MECH5051/5052 Senior Design

Low-Cost Sensor to Measure Evaporation

Group #10

Jake Robida, Jiabin Hu, Xiangshu Wang

University of Cincinnati

Senior Design

Bellur, Kishan

April 21, 2023

Table of Contents

Executive Summary

This project proposed an economical solution to monitor evaporation from large bodies of water based on the Eddy-Covariance method. An Arduino-based system is established as the prototype to verify the feasibility of this proposal. The system consists of three main components the microcontroller, the wind speed sensor, and the humidity sensor. All materials of the system are off-the-shelf electronic components, and the total cost was under $50. The project also provides advice for future improvements and alternatives to the current design.

1. Project Description

1.1 Purpose of project

Monitoring the transfer of heat, water, and carbon from terrestrial and aquatic environments into the atmosphere is the most important step towards understanding global and regional climatological and hydrological processes that impact human and environmental health. Evaporation from oceans and large lakes, for example, is not only the major source that forms terrestrial precipitation but is also critical in regulating heat content of these water bodies. Initiatives taken by multiple agencies and academic institutions have spurred international collaborations to study lake evaporation using eddy covariance (EC) stations. However, there are still enormous gaps in our understanding. These gaps have remained, in part, because of the cost and maintenance requirements of full EC stations. The capital cost for one station is typically in the range of $70,000, and deploying one costs several tens of thousands of dollars more. This proposal aims to fill that gap through the development of low-cost sensors that collect key variables for eddy-covariance algorithms. Many more of these new sensors could be produced with better temporal and spatial sensitivity, and at a fraction of the cost.

1.2 Specific objective of the project

The major output of the project is a proof-of-concept demonstration of EC sensors that are inexpensive ($\sim$ $100), fast ($\sim$ 100 Hz) and compact measurement space ($\sim$ 10 cm). The secondary outputs of the project include answering a wide variety of fundamental questions of importance and providing possible alternative methods to further improve the design.

1.3 Limits and scope of the project

Since this is a proof-of-concept design, the actual performance of the sensor remains unknown to us and requires more experiments to verify accuracy. Moreover, as the sensor will not be a fully completed product, but a prototype that verifies the design concept of making such low-cost EC sensors, the case design, additional functions such as remote control will not be included in this semester's output. The performance of the sensor may not be comparable with those mature sensors that can be found on the market. Improvements are required in the future.

2. Engineering Standards and Codes

2.1 American Society of Mechanical Engineers (AMSE) Y14.5 – 2009

AMSE Y14.5 – 2009 is a widely accepted and used dimensioning and tolerance standard when drafting engineering drawings. Dimensions and tolerances are easily visible to users. The proof-of-concept eddy covariance system required two 3D printed testing parts, an HC-SR04 sensor holder and a reflector shield. Using AMSE Y14.5 – 2009 dimensioning and

tolerance two drawings were created using Siemens NX. If the STL or PRT files were to be lost, future senior designing teams can easily remodel the parts. A hard copy of AMSE Y14.5 – 2009 is available in the College of Engineering and Applied Science (CEAS) library.

2.2 Open-Source System – Circuit.io

Arduino and Raspberry Pi are both open-source platforms. This means codes and projects for these systems are widely available and can be modified. Circuit.io allows users to drag and drop electronic components, such as a HC-SR04 or BME280 sensor, to either an Arduino or Raspberry Pi. Circuit.io will automatically wire the circuit and generate sample code for rapid prototyping. A step-by-step wiring instruction and bill of materials is also generated.

3. Constraints

3.1 Economic

A fully equipped eddy covariance system costs $50,000. Using a microcontroller or microprocessor like Arduino Uno or Raspberry Pi and low cost ultrasonic and humidity sensor like HC-SR04 and BME280 a system was designed for under $100.

3.2 Environmental

The casing for the low-cost eddy covariance system will be additively manufactured through 3D printing. Thermoplastics that are commonly used in 3D printing can be reused rather than tossed into a landfill. However, the electronics in the system need to be disposed of according to local laws and guidelines.

3.3 Manufacturing

Additive manufacturing will be used to manufacture a case for the system. 3D printing allows future senior design teams to model around the complex geometry of the sensors as well as keep the manufacturing cost low.

3.4 Sustainability

To manufacture the system's casing a recycled filament for 3D printing will be used.

3.5 Ethical

There are no ethical constraints to consider as this is a measurement system. However, future senior design teams must ensure to "do no harm".

3.6 Health and Safety

Electronics were used extensively in the system. Standard lab health and safety guidelines set forth by the University of Cincinnati (safety glasses, gloves, etc.) were used when appropriate.

3.7 Specifications

The system needs to incorporate off the shelf electronic sensors (Arduino Uno, Raspberry Pi, HC-SR04, BME280). System must have 100 Hz measurement and have a compact form factor less than 10 cm. The casing for the system must be water resistant and float.

4. Project Management

Figure 1, Figure 2, and Figure 3 shows the Gantt chart which depicts the progress of our program. The schedule of the project has been adjusted several times to keep in correspondence with the actual challenges during practice, which is marked in yellow on the schedule Gantt chart, in contrast to the original schedule marked in green.

There are several reasons explaining our adjustment on actual schedule:

- Several significant schedules were dismissed from the schedule at the beginning, such as midterm review, spring break, and CEAS EXPO.
- Since it is completely proposed based on scratch (proof of concept), there were wrong estimates of the project itself, and the difficulty of some parts is underestimated, such as the accuracy of ultrasonic sensor, extracting raw data from ultrasonic sensors, and searching suitable IR sensor.
- The project development time was relatively short, and the initial workload was relatively small.

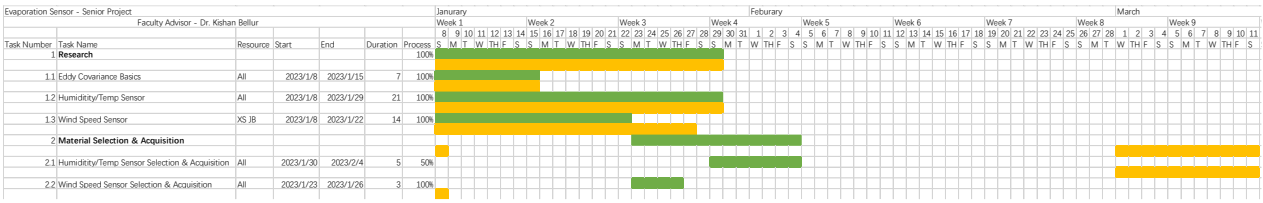


Figure 1 Gantt Chart 1

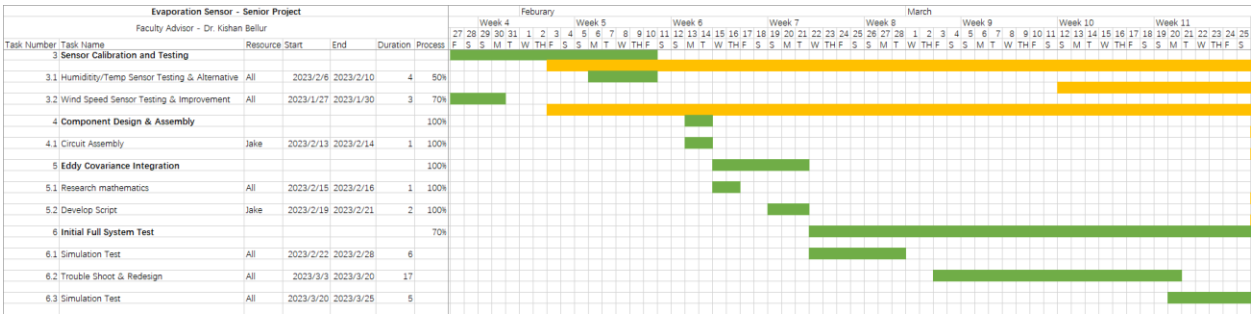


Figure 2 Gantt Chart 2

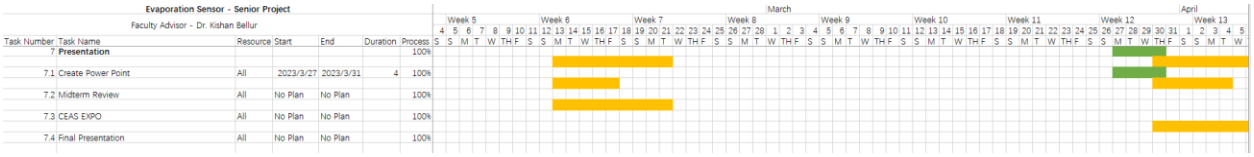


Figure 3 Gantt Chart 3

5. Prototype Development, Construction, and Testing

5.1 Eddy Covariance

5.1.1 Covariance

In probability theory and statistics, covariance is a measure of the joint variability of two random variables. For two independent random variables X and Y, the covariance of X and Y is

$$cov(X,Y) = E\big[\big(X - E(X)\big)\big(Y - E(Y)\big)\big] = E[XY] - E[X]E[Y] \quad (1)$$

Where E[X] is the expected value of X, or the average of X.

5.1.2 Assumptions

Several assumptions when measuring the vertical flux in the most cases.

- Assumption 1: Air density fluctuations are assumed to be negligible.
- Assumption 2: Mean vertical flow is assumed to be negligible for horizontal homogeneous terrain (no divergence/convergence).

5.1.3 Eddy Covariance [2]

In turbulent flow, vertical flux can be presented as:

$$F = \overline{\rho_d w s} \quad (2)$$

Which means the flux is equal to a mean product of dry air density ρ_d, vertical wind speed w, and the dry mole fraction of the gas of interest s. Applying Reynold decomposition to this equation, it will become

$$F = \overline{(\bar{\rho}_d + \rho_d')(\overline{w} + w')(\bar{s} + s')} \quad (3)$$

$\overline{Var}$ represents the mean of variable and Var' represents the deviation of variable. where $\overline{\rho_d}$ represents the mean of air density, w' and s' represent the deviation of vertical wind speed and dry mole fraction of the gas of interest respectively.

