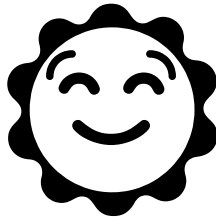

Cloud Motion Vector Sensor System

Monitoring and Predicting Output Power of a Photovoltaic System in Real-Time

Worcester Polytechnic Institute
Department of Electrical and Computer Engineering
Major Qualifying Project



A Major Qualifying Project
submitted to the Faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
degree of Bachelor of Science
Submitted 4/24/2023

Authors:

Erin Carter
Megan Hanlon
Emma Pruitt
Olivia Rockrohr

Advisors:

Dr. Tian Guo
Dr. Gregory Noetscher
Dr. Ziming Zhang

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see <http://www.wpi.edu/academics/ugradstudies/project-learning.html>

Table of Contents

Acknowledgements.....	4
Abstract.....	5
Executive Summary	6
Authorship.....	7
Table of Figures	8
Table of Tables	9
1 Introduction.....	10
2 Background.....	13
2.1 Problem Statement.....	13
2.2 Current Solutions	13
2.3 Our Solution.....	15
2.3.1 Electrical Components System Design.....	16
2.3.2 Power Calculations	17
2.3.3 Program System Design.....	17
3 Methods.....	19
3.1 Proposed Electrical System Design	20
3.1.1 Microcontroller	20
3.1.2 Cellular Module	21
3.1.3 Light sensors	22
3.1.4 Multiplexer.....	22
3.1.5 Imaging Sensor	22
3.1.6 Wire connectors	23
3.2 Estimated Power of System	23
3.3 Proposed Integration of Electrical Components	24
3.4 Proposed Hardware System Design.....	25
3.4.1 Housing Designs	25
3.4.2 Weatherproofing	27
3.4.3 User Interaction & Feedback	29
3.5 Proposed Data Analysis Design.....	30

4 Results and Modifications.....	31
4.1 Component Testing.....	31
4.1.1 Hardware and Weatherproof Testing.....	31
4.1.2 PCB Debugging and Interface Testing	33
4.1.3 Light Sensor Testing	36
4.1.4 Battery and Solar Panel Testing.....	37
4.1.5 Cellular Connection and Strength Testing.....	42
4.1.6 Data Analysis Testing	44
4.1.7 AI Image Processing Testing	44
4.2 Field Testing	45
4.2.1 Portability.....	46
4.2.2 Reliability.....	47
4.2.3 Usability.....	47
5 Conclusion	48
6 Recommendations.....	49
6.1 Electronics.....	49
6.2 Hardware.....	49
6.3 Software	49
References.....	51
8 Appendix.....	56
Detailed Data	56
Revised PCB Design.....	57
Revised Project Code.....	58
Startup Guide	68

Acknowledgements

We would like to express our gratitude and appreciation to Dr. Gregory Noetscher, Dr. Ziming Zhang, and Dr. Tian Guo for advising this project and providing insight and guidance which facilitated the completion of this project. We would like to thank Worcester Polytechnic Institute for allowing us the use of lab space, the prototyping lab, and the use of an on-campus site for testing the CMVS system. We are also grateful for the assistance of graduate student Evan Sauter for helping with design and modeling of the CMVS system. We would also like to thank William Appleyard, the Technician in the Electrical and Computer Engineering department, for facilitating the acquisition of components for the project.

Abstract

Changes in light levels often cause fluctuations in the output power of photovoltaic (PV) arrays. This unpredictability causes issues for the electric grid such as reverse power flow and voltage fluctuation. We designed a cloud motion vector system (CMVS) using an array of light sensors to calculate the size, speed, and direction of clouds and predict changes in power generation. We also began implementation of a camera and machine learning image processing to supplement the system's accuracy.

Executive Summary

While solar power is increasing in prevalence and has many sustainability advantages, several drawbacks remain regarding its integration into the power grid. One such drawback is that PV systems are non-dispatchable, meaning the output power cannot be adjusted on demand. This can cause reverse power flow and voltage fluctuations, both of which are unhealthy for the electric system. Thus, as solar contributes an increasing proportion of energy to the grid, it is important to predict solar power output more accurately in order to mitigate these issues.

All the current solutions either implement light and temperature sensors or imaging sensors, both having their own drawbacks, mostly relating to the accuracy of shading or depth predictions. The more advanced solutions are costly and still in testing. Our Cloud Motion Vector System (CMVS) is intended to be a cheap, portable, and easily installable solution for predicting irradiance and PV power output. The CMVS relies on the use of nine infrared light sensors to detect changes in light levels over time and determine the speed and direction in which clouds are moving. The system provides updates every few seconds and is able to provide information quickly and reliably on cloud cover due to its proximity to the solar array.

Our system will include 9 light sensors, a microcontroller to control these sensors, and a cellular card to reliably upload the data from the light sensors anywhere there is cellular signal. This will significantly increase the usability of the CMVS system in the field as Wi-Fi, the communication protocol used in previous iterations of this project, is not a reliable option on PV farms. We also created an enclosure for our system suited for the environment of typical solar farms that is resistant to UV radiation, heat, subfreezing temperatures, and water.

After building the CMVS prototype, we thoroughly tested the functionality of the device. This included testing the weatherproofing, power, user interface, cellular connection, data analysis, and sensor accuracy. Field testing was also conducted to determine the portability, reliability, and usability of the device.

Based on the tests, we designed a prototype that can collect all the necessary data to predict cloud motion based on the current algorithm being implemented. It is portable, reliable, and easy to use. This device is a good framework to easily begin to analyze cloud motion.

Authorship

Section		Primary Author
Acknowledgments		Megan Hanlon
Abstract		Emma Pruitt
Executive Summary		Olivia Rockrohr
Introduction		Emma Pruitt
Background	Problem Statement	Emma Pruitt
	Current Solutions	Megan Hanlon
	Our Solution	Megan Hanlon
	Electrical Components System Design	Olivia Rockrohr
	Power Calculations	Olivia Rockrohr
	Program System Design	Erin Carter
Methods	Microcontroller: Adafruit Huzzah32-ESP32 Feather board	Olivia Rockrohr
	Cellular Module: LTE Cat-M Notecard	Olivia Rockrohr
	Light sensors: TSL2591 light sensors	Erin Carter
	Multiplexer: TCA9548A I2C multiplexer	Megan Hanlon
	Imaging Sensor	Erin Carter
	Wire connectors	Emma Pruitt
	Estimated Power of System	Emma Pruitt
	Proposed Integration of Electrical Components	Olivia Rockrohr
	Housing Designs	Megan Hanlon
	Weatherproofing	Emma Pruitt
	User Interaction & Feedback	Olivia Rockrohr
	Proposed Data Analysis Design	Erin Carter
	Results and Modifications	Hardware and Weatherproof Testing
PCB Debugging and Interface Testing		Olivia Rockrohr
Light Sensor Testing		Olivia Rockrohr
Battery and Solar Panel Testing		Olivia Rockrohr
Cellular Connection and Strength Testing		Emma Pruitt
Data Analysis Testing		Erin Carter
AI Image Processing Testing		Erin Carter
Field Testing		Megan Hanlon
Portability		Emma Pruitt
Reliability		Emma Pruitt
Usability		Megan Hanlon
Conclusion		Olivia Rockrohr
Recommendations	Electronics	Olivia Rockrohr
	Hardware	Megan Hanlon
	Software	Erin Carter

Table of Figures

Figure 1: Cumulative Installed Solar Energy Capacity	10
Figure 2: Solar PV Module Prices	11
Figure 3: Key Milestones in the Exponential Growth of Solar and Wind Energy	12
Figure 4: Proposed Block Diagram of Overall System	19
Figure 5: Adafruit Huzzah32-ESP32 Feather Board Diagram	21
Figure 6: Proposed PCB design	25
Figure 7: Proposed 3D Design of Main Housing.....	26
Figure 8: Proposed 3D Design of Sensor Housing	26
Figure 9: Ingress Protection Rating System.....	32
Figure 10: Top and Bottom view of the assembled PCB.....	34
Figure 11: Schematic indicating errors in red marks, numbers from previous figure correlate. ..	34
Figure 12: Bottom view of the PCB with modifications	35
Figure 13: Light sensors outdoors, exposed to partly cloudy suns around midday. Going from left to right is no shadow cast, slight shadow cast, and dark shadow cast	37
Figure 14: Battery Voltage over 2 hours indoors with linear prediction line	38
Figure 15: Battery Voltage with Solar panel over 2 hours indoors with linear prediction line	39
Figure 16: Battery Voltage over 1.25 hours outdoors with linear prediction line	41
Figure 17: Cellular Testing Setup located on East Garage Roof	42
Figure 18: Cellular connection in AK316.....	43
Figure 19: Cellular connection at testing locations 1-2-3 depicted in fig. 17 from left to right ..	43
Figure 20: Lines showing clouds movement tracked with sparse optical flow	44
Figure 21: Weather and temperature data from February 24-27, 2023	45
Figure 22: Field testing conditions on Roof of East Garage.....	46
Figure 23: Example Output data from ThingSpeak	56
Figure 24: Revision 4 of Schematic.....	57
Figure 25: Revision 4 of PCB Layout.....	57

Table of Tables

Table 1: Estimated Power Consumption of Each Electrical Component	24
Table 2: Summary of Results for Controlled Test	38
Table 3: Summary of Results and Comparison for Solar panel test	39
Table 4: Summary of Results and Comparison for Outdoor test. Initial battery voltage for the control test is starting at the same voltage as the outdoor test to easily compare change of voltage over same time interval	40
Table 5: Description of Cellular parameters tested [60]	42

1 Introduction

Solar power is the fastest growing renewable energy source, accounting for almost 3% of U.S. energy generation in 2021 [1],[2]. The U.S. has the second-most installed solar capacity at over 90 Gigawatts, behind only China at slightly over 300 Gigawatts, as shown in Figure 1 below [3].

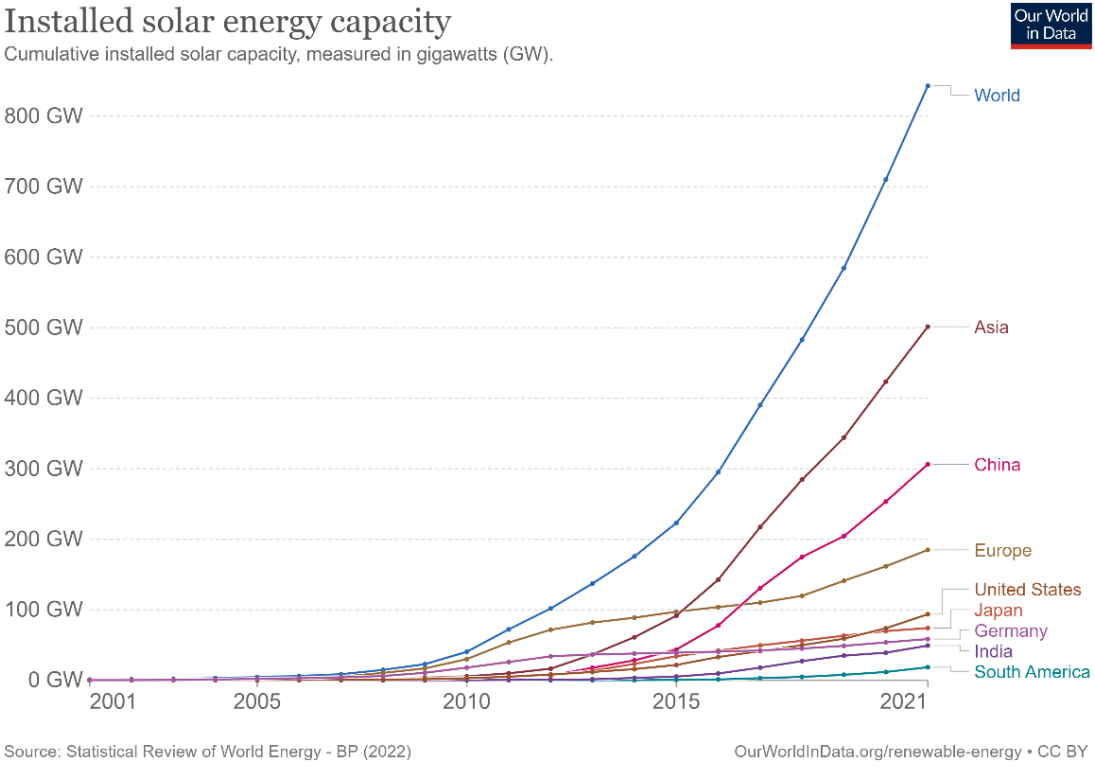
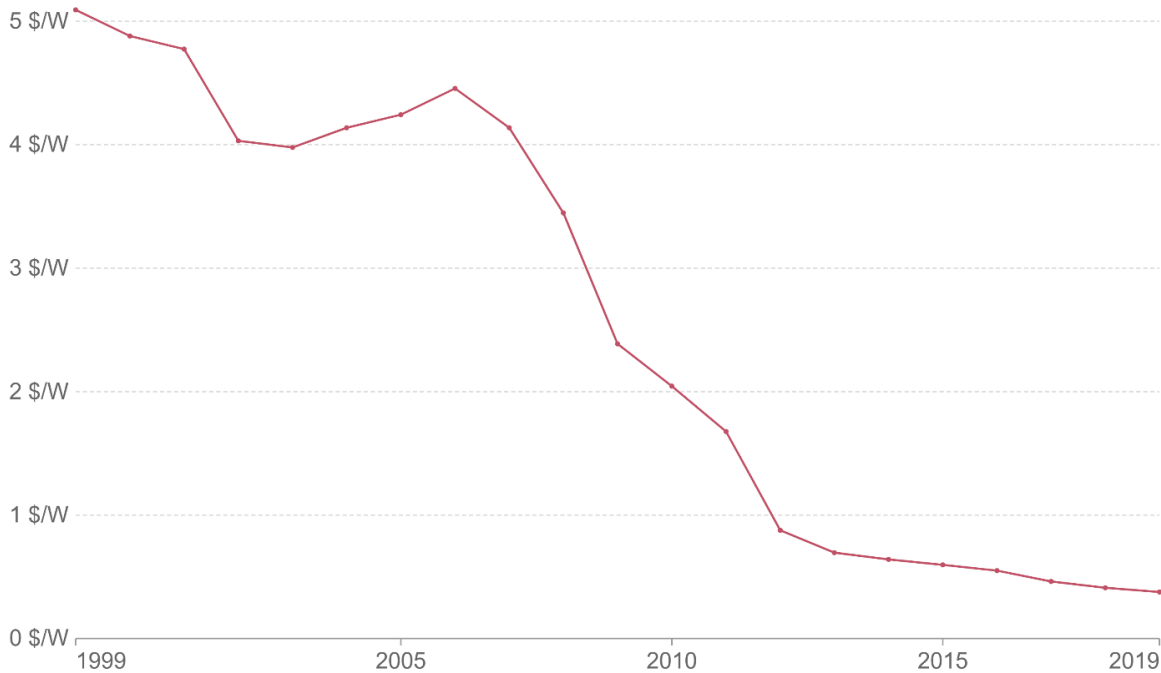


Figure 1: Cumulative Installed Solar Energy Capacity

One major cause of the rapid expansion of solar is the drastic reduction in price for constructing new solar modules. As shown in Figure 2 below from Our World in Data (2020), the price of solar photovoltaic (PV) modules dropped almost 93% from 1999 to 2019, making solar significantly more affordable and profitable [4]. In fact, the price of new solar generation has been reduced so much that it is beginning to undercut even the cheapest and least sustainable fossil fuel generation [5].

Solar PV module prices

Global average price of solar photovoltaic (PV) modules, measured in 2019 US\$ per Watt.



Source: LaFond et al. (2017) & IRENA Database

OurWorldInData.org/energy • CC BY

Figure 2: Solar PV Module Prices

Another major cause of solar expansion are the numerous policies, legislation, and investments implemented by governments and organizations worldwide to encourage and fund solar development, including the RE100 campaign, Paris Agreement, and country-specific 100% renewable energy targets [6]. These and other solar milestones are graphed against the solar and wind share of global electricity generation in Figure 3 below.

Key Milestones in the Exponential Growth of Solar and Wind Energy

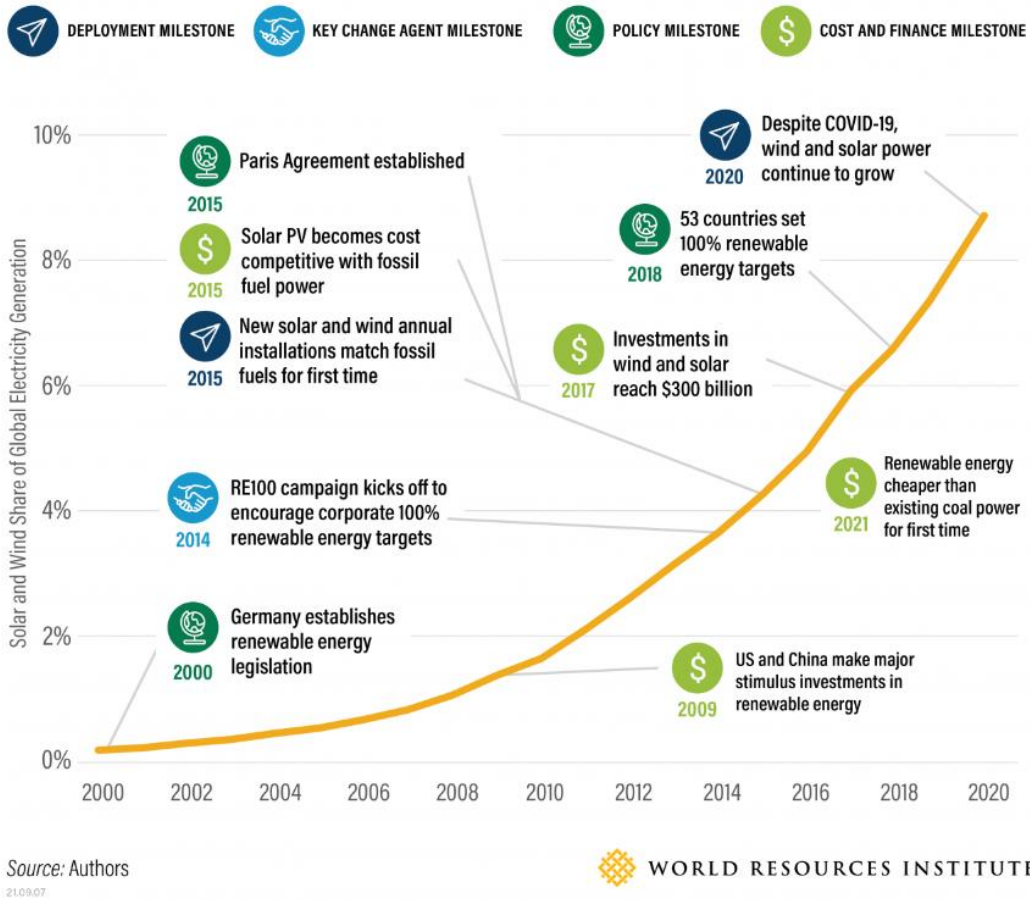


Figure 3: Key Milestones in the Exponential Growth of Solar and Wind Energy

However, as solar provides a growing percentage of the world's energy generation, some negative effects on the current electric grid are becoming apparent.

