Asia Pacific Research Initiative for Sustainable Energy Systems 2012 (APRISES12)

Office of Naval Research
Grant Award Number N00014-13-1-0463

Computational Fluid Dynamics (CFD) Applications at the School of Architecture, University of Hawaii: Development of Data Verification Process for Internal CFD Simulations

Task 7

Prepared For
Hawaii Natural Energy Institute

Prepared By
Sustainable Design & Consulting LLC, UH Environmental Research and Design Laboratory, UH Sea Grant College Program & HNEI

January 2015

Project Phase 1- 7.B

REPORT ON DEVELOPMENT OF A DATA VERIFICATION PROCESS FOR INTERNAL CFD SIMULATIONS

8

FINAL

January 5, 2015

Prepared by:
Manfred J. Zapka, PhD, PE (Editor)
Eileen Peppard, M. Sc.
Tuan Tran, D.Arch
Stephen Meder, D.Arch, Director
A. James Maskrey, MEP, MBA, Project Manager

ERDL

HNEI

Contract # N000-14-13-1-0463

Computational Fluid Dynamics (CFD) Applications at the School of Architecture, University of Hawaii

Project Phase 1 – 7.B

Develop Skill Set for Internal CFD Analysis and Validation at the Building

Project Deliverable No. 8

Report on Development of Data Acquisition Procedures for Internal CFD Simulation Validation

Prepared for Hawaii Natural Energy Institute

in support of

Contract #N000-14-13-1-0463

January 5, 2015

Prepared by:

Manfred J. Zapka, PhD, PE (Editor) (*)

Eileen Peppard, M. Sc. (**)

Tuan Tran, D.Arch (**)

Stephen Meder, D.Arch, Director (**)

A. James Maskrey, MEP, MBA, Project Manager (***)

(*) Sustainable Design & Consulting LLC

(**) Environmental Research and Design Laboratory (ERDL), School of Architecture, University of Hawaii at Manoa

(***) Hawaii Natural Energy Institute (HNEI), University of Hawaii at Manoa

TEAM EFFORT AND ACKNOWLEDGEMENTS

The project work performed on the development and calibration of the field measurement process for internal CFD validation was a significant team effort. The experiences gained will help the team to successfully conduct the subsequent phases of the project. The documentation of the work performed and lessons learned will help incoming team members and expedite their learning curve in relevant research efforts.

The authors would like to thank the staff of the Environmental Design & Research Laboratory (ERDL) for their assistance in carrying out parts of this research study. The authors especially acknowledge the valuable contribution of postdoctoral fellow Aarthi Padmanabhan, D. Arch. The authors also acknowledge the dedicated work by research assistants Mike Poscablo, Christian Damo, Reed Shinsato and Steven Chen. Christian Damo has graduated in the meantime and we wish him all the best for his professional and personal future.

The authors would also like to thank the management and staff of Outreach College for their cooperation and willingness to play an important part of the successful completion of the project. Without their patience in allowing the measurements to occur within their work place, the project work could have not been successfully completed.

TABLE OF CONTENTS

APPENDICES

SECTION 1 – EXECUTIVE SUMMARY

The present report on the development of field test procedures for validation of Computational Fluid Dynamics (CFD) inside buildings is part of a CFD research program which is sponsored by the Hawaii natural Energy Institute. The research program endeavors to develop advanced building modeling skills at the Environmental Design and Research Laboratory (ERDL) of the School of Architecture, University of Hawaii at Manoa.

The present report summarizes the work performed by the ERDL CFD research team developing procedures to prepare, conduct and process data acquisition inside buildings, whereas the data will be used to validate "internal" CFD investigations of. The term "Internal CFD" refers to air flow occurrences and related physical properties inside an enclosed building space, as opposed to "external CFD" which is CFD analysis of wind movement and related physical properties outside and around a building. External CFD analysis was carried out at an earlier phase of the research project.

The test site was Room 301, a 5,000 square foot naturally ventilated office space on the third floor in the Sinclair Library on the University of Hawaii campus. Room 301 was being used as the test site for another ERDL research project and the project work for this study could use some of the equipment and logistic support of the other ERDL project. The research team selected the instrumentation and data acquisition system (hardware and software) to carry out air velocities and differential pressures measurements inside the space that was used as the test site.

Eight hot wire anemometers (range 0-5 m/s full range, five low differential pressure transducers (one 0-10 Pa and four 0-25 Pa full range) and one weather station (wind speed and direction sensors) were used for the measurements. The anemometers were installed inside Room 301 during the four days of measurements. Each differential pressure transducer was equipped with two pressure tubing sections and the tubing terminals were placed at two points between which the pressure difference was being sampled. The tubing was placed during the four days of testing. The weather station was placed on the roof of Sinclair Library building recorded wind data during the test.

Room 301 was equipped with two sets of exhaust fan units, each of them having two propeller fans to extract high air volumes at low pressure from Room 301. Using the exhaust fans provided the research team with the opportunity to induce air movement in Room 301, even without sufficient external wind. Incident wind on the North wall of Room 301 increased the ventilation driving force. The air movement inside Room 301 was controlled by selectively opening and closing North facing louver windows and by turning the two exhaust fan units on or off.

Four test scenarios were defined to test air flow with a certain combination of open windows and operating fan units. Initial CFD scoping simulations were carried out to determine the approximate airflow in Room 301 under the different test scenarios and to select the placement of anemometers and differential pressure sensors at locations in room with anticipated high velocity and pressure gradients.

Tests were conducted in Room 301 on four days after regular business hours, when the office was not occupied, in order to establish the test scenarios without interfering with regular office work. On each of the four days, air velocity and differential pressure measurements were taken for about 30 minutes for each of the four test scenarios. The data was recorded, conditioned and analyzed with a database data analysis program. Data acquired during the tests by the weather station was filtered, and air velocity and pressure data collected in Room 301 was correlated with the weather station data.

The data acquired by the weather station was used as boundary condition input for the final CFD simulations of the four test scenarios. The results of the measurements in Room 301, correlated with the weather station and conditioned to obtain mean values, was then used to validate the results of the final CFD simulations.

The outcome of the work on this study was the successful completion of all study objectives. The research team gained important working knowledge and developed robust procedures to conduct measurements of relevant physical properties that are related to natural ventilation and air movement of internal spaces.

SECTION 2 – OBJECTIVES

The overall goal of the project work presented in this report was as follows:

> Gain proficiency in the research team in developing and executing a test procedure to select, test and deploy instrumentation for air velocity and differential pressure inside building spaces to obtain data that can be used for validation of internal CFD simulations.

The following work task objectives were required to

- Select a suitable test space for the measurements.
- Select instrumentation for internal air velocity and differential pressure measurements which were suitable for airflow patterns and differential pressure conditions inside a space.
- Determine the site-specific conditions for instrumentation deployment and actual filed measurements inside the test space
- Perform initial CFD scoping simulation to determine preferred locations inside test space where instruments should be deployed, which were best suited for validating results of internal CFD simulations inside the test space.
- Develop and test a data acquisition and analysis procedures for the data air velocity and differential pressure measurements.
- Test the instrumentation and analysis and reduction procedures in the lab of the team before deployment in the test space.
- Custom build and Install the exhaust fan units in the test space.
- Coordinate with the test space occupants to perform the tests at times that were outside regular working hours.
- Install the instrumentation and data acquisition system during three days of data acquisition at the test site.
- Perform the final data analysis and provide the results to the CFD team for validation of the final CFD simulations of air flow through the test space.

SECTION 3 – APPROACH

This section describes the approach of required work tasks to develop the test measurement procedure, prepare and deploy the instrumentation and data acquisition at the test site. Figure 3.1 shows the diagram of the interrelated work task performed. The numbers in Figure 3.1 correspond to the work tasks described in this section.

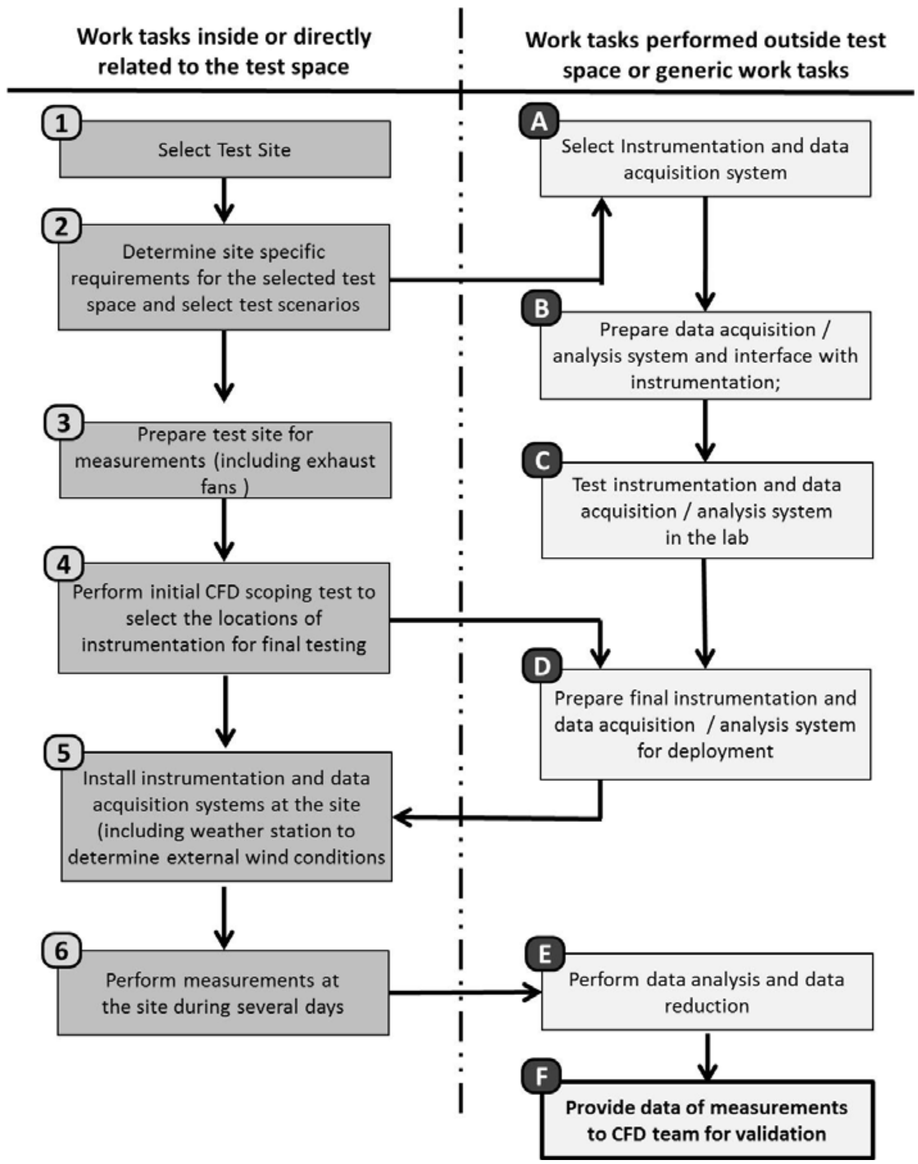


Figure 3.1: Diagram describing the interrelated work tasks performed

The work tasks, which were carried out under this report, can be divided into two categories:

- Work tasks that are carried out inside the test space (e.g. the test site) or tasks that are directly related to the preparation of the site.
- Work task that represent generic tasks to prepare and execute measurements of air velocities and/or pressure differentials.

3.1 Work Tasks Inside or Directly Related to the Test Space

The work tasks that are carried out either inside the test space or are directly related to the test space are the tasks indicated as 1 through 6 in Figure 3.1. These tasks are described in the following:

(1) Selection of the test site (test space): The CFD team could take advantage of a test set-up that was installed natural ventilation performance and occupant comfort in Room Sinclair 301 (Room 301). The test site is described in more detail in Section 4.

The test space was equipped with two exhaust fans units. Each of the exhaust fan unit had with two propeller type 30-inch whole house fans. The exhaust fans made it possible to induce air flow inside Room 301 with a good approximation of the exhausted air rate. While the CFD research team did not have access to a wind tunnel to validate internal air movements, Room 301 offered a unique opportunity to the team to carry out full scale measurements of air velocity and differential pressure in an effort to validate CFD simulation results for the same room.

(2) Determine site specific requirements for the selected test space and select test scenarios: The air flow pattern inside Room 301 could be adjusted by the inlet and the outlet conditions of the room. The air inlet openings of Room 301 were the North facing windows. These windows were louver or jalousie windows, which stretched over the entire length of the North side of Room 301. The air flow patterns inside Room 301 could be adjusted by selectively opening and closing window sections on the North side of the room. The air flow patterns inside the room could also be adjusted by selectively turning the four fans on or off. The CFD research team identified four test scenarios under which different air flow patterns could be established inside Room 301 by combinations of window sections open or closed and fans or off. The four test scenarios were selected on the basis of internal structural elements and cubicle walls forming preferred passageways for air flow.

(3) Prepare test site for measurements (including exhaust fans): The test site, e.g. Room 301, had to be prepared for the measurements. The exhaust fan units on the South side of Room 301 were tested to operate under the selected test scenarios. The windows on the South were mostly louver or jalousie windows. The team had to make sure that the windows could be closed during the test

runs. Leaking windows would have admitted air flow from the South and negatively affected the effectiveness of the exhaust fans to discharge the desired high volumes of air. In order to prepare the louvered windows on the South side of Room 301 hinges had to be oiled and several windows had to be forces in either closed or open position.

(4) Perform initial CFD scoping tests to select the locations of instrumentation for final testing: With the four test scenarios selected initial CFD scoping simulations were performed for each test scenario to obtain an approximate idea about the air flow conditions inside Room 301. The CFD simulations used approximate mass flow rates through the North windows in accordance with the maximum exhaust rates of the two fans units. The resulting color contour maps for each test scenario were examined for locations inside Room 301 with high velocity gradients. The selected locations were then identified in the layout map of Room 301 for the placement of anemometers during measurements. The locations for deployment of anemometers were kept the same for all four test scenarios.

The initial CFD scoping runs furthermore identified preferred locations where differential pressures were to be measured in Room 301. The CFD team anticipated that pressure losses across the openings of Room 301 would yield the most useful indicators of pressure losses of air entering and leaving the room.

(5) Install instrumentation and data acquisition systems at the site (including weather station to determine external wind conditions): This work task included the actual deployment of the anemometers and differential pressure transducers, as well as the data acquisition systems in Room 301. The instrumentation and data acquisition had to be installed and taken down for each day of measurements. The anemometers and differential pressure transducers were placed at locations inside Room 301 which would interfere with the daily operation inside the office space of Room 301. The measurements could therefore only be conducted during the evening hours or on the weekend.

(6) Perform measurements at the site during several days: The air velocity and differential pressure measurements were conducted on several days. The different days featured low as well as strong outside winds. High incident wind speed impinging on the North facing windows exerts pressures that increase driving forces on the air inside Room 301, in addition to the mechanical ventilation from the exhaust fans. In order to obtain a range of ventilation performance for Room 301 test measurements were conducted on four days with low and stronger external wind coming from the North-East.

3.2 Work tasks performed outside the Test Space

The work tasks that were performed outside Room 301 were to prepare instrumentation and data acquisition for deployment at the test space, and to prepare the data results for analysis by the CFD team for validation of final CFD runs.

(A) Select Instrumentation and Data Acquisition System: The research team used an existing pool of equipment including one weather station, several anemometers and five differential pressure transducers from previous studies. No new instruments were acquired, and the instrumentation and data acquisition was adjusted to the study's internal air flow conditions.

(B) Prepare data acquisition / analysis system and interface with instrumentation: While the test hardware was the same as used for external CFD validation, the software that controlled the instruments and data acquisition during internal CFD validation measurements differed from previous codes. The research team also had new team members who used different software interfaces and coding.

(C) Test instrumentation and data acquisition / analysis system in the lab: The anemometers, differential pressure transducers and the data acquisition system were prepared and tested under laboratory conditions before being deployed in the field. The wind sensors and pressure transducers required correct signal excitation and the measured signals required calibration and signal condition. A simple ad-hoc wind tunnel was used to establish qualitative wind speeds and pressure differentials.

(D) Prepare final instrumentation and data acquisition / analysis system for deployment: The results of the initial CFD scoping g simulations runs (refer to task (4)) indicated to the team at what locations and at what height (e.g. above floor) the anemometers were to be installed for the measurements. The locations of the anemometers determined the cabling between sensors and data acquisition multiplexer. The differential pressures were measured with two pressure tunings connected to the load n high pressure tab of the transducers. The length of the pressure tubing was selected, and the tubing sections serving the same pressure transducer were held at the same length. With the final preparations for field deployment completed for the instrumentation and data acquisition system, the measurements in the test space were ready to be undertaken (refer to task (5)).

(E) Perform data analysis and data reduction: With the measurements in the test space completed, the team had to prepare the obtained data of air velocities and differential pressures for further use in validation of theoretical CFD simulation results for all four test scenarios.

SECTION 4 - METHODOLOGY

This section describes the methodologies for preparation of the test space, instrumentation and data acquisition / analysis used in the project work.

4.1 Description of Test Site in Room 301 of Sinclair Library

The site for this investigation was Sinclair Library (address: 2425 Campus Road, Honolulu, HI 96822) on the west side of the University of Hawaii Manoa campus, adjacent to University Avenue (see Figures 4.1. and 4. 2). The 123,000 square foot Sinclair Library building was completed in 1956. Although much of the building has been converted to air-conditioned space, the space used in this study, Room 301, is naturally-ventilated. Room 301 is a 4,800 square-foot open office on the third (top) floor on the west side of the building (Figures 4.3 and 4.4).

