ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

ML-based Weather Forecast on Arduino

Team #26

XUANYU CHEN
(xuanyuc2@illinois.edu)
ZHEYU FU
(zheyufu2@illinois.edu)
ZHENTING QI
(qi11@illinois.edu)
CHENZHI YUAN
(chenzhi2@illinois.edu)

Sponsor: Cristoforo Dimartino

TA: Yi Wang

March 24, 2023

Contents

1 Introduction **1**
1.1 Problem & Solution Overview . . . 1
1.1.1 Problem Statement . . . 1
1.1.2 Solution . . . 1
1.2 Visual Aid . . . 2
1.3 High-level requirements list . . . 2

2 Design **3**
2.1 Block Diagram . . . 3
2.2 Subsystem Overview . . . 4
2.2.1 Software: ML-Based Forecasting Subsystem . . . 4
2.2.2 Hardware: Weather Data Collection Subsystem . . . 8
2.3 Tolerence Analysis . . . 15
2.3.1 Hardware . . . 15
2.3.2 Software . . . 17

3 Cost and Schedule **18**
3.1 Cost . . . 18
3.2 Schedule . . . 19

4 Ethics and Safety **20**
4.1 Ethics . . . 20
4.2 Safety . . . 20

References **21**

1 Introduction

1.1 Problem & Solution Overview

1.1.1 Problem Statement

We develop a weather forecasting system that leverages the power of machine learning to produce accurate and timely weather predictions for our surrounding areas. The accuracy of weather forecasting is critical for making proper plans and preparing for extreme conditions, which is particularly important in areas where traditional weather stations are not available, or their predictions are not reliable due to the distance and altitude of the region. By building our own machine learning-based system, we aim to overcome these limitations and provide more reliable, location-specific weather forecasts for our community.

Our system will analyze vast amounts of data from various sources, including weather sensors and historical weather data, to generate real-time weather predictions. We will also incorporate advanced techniques such as pattern recognition and artificial neural networks to create a comprehensive and accurate weather forecasting model. The system will continuously learn from the new data to improve the accuracy of the predictions, making it a self-adaptive system that can handle changes in weather patterns over time.

We envision our system to be an accessible and user-friendly tool that can be easily accessed by anyone. The system will provide weather predictions for specific locations, which can be tailored to the needs of individuals or businesses in those areas. Additionally, we hope to develop a web application that will allow users to access weather forecasts on the go, giving them the ability to make informed decisions based on the most up-to-date weather information.

1.1.2 Solution

A weather forecast system can be created by using a few different hardware components and software tools. Our solution mainly consists of two parts: weather stations and forecasts. As to the weather measurement and data collection part, temperature, humidity, barometric pressure, and rain sensors are considered the main components. For the other part, a Machine Learning (ML)-based algorithm is to be applied for data analysis and weather predictions. Also, if time permits, we plan to visualize our timely weather prediction on a website based on Flask or other web application frameworks.

1. Hardware Subsystem
   Due to the complexity of weather conditions, our system incorporates the following weather indicators and their corresponding collectors: a humidity and temperature sensor, a barometric pressure sensor, a rain sensor, a light sensor, and an anemometer for wind speed. The aforementioned equipment will be integrated into Arduino, covered with a waterproof enclosure. The power supply for the weather station consists of batteries and a solar panel charging circuit. The outdoor weather data

collector will send to an indoor Arduino receiver with a display screen via wireless communication. Then, the receiver will send the collected data to the computer via the serial interface.

2. Software Subsystem
   A practically usable weather forecast system is supposed to make reliable predictions for real-world multi-variable weather conditions. We apply Machine Learning techniques to suffice such generalization to unseen data. To this end, a high-quality dataset for training and evaluating the Machine Learning model is required, and a specially designed Machine Learning model would be developed on such a dataset. After a preliminary investigation, we have downloaded Haining's weather datasets for the most recent 40 years from the OpenWeather platform. And we believe auto-regressive models could serve as practical solutions for our model. Once a well-trained machine-learning model is obtained, we will deploy the model on portable devices with easy-to-use APIs.

1.2 Visual Aid

The entire system basically looks like the picture above (1).

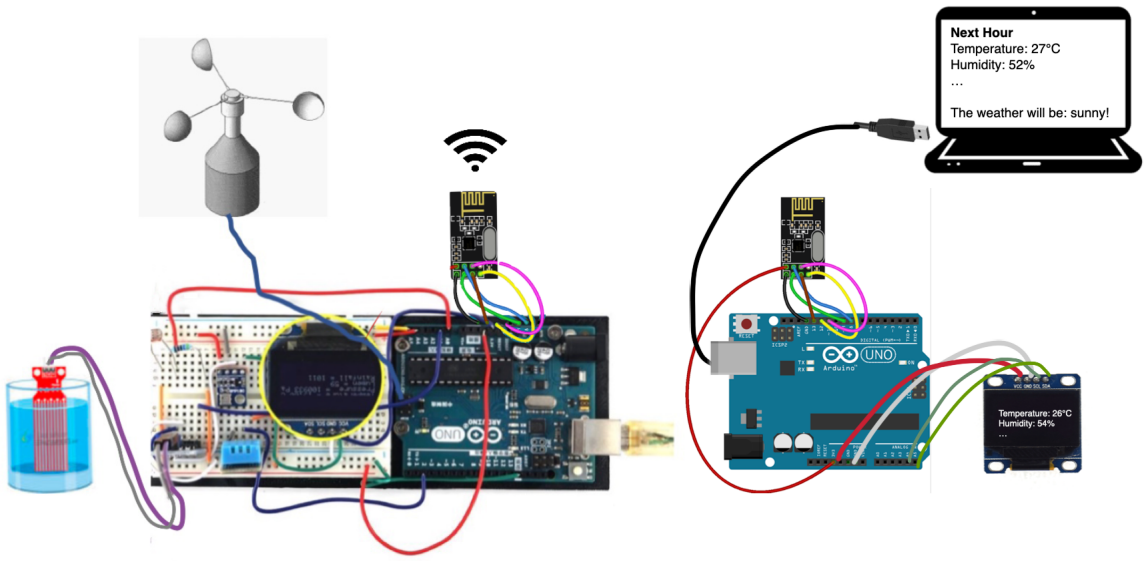

Figure 1: The entire system looks like this.

1.3 High-level requirements list

- The weather measurement prototype with sensors should be able to accurately collect the temperature, humidity, and barometric pressure. etc. To be more specific, our collected weather parameters will be compared to real-time data from a meteorological station and should be within a reasonable margin of error. For example,

temperature error should be within 5%, humidity error, and barometric error within 10%.