2 Background

2.1 Problem Statement

While solar power is increasing in prevalence and has many sustainability advantages, several drawbacks remain regarding its implementation in the grid. Most electric distribution systems were designed for power flowing in just one direction – top down from the big generators through the transmission system and then distribution substations to the end users [7]. However, the irregularity of PV power generation when embedded within the distribution system can cause reverse power flow, in which the generated power exceeds the local load demand, and power flows back up the distribution system, potentially damaging the utility grid [7],[8].

PV systems are also non-dispatchable, meaning the output power cannot be adjusted on demand, and voltage generation fluctuates as a result [7]. Voltage fluctuations can decrease the lifespan of network equipment or cause system-wide brownouts [9]. Even minor voltage fluctuations, which would not damage network equipment, can cause voltage flicker, or voltage fluctuations defined by corresponding variations in lighting, noticeable to the consumer [10]. To prevent issues like brownouts or voltage flickers, electricity generators need to quickly increase energy production when solar energy contribution decreases, such as when the sun sets or is obstructed for a significant period of time [11]. As solar contributes an increasing proportion of energy to the grid, it is important to be able to more accurately predict solar power output to avoid these issues.

2.2 Current Solutions

Many different devices and algorithms have been designed to predict cloud cover and PV power output to overcome the drawbacks associated with PV power generation. Electronic devices for this purpose implement different combinations of light, temperature, and imaging sensors. The Smart Monitoring Device (SMD) uses sensors that measure the voltage, current, temperature, and irradiance at individual PV panels, and transmits the data to an Internet of Things (IoT) server [12]. Weather data including irradiance, wind speeds, and atmospheric temperature are obtained from a locally installed weather station and are also transmitted to the IoT server. Sukič and Štumberger (2017) developed a system composed of a Raspberry Pi embedded computer and an OmniVision image sensor for cloud forecasting [13]. This system also transmits data to an Internet

of Things (IoT) server. The total-sky imager (TSI) is a device that utilizes light sensors for imaging and a heated, rotating hemispherical mirror to enhance optics [14]. Ryu, et al. (2019) used the TSI to take 2048-by-1536-pixel images every ten seconds during daylight hours to predict solar irradiance and PV power output [15].

Once the data is collected, several algorithms have been used to analyze and present the data. The SMD utilizes Kalman methods, machine learning, neural networks, and autoregressive models for shading prediction [12]. Sukič and Štumberger (2017) developed a new image processing algorithm to analyze the data from their image sensor system [13]. They found that a light reflection appears in some of the images, negatively impacting the algorithm's recognition of the clouds in the photo. The photos were decomposed into individual RGB (red-green-blue) colors, revealing that reflection appears most prominently in the red color. Their algorithm removes the red color parts of the photos to improve the accuracy of the predictions. The TSI has an embedded image-processing algorithm to capture and display images [14]. Ryu, et al. (2019) used a convolutional neural network (CNN) to analyze the TSI data and forecast irradiance [15].

Some research teams developed irradiance prediction models based on preexisting images or meteorological data. Park, et al. (2021) compared several models including a classical regression model, deep learning methods, and boosting methods [16]. Using images from the Waggle cloud dataset and the SWIMSEG dataset, the U-Net architecture, a deep neural network, was found to segment cloud pixels most accurately. However, this did not necessarily translate to the most accurate estimate of solar irradiance, because the network grouped various types and thicknesses of cloud pixels into the same category. The study suggests that infrared cameras could be used to collect thermal data and estimate cloud thickness, which could be incorporated into the cloud segmentation network to better estimate solar irradiance on hazy and overcast days. Cai and Aliprantis (2013) developed a cloud shadow model used to predict irradiance and PV power output using cloud cover and the cloud velocity data from the National Renewable Energy Laboratory's Baseline Measurement System [17]. The model creates cloud shadow patterns by modeling the clouds as fractals; it currently does not account for cloud thickness and has not been tested for time intervals beyond one hour. Lu, et al. (2021) tested a combination of the Cascade Causal Long Short-Term Memory (CCLSTM) and Super-Resolution Network (SR-Net) prediction models using TSI data from the Atmospheric Radiation Measurement Climate Research Facility [18].

CCLSTM was used to estimate the shape and speed of cloud motion and the SR-Net was used for improving image resolution and quality. The study found that the camera bracket and a resulting shaded area, or “shading belt,” are visible in TSI images. The CCLSTM and SR-Net lack accuracy when the camera bracket, shading belt, or high wind speeds interfere with the quality of TSI images.

2.3 Our Solution

All of the current solutions either implement light and temperature sensors or imaging sensors, both having their own drawbacks, mostly relating to the accuracy of shading or depth predictions. The more advanced solutions, such as the TSI and associated prediction models, are both costly and still in testing. Our Cloud Motion Vector System (CMVS) is intended to be a cheaper, more portable, and easily installable solution for predicting irradiance and PV power output. The team that previously worked on the CMVS conducted interviews with utility companies, residential solar providers, and research laboratories that focused on renewable energy, and found that utility companies would be the main market for this product [19]. The team found that the integration of local energy storage and faster switching times are currently a significant consideration of utility companies.

Our CMVS relies on the use of nine infrared light sensors to detect changes in light levels over time to determine the speed and direction in which clouds are moving. The system provides updates every few seconds and is able to provide information quickly and reliably on cloud cover due to its proximity to the solar array. In addition to providing advance warning of cloud cover, the CMVS is also used to calculate the output power of the solar array based on temperature and irradiance values so that backup energy storage can be switched on at the right times to smooth the output of the array. As a future upgrade for the CMVS we added the groundwork for a camera to be used in addition to the light sensors, mapping out cloud cover from a series of pictures and allowing for a more accurate prediction of cloud shape, ultimately allowing for more precise knowledge of how much the power output from the solar panel will decrease. The system is self-contained, portable, and inexpensive to produce. This allows for much greater flexibility than other models.

2.3.1 Electrical Components System Design

The CMVS system will rely heavily on an electrical system as this will achieve most of the functions described previously. This includes irradiance, temperature, and imaging sensors. Within each of these categories, there are a plethora of choices, each having their benefits and drawbacks. Power is another important aspect of our system as we would ideally like our device to operate without the need for outlets, solely relying on its own power supply. Thus, low power consumption and accurate power monitoring are important factors for the CMVS system to make it an appealing device.

Light sensors are an integral part in the CMVS system used to measure and collect irradiance values. They are passive devices that convert light energy into an electrical signal output [20]. There are many kinds of light sensors, but the most popular ones used in applications for measuring irradiance for photovoltaics include phototransistors, photoresistors, and photodiodes [20]. Thermophile devices are also used to measure irradiance values [20]. Photodiodes for example, are used in inexpensive pyranometers, devices that collect irradiance values [21]. They are small photovoltaics made from semiconductor films that convert band radiation into electrical signals. Near sunrise and sunset, a phenomenon known as Cosine Response Error could produce a measurement error in light sensors. This is because at low solar altitude angles, some of the incident light is reflected back, causing a much lower measurement than expected. Cosine error is measured by solving for the ratio, $X(\theta)$, between the measured radiation, $E_{DIR}(\theta)$, and the expected radiation, $E_{DIR}(\theta = 0^\circ) \cdot \cos \theta$, as shown below [22]:

Equation 1

$$X(\theta) = \frac{E_{DIR}(\theta)}{E_{DIR}(\theta = 0^\circ) \cdot \cos \theta}$$

A black cylinder casing and a small white plastic disk cover is one method to somewhat mitigate this error [21].

A temperature sensor will be used to measure and collect temperature values from the photovoltaic panel for accurate power prediction using Simulink. There are two ways to measure temperature: 1) through contact (conduction) or; 2) non-contact (convection). Contact temperature sensors must be in physical contact with the object being measured, while non-contact temperature sensors use convection and radiation to measure the temperature. Thermocouples, thermistors,

resistance temperature detectors (RTD), and semiconductor-based ICs are some common temperature sensors [23]. These sensors measure changes in voltage or resistance to determine temperature. A unique approach to measuring temperatures for photovoltaic applications is the use of fiber-optic sensors [24]. It has been argued that these are a better alternative to the conventional sensors like thermocouples and RTDs. They have higher accuracy, linear response, higher resolution, and faster response time. Ideally, we would want a temperature sensor that would be able to accurately measure the temperature in close proximity to the solar panels as well as withstand the extreme temperatures that can occur during the summers and winters.

2.3.2 Power Calculations

The method of estimating the total power the electrical system needs is based upon the summation of the current ratings of each component combined with an estimate of how long the system will be on versus off (duty cycle). This is a common method to estimate power usage and can be used to estimate how long a system can last if powered by an external battery source. Another method of estimating power usage is to measure and collect current values using an oscilloscope for the whole system over a set period [25]. Averaging the current values recorded in that period will give an approximate value for the average current draw of the system which can then be used to determine the power draw and lifespan of the system.

To develop a low power consumption system, the components making up that system must be power efficient [26]. But another major factor for designing low power consumption IoT systems is how long the system is in which operating mode. The mode in which the device is operating greatly affects power consumption as the different modes consume vastly different amounts of current and thus power.

2.3.3 Program System Design

For the tracking system to be useful, the effect that changes in light levels will have on a solar array must be determined. A maximum power point tracking algorithm (MPPT) is a method of determining the highest amount of power a solar panel is capable of generating based on environmental factors [27],[28]. The two variables that affect solar panel output the most are temperature and irradiance. Additional factors include the wavelength of the light that hits the solar panel and the amount of light that is reflected off the panel rather than absorbed, but these are controlled by the initial installment of the solar panel and have less day-to-day variance. The best

conditions for solar panels are high irradiance and low temperature. However, due to slight variations between different solar panels, the maximums may be at different values [29],[30], which MPPT algorithms can find using methods such as perturbation and observation (P&O), incremental conductance (IC), and fractional open circuit voltage (FOCV) [28].

The simplest MPPT is FOCV, which measures the voltage, temperature, and current of the solar panel. However, it disconnects the solar array from the load, leading to power loss [31]. P&O is another type of MPPT which tracks the power output of the solar array and adds small changes to the current set point of the system and observes whether the power output increases or decreases. This method does not cause power loss because it measures power instead of voltage, but it still has other weaknesses. It requires that the current steady state not be the maximum or minimum power output, and that there is only one maximum power output [27]. Incremental conductance compensates these weaknesses in P&O by perturbing in the other direction once it finds the maximum or minimum generation, allowing it to find more than one maximum or minimum [28].

While light sensors can be used to directly output useful data which can immediately be processed, to get a more detailed description of the sky, cameras are necessary, which do not return information that can be easily deciphered. In order to get useful data, machine learning can be used to take variables such as pixel color, image dimension, and image depth in order to determine characteristics about that image.

A neural network is a type of machine learning that is designed to function similarly to the human brain. It is made up of layers of different nodes called neurons that each represent a function. If the value of the output of a node is above a certain threshold the value is passed to neurons on the next layer, until a decision is made based on the final output of every neuron. Each neuron has a weight associated with it that is applied to change the final value of the output. Changing the weights of the neurons is how the network is trained to correctly make decisions [32].

3 Methods

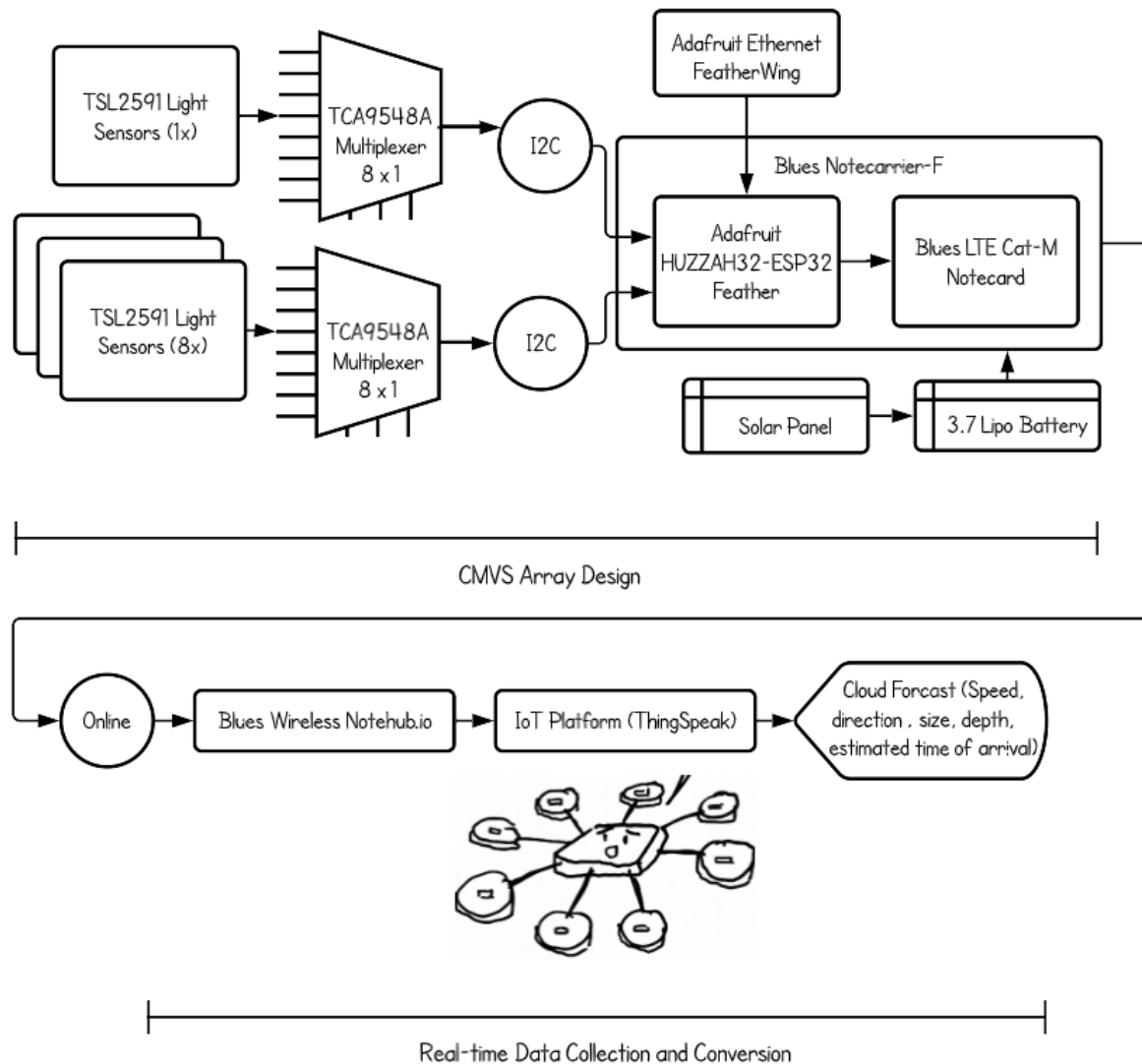


Figure 4: Proposed Block Diagram of Overall System

Our approach to the electrical design of the system, seen in Figure 4 above, was based on the recommendations of the previous MQP group and research on the solutions that currently exist. The previous MQP group conducted a series of interviews with major utilities, residential solar providers, and renewable energy research laboratories and found that the main customer for this system would be utility companies. The main problem that the system could address was found to be output voltage fluctuations/voltage flicker caused by clouds passing over solar arrays, and the desired function of the system would be to provide fluid switching to supplemental power (local

energy storage) to smooth the output power level. This can be accomplished by predicting when clouds are going to pass over a solar array and switching on the supplemental power source slightly before that occurs to account for its ramp rate [33].

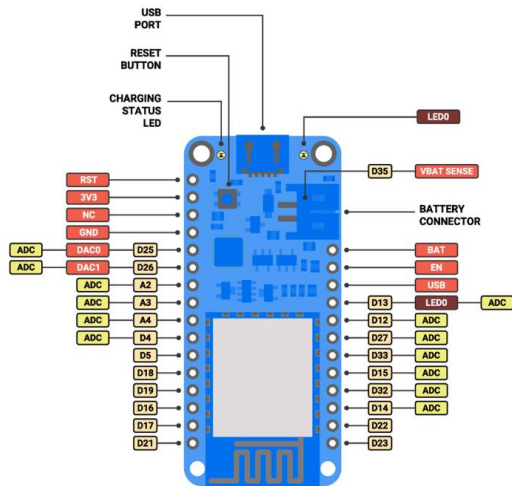
3.1 Proposed Electrical System Design

3.1.1 Microcontroller

With a device that uses multiple types of peripherals and sensors, a microcontroller is an integral part of the design. Previous MQP groups used various microcontrollers like the BeagleBone Board, Arduino Nano, and Raspberry Pi. Each of these microcontrollers has unique features that were explored, but they all have the downside of not having integrated internet communication. With each of these, an external Wi-Fi module board would have to be connected and configured in order to have the CMVS system upload data in real time.

In order to make a streamlined IoT system, we decided to use a microcontroller with an integrated chip designed for internet communication. This meant using a microcontroller with an esp-32 chip. These chips are widely used to develop microcontroller boards for making IoT systems. This chip includes Wi-Fi, Bluetooth, Ethernet, and Low Power support, making it ideal for our CMVS system [34].