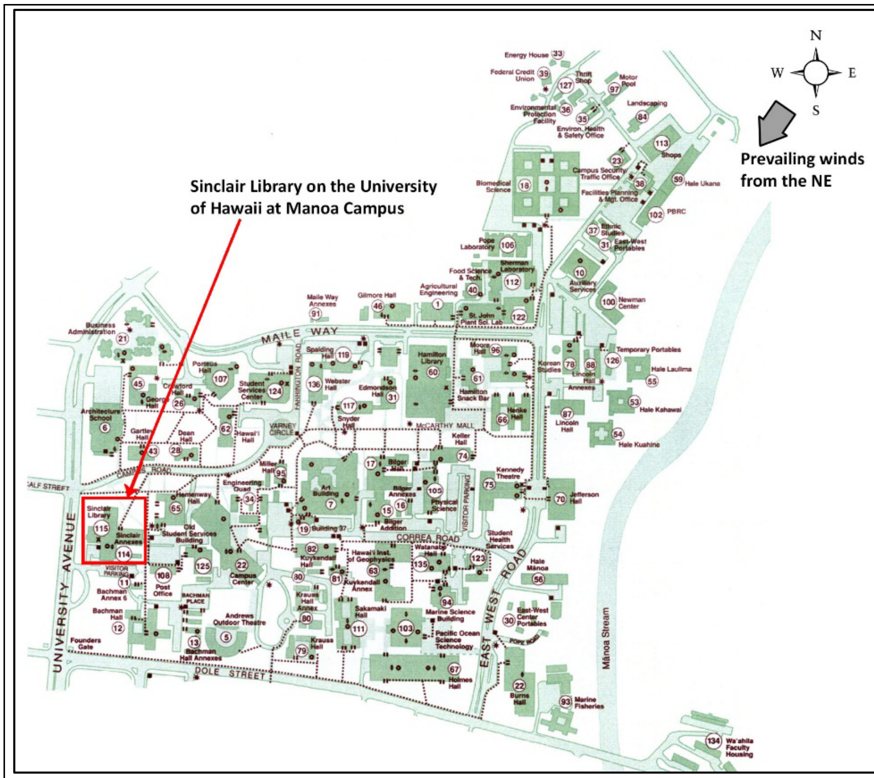


Figure 4.1:
Location of Sinclair Library, the site of the stury, on University of Hawaii Manoa Campus

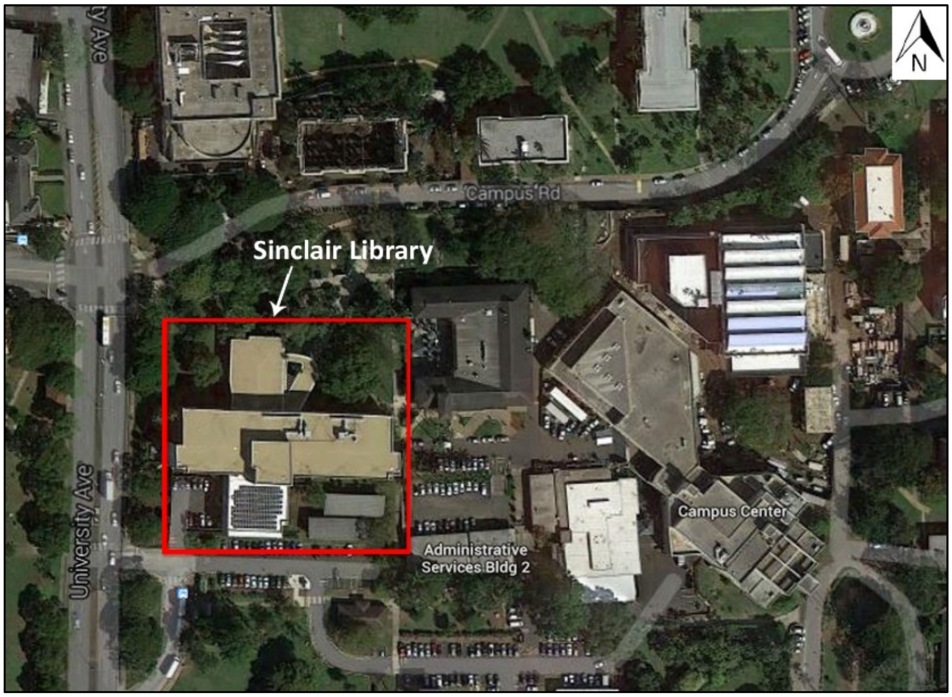


Figure 4.2: Sinclair Library building on the University of Hawaii Manoa Campus (Aerial image obtained from Google Earth)

Figure 4. 3: Sinclair room 301 north side

Figure 4. 4: Sinclair room 301 south side

Figure 4.5 shows a section of the North wall of Room 301, which consists of pane windows and glass louver windows over its entire 70 feet length. Figure 4.6 shows the South wall of Room 301 which consists of one pane window, three sets of sliding glass doors, a single glass door and the rest consists of wooden louvers. Figure 4.6 shows the 11 feet wide lanai which stretches of the entire length of Room 301 and has a 3.3 feet high parapet wall. The roof overhang covers the lanai. The West wall of Room 301 is brick and has no openings. The East wall of Room 301 is an interior wall, which has a double door to an adjacent air-conditioned room. The access to Room 301 is through the adjacent air –conditioned building.

Figure 4. 5: North wall of Sinclair room 301 with window panes and glass louvers.

Figure 4. 6: South wall of Sinclair room 301 with glass doors, wood louvers, and attached lanai.

The ceiling height inside Room 301 is 10 feet. Luminaries are set flush with the suspended ceiling (see Figure 4.7). The office space inside Room 301 has numerous cubicles with wall heights of 30”, 37”, 50”, 44”, 64”, and 82” height. Two exhaust fans units with two propeller fans (Master Flow model 30BWHFS; 30”; 6,000-CFM belt-driven whole house fans) were temporarily installed in conjunction of another research project. Figure 4.8 and 4.9 show the two exhaust fan units installed in the South of Room 301.

Figure 4.7:
Sinclair Room 301 showing luminaries and cubicle walls inside the office space.

Figure 4. 8: A pair of exhaust fans installed in a custom-built housing

Figure 4. 9: Placement of the two exhaust fan unit on the south lanai (Note that the fan housings remained open to the South

4.2 Selected Instrumentation

Three types of measurement devices for air velocity and differential pressures were used in this study:

Weather station (temporarily installed during several days of measurements): Wind velocity and wind direction were measured and logged with an Onset HOBO U30 weather station installed on the west side of the roof of Sinclair, directly above Room 301 (see Figure 4. 10). The wind velocity and direction indicator sensors were 9.5 feet above the roof and 50 feet above the ground. Recorded data resolution was set at 5 Hz.

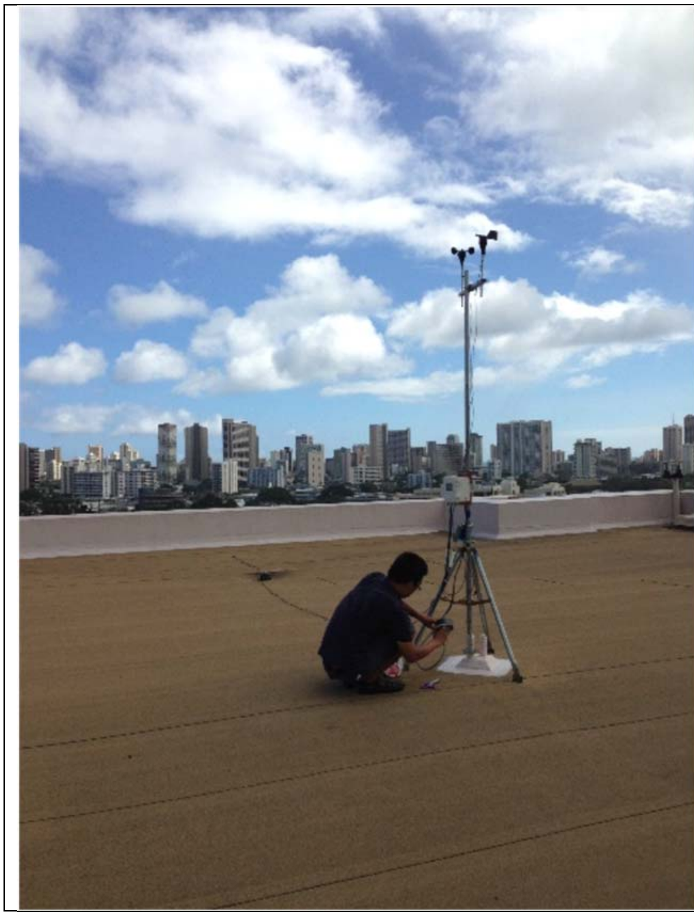
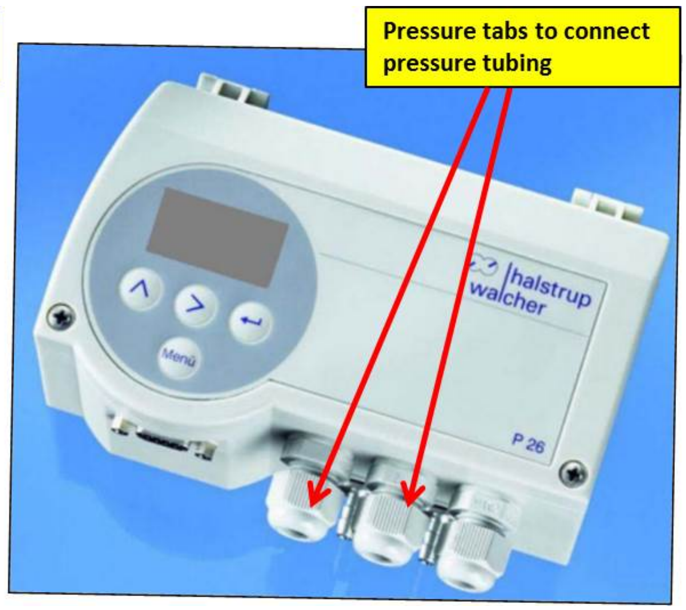


Figure 4.10:
The weather station (Onset Hobo U30 data logger) with wind velocity and direction sensors was installed on roof of Sinclair library during the duration of the measurements

Differential pressure transducers: Differential pressure was measured across selected points inside Room 301. Two types of differential pressure transducers were used. Four Setra differential pressure transducers, Model 264, were used, which had a range of 0 to 0.1 inches of water column (zero to 25 Pascal) and 0.5% accuracy of full scale. One Halstrup Walcher differential pressure transducer Model P26 was used, with a range of 0 - 0.055 inches of water column (0 - 10 Pascal) and 0.5% accuracy of full scale. Figure 4.11 shows the two type of differential pressure transducers used in the study.

(a) Setra differential pressure transducer Model 264



(b) Halstrup Walcher differential pressure transducer Model P26

Figure 4.11: Two types of differential pressure transducers used in the study

Both types of differential pressure transducers were connected to pairs of flexible vinyl pressure tubing ($^3/_{16}$" internal diameter for the Setras and ¼" for the Halstrup Walcher). Each transducer had two pressure tabs, one high and one low pressure tab, which were connected to the pressure tubing. The two pressure tubing terminals connected the transducers to two points of pressure measurements, thereby detecting the pressure difference between these points. The pressure tubing extended to pressure tubing terminals, which were PVC pipe sections with holes drilled in and brass barbed fitting to connect the tubing. Tubing was color-coded to track which tubing was to be connected to the high or low side of the pressure transducer. Figure 4.12 shows one instrument unit to measure differential pressures, including transducer, pressure tubing and pressure tubing terminals.

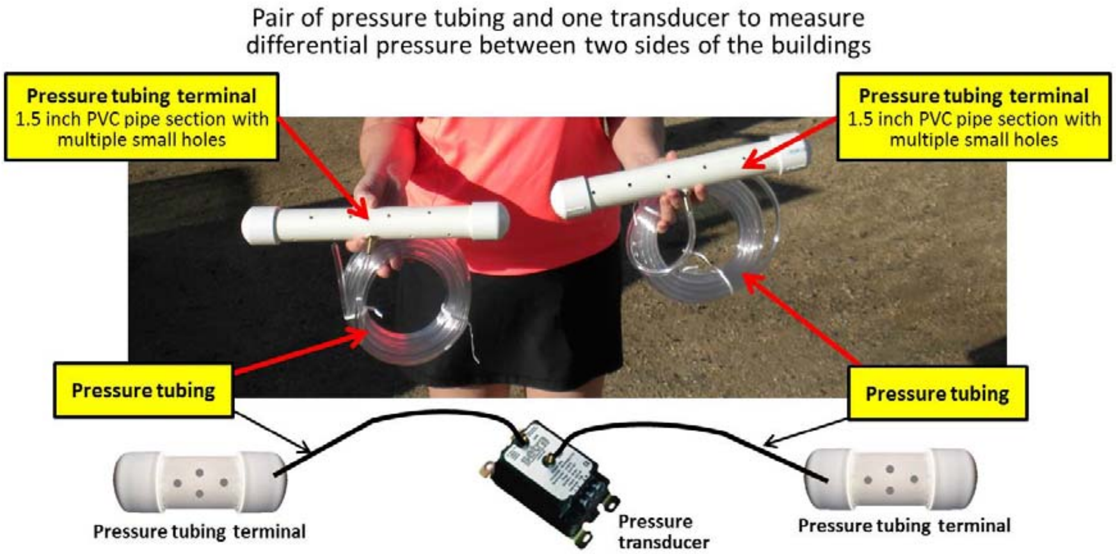


Figure 4.12: One unit to measure differential pressures, including differential pressure transducer, pressure tubing and pressure tubing terminals

Anemometers: Air velocity was measured with Degree Controls Accusense hotwire anemometers model F900-0-5-1-9-2 with the XS blade which has a range of 0 - 5 m/s air velocity and an accuracy of 0.5 % of reading or 1% of full scale.

Data acquisition was done with a National Instruments USB-6341 device X series DAQ (referred to as the DAQ) and a laptop computer equipped with National Instruments Signal Express software and set to log at a 5-Hz resolution. The anemometers were wire-connected to the DAQ using standard AC extension cords. Although these cords are bulky and heavy, they are easy to piece together to extend for a large site when needed. Pressure transducers were wired to the DAQ using 24-gauge stranded 2-conductor audio cable (these had been previously wired for another study, so extension cords were not used). Details of the wiring of the sensors to the DAQ can be found in Appendix A.

Additional instrumentation: Temperature (dry bulb and globe temperature) and relative humidity were logged using Onset Hobo ZW wireless sensors, a Hobo ZW receiver, and a laptop computer with Hoboware software. Data resolution was 1-minute (the maximum for that system). The globe thermometers were made with an Onset air/water/soil temperature probe with a standard table tennis (Ping-Pong) ball glued onto it using 5-minute 2-part epoxy, and then spray-painted with a black matt paint (instructions from George Loisos of Loisos + Ubbelohde Architects, personal communication, 2011).

4.3 Set-up of Instrumentation and Data Acquisition

These studies were run after 6:00 PM, when the office was mostly unoccupied. The DAQ device and laptop computer were located in the center of the room. Anemometers were mounted on stands at a height of 4 feet and 7.5 feet using a wire wrapped around the stand and holding the tip of the anemometer 3 inches away from the stand (see Figure 4.13). Pressure transducers where placed centrally in the room with flexible vinyl tubing extending to the test locations.

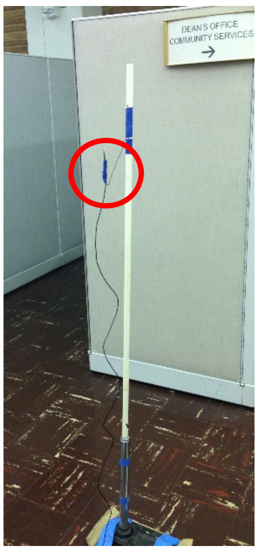


Figure 4.13:
Anemometer installed on stand with tip held 3" from stand at height of 4' above the floor

The sensors were arranged in the room as shown in Figure 4.14. The orange rectangle in the center indicates the DAQ device and laptop computer used for data acquisition. The sensor ID number, ai-01 for example, was based on the position in which it was connected to the data acquisition device. The acquisition software used this position as a default ID.

The height above the floor of the sensors (anemometers) or pressure tubing terminals (for the pressure transducers) is indicated in Figure 4.14 in brackets, [4'-0"] for example. Anemometers set to different heights at the same location were attached to the same stand. Sensors ai-09, ai-10, ai-12, and ai13 are the Setra pressure transducers, and ai-11 is the Halstrup Walcher pressure transducer. The red dotted lines in Figure 4.14 indicate the vinyl tubing for the pressure transducers which are attached to the "high" or the "low" pressure tabs on the differential pressure transducers (labeled H or L).

The grey lines in Figure 4.14 indicate cubicle partitions. The L0 labels in Figure 4.14 indicate the wood louvers on the South side of Room 301, which remained closed during the measurements (except for those at the face of the fans which always remained open). The L1 through L4 labels in Figure 4.14 indicate sections of glass louver windows on the North side of Room 301. During four different test scenarios, a combination of the louvers sections were open and closed, in order to create different boundary conditions, e.g. air inlet conditions. The L1 section of louver windows was located in the director's office, which was locked. Therefore those louvers were left closed for all four test scenarios. The director's office was separated from the rest of space by an 82-inch high internal wall.