- A machine learning algorithm should be successfully trained to make predictions on the weather conditions: rainy, sunny, thunderstorm, etc. This is a multi-classification problem and we hope our macro-average precision could reach 0.75 or higher.
- Our system can collect and forecast the weather in Haining, in real-time, and/or longer-period forecast. To be specific, our hardware subsystem could collect and upload real-time weather information to the software subsystem every 30 minutes. And our software subsystem could predict the following 24 hours' weather conditions based on the recently inputted 48 hours' weather parameters. For longer-period prediction, our system will be able to predict the weather up to 7 days later, but it could be a great challenge to predict accurately in this case as you can imagine.
- The forecast weather information could be demonstrated elegantly through some UI interface. A display screen would be a baseline, and a web application on a phone or PC would be extra credit if time permitted.

2 Design

2.1 Block Diagram

Figure 2 illustrates our entire design.

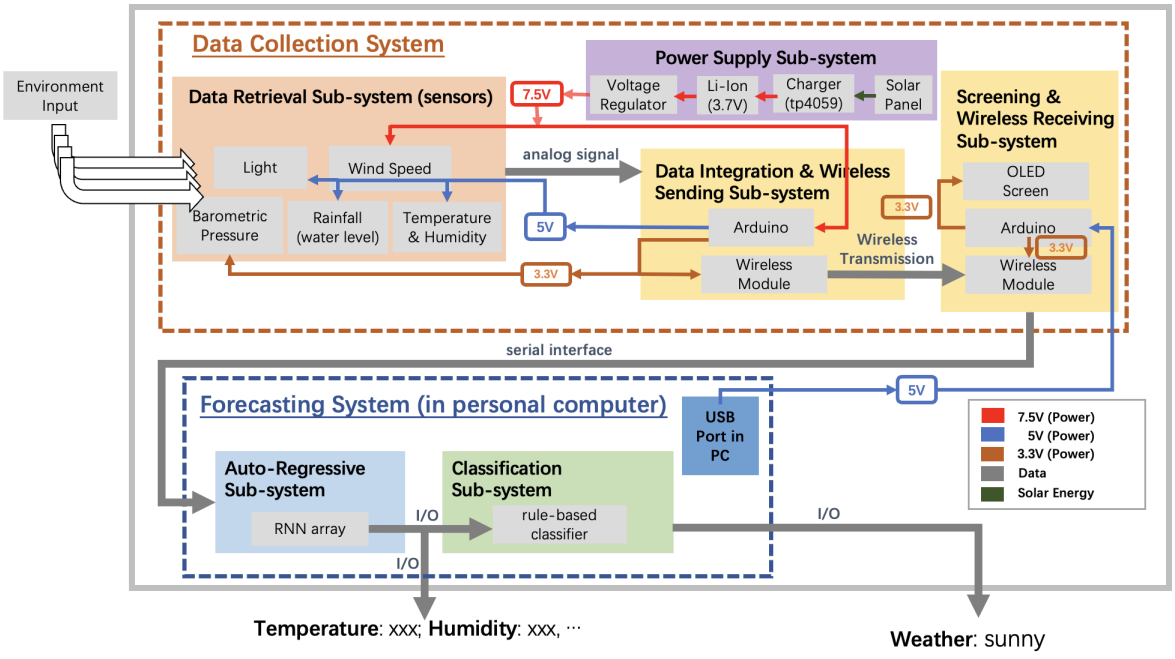

Figure 2: Block Diagram.

Our design consists of two parts: a data collection system and forecasting The power supply module consists of a lithium-ion battery with a boost converter, and a solar-powered battery charger. Five different sensors are used to retrieve environmental data, and those data will be transmitted to our data integration sub-system through an analog signal. We use Arduino to integrate those incoming data and then transmit them via a wireless communication module to the indoor Arduino receiver. Finally, the receiver uses a serial interface to send those data to a personal computer for further training and forecasting. As for our forecasting system, we leverage an RNN-based model to process time series and make predictions, and a final weather condition would be determined by a rule-based classifier given the predictions. Details are shown in subsequent sections.

2.2 Subsystem Overview

Our weather forecast system mainly consists of two parts: software for the Machine Learning training model and hardware for the weather station.

2.2.1 Software: ML-Based Forecasting Subsystem

Deep Learning (DL) models, an important branch of ML models, have shown their extraordinary ability to discover complex patterns and features of various types of data. In various families of DL methods, auto-regressive models are specifically designed for processing and generating time series. Since our algorithm's goal is to predict the following values in a sequence of values after taking in the current ones, we leverage auto-regressive models' ability to achieve our goal.

LSTM: Long Short-Term Memory We choose LSTM [1] as our base model. LSTM is a type of recurrent neural network (RNN) that is commonly used for sequence prediction tasks including time series forecasting, speech recognition, and natural language processing. It is particularly useful when dealing with sequences of variable length and long-term dependencies between data points. In the context of predicting values in a sequence of weather data, LSTM can be used to learn the underlying patterns in such data and make accurate predictions based on the current input.

The basic idea behind LSTM is to use a memory cell (Figure 3) to store information from previous time steps and selectively forget or remember certain information based on the current input. The memory cell consists of three gates: the input gate, the forget gate, and the output gate. The input gate determines how much new information is added to the memory cell, while the forget gate determines how much old information is retained. The output gate controls how much information is used to make the prediction.

Training LSTM To use LSTM for predicting, we first need to train the model on a dataset of input-output weather data pairs. The forward propagation of a single LSTM

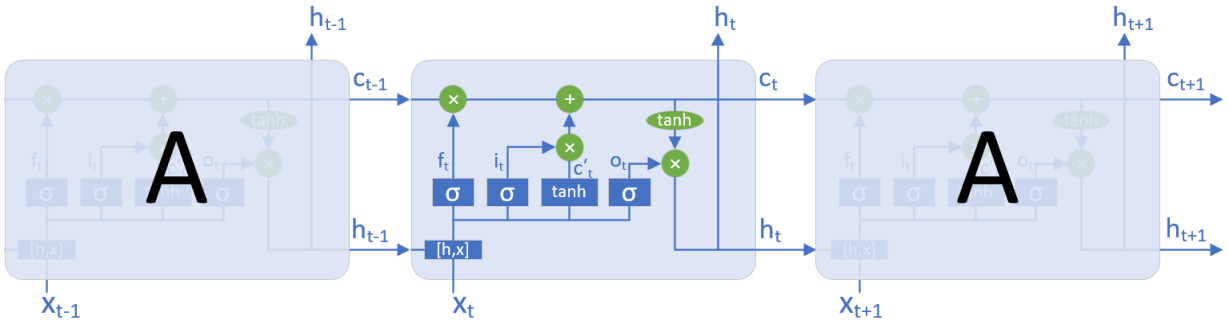

Figure 3: Illustration of an LSTM cell.

cell at time t is formulated as follows:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{1}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2}$$
$$g_t = tanh(W_{ig}x_t + b_{io} + W_{hg}h_{t-1} + b_{hg}) \tag{3}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$
$$h_t = o_t \odot tanh(c_t) \tag{6}$$

Where x_t is the features of interest (including temperature, pressure, humidity, wind speed, rainfall, and month) that have been normalized, and h_t is the embedding calculated for x_t. LSTM cells are concatenated together into an encoder, after which an MLP (Multi-Layer Perceptron) is attached to map the hidden states into expected data. The LSTM and the MLP constitute a forecaster, which is trained with an MSE loss function and is expected to predict one future value based on existing ones (in an auto-regressive manner). The entire training procedure is illustrated in Figure 4, and is formulated in Algorithm 1.