There are an abundant number of esp-32 microcontrollers on the market, but the Adafruit Huzzah32-ESP32 Feather Board will be used in the current iteration of our prototype [35]. This board is the most appropriate in terms of our functional needs as well as providing reliable and detailed documentation about the microcontroller. Unlike many esp-32 microcontrollers, it is very compact and provides 28 I/O pins, enough pins for all the sensors and peripherals we could need. It also has a built-in JST connector to plug in an external Lithium-Ion/Polymer battery to power the board, unlike many other esp-32 microcontrollers. This small difference may be beneficial for us in feasibility of switching out batteries, determining battery status, and adding solar panel powering capabilities.



Device Summary

- Microcontroller: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 28
- Analog Input Pins (ADC): 8
- Analog Outputs Pins (DAC): 2
- UARTs: 3
- SPIs: 2
- I2Cs: 3
- Flash Memory: 4 MB
- SRAM: 520 KB
- Clock Speed: 240 Mhz
- Wi-Fi: IEEE 802.11 b/g/n/e/i:
 - Integrated TR switch, balun, LNA, power amplifier and matching network
 - WEP or WPA/WPA2 authentication, or open networks

Figure 5: Adafruit Huzzah32-ESP32 Feather Board Diagram

3.1.2 Cellular Module

The previous MQP group had switched between using Wi-Fi and Ethernet as methods of transferring data from the sensors to be analyzed. After researching which methods would be better for solar farms, we chose a different option, data transfer over a cellular connection. This was first recommended by a photovoltaic system design company from Massachusetts, Neo Virtus Engineering. They recommended this option as most of the equipment in photovoltaic systems have their own proprietary communication method or have a data acquisition system (DAS), which is usually a self-contained system that collects information like temperature, but it is not usually accessible to external devices. We also emailed a few people from utility companies and solar farms that the previous MQP group interviewed to ask about which communication method we should use for our project, and they all recommended using cellular or said that Wi-Fi and ethernet would not be readily available on solar farms.

Solar farms, in general, do not have reliable access to Wi-Fi or ethernet, but cellular data is accessible without the need for additional hardware. For compatibility purposes we also attached an ethernet port to the module so areas without cellular reception will still be able to use the device.

Sensors

3.1.3 Light sensors

Even more important than microcontrollers are the sensors involved in measuring and collecting the data needed to predict cloud coverage. Our design will implement 9 light sensors and a temperature sensor. Light sensors are passive devices that convert light energy into an electrical signal output. These sensors are an integral part in the CMVS system to measure and collect irradiance values [36].

The Adafruit TSL2591 light sensor was chosen as it has a wide spectrum of light it can detect and measure accurately; from 188 μ Lux to 88,000Lux. It also has the capability to measure infrared, full-spectrum, and visible light. The on-board ADC allows the sensor to be used on any microcontroller. It is also designed to draw a very low amount of current, making it ideal for “low power data-logging systems” [37]. The main issue with this light sensor is that the I2C address can't be changed. As seen in Figure 5 there is only one pair of pins (SCL and SDA) that I2C sensors can communicate with on the microcontroller. This is fine in most cases as most I2C sensors have their own unique address so they can utilize the same clock and data lines. With the TSL259 light sensor, this is not the case. Therefore, two multiplexers are needed to create unique addresses for all 9 light sensors to use the same SCL and SDA pins.

3.1.4 Multiplexer

A multiplexer is a component that selects between several analog or digital input signals and forwards the selected input to a single output line. The TCA9548A I2C multiplexer was chosen as it has enough channels to connect 8 of the light sensors and its electrical specifications are compatible with the MCU. An additional multiplexer is needed for the ninth sensor because each multiplexer has only eight channels. A second multiplexer also facilitates the potential integration of additional sensors without needing to significantly change the circuit design.

3.1.5 Imaging Sensor

In addition to light sensors, we also wanted to explore using a camera to track clouds. Cameras can provide information about cloud distance and size, which would allow the CMVS to more accurately predict decreases in irradiance and when the solar panel should switch to secondary power. The camera module we chose, the ELP-USBFHD01M-BL180, has a field of view of 180 degrees and 2 megapixels resolution, which is more than enough to get an accurate picture of the sky [38]. The camera will be connected to a Raspberry Pi to process the data because

the Raspberry Pi can be integrated into the CMVS more easily than a basic computer or laptop and is more portable. This data will then be used to train a machine learning algorithm to identify cloud information.

3.1.6 Wire connectors

The previous MQP group had significant trouble with the connectors they used between the wires to the light sensors and the central PCB. They spliced pin headers onto the ends of the wires, glued them together, and inserted them into screw terminal connectors on the main PCB and glued them onto the pins of the light sensors. This caused issues in attaching or detaching wires during cold weather conditions. Since the wires were inserted at a 90° angle, they also sometimes would break strain.

We decided to use JST SH connectors to connect the wires to the light sensors because they already have integrated SH connectors, are possible to disconnect (not glued onto the pins), create less strain on the wires, and are smaller. We used JST PH connectors to connect the wires to the main PCB because they are easier to connect and disconnect (no screwing/unscrewing needed, just push in/pull out one piece), create less strain on the wires, and are smaller (slightly larger than SH, but still smaller than the previous iteration), which helps reduce the size of the PCB. We decided to use PH instead of SH connectors on the PCB because we believed that the larger connectors would be easier to connect/disconnect, however in practice the smaller connectors were easier to use, so in future iterations of the PCB the JST SH connectors should be used on both ends of the wires. This could also slightly reduce the size of the PCB, as they are smaller.

3.2 Estimated Power of System

For our CMVS device to function with little input from solar farm operators, it needs to power itself independently. This means it needs a battery that can keep the device running for long periods of time. However, for this to work, the circuit needs to consume the minimum amount of power possible. The method we used to accomplish this goal was to program the circuit to run an active/idle cycle for all components. The previous MQP group was able to collect and transmit data in less than two seconds using a similar circuit, and the advisor who ran the project last year recommended collecting data points once every 10 minutes. Our duty cycle, therefore, is 2 seconds active out of a 600 second period, or 0.333%. We collected conservative estimates of the active

and idle power consumption of each component in the circuit from their datasheets, then used those estimates to calculate the active, idle, and average power consumption of the circuit, with values shown in the table below:

Table 1: Estimated Power Consumption of Each Electrical Component

Power Consumption	Active Current (μA)	Idle Current (μA)	Number of Parts
TSL259 Light Sensor	325	4	9
TMP102 Temperature Sensor	40	0.5	1
TCA9548A Multiplexer	35	2	2
ESP32 Feather Board	80 mA	5	1
LED	40 mA	0	2
Cellular Module	250 mA	8	1
Total	413 mA	51.5	-
Average = 1.428 mA			

From this data, we selected a battery that would integrate easily with the circuit and have enough capacity to run it for a reasonable amount of time. We decided on the Adafruit 3.7V Lithium-Ion Polymer Battery with 1200mAh capacity, which, based on our average power consumption, could power our circuit for 35 days.

So that the absolute minimum maintenance is required for our device, we also decided to incorporate a small solar panel that could recharge the battery. We chose a solar panel that had a maximum current equal to the maximum charging rate of our battery, 500 mA, so the solar panel would never damage the battery [39]. Over 24 hours the circuit will consume $1.428 \text{ mA} * 24 \text{ h} = 34.272 \text{ mAh}$, and the solar panel can generate that much if it is exposed to direct sunlight (maximum current) for $\frac{34.272 \text{ mAh}}{500 \text{ mA}} * 60 \frac{\text{minutes}}{1 \text{ h}} = 4.113 \text{ minutes}$. Because the solar panel needs so little exposure to sunlight per day to keep up with the circuit's power consumption, we are confident that it will be able keep the battery consistently charged.

3.3 Proposed Integration of Electrical Components

Given the multiple sensors and additional chips, designing a PCB to integrate these components will allow us to have a compact and portable device that is durable and easy to set up. Below is the final iteration of our PCB which includes the cellular notecard, ethernet module, connectors for peripherals, and integrated multiplexer chips.

We first did a trade study on all the components that we would need for the PCB. After creating a rough bill of materials, we created a schematic capture which included creating custom symbols for some of the chips. Then we moved to the layout and routed the board. We had a couple iterations of our PCB as we modified some of the components and improved the design with guidance from another WPI student with significant experience with PCB design.

The most recent iteration of the board was approved to be fabricated. The fabrication process took some time as specific files were needed to fabricate the board. These files included BOM file, Gerber files, and pick and place files. Once these files were correctly formatted, we submitted them to JiaLiChuang Printed Circuit Boards (JLCPCB) and chose to have all the surface mount devices soldered in house. Our final iteration is a 4-layer PCB that is 100mm by 100mm. Due to the size of the board, it was very inexpensive to produce, costing just a little over \$10 to produce 5 boards.

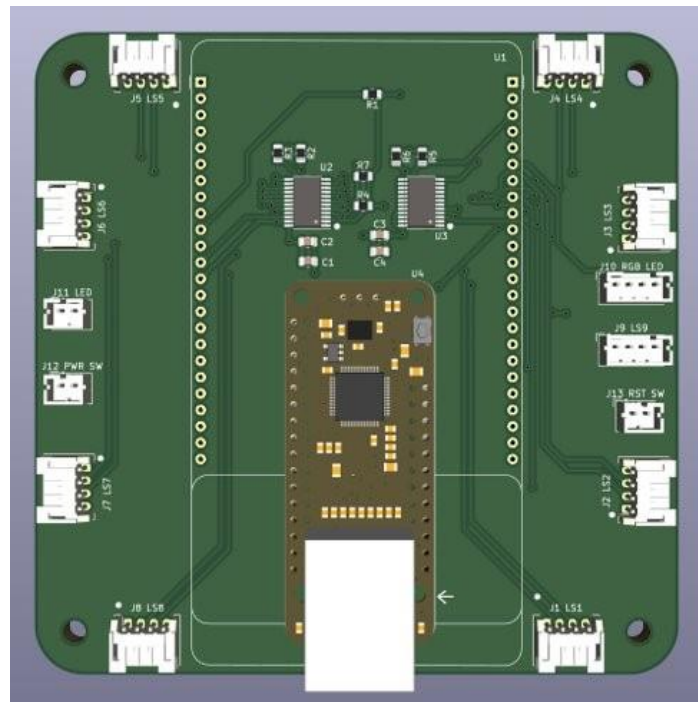


Figure 6: Proposed PCB design

3.4 Proposed Hardware System Design

3.4.1 Housing Designs

We 3D-printed two housing structures to protect the electrical components of the CMVS system: main housing and sensor housing. The main housing, shown below, protects the PCB,

central light sensor, battery, two buttons, and two LEDs. The main housing was printed using the LulzBot TAZ 6 due to the size restrictions of the Ultimaker 3. The LulzBot TAZ 6 is compatible with Polylactic Acid (PLA) and Thermoplastic Polyurethane (TPU). TPU is flexible and elastic, which makes it more difficult to print [40]. The design was printed with PLA because it does not require the flexibility of TPU.

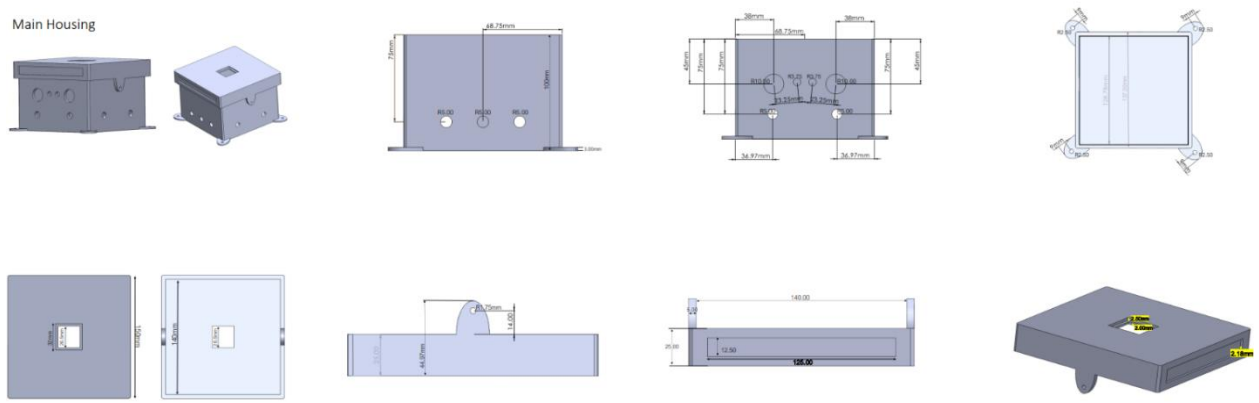


Figure 7: Proposed 3D Design of Main Housing

The sensor housing, shown below, protects the eight outer sensors of the CMVS system. Eight copies of the sensor housing design were printed using the Ultimaker 3 for better quality. The Ultimaker 3 is compatible with PLA, ABS, polypropylene, nylon, PETG, polycarbonate, and TPU 95A. The sensor housing was also printed with PLA due to its good UV resistance compared to the other filament options and its capacity for post-processing, such as sanding or drilling, for more accurate screw holes or fastening pegs [41]. PLA can be susceptible to absorbing water and swelling, but only if submerged in impure water for a long period of time [42].

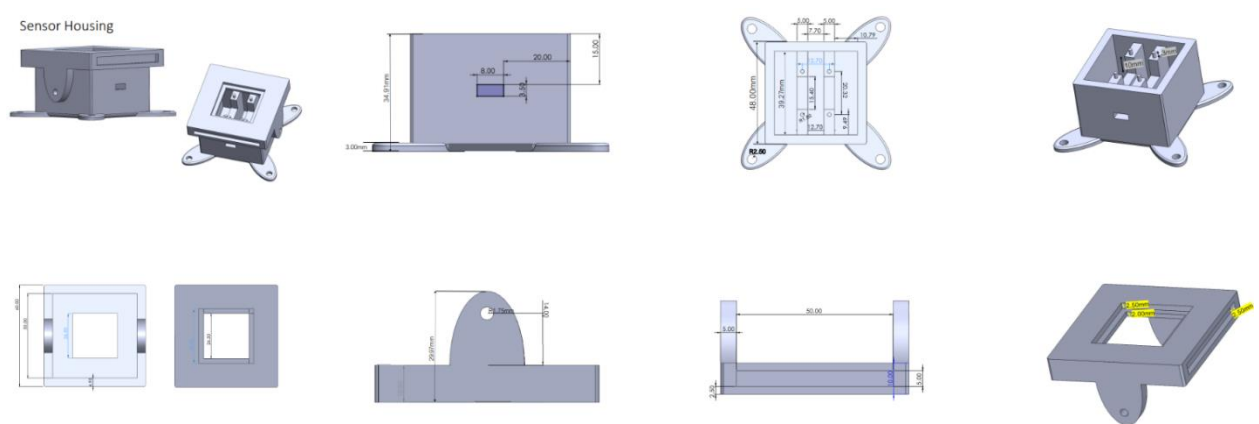


Figure 8: Proposed 3D Design of Sensor Housing

To optimize the designs for mechanical strength, the thickness of all the enclosure walls is at least two millimeters [43]. Fillets are also included on the external edges to reduce stress concentrations and make the parts easier to print [43]. The structures were designed to be able to add gaskets for waterproofing between the lid and the base. Due to the uncertainty of the gasket thickness and compression tolerance, the screw holes for connecting the lid to the base were drilled after printing.

The main housing and sensor housing are similar except for a few considerations such as size. Both the main housing and sensor housing have a square opening in the lid with a lip where a quartz pane can be affixed. The quartz pane is used to keep the enclosure water resistant while allowing the light sensor to detect light. Both the main housing and sensor housing have tabs to attach the lid to the base with screws and tabs to attach the base to a mounting apparatus with screws.

The main housing is larger to accommodate for the size of the PCB. The main housing includes additional openings for two buttons and two LEDs, as well as wiring the eight sensors and the solar panel. The sensor housing includes pegs on the base to fasten the light sensors against the quartz pane. The main housing does not include these pegs due to the placement of the PCB in the base. Velcro will be used in the main housing to fasten the light sensor and the battery.

3.4.2 Weatherproofing

In order to create a CMVS device that could function in the field, we had to design an enclosure suited for the environment of typical solar farms. This required a device that is resistant to UV radiation, heat, subfreezing temperatures, and water. First, we ensured that every component used in the device had a temperature range of well below -20°F to well above 100°F because temperatures in New England usually reach similar highs and lows at least once a year. Second, we chose the material for our 3D-printed housing structures to be UV-resistant because the device needs to be located in an open area with full exposure to the sky, and therefore the sun. Third, we had to find methods for waterproofing every part of our device, because it will be exposed to rain, snow, and everything in between, and electronics can be damaged when exposed directly to water.

One way to achieve this is by designing our device with components rated as being waterproof. The foremost rating system, Ingress Protection (IP), developed by the International Electrotechnical Commission (IEC), grades the resistance of an enclosure against intrusion of dust

or liquids [44]. When this is not possible, common in electronic components and PCBs, there are common compounds that can be applied to waterproof electronic materials, some of which include epoxy, silicone, and acrylic [45].

For our 3D-printed housing structures, we followed several guidelines on designing and printing waterproof 3D parts including: designing very simple parts (square boxes with lids) with minimal holes and thick walls (more than 4mm), and printing with a filament that is less prone to leaking (PLA) [46],[47].

We decided to use a screw-on lid design for the housing structures, which meant we needed to seal the seam between the lid and housing. Because of the similarity of the lid seam to the seam of a door or window (a space between two flat, linear surfaces that needs to be opened and closed frequently) we decided to seal it like one would a door or window, namely with a gasket. We decided to use a Neoprene-based adhesive foam seal tape due to Neoprene's good weather/water resistance and cost-effectiveness, as well as the flexibility of an adhesive tape that could be cut to any size and applied immediately to a surface [48][49].

In order to protect the light sensors from the weather, while also allowing them full view of the sky, we decided to integrate quartz glass panes into our housing structures. We needed a material to seal the gaps between the panes and housing as well as any imperfections in the housing itself, so we chose a quick-set epoxy syringe product due to its ability to bond well with both plastic and glass, its common use in waterproofing 3-D prints, and its ease of use (no mixing required) [50][51].

The last and most complicated type of gaps we had to seal were the holes for the external wires connecting the radial sensors to the main circuit. We decided to consult with the ECE Department's Electronics Technician, Bill Appleyard, on what type of sealant to use as he is very experienced in making electronics projects, and he recommended Hot Melt, a highly viscous, water resistant sealant that bonds well with plastic, perfect for filling the wire holes in our application [52].

