DISTRIBUTED ENVIRONMENTAL SENSING SYSTEM FOR THE JOHNSON MUSEUM

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Mingyang Feng (mf783), Yingjia Zhang (yz2723)

MEng Field Advisor: Hunter Adams

Degree Date: May (mf783), Dec (yz2723), 2022

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Distributed Environmental Sensing System for the Johnson Museum

Author: Mingyang Feng, Yingjia Zhang

Abstract:

This project is a collaboration with the Herbert F. Johnson Museum of Art on campus. Art museum staff must regularly gather temperature and humidity measurements from throughout the museum. These measurements inform maintenance schedules and display locations for sensitive artwork. Light exposure could also inform these schedules, but the museum does not presently measure it with the same regularity. To help museum staff gather measurements, we developed an IoT system which allows them to remotely monitor the real-time environmental conditions throughout the museum. In particular, the system measures temperature, humidity, ambient light, and ultraviolet light.

Each node in the IoT system is composed of multiple deployable sensors controlled by a low-cost and low-power microcontroller named NodeMCU. The IoT system gather data from these sensors at a programmable rate. All data are aggregated in a remote database and displayed on a personal website for users to access via the internet. A one-month test has been performed in the museum to verify the system works as per the requirements.

Executive Summary

This project is designed to provide an IoT system that allows museum staff to remotely monitor the real-time environmental conditions throughout the museum. In the design process, we investigated a variety of sensors and different communication protocols. We also performed usability and feasibility studies on multiple data aggregation and visualization schemes. Chapter 3 summarizes the results of these investigations, which ultimately led us to selecting the NodeMCU microcontroller, a particular suite of sensors that are optimized for indoor measurement range and accuracy, and WiFi as the mechanism for communicating data from these sensors to the user. We compared four potential solutions for data aggregation and visualization. The final decision is to use 000webhost, a free web hosting service that stores data in a cost-effective manner, and creates a user-friendly interface for visualization.

Each sensing setup in the museum consists of multiple sensors controlled by a NodeMCU which wakes up at predefined intervals, reads temperature, humidity, ultra violet and ambient light data from sensors, and then posts readings via the campus WiFi to a third-party hosting server. Then the server inserts data into a remote MYSQL database. A website is developed to plot line charts showing the sensor readings change over time. When the user enters the URL in the browser, the server selects data from the database and then sends a webpage to the user device via the internet. Each sensing setup at different locations are associated with an independent webpage, which can be accessed through the homepage.

Two prototypes have been tested at different locations in the Johnson Museum for one month. The sensor data and its changes justify that the hardware wiring and program logic are reasonable. However, the system is sensitive to the WiFi signal strength. In addition, the tests to date have used building power, so circuit and software need to be optimized for long-term battery powering.

Individual Contribution

Mingyang Feng:

- Compared different microcontrollers and made the decision to use NodeMCU, compared different sensors.
- Wired up, programmed and tested the sensors, including DHT22, VEML6070 and VEML7700.
- Registered the MAC address of NodeMCU and tested WiFi connectivity.
- Investigated and verified the first, second and fourth schemes of data aggregation and visualization.
- Found a free web hosting service for the fourth scheme, compared four schemes and chose the fourth one.
- Modified the fourth scheme, added sleep mode, posted and stored the data from different locations in the database, developed the main page and chart pages.
- Improved the user-interface, added zoom in/out function, multiple series in a single chart and data illustration.
- Tested the system in the Johnson Museum.

Yingjia Zhang:

- Compared different sensors and microcontrollers.
- Wired up, programmed and tested the sensors, including DHT22, VEML6070 and VEML7700.
- Registered the MAC address of NodeMCU and tested WiFi connectivity.
- Investigated and verified the first and third schemes of data aggregation and visualization.
- Compared four schemes and chose the fourth one.
- Improved the user-interface, added auto-refresh function and the current data display.
- Tested the system in the Johnson Museum.

Table of Contents

1. Introduction

The Herbert F. Johnson Museum of Art on campus stores a lot of precious artworks. Art museum staff must regularly gather temperature and humidity measurements from throughout the museum. These measurements inform maintenance schedules and display locations for sensitive artwork. Light exposure could also inform these schedules, but the museum does not presently measure it with the same regularity. To provide convenience, an IoT system is developed which allows the museum staff to remotely monitor the real-time environmental conditions throughout the museum. In particular, the system measures temperature, humidity, ambient light, and ultraviolet light.

For the design stage, there are design problems and requirements to be discussed. First, low-cost and power-efficient microcontrollers and sensors with wide measurement and high accuracy need to be determined. Second, the communication protocol needs to be determined. The third issue is low-cost data aggregation and visualization. The goal is to build an interface that is easy to use even for people with no technical background.

The implemented IoT system involves sensing, communication data aggregation, and data visualization. After performing a trade study, NodeMCU [1] is chosen as the optimal microcontroller and it has a built-in WiFi module named ESP8266 [2]. Also, DHT22 for temperature and humidity [3], VEML6070 UV light [4] and VEML7700 for ambient light [5] are picked. For communication part, RedRover, one of campus WiFi networks, is utilized since it allows more flexibility compared to Bluetooth. The MAC address of NodeMCU has been registered to connect to RedRover that requires login process [6-7]. For data aggregation and visualization, a scheme based on hosting server [8] is adopted and modified to meet museum's requirements. A free web hosting service [9] is chosen to store the sensing data and a web page is built to visualize both the current and historical data. The environmental data are collected from all nodes in the museum at predefined intervals. At each node, a microcontroller gathers readings from a temperature and humidity sensor, an ambient light sensor and a UV light sensor. Then all data will be transmitted via WiFi to a remote database and displayed on a website in real time.

A test has been conducted in the Johnson Museum for one month. The sensor data and its changes prove that the system run smoothly. However, the system is sensitive to the WiFi signal strength. In addition, the tests to date have used building power, so circuit and software need to be optimized for long-term battery powering.

The report begins with a brief background and overview of the project. Second, the design problems and requirements are enumerated. Third, multiple solutions for sensing, communication and data aggregation and visualization are investigated and compared. Fourth, the implementation details are then demonstrated. Fifth, the test results are shown and compared to our original projections and expectations. Finally, future work is discussed.

2. Design Problems and Requirements

2.1 Initial System Requirements

Museum staff initially required real-time monitoring of environmental data (temperature, humidity, light exposure) at all locations to help inform maintenance requests and display locations of sensitive artworks, preferably with a central receiver that collects and displays all data.

2.2 Evolution of System Requirements

After discussion, we believe that we can develop an IoT system to meet the project requirements, so we drew up a scheme and refined the original requirements, which are divided into a sensing part, a communication part and a data aggregation and visualization part.

Sensing part: 1) Functions of collecting and transmitting environmental data are needed, so a sensor setup composed of multiple sensors controlled by a microcontroller should be placed at each location; 2) A large number of low-cost sensing devices are expected to cover the entire museum, so the microcontroller and sensors need to be small so as to be placed anywhere. Therefore, battery power supply is needed instead of the building power, which means the selected microcontroller needs to be power-efficient and it would be better to have a sleep mode to further save power. For sensors, accuracy and measurement range enough for indoor environments are required; 3) Since artworks are sensitive to both sunlight and room light, it is necessary to measure both ultraviolet intensity (from sunlight) and ambient light intensity (from artificial light).

Communication part: The communication protocol needs to allow the sensor setup to send data to the central receiver wirelessly. The data transmission should be fast and stable, and it would be better not to be affected by indoor layout. Meanwhile, the system is expected to support users to view data anytime and anywhere using any type of devices (cellphones or PCs).