During training, the LSTM learns to recognize patterns in the data and adjust its parameters to minimize the difference between its predictions and the actual output values. Once the model is trained, we can use it to predict the next values of some weather conditions of interest based on the current input in real-world settings.

Rule-based Classifier A rule-based classifier analyzes data based on a set of pre-defined rules. In the context of weather forecasting, this means creating a set of rules that dictate how different weather variables should be interpreted and combined to arrive at an overall description of the weather.

To create a rule-based classifier for weather forecasting, we would first need to identify the variables that are most relevant for predicting the overall weather conditions. This might include variables such as temperature, precipitation, wind speed, humidity, and more.

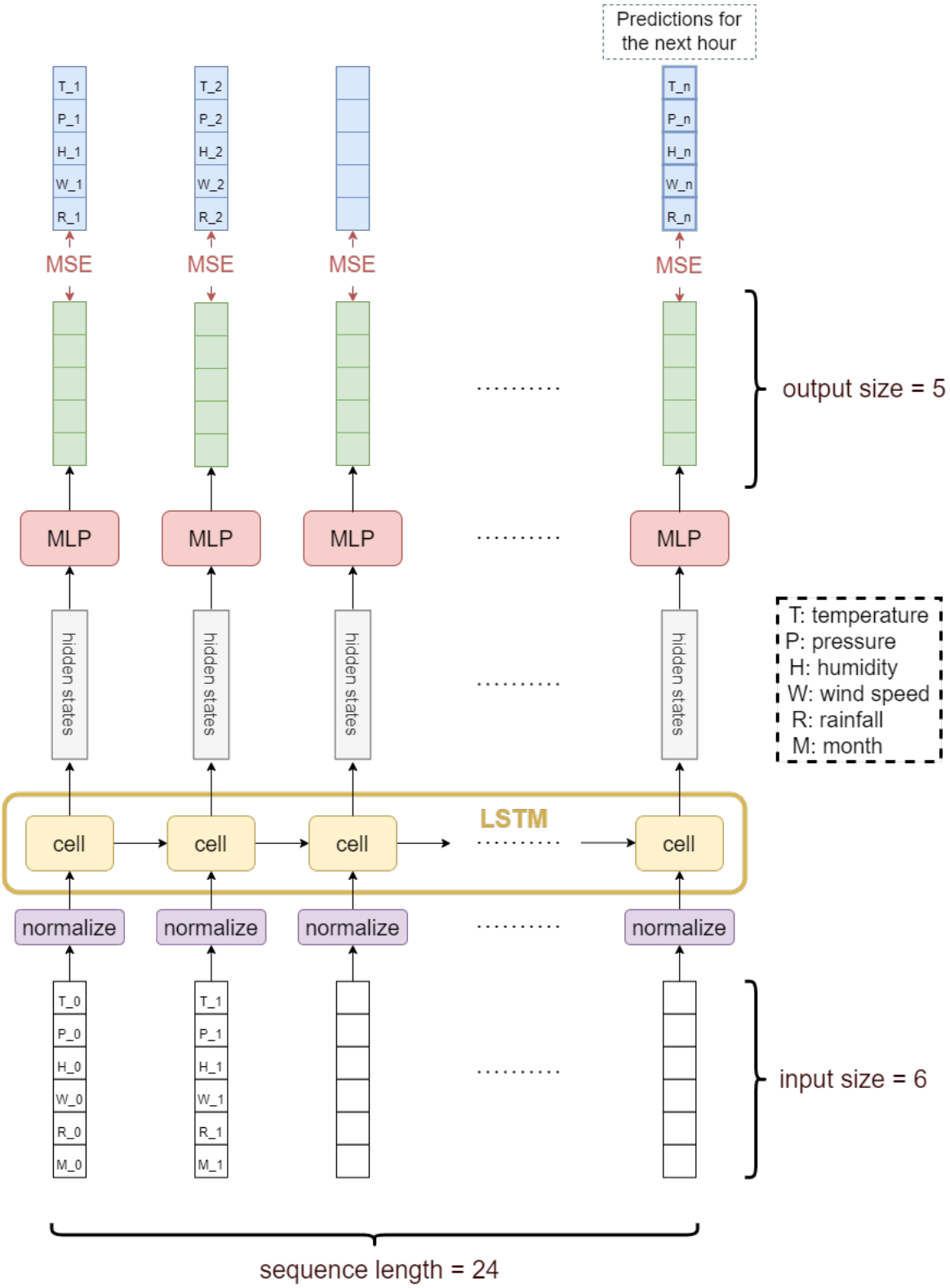

Figure 4: Illustration of the training procedure.

Algorithm 1 Training a Forecasting Model.

Require: T: number of epochs; S: training dataset; f: LSTM; g: MLP; D: output size
Ensure: Trained model $(f \cdot g)(\cdot)$
  Initialize f and g.
  for epoch t from 0 to T-1 **do**
    for batch b in S **do**
      get sequence data x and label y.
      get LSTM output: $h \leftarrow f(x)$.
      get MLP output: $\hat{y} \leftarrow g(h)$.
      compute MSE loss: $l \leftarrow \frac{1}{D}\sum_{i=0}^{D-1}(y_i - \hat{y}_i)^2$.
      backward propagation from l.
    end for
  end for

Once we have identified the relevant variables, we would need to establish rules for how these variables should be combined to arrive at a weather description. For example, we might manually set a rule that if the temperature is above a certain threshold and the precipitation is below a certain threshold, the weather is classified as sunny. Or, we can train a decision tree to make these rules automatically.

As we establish more rules and combinations of variables, the accuracy of the classifier should improve, allowing us to provide more accurate and detailed descriptions of the weather. As for why we do not use a Deep Learning model like MLP for such a purpose, the main reason is that such models are hard to interpret and understand. With a rule-based classifier, it is easy to see exactly which rules are being applied and how they are being combined to make a prediction. With a Deep Learning model, however, it can be more difficult to understand how the model is making its predictions and what factors are most important for determining the overall description of current weather conditions.