We also made an effort to limit bare wire connections, bent wires, and unnecessary holes in the housing, and used heat shrink tube to cover any bare wire-to-wire connections to further increase the water- and cold-resistance of the device.

3.4.3 User Interaction & Feedback

Weatherproofing our device makes it more difficult to access the electronics. To make our device more user friendly, adding external buttons and LEDs will improve the user interface as it allows some interaction and feedback from the device.

For user interaction, adding external buttons to the housing will allow users to do basic interactions with the CMVS device. The most important functions that need to be easily accessible are powering the device and resetting the device which will reset communication and power. Being able to reset the device is an important function to easily access if the user encounters malfunctions.

It is critical that the external buttons have an IP rating of at least a 3 which is protection against spraying water at an angle up to 60° [44]. For the power button specifically, since we are using it to connect and disconnect the LiPo battery to the rest of the system, a single pole single throw switch (SPST) will be sufficient. ZF electronics produces a series of waterproof rocker and pushbutton switches that are ideal for our application. They are inexpensive, are designed with a snap-fit mount and both have IP ratings of IP65 (dust tight and water resistant) [53][54]. Specifically, within this series, the KCA2ANA1BBB seems adequate as it is currently priced at \$4.04 and is currently in stock (Rocker Switch)[53]. For the reset button, a normally open push button, is most appropriate as it is the type of button also used within the MCU. Specifically, within this series, the KFB2ANA1BBB seems adequate as it is currently priced at \$4.04 and is currently in stock (Push Switch)[54].

For user feedback, adding LEDs to the housing will allow the user to get some indications of the current status of the CMVS device. The most important statuses that need to be indicated include the power of the system, whether the system is connected to communicate and transfer data, and when the system is resetting. For each of these statuses, they have a couple different states. For power, this includes low power and sufficient power. For communication, this includes connected or trying to connect. For the reset status, there is only one state which is when the device is resetting. Due to the various states that each of these statuses have, a multicolored LED would be ideal to concisely indicate what state each status is in. Thus, one RGB LED will be used for power and communication status and one blue LED will be used to indicate reset status. The RGB will be green when there is sufficient power to the device and red when power is low. It will also

indicate when it is trying to connect to communication by blinking and will stay solid when there is a solid connection. The blue LED will simply light when the reset button is pressed.

3.5 Proposed Data Analysis Design

Information on the cellular card is stored in notecards which can be created using Arduino methods or directly defined using JavaScript Object Notation (JSON). These notecards are then stored in files so that they can be organized into different groups to make data transfer easier. In order to transfer data using the cellular card we used a software called Notehub. Notehub was created for use with the brand of cellular card we have and allows sensor data to be transferred easily using JSON. The data in these files can then be sent to other websites for analysis. To send data between the two sites, the user must create a route on Notehub specifying the website name, the file to be transferred, and the channel it is sending data to. We decided to build off the previous group's code and use ThingSpeak.

ThingSpeak uses MATLAB to analyze data points and create visualizations of the data. It collects data in channels with an input of up to 8 different variables which are automatically displayed on a graph and can also be exported as a csv file [55]. Because the CMVS uses 9 sensors, we needed 2 different routes from Notehub to be able to transfer all the data, and we also ensured the data was stored in 2 different files to decrease the likelihood of any potential bugs. From there we added MATLAB code to determine cloud speed and angle based on the irradiance data.

Simulink is a modeling software that can be used to predict outcomes of different events. For this project we used it to predict the power output of solar panels. Temperature and irradiance data from the CMVS along with information on how much power the solar panel produced is used in maximum power point tracking algorithms to determine the maximum and minimum solar output. This information can then be used to predict power output in the future so the CMVS can provide an estimate of power loss or gain as irradiance and temperature change.

4 Results and Modifications

We conducted several tests to verify parts of our device and characterize its performance. Our goal was to build a robust, reliable system that would be able to collect all the information needed to predict cloud motion. The first stage of tests focused on testing individual parts of the CMVS system from mechanical components to electrical components. After collecting results on the performance of these individual parts, we integrated the whole system and conducted field tests. This pushed the CMVS to its limits as it was exposed to extreme temperatures and various weather conditions.

4.1 Component Testing

The component tests conducted focused on important features of the CMVS system. Based upon our proposed description of the functionality of this device, these features are: weatherproof, power, user interface, cellular connection, data analysis, and sensor accuracy.

4.1.1 Hardware and Weatherproof Testing

Following the IEC 60529 for Ingress Protection (IP) rating system shown below [44], we tested if water was able to penetrate the 3-D printed sensor housings when hit by falling water droplets, jets, and splashing water using a combination of faucets and shower heads to produce the desired effects. We put paper towels inside the enclosures and visually confirmed if any water got inside after each test by opening the enclosures and checking if the paper towels had been darkened by water. We tested one prototype weatherproofed sensor housing with the IEC IP ratings that simulated outdoor weather conditions and were not expected to damage the housing. Then we weatherproofed the rest of the sensor housings and tested them using a shower to simulate falling and splashing raindrops.











0	No protection		-
1	Protected against vertically falling water drops		Vertically falling drops shall have no harmful effects
2	Protected against vertically falling water drops when enclosure tilted up to 15°		Vertically falling drops shall have no harmful effects when the enclosure is tilted at any angle up to 15° on either side of the vertical
3	Protected against spraying water		Water sprayed at an angle up to 60° on either side of the vertical shall have no harmful effects
4	Protected against splashing water		Water splashed against the enclosure from any direction shall have no harmful effects
5	Protected against water jets		Water projected in jets against the enclosure from any directions shall have no harmful effects
6	Protected against powerful water jets		Water projected in powerful jets against the enclosure from any direction shall have no harmful effects
7	Protected against the effects of temporary immersion in water		Ingress of water in quantities causing harmful effects shall not be possible when the enclosure is temporarily immersed in water under standardized conditions of pressure and time
8	Protected against the effects of continuous immersion in water		Ingress of water in quantities causing harmful effects shall not be possible when the enclosure is continuously immersed in water under conditions which shall be agreed between manufacturer and user but which are more severe than for numeral 7
9	Protected against high pressure and temperature water jets		Water projected at high pressure and high temperature against the enclosure from any direction shall not have harmful effects

Figure 9: Ingress Protection Rating System

The following tests were applied to the prototype sensor housing: vertically falling water droplets, vertically falling water droplets with enclosure tilted up to 15°, water sprayed up to 60° from vertical, water splashed from any direction, jets of water from any direction. The housing was checked after each test, and the housing successfully kept water from entering for all five tests. The remaining tests included in the IEC 60529 standard risked damaging the enclosure and do not reflect the conditions the device will face, so they were not included.

During testing, the silicone sealing one of the screw holes was pushed out slightly by the screw, however water was not able to enter. In subsequent weatherproofing iterations, silicone was applied with the screw already in the screw hole so that the screws did not push the silicone out when fully inserted. In addition, water was observed to reach as far as the outer edge of the silicone and gasket, but it did not seep in past them, so it appears that the weatherproofing was very effective.

Two sensor housings were tested at a time, placed on the floor of a conventional shower with water from the shower head directly hitting them for 60 seconds. Paper towels were inserted inside the housings before they were placed in the shower, and after 60 seconds the housings were removed, opened, and the paper towels and inner surfaces of the housing were thoroughly inspected for any moisture. After testing all 8 sensor housings with this method, none contained any internal moisture. However, while fastening the lids to the main body of the sensor housings with the screws, the screw hole tabs on two different lids snapped off. Upon inspection it appeared that the tabs snapped in places where the 3-D prints had a slightly misprinted layer. It may be advisable to increase the infill on the tabs in future prints or to find a different way to fasten the lids to the sensor housings as the current method may put too much strain on the tabs. We did not have enough time to pursue either of these solutions, so we glued the tabs back on with epoxy adhesive and attached the lids gently without tightening the screws all the way, and no further tabs snapped off.

We were not able to finish weatherproofing the main housing until right before we needed to test the fully assembled system. Given the success of the sensor housing weatherproofing, we applied the same techniques to the main housing and tested its weatherproofing in the field, as explained in the next section.

4.1.2 PCB Debugging and Interface Testing

After fabricating version 3 of the CMVS PCB design, which has 4 layers (two signal layers and two plane layers (ground and power) and assembling all necessary parts we did a continuity check and compared it to the schematic. There were several issues identified in the following figure.

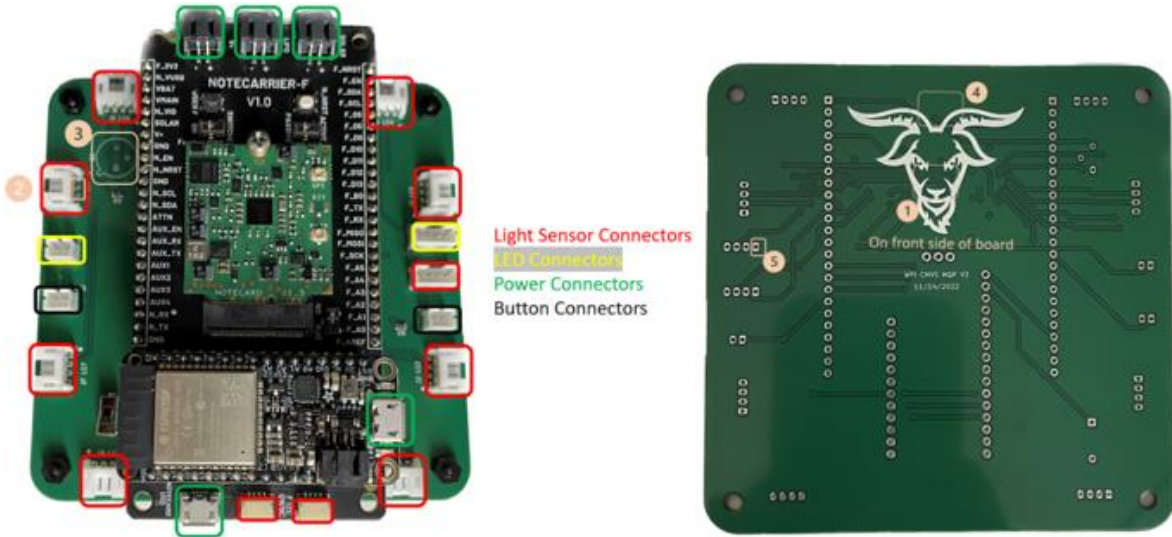


Figure 10: Top and Bottom view of the assembled PCB

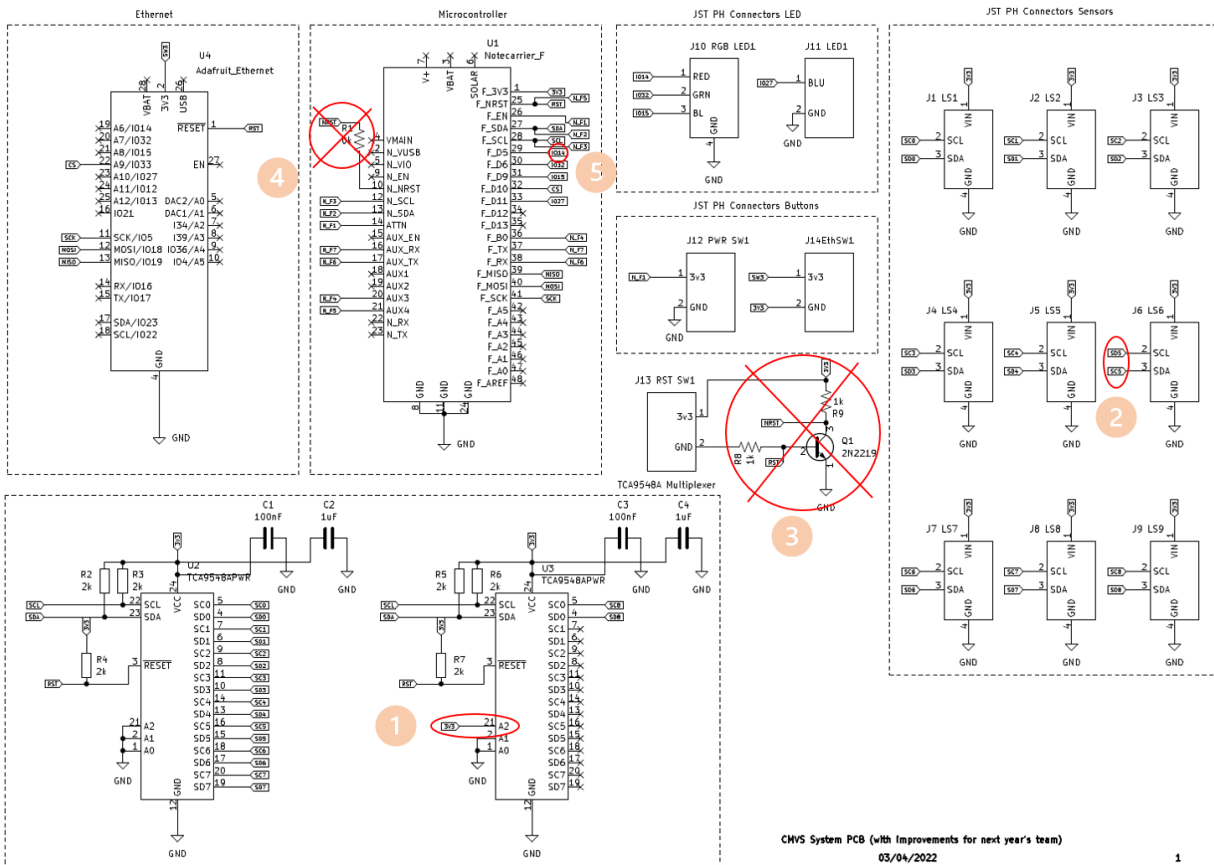


Figure 11: Schematic indicating errors in red marks, numbers from previous figure correlate.

The errors mainly were due to forgetting to include ground symbols on the schematic which would include that route to the ground net. There was also an interesting error involving a missing route between the red pin on the RGB LED and the associated GPIO pin on the notecard.

Other than some missing connection points we determined that both the esp32 and the notecard both run on the same logic level for getting them in the reset mode. Both are active low resets. With this known, a transistor is not necessary. This original scheme was implemented as we thought the two chips were opposite logic levels and thus implemented a not gate to have them both reset at the same time. There were thus some extraneous routes that were disconnected. There were also two minor errors that did not affect the performance of the device but are noted in the figure as it is not the desired choice.

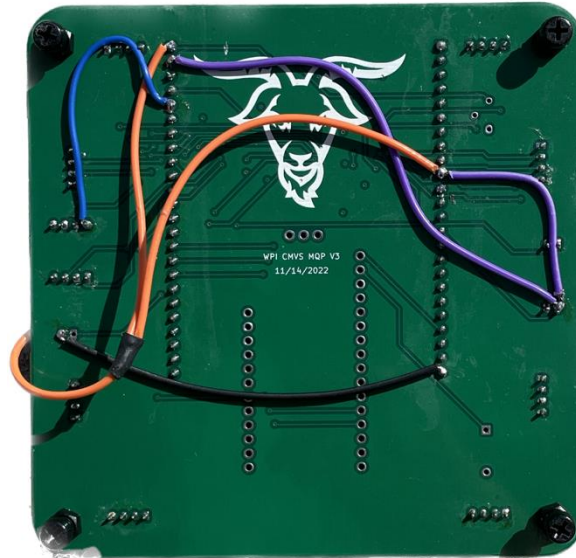


Figure 12: Bottom view of the PCB with modifications

Once all the necessary modifications were done on the PCB as shown on figure 12, the device operated as desired with all peripheral components attached. This included all 9 light sensors, RGB LED, blue LED, reset button, and power button.

4.1.3 Light Sensor Testing

We tested several different external parameters that could affect the accuracy and performance of the light sensors. The parameters that we identified and tested include: the LED on the light sensor, the sensor housing, sensor interdependency, and sensor calibration. We made the necessary modifications if the light sensors were greatly affected.

While we were testing the light sensors, we noticed that there is a small but bright green LED on each light sensor, indicating it is receiving power. We were curious if the light emitting from these LEDs were affecting the readings of the light sensors. To test whether this was the case or not, we tested our light sensors in a dark room. Given that the light sensors usually measure between 0-1000Lux, it should be very close to 0. We got the initial reading from the sensor in its natural state. It was about 1-2Lux. When we blocked out as much light from the LED using a Q-tip the readings only changed slightly, if at all, to 0-1Lux. We can conclude that the green LEDs on our light sensors do not significantly affect the measurements.

Once we integrated our light sensors into the light sensor housings we also wanted to see if the quartz glass pane and housing would influence the light readings. It does seem to have a slight effect of about 50Lux. This value is negligible as light readings outdoors are usually in the tens of thousands.

When testing the light sensors by covering each sensor individually and checking the output of all 9 sensors, we noticed that the light sensors connected to the first multiplexer changed their values when we covered the ninth sensor attached to the second multiplexer. This showed that there was an interdependency between the ninth sensor and the other 8 sensors. The source of the error was that we had forgotten to change the I2C address of the second multiplexer to be different from the address of the first one. Once we changed the address of the second multiplexer, we covered each sensor individually again, and this time only the value on the covered sensor decreased, so the interdependency was eliminated. Whenever we set up the system in the future, we repeated this test to make sure there was no sensor interdependency affecting the data.

We noticed while testing the light sensors outdoors, the output was maxing at a value of 37889 Lux. We were able to determine this as we tested the sensors with no shadow casting over them, slight shadow casting over them, and dark shadow casting.



Figure 13: Light sensors outdoors, exposed to partly cloudy suns around midday. Going from left to right is no shadow cast, slight shadow cast, and dark shadow cast

There was a noticeable difference in Lux values between the no cast and slight cast but comparing the slight cast and dark cast, the Lux values were the same. This signaled to us that the sensors needed to be recalibrated. We changed two aspects of the sensor, gain and integration time. Gain is the amount of amplification that is applied to the signal being measured from the sensors. We changed the sensors gain from medium to low as we recorded Lux values on a magnitude of 100x greater outdoors compared to indoors. We also changed the integration time, which is the period of time the sensor is sampling and averaging the data points. We decreased the integration from the initial 300ms to 100ms. This did slightly improve the values measured but the light sensors still maxed out when exposed to direct sunlight from late morning to late afternoon. We discovered that the sensors could only detect up to about 88,000 Lux, while direct sunlight can reach about 130,000 Lux [56],[57]. We covered the sensor windows with a neutral density 3 gel filter sheet (used in photography) to reduce the intensity of light by about 48.2% and applied a multiplier to the sensor values of 2.075 to compensate for the reduction [58]. This fully prevented the sensors from maxing out in bright sunlight.

