Modeling and Analyzing Wind Velocity at Entrance Doors to Avoid Accidents

Abu Asaduzzaman, Luke Mercer, Md Raihan Uddin
Department of Electrical and Computer Engineering
Wichita State University
Wichita, Kansas, USA
abu.asaduzzaman@wichita.edu

Yoel Woldeyes
School of Computing
Wichita State University
Wichita, Kansas, USA
yswoldeyes@shockers.wichita.edu

Abstract—**There are safety threats due to unexpected uncontrolled sudden opens and shuts of entrance doors. This work aims to develop a computer-simulated wind velocity model to study the doors' risky behavior by analyzing the relationship between wind velocity and the corresponding door movements. We develop a microcontroller-based system to detect when a door is opened, and record the wind velocity and door open distance when the door is opened and closed. This process is completed using an anemometer to measure the wind velocity, a magnetic door switch to detect when the door opens, using an ultrasonic sensor to measure the door distance, and calculating the time the door was open using the Arduino timer. The experiments are conducted inside a room, where wind speed and maximum door open distance can be controlled. The preliminary results show that the door open speed and distance increases significantly with increased wind speed. The proposed model can be extended as a potential remedy to dangerous threats for buildings and building occupants.**

Keywords—modeling and analyzing, wind velocity, microcontrollers, air-door

I. Introduction

Predicting wind velocity involves forecasting the speed and direction of airflow in a specific region. Approaches to modeling this phenomenon include mathematical, physical, and computational models. Mathematical models recreate wind's physical features, whereas physical models deploy replicas that simulate wind flow through an environment. Computational methods generate computer simulations that project the behavior of air movement within a designated zone. Practical prediction measures employ multiple computing techniques, such as prediction error [1]. This technique generates randomly produced white noise with no set means or unit variance while constructing power system applications related to modeling wind velocity.

The modeling of wind velocity has led to the development of a vital technology that enables the detection of open doors and records both wind velocity and door open distance. Specifically, this technology finds application in creating systems for detecting instances where doors are opened due to wind power. By accurately measuring the range of windspeeds capable of opening doors, buildings can be fortified against any possible damage from such occurrences [2].

The system comprises distinct components like anemometers that capture wind velocity data alongside magnetic door switches responsible for detecting door openings. Additionally, ultrasonic sensors measure open distances while timers capture every instance when a door opens [3]. An Arduino board then analyzes all accumulated data appropriately while storing it effectively for future reference. Mainly designed to collect data on the actual weather conditions or scenarios leading to open doors due to unlikely challenges like high-speed winds, this refined technology determines if an uncontrollable limit makes the door susceptible under given environmental conditions requiring processing and action underway fast.

In this work, we set up a microcontroller with components to measure the wind velocity, door opening and closing, and the door open distance. We use an anemometer to measure wind velocity, a magnetic door switch to determine when the door is opened (and closed), and an ultrasonic sensor to calculate the distance between the sensor and the door. The data collected are analyzed to develop a clear and concise graphic. The system is relatively portable with a simple setup, easy data collection process, and a system to graph this data.

II. Related Work

Timely identification of the door opening and closing in a building is critical for fire prevention and control [4-6]. An air-door opening and closing identification algorithm is proposed by Shang et al. using a single wind-velocity sensor [7]. The air-door is a device that is used for adjusting the volume and direction of air flow in a mine ventilation system. Studies show that the wind-velocity data in a mine are not smooth lines under the action of turbulent pulsation. As shown in Figure 1, the abnormal fluctuation time of the wind-velocity data appears outside the air-door opening and closing time. It is observed that there is no obvious correlation between the multiple abnormal fluctuations in the wind-velocity data and the air-door opening and closing time monitored by the switching sensor. The wind-velocity data can be used to avoid safety accidents and production activities caused by ignoring the abnormal fluctuation of the wind speed.

Part of this work is funded by the Mentorship 360 program.

In this work, a computer-based wind velocity model is developed to explore the relationship between wind velocity and the corresponding door movements.

