Optimized Sailing Trajectories:
A Bluetooth Network of Sailboat Sensors

Joseph Skinner, Lucas Hartlage, Ramess Rogers
Dept. of Computer Engineering
University of California Santa Cruz

***Abstract*—This paper discusses the design and implementation of a sailboat sensor package. The prototype is designed to be implemented on the *RS Quest* dinghy sailboat with views to expand to other boats at reasonable hobbyist cost. In collaboration with a computer science team, data is to be passed to a mobile phone application for processing, optimal routing, and display. The computer engineering team has designed, tested, and implemented the sensor packages and established system-wide Bluetooth communication. Rough sail trim is read by a custom-built boom angle sensor. Boat velocity, attitude, course, and heading are determined from the *BerryGPS-IMUv3* and *MPU-9250*. The apparent wind vector is read from the *Davis 7911* anemometer/wind vane. In conjunction with data from the GPS and IMU, polar speed plots, and the triad of wind vectors are calculated to confirm validity of data.**

I. Introduction/Motivation

A. Background

This project began as an attempt to improve the abilities of novice sailors in a learning environment. On a small vessel, it can be difficult to fit everyone on board, which may mean students are left to travel under their own supervision. The instructor's duties are made strenuous by having to communicate with students from a distance. As a result, new sailors may find their learning progresses more slowly than it would with experience out in deeper waters. At the same time, experienced sailors may find their progression decreases as improvements in technique become less obvious. Having a system that guides a crew into optimal paths would greatly help them improve their technique and performance.

B. Significance

In the process of learning to sail buoy courses, two of the team members discussed navigation strategies. Through several races and much trial and error, their techniques improved. However, there was a problem to solve – true and apparent wind vectors, as well as the sail's angle of attack in relation to apparent wind, were subject to human error. Thus, their route was only as good as their sailing skills and judgement allowed. A sensor system was proposed to measure the triad of wind vectors and the sail's angle of attack. This developed into ideas about what makes an ideal navigation algorithm. With known marks, wind vectors, and the potential speed of our boat at different points of sail, an optimal course could be deduced.

A very skilled sailor may be able to do this naturally, but novice and even experienced sailors could benefit from knowing their course with respect to a derived optimum. Being both sailing and engineering students, there is a desire to improve our results with quantitative verification. Competitive and even some casual sailors also have this desire. Overall, simulated conditions and projected course routes would benefit any student of sailing.

C. Solution

To calculate an optimal sailing route, a wide range of environmental data must be taken into account. That data includes apparent and true wind speed/direction, boat velocity, the boat wind vector, boat heel, and GPS position. An anemometer, wind vane, IMU, GPS, and boom angle potentiometer were used to provide such data. The IMU, GPS, anemometer, and wind vane were operated by a *uC32* at the top of the mast, while the boom angle sensor was operated by a different *uC32* mounted on the side of the mast.

After all the data was collected and processed, it was sent over Bluetooth to a custom app on the user's mobile device. The mobile device then runs the best route algorithm using a wind vector map data from online, as well as the data supplied by the sensors, and displays the route in a convenient format for the user. The app also displays boom angle and wind vector information.

Since this project required much data processing and relied on the accuracy of such data, modular testing was done at all steps. By using fans, mounting the wind sensors on cars, and mounting the sensor packages on the RS Quest, we were able to test wind speed and direction data collection. This later provided a control for reference when testing data processing and offset calculation by adding a rocking motion to the same tests. After mounting the sensor package on the boat, testing was still done, but prior tests confirmed the reliability of the data.

D. Prior Art

Using the capabilities of computers to guide sailboats is not a new concept. Within the broader category of boats, there have been numerous fully-autonomous vessels successfully designed to travel vast distances without human intervention. The SeaCharger is a prime example. The 8-foot long, solar-powered miniature vessel was reported to be able to travel from Half Moon Bay in California across the Pacific to Hawaii [2]. The vessel makes use of standard hobbyist motors and servos, as well as an Arduino Mega to act as the logical center. According to the designer, sensors were kept to a minimum for cost and complexity reduction [2]. The owner was able to monitor the SeaCharger's position on their phone in a similar way to what our project intends to do with the user's phone.

At the other end of the spectrum, researchers at the Swiss Federal Institute of Technology in Zurich were able to build a fully autonomous sailboat with the ability to calculate and follow the shortest path to its destination [3]. The boat uses a unique rudder and keel design to increase stability in harsh weather conditions. Sensors similar to ours were also used to provide data back to its primary algorithms. The boat is able to follow a predefined path as efficiently as possible, and it is here the path of our own project diverges [3]. Our project calculates the fastest route and updates it on the go as new information arises from the sensors. Additionally, while the autonomous sailboat project used more expensive materials such as carbon fiber on a newly-designed boat [3], our project will use cheaper, easily accessible sensors and components similar to the SeaCharger, yet still be compatible of installation on most contemporary sailboats.