The two exhaust fan units are indicated in Figure 4.14 by "Fan A" and "Fan B". Each exhaust fan unit had a pair of 30-inch propeller fans in the same housing. When a fan unit was turned on, both fans in the unit were operating at a "high" setting.

The prevailing wind is from the north side. The reference wind was measured by the weather station on the roof. This wind data was input into the CFD model.

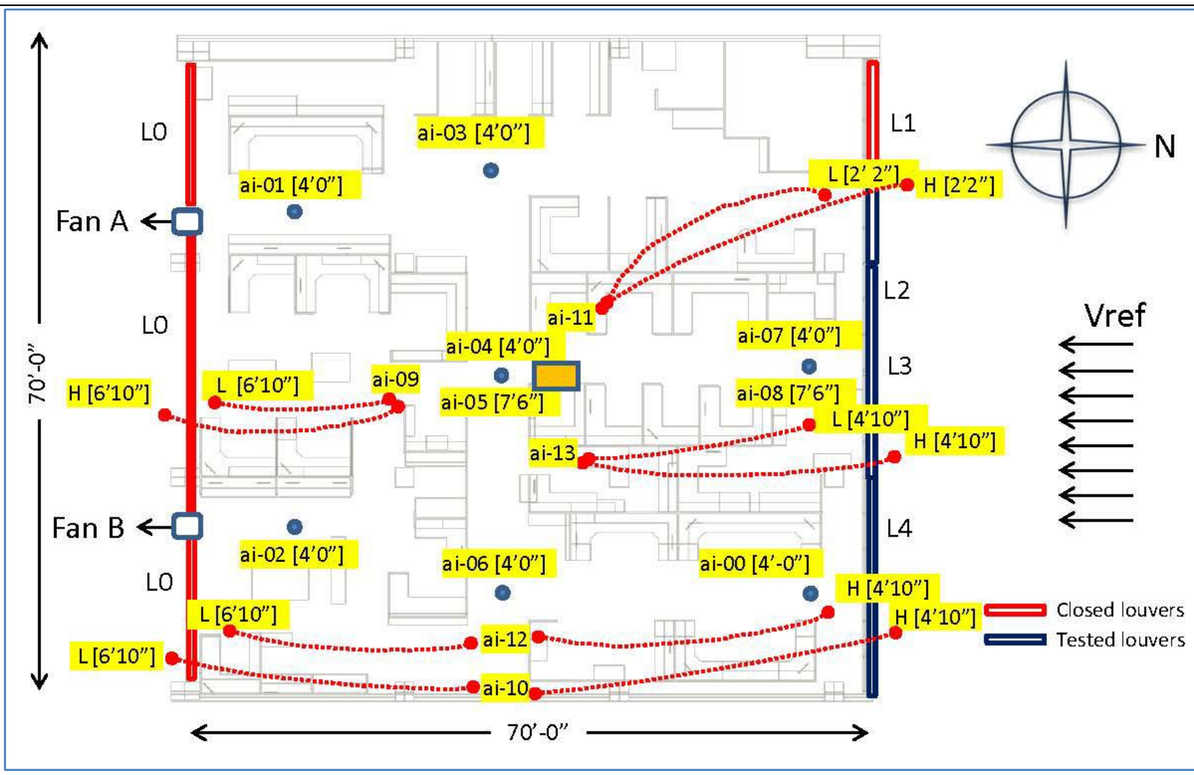


Figure 4.14: Location and identification of instrumentation (anemometers and differential pressure units, exhaust fan units and sections of movable windows inside Sinclair Room 301

Test scenarios: Four test scenarios were used in which different sets of louver windows on the North side were in the open or closed position and exhaust fan units fans were either on or off. The four test scenarios (e.g. Scenarios 1 through 4) are described in Table 4.1. Other than louvers at the face of the fans, all louvers on the south side (L0) were closed. These studies were conducted in four separate trials, on August 21, September 12, October 10, and October 30, 2014.

Table 4.1: Test matrix of the four test scenarios used to adjust the air flow conditions inside Room 301; the test scenarios represented combination of sections louver windows on the North wall of Room 301 either open or closed and exhaust fan units on the South side of Room 301 either operating or not operating

Label	Type	Scenario 1	Scenario 2	Scenario 3	Scenario 4
L0	Louver	closed	closed	closed	closed
L1	Louver	closed*	closed*	closed*	closed*
L2	Louver	open	open	open	closed
L3	Louver	open	open	open	closed
L4	Louver	open	closed	closed	open
A	Fan	on	on	on	on
B	Fan	on	on	off	on

** These louvers are closed in a locked office that was inaccessible.*

4.4 Locations of Instrumentation Determined in the initial CFD Scoping

The locations of anemometers and pressure tubing terminals were selected on the basis of initial CFD scoping simulations. The CFD scoping tests used approximated boundary conditions and a coarse mesh to provide the team with a basic understanding of internal air movement under the four test scenarios. The locations of anemometers and pressure tubing terminals in Room 301 were the same for all four test scenarios, and reflected preferred locations inside Room 301 where high velocity gradients were identified on the contour maps for all four test scenarios.

Figure 4.16 shows the layout of Room 301 and boundary conditions on the North and South side of the room which were used to define the input for the initial CFD scoping simulations. Figure 4.17 shows the four initial scenarios (scenarios A through D) and the CFD results depicted as colored contour maps indicating air velocity distribution inside Room 301. It should be noted that the scenarios A through D shown in Figure 4.17 were not the same as the final test scenarios 1 through 4, as described in Section

4.3. It was found that one final test scenario should include the operation of only one exhaust fan unit (e.g. fan unit A operating and fan unit B not operating).

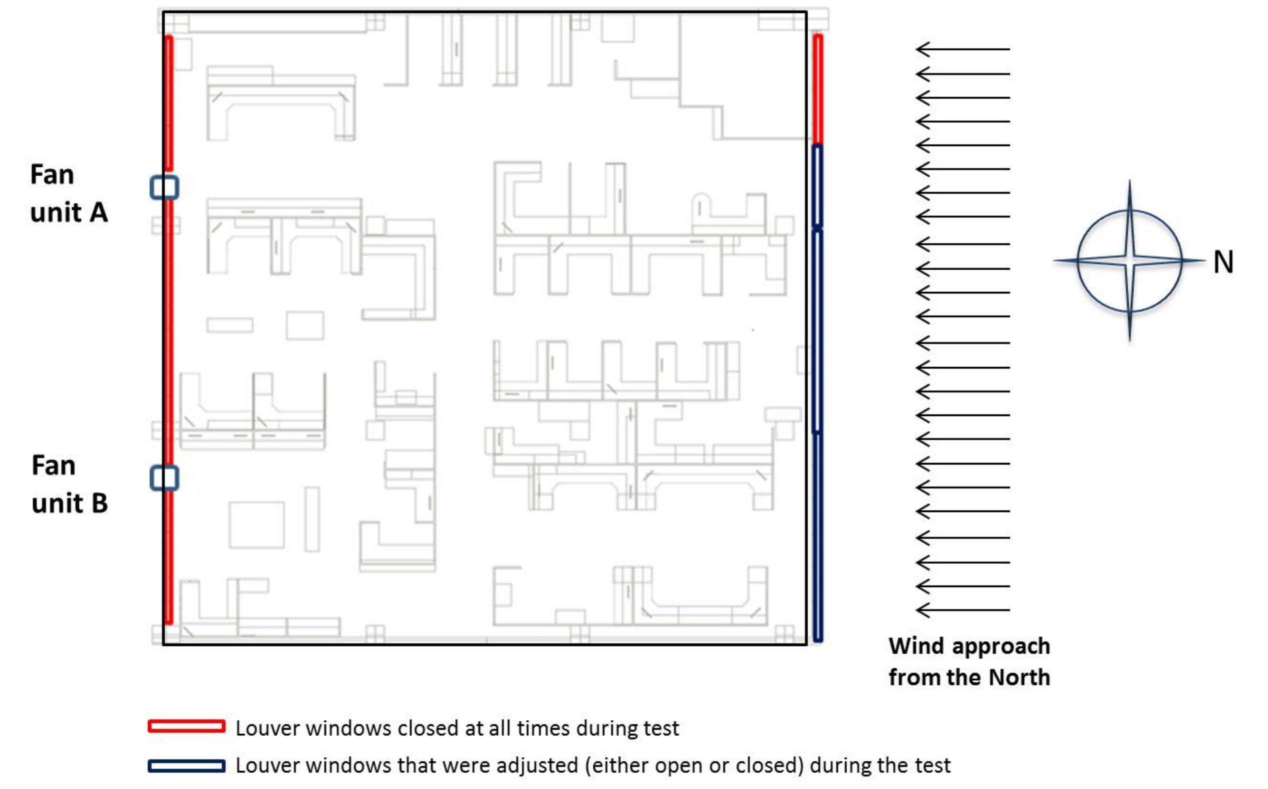
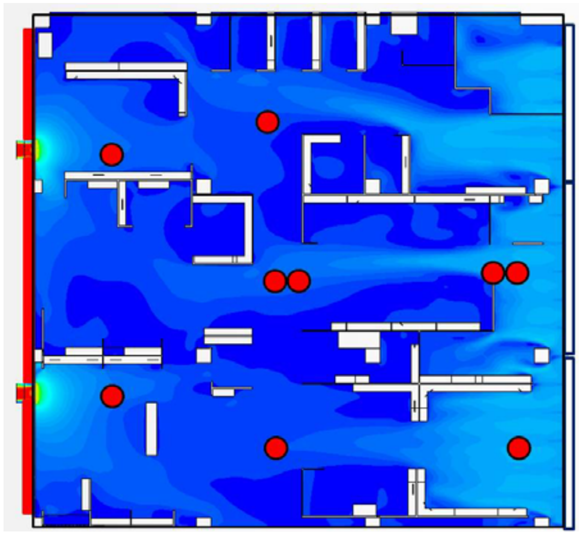
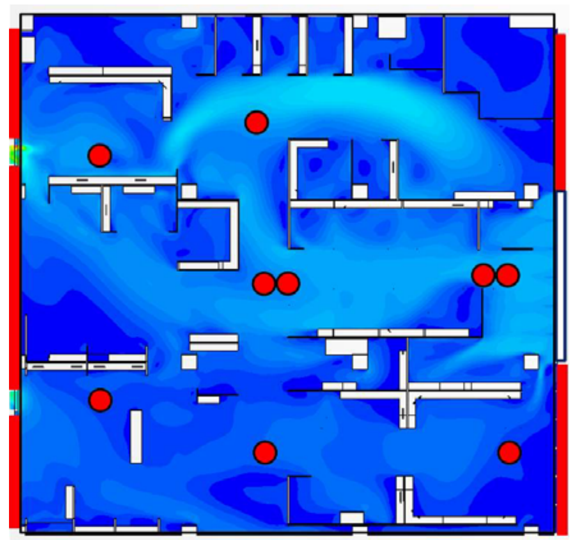


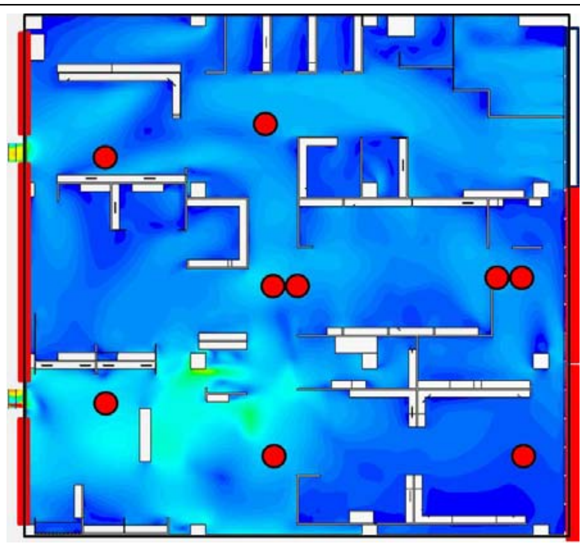
Figure 4.16: Layout and boundary conditions of Room 301 used in the initial CFD scoping simulations



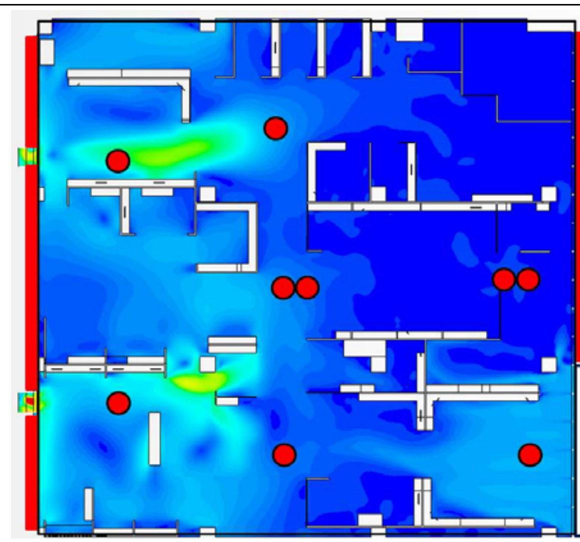
Scenario a used in the initial CFD scoping tests;
All widows on the North wall are open



Scenario B used in the initial CFD scoping tests;
Only section of widows on the North wall are open



Scenario C used in the initial CFD scoping tests;
Only section of widows on the North wall are open



Scenario D used in the initial CFD scoping tests;
Only section of widows on the North wall are open

Louver windows closed at all times during test

Louver windows that were adjusted (either open or closed) during the test

2 anemometers at same location, installed at different heights on the same stand

1 anemometer installed at location

Figure 4.17: Four scenarios A through D of air flow though Room 301 calculated with initial CFD scoping tests;
Note: the locations of anemometers in Room 301 was kept the same for all four scenarios

4.5 Running the Tests and Data Recording

The Signal Express software was setup ahead of time for these particular sensors and the project was saved. (A lengthy discussion of how this was done can be viewed in this video http://youtube/1BUXW_IhiDc.) Before data collection commenced for each of the four trials, caps were removed from the anemometers and technicians checked that the signals from the sensors were within the expected range to ensure the wiring was delivering a proper signal. If not, trouble-shooting was conducted on the wiring connections. For each of the four trials, data was collected for calibration first. Caps were placed on the anemometer tips and the tubing was removed from the pressure transducers. Data was collected at the 5-Hz resolution for a minimum of 10 minutes. The average reading for this calibration period was used to adjust the measurements taken during the scenarios. After calibration data was collected, caps were removed from the anemometers and tubes were replaced on the pressure transducers (color coding the tubing aided in re-connecting the correct corresponding tube to its port). The orientations of the anemometer blades were adjusted to be perpendicular to the assumed main direction of the air movement.

The data recording on the Signal Express software was started and stopped for each scenario to store data in separate files. Data was exported from Signal Express in a TDMS format (specific to National Instruments' software programs). Data was processed with Python scripts to convert to CSV format, add timestamps, calibrate and scale values, and average the 5-Hz data to 1-second resolution (Appendix B). A precision of 5 decimal points was used. The Signal Express software can be set to do the calibration and scaling calculations, but by doing it as a post-processing step, field measurements can be taken quickly without spending time to take the averages of the calibration run and inputting those values in the software settings. Post-processing allows ample time to carefully review calibration data for outliers before calculating the average for calibration. Occasionally, an outlier data point was detected and removed. One instance of an outlier was attributed to a technician stepping on tubing connected to a pressure transducer. Processed data was uploaded to a PostgreSQL database on a server. It retrospect, it was realized that tagging data with a trial number and scenario number would have been helpful (see completed data table structure in Table 4.5.1), so columns were added and the database was updated with SQL statements such as these:

```
UPDATE "2014-10-30-sinclair-indoor" SET trial_no = 4 WHERE datetime_stamp >
'2014-10-30 18:19:59' and datetime_stamp < '2014-10-30 19:59:00';

UPDATE "2014-10-30-sinclair-indoor" SET scenario_no = 1 WHERE datetime_stamp
> '2014-10-30 18:57:17' and datetime_stamp < '2014-10-30 19:07:59';
```

Table 4.2: PostgeSQL table structure for sensor data

datetime_stamp	sensor_id	value	trial_no	scenario_no
2014-10-30 18:57:54	ai11	0.04599	4	1
2014-10-30 18:57:55	ai1	0.43637	4	1
2014-10-30 18:57:54	ai9	0.00743	4	1
2014-10-30 18:57:54	ai10	-0.00531	4	1
2014-10-30 18:57:54	ai12	-0.00279	4	1
2014-10-30 18:57:55	ai0	0.3589	4	1
2014-10-30 18:57:54	ai6	0.74596	4	1
2014-10-30 18:57:54	ai8	0.21163	4	1
2014-10-30 18:57:54	ai5	0.65505	4	1

4.6 Analyzing the Data

Four trials were run in the latter half of 2014, and data from the last trial on October 30 was used for this analysis because the wind velocity was the highest on that night. Sensor data was analyzed using Tableau data visualization software. Weather station wind data was filtered for directions ranging from 30 to 60 degrees from north and velocity was filtered for $\geq$ 0.05 m/s (the threshold for the Onset U30 weather station). The reference wind used in the CFD simulation model was given the properties of the median direction and velocity calculated from this filtered wind data from the weather station. When using a "live" connection to the database, Tableau will not calculate the median. It can only be done when the data is extracted.