Requirements	**Verification**
Predicted values for each weather indicator should deviate from true values in an acceptable range. Data predicted would be compared with 1) previously measured data and 2) online weather forecasts. We expect a 10% and 50% relative difference from online values and the last measured values, respectively, for all five parameters.	The predicted value $\hat{y}$ and the true value y (measured at the next time step) for the next time step should satisfy $\frac{\lvert y-\hat{y}\rvert}{\lvert y\rvert} \leq 10\%$ (after normalization). The adjacent predictions should satisfy $\frac{\lvert y_t-y_{t-1}\rvert}{\lvert y_{t-1}\rvert} \leq 50\%$. If this threshold has been exceeded for a certain amount of time steps, we would be informed by the system, and some improvements may be needed.

2.2.2 Hardware: Weather Data Collection Subsystem

Our weather data collection subsystem is used for collecting real-time weather data and transmitting them to our software subsystem for prediction. This subsystem will be deployed on Arduino Uno, a microcontroller board based on the ATmega328P. Applied sensors include a temperature and humidity sensor, a barometric pressure sensor, a raindrop sensor, a light sensor and an anemometer. An indoor OLED screen will be used to display real-time collected data, which are used for weather prediction in the next step. Also, we designed a solar-assisted power supply module, so that our system could utilize solar energy to power itself and also make its battery endurance enhanced.

Sensing module We must appropriately choose the kinds of sensors we use so that the measured types of weather parameters could be consistent with those types in our datasets for training ML models. Also, we should make sure those selected data types are comprehensive enough to be generalized to an output conclusion of weather. After a thorough discussion, we decided to use a temperature and humidity sensor, a barometric pressure sensor, a water level sensor, a light sensor, and an anemometer for sensing. Layout design is included in figure 9 and 8. Detailed decisions for sensors selection are as follows:

Temperature and humidity are definitely the most important weather parameters. We choose to use DHT11 module [2] because its sensing range perfectly matches the weather condition in our measured area, Haining. Its humidity range is 20-90$\%$ RH with $\pm 5\%$ RH accuracy, and its temperature range is 0-50 $^{\circ}C$ with $\pm 2^{\circ}C$ accuracy. It's a low-cost, long-term stable combined sensor that allows long-distance signal transmission and precise calibration.

For barometric pressure, we choose to use BMP 180 module [3] to sense because it is able to precisely detect the changes in real-time pressure, which contributes to the high sensibility of our data collection system. Also, the chip consumes less than 1mA during measurements and only 5μA when idle, its low power consumption makes it a good choice for our system. Moreover, we choose [4] as our Light Sensor and [5] as our Anemometer Sensor.

To measure the rainfall per hour, we design a small-sized rain barrel 6 to collect the raindrops which is further used [6] for sensing the accumulated water level. According to our dataset, the rain volume in our measured area usually falls into the range $[0, 2mm]$. The maximum rate ever detected was around 20mm. Given that the water level sensor is able to measure at most 40mm height, our rain barrel is estimated to function without manual intervention for at least 4 hours during extreme rains. When the water level rises above 35mm, a warning is raised on the display screen of the indoor receiver.

Requirements	Verification
1. The sensor subsystem should be able to measure and output accurate data. To ensure these components function correctly, we have to support a suitable voltage source. The 5V logic microcontroller, Arduino Uno will act as the main voltage source. Specifically, the voltage applied to the anemometer should be above 7V.	1. Measure its output voltage using an oscilloscope, ensuring that the output voltage stays stable within 5% of 5V. As to the anemometer, use a multimeter to measure the output voltage of the DC-to-DC Boost Converter module, connect the battery first to the input and gradually rotate the potentiometer until it shows 7.5V on the multimeter.
2. The temperature and humidity sensor should be able to send out an 80μs-long low-voltage-level response signal to indicate data preparation and then pull up the voltage.	2. Measure the voltage level with an oscilloscope, if the signal is always at a high-voltage level, it suggests that the module isn't responding properly and then we need to recheck the connection.
3. Our collected data should reach certain accuracy. Data readings will be compared to the weather station's reports in our measured area. We expect a 5% relative difference in temperature, and 10% in other parameters.	3A. Adjust our data readings to remove the effect of altitude. For example, as to the barometric pressure values, we have to adjust the data to make it appear that the measurement was taken from sea level. Adjustments can be made by libraries, such as the function *sealevel(P, A)* in the Sparkfun library. 3B. Calculate the relative difference between the adjusted data and that from the weather station.

Data transmission module Weather data collected by sensors needs to be transmitted to our personal computers for machine-learning weather prediction. Since our sensors will be placed in an outdoor environment that probably gets caught in rain, we decide to use wireless communication (not traditional wire connection)to let the data come indoors. Also, we would like our sensing information could be demonstrated through an OLED screen real-time. As a consequence, we design our data transmission module to be two sub-modules.

- **Wireless transmission sub-module using transceiver**: As Figure 8 demonstrates, we design to use NRF24L01, an SPI-connected digital transceiver, to wirelessly transmit weather data from the left Arduino which is placed outdoors with weather sensors, to the right Arduino which is placed indoors near our personal computer.

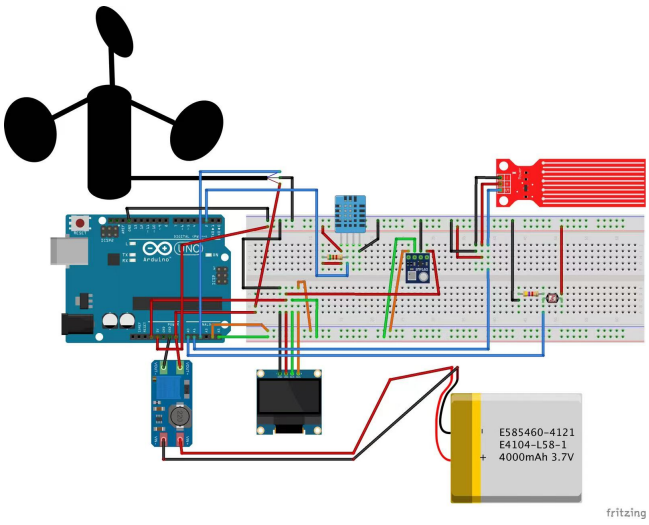

Figure 5: Interfacing of sensors with Arduino Uno

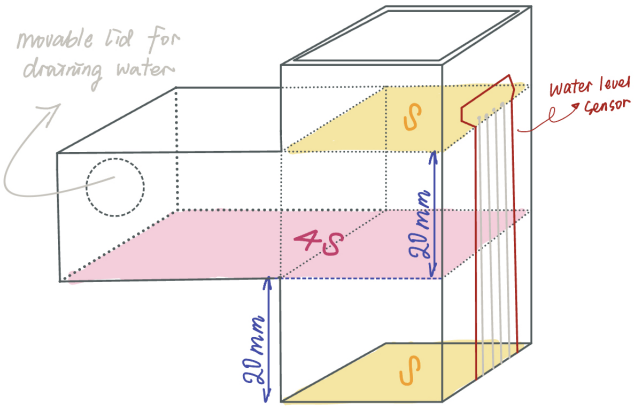

Figure 6: Physical sketch of the rain barrel

- **Wired transmission sub-module using serial port**: We use the serial port to transmit data from the right Arduino in Figure 8, to our personal computer. Also, we will display all of sensed weather data on an OLED screen installed on the right Arduino.