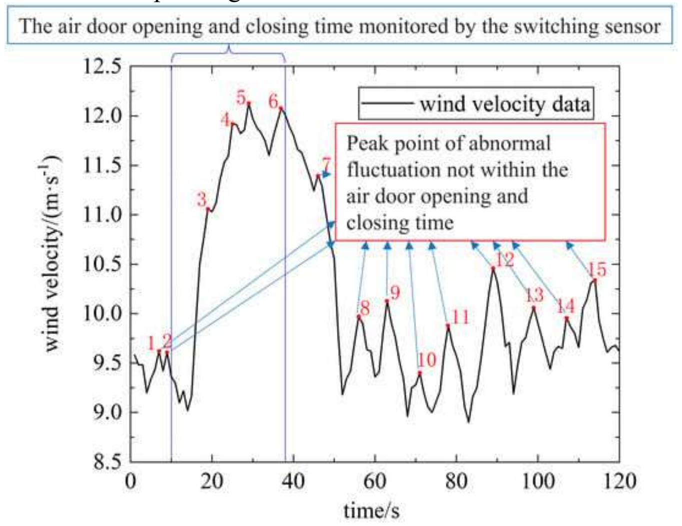


Figure 1. Wind velocity from an air-door in a mine [7]

III. Simulated System

In this section, we describe the system developed to conduct the experiments.

A. Hardware Used

Important hardware components used are:

- Microcontroller: Arduino Uno [8]
- Wind velocity sensor: Anemometer [9]
- Door switch: Magnetic switch [10]
- Distance sensor: Ultrasound distance sensor [11]

Figure 2 illustrates how the components are connected. The Arduino Uno is chosen as it provides enough input/output pins, supports for connection and communication with a computer, and good timing processing. The Uno does not have on-board storage, so it must be connected to a computer for the entire time of operation. The Uno is programmed in C++ with a plethora of supported libraries and additional functions.

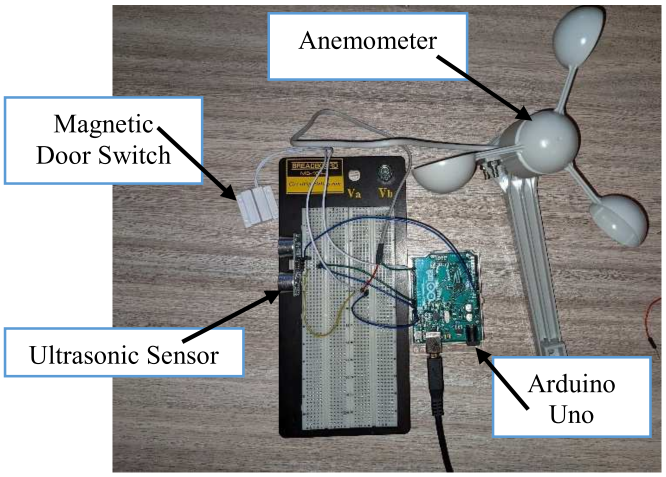


Figure 2. System component connection

The Uno has a setup function that is called when a program is first started to run code that is only needed to run once. This is where input/output (I/O) pins are setup using pinMode(#, Mode) with # representing what pin is being setup and Mode being how the pin is setup. The pins (see Figure 3) used in this project are pin 2, 4, 12, and 13. An Arduino Uno is very versatile so the specific pin for each usage does not matter if it is an I/O pin. The pin modes used in this project are INPUT_PULLUP, which uses code to access 20k Ohm pullup resistors that are connected to +5V. This brings the assigned pin to a default HIGH reading when digitalRead() is used, and when the pin is connected to ground it will read LOW when digitalRead() is called.

To start with usage for the anemometer and door switch, one of the wires coming from the device (device being anemometer or door switch) is connected to a digital pin; in this case, the anemometer is connected to Pin 2 and the door switch is connected to Pin 4. The other wire from the device is connected to a ground on the Arduino Uno. In the Arduino code using pinMode() Pins 2 and 4 are set to INPUT_PULLUP. Then command digitalRead() is used to determine the voltage on the pin. If the voltage is 5V, or the pin is not connected to ground then digitalRead() returns HIGH, if the pin is connected to ground via the device then digitalRead() will return LOW. This is used to determine when the switches in the device(s) are closed and grounding the pin.

