important component that we used was a magnetic compass that outputs a signal that could be processed to get a bearing in degrees. This product worked so well that we decided to adapt it to most of our other sensors and built the wind vane, boat bearing sensor and the sail bearing sensor using this same CMPS10 circuit.

### 1.1 Specifications

Our project encompasses several key aspects of design that come together in the end to form the desired system and generate the correct outputs. The mainframe of the entire concept is the software portion. For this part, our group came together and, applying knowledge, that came from both first hand experience in the field of sailing and sailing research, developed some equations that would take in variables such as the wind direction, desired sailing direction and the actual direction the boat was facing, would output an optimal location for both the sail and the tiller. These outputs are the most important part of the design for they will be processed and shown via our hardware implementation of LED displays so that the beginner sailor can correctly estimate the accurate positions of said components without the need of an instruction from someone who is not on the sailboat. From here, the rest of the project became a circuit building experiment, where we had to try different layouts of multiplexers and boards and ultimately make a decision to change the LED design into a different style that would lead to the optimal positioning of the components and to a user friendly experience for the sailors.

Our project proved to be entirely feasible in both design and implementation given the time constraints that were in place. During the course of the project, we made numerous changes, such as the LED displays, the solar power source addition and the

sensor choices, but in the end, the system created has all the capabilities we had wanted since the beginning and more. The main system consists of the three input sensors and the added push button, which provides a fourth input signal that sets the desired sailing direction. In order to obtain these signals, we created sensors using the compass circuits that are listed in the following sections, and used those readings to process and output a signal which tells the sailors where to move the sail and the tiller given changes in the wind direction, boat bearing or desired travel direction. In addition to the main objectives of the project, we were able to implement all of this using a solar source of energy for the entire system, which is extremely important considering the system is designed to be used at sea where there is no source of electrical power. Another additional component of the project was the casing of the system. Since it is designed to work on a boat, water resistant casing was extremely important to the design and implementation of a working prototype. We secured most circuitry inside clear plastic casing in order to protect the components and prolong the life of the product.

## **1.2 Subprojects**

Our project was divided into many different areas, or subprojects, that we needed to address. Each group member took care of some areas and we worked together in others, but in the end all parts came together to form the entirety of the project. The programming part of the project was hard because we had to make a program that took inputs from our different sensor circuits and processed them in order to obtain a suitable output that would update our LED displays accordingly. In addition to this, we had to work on the hardware portion of the project by wiring and testing our product to see that everything was working along with the software portion. This took us some time in order to adjust the software to better fit our values from the actual compass circuit as well as the other inputs we were using. In addition to these two main parts, many of the specifications in our project had to be developed independently, such as the casing, which had to be water resistant, or the power source for the circuits, which was made using a solar cell and battery. Finally, the combination of all the different parts into a system that works together to provide the necessary outputs along with the different needs that needed to be addressed by the project was done and we were able to develop a fully functional prototype for a project that could be turned into a commercial product.

## **2. DESIGN PROCEDURE**

### **2.1 Design Decisions**

The first major decision that was made by our group was which programmable microcontroller to use in order to achieve the desired integration of software and hardware. After some research, we made the decision to use the Arduino UNO™ microcontroller because of the user-friendly interface and the vast range of products and circuits that have been designed to integrate along with it. Besides, it will also allow us to build a real-time system. Since the programming language is very similar to C++, we were able to easily grasp most of the concepts or commands that we needed in order to program our system.

One of the decisions we had to make on the components was whether to build a wind vane or buy a digital one online [9]. The advantages to building it were mostly the costs and the added complexity of the overall project so in the end we made the choice to design a wind vane using our already compatible CMPS10 Tilt compensated compass [6].

We also had to decide on the power source we were going to use for the system. The simple choice was to make it battery operated and have the users be responsible for batteries but, as stated before, we added a new level of complexity and creativity to our design by adding a solar power source. With this solar power source, we not only add power, but we also remove one of the negative selling points which would be the tediousness of replacing batteries and possibly damaging a water resistant seals in the process. In addition to this, with technology making a shift towards green energy more and more each day, we secure that our project will be one to keep up with the times.

Finally, to input the designated direction we have decided, as we will explain later, to use a push button that will, using the electronic compass, take the current bearing direction and store it. Using this and our other inputs, our outputs can be calculated. Before we decided to use the button we had thought to use an LCD input display. This display would have allowed the sailor to input a heading degree but after brainstorming we realized that, given uneasy conditions, changing this heading while sailing would be very difficult. For example, say an instructor wanted a group of sailors to head in a certain direction, in order for all the students to set that heading the instructor would have to tell each of them the exact number to enter which can be very difficult to communicate. Therefore, the push button allows an easier general method of entering a designated direction.

## 2.2 Applicable Engineering Standards

In keeping with the Code of Ethics for Engineers, we have made some decisions regarding our project that have influenced it both negatively and positively throughout the process. One of the main points of ethics is the coding we have used, for it has been influenced with pieces of other, unrelated, projects we have found during our research. As we have looked at and used some of the same methods used in these other projects, we have given the appropriate acknowledgement in our reports. The reason why we looked for guidance in these other projects was because others have used the same components that we had selected, which were difficult to simply figure out while integrating to Arduino UNO™ circuits.

In addition to the intellectual ethics, we also tried to maximize user safety and security by making the water resistant casing a big part of the project. This addition greatly decreased the risks of the product being damaged, misleading users or malfunctioning and causing a problem for the drivers. In addition to this, by securing the main product in a water resistant case, we make the product more durable and sturdy to withstand constant use and endure the wear and tear that comes with an outdoor product.

### 3 DESIGN DETAILS

The main component we are using, as we have previously stated, is an Arduino UNO™ microcontroller. We are using this as our main controller for our project and for the processing of variables to generate the corresponding outputs. The programming of the Arduino UNO™ is done very similarly to C++ and we were able to use this to our advantage by using previous academic knowledge. The simplicity of the interface and the appropriate speed of processing were also selling points in choosing this microcontroller. In addition to this, the Arduino UNO™ has a very low power consumption. This has helped us, since we are using an alternative power source and high voltages and currents are hard to obtain.