For weather station wind velocity and direction, a wind rose was created in Tableau by plotting wind direction and wind velocity on circular graph (see Appendix C for image that was imported) using the following calculated field as measures:

```
Winddir_x = [windvelocity]*cos(radians(90-[Wind Direction (degrees)]))

Winddir_y = [windvelocity]*sin(radians(90-[Wind Direction (degrees)]))
```

Other attributes of the wind rose:

- Wind velocity is represented by the distance from the center of the graph.
- Data points are not aggregated, so each measurement is shown and the dot is slightly transparent. (Data aggregation must be de-selected in Tableau.)
- Data points that overlap will make areas of frequently occurring wind direction and wind velocity combinations appear darker
- A YouTube video demonstration by Tuan Tran: http://youtube/YfLzCsgTYRk

Graphs and summary tables for air velocity data and pressure differentials were created in Tableau, filtering for trial and scenario number, sensor ID, etc. Basic instructions on how to create similar types of graphs in Tableau can be seen in these **videos**: temperature and humidity graphs by Eileen Peppard http://youtube/qMhiZ7ie8Ps and energy and voltage graphs by Eileen Peppard http://youtube/lnZhZ9-uNWY .

All air velocity and pressure differential data was filtered to match the timestamps of the filtered weather station data. The air velocity data had negative numbers filtered out. The pressure differential data from Setra transducers had negative numbers filtered out, but the data from the Halstrup Walcher pressure transducer is bi-directional, so a calculated field was created to keep negative values for only that sensor by using the following formula:

```
IF [Sensor Id]!= "ai11" AND [Value] < 0 THEN NULL ELSE [Value] END
```

SECTION 5 - RESULTS AND DISCUSSION

This section summarizes and discusses the results obtain in the test measurements.

5.1 Presentation of Weather Station Data

Weather station data was filtered for wind velocity $\geq$ 0.05 m/s and direction 30-60 degrees. The median values for each scenario are presented in Table 5.1. These values were used as the reference wind conditions for the CFD simulation. Figure 5.1 is a wind rose created with the weather station data during scenario 1. The distance from the center represents the velocity in m/s. Figure 5.2 is a trend line for the wind velocity during scenario 1. Additional weather data can be found in Appendix D.

Table 5.1.1: Median wind direction, median wind velocity, and N (number of observations) for all four test scenarios (scenarios 1 through 4) after weather station data was filtered for velocity $\geq$ 0.05 m/s and for direction ranging from 30 to 60 degrees from north

Scenario	Median Wind Direction (degrees)	Median Wind Velocity (m/s)	N
1	47.0	3.78000	504.0
2	49.1	4.28000	798.0
3	47.7	4.28000	854.0
4	49.1	3.78000	602.0

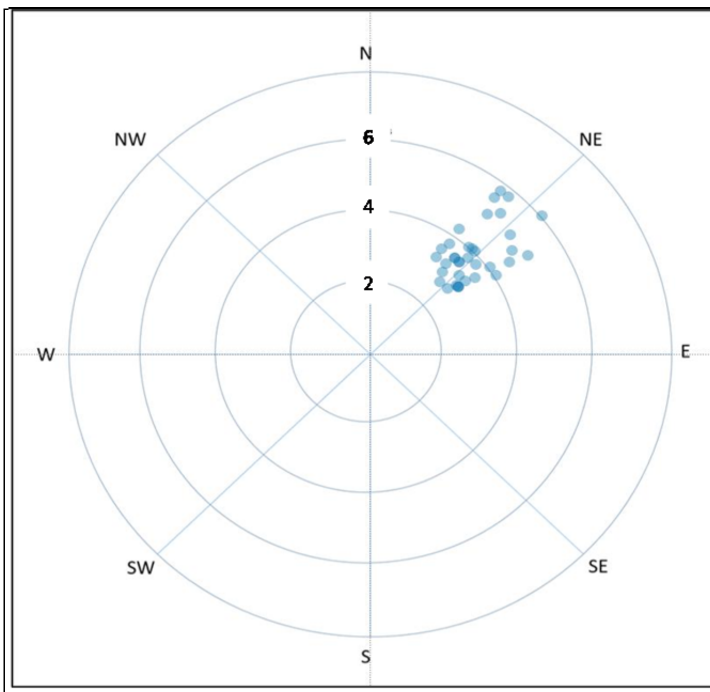


Figure 5.1:
Wind rose showing wind velocity (m/s; after filtering to > 0.05 m/s) and wind direction (after filtering for 30 – 60 degrees) measured by weather station during scenario 1

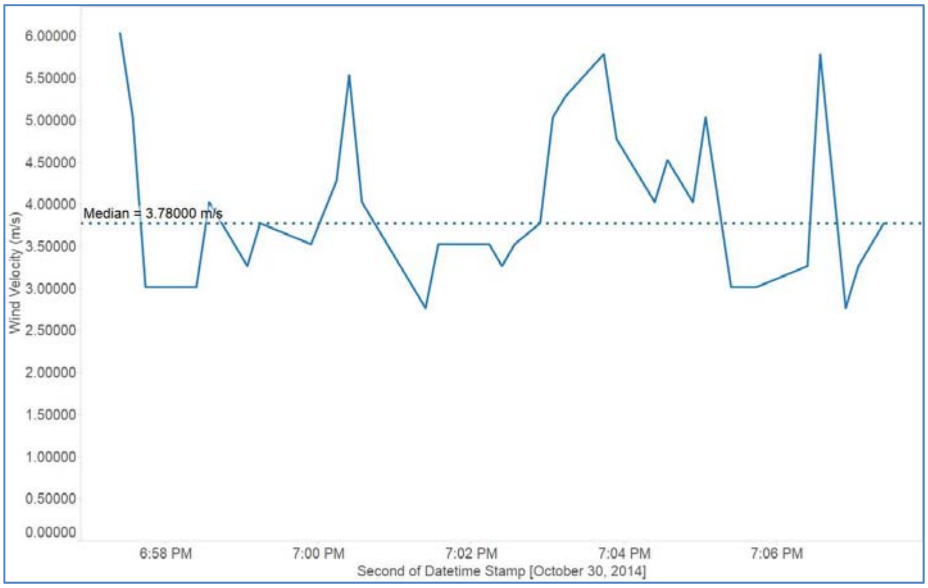


Figure 5.2: Trend line for wind velocity after filtering velocity to > 0.05 m/s and wind direction to 30 – 60 degrees, measured by weather station during scenario 1

5.2 Presentation of Air Velocity Data

Median air velocities calculated from anemometer data during all four scenarios are shown in Table 5.2. All data was filtered to remove negative values, and filtered to keep only data from timestamps that correspond to filtered weather station data (wind velocity >0.05 m/s and direction 30-60 degrees). Figure 5.3 shows a trend line for air velocity data from anemometer ai0 during scenario 1. A complete set of data can be found in Appendix E.

Table 5.2.: Median air velocities (m/s) for anemometers during all four scenarios

	scenario			
Sensor Id	**1**	**2**	**3**	**4**
ai0	0.67010		0.00494	1.04626
ai1	0.38909	0.56566	0.37840	0.32290
ai2	0.88446	0.85034	0.40430	0.86597
ai3	0.10444	0.19139	0.12458	0.06847
ai4	0.17902	0.61870	0.37265	0.21623
ai5	0.44745	0.87849	0.34220	0.16132
ai6	0.62387	0.00763	0.00064	0.73733
ai7	0.11854	0.25523	0.08442	
ai8	0.34113	0.45583	0.19601	

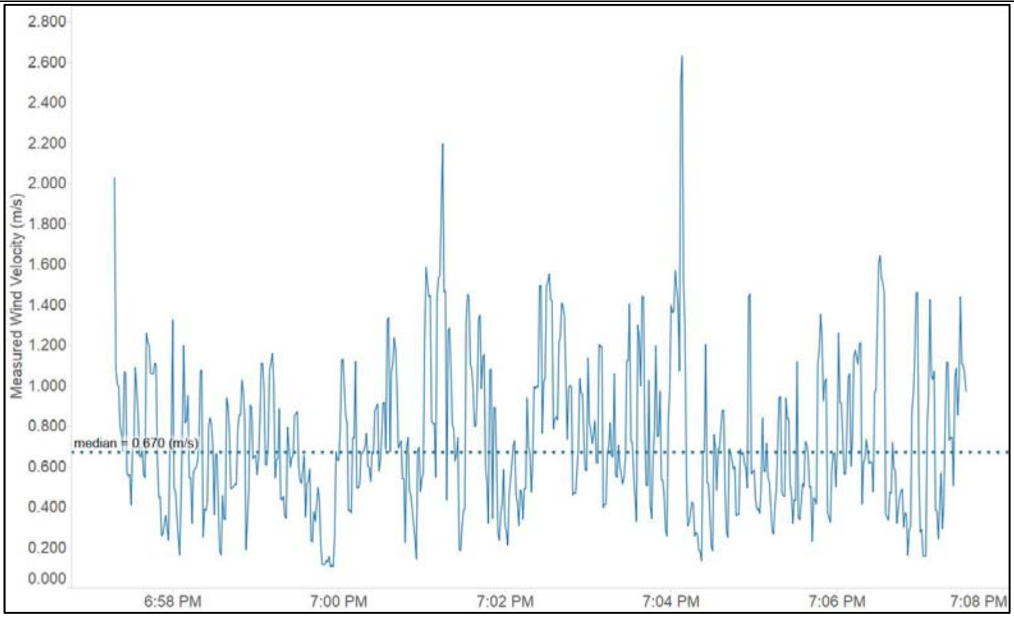


Figure 5.3: Trend line for air velocity measurements taken from anemometer ai0 during scenario 1

5.3 Presentation of Differential Pressure Data

Median pressure differentials calculated from pressure transducer data during all four scenarios are shown in Table 5.3. Data was filtered to remove negative values for the Setra transducers only (but not ai11, the bi-directional Halstrup Walcher sensor), and filtered to keep only data from timestamps that correspond to filtered weather station data (wind velocity >0.05 m/s and direction 30-60 degrees). Figure 5.4 shows a trend line for pressure differential data from pressure transducer ai10 during scenario 1. Additional data can be found in Appendix F.

Table 5.3: Median differential pressures (in. w.c.) for differential pressure transducers during all four scenarios 1 - 4

	scenario			
Sensor Id	1	2	3	4
ai9	0.00831	0.01193	0.00862	0.01292
ai10	0.01278	0.01301	0.01360	0.01183
ai11	0.00913	0.03016	0.00796	0.01498
ai12	0.00100	0.00042	0.00086	0.00075

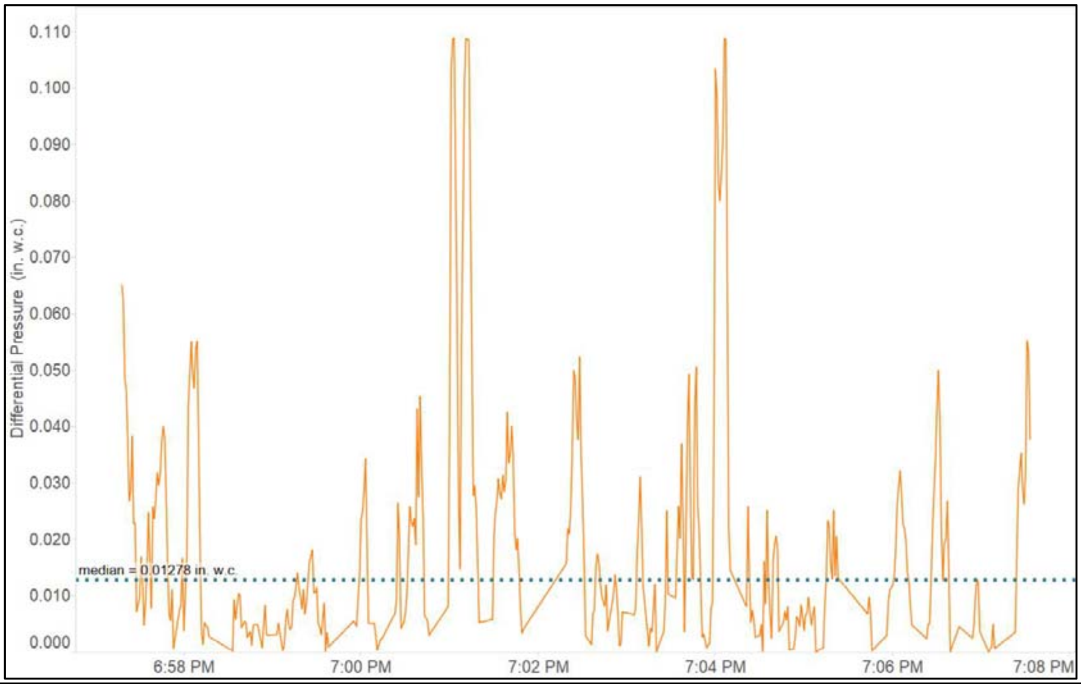


Figure 5.4: Trend line for differential pressures measured by pressure transducer ai10 in scenario 1

SECTION 6 - CONCLUSIONS AND LESSONS LEARNED

This section summarizes the conclusions and lessons learned from the project work of this study. These conclusions and lessons learned apply to test sites which have similar conditions as the present test site on the University of Hawaii Manoa campus.

Prepare the test site and general advice

- Plan trials for when the space is unoccupied and notify the occupants.
- Print out data sheet with diagram of sensor locations, and matrix of scenarios and put on clipboard. Do this for every trial and label with the date.
- Tools to have on hand: screw drivers, voltage meter, tape, and marker.
- At the test site, be sure there are no fans running.
- Advise test site occupants of the plan to start the trial.

Prepare the instrumentation for the test site

- Prepare all wiring, stands, tubing, and terminal ends for sensors. Number and color-code parts as needed. Use something to neatly store the wrapped wires and extension cords (hook and loop straps work well).
- The Setra pressure transducer that measures current instead of voltage is more trouble to setup, so if you don’t need all the sensors, leave that one out.
- Have all instruction booklets for sensors and DAQ device on hand for trouble-shooting purposes.
- Prepare laptop computer:
    - Setup the Signal Express “project” ahead of time, configured to record all voltage and current inputs, and save with date of trial.
    - Have fully charged laptop, power cord, and optical mouse in a computer bag.
- Set the weather station data resolution to 5 Hz.

Deploy the instrumentations and data acquisition

- After connecting sensors to the DAQ device, check that the signals look as expected, if not:
    - Make sure the sensor is checked in the Signal Express software to display the reading.
    - Check the sensor is getting power (use a multimeter).
        - Make sure the correct voltage from power supply is sent to sensor – check both the DAQ and the sensor sides.
        - Check the sensor is not damaged.
    - Re-tighten all wire connections.
- Always do a calibration run first.
- During data collection:
    - Don't step on flexible tubing for pressure sensors
    - Don't rush past sensors, creating air movement
- Write down the start time for the scenario on the data sheet. Don't rely on memory for which trial is which in the event that you repeat a scenario and keep both sets of data. In lieu of that, you can name the SignalExpress folder with the scenario number.
- When finished, reset the weather station data resolution back to its original setting.
- Photograph data sheet with a cell phone and email it as a backup.
- In the database, use the same naming conventions for columns of data (e.g. always use "datetime" instead of changing to "timestamp" on the second trial). The table for a previous trial can be copied and the data deleted to ensure the same names and datatypes are used. It would be helpful to have the units in the column name (e.g. air velocity_mps).
- For weather station data, be sure to export the data in consistent units from the Hoboware software. In this case, m/s was chosen instead of mph.
- In future, tag the data with the trial and scenario number during the post-processing stage, using the Python script (script will need editing).

ACRONYMS AND UNITS

ACRONYMS

CFD	Computational Fluid Dynamics
CSV	Comma-separated Value Files
DAQ	Data Acquisition
PVC	Polyvinyl Chloride
TDMS	LabVIEW Test Data Exchange Stream (National Instruments Corporation)

UNITS

CFM	Cubic Feet per Minute
Hz	Hertz
in. W.C.	Inches of Water Column
m/s	Meter per Second
Pa	Pascal
”	Sign for inches
'	Sign for feet

APPENDIX A

A DAQ WIRING

Appendix A: Wiring of the National Instruments DAQ

Introduction

The National Instruments USB-6341 is a data acquisition device (DAQ) with multiple channels (Figure A.1). This provides a means to monitor multiple wired sensors deployed for an experiment. For this study, the device was used to monitor pressure differential and air velocity. The setup was used for multiple short-term trials at different sites. Therefore, it was beneficial to create a break-out board for the DAQ that would allow for easier connections. This was a quick and effective solution for rapid sensor deployment, reducing the setup time by at least 50% while keeping the sensors organized.

Figure A.1: National Instruments Data Acquisition Device (DAQ).

Break-out Board

The break-out board for the DAQ is a simple wooden frame to protect the device with terminal heads (Figure A.2). Using these terminal heads avoids having to repeatedly re-wire the DAQ itself, preventing damage to the DAQ. The break-out board is uniquely made for the sensors that were repeatedly deployed for this study. The break-out board terminals can be seen in Figure A.3 which shows all the connections and labels corresponding to the default labels in the Signal Express software.