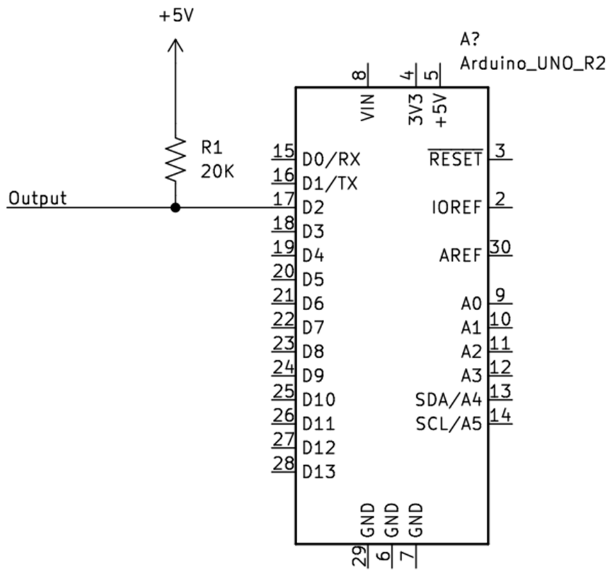


Figure 3. Arduino pin-mode pullup diagram

The anemometer, i.e., wind velocity measurement sensor, is connected to Pin 2 and ground of the Arduino. The device is bi-directional, so it does not matter which wire is connected to which pin. When the anemometer is rotating due to wind, the switch closes and Pin 2 goes to low, which triggers an interrupt as attachInterrupt() was set on Pin 2 to go off when the pin goes from HIGH to LOW. The interrupt adds 1 to a counter that was set to 0, so that every time the anemometer makes a rotation count increments by one. After 1 second the count is divided by

1.492 to get the wind velocity in miles per hour. After every second the wind speed is put into the serial monitor window after one comma, and the count variable is reset.

The magnetic door switch is on Pin 4. The door switch is bi-directional like the anemometer, so one pin just needs connected to Pin 4 and the other connected to ground. When the two parts of the door switch are close together, roughly 0.5 inches, the switch is closed and Pin 4 is connected to ground and the pin reads LOW in the code. The door switch piece with the wires is setup on the door frame, and the loose piece is setup on the edge of the door such that when the door is closed they are within 0.5 inches. The code uses digitalRead() to measure if the door is open or closed, and when it is opened a Boolean is set indicating such so that the door opening distance will start recording, and "opened" is put in the serial monitor window. When the door is closed, this Boolean is flipped and "closed" is put in the serial monitor window.

Finally, the door opening distance measurement system uses the ultrasonic sensor HC-SR04. This works slightly differently from the other sensors. This uses Pin 12 on the Arduino as the output connected to the Trig on the ultrasonic sensor, Pin 13 connected to the Echo pin on the ultrasonic sensor, then +5V to Pin VCC and ground to Pin GND. When the code indicates the door is opened, the ultrasonic sensor puts the trig pin to the high for 10 microseconds, then puts it to low. Then it uses pulseIn on the Echo pin to measure how long until the 10-microsecond signal takes to return to the sensor. It takes that time value and converts it to centimeters by dividing by 29 then dividing by 2 per the device specs. This value gets posted to the serial monitor after two commas. We set this up so that when the door is closed it goes past the door, but when the door opens it swings into the ultrasonic sensor line of sight.

The data is all read into the Serial Monitor window using Serial.print(). This is the serial communication between the Arduino and the computer it is connected to. The formatting of the data is crucial to how the graphing system works. The Python script is discussed more later, but for now the important part is that the data is put into an Excel using comma delimiting format, so commas are used to space the data into columns based on what data it is. The opening/closing markers are placed without a comma before, the time data is placed without a comma before as well, the wind speed data is placed after one comma, and the door distance data is placed after two commas. A sample of the serial monitor window is shown below.

Opened	← This flag shows the door is opened.
6886	← Time in milliseconds since start
,,26	← Door distance measurement
6999	
,,24	
,14	← Wind velocity
…	
Closed	← This flag shows the door is closed.

The Opened and Closed texts have no comma before, the distance data shown has two commas before it, the time in milliseconds since start of program has no comma before, and the wind velocity shown has one comma before.

Figure 4 illustrates the connections among the Arduino, anemometer, magnetic door switch, and ultrasonic sensor.