Open the parentheses of the formula 3, formula 4 could be got:

$$F = \overline{\overline{\rho_d}\overline{w}\overline{s} + \overline{\rho_d}\overline{w}s' + \overline{\rho_d}w'\overline{s} + \overline{\rho_d}w's' + \rho_d'\overline{w}\overline{s} + \rho_d'\overline{w}s' + \rho_d'\acute{w}\overline{s} + \rho_d'w's'} \quad (4)$$

The average deviation is zero. Take the dry air density as an example. The deviation of dry air density is expressed:

$$\rho_d' = \rho_d - \overline{\rho_d}$$

$$E(\rho_d') = E(\rho_d - \overline{\rho_d}) = E(\rho_d) - E(\overline{\rho_d}) = \overline{\rho_d} - E(\overline{\rho_d}) \quad (5)$$

Because $\overline{\rho_d}$ is constant,

$$E(\overline{\rho_d}) = \overline{\rho_d}$$

Therefore,

$$E(\rho_d') = 0$$

Regarding the average deviation is zero, $\overline{\rho_d}\overline{w}\acute{s} = \overline{\rho_d}\acute{w}\bar{s} = \acute{\rho_d}\overline{w}\bar{s} = 0$. Then formula 5 is simplified to:

$$F = \overline{\rho_d}\overline{w}\bar{s} + \overline{\rho_d}\overline{w's'} + \overline{w}\overline{\rho_d's'} + \bar{s}\overline{\rho_d's'} + \overline{\rho_d'w's'} \quad (6)$$

Applying assumption 1, the air density fluctuations are assumed to be negligible, $\acute{\rho_d} = 0$, therefore,

$$\overline{w}\overline{\rho_d's'} = \bar{s}\overline{\rho_d's'} = \overline{\rho_d'w's'} = 0$$

$$F = \overline{\rho_d}\overline{w}\bar{s} + \overline{\rho_d}\overline{w's'} \quad (7)$$

Applying assumption 2, the average of vertical flow is assumed to be negligible, $\overline{w} = 0$, then the eddy covariance formula is derived:

$$F = \overline{\rho_d}\overline{w's'} \quad (8)$$

5.2 Wind Speed

5.2.1 HC-SR04 Ultrasonic sensor

The HC-SR04 is a popular ultrasonic sensor that is commonly used for distance measurement in robotics, automation, and other projects because it is a low-cost and reliable sensor that can measure distances from 2 cm to 4 m with an accuracy of up to 3 mm. The detailed parameters are shown in Table 1. It consists of an ultrasonic transmitter and receiver.

The HC-SR04 sensor has four pins: VCC, GND, Trig, and Echo. The VCC and GND pins are used to provide power to the sensor, while the Trig pin is used to trigger the sensor to send out an ultrasonic pulse. The Echo pin is used to receive the ultrasonic pulse after it bounces off an object and returns to the sensor.

To measure the distance, the Trig pin is set to a high level for a short period of time (10 μs), which sends out eight ultrasonic pulses. The pulse travels through the air until it hits an object, and then returns to the sensor. The Echo pin detects the reflected pulse, and the time it takes for the pulse to travel to the object and back is measured. The formula is shown in Section 5.2.2.

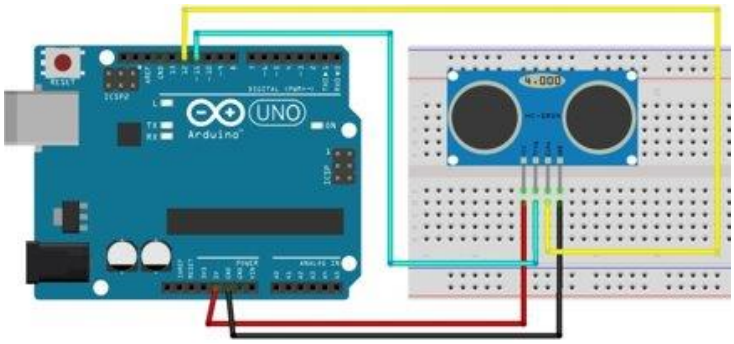


Figure 4 Circuit of HC-SR04 [4]

Table 1. The parameters of HC-SR04 Sensor

Parameter	Value
Operating Voltage	5V DC
Operating Current	15mA
Operating Frequency	40KHz
Min Range	2cm / 1 inch
Max Range	400cm / 13 feet
Accuracy	3mm
Measuring Angle	<15°
Dimension	45 x 20 x 15mm

5.2.2 Equation development

In nature, the wind will directly act on the sound propagation process, and its formula is,

$$S_{s\,actual} = S_{s\,nowind} + S_w \tag{8}$$

where $S_{s\,actual}$ is speed of sound in wind, $S_{s\,nowind}$ is the speed of sound in the condition of no wind and S_w is speed of sound. The speed of sound could be measured through the ultrasonic sensors (as shown in Figure 5) using following formula.

$$S_{s\,nowind} = \frac{2D}{T_{no\ wind}} \tag{9}$$

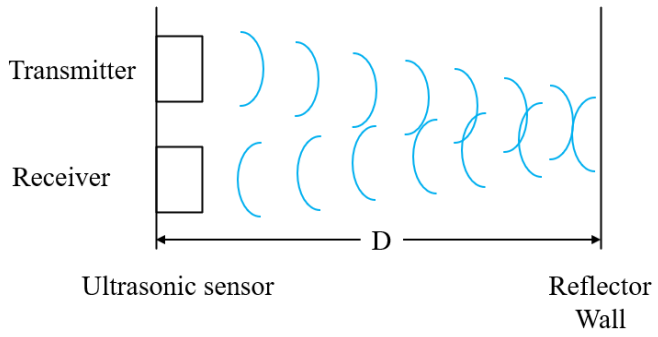


Figure 5 Ultrasonic sensor

where $T_{no\ wind}$ is the time of flight (TOF) that an ultrasonic wave takes to arrive at the receiver in the condition of no wind. D is the distance between the reflector and ultrasonic sensor. Assume the wind exists, according to the formula 8, the practical speed of sound when ultrasonic wave emitted from transmitter to reflector is $S_{s_{nowind}} + S_w$ and the practical speed of sound when ultrasonic wave emitted from reflector to receiver is $S_{s_{nowind}} - S_w$. Formula 11 could be established, and the speed of wind can be calculated.

$$\frac{D}{S_{s_{nowind}} + S_w} + \frac{D}{S_{s_{nowind}} - S_w} = T_{wind} \tag{10}$$

$$S_{wind} = \sqrt{S_{s_{nowind}}^2 - \frac{2DS_{s_{nowind}}}{T_{wind}}} \tag{11}$$

5.3 Humidity

5.3.1 BME 280 Humidity, Temperature and Pressure Sensor

BME280 has a compact form of 2.5 mm ×2.5 mm× 0.93 mm making it ideal for this project. BME280 is Arduino and Raspberry Pi compatible with four pins: VIN as the power supply, GND as ground, SCL as a serial clock, and SDA as a serial data pin [7]. Temperature is measured in degrees Celsius, atmospheric pressure is measured in hectopascals and relative humidity is measured as a percentage. The operating range for temperature is -40 to 80 ℃, 0 to 100% for relative humidity, and 300 to 1100 hPa for atmospheric pressure.

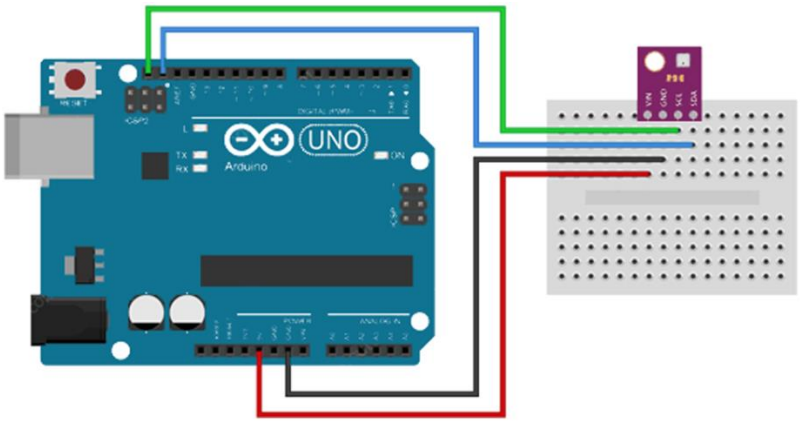


Figure 6 BME280 Wiring

To test the accuracy of the BME280, the sensor collected data from a room with no additional wind or humidity sources for one minute. The temperature was compared to a meat thermometer, the relative humidity was compared to a DHT11 humidity sensor, and the atmospheric pressure was compared to The Weather Channel's atmospheric pressure data for Cincinnati, Ohio.

Table 2 BME280 Temperature Test

Measurement Device	Temperature (°C)	Percent Error (%)
BME280	19.5	0.9051
Meat Thermometer	19.3	

Table 3 BME280 Relative Humidity Test

Measurement Device	Relative Humidity (%)	Percent Error (%)
BME280	38.7	4.3461
DHT11	40.5	

Table 4 BME280 Atmospheric Pressure Test

Measurement Device	Atmospheric Pressure (hPa)	Percent Error (%)
BME280	991.53	3.2706
The Weather Channel	1025	

All three parameters fell under the accepted 5% error.

5.3.2 Equation Development

Psychometrics, the study of systems with dry air and water vapor, definitions and principles were used to develop an equation for the water vapor mole fraction. To use psychometrics the Dalton Model, the idea that mixture components behave as an ideal gas in the same volume and temperature of the mixture, was assumed. Individual components of a mixture do not exert the mixture pressure, rather exert a partial pressure. The partial

pressure, "Of component i, p_i, is the pressure that n_i moles of component i would exert if the component were alone in the volume V at the mixture temperature T" [10]. The sum of partial pressures is equal to the mixture pressure.

$$p_i = \frac{n_i RT}{V}$$

$$p = \sum_{i=1}^{j} p_i \tag{12}$$

The mixture used in this equation development is moist air and its components are dry air and water vapor.

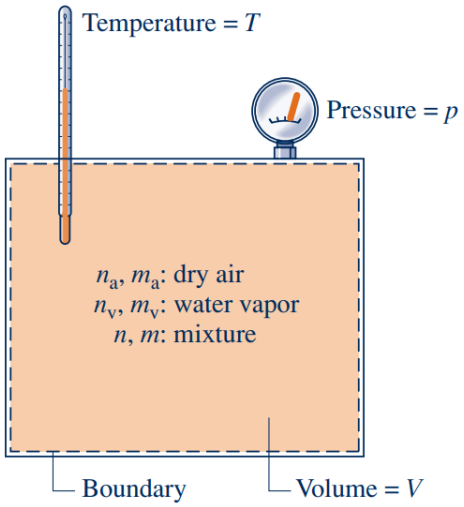


Figure 7 Dalton Model: Dry Air and Water Vapor in Moist Air Mixture

The moist air mixture is assumed to behave as an ideal gas; thus, the mixture pressure is

$$p = \frac{nRT}{V} = \frac{mRT}{MV} \tag{13}$$

where n, m, and M are the moles, mass, and molecular weight of the moist air.

Using the Dalton Model, the sum of partial pressures from the dry air and water vapor are equal to the mixture pressure,

$$p = p_a + p_v \tag{14}$$

where p_a is the partial pressure for dry air and p_v is the partial pressure for water vapor. To determine the partial pressures, the ideal gas law is used,

$$p_a = \frac{n_a RT}{V} = \frac{m_a RT}{M_a V}$$

$$p_v = \frac{n_v RT}{V} = \frac{m_v RT}{M_v T} \tag{15}$$

where n_a, m_a, and M_v are the moles, mass, and molecular weight for dry air and n_v, m_v, and M_v are the moles, mass, and molecular weight for water vapor.