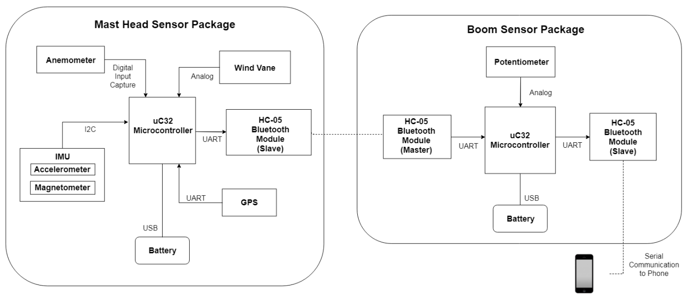


Fig. 1. *System Diagram of Sensors and Components*

II. Anemometer and Wind Vane

A *Davis 7911* combined anemometer and wind vane was used to detect wind speed and direction. The wind vane used a 22 $k\Omega$ continuous potentiometer to output an analog voltage dependent on the orientation of the vane. The anemometer used a magnetic switch with a contact closure to ground. Given this sensor was meant to be used in a plug-and-play fashion with a weather station module, the datasheet provided a vague description of its circuit. Standalone testing of the anemometer circuit by supplying power to the magnetic switch alone was unsuccessful. Through further testing, voltage needed to be concurrently applied to the yellow lead – likely supporting operation of the potentiometer and the magnetic switch. See the *Davis 7911* circuit (fig. 2) with appropriate power, ground, and output configuration.

A. Wind Vane Testing

Initial scope readings of the analog wind vane voltage showed good linearity. However, when reading the signal with the *uC32* analog pin, its corresponding analog value was very non-linear. This was assumed to be related to some impedance feedback from the *uC32* and was solved by passing the signal through an OpAmp Follower before the analog input(see fig. 2). To acquire an accurate function of analog value to degrees,

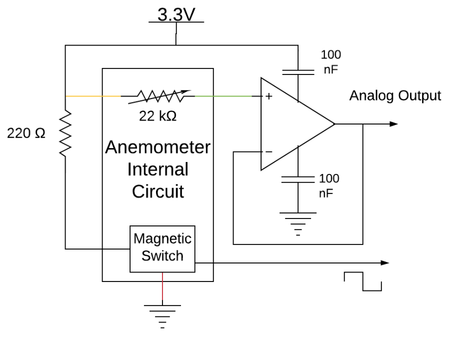


Fig. 2. *Davis 7911 Circuit*

data was collected at $5°$ increments from $0° - 360°$. While the resolution was good across most of the range, there was a dead-band from $345° - 10°$(fig 3). This was likely due to mechanical tolerances as the potentiometer wiper crossed from its area of maximum to minimum resistance. Initial linear fits included this dead-band with consequences to the accuracy across the range, especially further from $180°$. Excluding this dead-band, $\pm 1°$ accuracy is obtainable between $7° - 349°$. Some digital filtering was applied to the signal, averaging out an accumulated sum over $50\ ms$. Following is the current fit function to yield wind direction in degrees:

$$deg[°] = \frac{degVal - 1034.49}{-2.94} \tag{1}$$

The results are decent considering the data sheet specifies $\pm 7°$ accuracy. To compensate for the decreased operating range, the current design choice is to mount the wind vane such that the dead-band is along the centerline and towards the stern. Thus, the dead-band is uncommonly read for sustained periods of time.

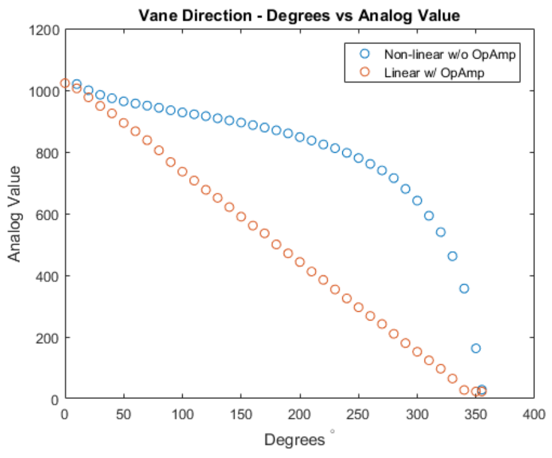


Fig. 3. *Wind Vane Test Data*

B. Anemometer Testing

The anemometer output a square wave with a falling edge upon each full rotation. Initial testing was done with a fan to determine the range of output frequency – a maximum of only a 7.2 Hz signal. To read this signal with the *uC32*, either a pulse count could be accumulated over a known time, or time between pulses calculated. Given the low frequency range of the signal, the design choice was to use the *Input Capture* peripheral module to count time between pulses. This module captures the time count and stores it into its associated register when a configured triggering condition occurs. Based on the *uC32* 40 MHz peripheral clock, the module was configured to capture on every falling edge, with the largest prescalar (1:256), a 32 bit timer, resulting in a rollover period of 7.6 hours. These design choices were made to minimize complexity of frequency calculations at negligible cost to accuracy of the low frequency signal. Initial mapping from frequency to apparent wind speed was somewhat crude, but yielded a decent function for further development. Test runs were done in low wind conditions in an attempt to simultaneously log vehicle speed from a *Garmin* GPS with the anemometer frequency. While the data appeared scattered, a least squares approximation yielded a 93% linear fit with a very small offset indicating no true wind offset.

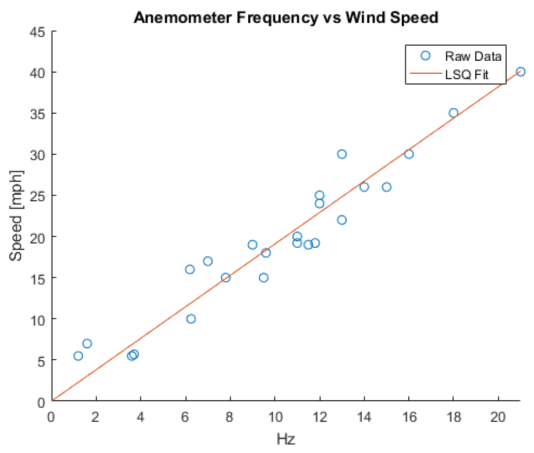


Fig. 4. *Initial Wind Speed Frequency Logging*

$$windSpeed[mph] = \frac{(freq + .007)}{0.524} \quad (2)$$

This mapping function was used in the next stage of testing, during which the GPS and anemometer C libraries were combined for simultaneous data logging. To eliminate current wind conditions, data was taken, when driving both easterly and westerly, and averaged at matching GPS speeds. There were data dropouts (fig. 5) that have been determined to be a loose wiring problem and solved by transferring the anemometer circuit (fig. 2) from a bread-board to soldered perf-board.