4.1.4 Battery and Solar Panel Testing

Power is an integral part of our device as it is designed to be powered solely from a single 2000mAh 3.7V Lithium-Ion Polymer battery. We first characterized the battery with the CMVS system running in its normal operation and with the normal loads. This is the controlled test in which we later compared to results from similar tests with specific parameters modified.

For the control test, we powered the CMVS system from a fully charged, 2000mAh 3.7V, LiPo battery in a temperature-controlled room and let the system run for 2 hours; we monitored the voltage every 5 minutes. The system was initially tested indoors to be able to characterize the battery under rated conditions (20°C - 60°C). Table 2 and figure 14 summarizes the results found.

Table 2: Summary of Results for Controlled Test

	Controlled Test
Initial Battery Voltage	4.04V
Testing Time	2 hours
Average Room Temperature	26.04°C
Ending Battery Voltage	3.87V

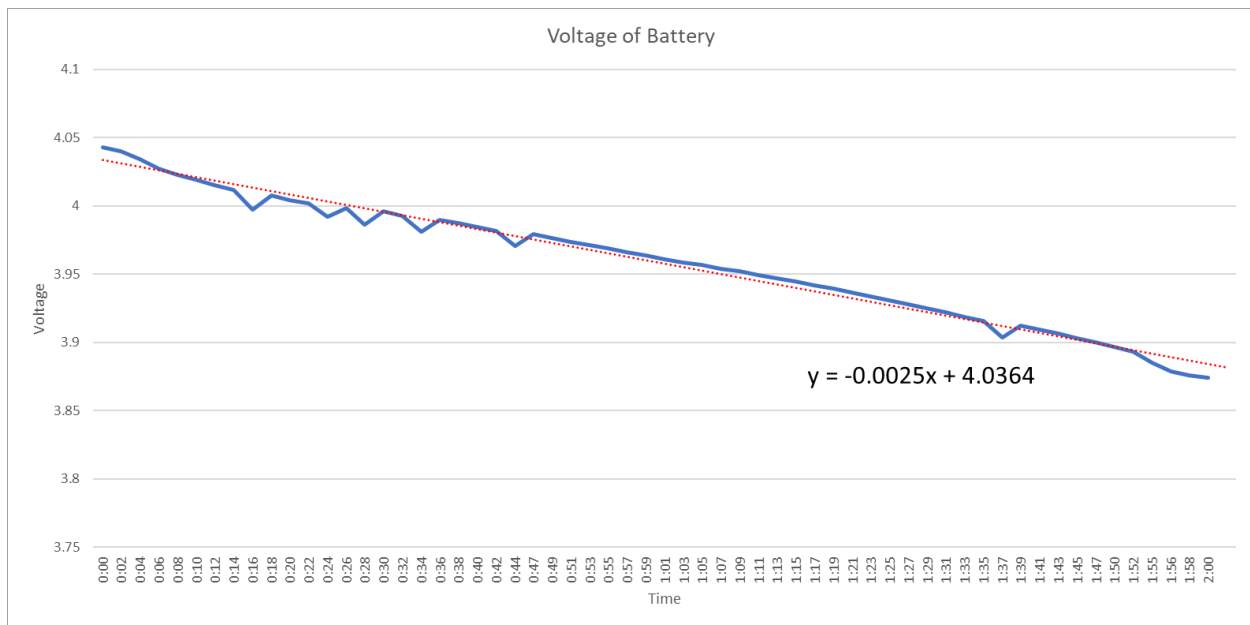


Figure 14: Battery Voltage over 2 hours indoors with linear prediction line

Based upon the sample voltage data collected over 2 hours, the predicted battery life would be 6 hours given all parameters are the same. The estimated rate of discharge is about $-2.5\text{mV}/\text{min}$. This is quite off to the estimated calculated battery life of 120 hours, or 5 days. This significant decrease maybe due to unexpected current draw from peripheral components.

We wanted to integrate a solar panel to be able to charge the battery and thus extend the battery life even longer. A similar test was conducted as described previously to determine whether

the 2.5W, 5V, 500mAh solar panel was generating enough current to charge the LiPo battery when the CMVS system was running.

To understand the impact of the solar panel we kept every parameter of the experiment the same as before except for the initial battery voltage which was partially discharged. A bright lamp was used in place of the Sun.

Table 3: Summary of Results and Comparison for Solar panel test

	Controlled Test	Battery with Solar Panel
Initial Battery Voltage	4.04V	3.83V
Measured Illuminance	NA	6500 Lux
Testing Time	2 hours	2 hours
Room Temperature	26.04°C	25.19°C
Ending Battery Voltage	3.87V	3.96V

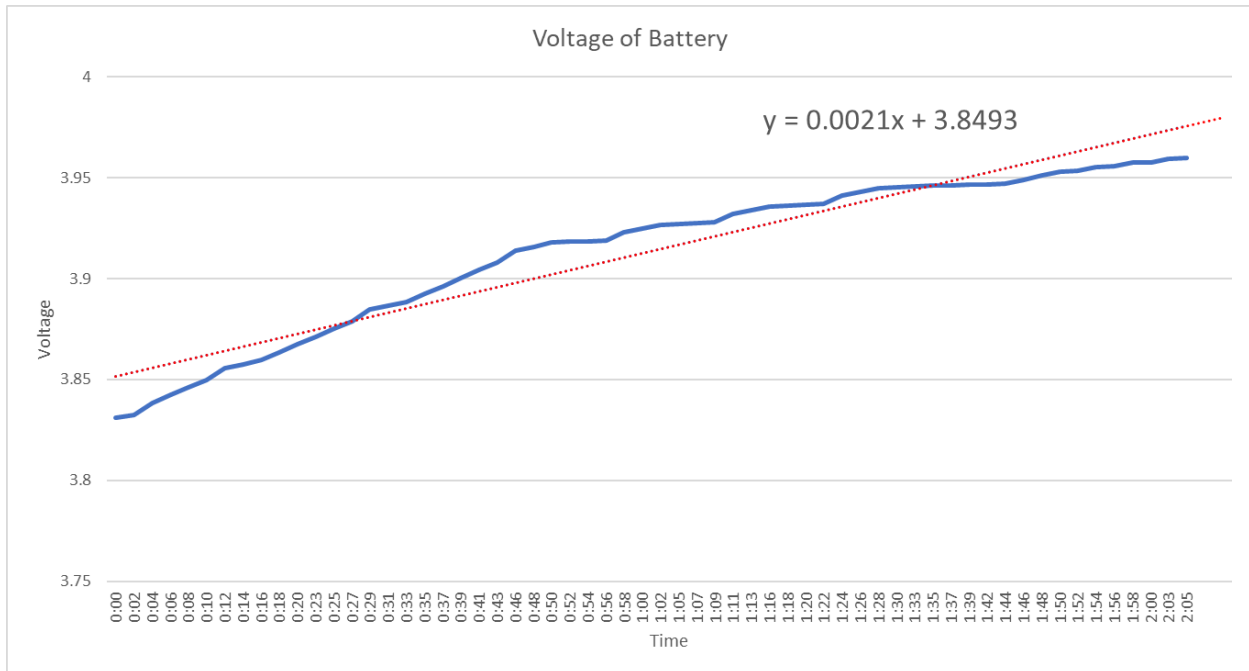


Figure 15: Battery Voltage with Solar panel over 2 hours indoors with linear prediction line

Based upon the sample voltage data collected over 2 hours, the CMVS system with the solar panel will charge the battery sufficiently to keep the battery at a constant voltage level or slowly increase the voltage depending on the light intensity. Based upon the measured Lux value

from the light source used, a Lux value of 6500 would be needed for a charge rate of about 2.1mV/minute. With this Lux value, it would take a little over 7 hours to completely charge the battery from a dead state. Thus, the solar panel is sufficient to charge the battery at or above the charge rate of 2.1mV/minute, when outdoors and exposed to sunlight, as the sun can output anywhere from 30,000 – 100,00 Lux. The battery would most likely fully charge within a couple of hours from a dead state when exposed to sunlight.

Given that the CMVS system will be exposed to temperatures much lower than the rated minimum temperature of 20°C when outdoors a test similar to the control test was conducted to determine how greatly the battery life is impacted when exposed to extremely low temperatures.

To understand the impact of temperature, we kept every parameter of the experiment the same as the control except for the ambient temperature the system was exposed to. This was done by running the system outdoors during a cold day in which the measured average temperature was much less than the rated minimum temperature.

Table 4: Summary of Results and Comparison for Outdoor test. Initial battery voltage for the control test is starting at the same voltage as the outdoor test to easily compare change of voltage over same time interval

	Controlled Test	Outdoor Test
Initial Battery Voltage	3.93V	3.93V
Average Temperature	26.04°C	11.54°C
Total Time system was on	1.25 hours	1.25 hours
Ending Battery Voltage	3.83V	3.77V
Discharge Rate	-2.7mV/min	-4.0mV/min

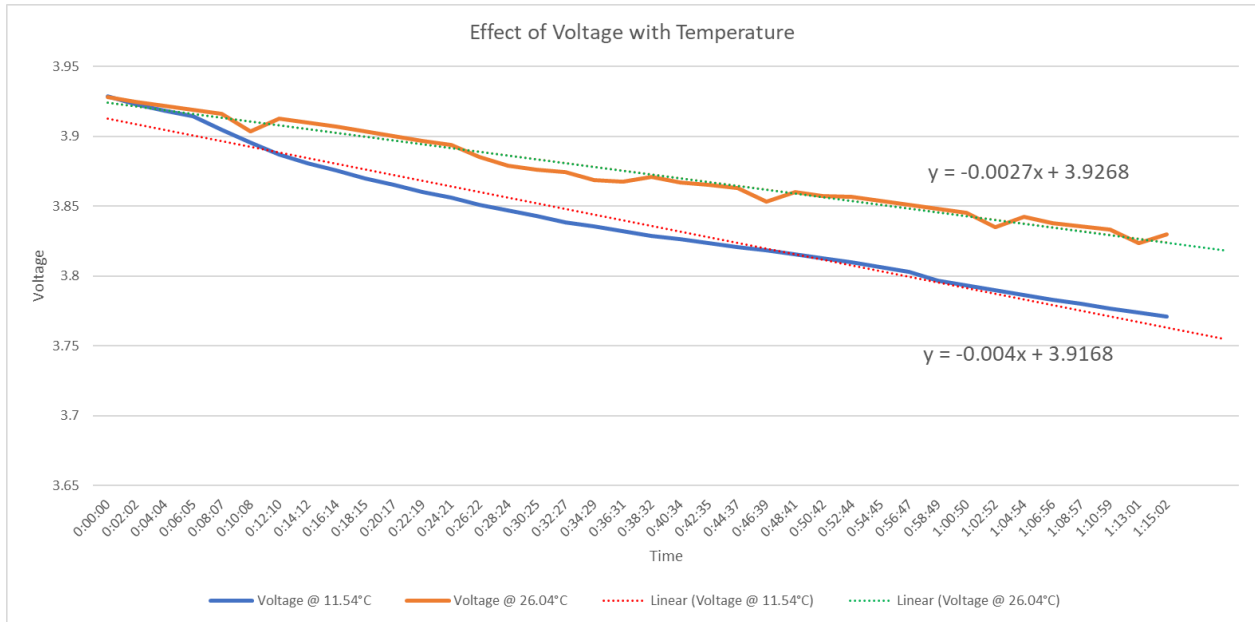


Figure 16: Battery Voltage over 1.25 hours outdoors with linear prediction line

While testing outdoors, we noticed that the LiPo batteries performance decreased sharply as seen in figure 16. The battery’s ending voltage after the 1.25 hour testing time was 3.77 voltage with an average temperature of 11.54°C. The estimated rate of discharge is about $-4.0\text{mV}/\text{min}$. Given this, the predicted battery life would be about 3.75 hours exposed to average temperatures of 11.54°C, a significant decrease from the control test of 6 hours. Temperature has a large impact on the performance of the battery, with temperatures outside the rated range sharply decreasing the lifespan of the battery.

Based upon these tests, the system being powered solely by the 2000mAh 3.7V Lithium-Ion Polymer battery does not seem realistic as it has only a 6 hour lifespan in ideal testing conditions, with temperature greatly reducing the performance when it is outside the rated range of 20°C - 60°C. The solar panel can charge the battery, even when it is slightly overcast, helping to increase the battery life. But this is not reliable as the system needs to be operating during very overcast or cloudy days as well. Thus, given the current setup, this system would work best in climates that are warm and sunny year-round, resulting in minimal maintenance needed to charge or replace the battery.

4.1.5 Cellular Connection and Strength Testing

We tested the cellular connection quality in our lab on the third floor of Atwater Kent and at three different locations on the roof of East Garage: at one end close to the solar panels, in the middle, and at the other end as shown below.

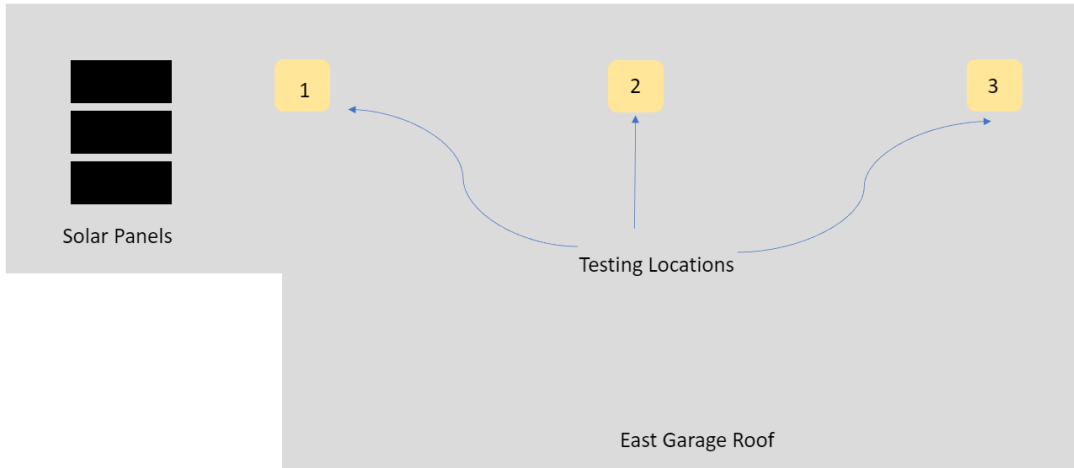


Figure 17: Cellular Testing Setup located on East Garage Roof

There were four main cellular connection quality parameters: RSSI, RSRP, SINR, and RSRQ. The basic descriptions and quality ranges of each of the parameters is shown below.

Table 5: Description of Cellular parameters tested [60].

The data elements to look at most closely in the responses from the previous two methods include:

- `bars` - The number of "bars" of cellular signal quality (0-4).
- `rat` - The "Radio Access Technology" in use.
- `band` - The LTE band in use (useful for matching up with the appropriate LTE antenna - this data is not available in `_session.qo`).
- `rssi` - The "Received Signal Strength Indicator" is a measure of cellular signal strength. The value is a negative number, where the higher the value (a.k.a. the closer to `0`), the stronger the signal.
- `rsrp` - The "Reference Signal Received Power". The value is a negative number, where the higher the value (a.k.a. the closer to `0`), the stronger the power.
- `sinr` - The "Signal to Interference and Noise Ratio". The higher the value, the better the signal strength.
- `rsrq` - The "Reference Signal Received Quality" value. The value is a negative number, where the higher the value (a.k.a. the closer to `0`), the better the signal quality.

As a rough guide, use the following table to measure the quality of your cellular connection:

Quality	RSSI (dB)	RSRP (dB)	SINR (dB/5)	RSRQ (dB)
Excellent	> -65	> -84	> 162	> -5
Good	-75 to -65	-102 to -85	150 to 162	-9 to -5
Fair	-85 to -75	-111 to -103	135 to 150	-12 to -9
Poor	< -85	< -111	< 135	< -12

The output of our tests are shown below. RSSI was good or better in all locations, RSRP was fair or better in all locations, SINR was poor in all locations, and RSRQ was poor in all locations except right next to the solar panels, where we did most of our testing. We were able to send data over cellular in all locations, but the low SINR value could be the reason why the cellular connection occasionally cut out for 10-30 minutes during testing.

```

Notehub sync attempt 1m ago
  iccid : 89011703278520569685 ,
  "imsi": "310170852056968",
  "imei": "864622040502524",
  "modem": "BG95M1LAR02A04_01.001.01.001",
  "band": "LTE BAND 12",
  "rat": "emtc",
  "rssi": -73,
  "rsrp": -106,
  "sinr": 87,
  "rsrq": -19,
  "bars": 1,
  "mcc": 310,
  "mnc": 410,
  "lac": 1026,
  "cid": 16090386,
  "updated": 1675719415
}

```

Figure 18: Cellular connection in AK316

```

Notehub sync attempt 2m ago
  "req": "card.wireless"
}
{
  "status": "{network-up}",
  "count": 1,
  "net": {
    "iccid": "89011703278520569685",
    "imsi": "310170852056968",
    "imei": "864622040502524",
    "modem": "BG95M1LAR02A04_01.001.01.001",
    "band": "LTE BAND 12",
    "rat": "emtc",
    "rssi": -69,
    "rsrp": -104,
    "sinr": 60,
    "rsrq": -3,
    "bars": 1,
  }
}

Sync needed - Notehub sync attempt - ...
{
  "status": "{network-up}",
  "count": 2,
  "net": {
    "iccid": "89011703278520569685",
    "imsi": "310170852056968",
    "imei": "864622040502524",
    "modem": "BG95M1LAR02A04_01.001.01.001",
    "band": "LTE BAND 12",
    "rat": "emtc",
    "rssi": -67,
    "rsrp": -100,
    "sinr": 61,
    "rsrq": -19,
    "bars": 2,
    "mcc": 310,
    "mnc": 410,
    "lac": 1026,
  }
}

Notehub sync attempt 52s ago
  "imsi": "310170852056968",
  "imei": "864622040502524",
  "modem": "BG95M1LAR02A04_01.001.01.001",
  "band": "LTE BAND 12",
  "rat": "emtc",
  "rssi": -71,
  "rsrp": -104,
  "sinr": 80,
  "rsrq": -20,
  "bars": 1,
  "mcc": 310,
  "mnc": 410,
  "lac": 1026,
  "cid": 16090386
}

```

Figure 19: Cellular connection at testing locations 1-2-3 depicted in fig. 17 from left to right

4.1.6 Data Analysis Testing

When we attempted to record data outside, we noticed that the sensors were sending maximum brightness no matter what we changed the settings to. There was a noticeable decrease as the sun set, but during the day the output was the same regardless of if it was sunny or overcast. We tried switching to new sensors, but were unable to get them to work with our current setup, so we decided to use filters to reduce light levels instead. We were able to get all sensors working with this setup, but could not get them to function consistently enough to get enough data to analyze.