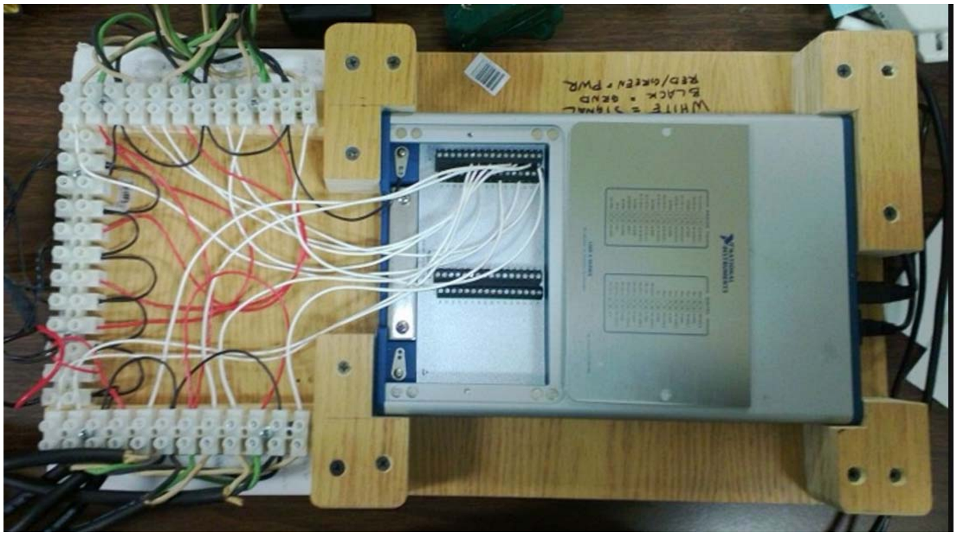


Figure A.2: Break-out board for the DAQ

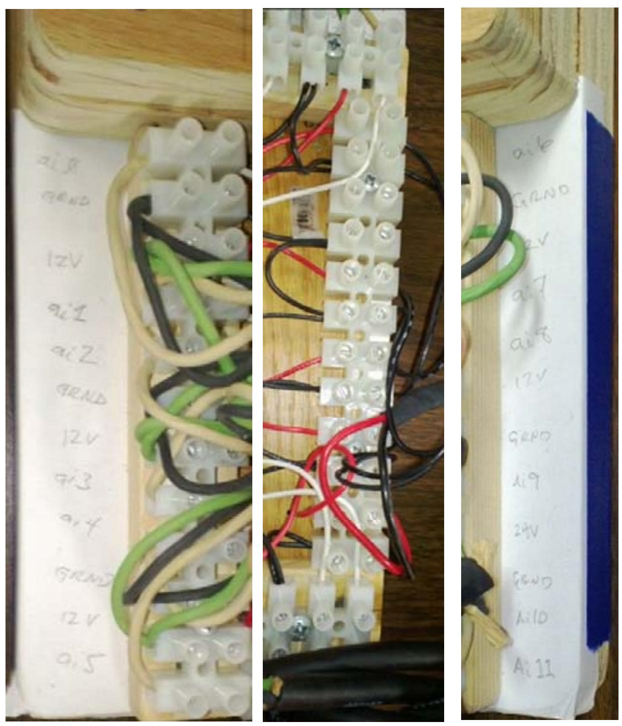


Figure A.3: Break-out board terminals and labeling.

Anemometers

The Accusense F900 anemometers were voltage-based and required 3 wires: a power supply, a ground, and a signal. They required a 7 - 13 VDC power supply and the adapter selected could power multiple sensors at the same time. The anemometers were wired to female plug adapters (Figure A.4) in order to attach to 3-prong heavy-duty extension cords. The male plug adapter was wired to the break-out board. Extension cords were used because CAT5 cables could not carry the required power to the sensors over a long distance. They were also better suited to the different environments in which the sensors might be deployed such as large outdoor areas where multiple extension cords could be coupled together.

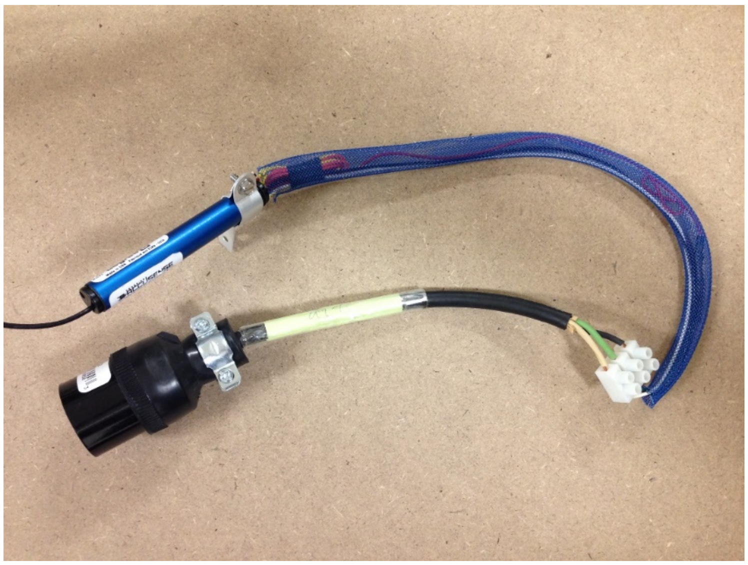


Figure A.4: Anemometer wiring adapted for attachment to heavy-duty 3-prong extension cord

Pressure transducers

The voltage-based pressure sensors required 3 wires: a power supply, a ground, and a signal. The Setra Model 264 pressure sensors used a power supply with voltage of 9 - 30 VDC which powered multiple sensors. The Halstrup Walcher P-26 pressure sensor was voltage-based and required a 24 V power supply. The pressure sensors were wired with 24 AWG wire (extension cords were not needed). The current-based pressure sensor (one of the Setra 264 sensors) required only 2 wires (a power supply and ground) and a resistor. The resistance was measured with a multimeter and this value in ohms was entered into the Signal Express software as part of the setup.

Voltage Wiring

The voltage wiring consisted of a power supply, ground, and signal. The power supply is the excitation for the sensor, while the ground provides a way to complete the circuit with the monitoring device. The signal is the data that the sensor transmits back to the DAQ. Table A.1 lists the color coding for the wires in the break-out board. Figure A.5 shows the wiring diagram.

Table A.1: Anemometer General Voltage Wire Color Coding.

Color	Type
White	Signal
Black	Ground
Red/Green	Power Supply

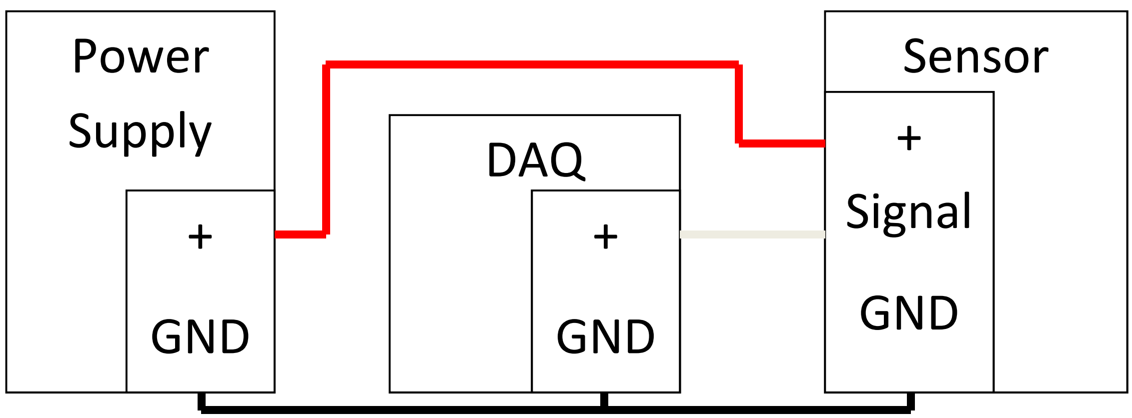


Figure A.5: Voltage Wiring Diagram

Current Wiring

The current wiring consisted of a power supply and ground. It also used a shunt-resistor to relate the corresponding current to a voltage through Ohm's Law. The DAQ measured the voltage drop across the resistor and determined the corresponding current. The power supply and ground simply created a current. The shunt-resistor is essential for this type of sensing. There are also other means of measuring through an induction method, but adding a resistor in parallel is adequate. Table A.2 lists the color coding for the wires in the current break-out board. Figure A.6 shows the wiring diagram.

Table A.2: General Current Wire Color Coding

Color	Type
Red/Green	Power Supply
Black	Ground

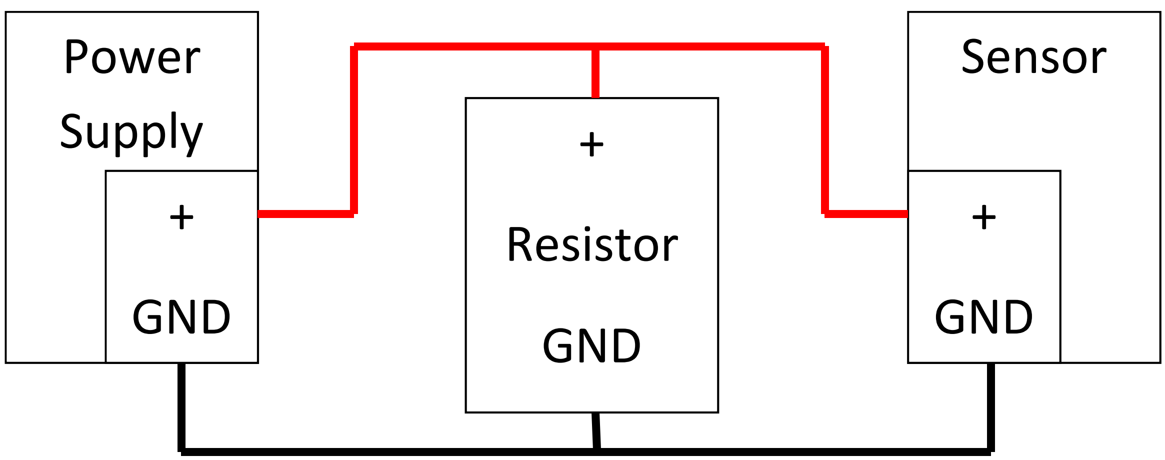


Figure A.6: General current wiring diagram

Discussion

The DAQ is an extremely useful tool for monitoring multiple sensors. This break-out board provided an effective solution for reducing the time to setup by at least 50%. The break-out also protects the DAQ from prolonged or immediate damage. There could be better alternatives to the break-out board, such as the Onset Hobo ZW wireless network which does not require wires to connect and power. The limitations of Hobo wireless system: it can attach only proprietary devices (the DAQ can utilize a large variety of sensors) and it has a maximum data acquisition resolution of 1 minute. A possible solution could be to create a simple transmitter-receiver device that could attach to each of the DAQ's channels. The data would be received from a wireless node with a unique sensor. This would help prevent the wiring problems.

APPENDIX B

PYTHON SCRIPTS

APPENDIX B: Instructions and Python Scripts to Process Sensor Data

Instructions:

1. Export the Signal Express data (TDMS file format) and copy into a folder named 'DataFiles.'
2. Create folders 'CalibratedCSVs', 'ConvertedCSVs', and 'ReshapedCSVs.' If this is the second time data is processed, move the previous files to an archive and leave these folders empty.
3. Run the Python script 'driver.py' which will process the raw data files. The 'driver.py' script will call upon the class file, 'NISignalExpressUtilitiy.py' to perform the necessary functions of converting to CSV format, adding timestamps, calibrating, scaling, and averaging data in the correct sequence.
4. Go into the folder 'ConvertedCSVs' and open the trial[1] CSV file.
5. For each device, average their values.[2]
6. Open 'NISignalExpressCalibrate.csv' and replace the 'preoffset' values with the negative averages calculated in step 5.
7. For 'ai11', which is the P26 pressure sensor, we apply the following equation:

$$preoffset = [|5 - Average| \times (-2)] - 10$$

8. Save the changes you made to the 'NISignalExpressCalibrate.csv' and then repeat step 2 (empty the folders).
9. Repeat step 3.
10. The folder 'CalibratedCSVs' should now contain the final results and the CSV files can be uploaded to the database.

[1] Trial CSV should be the one with the earliest time stamp.
[2] Using Excel Spreadsheet's AVERAGE function to complete the task.

Python Script saved as "driver.py":

```
import NISignalExpressUtility
import os
from DataFiles import __data__
from ConvertedCSVs import __converted__
from ReshapedCSVs import __reshaped__
from CalibratedCSVs import __calibrated__

data_location = os.path.dirname(os.path.abspath(__data__.__file__))
converted_location = os.path.dirname(os.path.abspath(__converted__.__file__))
reshaped_location = os.path.dirname(os.path.abspath(__reshaped__.__file__))
calibrated_location =
os.path.dirname(os.path.abspath(__calibrated__.__file__))

print "data: ", data_location
print "converted: ", converted_location
print "reshaped: ", reshaped_location
print "calibrated: ", calibrated_location

data_folders_locations = []
for folder_name in os.listdir(data_location):
    if ".py" not in folder_name:
        data_folders_locations.append(os.path.join(data_location,
folder_name))


for data_folders in data_folders_locations:
    NIExtract = NISignalExpressUtility.NISignalExpressUtility(\
        data_folders, converted_location,\
        reshaped_location, calibrated_location)
    NIExtract.convert_to_csv()
    NIExtract.reshape_csv()
    NIExtract.calibrate_output()
```

Reference file ‘NISignalExpressCalibrate.csv’ with sample calibration data:

```
id,premultiplier,preoffset,multiplier,offset
ai0,1,-0.024780644,1.25,0
ai1,1,-0.024960358,1.25,0
ai2,1,-0.068046865,1.25,0
ai3,1,-0.022794628,1.25,0
ai4,1,-0.024298522,1.25,0
ai5,1,-0.033858482,1.25,0
ai6,1,-0.042690422,1.25,0
ai7,1,-0.023685052,1.25,0
ai8,1,-0.022548731,1.25,0
ai9,1,-0.164202876,0.02,-0.001
ai10,1,-0.209948918,0.02,-0.001
ai11,2,-10.0572,0.0040146,0
ai12,1,-0.15691658,0.02,-0.001
ai13,1,-0.004277969,0.02,-0.001
```