Initial results had an average error of 1.06 kt and up to a maximum error of 2.74 kt between the recorded wind speed and *BerryGPS* ground speed. A python script was written to minimize the average error between the averaged east and west wind speeds with the recorded GPS speeds. Several data runs were fed through the script resulting in a scaling factor of 1.12. This fitted the wind data to have an average error of only $0.38kts$ and a maximum error of $1.96kts$. Going forward in testing, the following adapted fit function was used.

$$windSpeed[kts] = \frac{freq}{0.468} \quad (3)$$

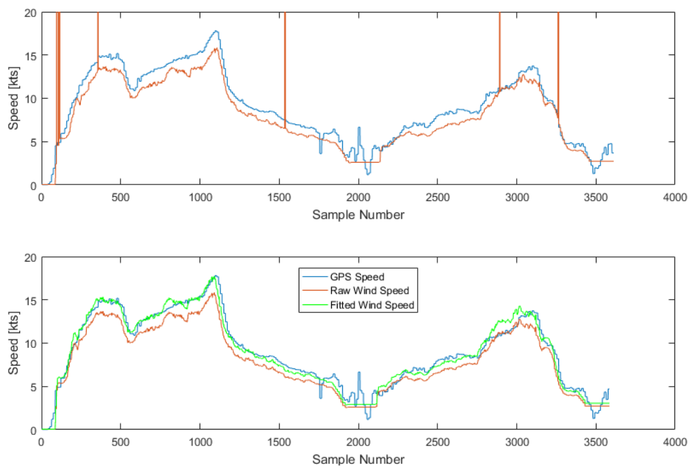


Fig. 5. *GPS Speed vs Wind Speed Tests*

III. Inertial Measurement Unit

The *LSM9DS1 IMU* chip was initially used to send roll and pitch angles to the *uC32* microcontroller assigned to the mast sensor package. The module makes use of the available I2C ports and reads the registers associated with the accelerometer, magnetometer, and gyroscope on all axes.

To calibrate the sensors onboard the IMU, it was necessary to record trial data for each axis and determine a sensitivity scaling factor and offset. Once applied, the accelerometer was able to measure gravity in *mG* with roughly 2.5% error.

The calibration method for both the accelerometer and magnetometer was to point each axis in the direction of the vector with the strongest magnitude detectable by the sensors. For the accelerometer, each axis was pointed in the direction of gravity, while for the magnetometer, the axes were pointed in the direction of the magnetic field vector. Null shifts were determined from magnitude discrepancies on opposing sides of the axis, and removed from the data.

The *LSM9DS1* IMU required more work than anticipated. To our surprise, the gyroscope and accelerometer were in a left-handed coordinate frame. Left-handed frames were not considered, as a misalignment matrix was to be generated for aligning the magnetometer and accelerometer. Since the misalignment methods assume both frames are right-handed, our complementary filter (fig. 6) methods were initially unsuccessful. The issue was corrected in software by flipping the orientation of the z-axis. Due to further issues with simultaneous GPS and IMU data collection from the module, the *LSM9DS1* IMU was replaced with a *MPU9250* IMU, and a similar process outlined above was repeated.

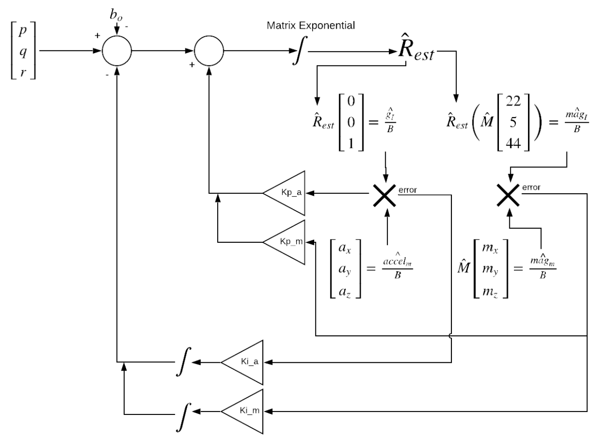


Fig. 6. *Complementary filter for IMU Attitude Estimation*

Our team had difficulties implementing a complementary filter algorithm in which roll and pitch were independent. Roll and pitch estimation worked well when the other was zeroed, and yaw estimation worked regardless of roll and pitch. However, roll and pitch were seemingly not independent of each other. It is still unclear if this was simply an artifact of the Euler Angle multiplication sequence to yield the DCM angles or due to a more in-depth problem. Our initial workaround was to place the IMU into gimbal lock position, enabling us to pull two independent angles. Given time constraints and efficacy of the method, we used the more familiar equations below to estimate roll and pitch components solely with an accelerometer [5]. These were run through a moving average to filter out high frequency noise and stabilize the roll and pitch results.

$$\phi_r = arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right)$$

$$\theta_p = arctan\left(\frac{-a_x}{a_z}\right)$$

Using the roll and pitch angles, a tilt-compensated magnetic heading was determined with the following equations [4].

$$X_h = X_m cos(\theta_p) + Z_m(\theta_p)$$

$$Y_h = X_m sin(\phi_r) sin(\theta_p) - Z_m sin(\phi_r) cos(\theta_p)$$

$$H = arctan\left(\frac{Y_h}{X_h}\right)$$

$$H = (\ H < 0\ )?\ H + 360 : H$$

IV. Boom Angle Sensor

The boom angle sensor was designed to translate the rotational motion of the boom into an electrical signal readable to the *uC32* microcontroller. The boom angle gives a very rough position of the sail; therefor, providing active feedback allows crews to make quick and easy changes for optimizing trajectory.