The water vapor mole fraction is derived from atmospheric pressure, temperature, and relative humidity. To start, the humidity ratio or specific humidity is defined as the ratio of the mass of water vapor to the mass of dry air [10].

$$\omega = \frac{m_v}{m_a} \tag{16}$$

The humidity ratio can be expressed in terms of density.

$$\omega = \frac{m_v}{m_a} = \frac{\rho_v V}{\rho_a V} = \frac{\rho_v}{p_a} \tag{17}$$

Dry air density and water vapor density can be expressed as

$$\rho_a = \frac{p_a}{R_d T} = \frac{p - p_v}{R_d T}$$

$$\rho_v = \frac{x_v M_v}{V_m} = \frac{x_v M_v p}{RT} \tag{18}$$

where $R = 8.314\ J\ mol^{-1}K^{-1}$ is the universal gas constant, $R_d = 287.058\ J\ kg^{-1}K^{-1}$ is the specific gas constant for dry air, $M_v = 18.02\ g\ mol^{-1}$ is the molar mass of dry air, x_v is the water vapor mole fraction, and V_m is the molar volume.

Substituting the densities, the following equation is obtained.

$$\omega = \frac{x_v M_v p}{RT} \times \frac{R_d T}{p - p_v}$$

$$x_v = \omega \times \frac{p - p_v}{R_d T} \times \frac{R}{M_v p} \tag{19}$$

The mixture pressure p can be obtained from the pressure measurement on the BME280 sensor. The humidity ratio ω can be found using pressure, temperature, and relative humidity measurements.

$$\omega = \frac{m_v}{m_a} = \frac{\frac{M_v p_v V}{RT}}{\frac{M_a p_a V}{RT}} = \frac{M_v p_v}{M_a p_a} = 0.622 \frac{p_v}{p - p_v} \tag{20}$$

The ratio of molar weights is approximately 0.622. To determine water vapor pressure p_v , the definition of relative humidity is used.

$$\varphi = \frac{p_v}{p_g} \tag{21}$$

Where φ is relative humidity and is measured from the BME280 sensor and p_g is saturation pressure of water, pressure at which water vapor is in thermodynamic equilibrium with it condense state [14]. Saturation pressure can be determined using the Clausius-Clapeyron Equation [16].

$$p_g = 0.6112 f(p) e^{\left(\frac{17.62T}{T+243.12}\right)}$$

$$f(p) = 1.0016 + 3.15 \times 10^{-5} p - \frac{0.0074}{p} \tag{22}$$

Where T is temperature measured by BME280 sensor in Celsius and p is atmospheric pressure measured by BME280 sensor in kPa.

5.4 Low-Cost Eddy Covariance System

The low-cost eddy covariance system prototype consisted of an Arduino Uno microcontroller, BME280 Humidity sensor, HC-SR04 Ultrasonic Anemometer, and power cable.

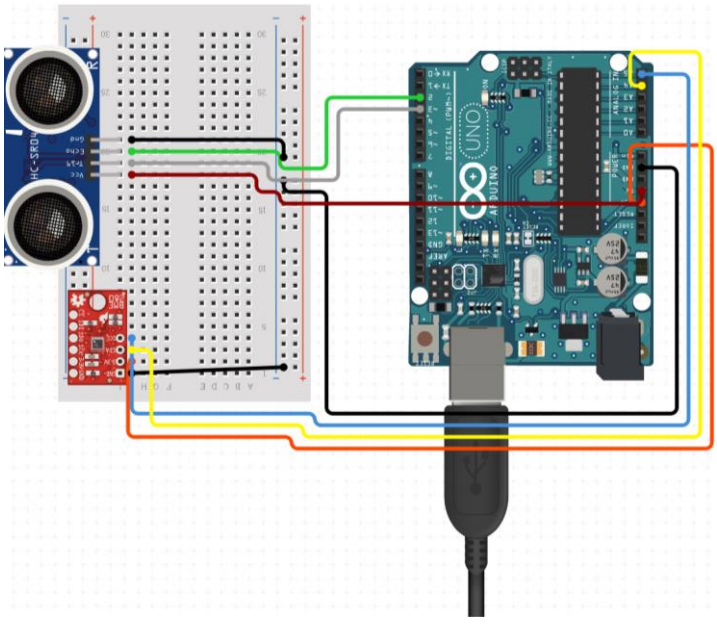


Figure 8 Low-Cost Eddy Covariance System Circuit Diagram

The pin connections for the HC-SR04 and BME280 to the Arduino Uno are found in Table 5 and 6. Data was collected through the power cable that was connected to a laptop.

Table 5 HC-SR04 Pin Connections

HC-SR04 Pin	Arduino Uno Connection
Vcc	5v
Trig	~3
Echo	2
Gnd	GND

Table 6 BME280 Pin Connections

BME280 Pin	Arduino Uno Connection
Vcc	5v
Gnd	GND
SCL	A5
SDA	A4

5.5 Testing and Results

There were four simulations each one minute in length to test the low-cost eddy covariance system. Simulation one had no additional wind speed or humidity. Simulation two introduced additional humidity but no additional wind speed. Simulation three had additional wind speed but no additional humidity. Simulation four had both additional wind speed and humidity. A hair dryer was used to simulate wind speed and a humidifier was used to simulate additional humidity.

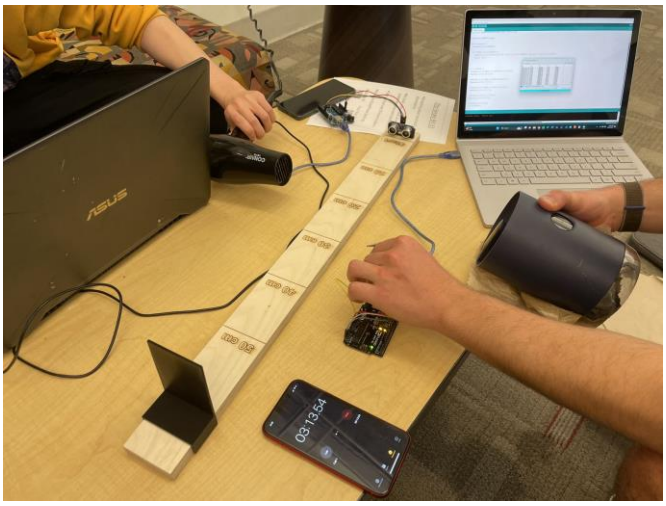


Figure 9 Additional Wind Speed and Humidity Test

Results of the test can be seen below.

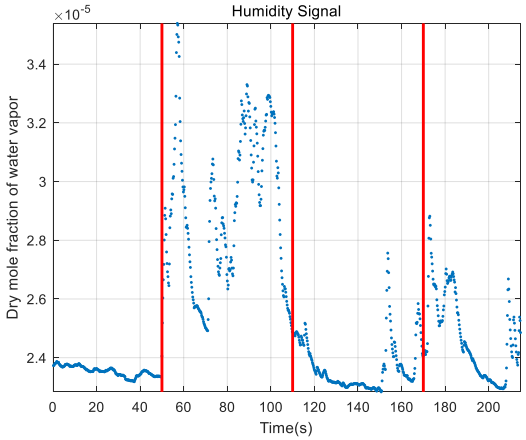


Figure 10 Humidity Signal

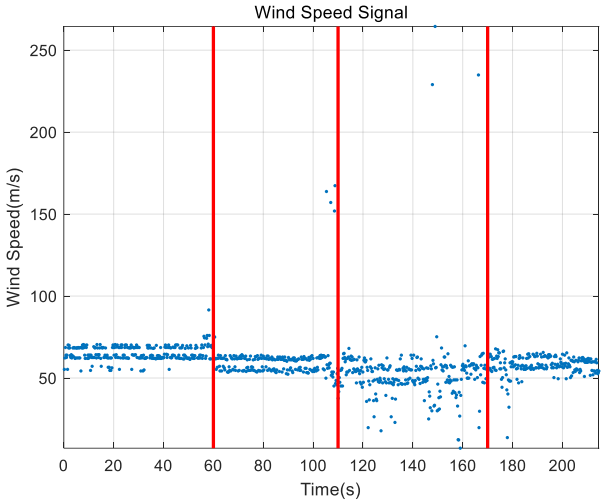


Figure 11 Wind Speed Signal

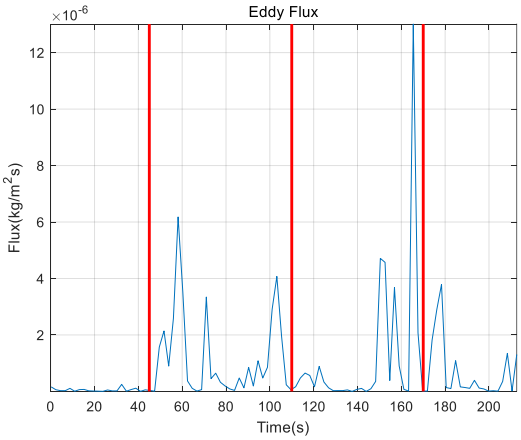


Figure 12 Eddy Flux

Each red line represents one minute in time. The humidity signal, Figure 10, was successful as there are clear fluctuations in the data when additional humidity was introduced. However, there are issues with the wind speed signals. The wind speed ranged from 50 – 70 meters per second which equates to a serve storm. The issue with the wind sensor is discussed in Section 6.

6. Design Alternatives and Basis for Selection

6.1. Wind Speed Alternatives

The result of the experiment for wind speed test in section shown in Figure 11, the average measured wind speed is about 50 m/s which is unprecise. Two reasons cause this problem. First, the air outlet area of the hair dryer is very small, it is difficult to generate an effective eddy which generates a huge vertical wind speed change. Second, the operating frequency of ultrasonic sensor influences the accuracy. The HC-SR04 ultrasonic sensor is a discrete sensor, it can only receive the data at discrete time. The time interval depends on operating frequency, the formula of time resolution is,

$$\Delta t = \frac{1}{fre}$$

Assume the TOF in no wind condition is $T_{no\ wind}$, the TOF (T_{wind}) could be only measured in these points,

$$T_{wind} = T_{no\ wind} + n \times \frac{1}{fre}\ (n \in 0, \pm 1, \pm 2, \pm 3 \dots) \tag{23}$$

Assume distance between the objective and sensor is 1 m, ultrasonic speed is 340m/s. In order to calculate the measured minimum wind speed, the value of n in Formula 23 is taken 1. Put n=1, formula 23 into formula 11, the equation will be derived:

$$S_{wind} = \sqrt{S_{ult}^2 - \frac{2DS_{ult}}{T_{wind}}} = \sqrt{\frac{340^2 m^2}{s^2} - \frac{2 \times 1\ m \times \frac{340m}{s}}{\frac{2 \times m}{340\frac{m}{s}} + 1 \times \frac{1}{fre}}} \tag{24}$$

Therefore, the relationship between the operating frequency of sensor and measured minimum wind speed is established. Figure 13 clearly shows this relationship and when the operating frequency is 40kHz which is the HC-SR04 operating frequency, the measured minimum wind speed is 22 m/s. We can consider the operating frequency of the sensor to be the largest source of error in wind speed experiments.