4.1.7 AI Image Processing Testing

In addition to light sensors, we also decided to explore cloud tracking via image processing. We started testing with data from the Total Sky Imager and used sparse and dense optical flow to track the clouds [61],[62]. However, we realized that dense optical flow did not have any means of pixel tracking between more than two frames, so it would be impossible to find the speed or direction of the cloud motion. We ended up taking several 5-minute videos of the sky with different amounts of cloud cover using an iPhone 13 camera, using sparse optical flow to find the direction and rate at which the clouds crossed the lens.



Figure 20: Lines showing clouds movement tracked with sparse optical flow

The sparse optical flow algorithm resets every 1000 frames. The slope is calculated based on the difference between the x and y positions of the pixels every 1000 frames, which was then averaged for every 1000 frame chunk in the video. With a video on a partly cloudy day that can be seen in Figure 20, we found that the clouds were travelling at an average slope of 0.55, which was determined as if each image was a coordinate plane of pixels starting at (0,0) in the top left corner. This slope is equivalent to -28 degrees from the horizontal axis in the video. Since the top of the frame was facing 15 degrees from north, the direction the clouds were travelling in was -103 degrees from north, or -13 degrees from east.

4.2 Field Testing

The entire CMVS system was set up on the roof of East Hall garage for three nights and two full days for field testing. Over the course of about 72 hours, the overall low temperature was 10°F, the overall high temperature was 34°F, the average temperature was about 30°F, and the total snowfall was about 2.2 inches. The system was outside from around 6:00PM on Friday, 2/24/2023 to 6:00PM on Monday, 2/27/23. The weather during this time is shown in Figure 21 below [59].

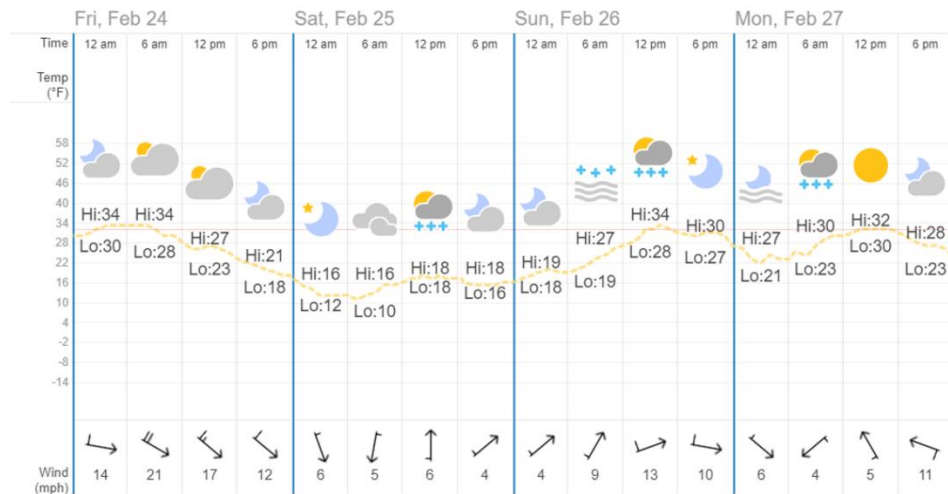


Figure 21: Weather and temperature data from February 24-27, 2023

Some of the connectors had weak or irregular connections both outside and inside the housing. The LED indicator would turn off on some of the sensors for a few seconds and then turn back on once the connector or wire positioning was adjusted. These issues were most likely exacerbated by the cold temperatures and icy conditions. A significant amount of snow covered

the housing enclosures and had to be brushed off. The housing enclosures remained dry inside except for slight condensation that was visible on the inside of some of the housing enclosures. The sensors and PCB continued to function properly. One notable drawback of the main housing when used in cold weather conditions is that the extremely cold temperatures greatly decrease the lifespan of the battery, requiring the use of a larger battery that does not fit inside the main housing.



Figure 22: Field testing conditions on Roof of East Garage

4.2.1 Portability

The CMVS system consists of the main housing – a 3D-printed box that is 100mm tall with a 137.5mm square base – on a thick wooden block – connected to 8 light sensor housings – much smaller boxes also mounted on thick wooden blocks – via ~10 foot long wires covered in protective plastic sheathing. The system does not weigh so much that it is uncomfortable to carry, weighing in at 11 pounds and 12 ounces total. It is slightly unwieldy, though, due to the long wires with the sensor housings at the ends. The method for carrying the device so far has been to bundle the individual wires with rubber bands and put it in a very long and wide cardboard box, or wrapping the wires around the main housing, putting the sensor housings close to the main housing, and carrying the whole system on the big circle of cardboard we use to determine what angle each sensor should be at. Overall, it is slightly unwieldy but not very heavy, and it is portable enough for one person to carry it. The system could likely be made more portable if the light sensors were

mounted at a slight outward angle on the main housing so that they could see enough of the sky but also so that the angle did not affect the data, however we did not have time to test this setup.

4.2.2 Reliability

The system had some reliability issues, namely the cellular card connection and the wire connectors between the main PCB and the light sensors. As discussed in the cellular connection section, some of the measured parameters for cellular connection quality were poor on the roof of East Garage. The cellular card was able to connect most of the time, but occasionally the connection cut out for 10-30 minutes at a time, which disrupted our ability to acquire the necessary 200 uninterrupted data points for our analysis code (approximately 1.5 hours of data collection).

The wire connectors also hindered our ability to collect data. We used the wires from the previous group but replaced the connectors they were using with the JST SH/PH connectors we needed for our system as discussed in the wire connectors section of the methods. Some of the old connections and some of our new connections were not quite strong enough, so we often had to redo wire splices and move the wires around until a solid connection was made. Sometimes the connections would fail while collecting data despite not being moved at all, which made collecting data over long periods of time difficult. In the future, the old wires should be replaced and all of the connections should be made and verified to be very strong to avoid connection issues. Overall though, we were able to collect data for hours at a time without any connection issues

4.2.3 Usability

Adding external buttons and LEDs to the housing allowed for the housing to remain enclosed and weather resistant while also allowing for some user interaction and feedback from the device. The system was also modified by adding external weatherproof wire connectors to allow for easier set up and minimize the number of times the housing needed to be opened while on site. Some of the connectors had weak or irregular connections both outside and inside the housing. Because the wire holes in the housing were sealed with silicone caulking, it was more difficult to address the connection issues. Inside the main housing, the wires had to bend slightly to connect to the PCB, making it more difficult to ensure reliable connections. Better connectors would most likely not have as many issues. With the current connectors, the main housing could be made larger to allow for easier connections, and a less permanent sealing method could be used on the wire holes to allow for connector-related troubleshooting.

5 Conclusion

We have designed and thoroughly tested the CMVS prototype that can collect all the necessary data to predict cloud motion based on the current algorithm being implemented. There can be improvements that can be made to further increase the capabilities of this device, improve the accuracy of the algorithm, and improve battery life. Overall, our device is portable, reliable, and easy to use. It is most useful in climates that are warm and sunny year-round, resulting in minimal maintenance and best performance. This device is a good framework to easily begin to analyze cloud motion.

6 Recommendations

6.1 Electronics

Based upon field testing the complete system, changing the JST connectors that are on the current PCB would improve the accessibility of connecting and disconnecting the wires. Figure 23 and 24 is an updated design of the PCB eliminating the issues described previously that were identified. This design is a good template to use for further iterations and improvements.

Including additional sensors, like an imaging sensor, would be necessary if implementing an image recognition approach to predict cloud motion using optical flow or another method. We currently have a 180-degree camera that can attach to a Raspberry Pi, but it is not integrated into the system. Any additional hardware added onto or replaced should ideally be low powered and draw minimal current to not overload the system and deplete the battery quickly.

Based upon the battery tests conducted, there are several power saving techniques to implement which will improve the battery lifespan. One of the obvious and simplest things to do is to replace the current battery with a larger capacity LiPo battery. Keep in mind that with a larger battery, more space is needed in the main housing and there will be added weight to the system. Also as mentioned before, anything connected to the system that needs power should draw minimal current.

6.2 Hardware

As previously mentioned, the main housing could be made larger to accommodate better connections, to allow for the option of using a larger battery, and to allow easier access to the PCB in general. The current main housing model has a square base that measures 137.5mm and has a height of 100mm. We recommend increasing the scale of the model by a factor of at least 0.25, making the base about 172mm and the height 125mm. Including threaded inserts for the lid-to-base screw holes in the main and sensor housing is also recommended to increase the durability and longevity of the prints.

6.3 Software

Besides the electrical components, improving power saving methods in the systems program can be implemented. This includes programming a sleep mode which triggers based upon the time the sun sets. The LEDs on the system are some of the largest sources of current draw. The

operation of these LEDs can be modified to also improve power efficiency. Currently, the LED indicating system power will stay on as long as the system is on. Changing the operation of this LED to turn off or blink intermittently will conserve some power. This also applies to the LEDs that are located on the light sensors.

We would recommend improving on the model used to predict cloud movement using the cameras. We would recommend continuing with sparse optical flow, because that is currently the most developed algorithm for the project, and there is a large amount of data from other projects that used it to track cars and other objects. Additionally, ThingSpeak gets data from Notehub unreliably, so figuring out a way for Thingspeak to receive data at regular intervals would improve the predictions from the light sensors.

References

- [1] Center for Climate and Energy Solutions. (2022). *Renewable Energy*. <https://www.c2es.org/content/renewable-energy/>
- [2] U.S. Energy Information Administration. (2022, February). *What is U.S. electricity generation by energy source?* <https://www.eia.gov/tools/faqs/faq.php?id=427&t=3>
- [3] Our World in Data. (2022, July 8). *Installed solar energy capacity*. <https://ourworldindata.org/grapher/installed-solar-pv-capacity>
- [4] Our World in Data. (2020, September 29). *Solar PV module prices*. <https://ourworldindata.org/grapher/installed-solar-pv-capacity>
- [5] International Renewable Energy Agency (2021, June). *Renewable Power Generation Costs in 2020*. <https://www.irena.org/publications/2021/Jun/Renewable-Power-Costs-in-2020>
- [6] Jaeger, J. (2021, September 20). *Explaining the Exponential Growth of Renewable Energy*. World Resources Institute. <https://www.wri.org/insights/growth-renewable-energy-sector-explained>
- [7] Nwaigwe, K., Mutabilwa, P., & Dintwa, E. (2019). An overview of solar power (PV systems) integration into electricity grids. *Materials Science for Energy Technologies*, 2(3), 629-633. <https://doi.org/10.1016/j.mset.2019.07.002>
- [8] Holguin, J. P., Rodriguez, D. C. & Ramos, G. (2020). Reverse Power Flow (RPF) Detection and Impact on Protection Coordination of Distribution Systems. *IEEE Transactions on Industry Applications*, 56(3), 2393-2401, doi: 10.1109/TIA.2020.2969640.
- [9] Masoum, M. & Fuchs, E. (2015). Introduction to Power Quality. *Power Quality in Power Systems and Electrical Machine*, 2. <https://doi.org/10.1016/C2013-0-18758-2>
- [10] Halpin, M.S. & Card, A. (2007). Power Quality. *Power Electronics Handbook*, 2. <https://doi.org/10.1016/B978-0-12-088479-7.X5018-4>
- [11] Office of Energy Efficiency & Renewable Energy. (2017, October 12). *Confronting the Duck Curve: How to Address Over-Generation of Solar Energy*. <https://www.energy.gov/eere/articles/confronting-duck-curve-how-address-over-generation-solar-energy>
- [12] Spanias, A. S. (2017). Solar energy management as an Internet of Things (IoT) application. *8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1-4. doi: 10.1109/IISA.2017.8316460.
- [13] Sukič, P., & Štumberger, G. (2017). Intra-Minute Cloud Passing Forecasting Based on a Low Cost IoT Sensor—A Solution for Smoothing the Output Power of PV Power Plants. *Sensors*, 17(5), 1116. <https://doi.org/10.3390/s17051116>

- [14] Yankee Environmental Systems. (2004). *Automatic Total Sky Imager Model TSI-880*. <https://www.yesinc.com/resource/products/skyimaging/tsi-880ds.pdf>
- [15] Ryu, A., Ito, M., Ishii, H. & Hayashi, Y. (2019). Preliminary Analysis of Short-term Solar Irradiance Forecasting by using Total-sky Imager and Convolutional Neural Network. *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*, 627-631. doi: 10.1109/GTDAasia.2019.8715984.
- [16] Park, S., Kim, Y., Ferrier, N. J., Collis, S. M., Sankaran, R., & Beckman, P. H. (2021). Prediction of Solar Irradiance and Photovoltaic Solar Energy Product Based on Cloud Coverage Estimation Using Machine Learning Methods. *Atmosphere*, 12(3), 395. <https://doi.org/10.3390/atmos12030395>
- [17] Cai, C. & Aliprantis, D. (2013). Cumulus Cloud Shadow Model for Analysis of Power Systems with Photovoltaics. *IEEE Transactions on Power Systems*, 28(4), 4496–506. doi:10.1109/TPWRS.2013.2278685.
- [18] Lu, Z., Wang, Z., Li, X., & Zhang, J. (2021). A Method of Ground-Based Cloud Motion Predict: CCLSTM + SR-Net. *Remote Sensing*, 13(19), 3876. <https://doi.org/10.3390/rs13193876>
- [19] Ferreira, J., Lewis, T., Sauter, E. (2022). *Cloud Motion Vector Sensor System Monitoring and Predicting Output Power of a Photovoltaic System in Real-Time* (Publication No. 53961) [Undergraduate major qualifying project, Worcester Polytechnic Institute]. Digital WPI.
- [20] Newark. (n.d.). *Light Sensors*. <https://www.newark.com/sensor-optical-light-sensor-technology?ICID=I-CT-TP-BROWSE-5>
- [21] Brownson, J. (2020). *Measurement devices: Technology of irradiance transducers*. PennState. <https://www.e-education.psu.edu/eme810/node/682>
- [22] Feister, U., Grewe, R., Gericke, K. (2018, January 26). A method for correction of cosine errors in measurements of spectral UV irradiance, *Solar Energy*, Volume 60, Issue 6, 1997, Pages 313-332. [https://doi.org/10.1016/S0038-092X\(97\)00030-3](https://doi.org/10.1016/S0038-092X(97)00030-3).
- [23] Gums, J. *Types of Temperature Sensors*. Digi-Key. [https://www.digikey.com/en/blog/types-of-temperature-sensors#:~:text=There%20are%20four%20types%20of,based%20integrated%20circuits%20\(IC\).](https://www.digikey.com/en/blog/types-of-temperature-sensors#:~:text=There%20are%20four%20types%20of,based%20integrated%20circuits%20(IC).)
- [24] Dhanalakshmi, S., Chakravartula, V., Narayanamoorthi, R., Kumar, R., Dooly, G., Duraibabu, D. B., Senthil, R. (2022). Thermal management of solar photovoltaic panels using a fibre Bragg grating sensor-based temperature monitoring. *Case Studies in Thermal Engineering*, Volume 31. <https://doi.org/10.1016/j.csite.2022.101834>.
- [25] Cadence PCB Solutions. (n.d.). *How to Measure the Power Consumption of a Circuit*. <https://resources.PCB.cadence.com/blog/2021-how-to-measure-the-power-consumption-of-a-circuit>
- [26] Jepson, C. (n.d.) *An Introduction to Low Power IoT*. Particle Industries. <https://www.particle.io/iot-guides-and-resources/low-power-iot/>

- [27] Author, N. C. (2021, March 25). *Maximum Power Point Tracking (MPPT) algorithms*. Imperix. <https://imperix.com/doc/implementation/maximum-power-point-tracking-mppt>
- [28] Chafle, S. R., & Vaidya, U. B. (1970). Incremental Conductance MPPT Technique FOR PV System. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Energy*, 2(6), 2719–2726. <https://www.rroij.com//open-access/incremental-conductance-mppt-techniquefor-pv-system.php?aid=41625>
- [29] Office of Energy Efficiency and Renewable Energy. (n.d.). *Solar Performance and Efficiency*. Energy.gov. <https://www.energy.gov/eere/solar/solar-performance-and-efficiency>
- [30] *Solar Panel Temperature | Effect on performance*. (2016, September 26). Solar Calculator. <https://solarcalculator.com.au/solar-panel-temperature/>
- [31] Ahmad, J. (2010, October 1). *A fractional open circuit voltage based maximum power point tracker for photovoltaic arrays*. IEEE Xplore. <https://doi.org/10.1109/ICSTE.2010.5608868>
- [32] IBM Cloud Education. (2020, August 17). *What are Neural Networks?* www.ibm.com; IBM. <https://www.ibm.com/cloud/learn/neural-networks>
- [33] Ferreira, J., Lewis, T., Sauter, E. (2022, March 25). *Cloud motion vector sensor system: Monitoring and predicting output power of a photovoltaic system in real-time*. Worcester Polytechnic Institute.
- [34] Zerynth Docs. (n.d.). *ESP-WROOM32*. Zerynth. https://olddocs.zerynth.com/r2.6.2/official/board.zerynth.cvm_espwroom32/docs/index.html
- [35] Adafruit. (n.d.). *Adafruit HUZZAH32 – ESP32 Feather Board*. <https://www.adafruit.com/product/3591>
- [36] Newark. (n.d.). *Light Sensors*. <https://www.newark.com/sensor-optical-light-sensor-technology?ICID=I-CT-TP-BROWSE-5>
- [37] Lady Ada. (n.d.). *Adafruit TSL2591 High Dynamic Range Digital Light Sensor – Pinouts*. <https://learn.adafruit.com/adafruit-tsl2591/pinouts>
- [38] ELP. (n.d.). *ELP 180 Degree Fisheye Lens 2MP 1080P FULL HD Camera Module USB2.0 OV2710 COLOR SENSOR*. <http://www.webcamerausb.com/elp-180-degree-fisheye-lens-2mp-1080p-full-hd-camera-module-usb20-ov2710-color-sensor-p-190.html>
- [39] Adafruit. (n.d.). *Lithium Ion Polymer Battery - 3.7v 1200mAh*. <https://www.adafruit.com/product/258>
- [40] Tractus3D. (2020). *TPU Material*. <https://tractus3d.com/materials/tpu/>