Data aggregation and visualization part: There should be a central receiver to collect data from all locations and a user-friendly interface for non-technical people to use. Ideally, there should be a database to store historical data. The implementation of this should ideally be low-cost since the system is designed to run for a long period of time and requires to send data frequently.

2.3 Final Design Specifications

Overall, the main design specifications are listed as follow:

- Choosing a microcontroller which is cheap, small and power-efficient.
- Choosing humidity, temperature, UV light and ambient light sensors that are optimized for measurement range and accuracy for indoor measurement.
- Choosing a fast and stable wireless communication protocol that allows sensing setups to be placed anywhere and users to view data anytime and anywhere using any type of devices (cellphones or PCs).
- Aggregating and storing the data in a cost-effective way and creating a user-friendly interface for the museum staff.

3. Solution Comparison

The system is divided into three parts. For each part, we compared a series of solutions and finally selected the best one.

3.1 Comparison of Communication Protocols

We first considered the wireless communication protocols including WiFi and Bluetooth. Compared to the limited cover range of Bluetooth, the museum has deployed WiFi networks covering the whole building. In addition, there are lots of walls in the museum that would weaken the Bluetooth signal strength. Therefore, we finally utilized RedRover, one of campus WiFi networks that allows more flexibility compared to Bluetooth and is easier to access compared to secured eduroam.

3.2 Comparison of Parts for Sensing Setups

3.2.1 Microcontrollers

Some most commonly used development boards are compared [10-12] (see Table 3-1) and NodeMCU is picked which is integrated with a WiFi module named ESP8266. It's cheap, small, and easy to communicate via WiFi. Also, ESP8266 consumes only 10uA in its deep sleep mode [13], avoiding changing the batteries frequently. Adafruit HUZZAH Breakout and Feather both have built-in ESP8266. However, Feather boards are more expensive than NodeMCUs and Breakout boards use FDTI cables to charge which is not commonly used by non-technical people. Raspberry Pi is powerful but has no sleep mode. It is more expensive and consumes much more power even when it's idle. So the optimal board is NodeMCU.

Table 3-1: Development Boards Comparison [10-12]

Model	**Raspberry Pi Series**	**Adafruit HUZZAH Breakout**	**Adafruit HUZZAH Feather**	**NodeMCU**
Price	$4-35	$9.95	$18.95	$6.49
Power Consumption	Idle: 80-540mA	Normal:80mA Deep Sleep:10uA		
Size	2.2 x 3.4 x 0.6 inch (4B)	1.5 x 1 x 0.2 inch (Loose headers)	2.0 x 0.9 x 0.28 inch (Loose headers)	1.57 x 1.57 x 1.57 inch (Package dimension)
Cable	USB 3.0 /USB 2.0	FDTI	Micro USB /FDTI	Micro USB

3.2.2 Sensors

DHT22: It is used to measure temperature and humidity. It has a wider range, a higher resolution and accuracy than DHT11 (Table 3-2) [3][14].

VEML7700: It is an ambient light sensor [4]. It has similar range to OPT3001[15] and Si1145 [16] but higher resolution (Table 3-3). In addition, resolution and sensitivity are totally adjustable to adapt to different conditions.

VEML6070: It is an ultra-violet light sensor [5]. It is sensitive enough for indoor measurement. Compared to the GUVA sensor [17], VEML6070 can adjust its intensity precision, which can adapt to a variety of environmental conditions (Table 3-4).

Table 3-2: Humidity and Temperature Sensor [3][14]

Model	**DHT 22**	**DHT 11**
Price	$8.49/1, $14.49/2	$ 10.29/5
Range	H: 0-100%	H: 20-80%
	T: -40-80 Celsius	T: 0 – 50 Celsius
Accuracy:	H: +- 2-5%	H: +-5%
	T: +- 0.5 Celsius	T: +- 2 Celsius
Resolution	H: 0.1%	H: 1%
	T: 0.1 Celsius	T: 1 Celsius
Sampling rate	Every 2 seconds	Every one second

Table 3-3: Ambient Light Sensor [5][15-16]

Model	**VEML7700**	**OPT3001**	**Si1145**
Price	$9.86	$10-13	$9.95
Interface	I2C	I2C	I2C
Range	0-120klux	0.01-83k	1-128klux
Resolution	0.0036 lux	0.01 lux	0.1 lux

Table 3-4: UV Light Sensor [4][17]

Model	**VEML6070**	**GUVA**
Price	$5.95	$6.5
Interface	I2C	I2C
Range	320-410nm	240-370nm
Resolution	Adjustable	Not adjustable

3.3 Comparison of Data Aggregation and Visualization Schemes

A local or remote server needs to be built as a central receiver. There are many online tutorials about data aggregation and presentation based on ESP8266 and NodeMCU.

We selected 4 typical solutions and implemented them one by one to verify their feasibility and compared them. Finally, plan 4 based on 000webhost is chosen. We modified it and adapted it to our project.

3.3.1 Plan 1: Local Server

The first possible plan is to build a local server on each NodeMCU. A NodeMCU gets its local IP address from the WiFi it connects to and delivers web pages to all users under this network [18]. Our thought is, to achieve data aggregation, we can set one of the boards as the host board and add links of all boards to the webpage of this board so that the web page displays the data from all locations.

However, this method cannot save historical data. Having to type the IP address to access the webpage is not friendly enough to non-technical people either. In addition, if NodeMCU acts as a local server, then it needs to be on all the time, which make it impossible to use deep sleep mode for battery powering.

3.3.2 Plan 2: Local Server with ESP-Now

The second tutorial sets a NodeMCU as the local server and other client boards can send information to the server via ESP-NOW protocol (Client boards access the WiFi hotspot on the server board; The range is about 20 meters.) [19]. When the server board collects data from client boards, it will then connect to an existing WiFi network and users can type IP address to access the webpages if they are under this local network (Figure 3-1). Based on this tutorial, the modifications could be: 1) Group NodeMCUs as multiple clusters. In each cluster, there is a server board running continuously and other client boards can wake up at predefined intervals and then send data to the server. Each cluster is put at different locations; 2) To access the webpages of each server easily, mDNS can be used to resolve a domain name to an IP address. In this way, users can type a domain name to access the webpage.

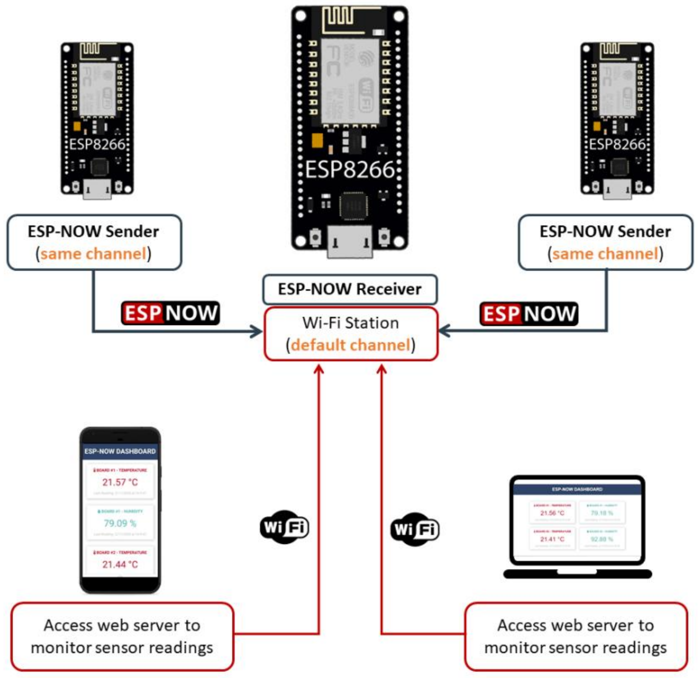

ESP8266
ESP-NOW Sender (same channel)
ESP NOW
ESP-NOW Receiver
Wi-Fi Station (default channel)
Wi Fi
Access web server to monitor sensor readings

Fig 3-1: Using ESP-NOW and Wi-Fi Simultaneously [19]

However, after experiments, there are still some issues left: 1) The range of WiFi hotspot was smaller when the server and clients were in different classrooms in the Duffield Hall; 2) Historical data still cannot be stored; 3) It has been tested that the mDNS protocol worked under a regular WiFi network, but failed under a campus WiFi like RedRover.