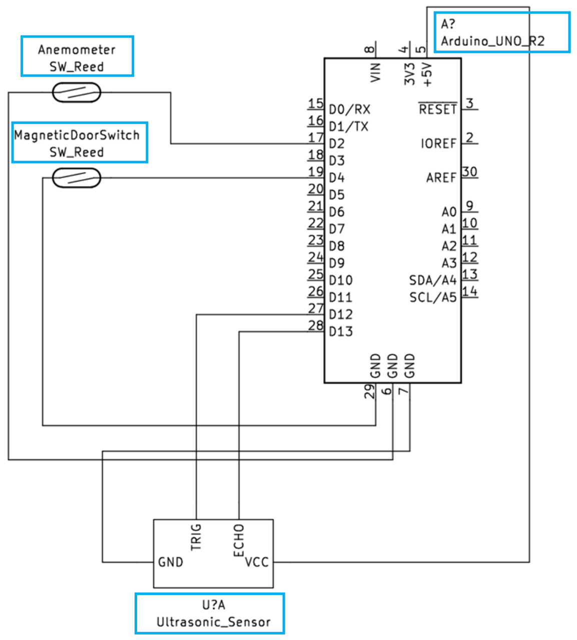


Figure 4. Component connection diagram

B. Data Collection

Wind speed, door open distance, and time are needed to calculate the door opening speed. Figure 5 shows the process how the Arduino goes through during data collection.

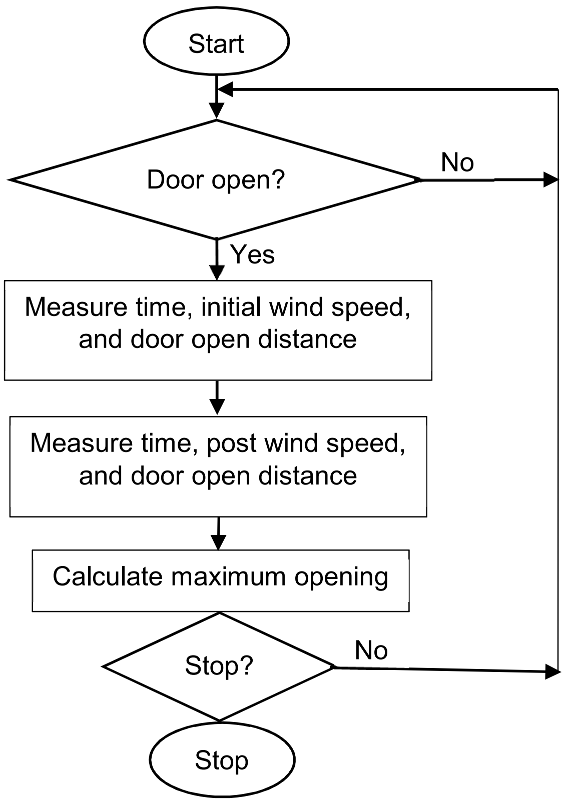

Start
Door open?
No
Yes
Measure time, initial wind speed, and door open distance
Measure time, post wind speed, and door open distance
Calculate maximum opening
Stop?
No
Stop

Figure 5. Data collection flowchart

C. Ploting Graphs

A Python script (say, WindVelocity.py) file is used to plot graphs. To start plotting graphs, the texts in the Serial Monitor window are copied and pasted into Notepad to save as a .txt file. Then the data is imported to Excel and saved as .xlsx in the same directory with the WindVelProject.py file. The Python script uses multiple libraries such as openpyxl, matplotlib, and numpy. Openpyxl is a python library that is used to read and write from excel workbooks or .xlsx files. In the script the workbook is initialized to wb_obj and the sheet is initialized to sheet_obj. With the workbook and sheet initialized, an individual cell can be accessed with the code shown. Cell_obj = sheet_obj.cell(row = #, column = #). This stores the value that is in the sheet into the variable. The value in the cell can then be accessed with cell_obj.value. The Python script initializes three variables to append with data from the workbook called 'dist' to store the ultrasonic sensor distance measurements, 'vel' to store the anemometer measurements, and 'time1' to record the time on each ultrasonic sensor measurement. Next a while loop goes through the first column until the cell is not empty, which means that the door is opened as signaled by "opened" in the text. Then it loops through the column from that point until the cell contains "closed." The data is stored in a row where opened and closed are found, and the rows between them with the first three columns are iterated through with sheet_obj.iter_cols, and the data in the first column going into time1, the data in the second column going into vel, and the data in the third column going into dist.

Next, the data in the dist and time1 have a lot of empty cells from the xlsx format. These empty cells are removed to simplify the graphing later in the process.