- [41] WPI Makerspace. (2022, September 6). *Material Selection Guide for FDM Printing*. https://canvas.wpi.edu/courses/9342/files/4887776/download?download_frd=1
- [42] Chapman, A. (2022, March 1). *How to 3D print waterproof parts*. Ultimaker. <https://ultimaker.com/learn/3d-print-waterproof-parts>
- [43] 3D HUBS B.V. (2022). *How do you design enclosures for 3D printing? A step-by-step guide*. <https://www.hubs.com/knowledge-base/enclosure-design-3d-printing-step-step-guide/>
- [44] International Electrotechnical Commission. (n.d.). *IP ratings*. <https://www.iec.ch/ip-ratings>
- [45] Cadence PCB Solutions. (n.d.). *Common Methods for Waterproofing Electronics Materials*. <https://resources.pcb.cadence.com/blog/2020-common-methods-for-waterproofing-electronics-materials>
- [46] Kočí, J. (2021, July 29). *Watertight 3D printing part 2: Airtight closable models*. Prusa Research. https://blog.prusa3d.com/watertight-3d-printing-part-2_53638/
- [47] BCN3D. (2020, June 18). *Waterproofing 3D Prints: 5 Easy Solutions*. <https://www.bcn3d.com/waterproofing-3d-prints-5-solutions-to-creating-water-resistant-models/>
- [48] Breiner Innovative. (n.d.). *Six Materials For Great Gaskets*. <https://breinerco.com/six-materials-for-gaskets/>
- [49] Modus Engineering Team. (2021, January 13). *27 Materials for Custom Gaskets and Their Purposes*. Modus. <https://www.modusadvanced.com/resources/blog/custom-gasket-materials>
- [50] Home Depot. (n.d.). *25 ml ClearWeld Quick-Set Epoxy Syringe*. <https://www.homedepot.com/p/J-B-Weld-25-ml-ClearWeld-Quick-Set-Epoxy-Syringe-50112/204986141>
- [51] O'Connell, J. (2022, October 29). *How to Make Waterproof 3D Prints*. All3DP. <https://all3dp.com/2/waterproof-3d-print-pla/#:~:text=Epoxy%3A%20Another%20post-processing%20method,good%20candidate%20for%20waterproofing%20prints>
- [52] Beardow Adams. (n.d.). *What is a hot-melt adhesive?* <https://www.beardowadams.com/news-and-blog/blog/what-is-a-hot-melt-adhesive>
- [53] Digi-Key. (n.d.). *ZF Electronics KCA2ANA1BBB*. <https://www.digikey.com/en/products/detail/zf-electronics/KCA2ANA1BBB/2027285>
- [54] Digi-Key. (n.d.). *ZF Electronics KFB2ANA1BBB*. <https://www.digikey.com/en/products/detail/zf-electronics/KFB2ANA1BBB/2027294>
- [55] ThingSpeak. (n.d.). *ThingSpeak for IoT Projects*. <https://thingspeak.com/>

- [56] Lady Ada. (n.d.). *Adafruit TSL2591 High Dynamic Range Digital Light Sensor – Overview*. <https://learn.adafruit.com/adafruit-tsl2591/overview>
- [57] AON2. (n.d.). *Illuminance*. <https://www.aon2.co.uk/illuminance/>
- [58] FIEWSZIHU. (n.d.). *Lighting Neutral Density Gels Filter Sheet 16x20 inches Kit, ND3,ND6,ND9 for Photo Studio Video Flashlight Led Light Photography*. Amazon. https://www.amazon.com/dp/B08818V6Y2?psc=1&ref=ppx_yo2ov_dt_b_product_details
- [59] Time and Date. (n.d.). *Past Weather in Worcester, Massachusetts, USA — February 2023*. <https://www.timeanddate.com/weather/usa/worcester/historic?month=2&year=2023>
- [60] Blues Inc. (n.d.). *Diagnosing Cellular Connectivity Issues*. <https://dev.blues.io/guides-and-tutorials/notecard-guides/diagnosing-cellular-connectivity-issues/>
- [61] ARM. (n.d.). *Total Sky Imager*. <https://www.arm.gov/capabilities/instruments/tsi>
- [62] Kuklin, M. (2021, January 4). *Optical Flow in OpenCV*. LearnOpenCV. <https://learnopencv.com/optical-flow-in-opencv/>

8 Appendix

Detailed Data

2023-02-27T20:31:06+00:00	386	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:32:29+00:00	387	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:33:51+00:00	388	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:35:14+00:00	389	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:36:37+00:00	390	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:37:59+00:00	391	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:39:22+00:00	392	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:40:44+00:00	393	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:42:07+00:00	394	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:43:29+00:00	395	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:44:52+00:00	396	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:46:16+00:00	397	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:47:39+00:00	398	37889	37889	37889	37889	37889	37889	37889	37889
2023-02-27T20:49:01+00:00	399	37889	37889	37889	35750	37889	37889	37889	37889
2023-02-27T20:50:25+00:00	400	37889	37126	37889	34584	37889	37889	37889	37889
2023-02-27T20:51:46+00:00	401	37889	36073	37889	33732	37889	37889	37889	37889
2023-02-27T20:53:08+00:00	402	37889	36732	37889	34564	37889	37889	37889	37889
2023-02-27T20:54:32+00:00	403	37889	37889	37889	36347	37889	37889	37889	37889
2023-02-27T20:55:53+00:00	404	37889	37889	37889	37888	37889	37889	37889	37889
2023-02-27T20:57:16+00:00	405	37889	37889	37889	37290	37889	37889	37888	37888
2023-02-27T20:58:41+00:00	406	37889	37888	37695	35936	37422	37889	36125	36126
2023-02-27T21:00:03+00:00	407	37888	36270	35595	34705	35447	37096	34350	34344
2023-02-27T21:01:26+00:00	408	35567	34260	33347	32760	33226	34589	32039	32027
2023-02-27T21:02:48+00:00	409	33160	32223	31118	30613	30968	32415	29962	29961
2023-02-27T22:09:31+00:00	410	35319	32475	32861	31038	32686	34338	32061	32051
2023-02-27T22:10:11+00:00	411	7800	7734	7352	7384	7343	7498	7029	7028
2023-02-27T22:11:35+00:00	412	6705	6682	6340	6349	6281	6414	6012	6008
2023-02-27T22:12:57+00:00	413	5590	5622	5275	5289	5224	5355	5005	5004

Figure 23: Example Output data from ThingSpeak

Revised PCB Design

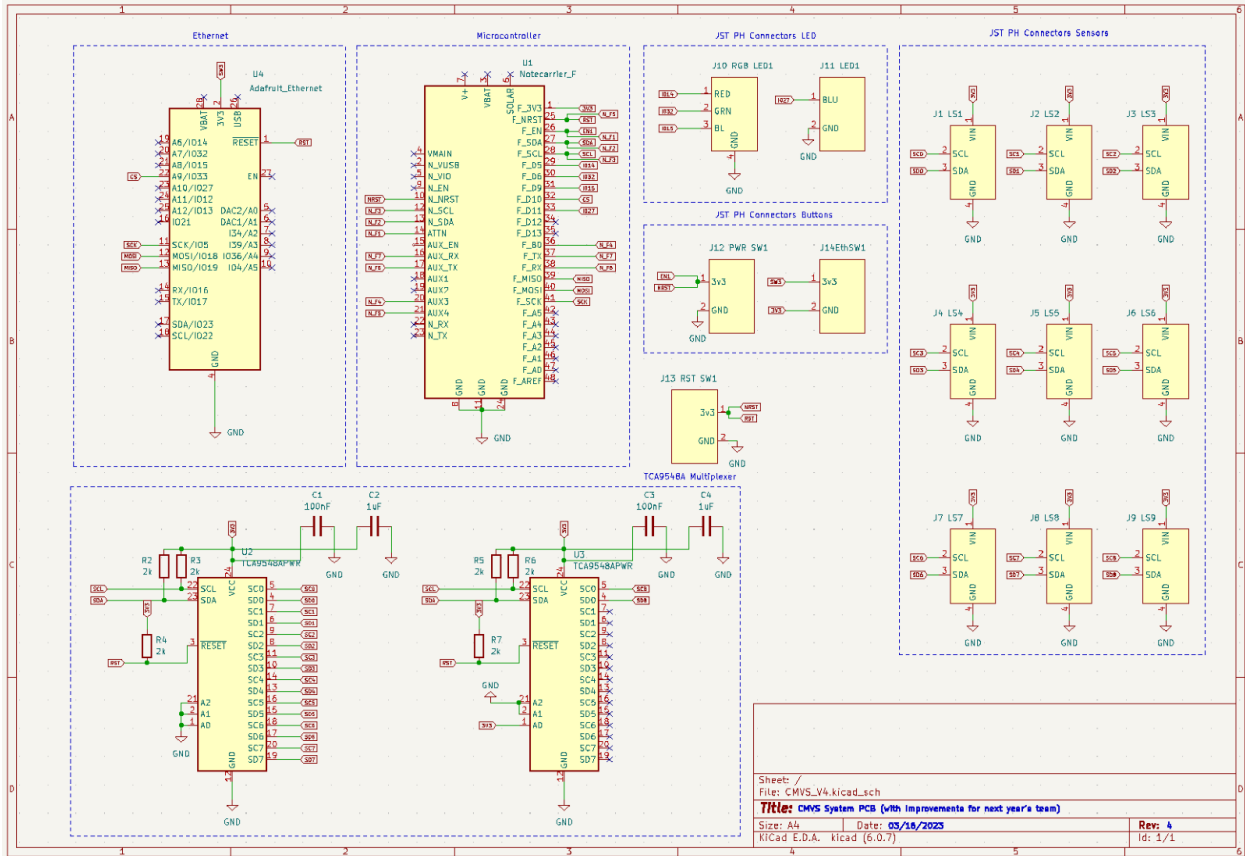


Figure 24: Revision 4 of Schematic

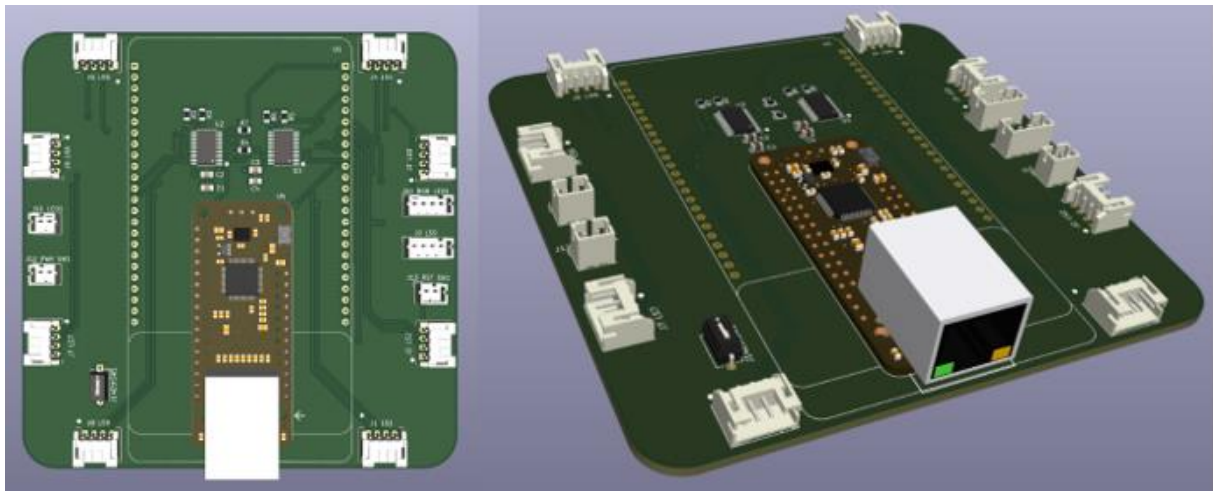


Figure 25: Revision 4 of PCB Layout

Revised Project Code

Other code used for the project can be found at: https://github.com/yayoi730/CMVS_2022

Final Code for TSL2590 Light Sensors

```
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <stdio.h>
#include <Notecard.h>

#include "Adafruit_TSL2591.h"
#define TCAADDR 0x70
#define TCAADDR1 0x74
#define PIN_RED 14 // GIOP14 /D5
#define PIN_GREEN 32 // GIOP32 /D6
#define PIN_BLUE 15 // GIOP15 /D9
#define LED_BLUE 27 // GIOP27 /D11

#define BLINK_INTERVAL 1000 // interval at which to blink LED (milliseconds)
#define BAT_VOLTAGE 4.1 // full battery life
#define THRESH_VOLTAGE 3.7 //point at which battery voltage should need to be charged soon

#define productUID "edu.wpi.oroockrohr:cmvs_cloud"

// Variables will change:
int ledState = 2; // ledState used to set the LED
int button_state = 0; // variable for reading the button status
boolean network = true;
boolean transition = false;
boolean reset = false;
int count = 0;
Notecard notecard;
Adafruit_TSL2591 tsl[9];
int sensorData[9];

unsigned long previousMillis = 0; // will store last time LED was updated
float currVoltage = 4; //stores the current voltage value of battery

void initialize_sensor(Adafruit_TSL2591 sensor, int sensor_num){
  sensor.setGain(TSL2591_GAIN_LOW); // 25x gain
  sensor.setTiming(TSL2591_INTEGRATIONTIME_100MS);
}

void tcselect (uint8_t i, int multaddr) {
```

```

if (i > 7) return;

Wire.beginTransaction(multaddr);
Wire.write(1 << i);
Wire.endTransmission();
}

void setup() {
  delay(2500);
  for(int i=0; i<9; i++){
    initialize_sensor(tsl[i],i);
  }
  Serial.begin(115200);

  pinMode(PIN_RED, OUTPUT);
  pinMode(PIN_GREEN, OUTPUT);
  pinMode(PIN_BLUE, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);

  notecard.begin();
  notecard.setDebugOutputStream(Serial);
  J *req = notecard.newRequest("hub.set");
  JAddStringToObject(req, "product", productUID);
  JAddStringToObject(req, "mode", "continuous");
  notecard.sendRequest(req);
  delay(500);
}
double sensorLuminance(Adafruit_TSL2591 sensor, int addr, int multaddr){
  delay(100);
  tcselect(addr,multaddr);
  Serial.print(addr);
  Serial.print(F(" : Irradiance=\t"));
  double l = sensor.getLuminosity(TSL2591_INFRARED);
  Serial.print(l);
  //Serial.println("-----");
  //delay(100);
  //Serial.println();
  return l;
}

void loop() {
  currVoltage = batLife();

```

```

//check battery voltage
if(network == true) //if bat is less than threshold then turn LED Red
{
  if(currVoltage <= THRESH_VOLTAGE)
  {
    redLed();
    ledState = 1;
  }
  else
  {
    greenLed(); //if battery p charged then keep led green
    ledState = 3;
  }
}
else //blinking led when network is not connected
{
  blink(ledState);
  if(currVoltage <= THRESH_VOLTAGE && transition == false)
  {
    ledState = 0;
    transition = true;
  }
  if(currVoltage > THRESH_VOLTAGE && transition == true)
  {
    ledState = 2;
    transition = false;
  }
}
Serial.printf("Voltage: %f\n", currVoltage); //prints out digital value of voltage
//delay(3000);

// control LED according to the state of button
if (reset == false) // if button is pressed
{
  digitalWrite(LED_BLUE, HIGH); // turn on LED
  reset = true;
}
else // otherwise, button is not pressing
{
  digitalWrite(LED_BLUE, LOW); // turn off LED
  //reset = false;
}

Wire.beginTransmission(TCAADDR1);
Wire.write(0); // no channel selected
Wire.endTransmission();

```

```

for(int i=1; i<9; i++){
  sensorData[i]=sensorLuminance(tsl[i],i,TCAADDR);
}
J *req = notecard.newRequest("note.add");
if (req != NULL)
{
  JAddStringToObject(req, "file", "sensors.qo");
  JAddBoolToObject(req, "sync", true);
  J *body = JAddObjectToObject(req, "body");
  if (body)
  {
    for(int i=1; i<9;i++){
      String luxStr = "luminosity" + String(i);
      char lux[12];
      strcpy(lux, luxStr.c_str());
      JAddNumberToObject(body, lux, sensorData[i]);
    }
  }
  notecard.sendRequest(req);
}
delay(10000);

Wire.beginTransmission(TCAADDR);
Wire.write(0); // no channel selected
Wire.endTransmission();
sensorData[8] = sensorLuminance(tsl[8],0,TCAADDR1);
req = notecard.newRequest("note.add");
if (req != NULL)
{
  JAddStringToObject(req, "file", "lightsensor.qo");
  JAddBoolToObject(req, "sync", true);
  J *body = JAddObjectToObject(req, "body");
  if (body)
  {
    JAddNumberToObject(body, "luminosity8", sensorData[8]);
  }
  notecard.sendRequest(req);
}
Serial.println();
delay(40000);

Wire.beginTransmission(TCAADDR);
Wire.write(0); // no channel selected
Wire.endTransmission();
req = notecard.newRequest("note.add");
if (req != NULL)

```

```

{
  JAddStringToObject(req, "file", "voltage.qo");
  JAddBoolToObject(req, "sync", true);
  J *body = JAddObjectToObject(req, "body");
  if (body)
  {
    JAddNumberToObject(body, "voltage", currVoltage);
  }
  notecard.sendRequest(req);
}
Serial.println();
delay(30000);
}

//Measuring the battery life by measuring the voltage across the battery returns float value of
voltage
float batLife()
{
  int batVolt = analogRead(A13); // read pin 13 which measures voltage across the battery
  //Serial.printf("raw adc value: %d\n", batVolt); //prints out digital value of voltage

  //write function to convert from digital to analog
  float voltage = batVolt * (6.6/4098.0);
  Serial.printf("Voltage: %f\n", voltage); //prints out digital value of voltage
  return voltage;
}

//function to set colors
int greenLed()
{
  // color code #006600 (R = 0, G = 102, B = 0)
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 102);
  analogWrite(PIN_BLUE, 0);
  return 1;
}

int redLed()
{
  // color code #FF0000 (R = 255, G = 0, B = 0)
  analogWrite(PIN_RED, 255);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 0);
  return 2;
}