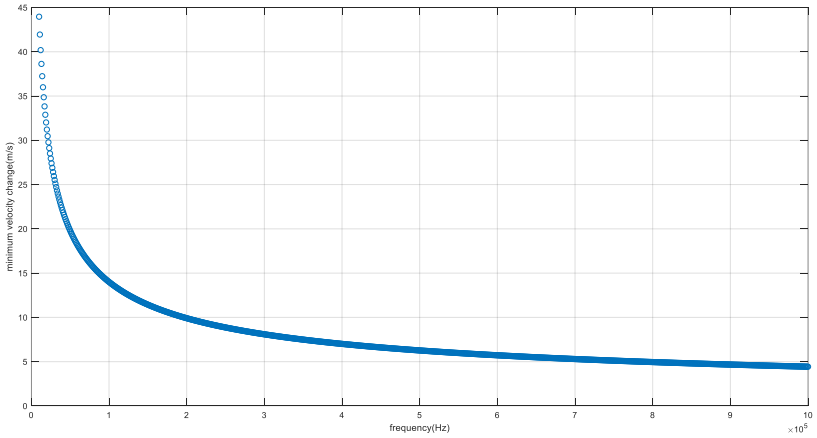


Figure 13 Frequency vs. Minimum Wind Speed

There are three alternative methods to improve the accuracy of wind speed test: change the operating frequency of ultrasonic sensor, inclined ultrasonic sensor layout and cross-correlation method [13].

6.1.1 Change the Operating Frequency of Ultrasonic Sensor

The easiest method to increase the accuracy of wind speed measurement is to replace HC-SR04 with a high operating frequency ultrasonic sensor, such as MA300D 1-1 (300 kHz).

These are the reasons why we do not choose this method. First, high operating frequency sensor means high price which violates the project goal of low-cost. Second, most high operating frequency sensor is only one transducer which required amplified circuit to generate the output signal, our group do not have experience with circuit.

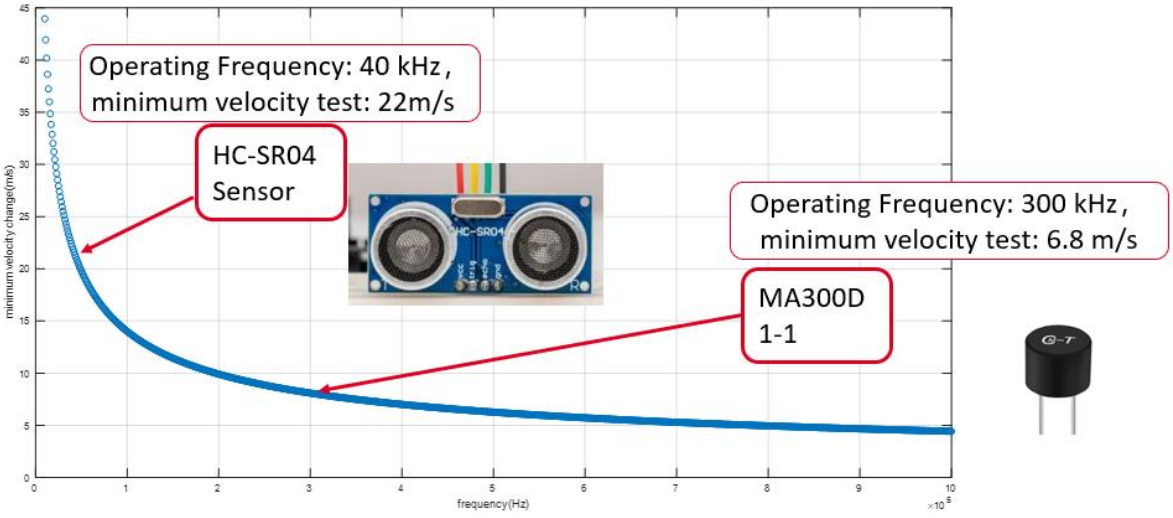


Figure 14 Different operating frequency sensor

6.1.2 Inclined Ultrasonic Sensor Layout

In our initial design, the ultrasonic sensor that we use to measure the up-flux wind speed is placed vertically. This alternative design requires to place the ultrasonic sensors with an inclination angle to the horizontal plane.

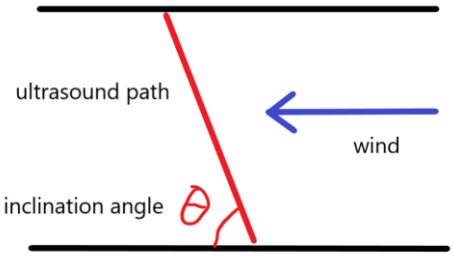


Figure 15 Inclined Ultrasonic Sensor placement

The principle of inclined ultrasonic sensor layout is that the accuracy of the wind speed sensor when the wind speed is high is higher than that when the wind speed is low and the accuracy will be influenced by the distance between the sensor and the reflective surface (the ultrasound path increases as θ decrease). The theoretical derivation is as follows:

The wind speed along the inclination sound path is

$$S_{incline} = S_{wind} \times cos\theta + S_{up-flux} \times sin\theta \tag{25}$$

S_{wind} is the horizontal wind speed, $S_{up\text{-}flux}$ is the up-flux speed, θ is the inclination angle.

Assume, there is no up-flux but only the horizontal speed initially, the initial time can be calculated as

$$T_1 = \frac{2D_{inclination}S_{ult}}{S_{ult}^2 - (S_{wind} \times cos\theta)^2} \tag{26}$$

The wind speed sampled at next time point will be (according to Formula 23)

$$T_2 = T_1 + \frac{1}{frequency} \tag{27}$$

Through which we can solve for the minimum wind speed change

$$\Delta S_{up-flux} = \frac{\sqrt{S_{ult}^2 - \frac{2D_{inclination}S_{ult}}{T_2}}}{sin\theta} - \frac{S_{wind}}{tan\theta} \tag{28}$$

This design could significantly improve the resolution of our measured data according to our calculations. Figure 16 shows the resolution of sensors with different frequency under varying inclination angle. It can be found that we can achieve the optimal resolution curve at

45 degrees inclination angle, which greatly improves the resolution compared with vertical ultrasonic sensor layout.

However, the inclined ultrasonic sensor layout required the measured horizontal wind speed. If we use another ultrasonic sensor to measure the horizontal wind speed, another large source of error will be introduced.

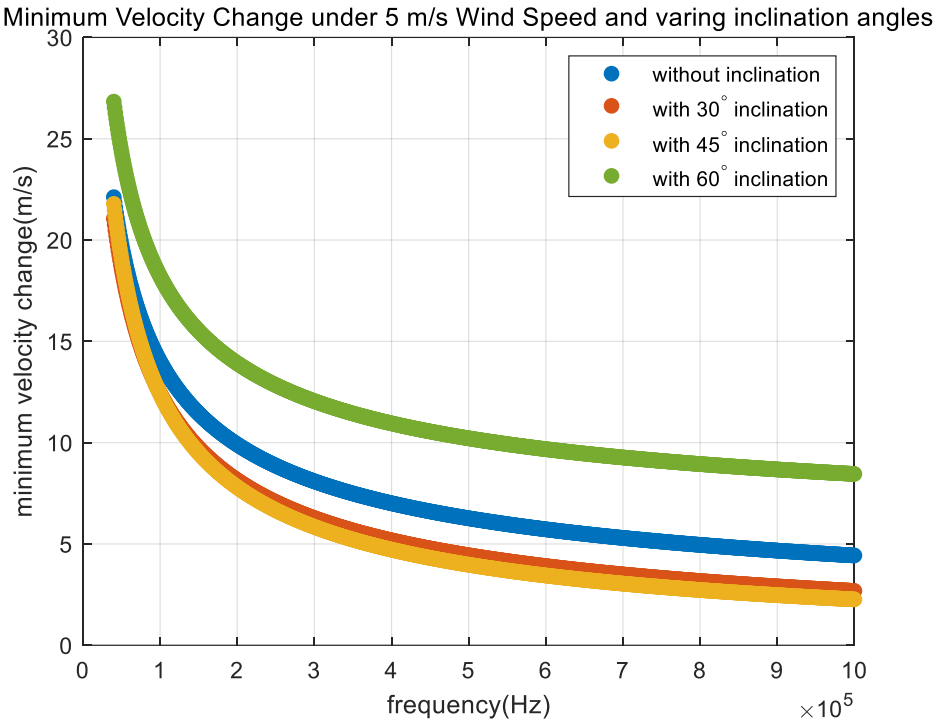


Figure 16 Resolution of measured data with different inclination angle

6.1.3 Cross-Correlation methods

Introduction

TOF estimation can be measured by various methods, but the most widely used ones are the threshold and cross-correlation methods [1].

The threshold method [12] is a simple and quick approach that involves determining the TOF by detecting when the received signal first exceeds a pre-defined threshold level after the transmission of a pulse train. This method is directly used in the HC-SR04, the echo pin output the result of threshold method.

The cross-correlation method [1] is a more suitable approach for estimating TOF. This method involves cross correlating the transmitted and received signals, and determining the time at which the correlation result reaches its maximum as an estimate of the TOF. Figure 17 shows an example of cross-correlation method. Queirós [13] proposed an improved

cross-correlation approach to estimate TOF. He used parabolic interpolation around the maximum of the cross-correlation's magnitude to get more accurate TOF.

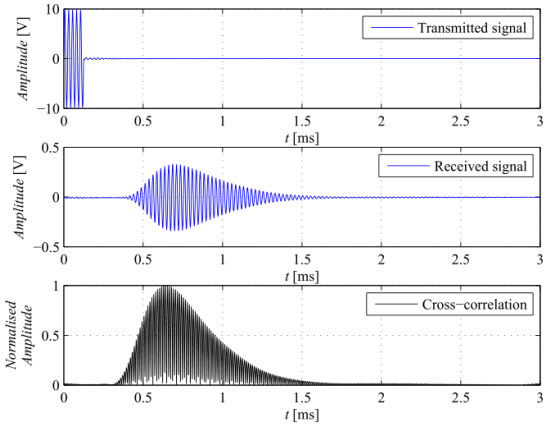


Figure 17 TOF estimation by cross-correlation[13]

Experimental procedures

The experimental procedures are shown below.