3.3.3 Plan 3: Web Hosting Server Using AWS

The third approach is to use Amazon Web Services (AWS) TimeStream database together with a free online visualization tool Grafana [20]. TimeStream is a database specially designed for IoT systems, which is fast and cheap. The sensing data is posted through AWS IoT core and stored in the database. With the TimeStream plugin in Grafana, data is easily retrieved and displayed as a line chart over time. But this approach requires paying annually to AWS. The free account of Grafana only supports ten dashboards, making it not cost-effective enough if we scale up our system. Also, there is no way to customize the webpage and charts, users need to login into accounts to view charts in a panel, which is not user-friendly.

3.3.4 Plan 4: Web Hosting Server Using 000webhost (Best Solution)

The final tutorial uses a paid web hosting service called Bluehost [8]. Similarly, the sensing data is inserted to remote MySQL database using PHP. A customized webpage is designed. When users access the webpage via the internet, the data is fetched from database using PHP and all data will be listed on the webpage.

We chose this tutorial as the best solution because we found a free web hosting service called 000webhost which also has a MySQL database and phpmyadmin. In this way, this approach has many advantages: 1) It is free of charge and able to store historical data; 2) It has no restrictions on the number of charts and boards; 3) The capacity of database is 1GB which can support long-term data storage. Assuming that data is collected every 5 minutes and there are 100 sensing setups, then data can be stored for more than one year; 4) It provides a free customized domain name; 5) Users can access the website anytime, anywhere using any type of devices. 6) It’s also easy to adjust the website if we scale up the system.

4. Design and Implementation

4.1 System Overview

We modified the fourth data aggregation and visualization plan to meet our project's requirements. Main adjustments include: 1) Used DHT22, VEML6070 and VEML7700 sensors; 2) Used two sensing setups; 3) Added deep sleep mode; 4) Connected sensing setups to RedRover; 4) Found a free hosting service with a free domain name; 5) Developed a homepage and two subpages for charts. 6) Improved the user-interface, added zoom in/out function, multiple series in a single chart, data illustration, auto-refresh function and current data display.

The final system consists of sensing, communication and data aggregation and visualization parts (Figure 4-1, icons from https://www.reshot.com/). The rationale is, each of the NodeMCUs in the museum wakes up at predefined intervals (2 minutes for the museum test), reads data from multiple sensors, and then posts readings via RedRover to a third-party hosting server provided by 000webhost. The server inserts data into a remote MySQL database. When the user enters the domain name in the browser, the server selects data from the database and then sends a webpage to the user device via the internet.

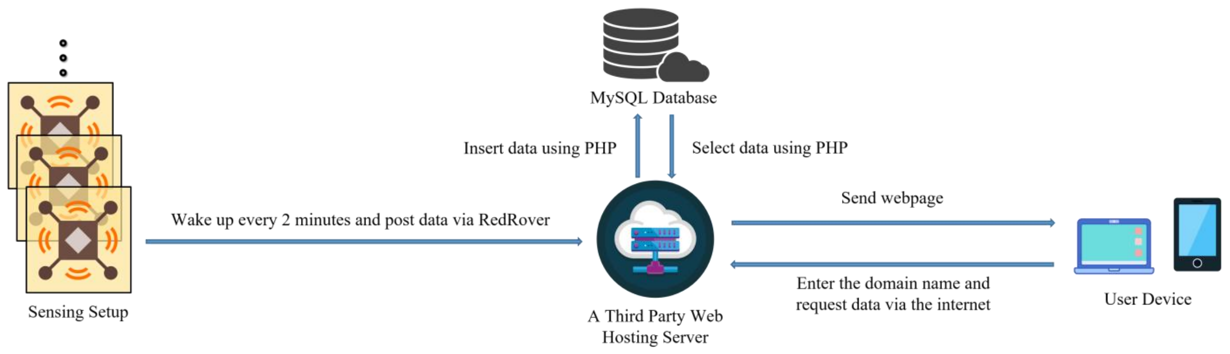

MySQL Database
Insert data using PHP
Select data using PHP
Send webpage
Wake up every 2 minutes and post data via RedRover
Enter the domain name and request data via the internet
Sensing Setup
A Third Party Web Hosting Server
User Device

Fig 4-1: System Overview

4.2 Sensing setup

The sensing setup consists of a DHT22 (temperature and humidity), a VEML6070 (UV light) and a VEML7700 (ambient light) controlled by a NodeMCU.

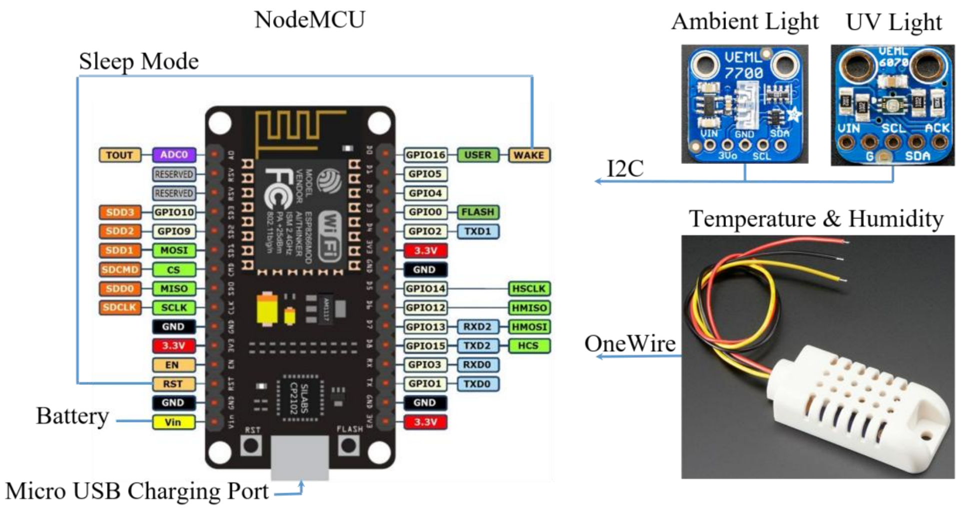


Fig 4-2: Sensing Setup

4.2.1 NodeMCU

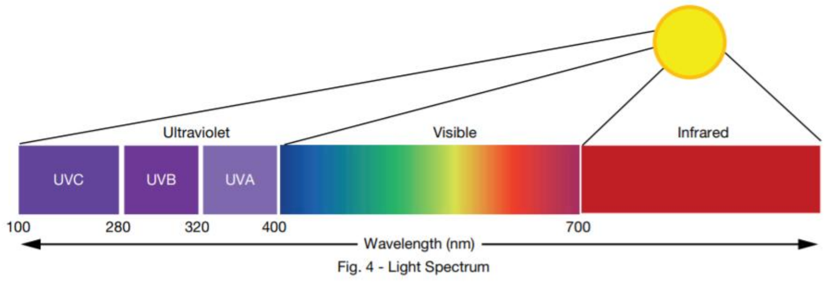
NodeMCU is programmed using C in Arduino IDE. It has a built-in WiFi module named ESP8266 running on 2.4GHz. There are 16 GPIO pins part of which support the OneWire protocol, one I2C pin, three 3.3V outputs, one 5-10V input, one 5V Micro USB jack, one analog output. NodeMCU is powered by 5-10V and the on-board regulator converts it to 3.3V to drive ESP8266 and devices connected to 3.3V outputs.

