

High School Kit and Assembly Instruction Manual for K-12 STEM Curricula

Prof Caleb Farny

GLOBE Mission Earth

Dept of Mechanical Engineering

Boston University



The material in this document is based upon work supported by NASA under grant award No. NNX16AC54A. Any opinions, findings, and conclusions or recommendations expressed in this material are those of author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.



Most recent update: 10/19

TABLE OF CONTENTS

1	Introduction	2
1.1	Weather, GLOBE, and Citizen Science (Why a Weather Station?).....	2
1.2	How to Use this Manual	2
1.3	Ratings and Info Boxes Explained	3
1.4	Materials (BOMs)	4
1.5	System Overview	7
2	Hardware Overview	8
2.1	Solar Panel	8
2.2	USB DC Solar Lithium Ion Charger	8
2.3	Rechargeable Lithium Ion Battery	9
2.4	BME280 Sensor.....	10
2.5	VERTER 5V Buck/Boost Board.....	10
2.6	Huzzah Breakout Board	10
2.7	Temperature Sensor (DS18B20)	11
2.8	Anemometer	11
3	Electronics Preparation.....	12
3.1	Huzzah Preparation	13
3.2	BME280 Board Preparation	13
3.3	Power Management Board Preparation.....	14
3.4	Buck/Boost Preparation	14
3.5	LED Button.....	14
4	Board Testing and Programming	15
4.1	Blink and Wifi Tests	17
4.2	ThingSpeak Account	18
4.3	Weather Station Code	19
4.4	Perma-proto Board Layout	19
4.5	Electronics Assembly and Testing.....	21
4.6	Anemometer Calibration	26
5	Hardware Mounting and Assembly	29
5.1	Radiation Shield Assembly.....	29
5.2	Anemometer Mount.....	33
5.3	Solar Panel Mount	34
5.4	Electronics Enclosure.....	35
5.5	Mounting the Electronics Enclosure.....	38
5.6	Weather Station Deployment.....	38
6	DIY Anemometer	39
6.1	Tools	39
6.2	Overall Design.....	40
6.3	Encoder.....	42
6.4	Structure.....	44
6.5	Hardware and Software.....	48
7	Appendices	53
A	Wiring Diagrams	53
B	Weather Station Arduino Code	55
C	Video web links.....	65
C	STEM Exercises	66
D	Manufacturer Guides	69

INTRODUCTION

1.1 WEATHER, GLOBE, AND CITIZEN SCIENCE (WHY A WEATHER STATION?)

The Earth's atmosphere is changing rapidly. To accurately track weather patterns over time and to meaningfully understand the environment, U.S. National Aeronautics and Space Administration (NASA) and the National Science Foundation (NSF) sponsors the Global Learning and Observations to Benefit the Environment (GLOBE) program to collect weather data worldwide. Not only do scientists contribute data to NASA GLOBE's program, but average citizens and students can participate as well to encourage "citizen science", educating the public while accomplishing GLOBE's main mission.

This weather station and instructional document has been designed with the following goals:

- Compatible with the instrument and measurement standards set by GLOBE
- An educational experience for the K-12 STEM audience
 - Assembly as a kit of easily-sourced parts by high school students, with a minimum of machining requirements and technical ability
 - Introduction to electronics soldering, circuit board layout and design, and Arduino coding
- Relatively low-cost, compared to the commercial market alternatives
- Measurement of air temperature ($\pm 0.5^{\circ}\text{C}$), humidity ($\pm 3\%$ accuracy), barometric pressure (± 1 hPa absolute accuracy), and wind speed (± 1 m/s worst case accuracy).

For relevant earth science background, as well as some instrumentation details, it is recommended you check out these links provided by GLOBE:

For the general GLOBE site: <https://www.globe.gov/>

For Temperature:

<https://www.globe.gov/documents/348614/93d4bb3c-79e3-4255-9fc8-537fc4f870dc>

For Pressure:

<https://www.globe.gov/documents/348614/a1ab9ecb-f8b1-41ba-9dd9-a420895f954b>

For Humidity:

<https://www.globe.gov/documents/348614/89f8c44d-4a99-494b-ba81-1853b80710b4>

1.2 HOW TO USE THIS MANUAL

This manual is split into six sections. Sections 2 to 5 describe the different components of the weather station, while the sixth is an optional educational exercise to explore how a measurement sensor operates. The manual describes all tools, materials, costs, and procedures to build your GLOBE weather station, as well as instructions for Wi-Fi data logging. Ratings, info boxes, and debugging tips are included as needed for clarity. The circuit board and hardware assembly steps are also illustrated through a series of tutorial videos, the links for which appear alongside the relevant sections in the manual. The Appendices have important reference information and STEM educational exercises, so be sure to read through these carefully as well. While this manual is a beta version, we sincerely hope it is clear and easy to follow.

Questions regarding any aspects of this document can be directed to Prof Caleb Farny, Dept of Mechanical Engineering, Boston University (farny@bu.edu, 617-353-8664). Special thanks for assistance with this project goes to Boston University Mechanical Engineering Senior Capstone project students (now alumni) Chitanya Gopu, David Martinez, Hayley Walker, Michelle Ye, Abdullah Babgi, Hester van der Laan, Mike Ward, Walid El Kara, and David Oluwadara.

1.3 RATINGS AND INFO BOXES EXPLAINED

Due to the DIY (Do It Yourself) nature of the project, many construction and hardware procedures have multiple options, or ways of achieving the same result. Additionally, any materials or components in the Bill of Materials (BOM) can be replaced if your school already has materials that would work. Because of this, when options are given in the manual, a rating system will be used to rank the engineering complexity, and material/tool availability of each option. The ratings are as follows:

Complexity



Very easy to understand and build. This is intuitive or just very simple and will not take very much time.



In the middle. Very understandable if you read the instructions carefully. May take a bit longer or be more involved to build.



A fun challenge! Learn some real engineering skills. Looking at additional resources given here will aid understanding, and instructor's help is recommended.

s



Not material intensive. Tools needed include office supplies and/or your hands.



May involve tools you may not have used before, but are fairly common, easily learned, and reasonably safe with the right technique: hammers, screwdrivers, allen wrenches, hand drills.



Check your school's resources before tackling this option. May require supervision and help. These tools require safety procedures, or are not as easily obtained. These include tap drills, soldering irons, or machine shop clamps. Easier alternatives will often be given.

In addition to these ratings, tips and more information are scattered throughout this manual.

1.4 BILL OF MATERIALS (BOMs)

Bill of Materials: Required items for Globe Mission Earth Weather Station					
Item	Vendor	Part #	Cost (\$)	Quantity	Total (\$)
ESP8266 Feather Huzzah WiFi microcontroller	Adafruit	3046	18.95	1	18.95
VERTER 5V USB Buck-Boost - 500mA from 3V-5V / 1000ma from 5V-12V	Adafruit	2190	9.95	1	9.95
BME280 I2C or SPI Temp/Humidity/Pressure sensor	Adafruit	2652	19.95	1	19.95
USB DC solar lithium ion charger	Adafruit	390	17.5	1	17.5
Huge 6 V 6W solar panel	Adafruit	1525	59	1	59
Lithium Ion Battery Pack - 3.7V 4400mAh	Adafruit	354	19.95	1	19.95
Anemometer	Adafruit	1733	44.95	1	44.95
16 mm Illuminated Pushbutton - Green Latching	Adafruit	1433	1.5	1	1.5
Perma-proto half-sized breadboard PCB - single	Adafruit	1609	4.5	1	4.5
2.1 mm DC jack adapter cable	Adafruit	2788	0.95	1	0.95
2.1mm female/male barrel jack extension cable	Adafruit	327	2.95	1	2.95
JST 2 pin cable	Adafruit	261	0.75	1	0.75
Terminal block - pack of 5	Adafruit	724	2.95	1	2.95
10k precision epoxy thermistor	Adafruit	372	4	1	4
Universal Solar Panel Bracket	Voltaic Systems		9	1	9
Davis Instruments Solar Radiation Shielding 7714	Amazon		72.5	1	72.5
1-gang horizontal weather proof extra duty while in use cover	Home Depot	MM420C	7.97	1	7.97
Bell 1-gang 3-outlets 3/4 in threaded weatherproof box	Home Depot	5324-0B	5.44	1	5.44
Simpson strong-tie ZMAX 18-gauge galvanized steel angle	Home Depot	A23Z	1.37	1	1.37
Simpson strong-tie ZMAX 18-gauge galvanized steel angle	Home Depot	A21Z	0.58	1	0.58
#8-32 x 12 in zinc-plated threaded rod	Home Depot	802087	0.98	2	1.96
Required part subtotal (\$)					305.69

Table 1: Required parts for the weather station.

The list above assumes that you already have the following tools:

- Soldering iron and solder
- Phillips Screwdriver, PH2 and PH00 (jewelry)
- Hex Wrench (1/8" wrench for 1/4" nuts; 5/64" wrench for #8 nuts)
- Wire cutter
- Wire stripper
- Electronics (or needle nose) pliers
- A handheld multimeter is helpful for diagnostic purposes (but not required)

Bill of Materials (Support items): Globe Mission Earth Weather Station					
Item	Vendor	Part #	Cost (\$)	Quantity	Total (\$)
Crown bolt 1/4 in-20 x 3/4 in Phillips-slotted truss-head machine screw	Home Depot	5251	1.18	3	3.54
Everbilt #8-32x1/2 in stainless Phillips-slotted round-head machine screw	Home Depot	814201	1.18	1	1.18
1/4-20 in stainless steel hex nut (pack of 4; need 5 total)	Home Depot	800051	1.18	2	2.36
Everbilt #8-32 stainless steel machine nut (pack of 4; need 8 total)	Home Depot	32001	1.18	2	2.36
Everbilt 1/4 in x 5/8 in galvanized steel flat cut washer	Home Depot	804096	0.12	10	1.2
Everbilt #8 stainless steel flat washer (12-pack)	Home Depot	800321	1.18	1	1.18
Gorilla Glue 2.8 oz sealant (or similar silicon-based waterproof sealant)	Home Depot	80910	4.97	1	4.97
Generic M2.5 Nylon Hex M-F Spacer/Screw/Nut Assorted Kit	Amazon		9.99	1	9.99
On stage EB9760 Exterior speaker mounting bracket	Amazon		8.95	1	8.95
Pyle Universal speaker stand mount holder - heavy duty tripod PSTND2	Amazon		27.01	1	27.01
Preformed 140 Jumper wire kit (NightFire Electronics)	Amazon		6.45	1	6.45
Electrical wire 22 gauge silicone hook up wire [10' black, 10' red], Haerkn	Amazon		6.38	1	6.38
USB cable - 6" A/miniB *	Adafruit	899	2.95	1	2.95
220 Ohm resistor pack (pack of 25; only need 1 resistor)	Adafruit	2780	0.75	1	0.75
2.2 kOhm resistor pack (pack of 25; only need 1)	Adafruit	2782	0.75	1	0.75
10 kOhm resistor pack (pack of 25; only need 2)	Adafruit	2783	0.75	1	0.75
4.7 kOhm resistor pack (pack of 25; only need 1)	Adafruit	2784	0.75	1	0.75
Additional part needs (\$)					81.52

Table 2: Additional parts for the weather station. Your school may have some of these already.

The total cost for both BOMs is \$387.21, not including price fluctuations, taxes, and shipping.

BOM 2: For Section 6 (DIY anemometer) use only

BOM #	Item	Listed Name (ID)	Vendor	Quantity	Cost (\$)	Purchase Link
1	PVC pipe and fittings	1.5" PVC pipe (1), 1 ft long, tee (2), 1 ½" to ¾" bushings (3), cap (1) (Home Depot)	Home Depot	-	13.18	Buy from your local hardware store. We went to Home Depot
2	Small PVC Pipe	Thick Walled Unthreaded PVC Pipe	Home Depot	1	4.68	https://www.mcmaster.com/#48855k22/=1762zcn
3	PVC Cement and Primer	PVC Cement and Primer	Amazon	1	9.10	http://preview.tinyurl.com/lgoxy8p
4	Aluminum Shaft	Anodized Aluminum rotary shaft	McMaster-Carr	1	11.83	https://www.mcmaster.com/#5911k11/=17eh288
5	Photo interrupter	1/8" Gap Photo Interrupter Slotted Optical Switch	Amazon	1	3.83	https://tinyurl.com/lzyurbj
6	Shaft Clamp	Clamping Shaft Collar	McMaster-Carr	2	4.40	https://www.mcmaster.com/#6157k12/=17h2u9p
7	Coffee Scoops!	Chef Craft 4 Piece Perfect Coffee Measuring Spoon Scoop, Black	Amazon	1 package	6.80	https://tinyurl.com/l6r2dga
8	Ball Bearing	R4A-2RS Bearing 1/4"x3/4"x9/32" inch Sealed Miniature	Amazon	1	5.37	https://tinyurl.com/lfabtbu
9	Conduit Clamp	2-in. PVC conduit clamp OR 2-in. Rigid 2-hole strap	McMaster-Carr	2	2.50	https://tinyurl.com/mob7hom
10	Super Glue	Super Glue	Amazon	1 package	1.88	https://tinyurl.com/mk7v95y
11	Screw Item 29	6-32 x 3/8" Phillips or Allen cap screw and nuts	Local Hardware Store	3	-	-
12	Screw Item 30	3-48 x 3/8" set screw or Phillips screw	Local Hardware Store	1	-	-
13	O-rings	¼" O-rings	Local Hardware Store	2	-	-
14	Handheld Anemometer	GM816 Digital Anemometer	Walmart	1	15.97	https://tinyurl.com/lokntax

Total Cost: \$44.68 (not including price fluctuations, taxes, and shipping costs).

Item 14 is optional to test for your handmade anemometer's accuracy.