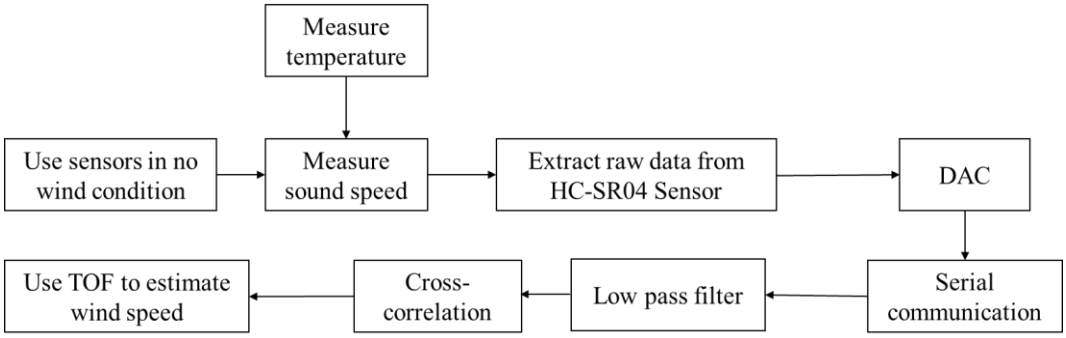
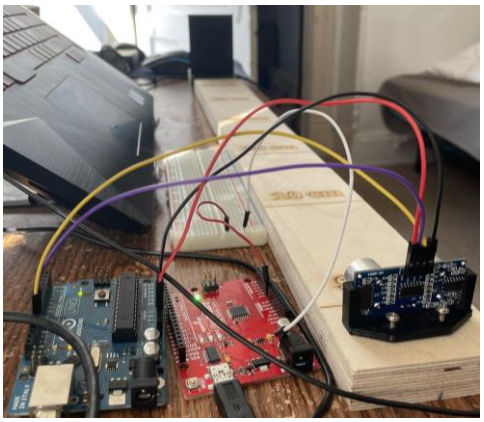
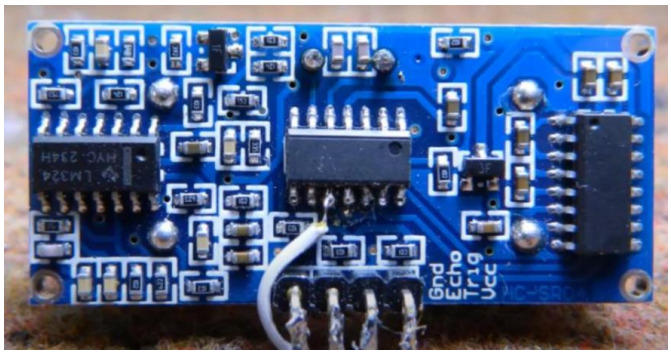


Figure 18 Flow chart of cross-correlation methods

Experimental Setup

The experimental setup is shown below. The raw data of received signal of HC-SR04 could be extracted directly from the 10th pin of middle chip in HC-SR04. The Arduino UNO is a one-core microcontroller, and it can only finish one task at same time. In order to get real-time output received signal (raw data), two Arduino microcontrollers are used in this experiment.





(a) Cross-correlation test **(b) Extract Raw data of received signal of HC-SR04**

Figure 19 Experimental Setup

Limitation & Test Result

The limitations of Arduino microcontroller are low memory storage and one core. Low memory storage means that the Arduino microcontroller cannot save large amounts of data at the same time and one-core means that the Arduino microcontroller cannot deal with multiple tasks at the same time. These two limitations lead to Arduino having to use serial communication to upload data to the computer for further processing. However, the speed of serial communication is limited by the baud rate, and it cannot be increased to infinite.

The baud rate is the bytes rate. 115200 baud rate means 115200 bytes/sec and one byte has 8 bits (also has a start bit and a stop bit). So, for each byte, the smaller time is 86.8 μs (1/11520). This is the result for testing characters with different lengths (The red block is time for the left number).

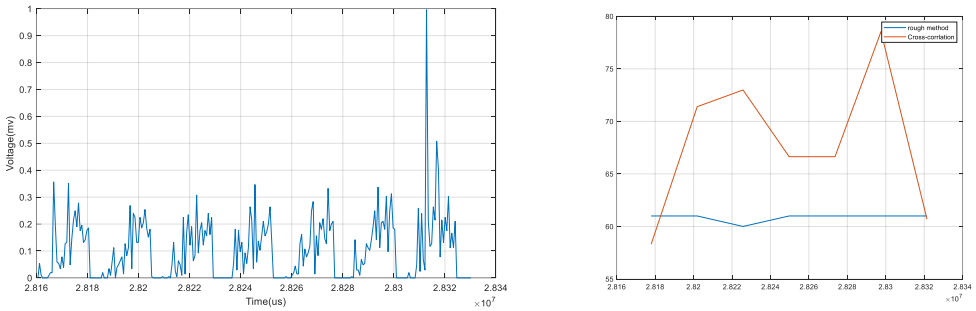
```
23106812, 444, 669, 208, 669, 200, 670
23111424, 448, 678, 204, 679, 208, 679
23116040, 440, 673, 212, 673, 200, 673
23120680, 448, 671, 204, 671, 208, 671
23125324, 444, 671, 212, 671, 200, 671
```

Figure 20 Result for time resolution of experiment

If we use 2,000,000 baud rates to send a four-digit number (the range of analog input of Arduino is 0-1024), the minimum theoretical serial communication is 20 us ($\frac{1\,s}{200{,}000} \times 4 = 20\ us$) and the maximum theoretical sampling frequency is 50k Hz (1/20us = 50 k Hz).

The sampling frequency of Arduino microcontroller is also limited by processing code speed. Using the Arduino build-in function (analogRead) to extract raw data from HC-SR04, the results are shown in Figure 21. Figure 21(a) verifies the feasibility of the cross-correlation method and Figure 211(b) proves that the experiment still has a large error. The reason is

that the average running speed of the analogRead function is 700us which results in 1,400 Hz sampling frequency.



(a) Raw data **(b) Cross-correlation result**

Figure 21 Experimental data by using analogRead function

Overall, the sampling frequency of Arduino microcontroller is limited by processing code speed and serial communication speed. Considering line break symbol and other code execution time, the average sampling frequency is about 16.67 k Hz in our experiment. Figure 22 shows the result of raw data.

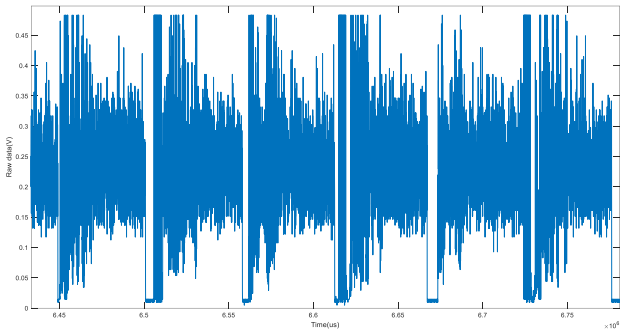


Figure 22 Raw data from HC-SR04

To prevent the noise, the low pass filter is used, Figure 22 shows the result,

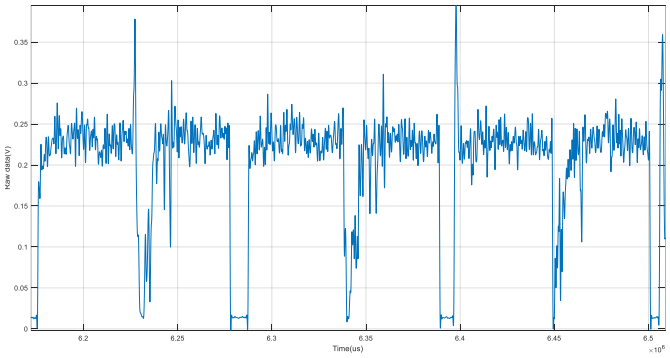


Figure 23 Raw data after low pass filter

After cross correlation of received signal and transmitted signal, the result of distance measurements compared with threshold method are shown below. The results are still not accurate, there is still long way to finish the testing of cross-correlation method.

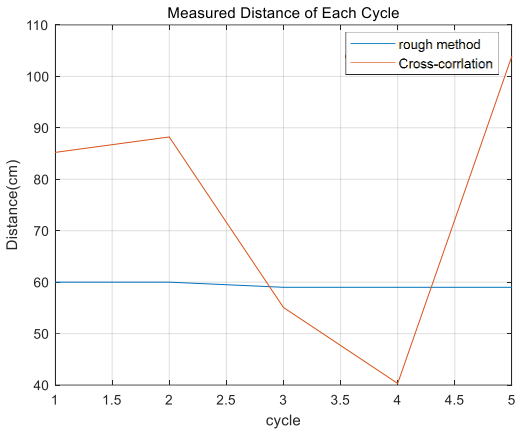


Figure 24 Cross-correlation and threshold method comparison

In order to increase the sampling frequency, the multi-core microcontroller (Ex. Raspberry Pi) should be used. It can finish the cross-correlation at one core and only upload a few results instead of all raw data to computer. In this case, the serial communication speed problem will be prevented.

Apart from that, since this method has not yet achieved the desired effect, the issues of data delay (caused by signal transmission, code processing) and data synchronization have not been considered right now.

6.2. Humidity Alternatives

The Beer-Lambert Law, the "Relationship between the attenuation of light through a substance and the properties of that substance" [6], was considered to determine the concentration of water vapor.

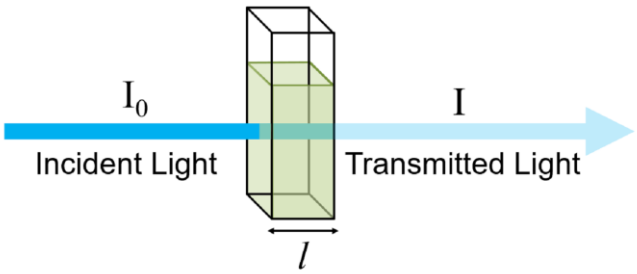


Figure 25 Light Transmitting Through a Medium

Transmittance is the ratio of transmitted light, I, to the incident light, I_0.

$$T = \frac{I}{I_0}$$

Once the transmittance is known, the absorption, A, can be determined.

$$A = -\log_{10} T$$

The Beer-Lambert Law can be represented as a linear relationship between absorbance, concentration, c, molar absorption coefficient, ε, and length of the medium, l [6].

$$A = \varepsilon c l$$

The unit for molar absorption coefficient is $M^{-1}cm^{-1}$, M for molar concentration, and cm for length. The Beer-Lamber Law can be rearranged as follows to determine concentration.

$$c = \frac{A}{\varepsilon l}$$

Visible light was not considered as a light source as visible light from the sun could affect the results. Infrared (IR) light was chosen because water vapor has high absorbance at certain wavelengths for IR light. However, other gases in the atmosphere, carbon dioxide, methane, and nitrous oxide, also absorb infrared light [5]. Therefore, a wavelength needed to be found where the transmittance for IR light through water vapor is low and the transmittance for the remaining gases is high. Using data gathered from the National Institute of Standards and Technology (NIST) a range of wavelength was found.