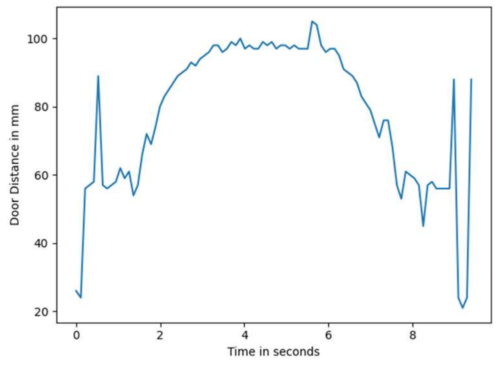
The next issue with the data is with the time measurements. This time value is found using the millis() command on the Arduino. This command returns the time in milliseconds since the microprocessor started the program. It is noticed that the time value does not start at 0 seconds from the door opening. The time value is obtained in milliseconds. To calculate the elapsed time, the minimum value of the list is stored into a variable to iterate through the time1 list and subtract each value by that variable to move the time range from 0 until the door is closed. Then each value in the list is divided by a thousand to convert the time to seconds for clarity. To properly display the velocity data as the speed is recorded each second, len(list) is used to find the length of the vel list, then using linspace from the numpy library to generate a list from 0 to the number of velocity data points spaced by 1 second. Matplotlib is the library used to graph the data. Usage in code of matplotlib requires generating a figure using the code shown below.

```
fig(ax1, ax2) = plt.subplots(2)
fig.suptitle('Wind speed and Velocity')
ax1.plot(time1, dist)
ax2.plot(timvel, vel2)
ax1.set_title('Door Distance in mm')
ax2.set_title('Wind Velocity in MPH')
plt.show()
```

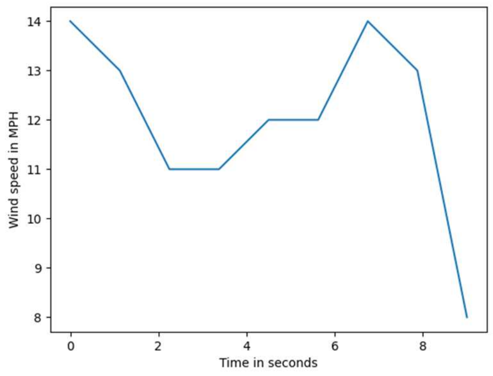
Here, fig(ax1, ax2) is the entire figure object with two subplots and their names; plt.subplots(2) is the command to generate the two plots. Then into ax1.plot and ax2.plot, the x axis values are put in first and the y values in the second part. Then ax1.set_title and ax2.set_title is used to title the individual plots. Finally, plt.show() displays the figure with the two subplots. A graph is shown in Figure 6.

IV. Experimental Results

The experiments are conducted inside a room, where the wind and door distance can be controlled. Figure 6 shows an assumed condition for normal operation of the door with typical open and closing times. As shown in the figure, the door distance in (a) has the assumed slow increase, plateau in distance, then the decrease in distance. An anomaly found is that at the beginning and end of the graph, there is the spike up and spike down respectively. The first spike up after 0 seconds could be from the delay in the door getting into the line of sight of the ultrasonic sensor. The second spike down after 6 seconds could be from the door going passed the line of sight of the ultrasonic sensor and coming very close, thus reading close to 20 mm distance, then spiking up as the door goes past the sensor and the sensor is reading other data. As shown in Figure 6 (b), the wind speed is high when opening the door. The wind speed goes lower as the door closes as expected.



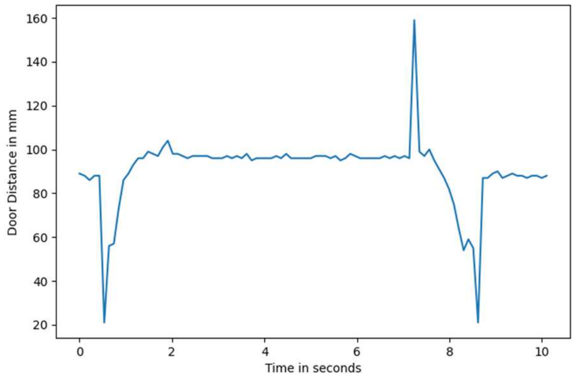
(a) Door distance Vs Time



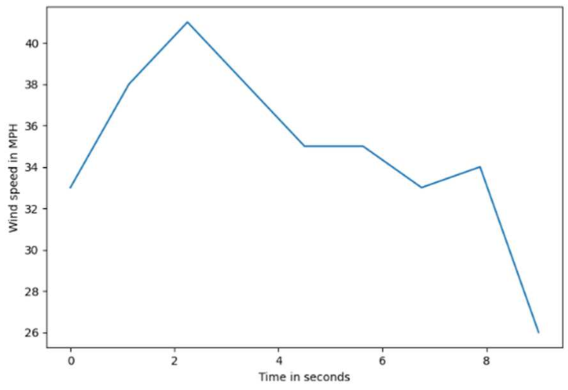
(b) Wind speed Vs Time

Figure 6. Door distance and wind velocity under simulated normal operating conditions