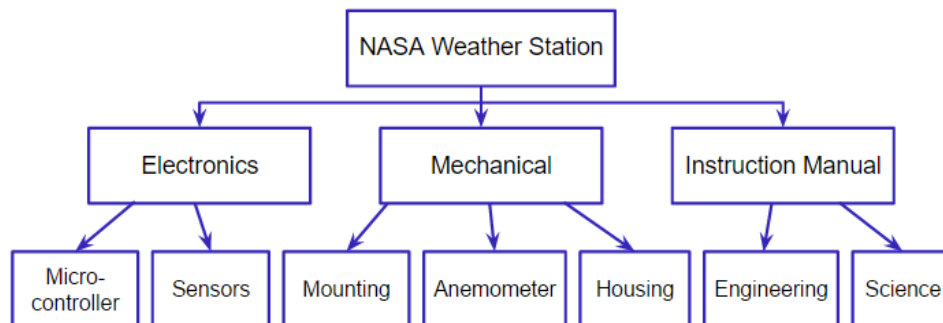
1.5 System Overview

The weather station (shown in Fig. 1) you will build is relatively cheap, solar powered, and wirelessly acquires data in real-time on an online platform called ThingSpeak, using an Arduino code. The system measures air temperature, humidity, pressure, and wind speed.



Figure 1: Completed weather station.

This project has three components: electrical, mechanical, and data transmission.



1. The electrical system includes the microcontroller, power management, and some of the weather sensors.
2. The mechanical components of the weather station kit include the electronics enclosure and mounting. The enclosure shields the sensors and electronics from rain and mechanical damage, and a radiation shield is used to house the temperature sensor. The mounting holds the electrical components to the tripod, and the tripod to the roof.
3. The Arduino code runs on a wifi-enabled board, which collects and transmits the data from the sensors. These data are then sent to a host website.

If this sounds complicated, don't worry! This manual is detailed and informative. You'll learn skills like how microcontrollers work, Arduino coding, basic breadboard wiring and soldering, how sensors work, how to take accurate measurements, how to build structures from real engineering drawings, and how to use tools safely. Have fun!



The weather station hardware incorporates 4 electronics boards, 2 external sensors, a solar panel, a rechargeable battery, and various structural components. Each board has a dedicated purpose, as described below. Note that integrating these components requires programming using the Arduino platform and some soldering to make the electrical connections. If you have little or no soldering experience we suggest that you watch some soldering tutorials on YouTube. We also recommend using a 60/40 (tin/lead ratio) solder wire. This section gives an overview for each major hardware component, and all soldering steps are explicitly described in Sec. 3.

2.1 Solar Panel

The solar panel provides the power for the entire system. The sun's photons provide energy that gets converted to electrical energy by the solar panel (see Appendix C). The larger the panel, the more energy is captured at any point in time. The amount of energy that gets produced constantly changes! As the sun rises and sets, the number of photons that hit the panel change due to the varying atmospheric conditions and incident angle of the sun on the panel surface; the same is true as the seasons change. The panel's power conversion efficiency is also variable, since the electrical components that convert the energy are affected by temperature. The panel is more efficient at colder temperatures, and less so at hot temperatures. The electrical power provided to our houses from the electrical grid has an alternating current (60 Hz AC), for logistical reasons, but since the sun is considered a constant source (over short periods of time!), the power produced from this panel is direct current (DC), or a constant current over a short period of time.

The panel we suggest you use is made by Voltaic Systems. When the sun shines at its peak intensity the panel produces 6W, which translates into approximately 6V. When the sun shines at lower intensities, the power produced, and voltage generated, is lower. The panel is mounted via two 4-40 screws to a bracket also manufactured by Voltaic. This bracket has a 30-deg bend to optimize the incident angle of the sun on the panel and is positioned on the weather station so as to prevent any other structures from making a shadow on the panel. The power is transmitted by a cable to which you'll need to attach a 2.1mm DC jack adapter AND 2.1mm extension cable in order to connect it to the USB DC solar power management board.

2.2 USB DC Solar Lithium Ion Charger ("Power supply board")

Accompanying tutorial video: <https://youtu.be/lfMBLsC3xTA>

This board interfaces with the solar panel, lithium ion rechargeable battery, and a buck/boost power conditioning board. It's manufactured by Adafruit and is specifically designed to work with these 3 components. Its purpose is to manage the power needed for the weather station. The

station is powered by the sun, which is an intermittent source. Each electrical component of the station needs a continuous supply of electrical energy, or electrical power (a certain amount of energy over a specified period of time), so when the sun provides this baseline power need, this board sends all of the solar panel power to these components. When the solar panel generates more than this need, the board sends the excess power to the rechargeable battery. Note that Adafruit has a document that describes the following information in more detail (<https://learn.adafruit.com/usb-dc-and-solar-lipoly-charger>); please read both descriptions.

Three components will need to be soldered to this board: a resistor, thermistor, and capacitor. Figures 4 and 5 show these components.

These types of rechargeable batteries have a temperature range over which they can be safely charged. However, when the temperature is below 0°C or above 50°C, the battery will be damaged if it's charged. This board is designed to incorporate an optional **thermistor** to monitor the temperature and stop the board from charging the battery in these extreme temperatures. A thermistor is temperature sensor that changes its resistance when the temperature changes. Adafruit has designed the board's circuitry to allow a 10 kΩ thermistor to be used; if the resistance from the thermistor gets too high (cold temperatures) or too low (high temperatures) the board will stop charging the battery. This feature is an add-on feature of the board, so you'll need to solder the thermistor wires to the board and remove the fixed resistor that ships with the board. We suggest positioning the thermistor below the board, so that the leads are fed up through the THERM terminal holes.

The board also has a baseline rate at which it will charge the battery. Since we're using a relatively big solar panel to provide the power needs of the station, we'll need to increase this charge rate (amount of electrical energy sent to the battery over time). You'll do this by soldering an additional (2.2 kΩ) resistor to the board, to the PROG terminals. Make sure the resistor lies flat to the board, and that the leads are trimmed after soldering.

The capacitor ships with the board and should be soldered as directed in Adafruit's document and video. This capacitor has a polarity (positive vs ground) for its wires, and each wire should be matched up with the correct hole on the board. The longer leg will be the positive wire, and the negative leg is marked with a negative symbol on the body of the capacitor. For our configuration, make sure the capacitor is soldered such that it's positioned at a 45° angle to the board, *towards* the center of the board. This means you should solder it after the thermistor and resistors have been soldered, and you should bend its wires to put it into this configuration before soldering and trimming its leads. The board is eventually supported via three nylon standoff screws.

2.3 Rechargeable Lithium Ion Battery

This battery is sold by Adafruit and is rated at 4400mAh. The only reason why we need a battery is because the sun eventually stops providing enough power for the station to keep functioning. It acts as a reserve power source that gets constantly charged during sunny conditions, and slowly discharged otherwise. The battery connects to the power supply board via a JST cable and is

housed in the metal weatherproof box. Lithium ion batteries are well-known fire hazards, so be careful to handle the battery properly and avoid mechanical damage to the battery.

2.4 BME280 Sensor

Accompanying tutorial video: <https://youtu.be/NbnqFafLHxU>

The BME280 board (Bosch manufactures the chip; Adafruit integrates it into a user-friendly board) features an on-board pressure, humidity, and temperature sensor that will be housed inside the electronics housing unit. This unit will have some air flow to the sensors that will allow the pressure and humidity data to be accurate, but the temperature won't be accurate due to the solar radiation effect and will be ignored. We'll use a different temperature sensor to monitor the air temperature. This board requires 4 electrical connections, so the first step is to solder 4 header pins, one to each of the following through-holes on the board: Vin, Gnd, SCK, and SDI. The first two connections will allow the board to be powered (from the Huzzah microcontroller board), and the last two connections will allow the data from the board to be properly sent to the microcontroller. The board has 2 protocol options for transferring the data (SPI and I2C); we'll use the I2C option. The board serves as the mechanical support for the buck/boost board.

To properly interpret the signals from this board we'll need to install a BME280 library when we program it in the Arduino IDE, as described below.

2.5 VERTER 5V Buck/Boost Board

The weather station has some tricky power requirements. The voltage requirements of the BME280, DS18B20, and Huzzah microcontroller are all between 3 – 5 V, while the anemometer needs 5-12 V. When the sun is at its peak, the solar panel produces 6.5 V, and when the sun is at its minimum, the battery provides 3.7 V – but these other components all have a shared need of 5 V! This means that sometimes we'll need to reduce the 6.5 V and other times we'll need to increase the 3.7 V. The buck/boost board solves this problem for us. It has an input range of 3 – 12 V, and either boosts (increases) anything below 5 V *up* to 5 V or it bucks (decreases) anything above 5 V *down* to 5 V.

You'll need to solder two wire terminals. The 3-12 Vin terminal should connect to the USB solar power board via a JST cable, and the 5 V terminal should connect to a red and black wire (soldered to the 5.2 V (+) and ground (-) row on the perma-proto board, respectively). The board is physically mounted to the BME280 board via two standoff screws.

2.6 Feather Huzzah Board

This is the microcontroller board, which means that it is programmable and interfaces with the electrical signals that are sent from the sensors. This board is optimal since it

- (a) has enough input connections for all the signals we need to capture,
- (b) has low power requirements,
- (c) is easily programmed via the Arduino IDE,
- (d) can send data via wifi.

It can also be reprogrammed wirelessly, in case we wanted to reconfigure it, but we won't be using that capability in this project. The heart of this board is an ESP8266 chip, which can be tricky to work with on its own, but Adafruit has built a board to condition the signals to and from this chip and developed the necessary library files that make it easier to use.

The Huzzah board requires 3-5 V for its power needs, which comes from the buck/boost board. It communicates to the BME280 and DS18B20 sensors via its General Purpose Input Output (GPIO) pins (these sense digital signals) and to the anemometer via its Analog Digital Conversion (ADC) pin.

The board is programmed via a USB cable, which plugs into the mini USB port on the board side and the USB port on your computer. Adafruit provides helpful documentation which we suggest you reference (<https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/overview>) in addition to this document. The control program is stored in the board's flash memory, which has the advantage of retaining the code even when the system isn't receiving power. This is helpful for us in case the solar panel isn't receiving any sun and the battery has been drained!

2.7 Temperature Sensor (DS18B20)

The temperature sensor used in the weather station directly digitizes the temperature data. This sensor requires 3 – 5.5 V input voltage (across its brown and black wires) and its signal wire is the white wire. The goal is to measure the air temperature, and one effect we need to avoid is direct absorption of the sun's radiation via the material used to make the sensor itself. We can avoid this by housing the sensor in a solar radiation shield, a structure that acts like a solar umbrella. The radiation shield that we'll use is manufactured by Davis.

2.8 Anemometer

An anemometer refers to a general class of devices that measures the speed of a fluid. The fluid that we're measuring here is air, and we'll measure the air speed with a cup anemometer, a common approach. Three cups are mounted to a vertical rod, and as the wind exerts a drag force on any of the cups, the cup assembly spins and an internal sensor counts the number of rotations. This means that the spin rate gets converted to a voltage that is transmitted through one of the wires that extends from the anemometer. The signal from the anemometer is different from the other sensors in this station, as it's an analog voltage that needs to be converted to a digital signal that we can transmit through the wifi signal and store in a computer. The analog signal gets digitized by the Huzzah board, so the blue signal wire from the anemometer must be directed to the ADC (Analog-Digital Conversion) pin on the Huzzah board. The anemometer can detect wind speeds between approximately 0.5 – 72 mph (0.2 – 32.4 m/s) and has an accuracy uncertainty of ± 2.2 mph (± 1 m/s).

The anemometer must be positioned high enough off the ground in order to get an accurate reading of the wind speed. We follow the GLOBE guidelines of positioning the anemometer at least six feet off the ground and away from any walls, since the wind speed decreases considerably near the ground. The station is also designed to prevent other structures on the station from blocking the wind flow.

3

ELECTRONICS PREPARATION



Now that we have an understanding of all the major components, we need to provide the correct electrical pathways for the power and sensor signals to flow between each of these components. We'll achieve these connections by making approximately 77 solder joints.



What is soldering? The primary goal of soldering is to provide an electrical connection between two wires and the secondary goal is to provide mechanical support so that the wires stay in place. Soldering is when any of various alloys (typically tin and lead) are fused and applied to the joint between metal objects to unite them without heating the objects to the melting point.

If you're new to soldering, check out this link: <http://www.wikihow.com/Solder-Electronics> Remember that the solder tip can be damaged, which will make future solder joints very difficult or impossible to make. Avoid damage by keeping the solder iron temperature to a minimal temperature, cleaning the tip with a wet sponge only when necessary, and minimizing direct mechanical force applied to the tip. Lead-free solder is available, but it's very difficult to work with and can often lead to failed solder joints. Now that you've had an introduction and you've read all the information on soldering → Proceed!

This link explains basic circuit theory: <https://learn.sparkfun.com/tutorials/what-is-a-circuit>

This manual is organized such that the electrical connections for each board or component is described separately (this section), and then tested separately before assembling the system in increasingly-bigger subsystems (Section 4). The link to each respective tutorial video appear where relevant, although not *every* section has its own accompanying video. The video tutorials are intended to demonstrate the physical connections and positions of each component, so they present a more linear approach. If you want to watch the system get assembled from beginning to end, the videos are bundled into modular steps and are detailed in Appendix C.

3.1 Feather Huzzah Preparation

The board can be ordered in multiple configurations, either with or without the header pins already soldered for you. If your board shipped with the header pins already soldered, you can skip to step 4 below.

If you need to solder the pins, refer to Fig. 3, as only 8 of the pin connections need to be soldered. These connections correspond to the red connections for the Feather Huzzah board. Note that you are *not* soldering the pins to the perma-proto board just yet – only the Huzzah board.