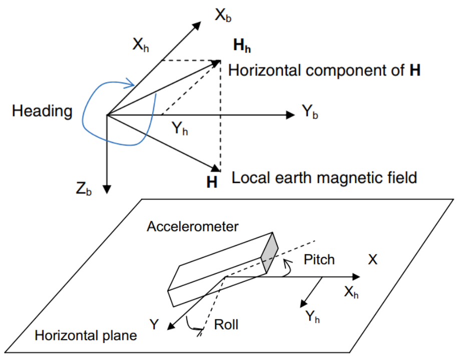


Fig. 7. *Tilt-Compensated Magnetic Heading [4]*

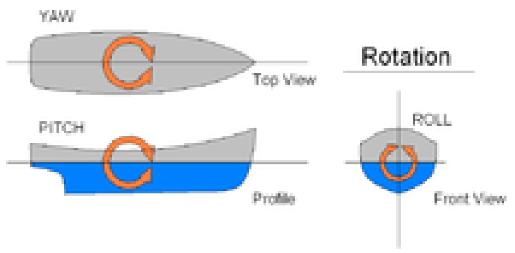


Fig. 8. *Axes Assigned to RS Quest dinghy.*

A. Software Implementation

To capture rotation as the boom rotates about the mast, we decided to directly connect a $10K\Omega$ potentiometer to the boom-mast hinge. Initially, a Panasonic EVU-F2AF30B14 potentiometer was used, however, this was later switched to a P160KNP-0FC20B10K (P160 B10K) model to improve accuracy. Placing the potentiometer directly below the hinge axis forces the boom and potentiometer to share the same axis of rotation. The potentiometer is connected to an analog pin of the *uC32* microcontroller.

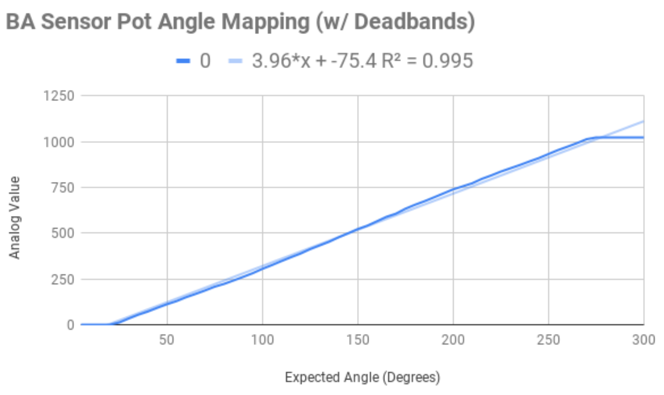


Fig. 9. *Potentiometer Analog Value vs. Set Angle w/ Deadbands (P160 B10K).*

In order to determine the angle of the boom from the rotation of the pot, the resulting analog value had to be mapped to a corresponding angle. The B10K model used has an expected range of 300°: due to two deadbands, the effective range is reduced to 250°. Analog values were read over several data runs at 5° increments over the full range. Least squares approximation was used to find the initial fit function. The function was inaccurate due to the presence of two dead-bands. From $0° - 24°$, the analog value measured zero, while from $275° - 300°$, the change became too small to accurately distinguish individual angles. As done with the wind vane, the solution was to place the potentiometer such that the boom never rotates into the deadbands. Because the boom has a total rotation angle of 120°, and the potentiometer can rotate 250°, the angle will be unaffected by the deadbands.

With a strictly linear potentiometer, the accuracy of the derived fit function

$$\theta = 270 - ((0.242 \cdot (AnalogValue)) + 29) - 116 \tag{4}$$

is directly affected by the least squares error. The potentiometer is mounted such that a 0° boom angle corresponds to the middle of the potentiometer's total range. In the function above, the value is offset by 116 to account for the aforementioned centering. The scaled and shifted analog value is subtracted from 270 to switch the rotation direction of the output angle to how the crew would perceive the angle.

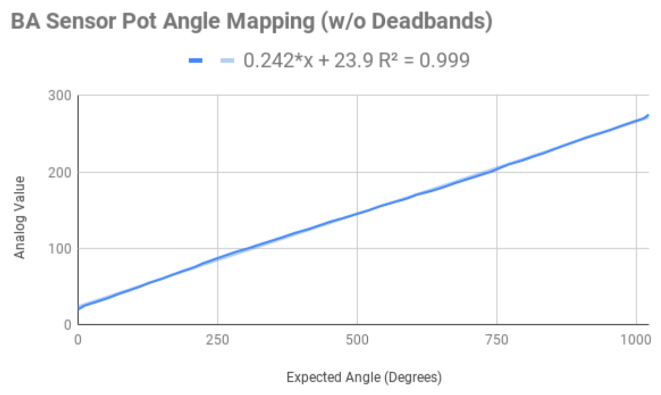


Fig. 10. *Potentiometer Analog Value vs. Set Angle w/o Deadbands (P160 B10K).*

B. Mount Design

The design of the mount is intended to establish the most direct connection between the potentiometer and the boom. The boom on the RS Quest is offset from its hinge by roughly 2 in. During normal operation, the boom can reach a pitch of 15° and when not in use can be folded down $-30°$ onto the cockpit.