4.2.2 DHT22 Sensor

DHT22 measures humidity and temperature. It is powered by 3.3V and uses the OneWire protocol. We use the Adafruit DHT22 library to program it. To save data storage, we only measure Celsius temperature and then compute Fahrenheit using the conversion formula.

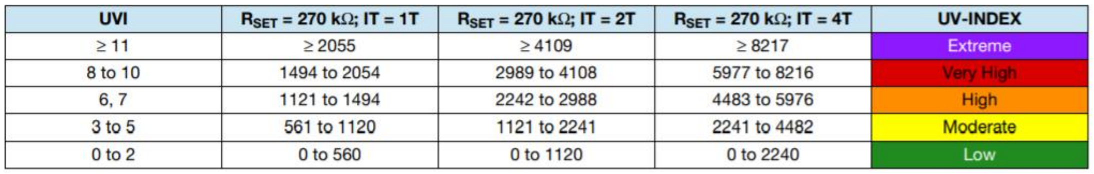
4.2.3 VEML6070 Sensor

VEML6070 is a 16-bit sensor powered by 3.3V and uses the I2C protocol. We use the Adafruit VEML6070 library to program it. According to the designing introduction of VEML6070 [4]: It detects the 320-410 nm range of UV light (Figure 4-3), which is related to sunlight; Its integration time is adjustable to change intensity precision; The integration time depends on the external resistor (our setting: R = 270 kΩ, IT = 2T); UV light raw data can convert to the UVI values (Figure 4-4).



Visible light has wavelengths between 400 nm and 750 nm.
UV light has shorter wavelengths, from 200 nm to 400 nm.
UV type A has light with wavelengths between 320 nm and 400 nm.
UV type B has wavelengths between 280 and 320 nm.
UV type C is between 200 nm and 280 nm.
While UVA and UVB reach earth, UVC is blocked by our atmosphere, so it does not cause harm.

Fig 4-3: Light Spectrum [4]



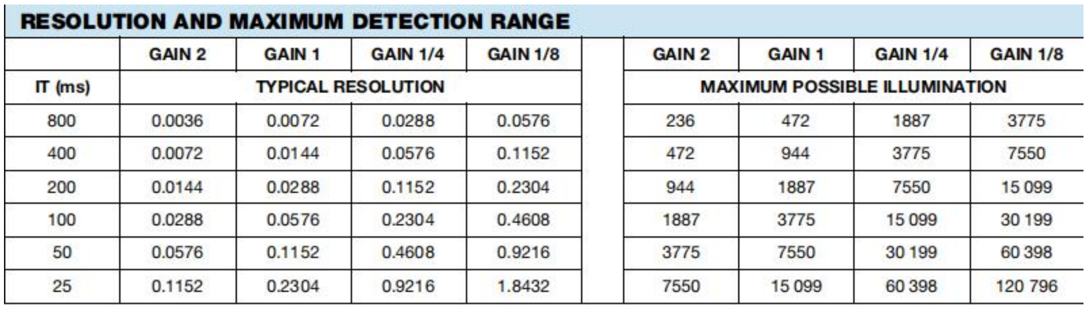
UVI	R_{SET} = 270 kΩ; IT = 1T	R_{SET} = 270 kΩ; IT = 2T	R_{SET} = 270 kΩ; IT = 4T	UV-INDEX
≥ 11	≥ 2055	≥ 4109	≥ 8217	Extreme
8 to 10	1494 to 2054	2989 to 4108	5977 to 8216	Very High
6, 7	1121 to 1494	2242 to 2988	4483 to 5976	High
3 to 5	561 to 1120	1121 to 2241	2241 to 4482	Moderate
0 to 2	0 to 560	0 to 1120	0 to 2240	Low

Fig 4-4: UV-INDEX [4]

4.2.4 VEML7700 Sensor

VEML7700 is a 16-bit ambient light (which is related to room light) sensor powered by 3.3V and uses the I2C protocol. We use the Adafruit VEML7700 library to program it. According to the VEML7700 datasheet [21], its integration controls precision and gain value controls resolution and maximum detection range (Figure 4-5; our setting: IT =

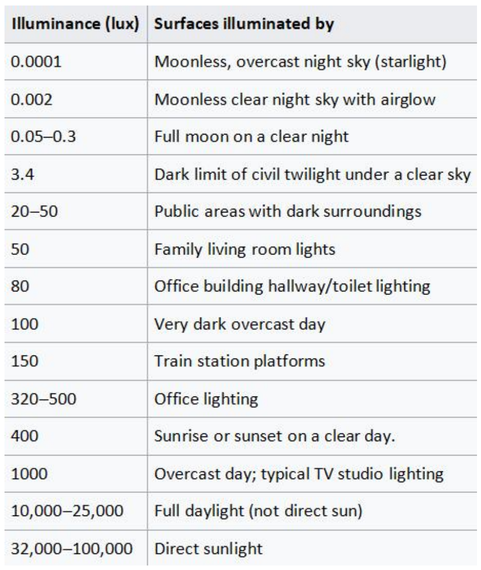
50ms, GAIN 1/8). There is a trade-off between maximum value and resolution. Resolution can be sacrificed for higher maximum value. Different Lux light raw data refers to different conditions [22] (Figure 4-6).



RESOLUTION AND MAXIMUM DETECTION RANGE

	GAIN 2	GAIN 1	GAIN 1/4	GAIN 1/8		GAIN 2	GAIN 1	GAIN 1/4	GAIN 1/8
IT (ms)	TYPICAL RESOLUTION					MAXIMUM POSSIBLE ILLUMINATION			
800	0.0036	0.0072	0.0288	0.0576		236	472	1887	3775
400	0.0072	0.0144	0.0576	0.1152		472	944	3775	7550
200	0.0144	0.0288	0.1152	0.2304		944	1887	7550	15 099
100	0.0288	0.0576	0.2304	0.4608		1887	3775	15 099	30 199
50	0.0576	0.1152	0.4608	0.9216		3775	7550	30 199	60 398
25	0.1152	0.2304	0.9216	1.8432		7550	15 099	60 398	120 796

Fig 4-5: Resolution and Maximum Detection Range [21]



Illuminance (lux)	Surfaces illuminated by
0.0001	Moonless, overcast night sky (starlight)
0.002	Moonless clear night sky with airglow
0.05–0.3	Full moon on a clear night
3.4	Dark limit of civil twilight under a clear sky
20–50	Public areas with dark surroundings
50	Family living room lights
80	Office building hallway/toilet lighting
100	Very dark overcast day
150	Train station platforms
320–500	Office lighting
400	Sunrise or sunset on a clear day.
1000	Overcast day; typical TV studio lighting
10,000–25,000	Full daylight (not direct sun)
32,000–100,000	Direct sunlight

Fig 4-6: Lux Provided under Various Conditions [22]

4.2.5 Deep Sleep Mode

The ESP8266 on the NodeMCU has a deep sleep mode to save power. When the GPIO16 Pin and the RST Pin are connected, ESP8266 can wake up automatically using the internal timer. The read calls of sensors are moved to the setup function. Every time the board wakes up, the setup function is executed once, which includes connecting to WiFi, reading data and posting them to the server. Then it enters sleep mode again. The power consumption on the table refers to the ESP8266 as a standalone chip, not the NodeMCU which has passive components that use more current [23].