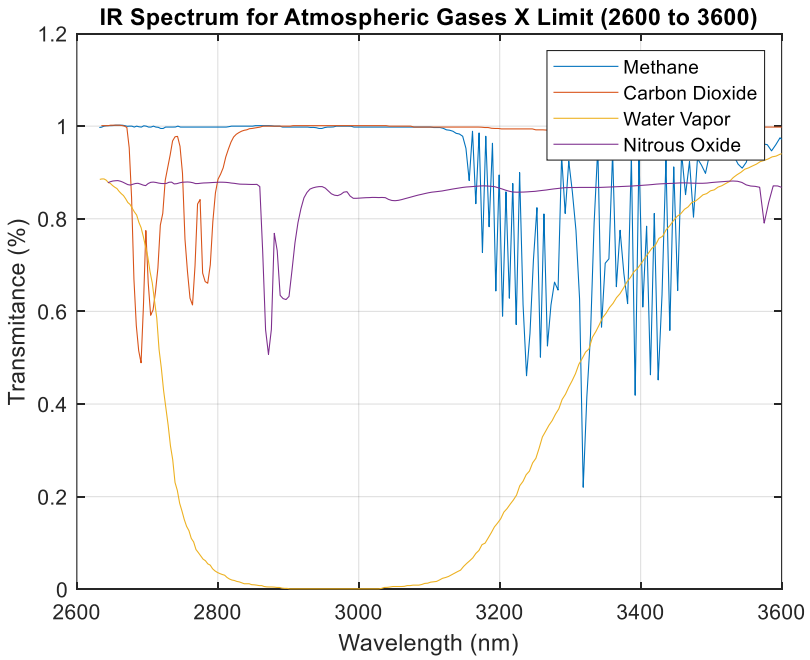


Figure 26 Transmittance vs IR Spectrum

Seen in the Figure 26, 2800 nm to 3200 nm has a low transmittance for water vapor but a high transmittance for methane, carbon dioxide, and nitrous oxide. The issue however is that commonly manufactured IR LEDs emit IR light at 840 nm or 950 nm wavelength. This is outside the range of the data provided by NIST. As a result, this method was not chosen and the use of the BME280 Humidity sensor was.

7. Conclusion

This project has successfully developed the mathematical background of a low-cost Eddy-Covariance system and built a prototype of the system. The prototype consists of three main components including the Arduino Uno board, the HC-SR04 ultrasonic sensor, and the BME280 Humidity sensor, which are all off-the-shelf electronic components. The total cost is under 50 dollars, satisfying the economic constraints. The prototype is able to use measured wind speed and humidity data to calculate the vertical flux from the surface, which verifies the feasibility of such low-cost EC system.

There are still some deficiencies of the prototype requiring future improvements to overcome. The main errors of the system result from the measurement of wind speed. This is caused by the low sampling frequency of the ultrasonic sensor, which can be solved by implementing cross-correlation to the sensor's signals or changing with sensors of higher sampling frequency. Another possible way to overcome the limitation of sampling frequency is to place the system with a 45-degree inclination angle to the horizontal direction. This could greatly increase the resolution of measured wind speed with the same sensor.

The overall performance of the BME280 Humidity sensor is sufficient for the system's requirements. There is also an alternative way to measure the humidity, which is the IR sensor. The IR sensor is working based on the Beer-Lambert Law but is currently limited by the light source wavelength.

8. Future Work

8.1 Wireless Data Acquisition

Currently data are gathered through the power cable which is connected directly to the computer. Since the system is designed to be deployed in multiple numbers on the water bodies to monitor the evaporation, the systems are required to be free standing; thus, wireless data acquisition is needed. Since several systems will be placed on the water bodies, it would be inefficient to gather the data by travelling between different stations.

The easiest way to implement wireless data acquisition is through Wi-Fi Modules of Arduino, which are inexpensive and powerful. However, there could be challenges in wireless data acquisition. Since all the sensors are working on extremely high sampling

frequency and a large quantity of data will be generated, it will be difficult to send the data with minimized delay and avoid conflicts between different systems. The maximum operating distance of the Wi-Fi modules should also be taken into consideration.

8.2 Advance Controller and Sensor

The Arduino Uno board is not a multithread processor and has very limited memory. In our situation, large quantities of data will be generated and processed. Insufficient memory of microcontroller will limit the performance of our system. The system needs to conduct multiple tasks at the same time, including sensor control, data processing, and communications, which requires multithread processor to handle it in case of huge errors in measurement.

The insufficient sampling frequency of the HC-SR04 ultrasonic sensor is the largest limitation of our system's performance. According to our calculations, in order to have the resolution of wind speed sensor at 1 m/s, the ultrasonic sensors should be able to operate at a sampling frequency of 992 kHz. This sampling frequency requirement is subject to sound path length and data processing methods. However, sensors with higher sampling frequency means higher price. It is important to balance the costs and performance in this project. Better data processing methods could be the most cost-effective way to improve performance.

8.3 Waterproof Casing

The goal of the low-cost sensor is to be placed in multiple numbers across the water bodies. To protect the electronics from water vapor corrosion and be able to float on the water, the housing for the system needs to be water-resistant and buoyant. Future senior design teams need to research on the mechanical structure and materials of the waterproof casings. Although currently there are no specified requirements, the size of the whole system is expected not to exceed the size of a suitcase.

References

[1] Barshan, B. (2000). Fast processing techniques for accurate ultrasonic range measurements. *Measurement Science and technology,* 11(1), 45.

[2] Burba, G. (2022). *Eddy Covariance Method for Scientific, Regulatory, and Commercial Applications*. LI-COR Biosciences

[3] Carbon dioxide. (n.d.). Webbook.nist.gov. https://webbook.nist.gov/cgi/cbook.cgi?ID=124-38-9

[4] Circuit of HC-SR04 [Circuit graph]. *Random nerd tutorials*. https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

[5] Climate Science Investigations South Florida - Energy: The Driver of Climate. (n.d.). Www.ces.fau.edu. Retrieved April 21, 2023, from https://www.ces.fau.edu/nasa/module-2/how-greenhouse-effect-works.php#:~:text=Water%20vapor%2C%20carbon%20dioxide%2C%20methane.

[6] Edinburgh Instruments. (2015). Beer lambert law | transmittance & absorbance. Edinburgh Instruments. https://www.edinst.com/blog/the-beer-lambert-law/

[7] Last Minute Engineers. (2022, October 15). Interface BME280 temperature, humidity & pressure sensor with Arduino. Last Minute Engineers. Retrieved April 21, 2023, from https://lastminuteengineers.com/bme280-arduino-tutorial/.

[8] Markwitz, C., & Siebicke, L. (2019). Low-cost eddy covariance: a case study of evapotranspiration over agroforestry in Germany. Atmospheric Measurement Techniques, 12(9), 4677–4696. https://doi.org/10.5194/amt-12-4677-2019

[9] Methane. (n.d.). Webbook.nist.gov. https://webbook.nist.gov/cgi/cbook.cgi?ID=74-82-8

[10] Moran, M. J., Shapiro, H. N., Boettner, D. D., & Bailey, M. B. (2014). Fundamentals of engineering thermodynamics. Wiley.

[11] Nitrous oxide. (n.d.). Webbook.nist.gov. https://webbook.nist.gov/cgi/cbook.cgi?ID=10024-97-2

[12] Parrilla, M., Anaya, J. J., & Fritsch, C. (1991). Digital signal processing techniques for high accuracy ultrasonic range measurements. *IEEE Transactions on instrumentation and measurement*, 40(4), 759-763.

[13] Queirós, R., Martins, R., Silva Girão, P. etc. (2006). A new method for high resolution ultrasonic ranging in air. *XVIII IMEKO WORLD CONGRESS*

[14] Water - Saturation Pressure. (2019). Engineeringtoolbox.com. https://www.engineeringtoolbox.com/water-vapor-saturation-pressure-d_599.html

[15] Water.(n.d.). Webbook.nist.gov. https://webbook.nist.gov/cgi/cbook.cgi?ID=C7732185

[16] Zhou, X., Gao, T., Takle, E. S., Zhen, X., Suyker, A. E., Awada, T., Okalebo, J., & Zhu, J. (2022). Air temperature equation derived from sonic temperature and water vapor mixing ratio for turbulent airflow sampled through closed-path eddy-covariance flux systems. Atmospheric Measurement Techniques, 15(1), 95–115.

Appendix A – MATLAB Code

BME280 Test

```
clear; clc; close all;
% Load test data
data = load('BME280_Test.csv');
temp_bme280 = data(:,2); % C
pressure_bme280 = data(:,3); % hPa
humidity_bme280 = data(:,4); % %
time_bme280 = data(:,5); % Unix time

% Temperature gathered from digital therometer, pressure gathered from
% The Wheather Channel, relative humidity gathered from DHT11 Module
digital_temp = 19.3; % C
pressure = 1025.05984940001*100; % Pa
humidity_dht11 = 40.5; % %


temp_error = abs((mean(temp_bme280)-digital_temp))/digital_temp * 100
pressure_error = abs((mean(pressure_bme280)-pressure))/pressure * 100
humidity_error = abs((mean(humidity_bme280)-humidity_dht11))/humidity_dht11...
    *100
```

IR Spectrum Plots

```
clear; clc; close all;
% Load .csv data gathered from National Institude of Standards and
% Technology (NIST)
ch4 = load('CH4.csv');
co2 = load('CO2.csv');
h2o = load('H2O.csv');
n2o = load('N2O.csv');

ch4_x = (ch4(:,1).^-1) * 10^7;
co2_x = (co2(:,1).^-1) * 10^7;
h2o_x = (h2o(:,1).^-1) * 10^7;
n2o_x = (n2o(:,1).^-1) * 10^7;

figure(1)
plot(ch4_x,ch4(:,2),co2_x,co2(:,2),h2o_x,h2o(:,2),...
    n2o_x,n2o(:,2))
xlabel('Wavelength (nm)'), ylabel('Transmitance (%)')
title('IR Spectrum for Atmospheric Gases'), grid on
legend('Methane','Carbon Dioxide','Water Vapor','Nitrous Oxide')

figure(2)
plot(ch4_x,ch4(:,2),co2_x,co2(:,2),h2o_x,h2o(:,2),...
    n2o_x,n2o(:,2))
xlim([2600 3600])
xlabel('Wavelength (nm)'), ylabel('Transmitance (%)')
```

```
title('IR Spectrum for Atmospheric Gases X Limit (2600 to 3600)'), grid on
legend('Methane','Carbon Dioxide','Water Vapor','Nitrous Oxide')
```

BME280 Water Vapor Concentration Post Processing

```
clear; clc; clear all;
% Load test data
data = load('final_test.csv');
temp = data(:,3); % C
p_t = data(:,4) * 100; % Pa
rh = data(:,5); % (%)
t = data(:,6); % ms

% Transpose the data
temp = temp';
p_t = p_t';
rh = rh';

% Constants
R = 8.314; % J mol^-1 K^-1
R_d = 287.058; % J kg^-1 K^-1
M_v = 18.02/1000; % kg mol^-1

a = [];
for i = 1:length(data)
    % Calculate saturation pressure
    p_g = 0.6112 * (1.0016 + (3.15 * 10^-5 * p_t(i)) - (0.0074 * p_t(i)^-1))*...
        exp((17.62*temp(i))/(temp(i)+243.12)); % Pa
    % Calculate water vapor partial pressure
    p_v = (rh(i) * p_g) / 100; % Pa
    % Calculate humidity ratio
    w = 0.622 * p_v / (p_t(i) - p_v); % unitless
    % Calculate mole water vapor fraction
    x_n = w * ((p_t(i) - p_v)/R_d) * (R/(M_v*p_t(i))); % unitless
    % Store into array
    a = [a,x_n];

end
```