Our main sensor is the CMPS10 digital compass. This compass is a circuit that uses magnetic north as its reference point to output its relative direction. Three of these compasses are used to establish the direction of the sailboat, wind, and position of sail. Given the high accuracy of this component, we have been able to use the exact direction of travel. In addition to the bearing, which is the main variable that we are looking to process, the compass also accounts for tilt, which is ideal for sailing considering sailboats heel over with strong winds. The compass has three communication interfaces: I2C, PWM, and Serial. While using the I2C interface you are able to obtain not only the bearing but also the tilt, pitch, and roll as well [5]. However, because we have decided to use this for all three input variables, we are using the PWM interface.

As discussed above, we have designed our own digital wind vane. While we do feel this is something that could be upgraded, financially and theoretically, we found that building our own would suffice. Built from a PVC pipe butt, slip ring, plastic, and a CMPS10 compass, we successfully built a working prototype. We knew we needed something that could spin freely without tangling which is why we purchased a slip ring. The ring, allowing up to six wire connections, allowed us to connect a RJ-11 coupler that was, as was the slip ring, fit snug and sealed into the PVC pipe butt. Attached to the top of the slip ring is a plastic tube and wind vane. Under the wind vane, our compass was Velcro-ed and connected with four wires protruding from the plastic tube.

Another component we are implementing in our project is a simple push button. We are using this button to input the designated heading direction. Upon the push of this button, the microcontroller simply holds the current value for the compass and save it as the designated direction value. This value will be compared to the current direction of components providing the user with relative output results.

Returns High time in Microseconds for direction

```
cb = pulseIn(9, HIGH);
cs = pulseIn(10, HIGH);
cw = pulseIn(11, HIGH);
```

Converts High time into Degree reading

```
cmpbear = (((cb/1000)-1)*10);
cmpsail = (((cs/1000)-1)*10);
cmpwind = (((cw/1000)-1)*10);
```



Figure 1

Above is a graph and command that is used in retrieving the compass direction. The compass sends a low signal for sixty-five milliseconds and following that sends a high signal for a time relative to the direction. “pulseIn” is an Arduino UNO™ programming command used to read the time of a pulse which we have used to find this time. Once the time is retrieved, we manipulate the value and acquire the actual degree heading from zero to three hundred sixty degrees.

### 3.1 Equations and Implementation Diagrams

Using our previous experience and sailing physics we have deduced the following equations for the sail position [1], [2]:

$$x = WD - SD$$

$$\text{No Go Zone} \quad 320 < x \leq 40$$

$$SP = \left( \frac{x - 40}{10} \right) \cdot 9 \quad 40 < x \leq 90$$

$$SP = \left( \frac{x - 90}{2} \right) + 45 \quad 90 < x \leq 180$$

$$SP = - \left( \left( \frac{270 - x}{2} \right) + 45 \right) \quad 180 < x \leq 270$$

$$SP = - \left( \left( \frac{320 - x}{10} \right) \cdot 9 \right) \quad 270 < x < 320$$

Equation Group 1

For the tiller:

$$TD = 0$$

$$SD = DD$$



$$TD = -SD \quad \text{Otherwise}$$

Equations Group 2

$WD$  stands for the wind direction,  $SD$  for the sailboat direction,  $SP$  for the sail position,  $TD$  for tiller position and  $DD$  for designated direction.

In the following lines, we describe all the parts involved in the project using diagrams.

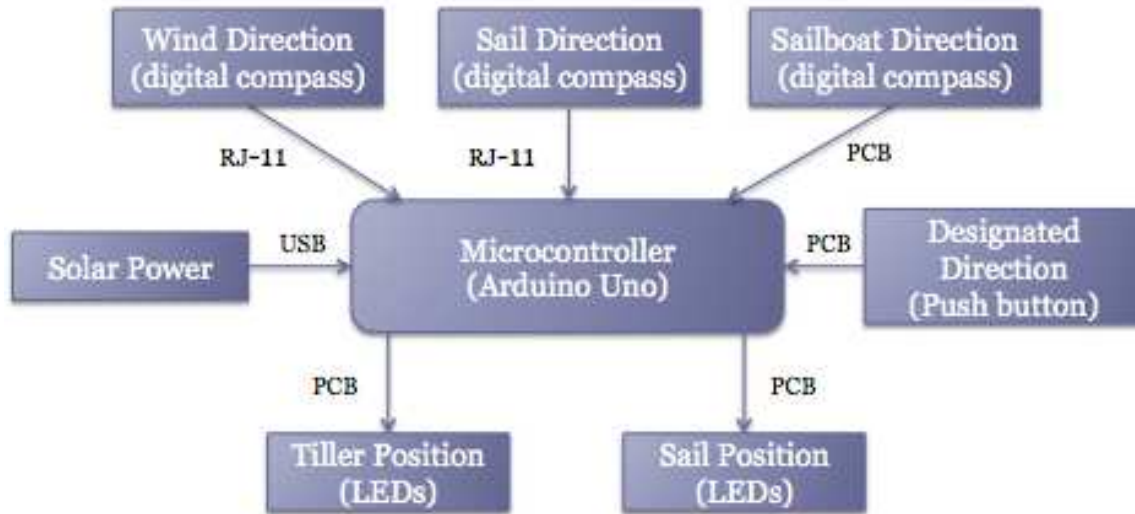


Figure 2

Figure 2 shows the general implementation of the system. As stated previously, the main component of the project is the microcontroller (Arduino UNO™). Our three sensors (wind, sail and heading directions) will input information to the Arduino UNO™ and using this data the microcontroller will output the tiller and sail position using LEDs displays. All the components are powered by the microcontroller and the microcontroller itself will be powered by the solar power system.