Table 4-1: Deep Sleep Mode [23]

Item	Deep-sleep
Wi-Fi	OFF
System clock	OFF
RTC	ON
CPU	OFF
Substrate current	~20 uA

4.2.6 Intefacing NodeMCU and Sensors

DHT22, VEML6070 and VEML7700 are all powered by 3.3V outputs of the NodeMCU. The wiring is as follows:

Table 4-2: Interfacing NodeMCU and Sensors

DHT22->NodeMCU	VEML6070->NodeMCU	VEML7700->NodeMCU
data output->D5	SCL->D1	SCL->D1
/	SDA->D2	SDA->D2

For two I2C sensors connected to the same I2C port, theoretically a pull-up resistor and a unique address are required to distinguish. However, the sensor we selected has built-in pull-up resistors, and the Adafruit VEML6070 and 7700 libraries already include their own addresses. Therefore, there is no need to distinguish during actual programming.

4.3 WiFi Connectivity

Compared to a secured network like eduroam, RedRover is a unsecured network more easily to access. The campus WiFi is different from a regular one because it needs a browser to log in, but NodeMCU is browserless. Therefore, we registered the MAC address of each NodeMCU in the Cornell network and added an extra parameter named bssid in the WiFi connection function. In addition, we set fixed WiFi channels number for each of NodeMCU so that they won’t waste power on channel searching.

4.4 Data Aggregation and Visualization

4.4.1 A Third-Party Hosting Server

000webhost is one of the few third-party providers that provides free web hosting service. Its free account provides a database to store up to 1 GB data, a ftp to store up to 300 MB website files, and an individual website with free domain name.

4.4.2 Data Transfer Protocol

When the NodeMCU reads data and connects to RedRover, HTTP POST is used to transmit data to the hosting server. If the server responds with “200 OK”, then the transmission succeeds.

4.4.3 Database Setting

Tables in the MySQL database need to be created first. Different locations have separate tables. The database name, username and password are needed so that the PHP file can insert the data of different locations into corresponding tables when the connection is successful.

4.4.4 Website design

The URL is museumsensing.000webhostapp.com. We use PHP to fetch data, HTML to build webpage and CSS to control format and layout. For each location, four line charts of temperature, humidity, ambient light and UV light are plotted. To make it organized, we created a homepage for our website. Each location is a hyperlink to redirect the user to the page displaying line charts. There is a PHP file for each location. If the connection to the database succeeds, the sensor values are fetched ordered by the timestamp. The charts of data values over time are plotted using the HighCharts library. To make the visualization real-time, we made some improvements: 1) The latest temperature, humidity, ambient light and UV light values together with their reading time and location are displayed at the top. The user can easily get the latest value without having to scroll down to every chart; 2) Zoom in/out function is added. The user can use the mouse (for PC side) or fingers (for phone side) to drag a rectangle to zoom in to see the details and click “Reset Zoom” on the top right corner to reset; 3) The web page is auto-refreshed and the interval is set to be the sleep interval of NodeMCU, so that the page roughly includes the latest values at any time; 4) Two series are added into the temperature chart to show both Celsius temperature and Fahrenheit temperature. The user can click on either one to hide the other; 5) Tables are provided below the UV light and ambient light chart to illustrate the value under different conditions.

5. Testing Results and Comparison

5.1 Deep Sleep Mode and Battery Powering Tests

The NodeMCU using deep sleep mode powered by 4 AA batteries has run successfully as expected.

5.2 Max Value Test for Veml7700

A test has been conducted to learn the highest lux value the museum may achieve to decide how much resolution we need to sacrifice. The final setting is IT = 50ms and GAIN = 1/8 to ensure values will not saturate under bright sunshine.

5.3 Museum Test

Two prototypes were made as shown in Figure 5-1. We did a one-month test in the museum with two sensing setups charged by consistent power source (building power). Table 5-1 shows the testing details of both locations. The sleep cycle is set to 2 minutes. The one in the lobby has been running successfully for one month and the other one in the basement has run for 4 days. The website is shown in Figure 5-2 and 5-3. Part of the ambient light and UV light readings from the lobby are shown in Figure 5-4 and 5-5, where one peak represents one day. The temperature and humidity changes are not as obvious as the lights (Figure 5-6, Figure 5-7).

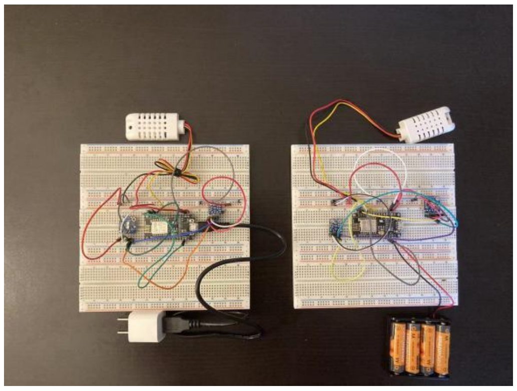


Figure 5-1: Prototypes

Table 5-1: Testing Details

Location	Duration	Power Supply	Sleep Cycle
Lobby	30 days	Micro USB Jack	2 min
Basement	4 days	Micro USB Jack	2 min