Calculation of the Influence of Resolution

```
%% Capstone
% Date: 02/08/2023
% Name: Hu Jiabin
% Description: Calculation the wind speed
%% Setup
clc;clear;close all

%% initial
Sound_speed = 340; % m/s; The speed of sound
```

```
%% The different resolution result in the different minimum velocity
% Word Part 4.1
% From the figure the minimul velocity detacted by the 0.3cm resolution
% will be 18 m/s
figure
x = 0:0.00001:0.01;  % m; different resolution in distance
D_given = 2;  % m; The distance between the reflector and the sensor is 2 meters
y = sqrt(Sound_speed^2.*x./(D_given+x)); % the formula 3 in Word

semilogx(x*100,y)

grid on
xlabel('Resolution(m)')
ylabel('minimum velocity change(m/s)')

%% The Difference distance between the reflector and sensors with resolution 0.3cm
% Word part 4.2
figure
D_Resoluion = 3e-3; % m; Resolution 0.3cm
D_given = 0.02:0.01:4; % m; Different Given distance

y = sqrt(Sound_speed^2.*D_Resoluion./(D_given+D_Resoluion)); % the formula 3 in Word

semilogx(D_given,y)
grid on
xlabel('Given Distance(m)')
ylabel('minimum velocity change(m/s)')

%% Using 0.3cm resolution, what speed could be measured
% Word part 4.3

figure
x = 0:3e-3:0.1; % discrete point of different times of resolution in distance
D_given = 1; % distance between the reflector and sensor is 1 meter

y = sqrt(Sound_speed^2.*x./(D_given+x));% the formula 3 in Word

plot(x,y,'o')
grid on
xlabel('n \times Resolution(m)')
ylabel('minimum velocity change(m/s)')

%% Given distance is 1m, the influence of frequency of sensor
% Word part 5.1

D_Given = 1; % m
frequency = 40e3 :1e3:1000e3;
T_nowind = 2*D_Given/Sound_speed;
```

```
T_wind = T_nowind + 1./frequency;
Wind_speed = sqrt(Sound_speed^2 - 2 * D_Given * Sound_speed./T_wind);
figure(5)
plot(frequency,Wind_speed,'o')
grid on
hold on
xlabel('frequency(Hz)')
ylabel('minimum velocity change(m/s)')
figure(6)
plot(frequency,Wind_speed,'o')
grid on
hold on
xlabel('frequency(Hz)')
ylabel('minimum velocity change(m/s)')
%% Using the resolution of distance to measure the influence of the resolution
% Word part 5.2
figure(4)
frequency = 10e3 :1e3:1000e3;
deltaT = 1 ./ frequency;
D_Resoluion = deltaT * 340 / 2;
D_given = 1 ;
y = sqrt(Sound_speed^2.*D_Resoluion./(D_given+D_Resoluion));
plot(frequency,y,'o')
grid on
xlabel('frequency(Hz)')
ylabel('minimum velocity change(m/s)')
%%
theta = [pi/3 pi/4 2*pi/3]; % inclination angle
Wind_speed = [5 10 20]; % wind speed horizontal
frequency = 40e3 :1e3:1000e3;
D = 1; % vertical distance
% D_incline = D/sin(theta);
% Inclination_speed = Wind_speed*cos(theta) + Up_speed*sin(theta);
for i = 1:3
   D_incline = D./sin(theta(i));
   T1 = 2*D_incline*Sound_speed/(Sound_speed^2-(Wind_speed(1)*cos(theta(i)))^2);
   Up_speed(i,:) = -Wind_speed(1)/tan(theta(i))+sqrt(Sound_speed^2-
2*D_incline*Sound_speed./(T1+1./frequency))/sin(theta(i));
end
for i = 1:3
   D_incline = D./sin(theta(2));
   T1 = 2*D_incline*Sound_speed/(Sound_speed^2-(Wind_speed(i)*cos(theta(2)))^2);
   Up_speed(i+3,:) = -Wind_speed(i)/tan(theta(2))+sqrt(Sound_speed^2-
2*D_incline*Sound_speed./(T1+1./frequency))/sin(theta(2));
end

figure(5)
plot(frequency,Up_speed(1:3,:),'o')
```

```
grid on
legend('without inclination','with 30^\\circ inclination','with 45^\\circ inclination','with 60^\\circ inclination')
title('Minimum Velocity Change under 5 m/s Wind Speed and varing inclination angles')
figure(6)
plot(frequency,Up_speed(4:6,:),'o')
grid on
legend('without inclination','5 m/s wind speed','10 m/s wind speed','20 m/s wind speed')
title('Minimum Velocity Change under varing Wind Speed and 45^\circ inclination angle')
```

Cross-Correlation of Raw Data from HC-SR04

```
%% Capstone
% Date: 03/27/2023
% Name: Hu Jiabin
% Description:
%% Setup
clc;clear;close all

%% read txt
% read ultrasonic
fileID = fopen('ultrasonic.txt');
Cult = textscan(fileID,'%f %s %f');
fclose(fileID);

% read ultrasonicamplified
fileID = fopen('ultrasonicamplified.txt');
Camp = textscan(fileID,'%f');
Campex = Camp{1, 1};

% the low pass filiter
Campex = lowpass(Campex ,500,19230);
fclose(fileID);
%% plot
% average sampling delta time
Delta = 50.7735;
x = 0:Delta:Delta*(length(Camp{1, 1})-1);
% the figure for detact distance
figure
plot(Cult{1, 1},Cult{1, 3})
ylim([58,62])
xlabel('Time(us)')
ylabel('Rough Distance(cm)')
% the figure for raw data
figure
plot(x,Campex * 5/1024);
grid on
xlabel('Time(us)')
```

```
ylabel('Raw data(V)')

%% process of cross-correlation
x = x(723:end);
Campex = Campex(723:end);
% prevent large noise
Campex(Campex == 99) = 0;
interval = 1083;
start = 5366;
% assume the transmit signal
pulse = [1 1 1 1 1 1 1 1];
for i = 1:5
    figure
    plot(x(start:start+interval),Campex(start:start+interval))
    grid on

    figure
    % cross-correlation
    [c,lags] = xcorr(Campex(start:start+interval),pulse);
    % find maximum point
    loc = find(c==max(c));
    if length(loc) ~= 1
        locs(i)= mean(loc);
    else
        locs(i)=loc;
    end
    stem(lags,c)
    start = start+interval;
end
figure
% figure of the result of cross-correlation
first_element=find(Cult{1, 1}>x(5366),1);
second_element=find(Cult{1, 1}>x(start),1)-1;
plot(Cult{1, 3}(first_element:second_element))
hold on
plot((locs-700)*52e-6 * 340/2 * 10)
grid on
title('Measured Distance of Each Cycle')
xlabel('cycle')
ylabel('Distance(cm)')
```

Eddy Covariance Calculation

```
%% Eddy Covariance
% Load Data
concentration_data = load('mole_water_fraction.mat');
concentration_data = concentration_data.a;
concentration_data = concentration_data(95:1094);
wind_data = csvread('wind_speed_test.csv')';
```

```
dt = (wind_data(1,1:1000)-2)/1e3;
wind_data = wind_data(2,1:1000);
figure
plot(dt, concentration_data,'.')
hold on
line([50 50],[0 max(concentration_data)], 'Color','r', 'LineWidth', 2)
line([110 110],[0 max(concentration_data)], 'Color','r', 'LineWidth', 2)
line([170 170],[0 max(concentration_data)], 'Color','r', 'LineWidth', 2)
axis([0 max(dt) min(concentration_data) max(concentration_data)])
xlabel('Time(s)')
ylabel('Dry mole fraction of water vapor')
title('Humidity Signal')
grid on
figure
plot(dt, wind_data,'.')
hold on
line([60 60],[0 max(wind_data)], 'Color','r', 'LineWidth', 2)
line([110 110],[0 max(wind_data)], 'Color','r', 'LineWidth', 2)
line([170 170],[0 max(wind_data)], 'Color','r', 'LineWidth', 2)
axis([0 max(dt) min(wind_data) max(wind_data)])
xlabel('Time(s)')
ylabel('Wind Speed(m/s)')
title('Wind Speed Signal')
grid on
% Data Processing
speed = zeros(100,10);
concentration = zeros(100,10);
for i = 1:100
   speed(i,:) = wind_data((i-1)*10+1:i*10);
   concentration(i,:) = concentration_data((i-1)*10+1:i*10);
end
density = 1.2041;
% Calculate Flux
flux = zeros(1,100);
for i = 1:100
    temp = cov(speed(i,:),concentration(i,:));
    flux(i) = density * temp(1,2);
end
time = zeros(1,100);
for i = 1:100
   time(i) = dt((i-1)*10+1);
end
figure
plot(time,abs(flux))
hold on
line([45 45],[0 max(abs(flux))], 'Color','r', 'LineWidth', 2)
line([110 110],[0 max(abs(flux))], 'Color','r', 'LineWidth', 2)
line([170 170],[0 max(abs(flux))], 'Color','r', 'LineWidth', 2)
```

```
axis([0 max(time) min(abs(flux)) max(abs(flux))])
grid on
xlabel('Time(s)')
title('Eddy Flux')
ylabel('Flux(kg/m^2s)')
```

Appendix B – Arduino Code

Measuring Wind Speed Using TOF

```
int trig_pin = 9; // Trigger pin connection

int echo_pin = 10; // Echo pin connection

float t; // Duration of the pulse

float sound_speed = 331 + 0.6 *20;// Speed of the sound

float wind_speed;// Speed of the wind

float distance = 0.6;// length of sound path

void setup() {

  Serial.begin(9600); // Braud rate of 9600

  pinMode(trig_pin,OUTPUT); // Set trigger as an output

  pinMode(echo_pin,INPUT); // Set echo as an input

}

void loop() {

  digitalWrite(trig_pin,LOW);

  delay(200); // Settle the trigger

  digitalWrite(trig_pin,HIGH); // Turn on the trigger

  delay(10); // 10 microsecond pulse

  digitalWrite(trig_pin,LOW); // Turn off the trigger

  t = pulseIn(echo_pin,HIGH);

  wind_speed = sqrt(abs(sound_speed*sound_speed - 2 * distance*sound_speed/(t/1000000)));

  Serial.print(millis());

  Serial.print(',');

  //Serial.println(t); // Record the duration of the pulse
```

```
  Serial.println(wind_speed); // Record the sound speed

  }
```