The mechanical solution to varying pitch is composed of two parts: a keyed vertical link and a horizontal arm, both with a pivot hole connecting them. The design allows the linkage to compensate for changing pitch during operation of the dinghy. The potentiometer is fixed to a sliding piece, while a bracket or rail is fastened to the mast mount to allow adjustment of the potentiometer position. A platform was printed to hold the box alongside the mast. All of the parts were drafted using Solidworks and 3D-printed in resin.

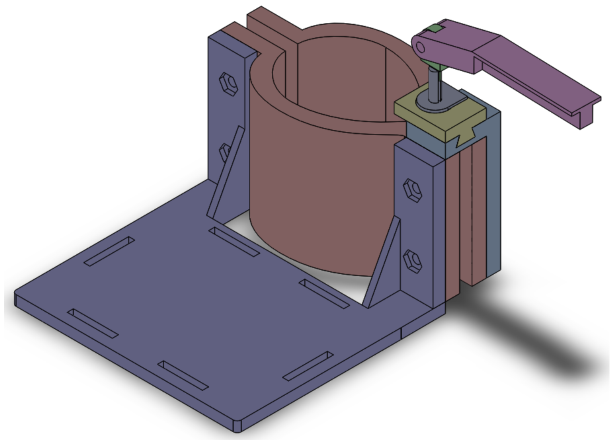


Fig. 11. *Boom Angle Mechanical Assembly*

The horizontal link in Figure 11 is designed to slide into a slot in the boom. Tolerances were chosen to accommodate the maximum accuracy of 3D printing (.002"), with additional buffering to prevent sliding parts from rubbing against each other. When printed, it was discovered that these tolerances were too tight. The piece tended to bind within the slot of the boom, which caused it to pull the potentiometer out of place whenever the boom pitched up or down. This was solved by reducing the length by 2 inches, sanding the piece down on all sides, and applying lubricant.

The vertical link is keyed to ensure a snug fit with the potentiometer. Initially, the vertical link was too tall to fit on the potentiometer while the mast mount was affixed high enough to clear the rigging. The height of the piece was reduced, and the keyed section made deeper. Implementing these changes improved the fit and gave a wider range of positions for the mast mount.

In addition to the horizontal linkage, a slider and bracket were incorporated into the design to allow the user to adjust the potentiometer position. These were placed in the assembly in Figure 11 to confirm a proper fit. In practice, the slider also had issues with tight tolerances. As with the horizontal link, this piece was sanded down and lubricated for smoother functionality.

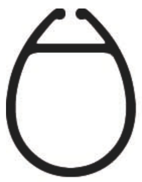


Fig. 12. *Cross-Section of Mast and Mast Mount.*

To support the boom angle sensor and *uC32*, a mast mount was designed. As shown in the cross-section in Figure 12, the mast has a unique shape, which made it difficult to measure. Tape was wrapped around the mast's length and width to get the respective measurements. However, because the tape was not able to hold the shape, the arc length of the mount (right) did not match the mast. A different approach was taken, in which modeling clay was wrapped around the mast to create a mold. A plaster positive was then made from the clay and used to build a fiberglass mount prototype. The fiberglass mount was used in conjunction with the mast cross-section to draft a new half-mast mount model.

V. Serial UART Communication

A. GPS

The *uBlox CAM-M8* GPS module mounted on a *BerryGPS-IMUv3* was selected to provide the necessary boat travel and location information to the system. The GPS output on the *BerryGPS-IMUv3* uses a serial UART connection to transmit data, and keeps the data packets in standard NMEA format. The GPS provides GGA, GLL, GSA, GSV, RMC, VTG, and ZDA sentences. Of these, the primary concerned packets are RMC (to provide latitude, longitude, track speed, and track angle) and GGA (to provide number of connected satellites, fix quality, and dilution of precision). On top of this data, timestamp information is polled from RMC, GGA, GLL, and ZDA sentences, as the system does not internally track time, but necessarily timestamps outgoing Bluetooth packets.

B. Bluetooth

Since the positions of the sensors are about 10ft apart, running a wire from the boom angle sensor to the *uC32* at the top of the mast for data processing was not a viable option. To solve this, a separate *uC32* is connected to the boom sensor package, and data is transmitted between the two via Bluetooth. The data from all mast sensor package sensors are combined into a custom packet, and sent to the boom sensor package. The boom sensor package adds it's own data to the custom packet before forwarding the new, complete packet to a mobile application for optimal path calculation and data display on a user interface.

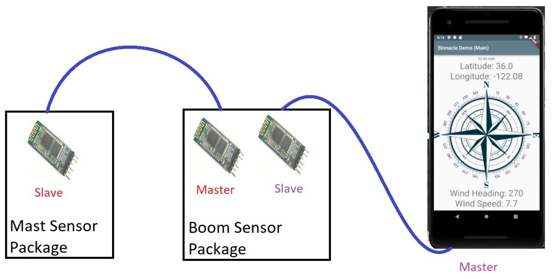


Fig. 13. *Bluetooth Network Configuration*

The custom "GNOST" Bluetooth packet follows NMEA convention, and has the following layout:

```
$GNOST,(filtered?),(lat),(long),
(wind speed),(wind dir),(gps speed),
(gps direction),(boom angle),
(compass heading),(timestamp),
*(checksum)\r\n
```

Since this packet follows NMEA convention, the same parser library used for the GPS can be slightly modified to support the building and parsing of GNOST sentences as well. The parser library method *processSentence()* takes an undefined NMEA sentence as input. If it is a RMC, GGA, or GNOST sentence, this is detected and the necessary data from these sentences is stored inside fields within the parser. At any time, *createGNOST()* may be called to populate a buffer with the current status of the data fields tracked by the parser in the format shown above. Combined with non-blocking serial data collection, the transition from data to Bluetooth packet is smooth and does not hinder the data processing done by the micro-controllers.