You might be curious why don't we directly transmit data from the sensing-part Arduino to our personal computer. Well, we do consider this plan before, however, this plan requires an additional router and other components connecting with the sensing-part Arduino, which is uneconomical and overly complex for this project. More importantly, this cause our OLED screen to have nowhere to place (installing it outdoors is meaningless), while two-stage transmission allows weather data to pass through to be displayed in the OLED screen installed in the indoor Arduino, which is easy and useful for us to observe the weather data in real-time. The workflow of the data collection and transmission process is shown in figure 7.

Requirements	Verification
1. Our whole transmission module should be able to transmit the data correctly without any data loss with a wireless transmission distance to be at least $6m$.	1. Keep the outdoor Arduino 6m away from the indoor Arduino. Then we can verify the requirement by checking the received data on our personal computer (of course it builds on that sensing module successfully achieves its requirements). If all data types are received and the real-time data keeps coming into our computer, then this requirement is achieved.
2. Our OLED screen installed on the indoor Arduino could demonstrate the real-time weather data successfully.	2. Check with eyes to see whether the OLED screen demonstrates all kinds of weather data types. If all data types are received and the real-time data keeps coming into our OLED screen, then this requirement is achieved.

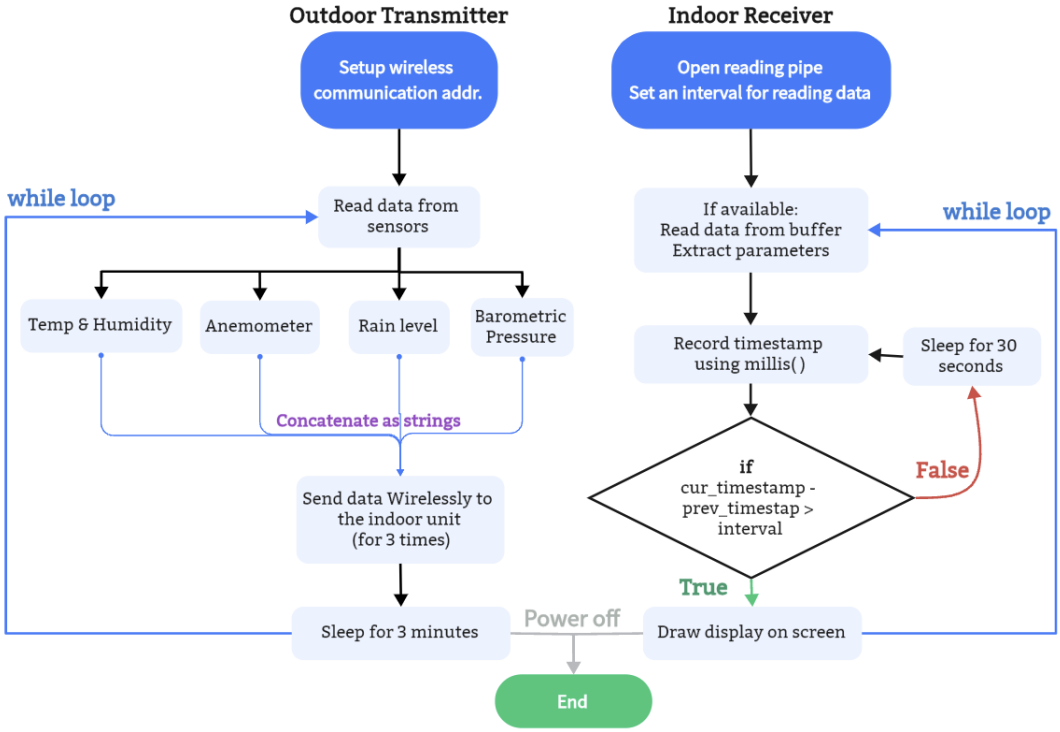

Figure 7: Workflow of weather data collection and transmission

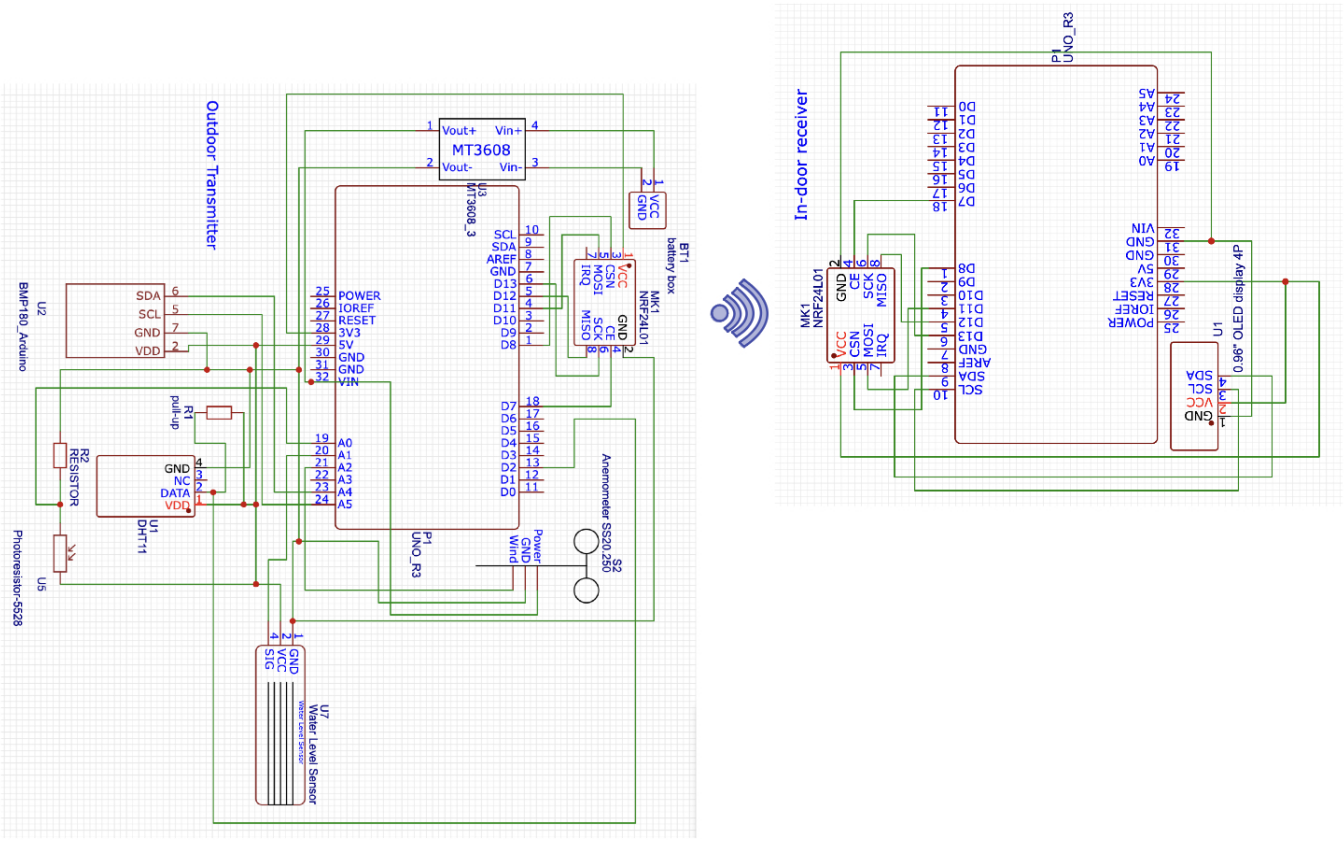

Figure 8: Wireless communication schematic

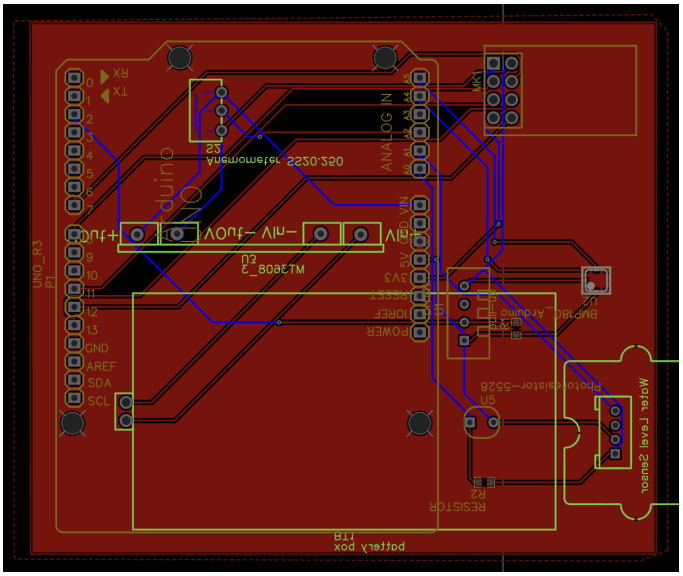

Figure 9: PCB layout of the outdoor transmitter

Note: *The water level sensor should be placed outside of the shell 6. It doesn't lie on the PCB board, thus it is partly included in figure 9.*

Power Supply Module