Soldering the header pins:

1. Position each header pin on each side of the board so that the short side sticks up through the pad holes and its standoff/longer pin side is accessible on the bottom side of the board. Note that it may be easiest to solder the remaining pins by temporarily placing the long end of the pins in the perma-proto board and the board resting on top.
2. Orient the Huzzah board so that the board is lying flat and its USB port is on the left side.
3. Apply solder to the short pin ends exposed through the upper side of the board. Starting with the top row side, solder the following connections:
 - VBatt
 - 12
 - 0
 - 5
 - 4

Solder the following connections on the bottom row side:

- 3V
 - GND
 - ADC
4. Mount a nylon standoff screw to the board. Select a 25 mm M2.5 standoff and a M2.5 nut. Position the nut below the through-hole on the RST side of the board and insert the screw down through the hole (hand tight is fine). One side of the power management board will be mounted above the Huzzah via this standoff screw. The correct standoff placement is shown in Fig. 5 and is identified with the blue mount hole symbol in Fig. 3 (position C17).

3.2 BME280 Board Preparation

Supporting tutorial video: <https://youtu.be/NbnqFafLHxU>

As with the Huzzah board, we need to prepare the BME280 board for its header pins but their solder connection to the perma-proto board will happen at a later step.

1. The board ships with header pins. Cut the strip so that you have at least six headers poking the shorter exposed side through the bottom of the board, aligned with the Vin terminal.
2. With the plastic header standoff positioned on the pin below the board, solder the exposed pin on the top of the board for the following terminals: Vin, Gnd, SCK, and SDI. The other pins don't need to be soldered.
3. Mount the nylon standoffs to the board. Select two 12 mm standoffs and secure each with a nylon nut in the two mounting holes at the corners of the board. The nuts should be flush against the bottom of the board. The buck/boost board will be mounted above the BME280

via these standoffs. The correct standoff placement is shown in Fig. 5 and is identified with the blue mount hole symbol in Fig. 3.

3.3 Power management board preparation (Solar LiPoly Charger)

Supporting tutorial video: <https://youtu.be/lfMBLsC3xTA>

1. Position the 2.2 k Ω resistor so that it's flat on the board, with its wires (leads) passing through the PROG terminals. Solder each wire and trim the excess length below the board.
2. The thermistor has two exposed electrical leads; they'll need to be physically separated in order to fit properly. Use a sharp edge (the edge of a scissor blade or wire cutter) to cut the plastic housing that connects the wires; an inch or so is fine.
3. Position the thermistor below the board, with its leads passing up through the THERM terminals (there's no polarity for this component). Solder each wire and trim accordingly.
4. Using a wire cutter, cut the small black surface mount resistor that is positioned between the THERM terminals (the thermistor effectively replaces this fixed resistor).
5. Position the capacitor so that its leads are correctly aligned with the polarity symbols on the board. The capacitor leads extending to the board should be long enough to allow you to angle the capacitor towards the center of the board (approximately 45 degrees to the board). The goal is to keep a low profile for the board (not too high) but not have the capacitor touch the black chip positioned in the center of the board (it gets hot).
6. Solder the capacitor leads and trim accordingly.

3.4 Buck/Boost Preparation

Supporting tutorial video (starting at 9:20): <https://youtu.be/lfMBLsC3xTA>

1. The board ships with a two-wire terminal and a USB terminal. You won't need the USB terminal. Position the two-wire terminal in the 3-12 Vin pad connections and solder the junctions on the bottom of the board.
2. Take another two-wire terminal and position it on the two connections on the opposite side of the board, between the + and - symbols. Solder the junctions on the underside of the board.

3.5 LED Button

Supporting tutorial video: <https://youtu.be/2d3HB5x8ny8>

The LED button serves as a way to save the battery power by turning off the power to the downstream components of the buck/boost board and to also provide a visual indicator of whether the system has sufficient power via the battery and/or the solar panel. The button requires 2.2 V to power its light. It has four terminals: a positive and negative terminal pair for the 2.2 V power, and a second pair for the control signal.

Fig. 2 shows the connection arrangement. The button integration is such that the positive wire from the buck/boost is only connected to the Huzzah positive voltage terminal when the button is pressed in and in the 'on' state. When this happens, power is directed back to the button via a voltage divider (the 10 and 2.2 k Ω resistor circuit), energizing the LED light. So, if the button is in the on state and the system has enough power to generate 3 V from the Huzzah board, the LED light will shine. If the button is in the on state but the buck/boost board can't generate a 5 V

output (due a drained battery and/or low sunlight conditions), the LED light won't have sufficient power to shine.

LED switch logic

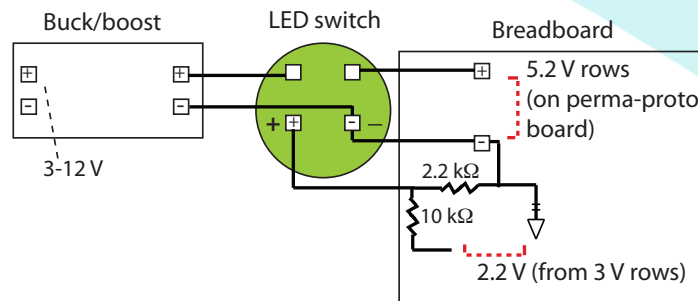


Figure 2: LED button wiring diagram.

To prepare the LED button, you'll need the button, three red wires, and two black wires. Note that these wires should be approximately 6 inches long. Referencing the LED button terminals in the figure:

1. Solder both black wires to the negative terminal.
2. Solder a red wire to the control terminal above to the positive terminal (top left, in the orientation above). This wire will eventually clamp into the positive output terminal of the buck/boost board.
3. Solder a red wire to the other control terminal. This wire will eventually clamp into the terminal port in the top positive voltage row (+5.2 V) on the perma-proto board.
4. Solder a red wire to the positive terminal. This wire will eventually clamp into the terminal port in the A26 position on the perma-proto board.

4

BOARD TESTING AND PROGRAMMING



These boards will eventually be wired together and physically constrained through a combination of solder joints, nylon standoff posts, and terminal connections. Most of these connections will be made on the perma-proto board, which is a more permanent and reliable solution than a standard breadboard. Before the Huzzah and BME280 boards are soldered to the perma-proto board, you should test them to make sure they're correctly setup. We suggest running two basic diagnostic tests to get started, both of which will rely on programming the Huzzah board and setting things up in the Arduino IDE.

To get started, download the latest version of Arduino IDE (currently 1.8.9; <https://www.arduino.cc/en/main/software>). The Arduino IDE (‘Integrated Development Environment’) is an open-source software programming environment that is specifically designed to program hardware, like the Huzzah microcontroller that we’re using here. Arduino also makes dedicated microcontroller devices, but these devices are more sophisticated and power-hungry than we need for this weather station. Once this software is downloaded and installed, you’ll need to install a few libraries and input some settings to Arduino. A library is a set of files that are written to provide a specific set of functions that the main programming environment doesn’t provide. In our case, we’ll need to provide the libraries for the BME280 board and DS18B20 thermometer since these are digital devices whose signal input and output needs to be carefully controlled and interpreted.

Download and install the following libraries. For the final three libraries, simply select the green ‘Clone or Download’ option on the Github page.

- ESP8266 (chip library): <https://github.com/esp8266/arduino>
Following the instructions here, the libraries are provided through a web resource, rather than being physically stored on your computer. You’ll need to direct Arduino to this resource by implementing the following steps in the Arduino software:
 - Open the Preferences window
 - Look for the *Additional Board Manager URLs field*, and enter: http://arduino.esp8266.com/stable/package_esp8266com_index.json
 - Go to the Tools > Board menu, and select the “Boards Manager...” option. Install the *esp8266* platform.
 - Go back to the Tools > Board menu and select ‘Adafruit Feather HUZZAH ESP8266’
- BME280 library: https://github.com/adafruit/Adafruit_BME280_Library/releases
We recommend that you download the v1.0.9 version (or higher) of its library. This will download a .h (header) and .cpp (C++) file.
- Adafruit sensor: https://github.com/adafruit/Adafruit_Sensor
- OneWire library (for the DS18B20 thermometer): <https://github.com/PaulStoffregen/OneWire>
- ThingSpeak (IOT web host): <https://github.com/iobridge/thingspeak>

Device connection

You’ll need to use a USB cable to transfer the code to the board’s memory (with a mini-USB connection on the board side). In order to get your computer and Arduino to recognize this cable, you’ll need to download one final file. This file is a .kext document, which your computer needs to recognize this type of external device. SI Labs develops this resource, so visit the link below to download this file:

https://www.silabs.com/community/interface/knowledge-base.entry.html/2017/01/10/legacy_os_softwarea-bgvU

and based on your operating system, download the appropriate package. If you’re working on a Mac, the version that was developed for the OS 10.12.x (and higher) operating systems has a widely-recognized problem and should be avoided. The legacy version works, however, so we recommend downloading the CP210x VCP Driver (under Mac OSX 10.6 Software).

4.1.a Blink Test

This test is a simple code to make sure the Huzzah microcontroller and USB cable communication are functioning properly. It will make the built-in LED on the microcontroller blink every half a second! Arduino provides a number of built-in example codes (Arduino calls these 'sketches') to get you started.

1. Go to File > Examples > ESP8266 > Blink


The following code should appear in a new window:

```
void setup() {
    pinMode(0, OUTPUT);
}
void loop() {
    digitalWrite(0, HIGH);
    delay(500);    //delay time is in milliseconds
    digitalWrite(0, LOW);
    delay(500);
}
```

This code sets the blue LED on the board to go to a 'high' state (full brightness) for half a second, and then a 'low' state (off) for half a second.

2. Plug the USB cable into your computer's USB port and carefully fit the other end onto the mini-USB port on the Huzzah board.
3. Next, go to Tools > Board and select the 'Adafruit Feather HUZAZH ESP8266' option.
4. Go back to Tools > Port and select the "dev/cu.usbserialport..." option.

Note: if this option isn't available and the USB cable is properly connected, you may need to restart your computer to make sure the proper .kext file is installed and recognized.

5. Compile the code by clicking the check-mark 'Verify' icon. 

This step will cause the Arduino IDE to check the code to make sure it's properly written.

The red LED on the board should shine with a dim intensity. If the LED has a bright intensity then it's not in bootloader mode (the ability to store the code in the board's memory). The recommended fix is to connect a 220Ω resistor between ground and the GPIO #0 pin; the LED should decrease to a dim intensity, indicating that it's now in bootloader mode. Once the board is in bootloader mode, disconnect the resistor from the ground connection.

Next, click on the right-arrow 'Upload' icon (the second button from the left on the sketch window) to upload the code to the board. A series of dots will appear, indicating upload progress.

The sketch will start immediately - you'll see the LED blinking. Hooray! (or perhaps Huzzah!)

Supplemental Information (explanation of code):

First, we have the 'setup' function. This is run when the reset button is pressed. It is also run whenever the board resets for any reason, such as power first being applied to it, or after a sketch has been uploaded.

In this case, there is just one command there, which, as the comment states tells the Arduino board that we are going to use **pin 0** as an output.

It is also mandatory for a sketch to have a 'loop' function. Unlike the 'setup' function that only runs once, after a reset, the 'loop' function will, after it has finished running its commands, immediately start again.

Once you've run the code, go back and change with the numbers to adjust the speed of the blinking. For example:

```
void loop() {
  digitalWrite(0, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(0, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

Inside the loop function, the commands first of all turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off and pause for another second.

4.1.b Wifi test

Once you've got the LED blinking, let's go straight to the fun part, connecting to a web server. The libraries that you loaded should provide an example that you can use for this step as well. Go to File > Examples > ESP8266Wifi > WifiClient. This sketch will try to connect to a basic website and will let you know if it was successful.

You'll need to edit two lines of code to tell the Huzzah which wifi network to connect to. These two lines are:

```
const char* ssid = "yourssid";
const char* password = "yourpassword";
```

Enter the wifi network name in place of `yourssid` and enter the password in place of `yourpassword` (keep the quotes for both entries). Put the Huzzah into bootload mode, then upload code via IDE.

4.2 ThingSpeak Account

There are two ways to see the data from the sensors. The Arduino IDE has a Serial Monitor (Tools > Serial Monitor) that shows any data streams or output characters that get reported back from the device when it's connected with the USB cable. You'll eventually want to make the station fully autonomous, so we'll use the ThingSpeak web portal system (it's free) to host the data streams that are generated from the weather station's sensors.

You'll need to create a ThingSpeak account. ThingSpeak (<https://thingspeak.com/>) is a data analytics software platform that can be used to receive the data your weather station collects in real time, and plots it on multiple channels.

Once your account is set up, make one channel for each data stream (temperature, pressure, humidity, wind speed). The weather station code will include a section that will allow you to specify the Channel number and API Key identifier. You can find these two parameters once your account is set up. If you're not sure how to find these, start by clicking on the Channels > My Channels menu item. Next:

- Click on 'Settings'. The Channel ID should be listed with your account information.
- Click on 'API Keys'. You'll want to reference the 'Write API Key'.

4.3 Weather Station Code

The code that you'll want to eventually load to the Huzzah for the complete functionality of the weather station is fairly long. It appears in Appendix B. Copy and paste it into a new Arduino sketch, and implement the following edits to the code:

1. Update ssid and password (similar to the Wifi test sketch above).
2. Update the myChannelNumber value (no quotes) and the myWriteAPIKey identifier (with double quotes).
3. After plugging in the USB cable to the Huzzah board and your computer, compile and upload the code to the board. If the board's red LED isn't in a dim state you should connect the free end of the 220 Ω resistor to a ground connection (e.g. the T5 terminal negative/ground port).

NOTE: Once the code is successfully uploaded, it will be permanently stored in the board's flash memory. This will *only* happen if the 220 Ω resistor is disconnected from its ground connection, so make sure you disconnect this resistor after the code has been uploaded. If you don't disconnect the resistor, the code will be erased once the board loses power.

4.4 Perma-proto Board Layout

Supporting tutorial video: <https://youtu.be/5YchgJTYrB8>

Start by laying out the wires (before soldering), resistors, terminals, and boards (all before soldering), with the wires sticking through the corresponding board holes. Make sure that you have the appropriate wire lengths and hole connections. It's much easier to solder than to de-solder, so **double-check** the wiring layout with the diagram in Fig. 3! Most of the components will need to be soldered from the bottom side of the board. The downside of this approach is that the bottom side doesn't have row and column labels, so once the wires are laid out we suggest you bend the wires so they stay in place when you flip the board over to solder the connections.

Note: If you have access to a multimeter, measure and record the resistance of R_1 and R_2 . They should each be 10 k Ω but these types of resistors are well known for varying from their nominal value, and differences from their nominal value could affect the accuracy of the measurements. Multimeters are also very useful for checking whether your solder joints were successful, by running a continuity test (<http://en-us.fluke.com/training/training-library/test-tools/digital-multimeters/how-to-test-for-continuity-with-a-digital-multimeter.html>).

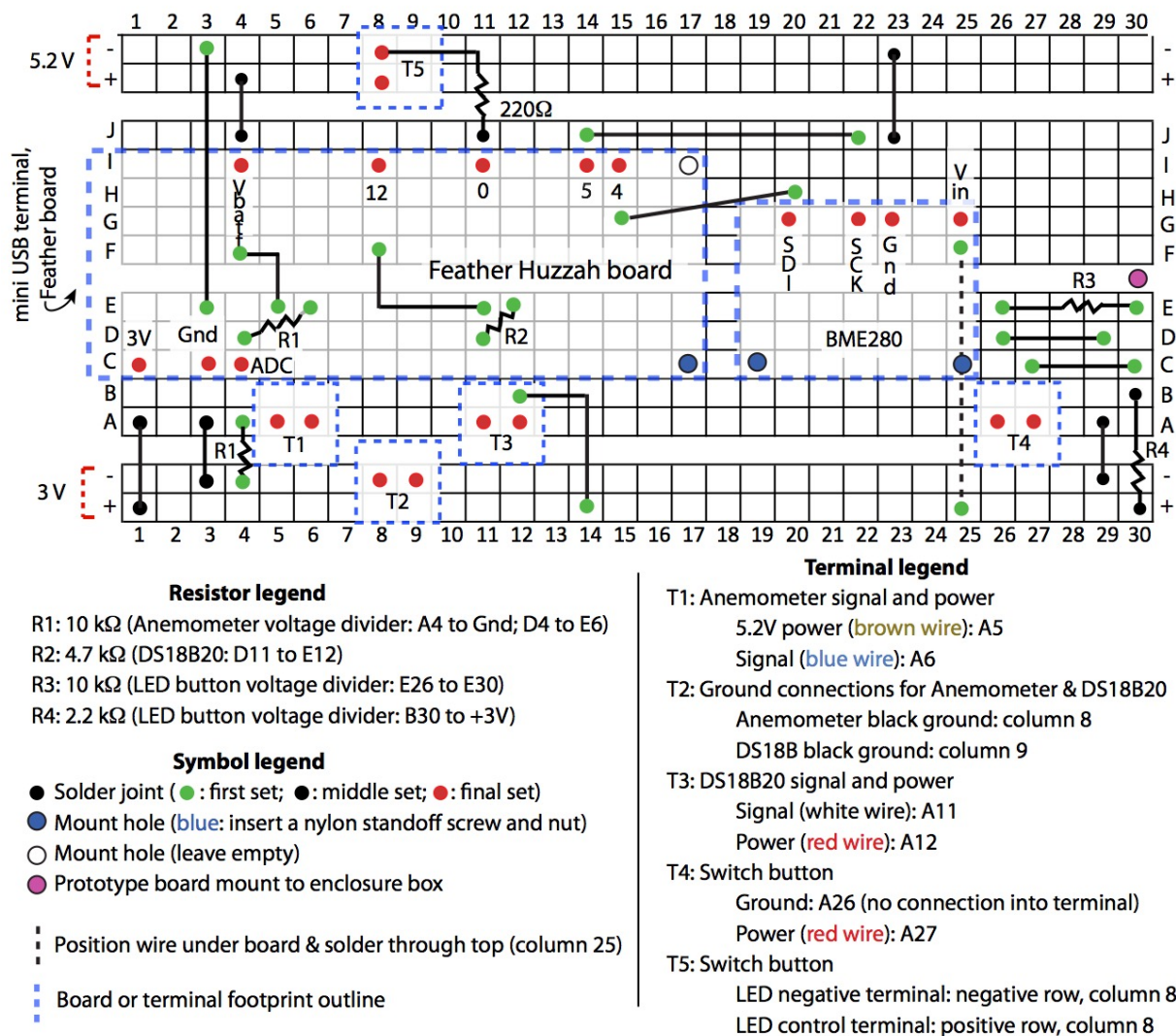


Figure 3: Perma-proto board wiring diagram. This diagram appears in Appendix A in a larger landscape format, since you'll want a large version when you prepare the board for soldering.

All solid circles in the diagram indicate a solder joint. All straight lines indicate a wire, and the lines with a ripple in the middle indicate a resistor. The key below describes the appropriate resistor value. The blue dashed line represents the boundary of a larger component (board or terminal). The order of the soldering doesn't matter for most of the steps, but some of the connections should be made first ("First priority connections"), and some of the connections should be made last ("Final connections"). The remaining solder joints (black solid circles) should be made in a systematic order from one side of the board to the other.

First priority connections (the solid green circles):

The wires between:

- E3 and column 3 of the -5.2 V negative terminal
- F4 and E5
- F8 and E11

- J14 and J22
- G15 and H20
- D26 and D29
- C27 and C30
- Positioning on the bottom of the board and soldering on the top: F25 and +3V (column 25)
- The resistors between:
 - D4 and E6
 - A4 and the ground terminal on the 3V rail (column 4)
 - D11 and E12
 - E26 and E30

Final connections (the solid red circles):

- The Huzzah board; only the eight circles for the board’s pins need to be soldered.
- The BME280 board; only the four red circles need to be soldered.
- The five terminals. With the exception of terminal 5, note that the terminal ports should face away from the center of the board, towards the nearest edge (bottom horizontal edge in the diagram). The T5 ports should face to the right side of the board (opposite the mini-USB port).

Note: The video shows 3 incorrect wire connections that were later fixed but not re-filmed. Figure 3 shows the correct connections and these mistakes are initially identified with pop-up animations in the videos but to clarify:

- The video shows a jumper wire extending between A1 and the bottom ground rail. This wire should extend between A1 and the bottom + voltage rail.
- The video shows a jumper wire extending between J15 and J22. This wire should extend between J14 and J22.
- The video shows a jumper wire extending between G14 and H20. This wire should extend between G15 and H20.

4.5 Electronics assembly and testing

Supporting tutorial video: <https://youtu.be/ScX1iTSYWiI>

Once these components are soldered we can start by testing one component at a time. We’ll start with the BME280 board and continue to add components until we’ve worked through each major subsystem. The solar power integration will come last, since the USB connection to your computer will provide sufficient power to evaluate all the sensors. The wiring diagram of the entire completed electronics system is shown in Fig. 4, and Fig. 5 shows a photo with the completed board and nylon support posts in place.

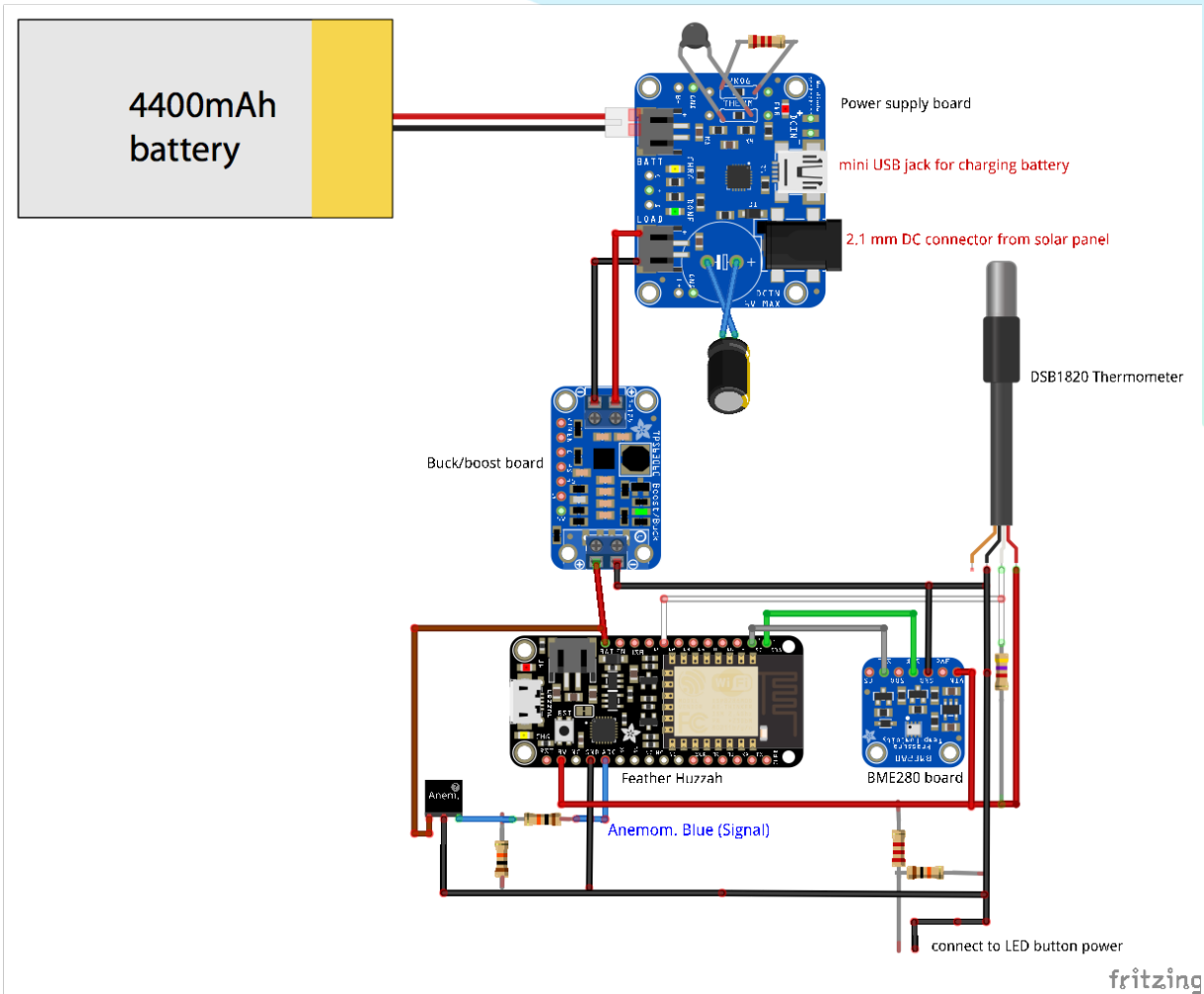


Figure 4: Board connection diagram. This diagram also appears in a larger version in Appendix A.

We'll eventually need the battery charged, so gather the power management board, the USB A/mini B cable, and the LiOn battery. Plug the battery's JST plug into the BATT port on the power board, and plug the USB's mini B terminal into the corresponding port on the board and the other 'A' terminal into a powered USB port (your computer or an AC power adapter plug). The red 'pwr' LED and the orange 'CHRG' LED on this board should illuminate. You'll know that the battery is fully charged when the green 'DONE' LED illuminates and the CHRG LED turns off.

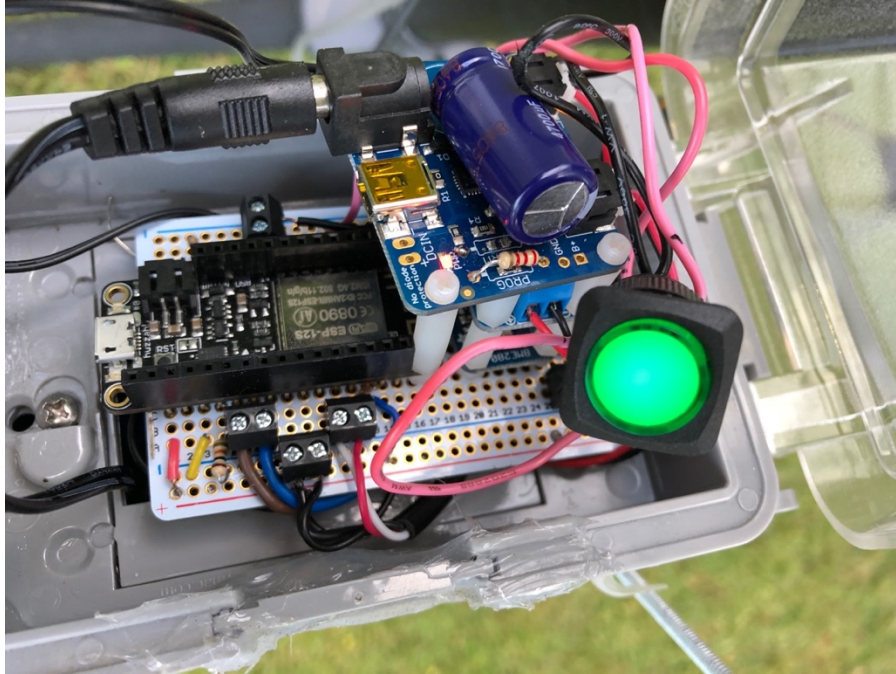


Figure 5: The completed board, with the nylon standoff posts in place for the boards.

4.5.1 Huzzah + BME280 evaluation

1. Connect the USB cable to your computer and to the Huzzah. You should do this step *before* the final station assembly.
2. The Huzzah's red LED should illuminate. Remember that this light has multiple brightness settings (off; dim; full). If the red LED becomes dim then it's ready to receive a new program ("in bootload mode"). A relatively bright intensity indicates that it's not in bootload mode and won't be able to update the program. In this case, solder a 220Ω resistor to J11 (and therefore, the GPIO #0 pin) and connect its free end to the negative port for the T5 terminal (ground, top row). The red LED should now shine in a dim state, indicating that the chip is now in bootload mode. Disconnect the resistor once the light is in its dim state.
3. Compile and upload the weather station code. The blue LED on the Huzzah board should illuminate during this process.
4. Open the Serial Monitor (Arduino: Tools > Serial Monitor). If all the connections are properly functioning, you should see the following output from the board:

```
GLOBE Weather Station
BME280 sensor identified
Connecting to <your wifi ssid>
WiFi connected
IP address: <your IP address>
Connecting to api.thingspeak.com
```

The average temperature, humidity, pressure, wind speed, ADC value, and corrected wind speed should all be reported next.

Note: GLOBE expects this type of instrument to report a reading every 30 minutes. The timing in the code is written so that 100 readings from each sensor are acquired prior to each 30 minute report time. The average value of these 100 data points then gets transmitted via wifi.

If, however, it's unable to connect to ThingSpeak, you should troubleshoot by checking that:

- Your wifi network is operational
- The wifi network settings are correct (these should have been verified in the previous Wifi test that you ran)
- The ThingSpeak library was properly downloaded.
- If you receive the following message,
`Could not find a valid BME280 sensor, check wiring!`

then:

- this could be an indication that the wire connections or solder joints that connect to the BME280 sensor need attention. Trace these connections and double-check that your solder junctions are correct; a continuity test with the multimeter may be helpful.
 - And/or it could be an indication that the BME280 libraries aren't correctly installed or there are multiple copies of these.
5. If the pressure, humidity, and temperature (this is the BME280 temperature sensor output, not the DS18B20 temperature sensor) values seem in the right ballpark, conduct an easy test to make sure the board is responding. You can do this by exposing the BME280 to a heat source and seeing if the temperature increases (and subsequently decreases once the heat source is removed). Easily available heat sources include your hot breath, a warm coffee mug, or the soldering iron tip. Just make sure this source is relatively close to the BME280 board but not so close (or so hot!) that it melts any of the solder junctions or board components!
 6. Next, check the data stream on your ThingSpeak account. Note that the temperature that gets transmitted to ThingSpeak is that of the DS18B20 thermometer, so the ThingSpeak temperature data will be inaccurate (for now). The data should update every 30 minutes.
 7. Disconnect the USB cable to power down the board.

4.5.2 Huzzah + BME280 + DS18B20 evaluation

1. Next, plug the three DS18B20 cables into the terminal connections on the board. We suggest feeding the black ground wire into the leftmost port of the T2 terminal (the ground – column 8 position). The white wire should feed into the left port of the T3 terminal (the A11 position) and the red wire should feed into the right port of the T3 terminal (A12).
2. Plug the USB cable back in to power up the board, and check the ThingSpeak data stream to make sure the temperature matches your room temperature.
3. Disconnect the USB cable once the DS18B20 temperature is verified.

4.5.3 Huzzah + BME280 + DS18B20 + USB DC solar power + buck/boost evaluation

To evaluate the anemometer we'll need at least 5 V, and the Huzzah only puts out 3 V. We'll need to add the power boards into the mix since the buck/boost provides 5 - 5.4 V. To do this you'll need to also gather the buck/boost board, power management board (and battery, at least

partially charged), a JST cable, wired LED switch, PH00 (jewelry) Phillips screw driver, two 12 mm M2.5 nylon posts, and two nylon screws. It will be helpful to reference Figs. 4 and 5.

1. If the battery is fully charged, disconnect the USB plug to the power management board.
2. Mount the buck/boost board to the BME280 board:
The buck/boost will be positioned directly above the BME280 board and held in place by two standoff posts. These posts should already be attached to the BME280 board. Align the buck/boost such that these posts are directly below its mount holes (on the side nearest the Adafruit flower logo on the buck/boost board, near the positive 3-12V terminal junction). Secure each post (temporarily) with a nylon screw; turn the screw until it's hand tight.
3. Connect the power management board to the buck/boost board, using a JST cable: Insert the JST's red wire into the positive terminal on the 3-12 V side of the buck/boost board, and gently tighten the screw to clamp the wire in place. Repeat, using the JST's black wire and the corresponding negative terminal.
Note: If the wires slip out from the terminal, it may be helpful to add a small bit of solder to each bare metal end of the wire, to give the clamp more material to grip to.
4. Connect the buck/boost board to the LED switch. Locate the free black wire from the switch's negative terminal and insert it into the negative port on the other terminal of the buck/boost board.
5. The LED switch button has two positions. The 'in' position connects the LED's control terminals and allows the electrical signal to flow across ('on' state). The 'out' position prevents the signal from passing across the control terminals ('off' state). Adjust the button so that it's in the out/off state.
6. Locate the free JST plug that's connected to the buck/boost's 3-12 V terminal and plug it into the LOAD port on the power management board.
7. A green LED should illuminate on the buck/boost board, indicating that the electrical connections between the power management and buck/boost boards are correctly attached.
8. Press the LED switch so that it's in the in/on state. Power from the battery should now be flowing to the perma-proto board and the LED switch should illuminate. The Huzzah red LED should also illuminate (don't be alarmed if it doesn't – sometimes it decides to not activate...). At this point you won't need the USB cable any longer.
9. Mount the solar charge board to the buck/boost and Huzzah boards:
 - a. Replace the nylon screws that secure the buck/boost board with two 12 mm nylon posts. The charge board will be supported via three posts: the two extending from the buck/boost and the 25 mm post extending from the Huzzah. The 12 mm nearest the Huzzah (in the middle of the three) will simply rest against the bottom of the charge board while the other two posts will be secured through the mount holes on the charge board with nylon screws. Orient the charge board such that the solar panel barrel plug faces the Huzzah board. Adjust the posts so that they can be secured via the nylon screws.

4.5.4 Huzzah+BME280+DS18B20+USB DC solar power+buck/boost evaluation + anemometer

Now we can evaluate the anemometer!

1. Put the LED switch into its off state, so that its light is off. The board will now be unpowered.

2. One end of the anemometer cable has 3 bare wires and the other end has a round 4-pin termination connector. The round connector attaches to the anemometer; leave it disconnected for now.
3. Insert the black (ground) wire from the anemometer into the other port on the T2 terminal (perma-proto board; this should be the ground/column 8 position). Clamp the wire by screwing the port screw (clockwise) with the PH00 screwdriver.
4. Insert the blue (signal) wire from the anemometer into the right port on the T1 terminal (A6 position) and adjust its screw.
5. Insert the brown (power) wire from the anemometer into the left port on the T1 terminal (A5) and adjust its screw.
6. Connect the other end of the anemometer cable to the anemometer by lining up the notch on the connector with the corresponding notch on the anemometer port.
7. Put the LED switch into its 'on' state.
8. Spin the anemometer (or use a fan) and monitor the wind speed reading on ThingSpeak or use a voltmeter to read its voltage (across T1 column 6 screw and any of the ground connections on the perma-proto board). The voltage should be 0.4 V when the anemometer isn't spinning, and between 0.4 and 2 V when the anemometer is spinning.

4.6 Anemometer calibration

4.6.1 Voltage Divider correction (optional but recommended for accuracy)

This step is helpful to perform as it will make the wind speed measurement more accurate, but it requires you to have a multimeter in order to measure resistance.

The anemometer generates a 0.4 V voltage when the wind speed is zero and a maximum output of 2 V. The maximum input voltage ($V_{in,ADC}$) of the Huzzah's ADC (analog-digital converter) channel is 1 V, which is why we need to use a voltage divider circuit to decrease the anemometer's output voltage ($V_{out,anem}$) in half. This voltage divider arrangement (using the two resistors that are connected to the anemometer output signal path) is the following:

$$V_{in,ADC} = V_{out,anem} \left(\frac{R_2}{R_1 + R_2} \right)$$

As noted above, the resistors in the wiring diagram specify a 10 k Ω value but most resistors deviate slightly from their nominal value. If both resistors have the same value (10 k Ω or otherwise) then the circuit will work as expected, but the wind speed calculation will be incorrect if they're significantly different from each other.

If the ratio of $R_2 / (R_1 + R_2)$ doesn't equal 0.5, the weather station code will need to be updated. In this case, go to the end of the code and find the line that reads:

```
float Kvd = 0.5; // voltage divider to match ADC 1 V max input
```

and change the 0.5 value to the value you get based off your measurements. Fixing this ratio will allow the maximum output of the anemometer to be measured by the ADC.

4.6.2 Huzzah board offset (required)

The second, required step, is to check the offset voltage that the Huzzah board *might* introduce to the measurement. When there's no wind speed, the anemometer should output 0.4 V, and the voltage divider should reduce this to 0.2 V. This voltage then gets digitized by the ADC component of the Huzzah board. The board has 10 bits of digital resolution over a 1 V full scale, resulting in 204.8 digital steps when the anemometer is at rest. Only integer digital steps are registered, so this 204.8 value will likely be rounded up to 205 digital steps. This value is what gets returned in the line of the code (near the bottom, in the `Wind()` function):

```
ADCmeas = analogRead(analogInPin); // number of digital steps that
need to be converted to a voltage via Vres
```

The code is written so that this value gets reported to the Serial Monitor. Check, and if necessary update the code, to make sure this value is correctly measured:

1. Connect the USB cable to the computer and Huzzah board.
2. Make sure the anemometer is properly connected to the Huzzah board via its own cable.
3. Make sure the anemometer is at rest and not spinning.
4. Open the Arduino IDE software and in Arduino, open the Serial Monitor.
5. Assuming the Huzzah board still has the `GLOBE_WeatherStationCode` stored in its flash memory, the Huzzah should periodically report the temperature, pressure, humidity, and wind speed. It should also report the "ADC Value". This value is the number of digital steps that the ADC pin is receiving from the anemometer. Record this value.
6. If the ADC Value is 205, nothing needs to be done, and the anemometer is correctly calibrated. The wind speed should also read as 0 mph, or something very close to 0 mph.
7. If the ADC Value is something *other* than 205 and the wind speed reading is something higher than 0 mph, you'll need to add an offset voltage for the board using the following steps:
 - a. Multiply the ADC Value by the board's voltage resolution (1/1024 -- or simply divide the ADC Value by 1024...). This value is the board's offset voltage.
 - b. In Arduino IDE, update the `GLOBE_WeatherStationCode` file to incorporate the accurate offset voltage. This value is considered in the `Wind()` function, near the bottom of the code. Locate the following line:

```
float Oadc = 0; // compensates for ADC offset (V)
```

and update the `Oadc` value to the offset voltage you calculated in step 7(a) above.

- c. Save the code, compile it, and upload it to the Huzzah board.
- d. Making sure the anemometer is still at rest, check the Serial Monitor output to confirm that the wind speed is now zero.

Final electronics check

The electronics are now fully connected and successfully tested. In order to assemble everything together, disconnect the anemometer and DS18B20 wires from the perma-proto board, and disconnect the battery cable from the solar charger board. Toggle the LED switch so that it's back in its out/off state to preserve the battery charge.

5

HARDWARE MOUNTING AND ASSEMBLY



The system is supported via a mounting bracket that clamps to the top of an adjustable tripod. This horizontally-oriented bracket supports the anemometer, radiation shield, and thermometer on one side, and the solar panel and system electronics on the other side.

The supporting tutorial video for this section provides an overview for mounting the assembled radiation shield, anemometer, solar panel, and electronics box: <https://youtu.be/itRORff3Mqc>

5.1 Radiation Shield Assembly

All of the components in this radiation shield assembly section are from Davis Instruments, and the steps below are edited selections from the user manual supplied by Davis (www.davisnet.com/product_documents/weather/manuals/07395-093_IM_07714.pdf).

1. Place the sensor cable into the notch on one of the clip mounts and hold it in place. Make sure to hold the clip mount so the raised semi-circle at the top of the notch faces up.
2. Position the second clip mount over the first, with the notch facing in the opposite direction, securing the sensor cable between the two notches. When positioning the second clip mount, make sure the raised semi-circle faces down.

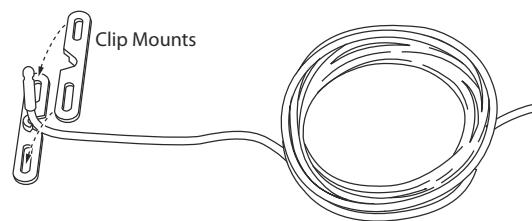


Figure 6: Temperature probe clip.

3. Position the clip mounts over two of the mounting posts on the closed plate. Make sure you orient the clip mounts as shown in the illustration below.
4. Attach the clip mounts to the mounting posts using two of the #4 x 1/2" pan head self-threading screws and two of the #4 flat washers.

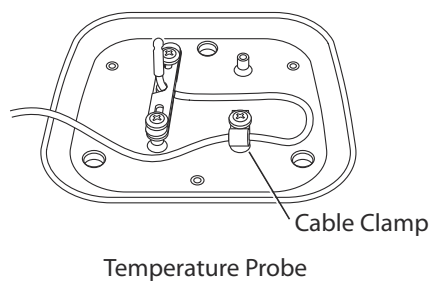


Figure 7: Temperature probe clamp.

5. Once secured, adjust the position of the sensor so the probe and approximately 1/4" (6 mm) of cable protrude from the clip mounts.
6. Place the cable clamp around the sensor cable approximately 8" (20 cm) from the probe.
7. Secure the cable clamp to one of the remaining mounting posts (using a #4 x 1/2" pan head self-threading screw and a #4 flat washer) so that a loop of cable is formed. Make sure to tighten the clamp with the flat side up and the bulge side down. Tighten the screw completely so that the cable cannot move within the cable clamp.

Attach the support plate

To attach the support plate, you will need the cover plate, the support plate, three #8 x 2 3/4" pan head screws, three 1" white nylon spacers, three #8 flat washers, three #8 split lock washers, and three #8 hex nuts.

1. Slide the three #8 x 2 3/4" pan head screws up through the non-threaded holes in the shield support plate. Make sure the side of the support plate marked "UP" is in fact on top as you slide the screws in from the bottom.
2. Place the cover plate over the screw ends protruding from the support plate.
3. Place a 1" spacer over each of the screw ends.
4. Secure the support plate and spacers to the cover plate using a #8 flat washer, #8 split lock washer, and #8 hex nut on each of the screw ends. Tighten until the support plate is firmly attached to the cover plate.

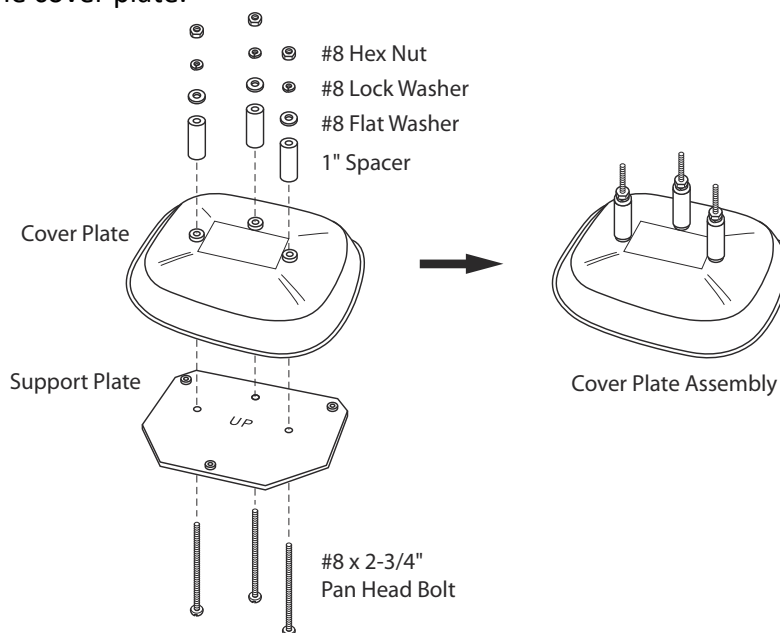


Figure 8: Temperature probe clip.

Assemble the Plates and Studs

To assemble the plates and studs, you will need the cover plate assembly as well as the flat plate and close plate with sensor, **one** #8 x 5" threaded stud with attached #8 push nuts, three #8 flat washers, three #8 split lock washers, and three wing nuts. Note that you will also need the two

#8 x 12" threaded rods from Home Depot; these will replace the other two threaded rods that ship with the radiation shield.

1. Locate the two #8 x 12" threaded rods and select one of the #8 x 5" threaded rods. Slide each of the threaded rods, with push nuts installed, through each of the three holes in the closed plate, flat plate, support plate, and cover plate. When inserting the threaded studs, make sure the short end (when measured from the push nut) goes through the plates. Screw the short end of the threaded stud in until the cupped side of the push nut bottoms inside the recess of the closed sensor plate. Tighten as much as you can by hand.

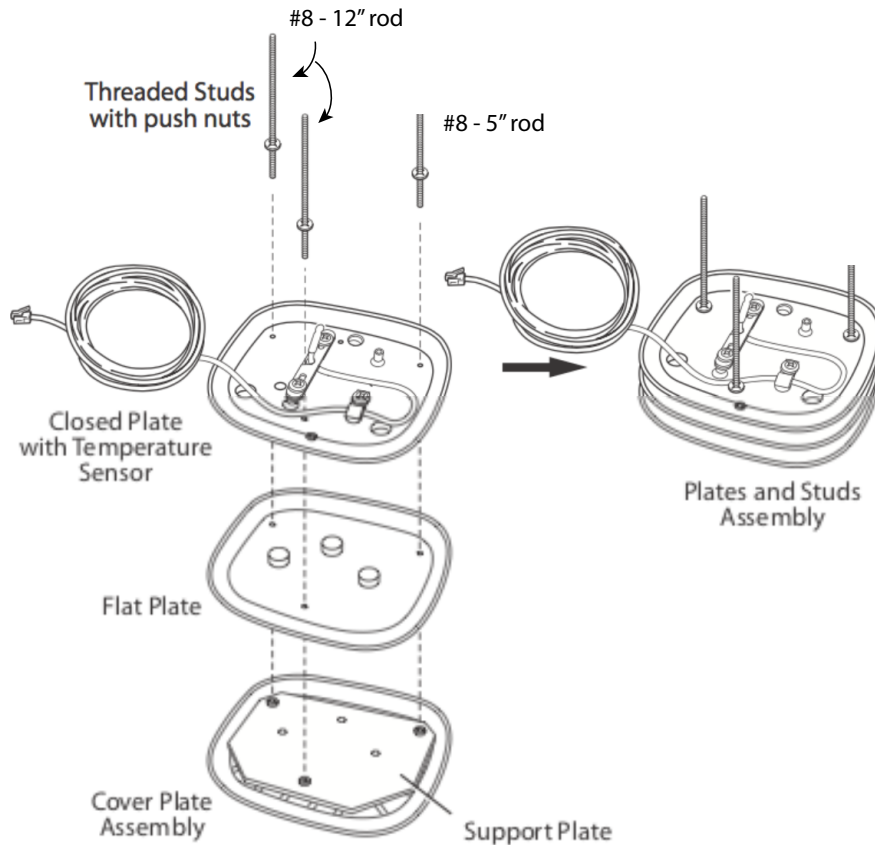


Figure 9: Threaded rod assembly

2. Slide the three open plates and the two remaining closed plates over the threaded stud ends protruding from the top of the plate and studs assembly.
3. Place the flat washers, lock washers and plastic wing nuts over the protruding stud ends.

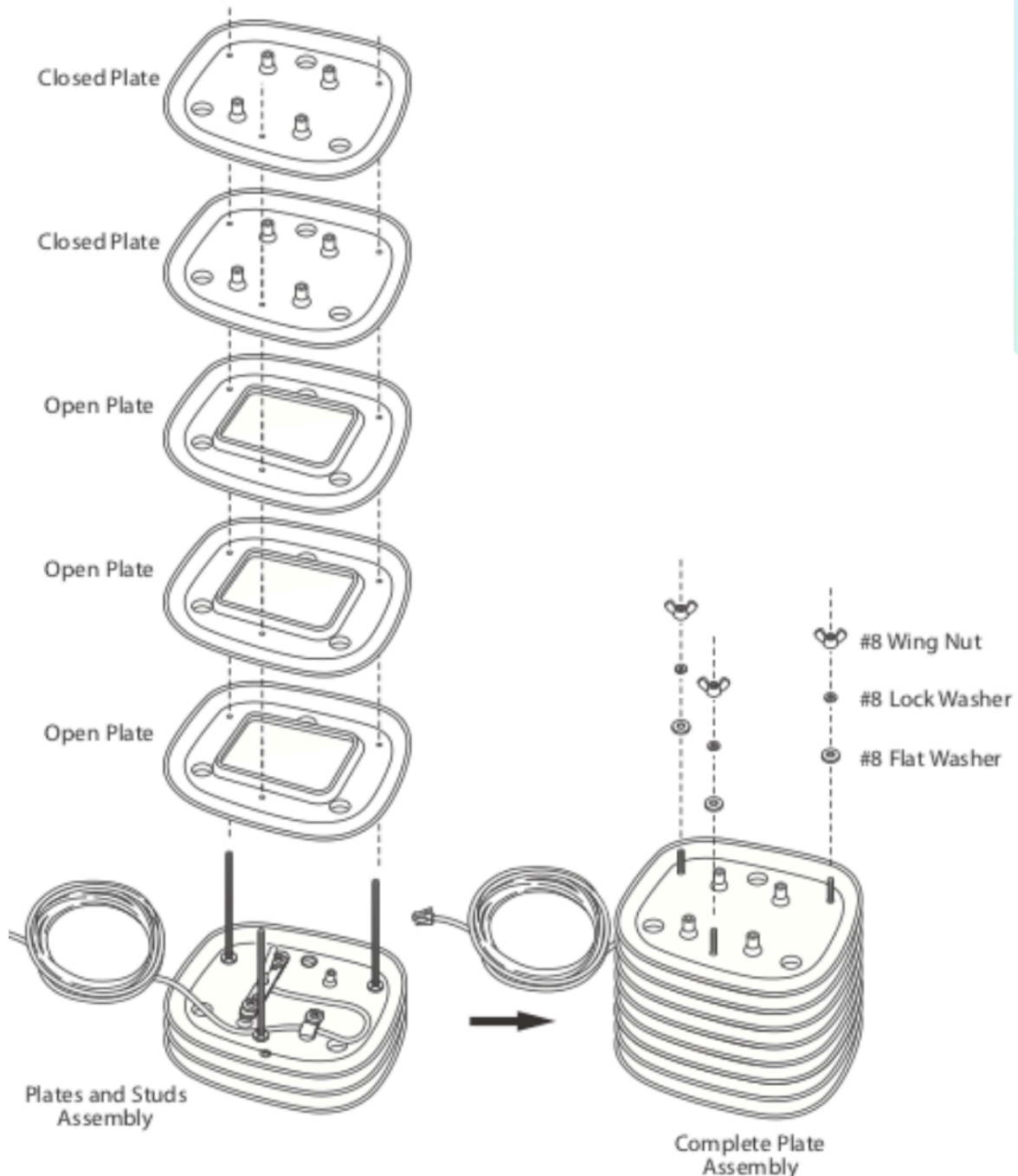


Figure 10: Threaded rod assembly

Mounting bracket

In order to mount the radiation shield to the white right-angle bracket, you will need the radiation shield assembly, and three of the following (each): #8 split lock washers, #8 flat washers, and #8 hex nuts. In order to mount the white bracket to the black tripod horizontal bracket, you will need two of the following (each): ¼-20 machine screws, flat washers, and hex nuts.

1. Slide the stud ends protruding from the top of the radiation shield assembly into the holes on the mounting bracket.
2. Secure the mounting bracket to the radiation shield using a #8 flat washer, #8 split lock washer, and #8 hex nut on each of the screw ends.
3. Secure the radiation shield to the top of the tripod mounting bracket with the ¼-20 screws, washers, and nuts (not shown in image) via the two larger holes in the white bracket.

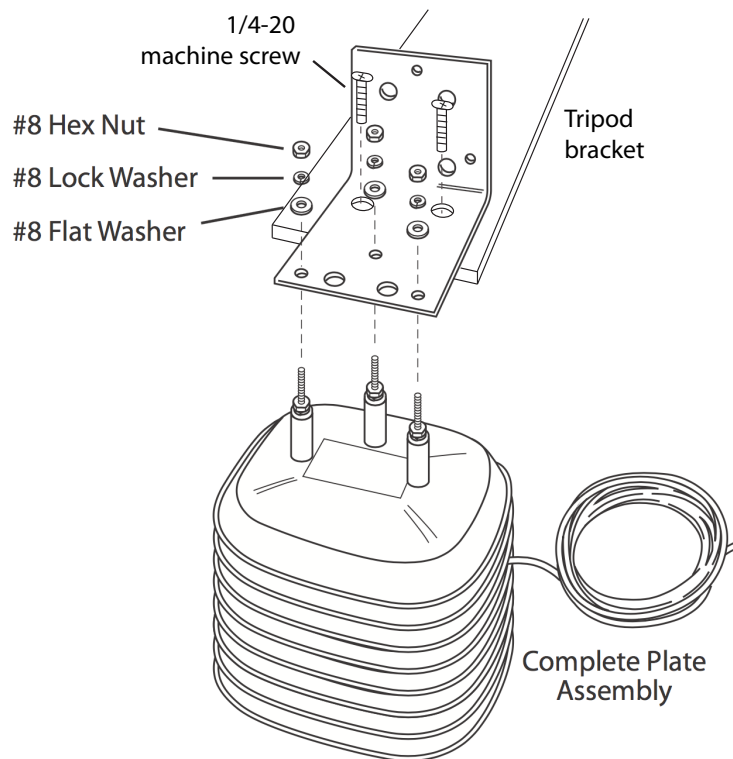


Figure 11: Radiation shield mount. The ¼-20 washers and nuts are not shown.

5.2 Anemometer Mount

The anemometer is supported by the A21Z (Simpson strong-tie) galvanized steel right-angle mounting bracket and four of the following (each): #8-32 machine screws, #8 flat washers, and #8 nuts. All of these components are from Home Depot or your local hardware store.

1. Secure the gray A21Z bracket to the white Davis bracket (see Fig. 12). Orient the gray bracket so that its vertical side is flush against the top of the white bracket's vertical side and so that the gray bracket's horizontal side is at the top (an inverted 'L' orientation). Insert two screws into the gray bracket (each of the holes on the bottom row of its vertical side) and through the white bracket (each of the two larger holes, second row from the top). Secure with a washer and nut. You may need to also use a ¼-20 washer due to the large hole size in the white bracket.
2. Secure the anemometer to the gray bracket by resting the bottom anemometer mount to the top of the horizontal side of the gray bracket. Secure the anemometer with two #8 screws, attaching the washers and nuts from the bottom side of the bracket.



Figure 12: Anemometer and radiation shield bracket mounts.

5.3 Solar Panel Mount

The solar panel is secured via two brackets, on the other side of the black tripod mounting bracket. You'll need the Universal solar panel angled bracket (Voltaic Systems), the solar panel, and the following items from Home Depot: the A23Z right-angle bracket (Simpson strong-tie), and four of the following (each): ¼-20 screws, washers, and nuts. Reference Fig. 1 for this step.

1. Position the right-angle bracket in the same inverted 'L' orientation as the other gray bracket. The horizontal side should mount to the top of the black tripod bracket. Align these brackets so that their outermost and center holes line up. Insert two ¼-20 screws through the top of the right-angle bracket and secure with the washers and nuts from below the black bracket. The right-angle bracket will be at a slight angle from the main axis of the black tripod bracket.
2. Separately, attach the angled bracket to the solar panel by unscrewing two of the black nuts on the long side of the solar panel. This should be the side that the power cable extends from. Align the wide side of the bracket with the two panel screws, carefully feeding the cable through the corresponding large hole in the bracket, and secure the panel with the two nuts.
3. Attach the angled and right-angle bracket by aligning the top of the angled bracket so that it's vertical and flush against the vertical side of the right-angle bracket. The angled bracket will have four vertical slots. Position the two brackets so that the bottom row of these vertical slots aligns with the outer holes of the right-angle bracket. Insert two ¼-20 screws, one into each vertical slot, and secure with a washer and nut on the back side of the right-angle bracket.