Python script saved as “NISignalExpressUtility.py”:

```
#-------------------
# Author: Christian A. Damo
# file name: NISignalExpressUtility.py
# rev. by: Reed Shinsato
# rev. date: 2014-07-11
#------------------
#
# Patch Notes: Cleaned up class
#
#------------------
"""
    This script converts .tdms files from NISignalExpress into .csv files.
    Then, it pushes the .csv files to a server with a specified ip.
"""

# Import Libraries
from nptdms import TdmsFile
import sys
import os
import datetime
import csv
notes = {'ai0':"NE Anemometer 4'",
       'ai1':"SW Anemometer 4'",
       'ai2':"SE Anemometer 4'",
       'ai3':"W Anemometer 4'",
       'ai4':"Center Anemometer 4'",
       'ai5':"Center Anemometer 7'6",
       'ai6':"E Anemometer 4'",
       'ai7':"N Anemometer 4'",
       'ai8':"N Anemometer 7'6",
       'ai9':"S Setra",
       'ai10':"E Outer Setra",
       'ai11':"NW P26",
       'ai12':"E Inner Setra",
       'ai13':"N Setra",}
# Create Classes
class GroupChannel:
    """
        This class holds the group and channel names from a metafile.
    """
    def __init__(self, meta_filename):
        print
        self._channel_type = ""
        self._start_time = datetime.datetime.now()
```

```
        self._meta_filename = meta_filename
        self._group_name = ""
        self._channel_names = []
        self.__get_group_name(self._meta_filename)
        # self.__str__()

    def __str__(self):
        print "\nCalling GroupChannels __str__()"
        print "Type: ", self._channel_type
        print "Start: ", self._start_time
        print "Group Name: ", self._group_name
        print "Channel Names: \n\n", self._channel_names

    def __get_start_and_names(self, meta_filename, temp_names):
        """
            This function stores the start time and channel names
        """
        # Check the meta_file for " ", which is the line of the channel
names.
        # Check the meta_file for "Log start time", which holds the start
time.
        meta_file = open(meta_filename)
        for line in meta_file:
            if line[0] == " ":
                 temp_names.append(line)
            if "Log start time" in line:
                self._start_time =
self.__start_convert_to_datetime_object(line)
        meta_file.close()

    def __start_convert_to_datetime_object(self, line):
        """
            This function turns the given line into a datetime object.
        """
        # Find each element of the given line
        # Turn the elements into a datetime object
        # Return the datetime object
        line = line.split(" ")
        date = line[3]
        date = date.split("/")
        year = int(date[2])
        day = int(date[1])
        month = int(date[0])
        time = line[4]
        time = time[:-1]
        time = time.split(":")
```

```
        hour = int(time[0])
        minute = int(time[1])
        second = time[2].split(".")
        second = int(second[0])
        begin_time = datetime.datetime(year, month, day, hour, minute,
second)
        return begin_time

    def __get_group_name(self, meta_filename):
        """
            This functions determines the group_name
        """
        # Get the start time and channel names
        # Determine the timestamp
        # Store the channel names
        # Determine the data type
        # Create the group name
        temp_names = []
        self.__get_start_and_names(meta_filename, temp_names)
        for line in temp_names:
            temp_line = line.split("-")
            timestamp = temp_line[0]
            timestamp = timestamp[5: -1]
            channel_name = temp_line[-1]
            channel_name = channel_name[1: -1]
            self._channel_names.append(channel_name)
            data_type = temp_line[1]
            data_type = data_type[1: -1]
            self._channel_type = data_type
            self._group_name = (timestamp + " - " + data_type +
                                " - " + "All Data")

    def return_group_name(self):
        """
            This function returns the group name.
        """
        # Retunr group_name
        return self._group_name

    def return_channel_names(self):
        """
            This function returns the channel names.
        """
        # Return channel_names
        return self._channel_names
```

```
    def return_start_time(self):
        """
            This function returns the start time.
        """
        # Return start_time
        return self._start_time

class NISignalExpressUtility:
    def __init__(self, data_location = os.path.dirname(\
        os.path.abspath(__file__)), csv_location = os.path.dirname(\
        os.path.abspath(__file__)), reshaped_location = os.path.dirname(\
        os.path.abspath(__file__)), calibrated_location = os.path.dirname(\
        os.path.abspath(__file__))):
        self._data_location = data_location
        self._csv_location = csv_location
        self._reshaped_location = reshaped_location
        self._calibrated_location = calibrated_location
        self._meta_voltage_filename = (os.path.join(self._data_location,\
            "Voltage_meta.txt"))
        self._meta_current_filename = (os.path.join(self._data_location,\
            "Current_meta.txt"))
        self._tdms_voltage_filename = (os.path.join(self._data_location,\
            "Voltage.tdms"))
        self._tdms_current_filename = (os.path.join(self._data_location,\
            "Current.tdms"))
        self._channel_names = []
        self._tdms_filenames = []
        self._GroupChannels = []
        self._tdms_to_csv_filename = str(os.path.split(data_location)[-1]) +\
            "_converted.csv"
        self._reshaped_filename = str(os.path.split(data_location)[-1]) +\
            "_reshaped.csv"
        self._converted_to_csv = False

    def __check_for_group_channels(self, tdms_filename, tdms_filenames,\
            meta_filename, channel_names, GroupChannels):
        """
            This function creates the GroupChannels.
        """
        # Create a tdms object
        # Create a GroupChannel object
        # Add to the list of channel_names
        # Add to the list of tdms_filenames
        # Add to the list of GroupChannels
        tdms_file = TdmsFile(tdms_filename)
        TypeGroupChannel = GroupChannel(meta_filename)
```

```
        type_channel_names = TypeGroupChannel.return_channel_names()
        channel_names.append(type_channel_names)
        tdms_filenames.append(tdms_file)
        GroupChannels.append(TypeGroupChannel)

    def __tdms_to_csv_file(self, channel_names, tdms_filenames,
                           GroupChannels):
        """
            This function will convert .tdms files into .csv files
        """
        # Get the data from the .tdms file
        #    Create a channel object from a tdms object
        #    Get the data from the channel
        #    Get the time from the channel
        # Clean up the channel_names to be a single array
        # Save the data into the output file
        #    Save the timestamp using delta from the start_time
        #    Save the data in a row relative to the channel_id column

        datas = []
        times = []
        for channel_names_index in range(0, len(channel_names)):
            for channel_name in channel_names[channel_names_index]:
                channel = (tdms_filenames[channel_names_index].
              object(GroupChannels[channel_names_index].
              return_group_name(), channel_name))
                data = channel.data
                datas.append(data)
                times = channel.time_track()

        temp_channel_names = []
        for channel_names_index in range(len(channel_names)):
            for name in channel_names[channel_names_index]:
                temp_channel_names.append(name)

        channel_names = []
        channel_names = temp_channel_names


        converted_file = open(os.path.join(self._csv_location,\
            self._tdms_to_csv_filename),"wb")
        writer = csv.writer(converted_file)

        new_row = list(channel_names)
        new_row.insert(0, "timestamp")
        writer.writerow(new_row)
```

```
        #iterates through each time element
        for times_index in range(len(times)):
            #created delta object
            delta = datetime.timedelta(seconds = times[times_index] + 1)
            #creates the actual printed time
            current_time = GroupChannels[0].return_start_time() + delta
            #creates new line with time as element 0
            new_row = [current_time]
            #iterating through the datas list
            for datas_index in range(len(datas)):
                data = datas[datas_index][times_index]
                new_row.append(data)
            writer.writerow(new_row)
            #print new_row
            #print len(new_row)
            #raw_input("Press Enter to continue...")
        converted_file.close()


    def reshape_csv(self):
        """
            This function reshapes the data to a specific table format.
        """
        # Find the correct input file
        # Create an output file
        # Write the header
        # Clean up the channel_names
        # Write the output row with the correct format
        #    Add time to the output row
        #        Add the channel_names to the row
        #        Add the data to the row
        #        Write the row with [time, channel_names, data]
        if self._converted_to_csv == True:
            channel_names = list(self._channel_names)
            input_file = (open(os.path.join(self._csv_location,\
                self._tdms_to_csv_filename), "r"))
            reader = csv.reader(input_file)

            reshaped_file = (open(os.path.join(self._reshaped_location,\
                self._reshaped_filename), "wb"))
            writer = csv.writer(reshaped_file)

            reader.next()
            new_row = ["datetime", "position", "value"]
            writer.writerow(new_row)
```

```
            temp_channel_names = []
            for channel_names_index in range(len(channel_names)):
                for names in channel_names[channel_names_index]:
                    temp_channel_names.append(names)
            channel_names = []
            channel_names = temp_channel_names
            new_channel_names = []
            for name in channel_names:
                name = name.split("_")
                name = name[1]
                new_channel_names.append(name)

            for row in reader:
                current_time = row[0]
                row = row[1:]
                for new_channel_names_index in range(len(new_channel_names)):
                    new_row = []
                    new_row.append(current_time)

new_row.append(new_channel_names[new_channel_names_index])
                    new_row.append(float(row[new_channel_names_index]))
                    writer.writerow(new_row)

            input_file.close()
            reshaped_file.close()

    def convert_to_csv(self):
        """
            This function extracts the tdms data to the correct format.
        """
        # Check for voltage and current group channels
        # Convert the tdms files to csv
        # Reshape the csv files to the correct format
        files_found_voltage = True
        files_found_current = True
        try:
            self.__check_for_group_channels(self._tdms_voltage_filename,\
                self._tdms_filenames, self._meta_voltage_filename,\
                self._channel_names, self._GroupChannels)
        except:
            print "Note: No Voltage Files"
            files_found_voltage = False
        try:
            self.__check_for_group_channels(self._tdms_current_filename,\
                self._tdms_filenames, self._meta_current_filename,\
```

```
                self._channel_names, self._GroupChannels)
        except:
            print "Note: No Current Files"
            files_found_current = False

        if files_found_current == True or files_found_voltage == True:
            self.__tdms_to_csv_file(self._channel_names,
self._tdms_filenames,\
                self._GroupChannels)
            self._converted_to_csv =  True

    def add_calibration(self, sensor_id, premultiplier,
                        preoffset, multiplier, offset):
        """
            This function adds a calibration to NICalibrate.csv.
        """
        # Open the calibration file
        # Check if the sensor_id is in the calibration file
        # Delete the calibration for the sensor_id
        # if it is in the calibration file
        # Append the new calibration to the calibration file
        calibration_filename = "NICalibrate.csv"
        delete_id = False
        sensor = str(sensor_id)
        with open(calibration_filename, "r") as calibration_file:
            for row in csv.reader(calibration_file):
                if sensor in row:
                    delete_id = True

        if delete_id == True:
            self.delete_calibration(sensor_id)

        with open(calibration_filename, "a") as calibration_file:
            row = (str(sensor_id) + "," +\
                   str(premultiplier) + "," +\
                   str(preoffset) + "," +\
                   str(multiplier) + "," +\
                   str(offset) + "\n")
            calibration_file.write(row)


    def delete_calibration(self, sensor_id):
        """
            This function deletes a calibration from NICalibrate.csv.
        """
        # Copy NICalibrate.csv into a temp.csv
```

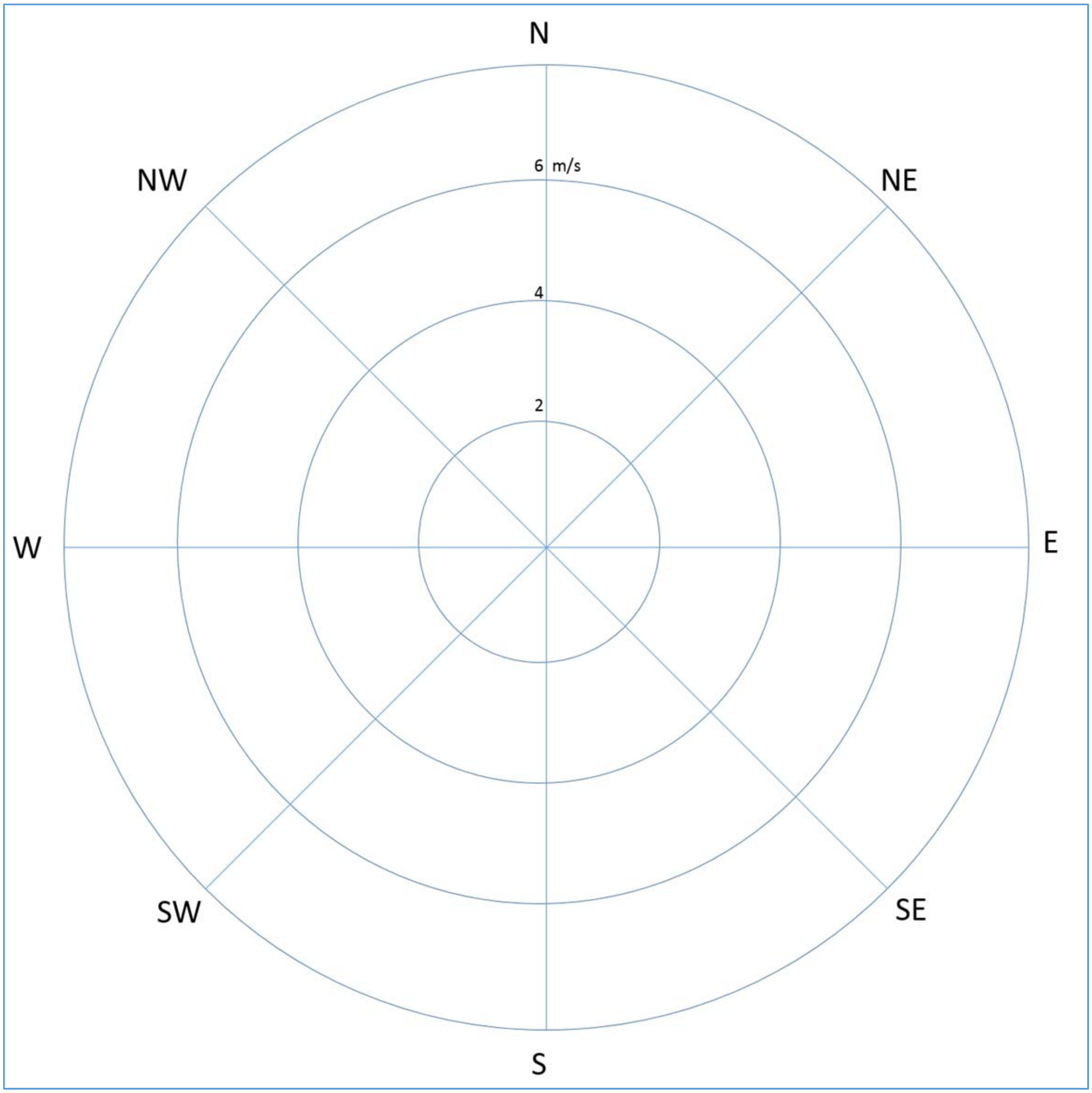
```
    # skipping the sensor_id to be deleted
    # Delete the old calibration file NICalibrate.csv
    # Rename temp.csv into the new calibration file NICalibrate.csv
    calibration_filename = "NISignalExpressCalibrate.csv"
    temp_filename = "temp.csv"
    sensor = str(sensor_id)
    with open(temp_filename, "wb") as temp_file:
        with open(calibration_filename, "r") as calibration_file:
            for row in csv.reader(calibration_file):
                if sensor not in row:
                    csv.writer(temp_file).writerow(row)

    os.remove(calibration_filename)
    os.rename(temp_filename, calibration_filename)


def __calibrate_value(self, sensor_id, value):
    """
        This function calibrates the value of the senor_id.
    """
    # Open the calibration file
    # Check if the sensor_id of the calibration file
    # to determine the multipliers and offsets
    # Return the value with calibration
    calibration_filename = "NISignalExpressCalibrate.csv"
    calibration_file = open(calibration_filename, "r")
    reader = csv.reader(calibration_file)

    value = float(value)

    premultiplier = 1.0
    preoffset = 0.0
    multiplier = 1.0
    offset = 0.0

    row = reader.next()
    for row in reader:
        if row[0] == str(sensor_id):
            premultiplier = float(row[1])
            preoffset = float(row[2])
            multiplier = float(row[3])
            offset = float(row[4])

    value = multiplier * ((premultiplier * value) + preoffset) + offset

    calibration_file.close()
```

```
        return value

    def clean_folder(self, folder_location):
        """
            This function deletes all the output files created.
        """
        folder_files_list = os.listdir(folder_location)

        for folder_files in folder_files_list:
            if ".py" not in folder_files:
                try:
                    os.remove(os.path.join(folder_location, folder_files))
                except:
                    pass

    def calibrate_output(self):
        """
            This function calibrates the output for given calibrations.
        """
        # Find the correct input file
        # Create a output file for the calibrated values
        # Write the header row
        # Go through each input row and callibrate the value of the row
        # Write the callibrated row to the output file

        input_file = (open(os.path.join(self._reshaped_location,\
            self._reshaped_filename), "r"))
        reader = csv.reader(input_file)

        output_filename = str(self._reshaped_filename).replace(".csv",\
            "_calibrated.csv")
        output_file = (open(os.path.join(self._calibrated_location,\
            output_filename), "wb"))
        writer = csv.writer(output_file)

        new_row = reader.next()
        writer.writerow(new_row)

        for row in reader:
            value = self.__calibrate_value(row[1], row[2])
            new_row = [row[0], row[1], round(value,5), notes[row[1]]]
            writer.writerow(new_row)

        input_file.close()
        output_file.close()
```

APPENDIX C

BACKGROUND IMAGE FOR WIND ROSE GRAPH

Appendix C: Background Image for Wind Rose Graph



APPENDIX D

ADDITIONAL RESULTS FOR WIND DATA

Appendix D: Additional Weather Station Results

Trial 4 was run on October 30, 2014 from 6:51PM to 7:58 PM to capture data for four scenarios. Weather station data was filtered to the range of 30-60 degrees and distribution can be seen in Figure D.1 (5-Hz data resolution had been averaged to one second). Wind direction and speed can be seen in the wind rose in Figure D.2. And the trend for wind speed is shown in Figure D.3.

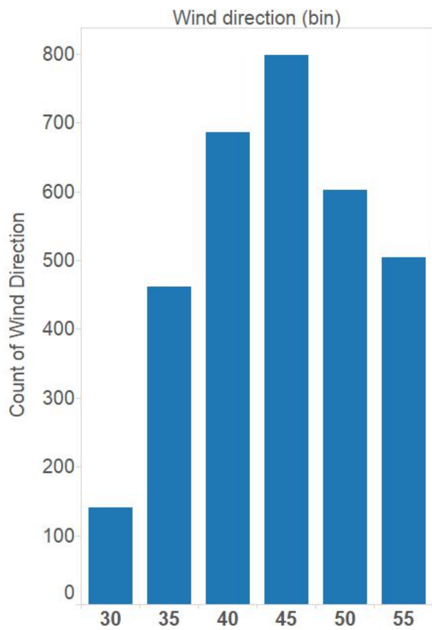


Figure D.1 Distribution of wind direction after filtering to 30-60 degrees

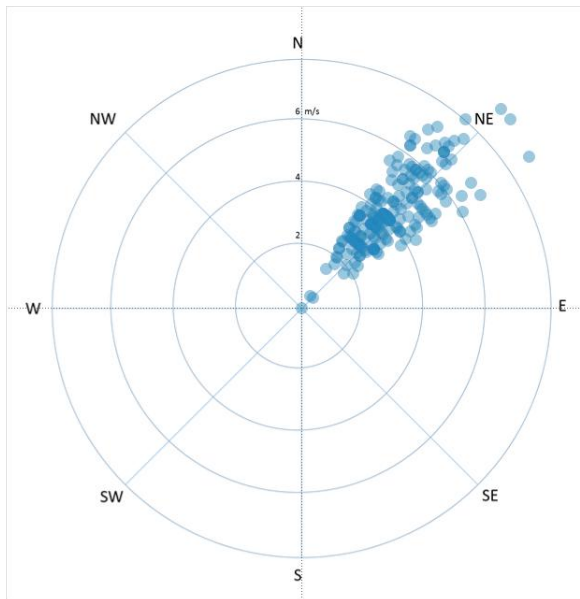


Figure D.2: Wind rose showing wind direction and wind speed (after filtering all data for wind directions 30-60 degrees) for trial run on October 30, 2014

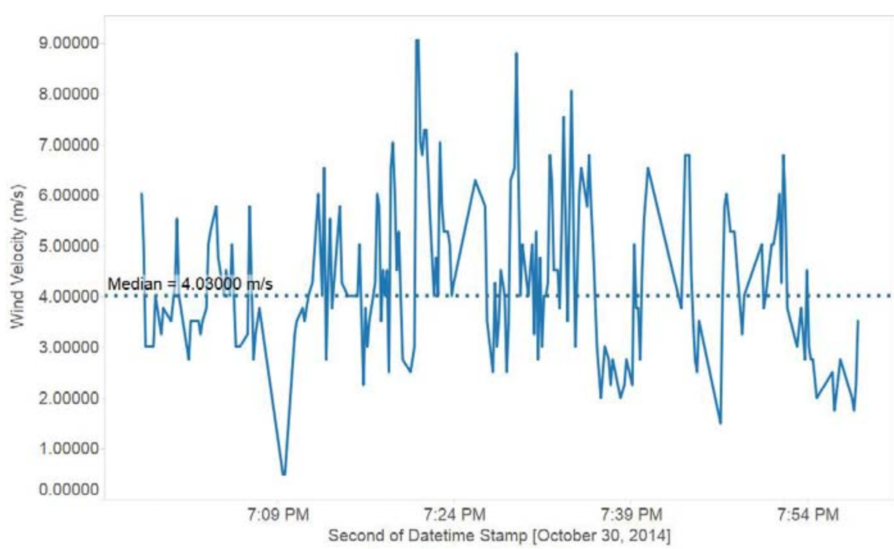


Figure D.1: Weather station wind speeds (after filtering all data for wind directions 30-60 degrees) on October 30, 2014

APPENDIX E

ADDITIONAL RESULTS FOR AIR VELOCITY DATA

Appendix E. Additional Air Velocity Data

Air velocity (m/s) trending graphs for all anemometers during all four scenarios on October 30, 2014 can be seen in Figures E.1 through E.9.