For our outdoor weather station, the power source of the Arduino Board is 2 * 3.7V 4000mAh batteries with a DC-to-DC Converter Module to boost the voltage supply from the battery. Arduino will then provides three voltage levels, including 3.3V for the water-level sensor, 7.5V (VIN) for the anemometer, and 5V for the remaining components. Battery-power supply is shown at the top left of figure 9. For the indoor receiver, Arduino will be charged with the USB cable.

To ensure that our system won't suffer from failure because of a dead battery, a solar panel charging system is designed to ensure the sensor module with the Arduino is powered continuously. The circuit is shown in figure 10. The Lithium-Ion battery will be charged through a charger, the TP4059, powered by the solar panel. The charger is chosen for its affordability and thermal feedback regulations. In our schematic, while the battery is being charged, the red LED will be on. On the contrary, the green LED indicates fully charged.

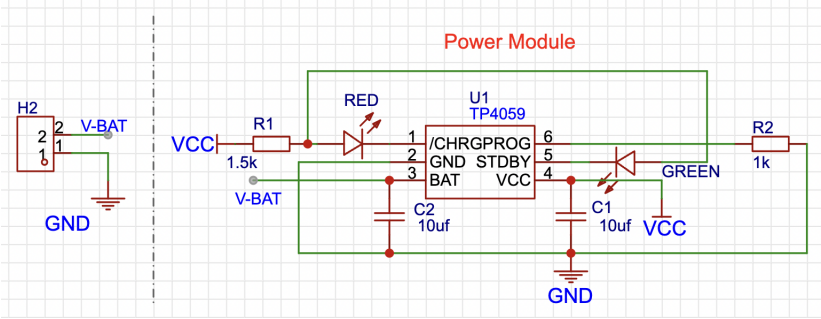

Figure 10: Power supply schematic

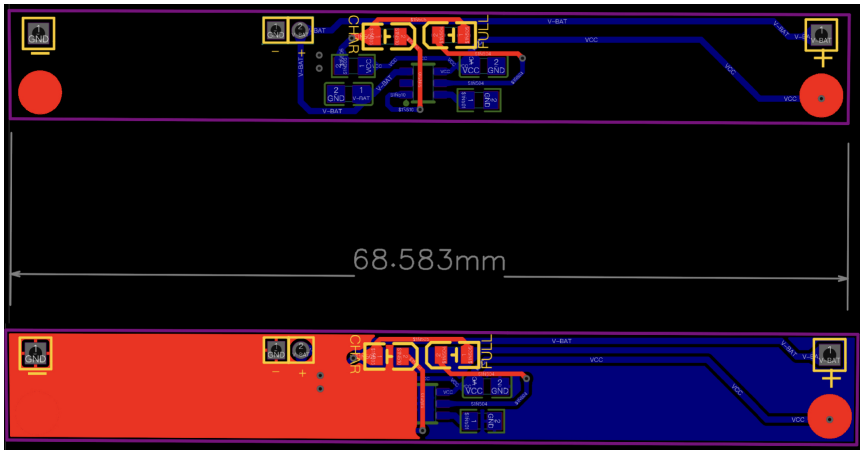

Figure 11: PCB layout of the power supply module

Requirements	Verification
1. The battery-power supply for the sensing module should provide over 3V for at least 24-hour duration.	1A. The battery should have sufficient capacity. According to our power analysis shown in table 1, the battery capacity should be at least $\frac{1000mW*24h}{3V*2} = 4000mAh$. Our current selection is a couple of 3.5V 4000mAh Li-ion batteries. It meets the requirement theoretically. 1B. And we will further verify the power usage through experiments under different weather conditions. Our plan is to discharge the battery at 400mA for 10 hours.
2. The solar charging circuit should supply at least 4.2V to charge the battery.	2A. According to our survey, the 5V solar panel we used is estimated to provide a 4.2V voltage supply on cloudy days, which should be enough. 2B. We will discharge the battery to 3V first and then measures the charging voltage and current under different weather conditions. If the voltage supply is not enough, we will connect the same solar panels together in parallel or switch to larger-area panels.