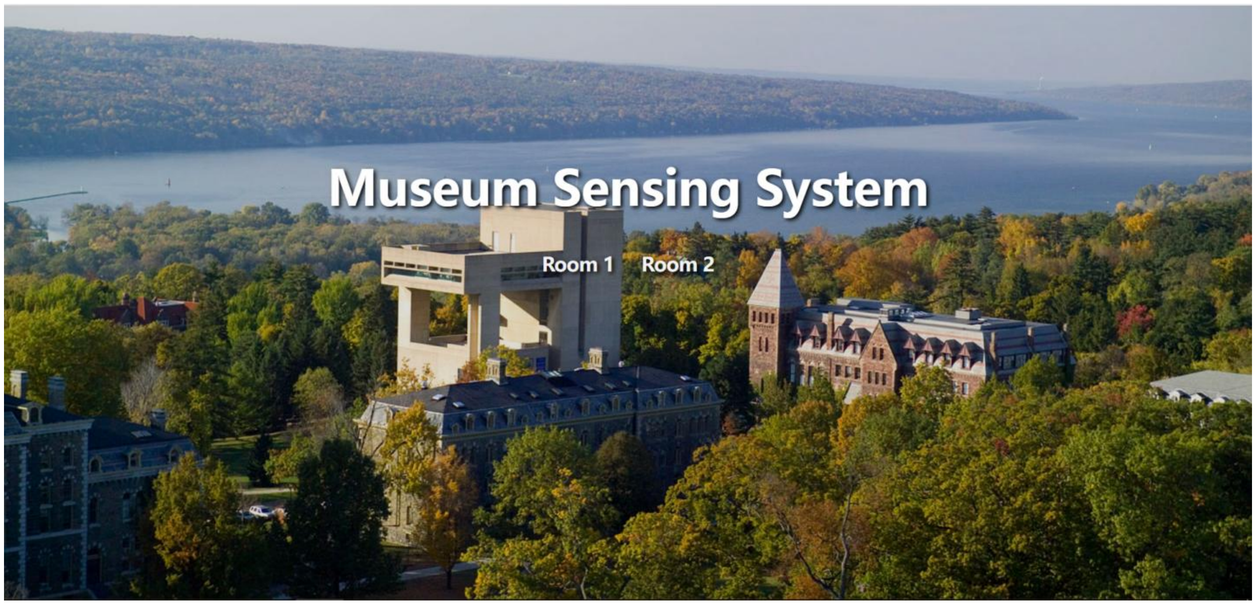
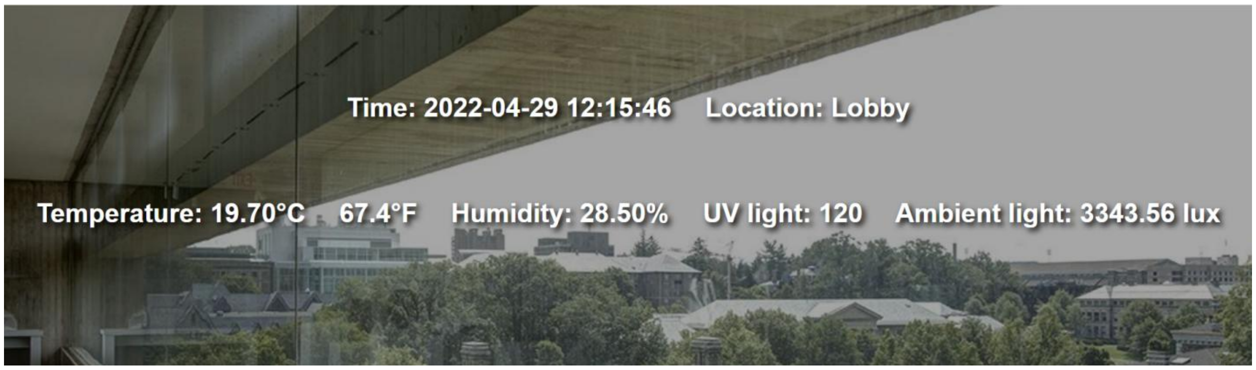
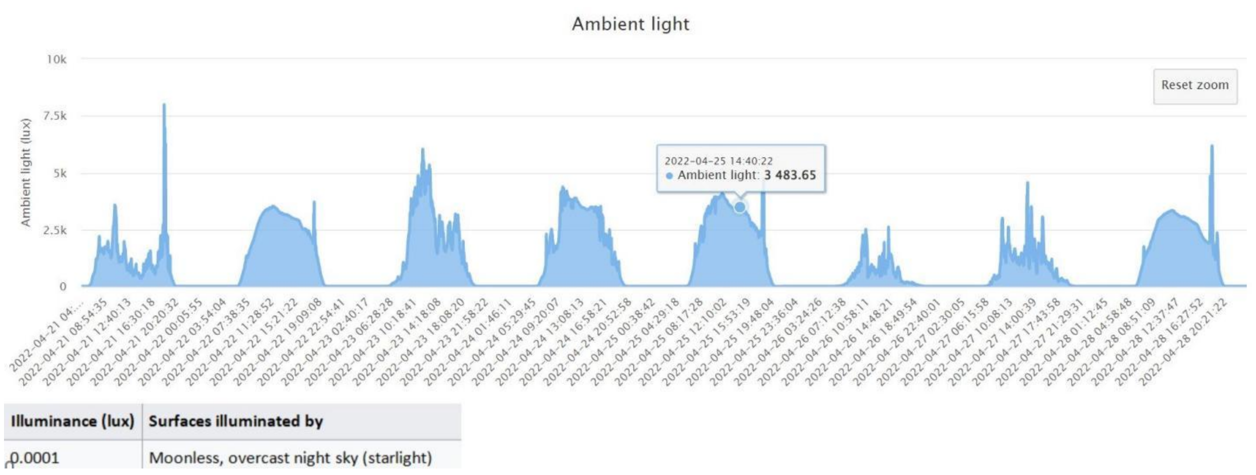


Figure 5-2: Homepage (background picture: [24])


Time: 2022-04-29 12:15:46 Location: Lobby

Temperature: 19.70°C 67.4°F Humidity: 28.50% UV light: 120 Ambient light: 3343.56 lux

Figure 5-3: Latest Values (background picture: [25])



Illuminance (lux)	Surfaces illuminated by
0.0001	Moonless, overcast night sky (starlight)