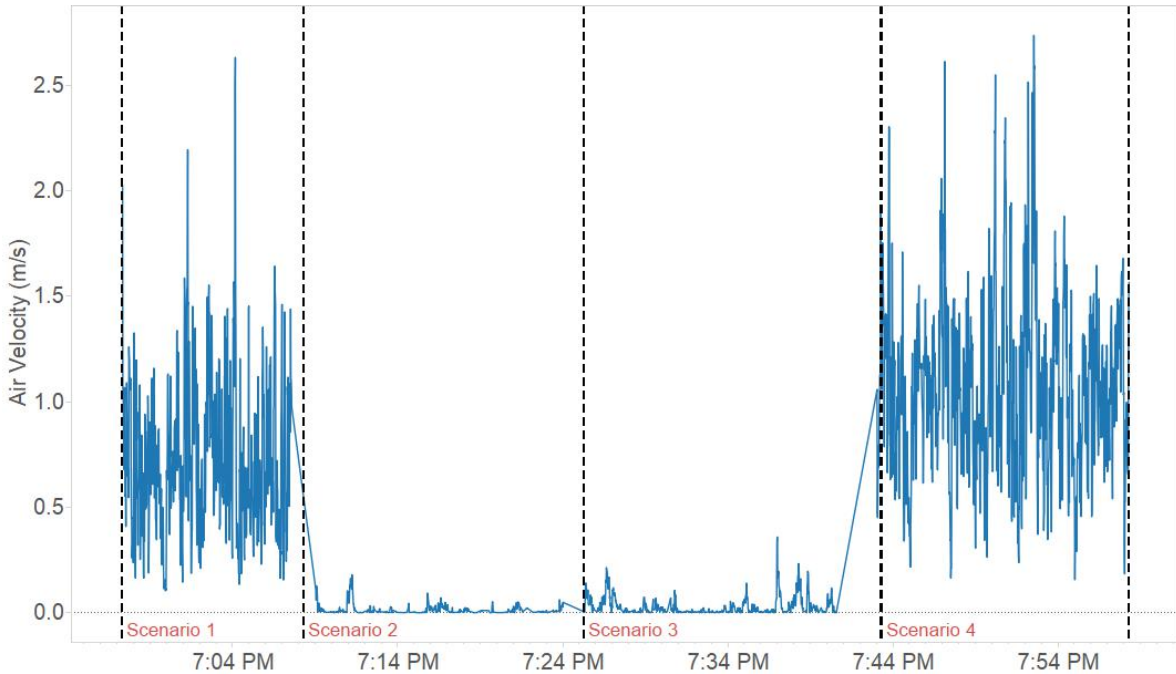


Figure E.1: Air velocity (m/s) trend for anemometer ai00

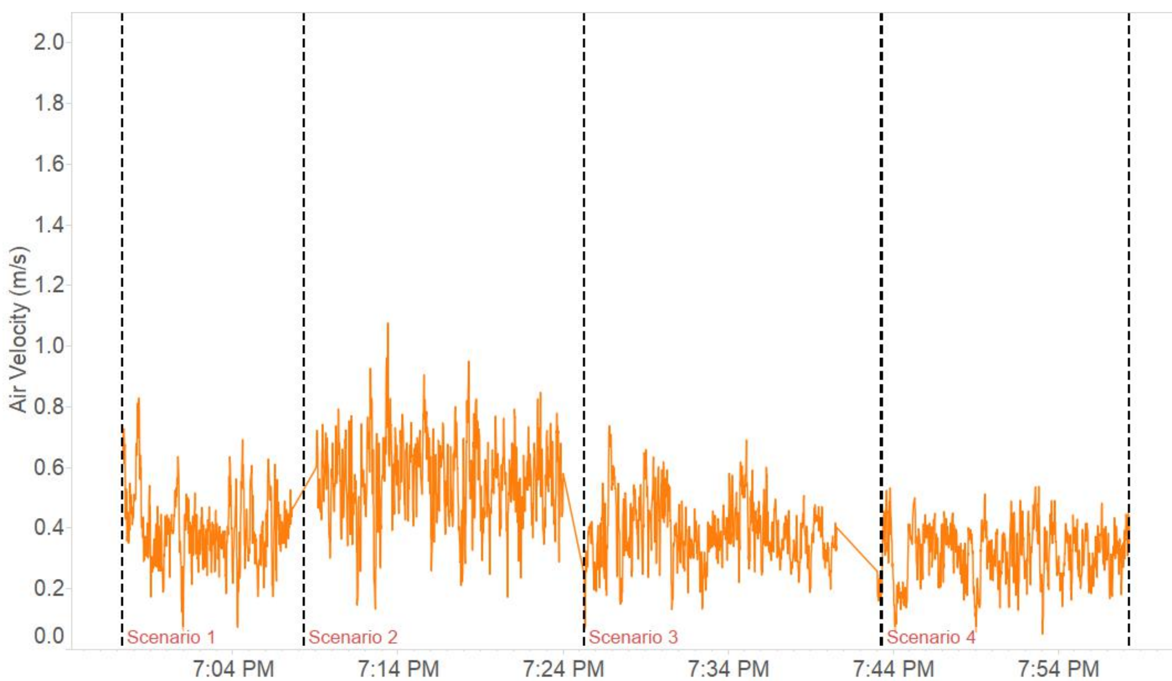


Figure E.2: Air velocity (m/s) trend for anemometer ai01

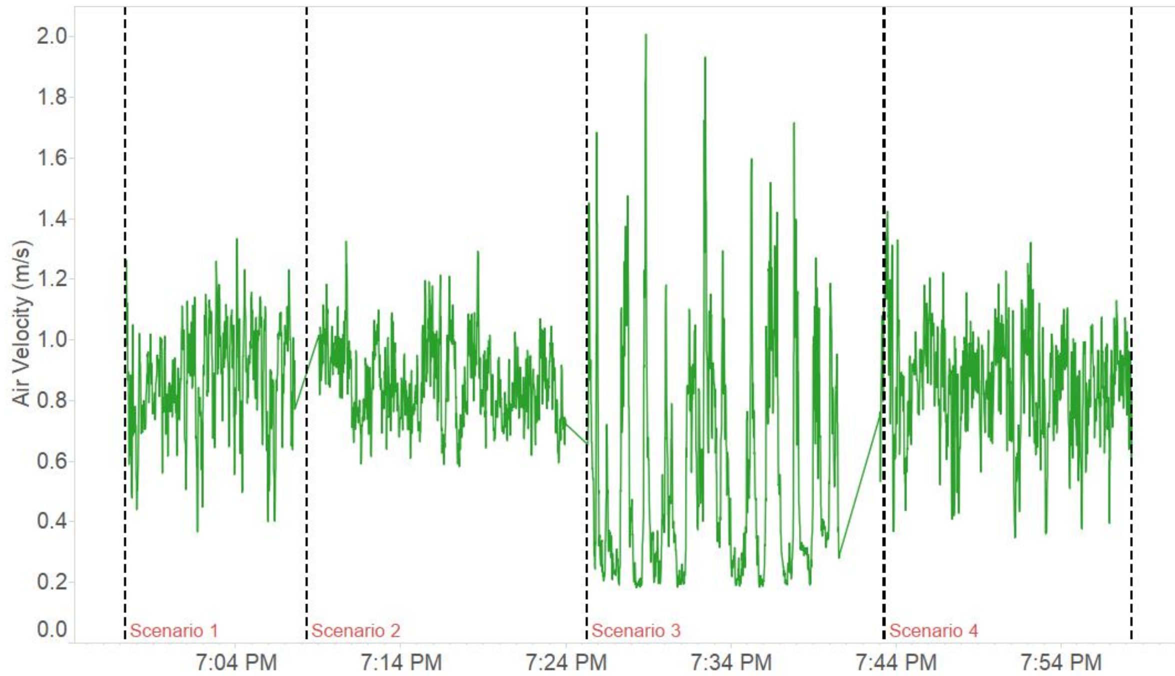


Figure E.3: Air velocity (m/s) trend for anemometer ai02

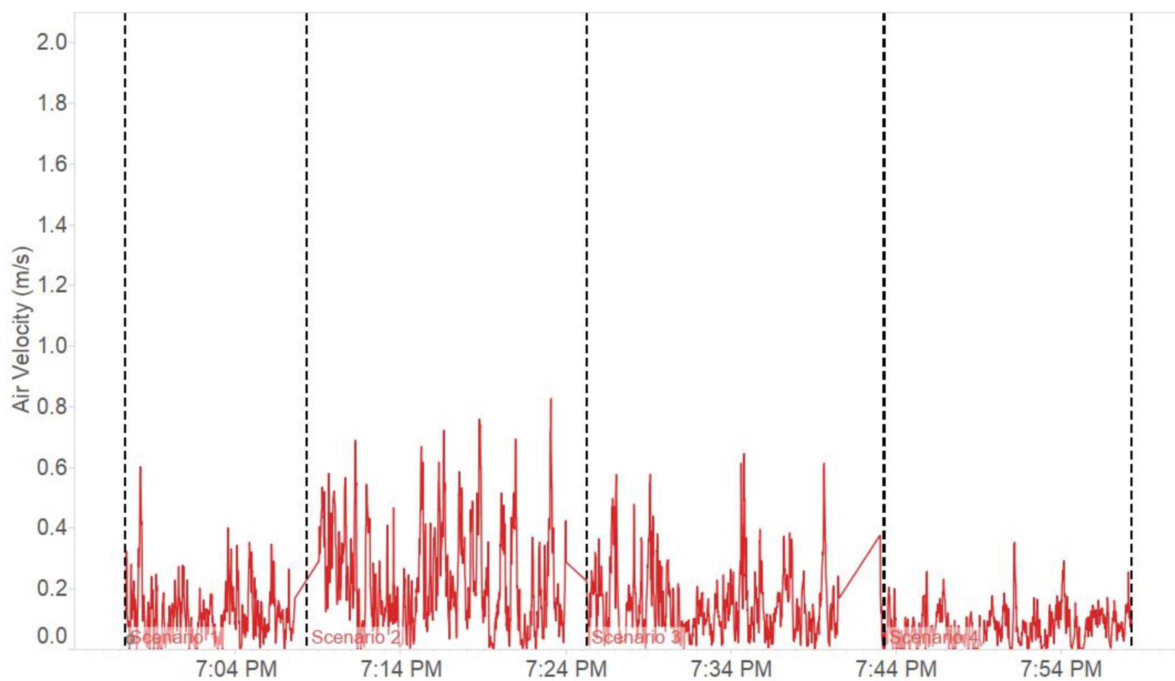


Figure E.4: Air velocity (m/s) trend for anemometer ai03

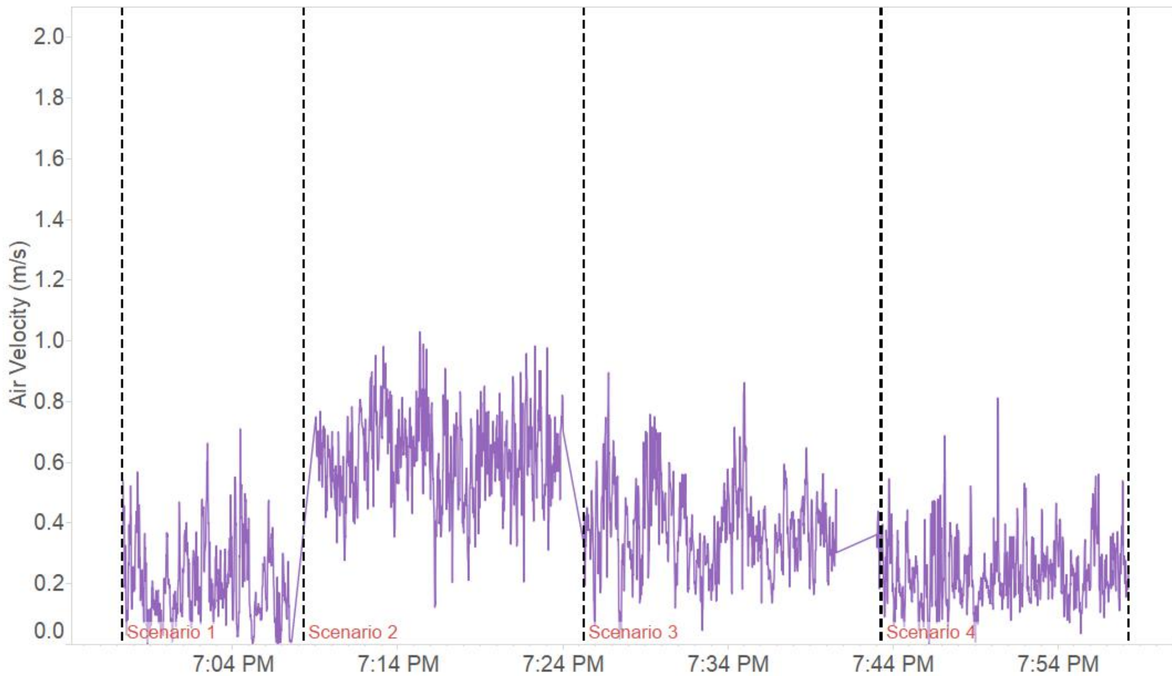


Figure E.5: Air velocity (m/s) trend for anemometer ai04

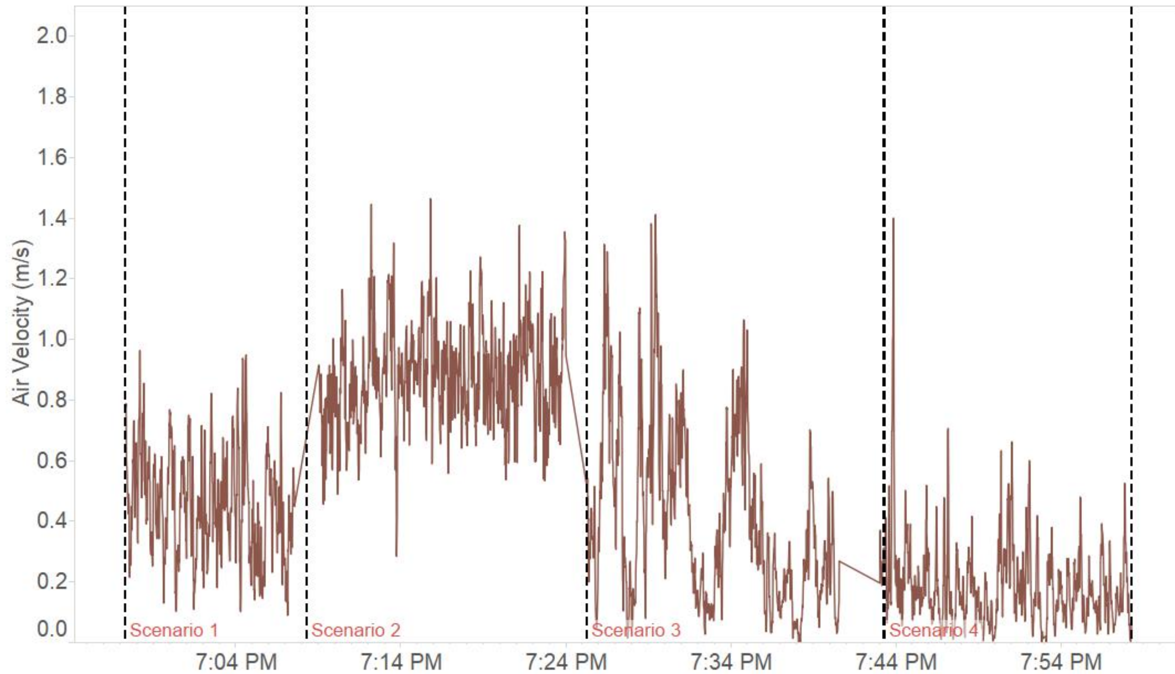


Figure E.6: Air velocity (m/s) trend for anemometer ai05

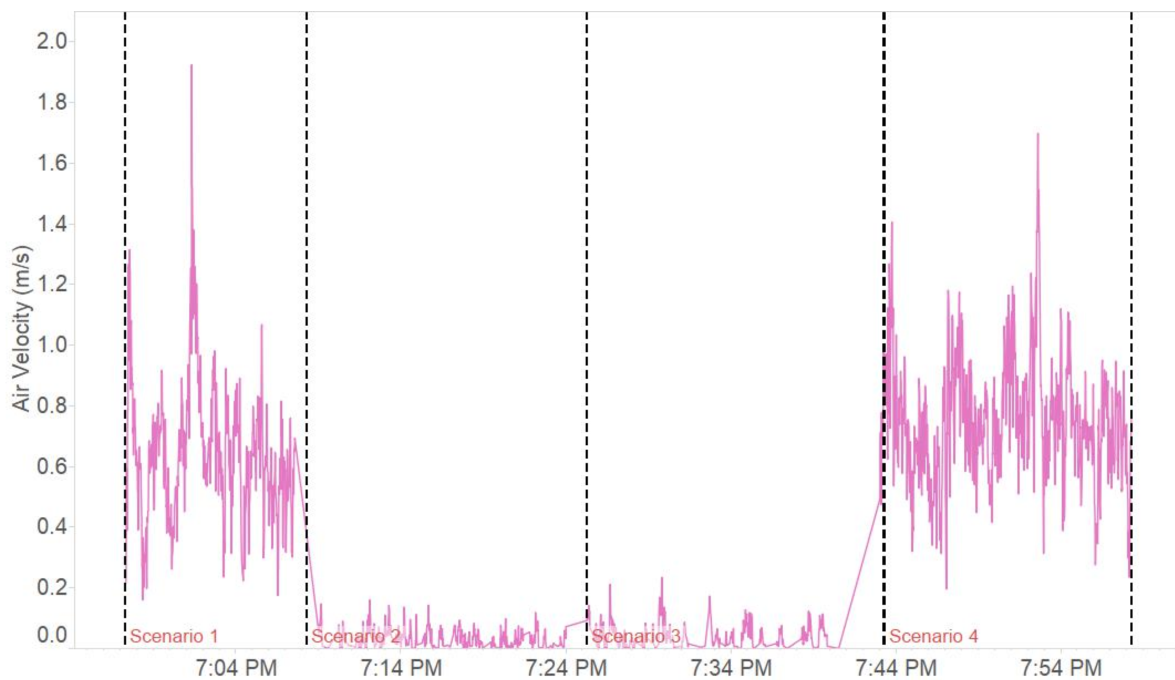


Figure E.7: Air velocity (m/s) trend for anemometer ai06

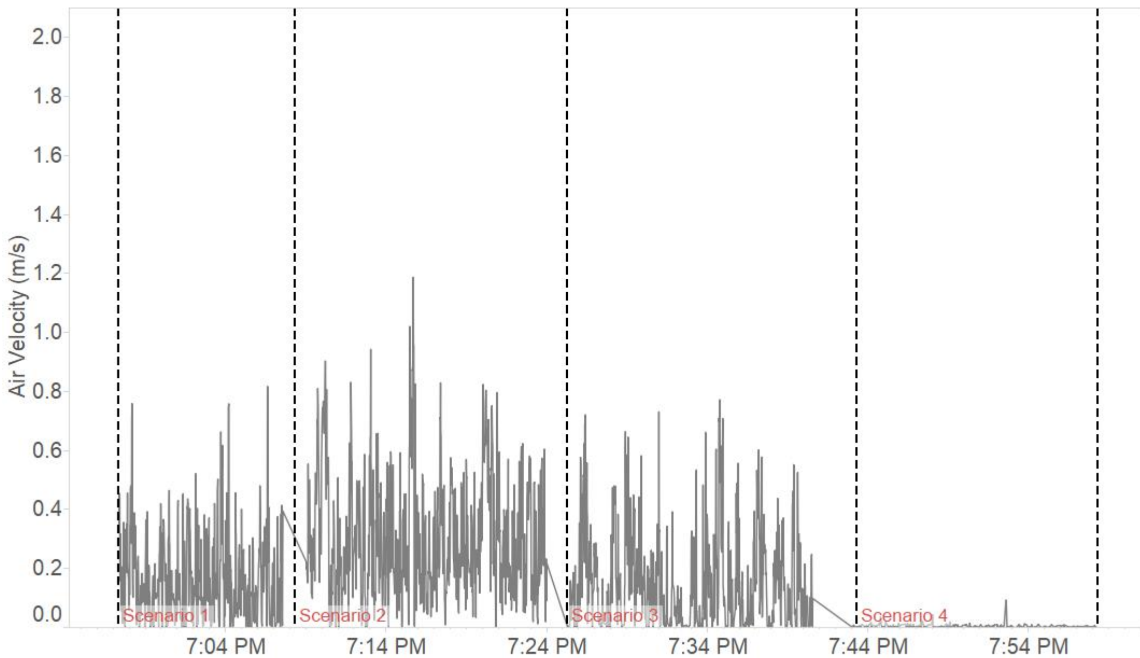


Figure E.8: Air velocity (m/s) trend for anemometer ai07

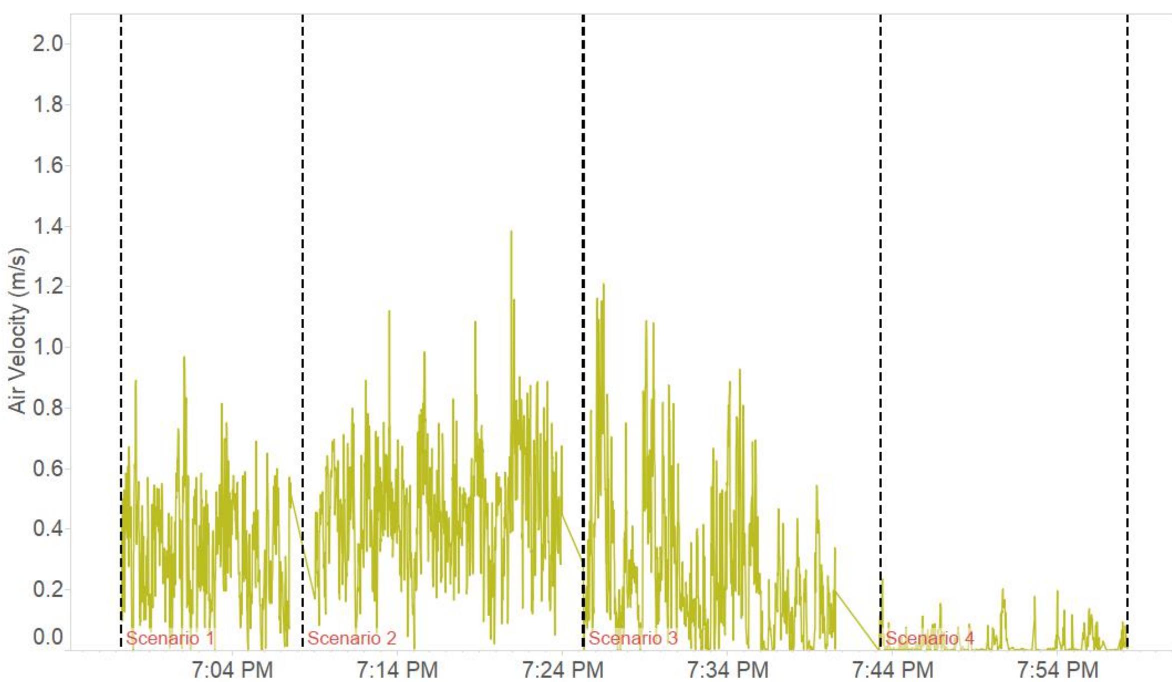