```

```

int offLed()
{
  analogWrite(PIN_RED, 0);
  analogWrite(PIN_GREEN, 0);
  analogWrite(PIN_BLUE, 0);
  return 3;
}

//blinking function
void blink(int state)
{
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= BLINK_INTERVAL)
  {
    if (state == 0) // led off and red
    {
      redLed();
      ledState = 1;
    }
    else if (state == 1)
    {
      offLed();
      ledState = 0;
    }
    else if (state == 2) // led off and green
    {
      greenLed();
      ledState = 3;
    }
    else if (state == 3)
    {
      offLed();
      ledState = 2;
    }
  }

  // offLed();
  // save the last time you blinked the LED
  previousMillis = currentMillis;
}
}

//create function to turn off device at night
void communication()
{
}
}

```

Code for Sparse Optical Flow Cloud Tracking

```
import time

import numpy as np
import cv2
import timeit

cap = cv2.VideoCapture('clouds.mp4')

# Params for corner detection
feature_params = dict(
    maxCorners = 100,
    qualityLevel = 0.3,
    minDistance = 7,
    blockSize = 7)

# Parameters for lucas kanade optical flow
lk_params = dict(
    winSize = (15, 15),
    maxLevel = 2,
    criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
               10, 0.03))

# Create some random colors
color = np.random.randint(0, 255, (100, 3))

# Take first frame and find corners in it
ret, old_frame = cap.read()
old_gray = cv2.cvtColor(old_frame,
                        cv2.COLOR_BGR2GRAY)
p0 = cv2.goodFeaturesToTrack(old_gray, mask = None,
                             **feature_params)

# Create a mask image for drawing purposes
mask = np.zeros_like(old_frame)
start = 0

while(1):
```



```

j = 0
final_total_slope = 0;
final_average_i = 0;
while (j < 1000):
    ret, frame = cap.read()
    frame_gray = cv2.cvtColor(frame,
                               cv2.COLOR_BGR2GRAY)

    # calculate optical flow
    p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray,
                                           frame_gray,
                                           p0,
                                           None,
                                           **lk_params)

    # Select good points
    good_new = p1[st == 1]
    good_old = p0[st == 1]
    if j == 0:
        slope_old = good_old
        total_x = 0
        total_y = 0
        final_i = 0
    # draw the tracks
    for i, (new, old) in enumerate(zip(good_new,
                                       good_old)):
        a, b = new.ravel()
        c, d = old.ravel()
        a = int(a)
        b = int(b)
        c = int(c)
        d = int(d)
        mask = cv2.line(mask, (a, b), (c, d), color[i].tolist(), 2)

        frame = cv2.circle(frame, (a, b), 5,
                           color[i].tolist(), -1)
    if j % 25 == 0:
        end = time.time()
        slope_new = good_new

```

```

for i, (new, old) in enumerate(zip(slope_new,
                                slope_old)):
    a, b = new.ravel()
    c, d = old.ravel()
    a += int(a)
    b += int(b)
    c += int(c)
    d += int(d)
    if c-a > 0 or c-a < 0:
        total_x += c-a
        total_y += d-b
    else:
        slope = 0
        final_i = i
    if total_x > 0 or total_x < 0:
        slope = total_y / total_x
    else:
        slope = 0
    slope_old = slope_new
    if final_average_i > 10:
        if slope < running_average + .25*running_average or slope >
running_average - .25*running_average:
            final_total_slope += slope
            final_average_i += 1
            running_average = final_total_slope / final_average_i
        else:
            final_total_slope += slope
            final_average_i += 1
            running_average = final_total_slope / final_average_i
    print("Average slope: " + str(running_average))
    #print("Pixels travelled = x:" + str(total_x/final_i) +
#       " y:" + str(total_y/final_i) + " in " + str(end-start) + " seconds")
    start = time.time()
    img = cv2.add(frame, mask)

    cv2.imshow('frame', img)
    j += 1
    k = cv2.waitKey(25)

```

```

        if k == 27:
            break

        # Updating Previous frame and points
        old_gray = frame_gray.copy()
        p0 = good_new.reshape(-1, 1, 2)
        ret, old_frame = cap.read()
        old_gray = cv2.cvtColor(old_frame,
                                cv2.COLOR_BGR2GRAY)
        p0 = cv2.goodFeaturesToTrack(old_gray, mask=None,
                                    **feature_params)
        print("Average for time frame: " + str(final_total_slope/final_average_i))
        final_average_i = 0
        final_total_slope = 0;

        # Create a mask image for drawing purposes
        mask = np.zeros_like(old_frame)
cv2.destroyAllWindows()
cap.release()

```

Startup Guide for 2023-2024

Created by 2023-2024 MQP team



This document will be the source of all nontechnical, non-research information for the MQP. Its purpose is to be a resource and guide for future MQP groups working on the Cloud Motion Vector Sensor System. To learn more about the system and the research conducted, please refer to previous MQP reports which detail the background, methods, and results of the project. See below the outline of topics that will be covered in this document. You can click on a specific topic which will direct you to that section.

Expectations of an MQP

MQP stands for Major Qualifying Project and is WPI's version of a senior capstone project. This is a requirement for all majors in order to obtain a bachelor's degree. You probably know all of this already so what really are the expectations of an MQP? It is a **student led** yearlong project. Unlike an IQP, you will not have strict deadlines and be spoon-fed what to do or how to do it. You must make your deadlines and determine what to do. You are the arbiter of your own fate. This can go great... or horribly wrong. The quality of your advisors, team members, and individual effort put in will generally determine the outcome.

The secret to a successful MQP is teamwork. Many MQP projects are interdisciplinary and thus require skills from multiple majors. For the CMVS project, the 2022-2023 group has mainly focused on redesigning the electrical and mechanical system almost from the ground up. This MQP may evolve to focus more on AI imaging and improving the algorithm for predicting cloud motion. There is also potential to further improve the hardware by implementing a camera.

You can read the previous group's detailed recommendations about the project in the 2022-2023 report. This is a good starting point for the project and developing your goals for the system.

MQP Deliverables

This is dependent on each department and advisor but in general:

1. Report
2. Poster board and presentation
3. Project deliverable

Please note that each department has its own set of requirements so be sure to research and figure out what you need to do. For ECE majors, the following link will direct you to the ECE MQP page in which it will detail what the ECE department is looking for in the report and presentation.

<https://www.wpi.edu/academics/departments/electrical-computer-engineering/major-project/guide>

Report

The final report, which usually includes at least a background, methods, and results section does not formally have to be submitted until the end of the last term of the MQP. It is quite a substantial document so doing one or two sections each term can help mitigate the all-nighters in C term.

Presentation

The presentation poster should be prepared two weeks before the scheduled project presentation day to allow time for modifications and printing. The ECE department will send out an email regarding information about the presentation and printing the poster. The speaking portion of the presentation should be prepared and practiced at least one week before project presentation day. Each individual should have roughly about 5 minutes to speak with a total presentation time of no more than 15 – 20 minutes.

Project Deliverables

The project deliverables are largely defined by you. They are the goals your team has made, usually in the beginning of A term, and direct what you do for the rest of the year. These deliverables don't have to be set in stone but developing some early on can help kick start the project on a good note.

For example, the deliverables that the 2022-2023 MQP group set were:

- I. Full Electrical functionality (End of A term goal)
 - a. All 9 Sensors work with ESP (we can collect data from the sensors)
 - b. Streamline connections between ESP and multiplexer and PCB design
 - c. Connectors between board and sensors that work in subzero temps
 - d. Fully functional circuit design with all electrical components integrated and working together
- II. Full Mechanical functionality (End of B term goal)
 - a. 3D printing Weatherproof housing for electrical hardware and sensors
 - b. Housing must be able to protect electrical hardware from rain, snow, extreme heat, sub-zero temps, etc. and still have everything work without issues
- III. Full Programming functionality (End of C term goal)
 - a. Be able to connect our device to the internet and transmit data
 - b. Upload light, temp, and irradiance data to Thinkspeak website
 - c. improve user interface to make it more accessible and easier to use
 - d. streamline data transfer process
- IV. Fully Functional prototype (BY MQP Presentation Day)
 - a. Have a fully function Cloud Motion Vector Sensor System that can achieve the things described in our project description

Suggested Startup Timeline

Below is a suggested outline of what to do in the first 4 weeks of A-term. Having a strong foundation of the project and research can help increase productivity and improve overall design and decision making.

- 1. Get Familiar with CMVS project1, 2 Week
 - a. Set up structure of meetings, mqp advisor meetings, team dynamic, etc.
 - i. Professor Noetscher prefers to meet weekly with some sort of presentation prepared describing what has been done
 - b. Preliminary research and reading previous MQP reports
 - c. Assessing current state of project and learning how it operates
 - d. Create project deliverables
- 2. Learn needed skills/certifications and assign tasks.....2,3,4 Week

- a. Get certification for makerspace as early as possible to use 3d printers, laser cutters, etc. See Helpful resources for more information
- b. Assign and relegate what each team member will generally work on throughout the whole project.
- c. Learn skills that will help in achieving your deliverables (software, soldering, etc.)

CMVS Operation and Quirks

Basic Operation

There are two external buttons and two LEDs on the main housing. The leftmost button is a toggle button which either turns on or off the whole system. This also resets the system. The push button to the right of the power button is the reset button. The RGB LED indicates the status of the battery, green meaning the battery is good and above 3.2V and red meaning the battery is 3.2V or below and needs to be charged or replaced. The blue LED indicates the system is in reset mode. There is a green LED on the cellular card which indicates the connection. There are also green LED's on each of the light sensors indicating power is being delivered.

Electrical Quirks

- There is an external power button and reset button that we designed to have easier access to operate the system when it is fully enclosed. The reset button essentially does the same thing as the power button as it just instantaneously shuts the system down and turns it back on. The power button keeps the system permanently off or on depending on the position of the switch. Do not press either of these buttons **when** the system is first turning on or in the initialization phase (Blue LED is lit up). Only when the blue LED turns off and you see a green LED blinking on the notecard can you then turn the system off or reset it again. This approximately takes 30-60 seconds.

If you do press the reset button or turn off the system while it is still initializing, when it turns back on it will be infinitely stuck in the initialization phase. You will see a red LED intermittently blink on the notecard. To fix this you must turn the power button off, then disconnect and reconnect the power source being used. Follow the procedure described above to avoid this happening again.

This quirk is determined to be part of the notecard design as when pressing the reset button designed on the notecard, the phenomenon described above occurs.

- The communication pins for the 6th sensor connector are switched compared to the other connectors in the current prototype. Thus, there is a special cable for this sensor that must stay together. This special cable is differentiated with a hairband near the connector end. This is fixed in the CMVS V4 kicad file.
- For the multiplexers, you can change the address of each of them using the three address pins A0, A1, and A2. Biasing to ground sets them to 0 while biasing to VCC sets them to 1. A2 is the most significant bit. In the current prototype, one multiplexer is set to address x70 and the other is x74. We originally wanted to set the multiplexer to x71 but biased the most significant bit instead of the least significant bit. We have fixed this in the CMVS V4 kicad file.

Mechanical Quirks

- The push button for the reset does not debounce. There is a latch that keeps the button pushed down if pressed far enough. This is not ideal, so only partially press the reset button until you see the blue LED light up indicating the system has gone in the reset mode. Otherwise, quickly press the button again to release the latch.

- The screws connecting the lid to the base of the main and sensor housing boxes should only be used as pins to keep the two parts together. The head of the screw should not be flush with the plastic. The holes are not threaded for screws, so inserting the screws too tightly will apply too much pressure to the tabs on the lid and could cause them to break; it will also eventually wear down the plastic and make it too loose.
- The wires have a lot of connections and many of them are inconsistent/weak; it would be best to redo the wires to reduce the number of connections and ensure that the remaining connections are strong and consistent to avoid frustrating connection issues.
- Etc...

Software Quirks

- The graphs in the ThingSpeak channels often do not show accurate data, and you need to download the data from ThingSpeak to be able to check it.
- The data will often be uploaded to Notehub with a different timestamp than ThingSpeak. This can lead to inaccuracies in the analysis code, as the gap between timestamps may be different from when they were originally recorded.

Helpful Resources

Bullet points marked with a * have their own quick startup guide further detailed at the end of this document sectioned Additional Notes.

Administrative

- Contact Colleen Sweeney to borrow lab keys for AK316 (if using the same lab). An initial security deposit of \$20 in cash will be needed. This security deposit will be returned once the key is returned at the end of the school year.
- Register your MQP through projects at the beginning of A term. You will also use [eprojects](#) to submit your final deliverables, like IQP.
- East Hall garage has open pedestrian access to test the CMVS; contact WPI police for parking access. This is also where the solar panels are installed.

Get Certification and Access to tools

- Join [WPI Makerspace Canvas Page](#), complete Prototyping Lab basic user training online and schedule in-person full user training for hands-on access to all Prototyping Lab FDM printers.
- Get training at Washburn labs to access all their tools related to mechanical fabrication like CNCs, routers, etc. Learn more here: <https://wpi.mfelabs.org/home/facilities>

Hardware

- Previous groups have made weatherproof housing using 3D printers in WPI's Prototyping Lab. Consult with the Prototyping Lab specialists or the Canvas page to decide which printer and filament is best suited for the project. If your print needs longer than 12 hours to print, you will need approval from Mitra Anand, the Makerspace Advanced Technology & Prototyping Specialist. The Prototyping Lab is usually open throughout the term with the hours shown below.

Prototyping Lab Regular Hours of Operation: (From October 24, 2022)

Sunday	10:00 am - 11:00 pm
Monday	10:00 am - 11:00 pm
Tuesday	10:00 am - 11:00 pm
Wednesday	10:00 am - 11:00 pm
Thursday	10:00 am - 11:00 pm
Friday	10:00 am - 01:00 am
Saturday	10:00 am - 01:00 am

- Through a lot of experimentation, we found that the minimum infill should be set to 15% and the maximum layer height should be 0.1mm. Otherwise, the prints were not strong enough to withstand drilling, let alone field testing.

Software

- KiCad: this is a free PCB software that can generate schematics and layouts. For those with no prior experience designing PCB's, this software has a large learning curve.
- Solidworks: this is a powerful CAD (computer aided design) software for designing 3D objects. For those with no prior experience using CAD, this software has a large learning curve. It is available to install through WPI: <https://hub.wpi.edu/software/504/solidworks>
- *Arduino IDE: This IDE is what is used to program the esp32

Websites

- *[JLC PCB](https://www.jlcpcb.com/): cheap, fast PCB fabrication (and assembly if parts are in stock)
- *ThingSpeak: <https://thingspeak.com>
- *Notehub.io: <https://notehub.io/>
- Route data between Notehub and ThingSpeak: <https://dev.blues.io/guides-and-tutorials/routing-data-to-cloud/thingspeak/>
- Overleaf:
- Digikey: Great website to look for electrical parts
- Github: https://github.com/yayoi730/CMVS_2022/

Financial

- Apply for [Makerbucks](#), a program offered by the Innovation Studio Makerspace that awards teams of students up to \$200 in funding to be used to build hardware and software prototypes.
- The ECE department allocates \$250 for each team member that is an ece student. So for 4 ece students in an MQP they would be allocated \$1000. Here is the process for getting these funds:
 1. You must contact Deb, the administrative assistant, to fill out the application to get these funds.
 2. Once the application is filled out and approved (usually within a day or two) you are assigned a team number.
 3. For anything that you want to order for the project you can simply email Bill Appleyard your BOM and your team number. He is in usually from 8am – 4pm.
 4. It can take up to a week to get in the ordered parts. Bill will send you an email notifying you when they arrive. You can pick them up in Bill's office in room ### from 8am- 4pm.
 5. For any other expenses that can't go through Bill (like ordering PCBs, etc.) you can fill out an expense report. You can contact Deb if assistance is needed.

Important Contacts

Name	Description	Location	Contact
William Appleyard	Electronics Technician: Can help with ordering parts/borrowing equipment	AK 111 (ECE shop)	wlappleyard@wpi.edu
Colleen Sweeney	ECE Administrative Assistant: Can provide lab keys and useful contacts in the department	AK 202 (main office)	sweeney@wpi.edu

Will Buchta	ECE Undergraduate Tutor: Can help with general review of PCB design (group should extensively review the PCB as well)		jwbuchta@wpi.edu
Mitra Anand	Makerspace Advanced Technology & Prototyping Specialist: Facilitates Makerbucks and can help with questions about Prototyping Lab	Innovation Studio Prototyping Lab	mvanand@wpi.edu
Deb Thompson	ECE Administrative Assistant: Can help with filling out expense reports	AK 202 (main office)	dlthompson@wpi.edu
Olivia Rockrohr	WPI Graduate, former MQP team member: point of contact for any questions regarding MQP		orockrohr@wpi.edu

Additional Notes

- After contacting NERC, NextEra, and DOE, we concluded that cellular would be the most practical communications protocol for this project. However, if needed, East Hall garage has 2 cat 6 ethernet ports to provide internet access.

JLC PCB

- JLC has most parts that you will most likely use when designing a PCB
- Make sure to check JLC documentation regarding specific formatting for the pick n place file and the BOM file. This is very important if you would like the PCB to be completely assembled with all the components.

Arduino IDE

- You need the Adafruit_VEML7700, Wire, stdio, and Notecard libraries in Arduino for the code to run. Other libraries these four libraries depend on are also necessary but should automatically download along with them.
- Baud rate needs to be set to 115200

Notehub.io

- The cellular card sends data to Notehub, which is then routed to ThingSpeak.
- Notehub has many tutorials and good documentation on how to use the cellular card.
- If necessary, you can set up another route between ThingSpeak and Notehub using the instructions provided in the link provided under Websites in Helpful Resources.

ThingSpeak

- All the data is analyzed using ThingSpeak using code that can be found or added under "Matlab Visualizations" in the apps tab