Water-proof Enclosure

To protect the circuit, we will also design a highly waterproof shell for our hardware subsystem. Due to such consideration, the PCBs would be put above the bottom of the outfit. For requiring measuring Temperature, Air Pressure, Light, Raindrops, Humidity, Wind speed, and direction, we need to carefully consider the distribution of the sensors and PCBs. As for the shell must be placed under sunlight, the shell should have considerable UV resistance. After taking Manufacturing costs into consideration, we think ABS/PETG plastic is the best choice. The outfit layout is shown in Figure 12.

Requirements	Verification
The enclosure of the box should totally prevent the dust and water jet into the outfit. Taking the IP waterproof as a reference, our outfit enclosure should at least reach the IP65 waterproof. This IP65 waterproof means no ingress of dust whatsoever permitted, and no ingress from precipitation permitted.	To reach such a waterproof level, we decided to use a rubber sealing strip in the connecting area between the cap and bottom. And for the holes on the side faces where wires go through, we would use epoxy resin to occupy the space. In our assumed test, water projected by a nozzle (6.3 mm) against the enclosure from all directions shall have no harmful effects. This would verify its waterproofing ability.

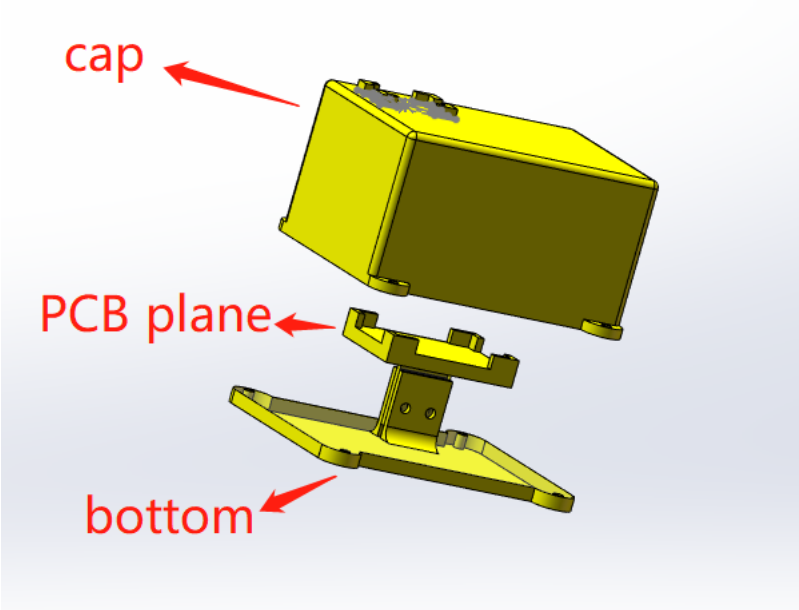

Figure 12: CAD model of the enclosure

2.3 Tolerence Analysis

2.3.1 Hardware

Power Supply Analysis for sensing subsystem We roughly estimate the working power for our remotely placed sensing subsystem as the following table states. It is essential to do the estimation since our sensing subsystem will be placed outside in an outdoor environment and will be powered by our self-designed power system.

Components	**Power**
Arduino Uno	Power: $\approx 450mW$ from Youtube [7]
Wind speed sensor	Power: $\leq 300mW$ from its data sheet [5]
Water level sensor	Working Voltage: $5V$ Working Current: $\leq 20mA$ Power: $\approx 100mW$
DTH11: Temperature and Humidity Module	Working Voltage: $5V$ Working Current: $0.3mA$ when measuring, $60\mu A$ when standby Power: $\approx 0.3mW$
BMP180 Digital pressure sensor	Working Voltage: $3.3V$ Working Current: $5\mu A$ Power: $\approx 0.02mW$
Light detector sensor	Working Voltage: $5V$ Working Current: $\leq 5\mu A$ Power: $\approx 0.03mW$
nRF24L01 Wireless Module	Working Voltage: $3.3V$ Working Current: $\leq 13.5\mu A$ Power: $\approx 44.55mW$
Other auxiliary resistors	Power: $\leq 100mW$

Table 1: Power Usage Analysis

As shown in table 1, the total power consumption is estimated to be 1W. Our battery component provides a 3.7V voltage supply and has $4000 * 2 = 8000mAh$, thus providing $3.7V * 8000mAh * 3600s = 106560J$. Our battery power supply can work for at most $\frac{106560J}{1W} = 29.6h$, which will be sufficient for our basic goal of a whole-day usage. Moreover, our chosen solar panel can supply at most 1.1A under the sun. Assume that sunlight is available for 4 hours on average (the assumption is reasonable referring to Haining's weather dataset). We are able to refill the battery $\frac{4000mAh*2}{1.1A*2} = 3.63$ with double-panel once a day at the best case. Based on the aforementioned estimation, our power supply system is able to produce adequate energy for our weather station.

2.3.2 Software

In the case of the forecasting model, potential problems may arise due to a lack of accuracy or precision in the data inputs.

Denote the model as a function f. Given weather data sequences $S = \{X_{temp}, X_{humidity}, ...\}$, where $X = [x_t, x_{t+1}, ..., x_{t+n-1}]$ of length n at time step t, the model predicts the next-step values $Y = \{y_{temp}, y_{humidity}, ...\}$ by $Y = f(S)$. Therefore, each predicted value y is contingent on all five types of measured data (temperature, humidity, pressure, rainfall, and wind speed). Since we use MSE (mean square error) to evaluate the prediction quality, which is calculated by $\mathbf{MSE}(Y, \hat{Y}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$, the error caused by wrongly measured data would be of **second order**. So the most important tolerance in the software part comes from the robustness of the model against the incorrectness of measured data.

If we do nothing to the input data, the error would be magnified to a greater extent due to the difference in units. If both deviate the same percentage from actual values, the error caused by pressure (of order 1×10^3) would be $10^{(3-1)*2}$ times as much as the temperature (of order 1×10^1). We leverage **normalization** on the input data to erase the dimensional differences (e.g. $19°$C of temperature and 1031 hPa of pressure) so that each weather feature contributes to the prediction results to the same extent. The L_p normalization for an input vector v scales the vector into range $[0, 1]$, which is calculated by:

$$v = \frac{v}{\mathbf{max}(||v||_p, \epsilon)},$$

where ϵ is a small value to avoid division by zero. Since all features are mapped to the same range, the difference in units would not be a problem anymore.