4. Make sure that the 2.1mm DC jack adapter cable is plugged into the solar panel's cable, and that the 2.1mm extension cable is subsequently plugged into the adapter cable. The other end of this extension cable will eventually plug into the power management DC jack.

5.4 Electronics Enclosure

The electronics board must be protected against mechanical and water damage but open to the ambient air so that the pressure and humidity can be accurately measured. These requirements are met by housing the electronics in a standard exterior-grade electrical outlet enclosure. The enclosure has two components: a metal box that allows standard electrical conduits to be inserted via a port, and a plastic cover box that has a hinged door. The two boxes attach to each other via two screw connections (a hole in the plastic box and a corresponding threaded mount in the metal box). The plastic box has a foam seal on the side that attaches to the metal box, and the metal box has two metal tabs that allow it to be attached to a bracket (see Figs. 13 and 14).



Figure 13: Electronics enclosure mount.

We'll use one of the three ports on the metal box to pass the electrical signals from the anemometer and thermometer cables, and the power cable from the solar panel. The metal box ships with threaded inserts to plug the two ports that we're not using. You'll need the battery, completed electronics board, the three cables described above, and the following items from Home Depot/local hardware store: Bell weatherproof box, weatherproof while-in-use cover, and silicon sealant.

1. The box ships with two metal tabs (each has two holes) and two screws. On the backside of the metal box you'll see a hole near each of the four corners of the box. These are untapped holes to which you can mount the metal tabs by inserting a screw into one of the holes. Referencing Fig. 14, screw a tab into the top and bottom holes and orient the tab so that it sticks out to the side of the box.
2. Screw a plug into the port on the top and bottom. The plug on the backside of the box should be left open. Both plugs should be sealed with silicon sealant, on the inside and outside of the box.

3. Wrap the battery with bubble wrap and secure with tape. Place the battery at the bottom of the metal box, with its cable available on the side of the box that has the open port.
4. Insert the cables (anemometer, thermometer, solar panel) through the bottom port of the box and wrap them either with a cable tie or a Velcro strap (note that the tripod might ship with a set of Velcro straps). Attach the 2.1mm DC adapter cable to the solar panel power cable, and adjust the four cables (anemometer, thermometer, solar panel, battery) so that you have enough length for them to pass through the metal box up through the plastic enclosure and attach to the electronics board.

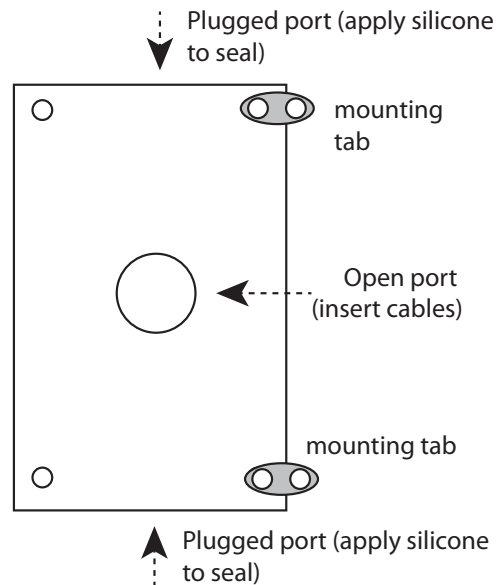


Figure 14: Preparation of the metal box.

5. The weatherproof plastic cover should ship with two #8-32 screws. These screws will be used to mount the cover to the box. Start by inserting one of the screws into the middle threaded hole on the side that has the open port. Leave approximately $\frac{1}{2}$ " of the screw exposed.
6. The electronics board has two larger-sized holes for mounting purposes. One hole is covered by the Huzzah board. Insert the second #8-32 screw into the second hole (near the BME280 board). This hole is surrounded by a copper ring that will likely pop out due to the tight fit of the screw. This is ok! Turn the screw until approximately $\frac{1}{4}$ " is exposed below the board. This screw will eventually pass through the plastic cover and will screw into the metal box, securing all three items to each other.
7. Prepare the plastic cover unit:
 - a. The cover should have its hinge oriented on the top side, opposite the open port. It will be easier to assemble everything with the cover removed, so temporarily slide out the plastic hinge pin that connects the plastic cover with the rest of its body.
 - b. The cover unit ships with three grey plastic tabs that allow an electrical power cord to pass into the unit. Make sure all of these tabs should be securely housed in their slots. Seal them with the silicon sealant on the inside and outside and allow for drying time.
 - c. Note also that the unit ships with multiple face plates for different electrical component mounting options. You should leave the default option (a basic surround piece) in place,

as it will support the electronics board and allow the cables sufficient room to plug in as needed.

- d. Position the unit so it's aligned with the metal box. This is achieved by sliding the #8 screw (in the metal box) into the corresponding through-hole in the plastic cover. Adjust the screw so that the connection is hand tight. Again, this screw should be on the side that has the open port and opposite the cover hinge.
8. Adjust the four cables so they feed up through the cover unit. With the unit oriented so that the hinge is at the top, position the battery cable so that it's on the lower left side of the open space, and the other cables so they're on the middle right side of the open space.
9. Position the electronics board in the cover unit such that its screw is aligned with the hole in the top side of the plastic cover unit. There are two holes on each side, and you'll want to use the one that's further from the edge. It likely has a raised plastic cover piece next to it. Turn the screw so that the board is secured in place but not so tight (yet) so as to allow the cables to be adjusted as necessary.
10. Insert the cable wires into the terminal ports. For this step you'll want a small Phillips-head screw driver; adjust each terminal screw so that its port is open. You should reference the Fig. 3 (or A1) wiring diagram.
 - a. Start with the ground wires and terminal T2; close the terminal port using the screwdriver after each wire is inserted. Insert the black wire from the anemometer into column 8 first, then the black wire from the thermometer into column 9.
 - b. The anemometer's blue signal wire should be inserted into the right port of the T1 terminal (A6) and its brown power wire should be inserted into the left port of this terminal (A5).
 - c. The thermometer's white signal wire should be inserted into the left port of the T3 terminal (A11) and its red power wire should be inserted into the right port of this terminal (A12).
 - d. Plug the battery's white JST plug into the power management board (upper-most board), in the 'BATT' port.
 - e. Plug the solar panel DC cable into the cylindrical black port on the same power board.
 - f. Note: If the battery is charged and/or the solar panel is exposed to direct sunlight, this board should enable at least one red 'PWR' LED light.
11. Adjust the mounting screw on the electronics board so that the board is secure but not crushing any of the wires.
12. **Note** that the board terminals can be finicky to adjust such that the wires are fully clamped down! **Double-check** that each of the wires is clamped into its terminal and that none of the wires are making electrical contact with another wire (i.e. no bare metal should be visibly touching the bare metal of another electrical component).
13. Position the LED switch so that it is visible through the cover and that the cover can be closed without crushing any of the components. Press the LED button in, so that power is sent from the buck/boost board to the rest of the electronics board. The LED light should illuminate once pressed in and sufficient power is available from the battery or solar panel.
14. Re-attach the plastic cover with its plastic hinge pin and close the cover over the unit. The cover is secured with a plastic clip that's part of the cover, and make sure it's securely closed.
15. Check the wifi signals to make sure the unit is properly functioning.

5.5 Mounting the Electronics Enclosure

For this step you'll need the electronics enclosure, the tripod/weather station assembly, and the following items from Home Depot: four #8-32 nuts and two #8 washers. The electronics enclosure will be supported by the two 12" threaded rods that extend from the radiation shield.

1. Once you're satisfied that the weather station is functioning, apply a generous coat of sealant to the edge between the plastic cover and metal box.
2. Unscrew the screws that hold the metal tabs on the back of the box by half a turn or so; the goal here is to be able to adjust the orientation of the metal tabs but to keep them attached.
3. Screw a #8-32 nut onto each of the 12" threaded rods (that extend out of the radiation shield), approximately 3" from the end of the rod.