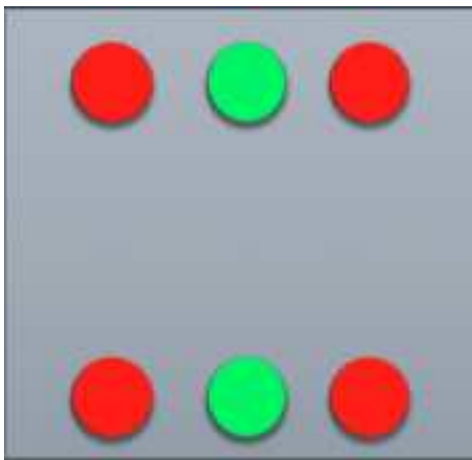


Figure 3

For the display implementation we are going to use something similar to the diagram in Figure 3. We used 3 LEDs for both the sail and tiller display; therefore, we needed 3 wires per display. Each LED means a different command. The red LEDs indicate the user to move the control component either left or right, while the green one means the component in question is in the correct position given the sensor inputs.

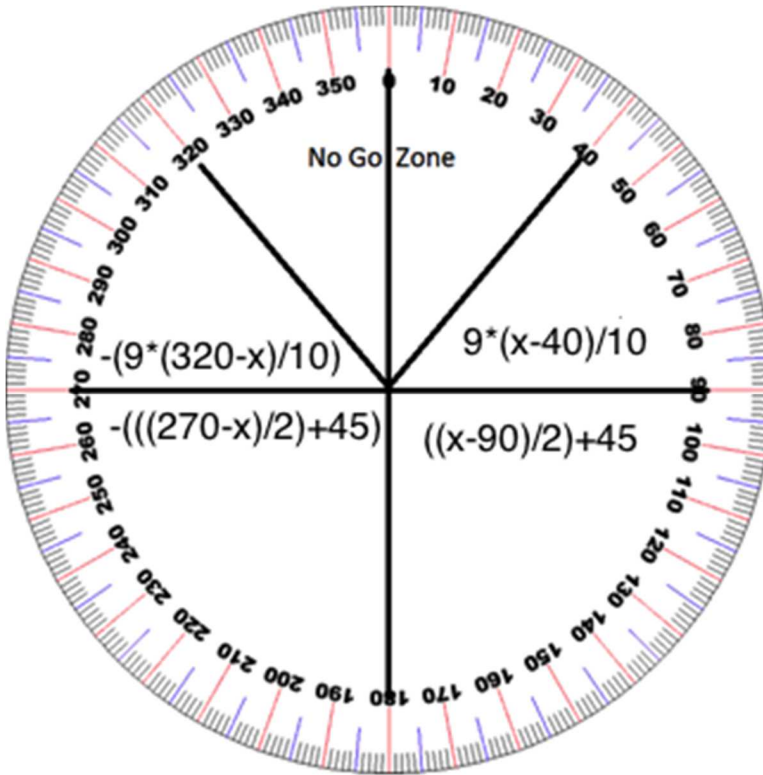


Figure 4

In this figure, the equations for the sail position are provided for a compass where the degrees are normalized to the position of the boat, which means the degrees on the compass are the difference between the wind and boat direction. We also can see the No Go Zone, which is the zone where sailing is not possible given sailing direction and wind direction.

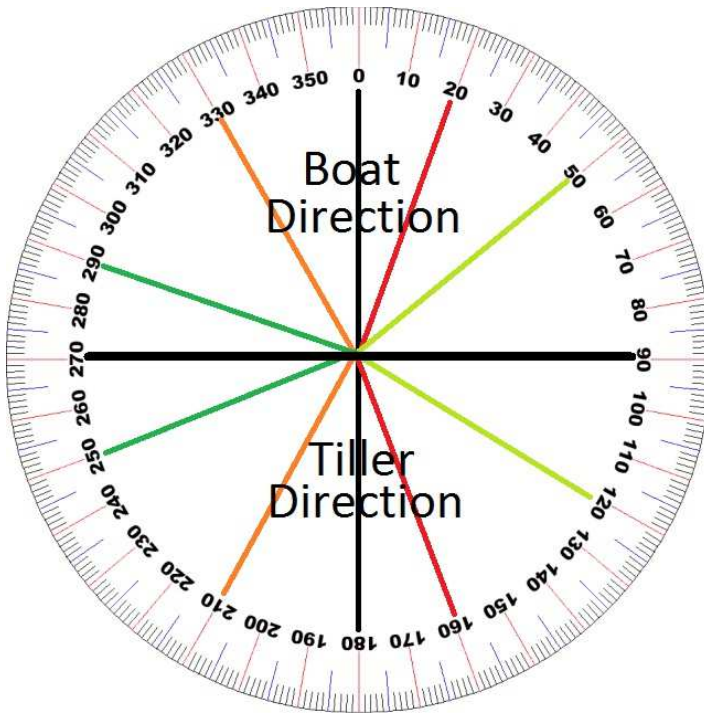


Figure 5

This compass provides a visual explanation of our equations regarding the tiller direction relative to the boat direction.

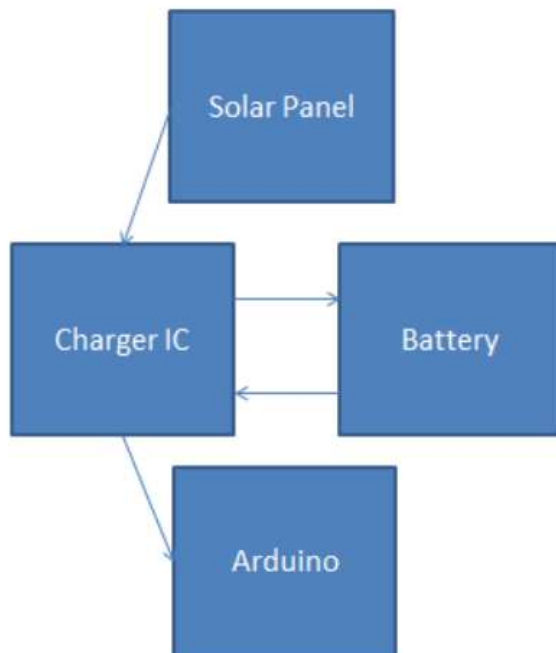


Figure 6

The solar power system [3], as we can see in the figure 6, was implemented using three components. The solar panel outputs a constant voltage of 6 volts to the charger IC and this circuit uses that power to charge the battery. However, these batteries work with 3.7

volts, which means the charger IC will also convert that voltage to 5 volts (USB) to power the Arduino UNO™. The IC also allows us to charge the battery using a mini-USB instead of using the solar panel. It is important to say that the higher the Ampere\*hour parameter of the battery is, the longer the autonomy of the system will be.