The two sensor packages each use a *HC-05* module to communicate via Bluetooth 2.0 protocol. Using AT commands, two were configured as slave devices and one was configured to master. This Master-Slave pair was also configured to automatically pair with each other. This provides a reduced amount of startup stress for the micro-controllers, and allows automatic re-connection if the link is interrupted during uptime.

Transmission speed between the two sensor packages is fixed at 12 Hz, a number determined by the output frequency of packets sent by the on-board GPS module. To communicate with the mobile device, a *SH-HC-08* Bluetooth 4.0 (or BLE) module is used on the boom sensor package. This is necessary for the system to be able to communicate with iOS devices, which require no lower than a Bluetooth 4.0 protocol when establishing a Bluetooth connection.

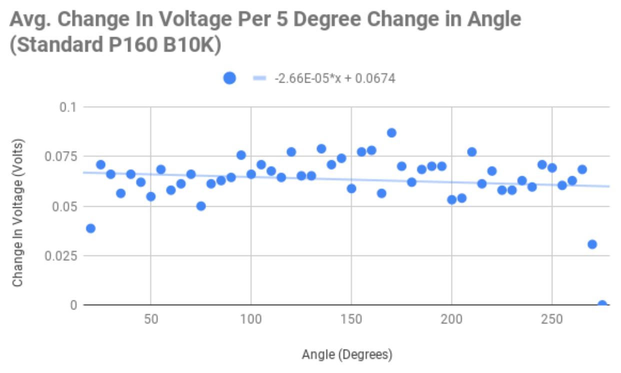
Despite receiving data from the mast package at 12Hz, the boom sensor package is unable to transmit at this rate due to the lower throughput of Bluetooth 4.0, and instead transmits at a rate of 1 Hz. The packet output by the boom sensor package is a filtered and processed version of the packets received from the mast sensor package.

VI. Results

A. Boom Angle Accuracy

The accuracy of the boom angle is determined by two factors: the accuracy of the fit function and the linearity of the potentiometer. If the potentiometer has poor linearity, any fit function using the measured voltage will not be strictly linear. For the EVU-F2A, a range of 230° and an input voltage of 3.3 VDC results in an expected voltage increase of $0.072V/5°$. In practice, the measured voltage increase was $0.082V/5°$. The error between these terms is roughly 12.195%.

To compensate for the EVU-F2A inaccuracy, multiple fit functions were used based on the measured voltage in different sections of the potentiometer range. Ultimately, switching to the standard B10K model (with better linearity) resulted in a



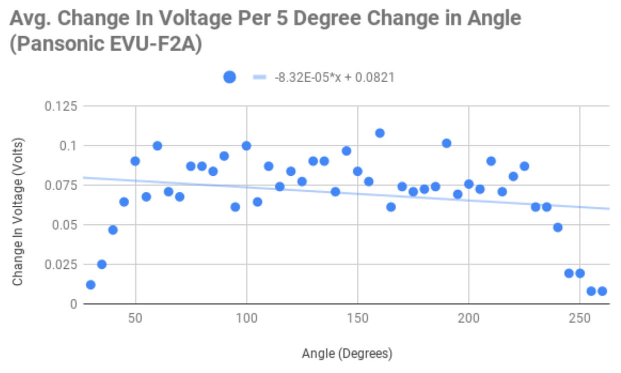


Fig. 14. *Change in Voltage per* 5° *Turn of Potentiometer.*

more accurate boom angle reading. For the B10K, the expected average voltage increase was $0.065V/5^\circ$, while the actual increase was $0.067V/5^\circ$, resulting in an error of 3.561%. As seen in Figure 14, the data points for the Panasonic EVU-F2A are spread over a wider range than in the B10K graph, indicating a greater tendency to vary from the expected voltage increase. It follows that the B10K has a higher linearity than the EVU-F2A.

Finally, the error of the least squares fit function was 0.999, suggesting a good fit to the measured data. The determined offset for the fit function is the only remaining source of error, and was reduced by rotating through angles with the whole system in place.

B. On-board Data Processing

As previously stated, the rolling and pitching motion of the boat has an unavoidable effect on the sensors, most notably the anemometer and wind vane. Without any data processing, the readings vary wildly and are unusable. In order to solve this, a circular moving average was placed on data from impacted sensors. Assuming a steady wind between the two extremes of a single pitch/roll of the boat, this method would yield reasonably accurate results.

In future implementations of the sensor package, this method would be replaced by using the IMU to calculate the exact motion vector of the sensor package, and offset the sensor data by that amount.

C. Wind Triad Results

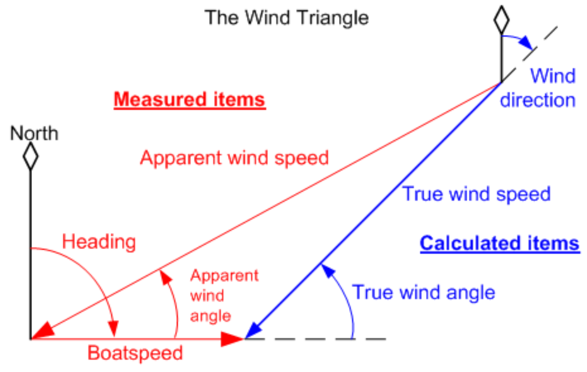


Fig. 15. *Measured and calculated results of the wind vector triangle.*

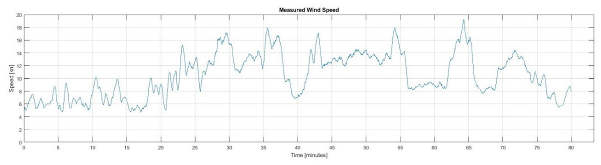
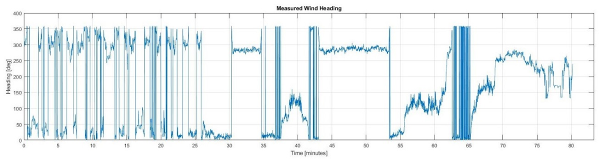