Besides, we continuously monitor and validate the model's performance and recalibrate it according to previously learned data patterns as necessary. We set the threshold of predicted values' deviation from the real values (measured in the next time step) to be **10%**. That is, the predicted value for the next time step $\hat{y}$ should satisfy $\frac{|y-\hat{y}|}{|y|} \leq 10\%$ (after normalization). If this threshold has been exceeded for certain times, we would be informed by the system and improve the model design. Incorporating more data sources or modifying the model's architecture may also be necessary to improve its accuracy and robustness.

Overall, anticipating and addressing potential failures in both hardware and software components is crucial to ensuring the reliability and accuracy of the weather forecasting system. It requires a comprehensive approach that combines proper maintenance, monitoring, redundancy measures, and continuous improvement to ensure optimal performance.

3 Cost and Schedule

3.1 Cost

Category	Item	Price
Board	Arduino Uno *2	¥ 300
Display	0.96 OLED screen	¥ 15
Sensors	Temperature & Humidity Sensor (DHT11)	¥ 10
	Barometric Pressure Sensor (BMP180)	¥ 5
	Water level sensor	¥ 3
	Adafruit/DFRobot Anemometer	¥ 200-240
	Light sensor module	¥ 5
Wireless Communication	nRF24L01 * 2	¥ 10
Power Supply	3.7V Battery *2	¥ 50
	Solar panel *2	¥ 50
	TP4059 Charger	¥ 5
PCB	/	¥ 50
GPU	NVIDIA RTX A5000 * 1	¥ 200
3D Printing Material	Enclosure	¥ 100
Weather Station Holder	/	¥ 50
Auxiliary Components	/	¥ 100
Labor Cost	¥$150 * 100 * 4$	¥ 60000
Total		¥ 1200 + *60000 (for labor)*

3.2 Schedule

Date	Zhenting Qi	Xuanyu Chen	Zheyu Fu	Chenzhi Yuan
03/20/23	Complete the data preprocessing code.	Complete power supply and wireless communication module schematic & PCB design	Verify sensor selection and complete sensor module PCB design	Design a rough outfit(v1) for the sensors' circuit. Decide the material used in the outfit printing.
03/27/23	Complete the weather forecasting ML model.	Arduino coding for sensors and consult on version 2 sensor schematic	Assemble sensor module (except Anemometer & rain barrel) and complete functionality testings.	Design a rough outfit(v1) for the sensors' circuit.
04/03/23	Complete the code for training and evaluating weather forecasting ML model.	Arduino coding for wireless module and experiments of data transmission.	Assemble power supply module and experiments of functioning duration.	Discuss the waterproofing methods. Add the holes for wire connection on CADs. Verify the Temperature resistance.
04/10/23	Debug all the code and integrate them into a first-version implementation.	Finalize sensor module PCB design including rain barrel and anemometer.	Finalize power and wireless module.	Print the outfit(v2). Modify the outfit CAD model(v3). Verify the UV resistance.
04/17/23	Conduct extensive experiments and analyze the results.	OLED screen display.	Webpage basic design.	Verify the waterproofing method on outfit(v3). Improve the waterproofing method.
04/24/23	Improve the model architecture and implement a last-version system.	Conduct experimental testing.	Complete data visualization. 2.	Check the PCB waterproofing methods. Verify the final version of the outfit and PCBs assembly.
05/01/23	Prepare final demo and design demo testing cases.	Prepare final demo and design demo testing cases.	Prepare final presentation.	Final check for the assembly.
05/08/23	Mock Demo			

4 Ethics and Safety

4.1 Ethics

According to the IEEE Code of Ethics 1 [8], we should hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices. Following this, We are committed to designing our weather forecast equipment to be both practical and ethical. We hope our design could be a next-generation solution for a mini-size meteorological station, and we will try to prevent any form of behavior that may endanger public order with our products.

Also, the IEEE Code of Ethics states that engineers shall maintain confidentiality and protect the privacy of others. We ensure that we use legally available datasets, like Haining's weather conditions from the OpenWeather platform, for training our machine learning models. To avoid ethical breaches, we may implement appropriate measures to safeguard the privacy of data, such as encryption and secure storage.

Moreover, In line with the IEEE Code of Ethics 4 [8], it is important to actively seek, accept, and provide sincere critiques of technical work while acknowledging and rectifying any mistakes. We should be truthful and practical when making statements or predictions based on available data, and give appropriate recognition to the contributions of others.

4.2 Safety

According to the safety guidelines from the ECE445 course website [9], our team will abide by the rules below:

1. Minimum of two people must be present at the lab at all times to ensure our safety. Especially when doing some procedures with electricity, we should pay extra attention to the safety of ourselves as well as our teammates. Also, when we are doing soldering, material cutting, and 3d printing, we should prevent ourselves from being burned or cut.

2. Everyone in our team will complete the mandatory online safety training, and submit our certificates of completion on Blackboard. This is to ensure our team has the necessary common sense for laboratory safety.

3. On the whole, our project does not involve many dangerous operations. However, we should always be careful and cautious to ensure that our safety and health will not be affected.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Electronics. ""Temperature and Humidity Module DHT11 Product Manual"." (2014), [Online]. Available: https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf (visited on 03/07/2023).

[3] B. Sensortec. ""BMP180 Digital pressure sensor Datasheet"." (2013), [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf (visited on 03/07/2023).

[4] S. H. S. Co. ""Ambient light detector Photosensitive sensor"." (2014), [Online]. Available: https://www.arduino.cc/documents/datasheets/HW5P-1.pdf (visited on 03/07/2023).

[5] Q. Electronics. ""Anemometer Wind Speed Sensor Datasheet (in Chinese)"." (), [Online]. Available: https://cdn-shop.adafruit.com/product-files/1733/C2192_datasheet.pdf (visited on 03/07/2023).

[6] K. Robots. ""Water Sensor Module User's Manual"." (), [Online]. Available: https://curtocircuito.com.br/datasheet/sensor/nivel_de_agua_analogico.pdf (visited on 03/23/2023).

[7] M. Klements. ""How Long Can An Arduino Run On Batteries? I Tested 6 Of The Most Common Boards"." (), [Online]. Available: https://www.youtube.com/watch?v=5cYN5-Spnos (visited on 03/24/2023).

[8] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/07/2023).

[9] ECE470. ""ECE470 Safety Guidelines"." (2023), [Online]. Available: https://courses.grainger.illinois.edu/ece445zjui/guidelines/safety.asp (visited on 03/07/2023).