Raw Data from HC-SR04

```
const byte adcPin = 0;  // A0
const int MAX_RESULTS = 200;
volatile int results [MAX_RESULTS];
volatile int resultNumber;
long current;
long volume=0;
char cstr[1];
// ADC complete ISR
ISR (ADC_vect)
  {
  if (resultNumber >= MAX_RESULTS)
    ADCSRA = 0;  // turn off ADC
  else
    results [resultNumber++] = ADC;
  }  // end of ADC_vect
EMPTY_INTERRUPT (TIMER1_COMPB_vect);
void setup ()
  {
  Serial.begin(2000000); // set baudrate
  Serial.println();
  // reset Timer 1
  TCCR1A  = 0;
  TCCR1B  = 0;
  TCNT1   = 0;
  TCCR1B  = bit (CS11) | bit (WGM12);  // CTC, prescaler of 8
  TIMSK1  = bit (OCIE1B);
  OCR1A   = 39;
  OCR1B   = 39; // 20 uS - sampling frequency 50 kHz
  ADCSRA  =  bit (ADEN) | bit (ADIE) | bit (ADIF); // turn ADC on, want interrupt on
completion
  ADCSRA |= bit (ADPS2);  // Prescaler of 16
  ADMUX   = bit (REFS0) | (adcPin & 7);
  ADCSRB  = bit (ADTS0) | bit (ADTS2);  // Timer/Counter1 Compare Match B
  ADCSRA |= bit (ADATE);   // turn on automatic triggering
}
void intToCharArray(int num, char* charArray) {
  static const char lookupTable[101][3] = {
    "00", "01", "02", "03", "04", "05", "06", "07", "08", "09",
    "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
    "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
    "30", "31", "32", "33", "34", "35", "36", "37", "38", "39",
    "40", "41", "42", "43", "44", "45", "46", "47", "48", "49",
    "50", "51", "52", "53", "54", "55", "56", "57", "58", "59",
```

```
    "60", "61", "62", "63", "64", "65", "66", "67", "68", "69",
    "70", "71", "72", "73", "74", "75", "76", "77", "78", "79",
    "80", "81", "82", "83", "84", "85", "86", "87", "88", "89",
    "90", "91", "92", "93", "94", "95", "96", "97", "98", "99",
    "99", // Entry for any input greater than 99
  };
  memcpy(charArray, lookupTable[num <= 99 ? num : 100], 3);
}
void loop () {
  while (resultNumber < MAX_RESULTS) { }
  /**if (volume==1000)
  {
  volume=0;
  current=micros();
  Serial.print("Timeeeeeeeeeeeeeeeeee(micros):");
  Serial.print(current);
  Serial.print("\n");
  }**/
    for (int i = 0; i < MAX_RESULTS; i++)
  {
    intToCharArray(results[i], cstr);
    Serial.write(cstr,2);
    //Serial.write(itoa(results[i], cstr, 10));
    Serial.write("\n");
  }
  resultNumber = 0; // reset counter
  ADCSRA =  bit (ADEN) | bit (ADIE) | bit (ADIF)| bit (ADPS2) | bit (ADATE); // turn ADC
ON
  volume++ ;
}
```

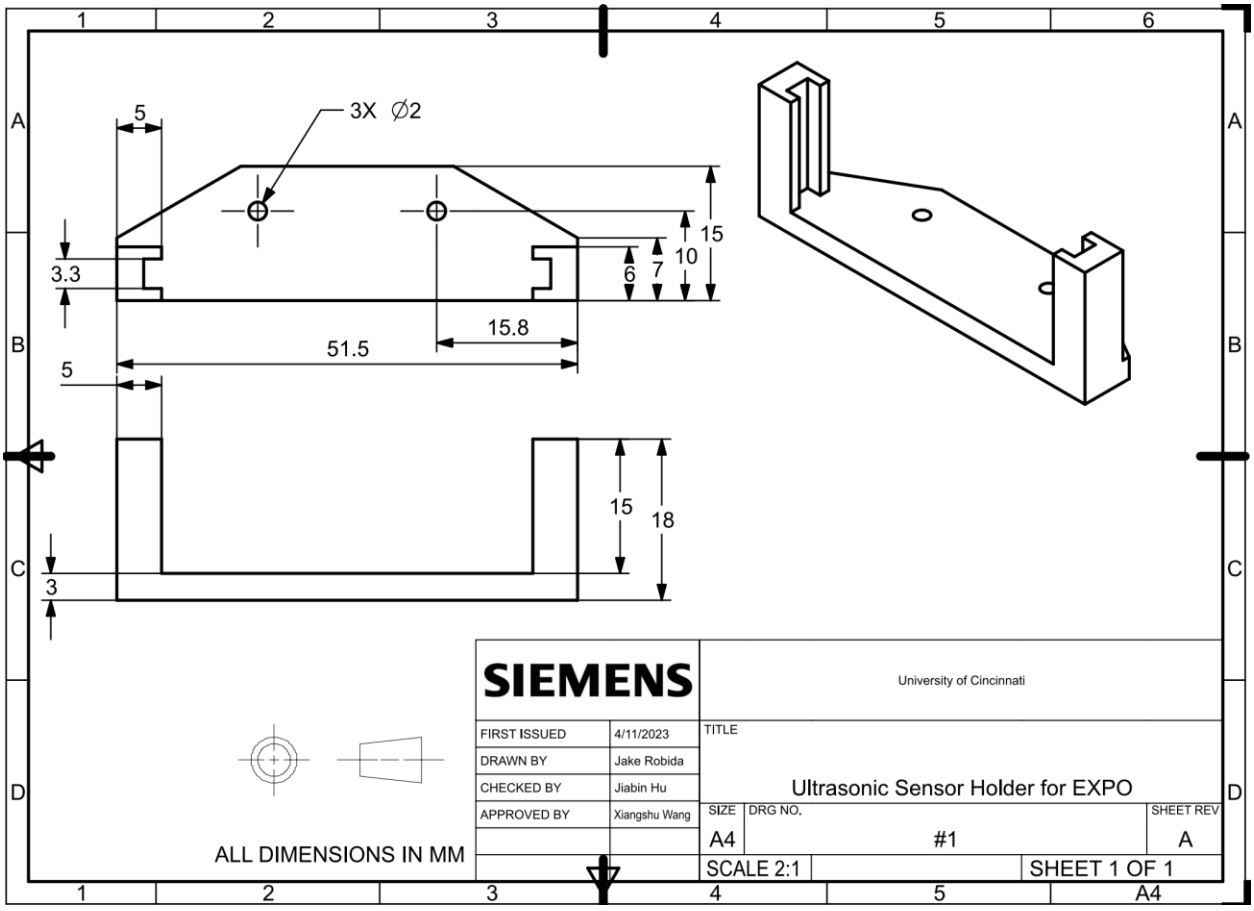
BME280 Temperature, Relative Humidity, and Atmospheric Pressure Measurement

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme;
void setup() {
  Serial.begin(9600);
  if (!bme.begin(0x76)) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
}
void loop() {
  Serial.print(bme.readTemperature());
  Serial.print('\t');
  Serial.print(bme.readPressure());
```

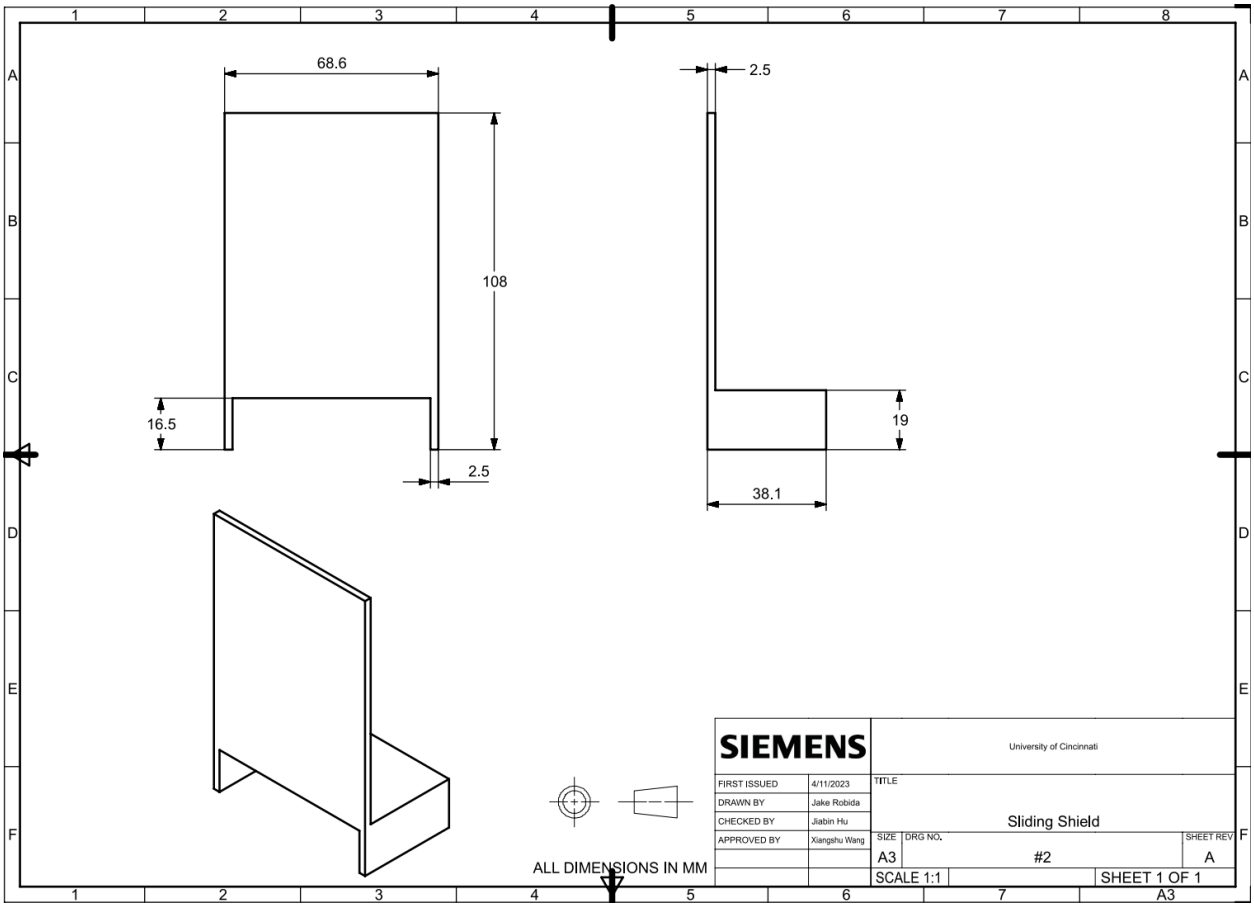
```
  Serial.print('\t');
   Serial.print(bme.readHumidity());
  Serial.print('\t');
  Serial.print(millis());
  Serial.print('\t');
  Serial.println();
  delay(200);
}
```

Appendix C – Engineering Drawings

HC-SR04 Sensor Holder



Reflector


68.6
108
16.5
2.5
2.5
19
38.1
ALL DIMENSIONS IN MM
SIEMENS
University of Cincinnati
FIRST ISSUED 4/11/2023
DRAWN BY Jake Robida
CHECKED BY Jiabin Hu
APPROVED BY Xiangshu Wang
TITLE
Sliding Shield
SIZE A3
DRG NO. #2
SHEET REV A
SCALE 1:1
SHEET 1 OF 1

Appendix D – Bills of Materials

	Product	Company	Price
Microcontroller	Arduino UNO	Arduino	$ 28.50
	Raspberry Pi 4 Model B 2019 (Not used)	Raspberry Pi	$164.99
Sensor	HC-SR04	AOICRIE	$ 1.42
	Humidity Sensor	Adafruit	$ 6.67
Wire	Dupont Cable	CHANZON	$ 7

Appendix E – Project Cost

Product	Cost
3x BME280 Humidity Sensor	$20.47
EXPO Poster Board	$17.24

The Arduino Uno board, breadboard, ultrasonic sensor, cables, and wires were available thanks to Dr. Bellur.