Figure 5-4: Ambient Light Testing Results

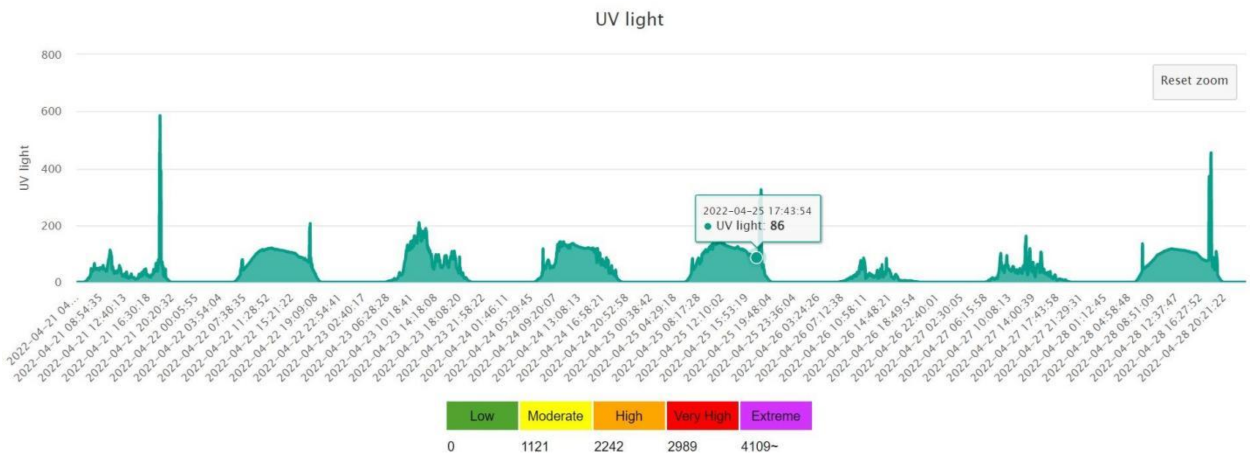


Figure 5-5: Ultraviolet Light Testing Results

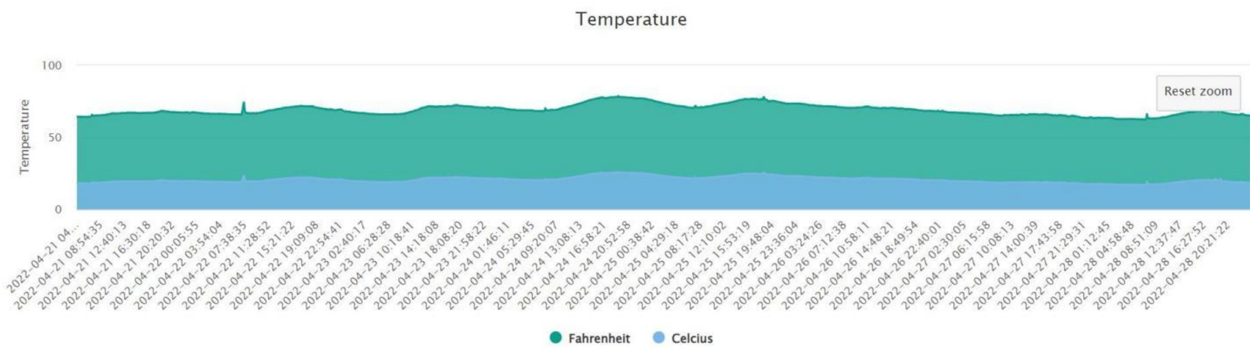


Figure 5-6: Temperature Testing Results

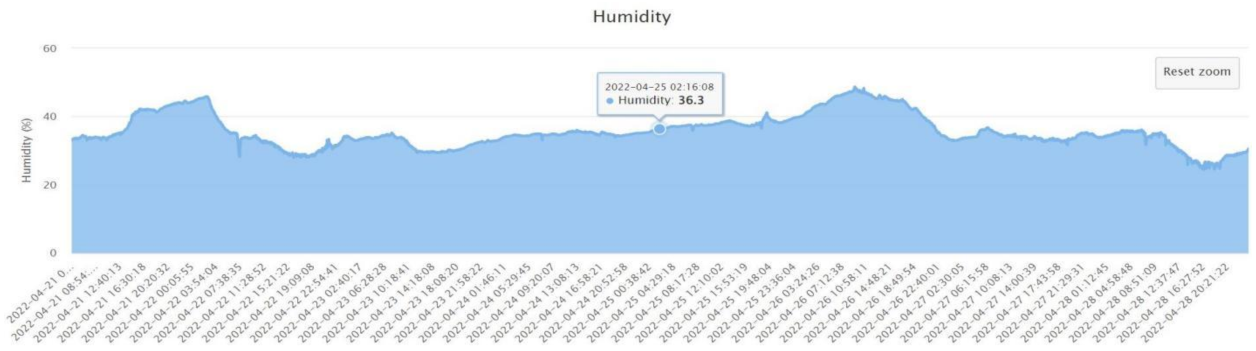


Figure 5-7: Humidity Testing Results

Unfortunately, the one in the basement only ran for 4 days. We guess that it was because the campus network signal was weak or unstable in this area, but we haven't investigated it yet. The UV light approximately remained zero (Figure 5-8) because there is no window in the basement and the ambient light value changed with the brightness of the indoor light (Figure 5-9).

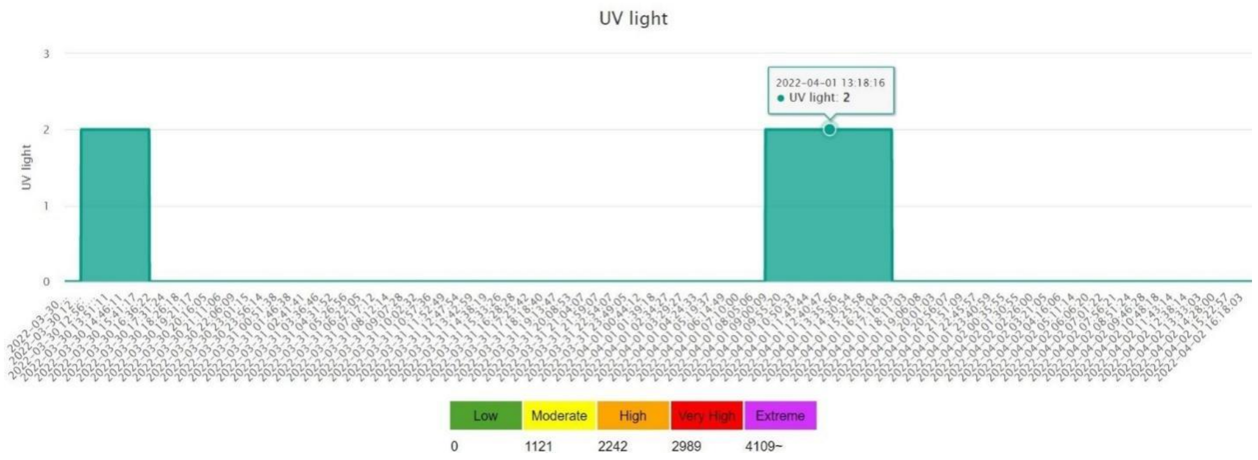
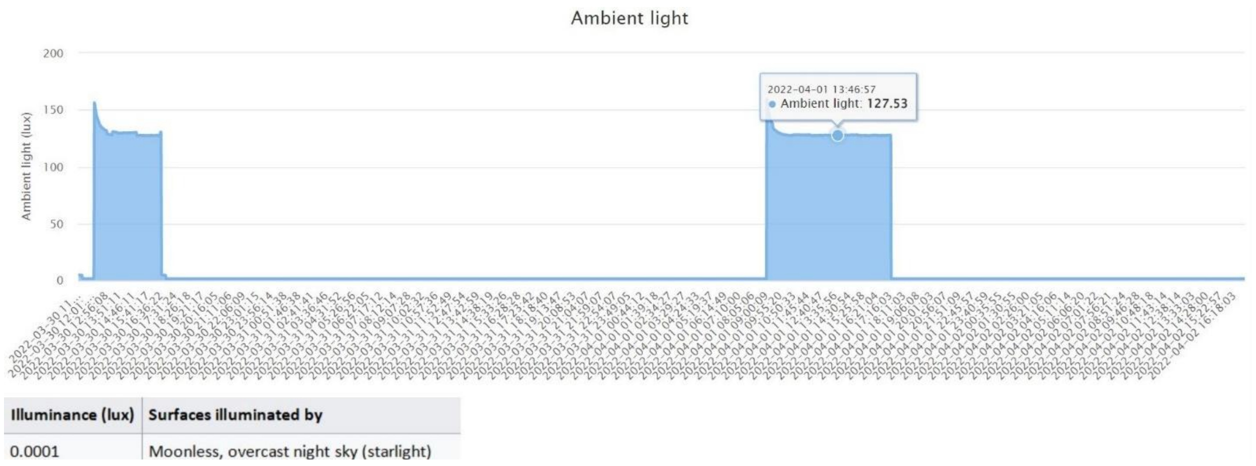


Figure 5-8: Ultraviolet Light Testing Results



Illuminance (lux)	Surfaces illuminated by
0.0001	Moonless, overcast night sky (starlight)

Figure 5-9: Ambient Light Testing Results

As expected, the website is friendly to non-technical people. Also, it's easy to adjust if the system scales up. The user can not only see the data changes over a long period of time but also investigate a single point on the chart. Overall, the results meet the basic design specifications.

5.4 Power Consumption Estimation

Theoretically, each sensing setup could run for over one year without changing batteries due to low power consumption of ESP8266. However, the actual duration would be much shorter due to the power consumption of peripheral circuits on NodeMCU.

A weather station (Figure 5-10) was built and tested which could operate for 3 weeks on 4 AAA batteries in [26]: 1) The Adafruit HUZZAH breakout board would read the DHT22 and BMP180, read the battery voltage, connect to a WiFi network dedicated for these devices, send the readings over MQTT and then go to sleep for 5 minutes; 2) A total of 0.164 mAh per working cycle of the device was consumed.

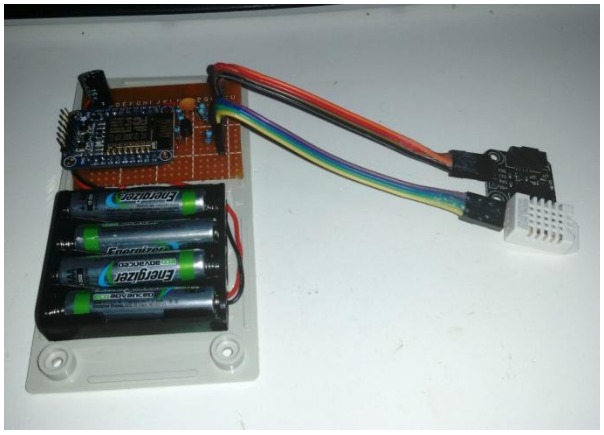


Figure 5-10: A weather station based on HUZZAH Breakout Board [26]

Therefore, the total power consumption in 3 weeks is:

$$0.164 \times 60 \div 5 \times 24 \times 21 = 991.872\ mAh,$$

which is the capacity of a AAA battery.