Fig. 16. *Measured Wind Data*

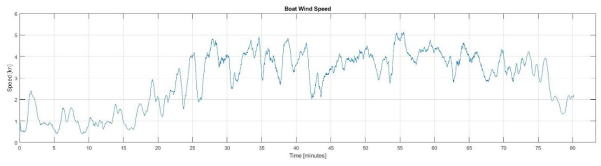
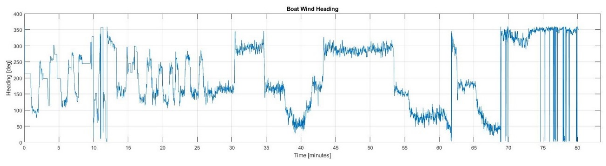



Fig. 17. *Induced Wind Data*

Using the data recording and data processing methods described in previous sections, many data runs were conducted in conjunction with post-processing calculations that mimicked those that would be done live on the mobile application.

The data collected and displayed in Figure 16 represents the measured wind speed and direction gathered from the anemometer and wind vane. Using the known heading of travel of the boat, this can easily be converted into a cardinal frame, which then represents the apparent wind vector from Figure 15. The data collected and displayed in Figure 17 represents the measured wind speed and direction induced by the motion

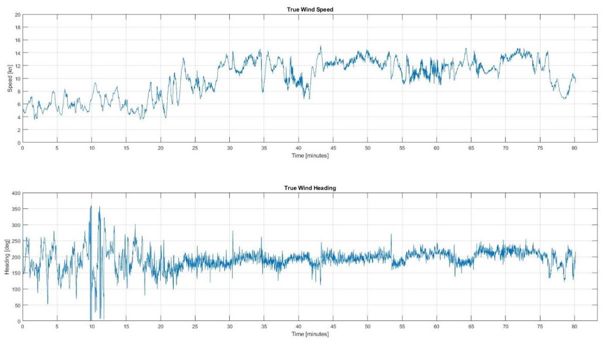


Fig. 18. *Calculated True Wind*

of the boat, which again, was gathered by the GPS. The data in 17 represents the boat speed vector in Figure 15.

Using the two known vectors – apparent wind and boat motion – true wind was calculated with simple vector math in post-processing. The result of this math is displayed in Figure 18.

Immediately apparent is the unstable data in the first 25 minutes of the data run. This is caused by the low speed, high frequency direction changes that occurred while leaving the harbor. On top of this, various obstacles such as other boats and shelter over the harbor caused irregular wind patterns. At 10 minutes into the data, the disturbance is caused by de-reefing the main sail, while rocking and turning the boat around slowly. Once leaving the harbor and entering the bay, normal turning frequency and consistent wind provided stable and reasonable data.

D. *Validity of the Data*

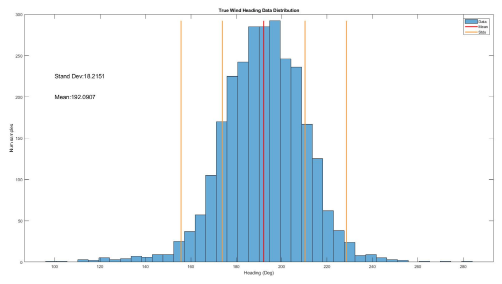


Fig. 19. *Distribution of True Wind Heading*

To verify the correctness of the calculated true wind, basic statistical analysis was performed in the form of a distribution plot, and a calculation of turbulence intensity. It is important to note that the distribution Figures 19, 20 and additional data analysis was performed only on the section of data taken in the bay, excluding the erratic behavior observed in the harbor during the first 25 minutes of the test run.

The calculated true wind heading showed a standard deviation of 18.21°, and a mean heading of 192.09°, matching the expected direction of the wind for that day. Additionally,

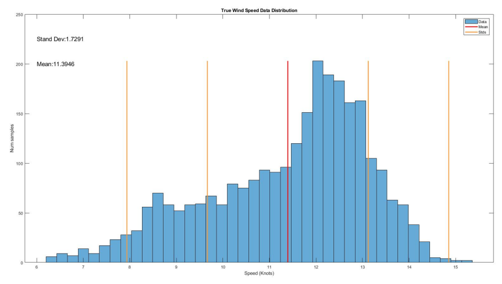


Fig. 20. *Distribution of True Wind Speed*

distribution graph 19 for the data is exceptionally normalized, showing great promise for the validity of the wind heading data gathering and processing.

The calculated true wind speed showed a standard deviation of 1.73 kt with a mean speed of 11.39 kt. Compared to the wind speed for the day, the mean speed is expected; however, the standard deviation was slightly higher than expected. This is further supported by the left-tapered nature of the distribution graph in Figure 20. This is believed to be caused by a wind shadow slightly cast onto the anemometer by the sensor package itself when the wind is coming from the boat's starboard side. The wind shadow would artificially cause lower wind speed measurements when oriented in this way. Even so, the standard deviation of 1.73 knots correlates to a turbulence intensity of 0.15, a respectable measurement in regards to the day's wind speed. Other wind studies show turbulence intensities varying between 0.05 and 0.3 for similar wind speeds.