Figure E.9: Air velocity (m/s) trend for anemometer ai08

APPENDIX F

ADDITIONAL RESULTS FOR PRESSURE DIFFERENTIAL DATA

Appendix F. Additional Pressure Differential Data.

Pressure differential (inches of water column) trending graphs for all pressure sensors during all four scenarios on October 30, 2014 can be seen in Figures F.1 through F.4. Sensor ai11 (the Halstrup Walcher P26) is the only one equipped for bi-directional measurement, although its absolute range is no greater than the other sensors. Negative values were filtered out for the other sensors (Setra 264s).

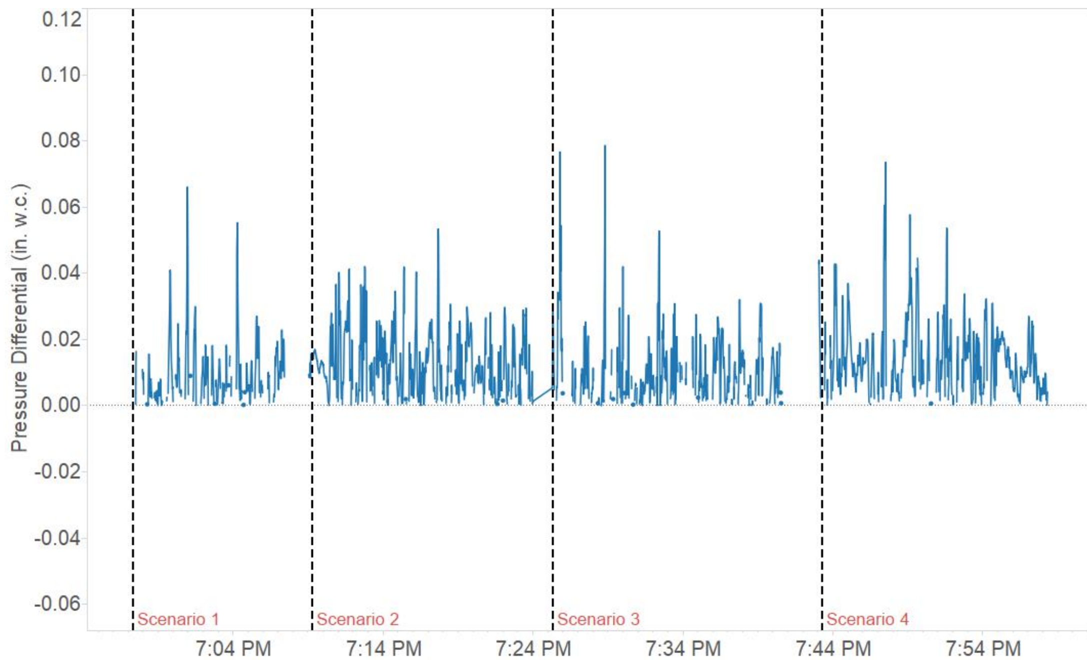


Figure F.1: Pressure sensor ai09

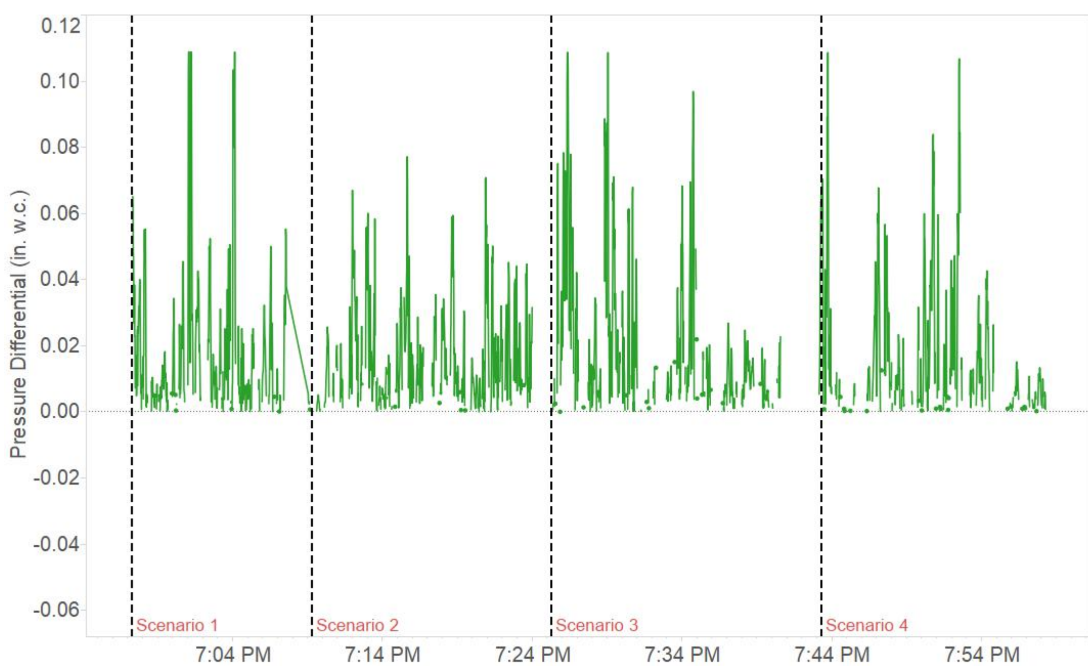


Figure F.2: Pressure sensor ai10

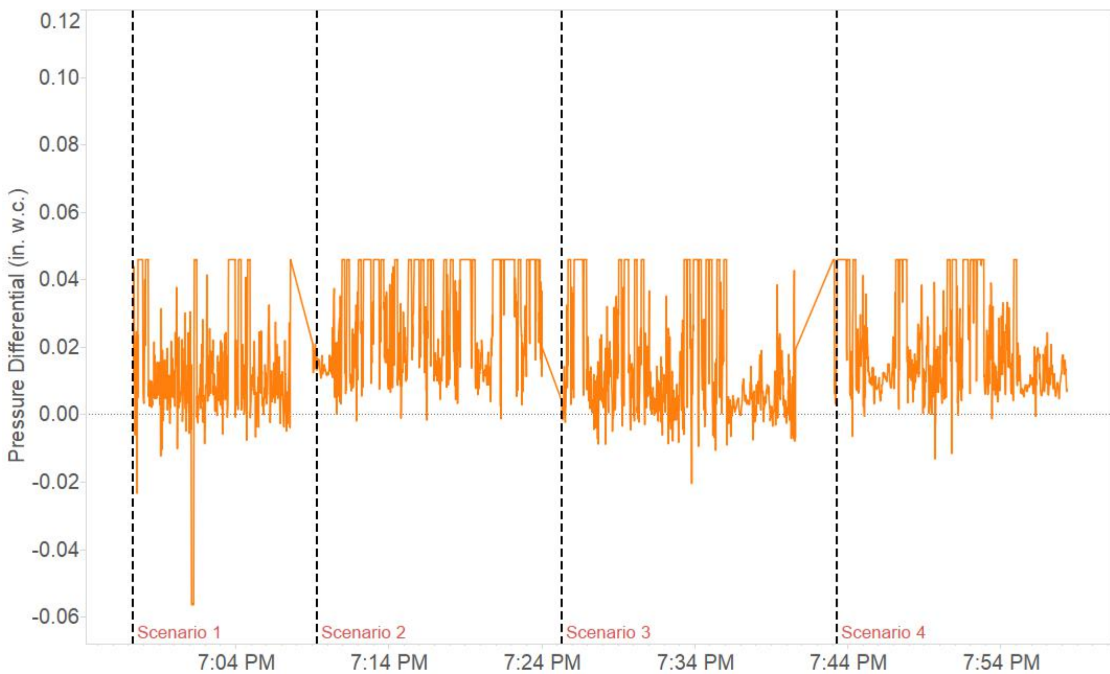


Figure F.3: Pressure sensor ai11

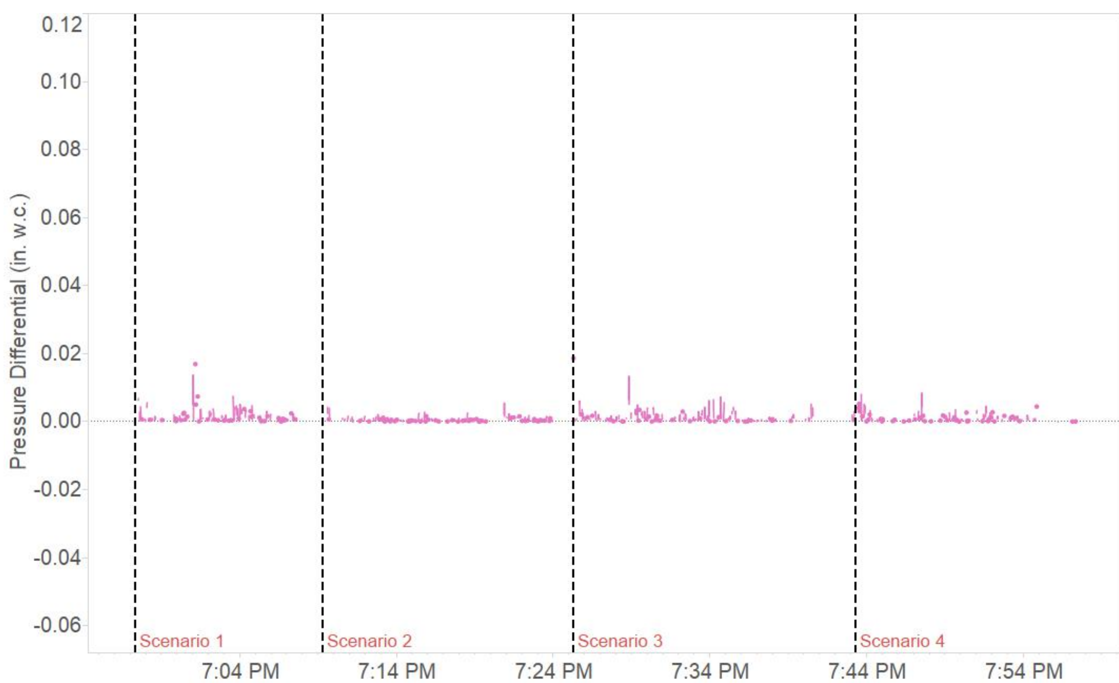


Figure F.4: Pressure sensor ai12