Compared to the settings in [26], we estimated that our system could also run for about 3 weeks if we use 4 AAA batteries and a 5-min sleep cycle.

6. Future Work

There are multiple aspects that need to be improved:

1) WiFi connectivity: For areas with weak or unstable signals, a reconnection function could be added. However, considering the network situation may not improve soon, the probability of reconnecting to the network in a short period of time is not high, so when the device disconnects midway, we can force it to go to deep sleep mode. Maybe the signal strength will restore after a couple of mintues.

2) Security: In order to make it easier to verify the feasibility of the system and prototypes, we emphasize too much on the price and ignore the security during the selection of data aggregation and visualization schemes. The free scheme based on 000webhost only provides basic security services without website backup and has watermarks on each of the webpages. Therefore, if the system scales up in the future, using a paid plan might be better. SSO login page also needs to be added to ensure security. In addition, a local server continuously running on a device with large storage capacity, such as Raspberry PI, might perform better to secure data.

3) Lower power consumption: A NodeMCU using deep sleep mode still consumes a lot of power during the WiFi connection. So reducing the amount of time of WiFi connection might help. The WiFi connection can happen after all data is read and we may set a static IP in NodeMCU to reduce the time spent by DHCP to allocate IP addresses to the boards [27].

4) A function of showing the current power consumption of batteries can be added.

5) We can add a feedback mechanism to the device. When the environment data is abnormal, the system will send an email to notify the museum staff.

6) Printed Circuit boards are needed to increase reliability of the sensing devices.

7. Conclusion

An IoT system is developed for the Herbert F. Johnson Museum of Art on campus. It provides convenience for museum staff to remotely monitor the real-time environmental conditions throughout the museum, which helps inform maintenance schedules and display locations for sensitive artwork. In particular, the system measures temperature, humidity, ambient light, and ultraviolet light simultaneously.

Each node in the IoT system is composed of multiple deployable sensors controlled by a low-cost and low-power microcontroller named NodeMCU. The IoT system gather data from these sensors at a programmable rate. All data are aggregated in a remote database and displayed on a personal website for users to access via the internet. A one-month test has been performed in the museum to verify the system works as per the basic requirements.

There are multiple aspects need to be improved, circuit and software need to be optimized for long-term battery powering, security and WiFi connectivity.

Acknowledgements

We would like to express special thanks to our advisor, Dr. Hunter Adams for his tremendous help and guidance throughout this project!

We also thank the ECE Department and the Johnson Museum staff for their help on this project.

Reference

[1] NodeMCU
https://www.amazon.com/HiLetgo-Internet-Development-Wireless-Micropython/dp/B010O1G1ES
[2] ESP8266 12E
https://components101.com/sites/default/files/component_datasheet/ESP12E%20Datasheet.pdf
[3] DHT22
https://www.adafruit.com/product/393
[4] Designing the VEML6070 UV Light Sensor Into Applications
https://www.vishay.com/docs/84310/designingveml6070.pdf
[5] VEML7700
https://www.amazon.com/Adafruit-4162-VEML7700-Lux-Sensor/dp/B07S9TD2W1
[6] Register a Device That Doesn't Have a Browser
https://it.cornell.edu/wifi-wired/register-device-doesnt-have-browser
[7] IoT Door Security System Uses Wi-Fi
https://circuitcellar.com/research-design-hub/projects/iot-door-security-system-uses-wi-fi-2/
[8] Visualize Your Sensor Readings from Anywhere in the World (ESP32/ESP8266 + MySQL + PHP)
https://randomnerdtutorials.com/visualize-esp32-esp8266-sensor-readings-from-anywhere/
[9] 000webhost
https://www.000webhost.com/
[10] Adafruit HUZZAH Breakout
https://cdn-learn.adafruit.com/downloads/pdf/adafruit-huzzah-esp8266-breakout.pdf
[11] Adafruit HUZZAH Feather
https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-huzzah-esp8266.pdf
[12] Power Consumption Benchmarks | Raspberry Pi Dramble
https://www.pidramble.com/wiki/benchmarks/power-consumption
[13] Low Power Solutions
https://www.espressif.com/sites/default/files/9b-esp8266-low_power_solutions_en_0.pdf
[14] DHT11
https://www.adafruit.com/product/386
[15] OPT3001
https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1652881150943&ref_url=https%253A%252F%252Fwww.google.com%252F
[16] Si1145
https://cdn-shop.adafruit.com/datasheets/Si1145-46-47.pdf
[17] GUVA
https://www.adafruit.com/product/1918
[18] ESP8266 DHT11/DHT22 Temperature and Humidity Web Server with Arduino IDE
https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humidity-web-server-with-arduino-ide/
[19] ESP8266 NodeMCU: ESP-NOW Web Server Sensor Dashboard (ESP-NOW + Wi-Fi)
https://randomnerdtutorials.com/esp8266-esp-now-wi-fi-web-server/#:~:text=Using%20ESP-NOW%20and%20Wi-Fi%20Simultaneously&text=What%20is%20this%3F,-Report%20Ad&text=The%20ESP32%20sender%20boards%20must,point%20and%20station%20(WIFI_AP_STA).

[20] HTTPS to AWS IoT Core
https://github.com/sborsay/AWS-IoT/tree/master/HTTPS_to_AWS_IoT_Core
[21] VEML770 High Accuracy Ambient Light Sensor With I2C Interface
https://www.vishay.com/docs/84286/veml7700.pdf
[22] Lux
https://en.wikipedia.org/wiki/Lux
[23] ESP8266 Deep Sleep with Arduino IDE (NodeMCU)
https://randomnerdtutorials.com/esp8266-deep-sleep-with-arduino-ide/
[24] Homepage Background Picture
https://localist-images.azureedge.net/photos/122291/original/28e6f2bd010bd166c431a6a7d0567
7374f52c6aa.jpg
[25] Latest Value Background Picture
https://museum.cornell.edu/sites/default/files/styles/headline/public/HFJ-FithSouth-homebanner.
jpg?itok=t5NcVQOF
[26] Reducing WiFi power consumption on ESP8266, part 1
https://www.bakke.online/index.php/2017/05/21/reducing-wifi-power-consumption-on-esp8266-
part-1/
[27] Seven Pro Tips for ESP8266
https://www.instructables.com/ESP8266-Pro-Tips/#:~:text=The%20connection%20time%20ca
n%20be,two%20their%20of%20power%20consumption.

Appendix

User Manual:

- Click on the location on the main page to show the corresponding webpage (Figure 5-2).
- The latest sensor readings, their timestamp and locations are shown on the top. (Figure 5-3)
- Scroll down to see individual charts. Hover over the chart to see the timestamp and the value of that point on the tooltip. Use the mouse (for PC side) or fingers (for phone side) to drag a rectangle to zoom in to see the details. Click “Reset Zoom” on the top right corner to reset.
  - On the temperature chart, both Celsius and Fahrenheit are displayed. Click on either one to hide the other.
  - The tables below the UV light and ambient light chart also illustrate the meaning of the light intensity.