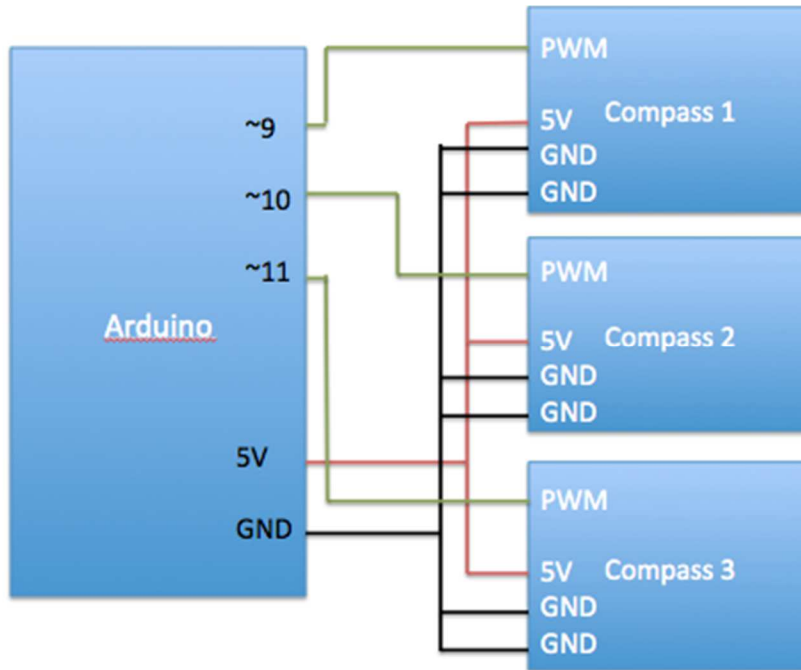


Figure 7

In figure 7 we show the connections between the compasses and the Arduino UNO™. In the end, because of cost and simplicity, we decided to use the CMPS10 compass to build every sensor that was necessary for this project, which means that each sensor was wired the same way to different ports of the Arduino UNO™ and processed using pulse width modulation. The physical design of the sensor components was different due to the distinct measurements that needed to be performed by each one, but the connections and the signal processing was the same for each one.

In order to achieve all the connections necessary for each compass, we created common rows in a PCB with both ground and 5V connections. From these common rows, we wired each of the compasses with the 5V and ground inputs that they required for operation. In addition to this, each compass output was wired to one of the Pulse Width Modulation input ports in the Arduino UNO™.

This entire idea is one of the aspects of the project that has changed the most since the initial stages. At first, the idea was to use a specialized wind vane that was compatible with the Arduino UNO™, however, the cost of this component was extremely high, which is why we decided to build a custom wind vane using the CMPS10.

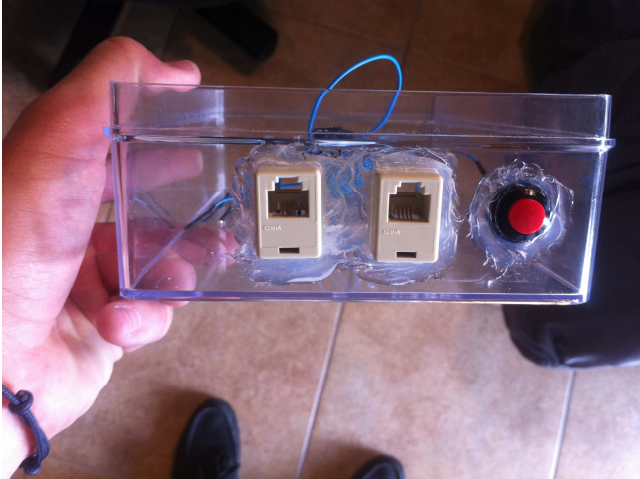


Figure 8

Because of the fact that the push button works as a switch, we have simply connected it to a voltage output on one side and to a digital input on the other side [4]. Therefore, when the sailor pushes the button, it will close the circuit and we will be able to set the designated direction. As can be seen in figure 8, the push button is located on the side of the main encasing and is based on simplicity. The sailor's bearing can be altered throughout the entire process and the system will adapt to the new bearing once the button is pushed.



Figure 9

The water resistant casing, seen in figure 9, is a very important part of this project as we mentioned in section 2.2. For the main components we used a transparent water resistant box. To input the data from the wind vane and sail direction sensor, we included two external connections that can be seen in figure 8. Finally, for the display, we used the box as the place where the LEDs were located. It is important that the casing be clear, in order to allow for the user to have clear visibility of the display housed within. To ensure the water-resistant feature we used silicone rubber around each hole.

## 4. DESIGN VERIFICATION

In the following lines we will explain how we checked and tested the correct operation of the system. We will start by explaining how we tested the system, show the results and finally conclude with the problems we encountered throughout the completion of the project.

### 4.1 Testing Procedure

To test the correct functioning of the system we have used two different procedures. The first one was done using the serial interface provided by the Arduino UNO™ connected to the computer. With this interface we were able to check the inputs (the readings from the compasses and the desired direction) and the outputs (the sail direction relative to the sailboat). Using an external magnetic compass, we checked that the readings from all our digital compasses matched and that the sail direction relative to the sailboat was correct (taking into account our equations). For the second procedure we used the LEDs displays. We set different scenarios by changing the sailboat direction and maintaining the wind direction. The LEDs display for the sail position provided the correct sail direction given the inputs in the scenario. To test the LED display for the tiller we just set a desired direction by pushing the button. This action would set the desired direction and the boat direction to the same value. By changing the direction of the sailboat, we observed how the display indicated the necessary movement to the tiller in order to return the box to the desired direction.

In conclusion, with the first procedure we could check that the compasses were working and the readings were correct. We also verified the functioning of the push button and that the outputs provided by the microcontroller were correct. With the second procedure, we checked the operation of the LEDs displays and verified that the equations were accurate.