E. *Polar Speed Graphs*

Using the data collected, post-processing was done to analyze the sailboat's speed as a function of its true wind angle. This is often done to determine the speed at which a boat can sail relative to the true wind as well as its maximum upwind and downwind velocities, known as VMGs. Polar speed graphs include plots for several discrete wind speeds, as an increase in such results in faster sailing speeds. While each polar speed plot has a different function of true wind angle, it is common to share a similar form. As a broad generalization, boats often sail fastest in a beam reach point of sail where the true wind angle is 90°. However, each boat has characteristic polar speed plots, and we calculated some of the polar plots for the *RS Quest*.

Using the post-processed data from a sailing session, true wind angle and boat speed data were grouped based on a corresponding true wind speed. Several groups of polar speed data were plotted based on an increasing true wind speed within a threshold range displayed in Figure 21. With knowledge of the wind conditions for that session and examining the likeness to polar speed form, a group of data was chosen. The boat speed data was averaged around an increasing true wind angle as shown in Figure 22.

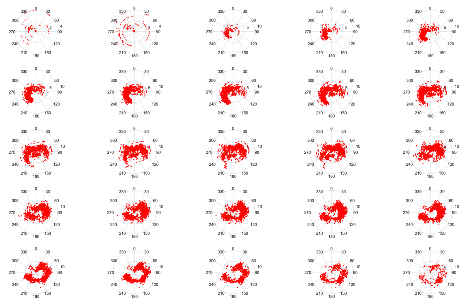


Fig. 21. *Polar speed data for incrementing true wind speeds.*

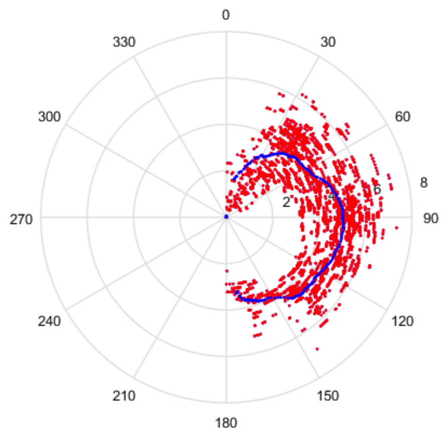


Fig. 22. *Polar speed data for* 10.5 kt *wind. Blue line is averaged across true wind angle.*

These processes were repeated for different sailing sessions during which the wind conditions were equally different. Averaged polar speed plots were extracted and plotted on the same graph in Figure 23. The beam reach point of sail is clearly observable to be the fastest under 5 kt and 10 kt wind speeds. Our best sailing run, shown in red, shows an average maximum boat speed of $5.1kt$ at $95°$ to the true wind. Maximum upwind VMG was achieved at $41°$, 2.8 kt and maximum downwind VMG was achieved at $159°$, 3.6 kt.

VII. Conclusion

Through the integration of the aforementioned GPS, sensors, Bluetooth modules, microcontrollers, and mechanical assembly, a sailboat sensor system was implemented in conjunction with a mobile application from our collaborative computer science team. This sensor system is capable of providing real-time GPS location, speed over ground, course over ground, an apparent wind vector, and boom angle. The post-processing methods were used to calculate true wind vector components and to analyze the polar speed performance of the *RS Quest*. With refinement, a more stable and accurate tilt-compensated magnetic heading could be implemented to provide a true heading versus course over ground heading.

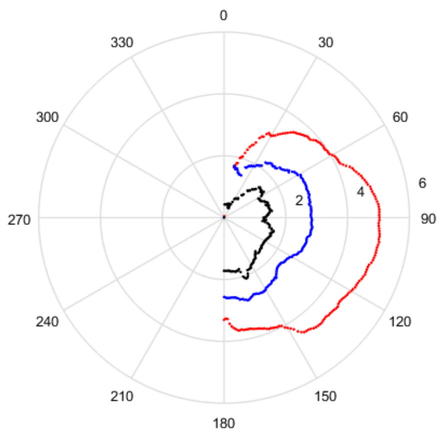


Fig. 23. Polar speed plots for *RS Quest* at 4 kt, 5 kt, and 10.5 kt

Further work would be needed to use an IMU to correct the induced winds from the rolling and pitching of the boat.

Acknowledgment

We would like to acknowledge: our fellow CMPS 116/117 team for their collaboration; our advisor Gabriel Hugh Elkaim for his counsel in this project; our professor Maxwell Dunne for his teaching; our TA Joseph Adamson for his dedicated support; and Rusty and Hillary from the UCSC Boating Center for the provided opportunity and sailing education.

References

[1] Elkaim, Gabriel Hugh. "Batch Misalignment Calibration of Multiple Three-Axis Sensors." 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR 2016). 2016.

[2] Hendrik Erckens , Gion-Andri Busser, Cedric Pradalier, Roland Y. Siegwart, "Navigation Strategy and Trajectory Following Controller for an Autonomous Sailing Vessel," IEEE Robotics Automation, Apr.2010

[3] Did a Solar-Powered Autonomous Boat Just Cross the Pacific Ocean? — Make: Make: DIY Projects and Ideas for Makers, 22-Aug-2016. https://makezine.com/2016/08/22/solar-powered-autonomous-boat/.

[4] "Using LSM303DLH for a Tilt Compensated Electronic Compass." STMicroelectronics, Aug. 2010, https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf

[5] Talat Ozyagcilar. "Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors" https://cache.freescale.com/files/sensors/doc/app$_note/AN4248.pdf$