Next, we conduct the experiments under increased wind speed. As shown in Figure 7, the door open distance increases significantly faster (in less two seconds, instead of more than four seconds); then stays at that distance for a while before quickly closing. The wind velocity is shown higher (more than 40 miles per hour, instead less than 15 miles per hour) when opening the door. As expected, the wind speed goes lower as the door closes.



(a) Door distance Vs Time



(b) Wind speed Vs Time

Figure 7. Door distance and wind velocity under simulated high wind environment

The results in a controlled environment show that the system works as intended to collect wind speed velocity and door opening distance over the time of the door is being opened. If the system (using microcontroller, wind velocity sensor, door switch, and distance sensor) is properly implemented, it is expected that it will help avoid accidents due to unexpected uncontrolled sudden opens and shuts of entrance doors.

V. Conclusion

This work studies a wind velocity model on doors' risky behavior under high gusty wind speed. The experiments are conducted using an Arduino Uno to collect and format the data, an anemometer to record the wind velocity data, a magnetic door switch to record when the door opens and closes, and a ultrasonic sensor to measure the door opening distance. Python is used with the matplotlib, numpy, and openpyxl libraries to plot graphs. According to the experimental results from a controlled environment, when the wind speed is increased from 15 miles per hour to 40 miles per hour, the door opens faster (in less two seconds, instead of more than four seconds).

Acknowledgment

Part of this work is funded by "Faculty Connection and Mentorship Through an Integrated Entrepreneurial Mindset Curriculum" project from the Mentorship 360 program.

References

[1] N. Abdel-Karim, M. Ilic, and M. Small, "Modeling Wind Speed for Power System Applications," in Wind Farm - Impact in Power System and Alternatives to Improve the Integration. InTech, Jul. 28, 2011. doi: 10.5772/17870.

[2] M. Ghodrat, F. Shakeriaski, D. J. Nelson, and A. Simeoni, "Existing Improvements in Simulation of Fire–Wind Interaction and Its Effects on Structures," in Fire, vol. 4, no. 2, p. 27, May 2021, doi: 10.3390/fire4020027.

[3] A. Loganathan, S. Albert, and Y. Uma, "Performance Enhancement of Wind Energy Conversion System at Low WindSpeeds," in International Conference on Combinatorial and Optimization, 2021.

[4] A. Abdulkareem, B.O. Adeyemi, T.E. Somefun, Tobi and V. Oguntosin, "Design and Construction of a Weather-Based Automatic Sliding Window," in Institute of Physics (IOP) Conference Series: Materials Science and Engineering, 2021.

[5] Panindre, P., Mousavi, N.S.S., and Kumar, S. "Improvement of Positive Pressure Ventilation by optimizing stairwell door opening area," in Fire Saf. J., 92(1), 195–198, 2017.

[6] Lee, J. "Numerical analysis on the rapid fire suppression using a water mist nozzle in a fire compartment with a door opening," in Nucl. Eng. Technol., 51(1), 410–423, 2019.

[7] Shang, W., Deng, L., and Liu, J. "A Novel Air-Door Opening and Closing Identification Algorithm Using a Single Wind-Velocity Sensor," in MDPI Sensors, 22(18), 1-21, 2022.

[8] Arduino controller (with ATmega328P). A000073/1050-1041-ND, 2023, https://www.digikey.com/en/products/detail/arduino/A000073/3476357

[9] Anemometer sensor (to measure the wind speed and wind direction). SEN-15901 ROHS, 2023, https://www.sparkfun.com/products/15901

[10] Magnetic door sensor. MC 38 NC, https://www.amazon.com/Window-Magnetic-Recessed-Security-Normally/dp/B074271Y6S?th=1

[11] Distance sensor: Ultrasonic distance sensor HC-SR04. SEN-15569, https://www.sparkfun.com/products/15569