### 4.2 Results

Using the Arduino UNO™-Computer serial interface we obtained the following results that match the desired results:

```
x = 20.75 sp = 0.00 cmpbear = 321.65 cmpsail = 136.72 realsp = 321.65 cmpwind = 300.90 cmpdirx = 0.00
x = 20.83 sp = 0.00 cmpbear = 321.53 cmpsail = 136.57 realsp = 321.53 cmpwind = 300.70 cmpdirx = 0.00
x = 20.21 sp = 0.00 cmpbear = 321.32 cmpsail = 136.32 realsp = 321.32 cmpwind = 301.11 cmpdirx = 0.00
x = 20.12 sp = 0.00 cmpbear = 321.15 cmpsail = 136.32 realsp = 321.15 cmpwind = 301.03 cmpdirx = 0.00
x = 20.44 sp = 0.00 cmpbear = 321.31 cmpsail = 136.15 realsp = 321.31 cmpwind = 300.87 cmpdirx = 0.00
x = 20.56 sp = 0.00 cmpbear = 321.46 cmpsail = 136.41 realsp = 321.46 cmpwind = 300.90 cmpdirx = 0.00
x = 18.60 sp = 0.00 cmpbear = 319.65 cmpsail = 136.57 realsp = 319.65 cmpwind = 301.05 cmpdirx = 0.00
x = 18.48 sp = 0.00 cmpbear = 319.65 cmpsail = 136.72 realsp = 319.65 cmpwind = 301.17 cmpdirx = 0.00
x = 18.50 sp = 0.00 cmpbear = 319.61 cmpsail = 136.40 realsp = 319.61 cmpwind = 301.11 cmpdirx = 0.00
x = 18.59 sp = 0.00 cmpbear = 319.67 cmpsail = 136.45 realsp = 319.67 cmpwind = 301.08 cmpdirx = 0.00
x = 18.56 sp = 0.00 cmpbear = 319.60 cmpsail = 136.57 realsp = 319.60 cmpwind = 301.04 cmpdirx = 0.00
```

Figure 10



In this picture we can observe different inputs and outputs in real time (each row represents the inputs and outputs at different instants). For example, for the first row:

$$x = cmpbear - cmpwind = 321.65^\circ - 300.90^\circ = 20.75^\circ \rightarrow$$

$$320^\circ < x < 40^\circ \rightarrow$$

No Go Zone ( $sp = 0^\circ$ )

For the second procedure, these are the scenarios we set:

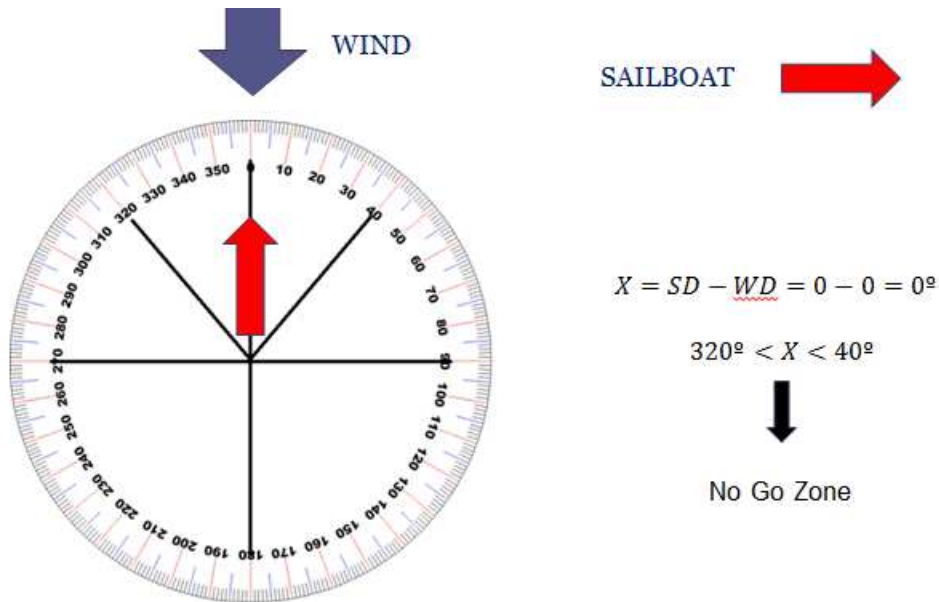


Figure 11

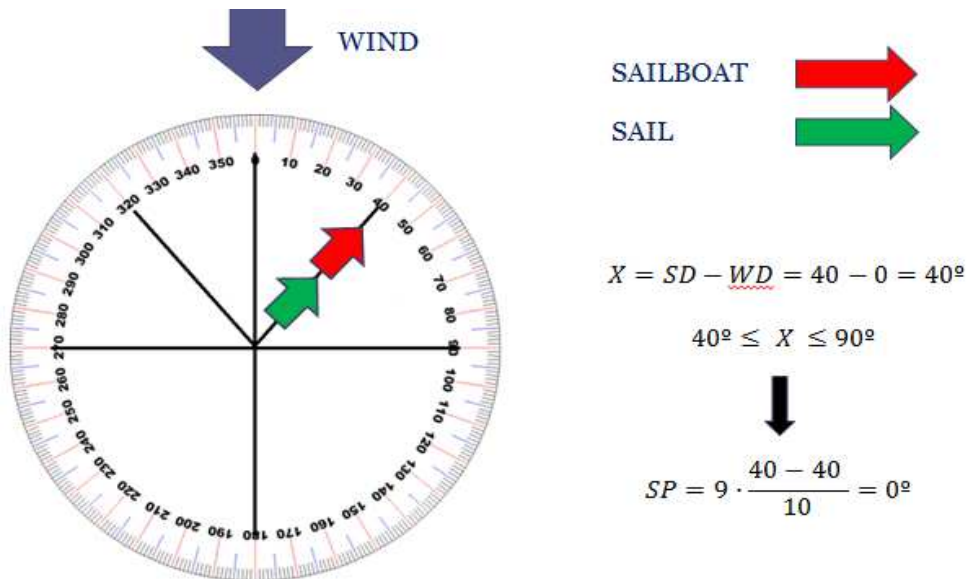


Figure 12

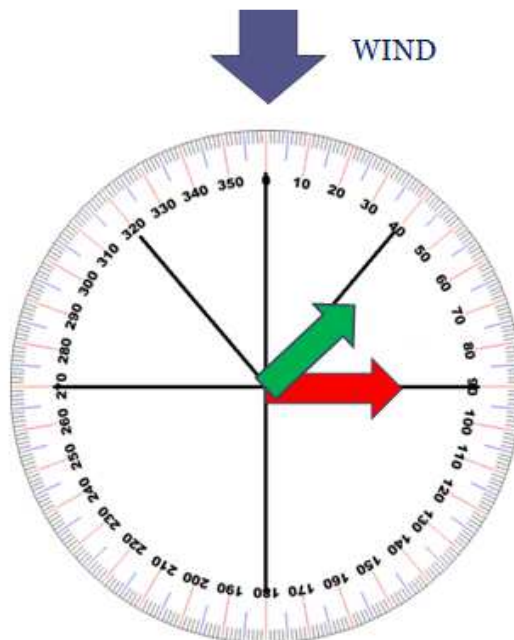




Figure 13

SAILBOAT   
SAIL 

$$X = SD - \underline{WD} = 90 - 0 = 90^\circ$$

$$40^\circ \leq X \leq 90^\circ$$



$$SP = 9 \cdot \frac{90 - 40}{10} = 45^\circ$$

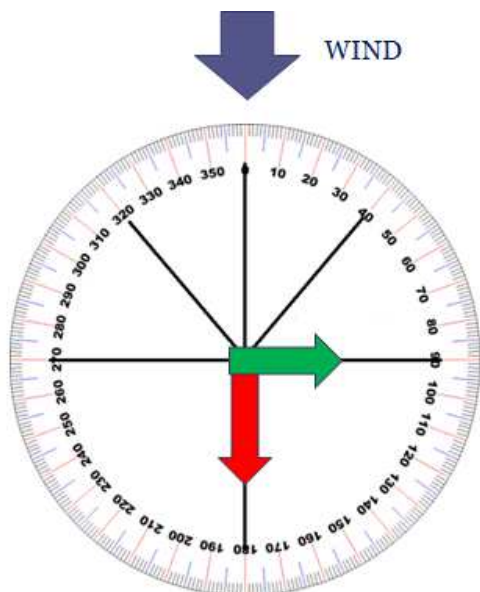


Figure 14

SAILBOAT   
SAIL 

$$X = SD - \underline{WD} = 180 - 0 = 180^\circ$$

$$90^\circ < X \leq 180^\circ$$



$$SP = \frac{180 - 90}{2} + 45 = 90^\circ$$



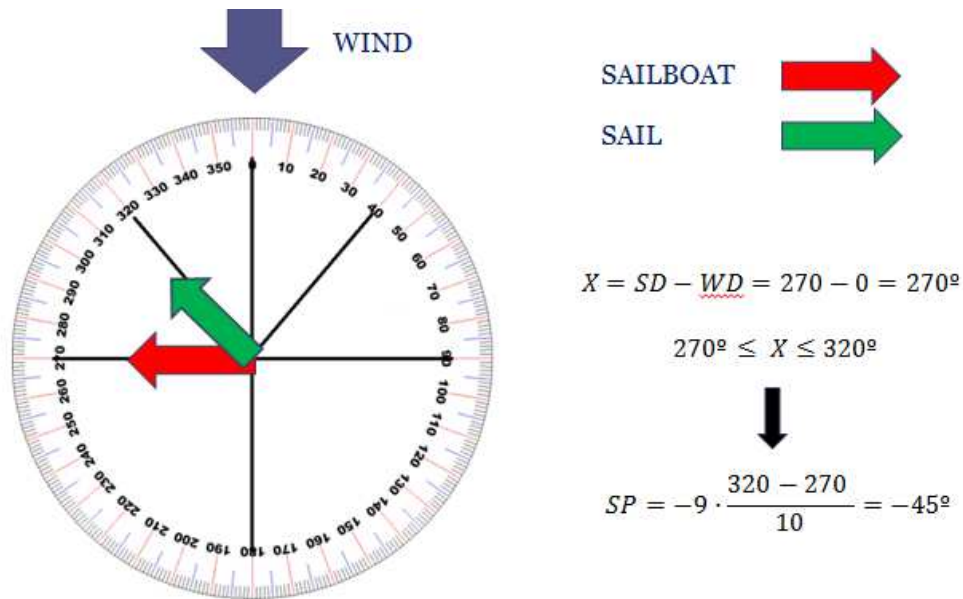


Figure 15

In the all the scenarios the display indicated the sail direction correctly.

It is also important to mention that the solar power system performed perfectly, even providing a greater autonomy to the components

### 4.3 Conclusions and Incidents

In both procedures, by using the serial interface with the computer and observing the LED displays, we obtained the desired results. However, we found some problems during this evaluation that finally were solved:

1. We had to change the initial code twice: in the beginning, we computed  $x$  as  $x = |\text{wind direction} - \text{sailboat direction}|$ . However, using this equation the system did not work accurately. Finally, we figured out that the value for  $x$  was wrong and the correct one was  $x = \text{sailboat direction} - \text{wind direction}$  and if  $x < 0$ ,  $x = 360 + x$ . Another problem we had happened in the code for the LED displays. The decision of when we had to turn on the left or right red LED was more complex than expected and we had to re-design that part of the code.
2. As we said in the report, we used the RJ-11 standard cable to connect the wind vane and sail compass to the Arduino UNO™. In the beginning we connected the pins in the sail compass to the plug in that box and the pins in the Arduino UNO™ to the RJ-11 plug in the main box according to the coupler wire color coordination but it turned out that each RJ-11 plug uses different colors for the connections. Therefore, the pins in the microcontroller did not match the pins in

- the compass. We also encountered this problem with the wind vane. We used a voltmeter to check these connections and make the appropriate adjustments.
3. Since the compass we are using does not include a case, sometimes, due to the sensitivity of the circuitry, when we touched the compass we interrupted the correct functioning and we had to restore it.
  4. The last important problem we had in the test was that we found that the compass in the main box was receiving interference from the USB power cable and other components. Since the compass works using electromagnetic properties, the different currents flowing through the wires in the box would cause problems because of the EM waves that they generated. This was noticed by checking the No Go Zone and we figured out that the consequence of this interference was a loss of precision in this compass (about 5°). To decrease this interference, we changed that wire, which reduced the loss in precision.

## 5. COST

The costs incurred in the building of our senior design project are the following:

Part	Quantity	Cost	
Compass (cmps10)	3	\$37.08	\$111.24
Jumpers	1	\$6.99	\$6.99
RJ-11 Coupler	2	\$1.95	\$3.90
RJ-11 Cable	2	\$0.99	\$1.98
Solar Power System	1	\$40.00	\$40.00
Switch	1	\$2.99	\$2.99
Push Button	1	\$2.99	\$2.99
PVC Cap	1	\$0.89	\$0.89
Resistors	7	\$0.10	\$0.70
LEDs (2pc)	3	\$1.99	\$5.97
Connection Bars	1	\$1.35	\$1.35
Velcro	1	\$2.97	\$2.97
Nuts and Bolts	4	\$0.50	\$2.00
USB Cable	1	\$4.00	\$4.00
Tiewraps	1	\$4.99	\$4.99
Heat Shrink Wrap	1	\$4.99	\$4.99
Plastic Case	1	\$4.99	\$4.99
Foam Sheets	2	\$0.05	\$0.10
PCB	1	\$2.49	\$2.49
Sealant	1	\$6.95	\$6.95
Arduino UNO™	1	\$29.99	\$29.99
Rotating Connection	1	\$17.50	\$17.50

Plastic Cards (WV)	1	\$0.50	\$0.50
Plastic Rod (WV)	1	\$0.20	\$0.20
		<b>Total</b>	<b>\$260.67</b>

Table 1

In the event of mass production, the production price would be significantly lower given the lower margins in large purchase as well as the specialization of different parts such as the Arduino UNO™ or the compasses.

## 5.1 Parts

Part	Make/Brand
Compass (cmpr10)	Devantech
Jumpers	Radio Shack
RJ-11 Coupler	Radio Shack
RJ-11 Cable	Radio Shack
Solar Power System	Seeed Studio
Switch	Radio Shack
Push Button	Radio Shack
PVC Cap	Shell Lumber
Resistors	Radio Shack
LEDs (2pc)	Radio Shack
Connection Bars	Radio Shack
Velcro	Shell Lumber
Nuts and Bolts	Shell Lumber
USB Cable	Arduino
Tiewraps	Shell Lumber
Heat Shrink Wrap	Radio Shack
Plastic Case	The Container Store
Foam Sheets	Shell Lumber
PCB	Radio Shack
Sealant	Shell Lumber
Arduino UNO™	Arduino
Rotating Connection	Adafruit
Plastic Cards (WV)	Shell Lumber
Plastic Rod (WV)	Shell Lumber

Table 2

## 6. CONCLUSIONS

Sail Assist: Electronic Aid System provides an alternative to the usual instructor interaction method of teaching sailing. Although this method of teaching works, it can be very expensive and requires the knowledge and instruction of another person. Our Electronic Aid System will assist sailors and eliminate the need for an instructor, coach boat, and lesson planning. It allows you to learn sailing at your own leisure, whenever you want. Both methods can also be used if found necessary, this will provide extra help and possibly a better understanding of what an instructor is teaching. Sailing lessons range in price but are never cheap. If our product were to be mass-produced, our alternative to learning how to sail would not only be cheaper but also allow multiple uses by multiple users.

The system we have created is quite complex and considers multiple variables and conditions. Designed to withstand splashes and rough conditions, the system has become a sturdy and reliable product. We made sure to implement a plug-and-play design that allows easy but reliable installation and removal on any sailboat model. Many minute details were taken into consideration and built accordingly including the constant sail movement, tilt of boat, and the importance of keeping your eyes looking ahead. There were multiple components in this project to be built including the enclosure, solar power system, main controller, electrical board, and all of them have been designed and placed wisely and with caution. We also ensure that it is water-resistant in case some water is splashed on the system or the humidity itself does not harm the main components. Finally, the most important part of the project, which is the product itself, was fully completed and presented. This is important because we have seen many good ideas die in the planning or production stages but the fact that this is a finished product gives our project a greater sense of credibility and us a greater sense of achievement and closure.

## 7. Appendix

### Compass Signal Processing and LED Display

```
// Pulse width modulation design for all three compasses
// Compass A, B, and C

float cb, cs, cw, x, cmpbear, cmpsail, cmpwind, cmpdirx, sailless,
sailmore, sailless_tiller, sailmore_tiller;
float sp = 0, realsp;
int button, sdisp, tdisp, left, right, middle, all, lsl, lsm, lsr;
int acc_tiller = 5, acc = 5; // level of precision for sail optimization

void setup()
{
  Serial.begin(9600); // Don't think this is necessary
  // Sail LED outputs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  // Tiller LED outputs
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  // Button input
  pinMode(8, INPUT); // this may need to digital?
  // Compass inputs
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
}

void loop()
{
  // Returns High time in Microseconds for direction
  cb = pulseIn(9, HIGH);
  cs = pulseIn(10, HIGH);
  cw = pulseIn(11, HIGH);
  // Converts High time into Degree reading

  cmpbear = (((cb/1000)-1)*10);
  cmpsail = (((cs/1000)-1)*10);
  cmpwind = (((cw/1000)-1)*10);
  // Command that takes a button input to set the heading
  button = digitalRead(8); // read input value // THIS MAY STOP THE
  LOOP AND WAIT TILL PUSHED
  if (button == HIGH)
  {
    cmpdirx = cmpbear;
    //cmpdirx = 300; //Example

    // Sail position equations copied from old code
    x = cmpbear - cmpwind;
    if(x<0)
      x=360+x;

    // This is used for comparing sail positions to adjust the resolution
    but we need to make it relevant to bearing

    sailless_tiller = cmpdirx - acc_tiller;
    if(sailless_tiller<0)
      sailless_tiller=sailless_tiller+360;
  }
}
```

```

    sailmore_tiller = cmpdirx + acc_tiller;
    if(sailmore_tiller>360)
        sailmore_tiller=sailmore_tiller-360;

    if(320<x || x<40) // Turns on all LEDs in NoGoZone // DO WE WANT THE
    Leds TO DIM AS THEY GET CLOSER TO GREEN?
        sp = 0;
    else if(40<=x && x<=90)
        sp = (((x-40)/10)*9);
    else if(90<x && x<=180)
        sp = (((x-90)/2)+45);
    else if(180<x && x<270)
        sp = -(((270-x)/2)+45);
    else if(270<=x && x<=320)
        sp = -(((320-x)/10)*9);

    // This makes the the desired sailposition relevant to the boat which
    can then be used to compare with cmpsail

    realsp = cmpbear - sp;

    if(realsp<0)
        realsp=360+realsp;
    if(realsp>360) // sp may be negative
        realsp=realsp-360;

    sailless = realsp - acc;
    if(sailless<0)
        sailless=sailless+360;
    sailmore = realsp + acc;
    if(sailmore>360)
        sailmore=sailmore-360;

    if(320<x || x<40) // Turns on all LEDs in NoGoZone // DO WE WANT THE
    Leds TO DIM AS THEY GET CLOSER TO GREEN?
        sdisp = 3;
    else
        sdisp = saillights(realsp);

    // Sail LEDs //
    // Switch Cases that turns on the LEDs
    switch (sdisp) {
    case 0: // 0 left
        { lsl = HIGH;
          lsm = LOW;
          lsr = LOW; }
        break;
    case 1: // 1 right
        { lsl = LOW;
          lsm = LOW;
          lsr = HIGH; }
        break;
    case 2: // 2 middle
        { lsl = LOW;
          lsm = HIGH;
          lsr = LOW; }
        break;
    case 3: // 3 all
        { lsl = HIGH;
          lsm = HIGH;
          lsr = HIGH; }
        break;
    }

```

```

//This emits the LEDs
digitalWrite(2, lsl);
digitalWrite(3, lsm);
digitalWrite(4, lsr);

// Tiller LEDs //
// Switch Cases that turns on the LEDS

sdisp = tillerlights(cmpbear);

    switch (sdisp) {
case 0:    // 0 left
    { lsl = HIGH;
      lsm = LOW;
      lsr = LOW; }
    break;
case 1:    // 1 right
    { lsl = LOW;
      lsm = LOW;
      lsr = HIGH; }
    break;
case 2:    // 2 middle
    { lsl = LOW;
      lsm = HIGH;
      lsr = LOW; }
    break;
}
//This emits the LEDs
digitalWrite(5, lsl);
digitalWrite(6, lsm);
digitalWrite(7, lsr);

Serial.print("x = ");
Serial.print(x);
Serial.print(" sp = ");
Serial.print(sp);
Serial.print(" cmpbear = ");
Serial.print(cmpbear);
Serial.print(" cmpsail = ");
Serial.print(cmpsail);
Serial.print(" realsp = ");
Serial.print(realsp);
Serial.print(" cmpwind = ");
Serial.print(cmpwind);
Serial.print(" cmpdirx = ");
Serial.print(cmpdirx);
Serial.print("\n");
}

int saillights(float realsp)
{
    if (sailless>sailmore){
        if (sailless <= cmpsail || sailmore >= cmpsail)
            return 2;
        else if (realsp<cmpsail){
            if ((cmpsail-realsp)<(360-cmpsail+realsp))
                return 1;
            else
                return 0;
        }
        else if (realsp>=cmpsail){
            if ((realsp-cmpsail)<(cmpsail+360-realsp))

```

```

        return 0;
    else
        return 1;
    }
}
else if (sailless<sailmore){
    if (sailless <= cmpsail && sailmore >= cmpsail)
        return 2;
    else if (realsp<cmpsail){
        if ((cmpsail-realsp)<(360-cmpsail+realsp))
            return 1;
        else
            return 0;
    }
    else if (realsp>=cmpsail){
        if ((realsp-cmpsail)<(cmpsail+360-realsp))
            return 0;
        else
            return 1;
    }
}
}
}

int tillerlights(float cmpbear)
{
    if (sailless_tiller>sailmore_tiller){
        if (sailless_tiller <= cmpbear || sailmore_tiller >= cmpbear)
            return 2;
        else if (cmpdirx<cmpbear){
            if ((cmpbear-cmpdirx)<(360-cmpbear+cmpdirx))
                return 1;
            else
                return 0;
        }
        else if (cmpdirx>=cmpbear){
            if ((cmpdirx-cmpbear)<(cmpbear+360-cmpdirx))
                return 0;
            else
                return 1;
        }
    }
    else if (sailless_tiller<sailmore_tiller){
        if (sailless_tiller <= cmpbear && sailmore_tiller >= cmpbear)
            return 2;
        else if (cmpdirx<cmpbear){
            if ((cmpbear-cmpdirx)<(360-cmpbear+cmpdirx))
                return 1;
            else
                return 0;
        }
        else if (cmpdirx>=cmpbear){
            if ((cmpdirx-cmpbear)<(cmpbear+360-cmpdirx))
                return 0;
            else
                return 1;
        }
    }
}
}
}

```



## REFERENCES

- [1] <http://syr.stanford.edu/SAILFLOW.HTM>
- [2] [http://ffden2.phys.uaf.edu/211\\_fall2002.web.dir/josh\\_palmer/basic.html](http://ffden2.phys.uaf.edu/211_fall2002.web.dir/josh_palmer/basic.html)
- [3] [http://www.seeedstudio.com/wiki/Lipo\\_Rider\\_V1.1](http://www.seeedstudio.com/wiki/Lipo_Rider_V1.1)
- [4] <http://arduino.cc/en/tutorial/button>
- [5] [http://www.robot-electronics.co.uk/htm/arduino\\_examples.htm#Tilt Compensated](http://www.robot-electronics.co.uk/htm/arduino_examples.htm#Tilt Compensated)
- [6] <http://nodna.de/Devantech-CMPS10-CMPS-10-Magnetic-Compass-Compass>
- [7] <https://www.adafruit.com/products/481>
- [8] <https://www.adafruit.com/products/341>
- [9] [http://www.davisnet.com/weather/products/weather\\_product.asp?pnum=07911](http://www.davisnet.com/weather/products/weather_product.asp?pnum=07911)