4. With the plastic cover facing upward, position the enclosure so that the 12" threaded rods pass through the two metal tabs, and slide it up until the top of the plastic enclosure rests against the bottom of the radiation shield.
5. Slide a washer and screw on a #8-32 nut on each 12" rod so that it keeps the enclosure in place. Adjust each nut from step 3 so that it's clamped against the top of the metal tab. The enclosure should now be held in place with the cables extending out the bottom of the enclosure.
6. Re-tighten the screw that holds each metal tab to the electronics enclosure.

5.6 Weather Station Deployment

At this point the system should be fully assembled and operational. If the sun isn't shining and the battery is fully drained, the system should come back online once the sun provides sufficient power for the system to operate. The system has been tested in Boston in March (cloudy conditions, 30 – 40°F) after starting out with a fully charged battery and was continuously operational for at least a year.

Keep the following aspects in mind when the system is deployed:

- Charge the battery fully before deploying the system. Put the LED switch into its off state to preserve the battery life until the system is ready to be deployed.
- For system mechanical stability, adjust the tripod legs so they're at their widest spread angle. Position a cinder block near each leg and wire it to the nearest hinge connection for that leg (or otherwise secure the tripod based on your system's roof configuration).
- For accurate anemometer readings, adjust the tripod height so the anemometer is at least 6 ft off the ground.
- The station should also be at least 6 ft from any heating and cooling vents.
- The station orientation should be such that the solar panel faces due south.
- Use a cable tie or Velcro strap to secure the various cables to the tripod.
- Once the station is set up and ready for measurements, make sure the LED switch is put into its on state and is illuminated.

6

DIY ANEMOMETER



This section describes how you can make, program, and assemble your own anemometer. As an educational exercise, it is *not* meant to replace the commercial anemometer that was described for the weather station kit above. It is important to keep in mind that this section is a beta version, meaning that it is a work in progress. It is also important to note that while the components required (see BOM #3) are expensive when summed, this system is slightly cheaper than the purchased anemometer option.

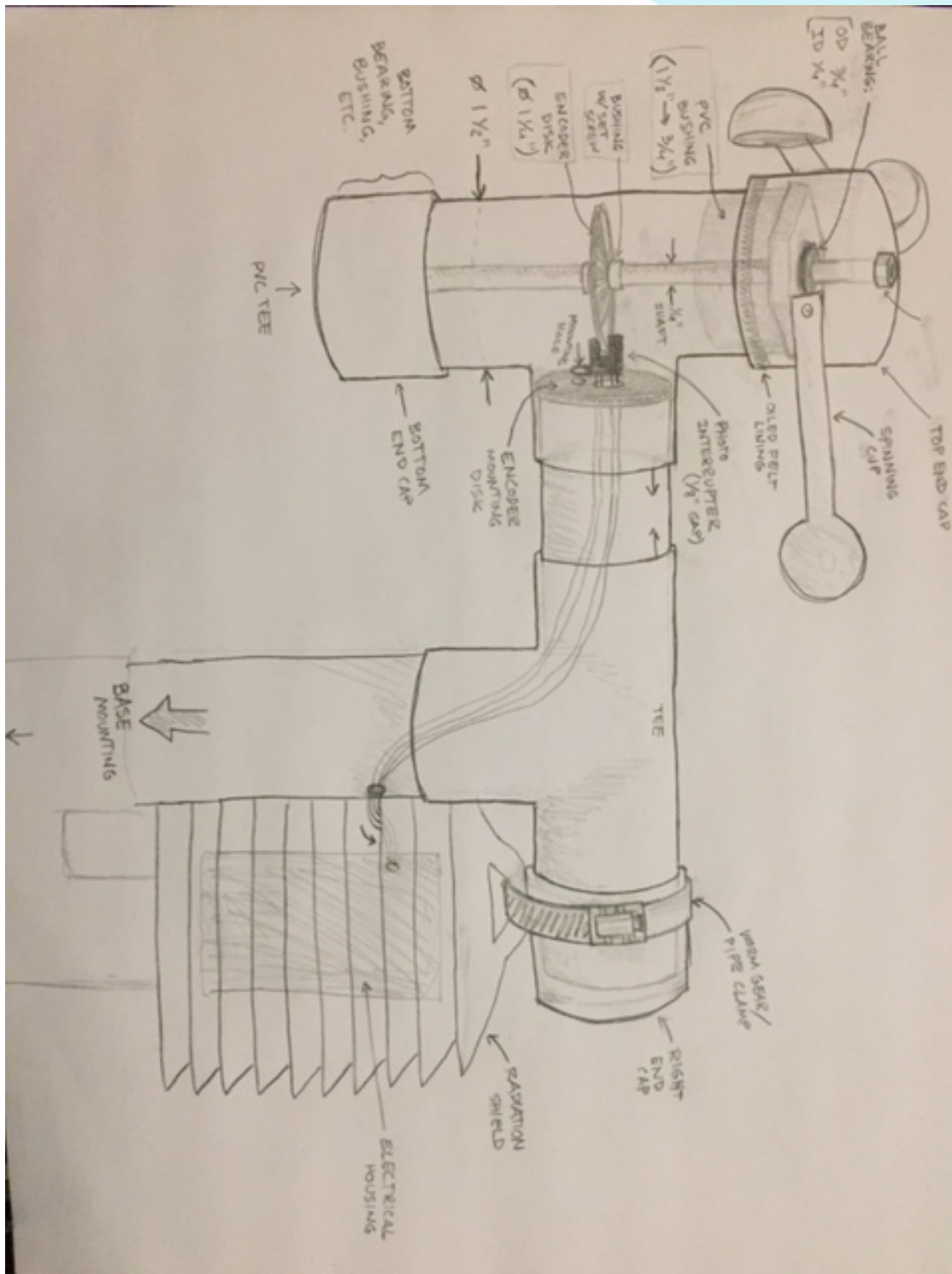
6.1 Tools

Some extra tools you'll need:

- Arduino Uno board
- Screwdriver
- Hammer
- Hand drill and drill bits
- Freezer or table clamp
- Table saw
- X-Acto knife
- Allen wrench set

6.2 Overall Design

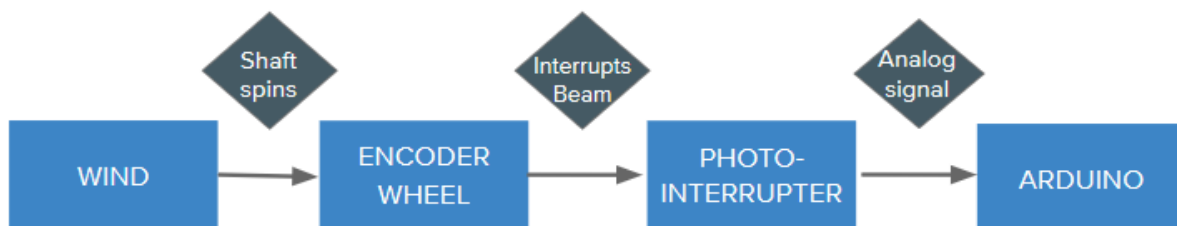
The final structure will look like this hand-drawn schematic:



To give you an idea of the final product, ours ended up looking like this:

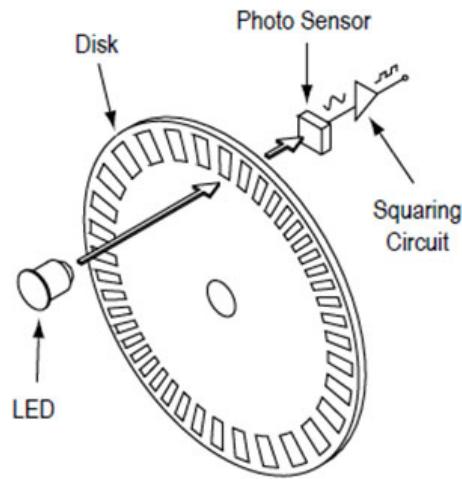


The main functions of the mechanism are as follows; the wind will catch the cups, which will spin the shaft. The shaft is attached to the encoder wheel, which will spin with it. As the wheel rotates, the photo interrupter beam will be interrupted, and will output an analog signal based on the amount of light the receiver can get. The signal will go to the Arduino, which outputs to the Serial Monitor. Your device will need to be calibrated, but we've provided the calibration information for our device, which will allow you to understand how to convert the output signal into a wind speed.



Let's get started.

6.3 Encoder



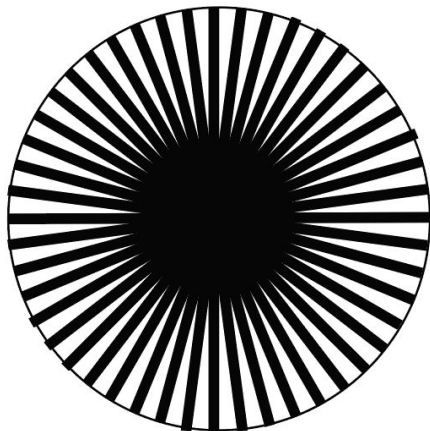
The heart of the anemometer is the encoder. An encoder is a mechanism to measure the speed at which something rotates. It usually does this by measuring the angle of a shaft relative to its initial angle, over time. An incremental encoder consists of a wheel with slits in it, and a sensor emitter and receiver on either side. Every time the sensor (called a photo interrupter) beam is blocked, a LOW signal is sent out to the Arduino. When the beam is not blocked (when it goes through a slit in the encoder wheel), a HIGH signal is sent to the Arduino. Here, we will make our own encoder to measure the speed at which a shaft rotates. The rotation is caused by the wind, similar to how wind moves

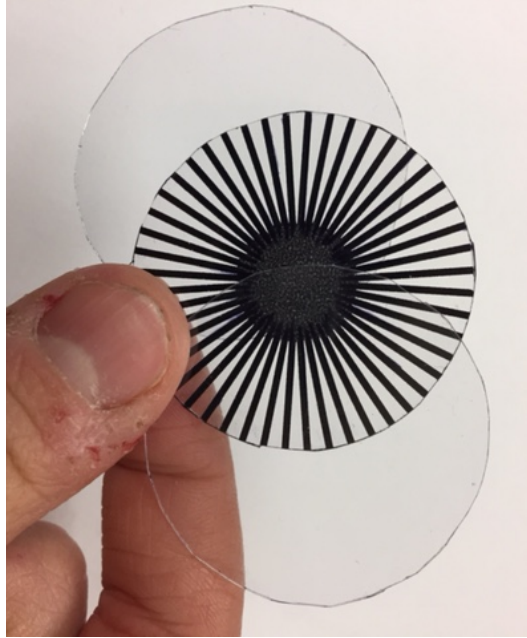
the blades of a turbine in a circle. For more information about encoders, visit this site:

<http://www.machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0>

We designed the encoder wheel with 48 black stripes, like this:

This beta Section 6 does not include an open source file with this design, but it is reproducible in any graphics program. The diameter is 1.5 inches.





Take this file to a local print shop (we went to FedEx) and get it printed on a thick, transparent projection sheet. Make sure it is the correct dimensions. Cut it out using an X-Acto knife (don't use scissors, which will bend your encoder wheel) or something similar. This sheet was a little thin, so we cut 2 pieces of lamination plastic to the same dimensions to stiffen it somewhat. Then we sandwiched the encoder wheel between these two pieces, and cut a $\frac{1}{4}$ " hole in all. We carefully super glued the edges of these together, as well as the inside edges of the hole. Now your encoder wheel is complete! Next, mount it onto the $\frac{1}{4}$ " aluminum shaft. Slide it on 4.35" from one side of the shaft, and slide two O-rings onto either side. Push them snugly up against either side of the encoder wheel, and add a drop of superglue underneath. The finished assembly should look like this:



6.4 Structure

The main body of the anemometer is made from a PVC tee. Two bushings fit into either side, with a 1 ½" PVC plug on the bottom, and a 2" PVC cap on top. First, mount the bearing inside a section of small PVC pipe. Take the small pipe (this is ¾" schedule 40 PVC pipe) and saw a 1" section off.



Make sure the table saw is positioned vertically.

Next, press fit the bearing into the small pipe piece. This works best by positioning the pieces in a table clamp and slowly closing the clamp, as shown below. Another method is to put the bearing in a freezer for about an hour, to thermally contract the metal. The bearing should then fit into the pipe with a few gentle hammer taps (ONLY tap the outside ring of the bearing, not the inside ring—or it will be damaged).

Position to tap bearing in with hammer



Position to press fit bearing with clamp



Next, cut a piece of larger 1 ½" diameter PVC pipe to a length of 1 ft. Then drill a ¼" hole into the top of the 2" PVC cap.

Assemble all components:

1. Stick the bushing/bearing assembly into the top part of a 1 ½" PVC tee.
2. Slide the shaft and encoder assembly into the bearing and bushing assembly by pushing the shaft into the bearing. Push it through until 1.7" of the 6" shaft sticks out of the top.
3. Stick a 1 ½" PVC plug onto the bottom of the tee.
4. Attach one of the small shaft clamps to the top of the shaft that sticks out of the bearing, and tighten with an allen wrench.
5. Slide the 2" PVC cap onto the aluminum shaft, so that it sits on the shaft clamp.
6. Attach the other shaft clamp onto the top of the PVC cap so that it is tight against the cap. Tighten it with an allen wrench. Make sure the cap does not rotate without the shaft.

The encoder/shaft assembly in the PVC tee looks like this:

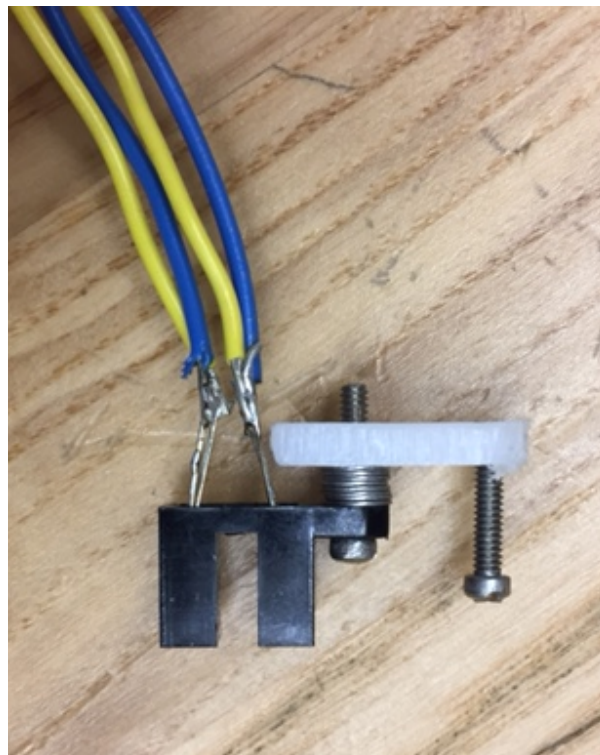


After step 4, the entire assembly should look like this:



Next, make the photo interrupter and mount assembly.

Cut out a small piece of HDPE and screw in two 3-48 holes, one at the top and one at the bottom. Mount the photo interrupter as shown:



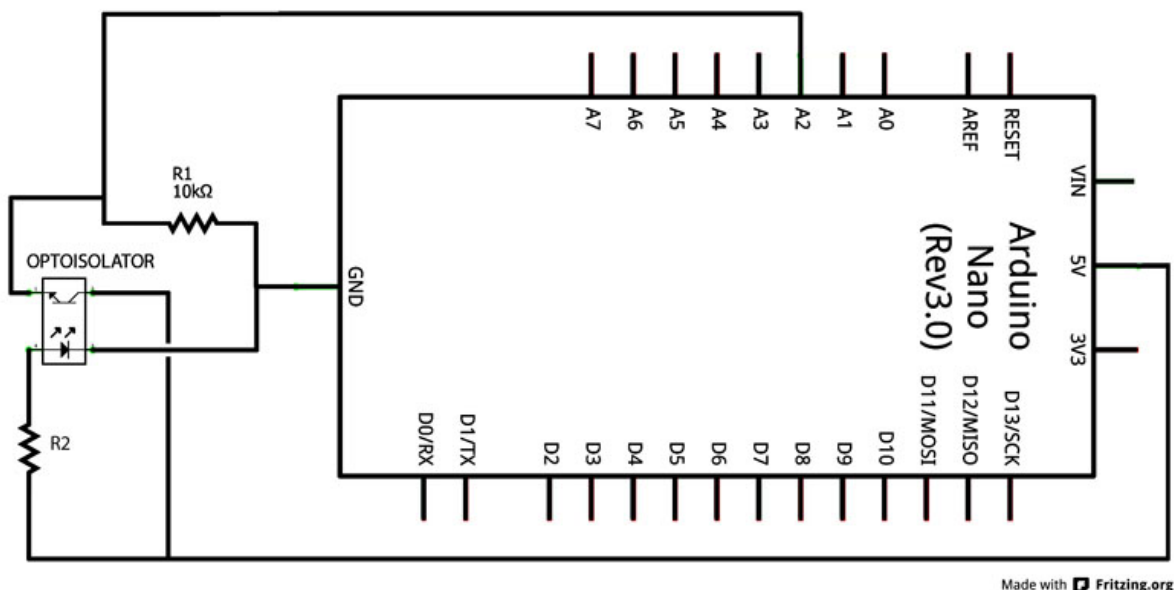
Solder 4 wires, one onto each photo interrupter pin, as shown above. Using different colors helps to identify the hardware connections. Next, drill a small hole perpendicular into the edge of the 1 ft PVC piece cut earlier. Mount the component as shown, making sure the mount is vertical:



Push the pipe piece into the perpendicular part of the tee, until it presses up against the tee ridge. This part is tricky, because the encoder wheel and photo interrupter must be aligned. Adjust the aluminum shaft up or down until the encoder wheel fits in between the emitter and receiver of the photo interrupter. When the alignment is correct, squeeze a few drops of super glue onto the shaft where the bearing is attached, making sure not to glue the bearing racings together. Your complete anemometer structure should now look like the first pictures in this section.

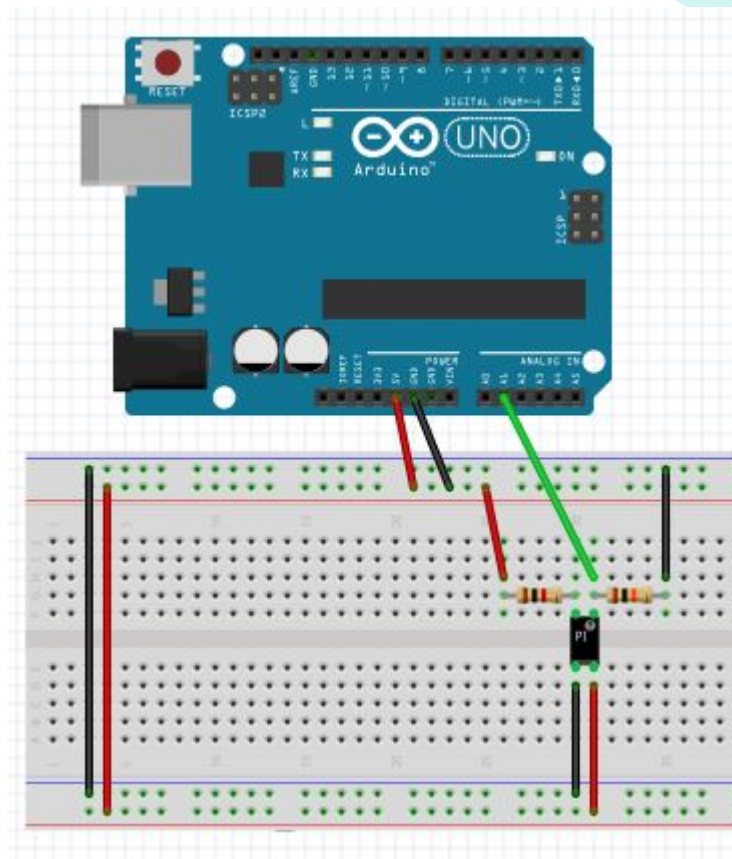
6.5 Hardware and Software

When an object passes between the emitter and detector of the photo interrupter, the beam is broken and the sensor outputs LOW. When the gap is free and the beam unbroken (the white

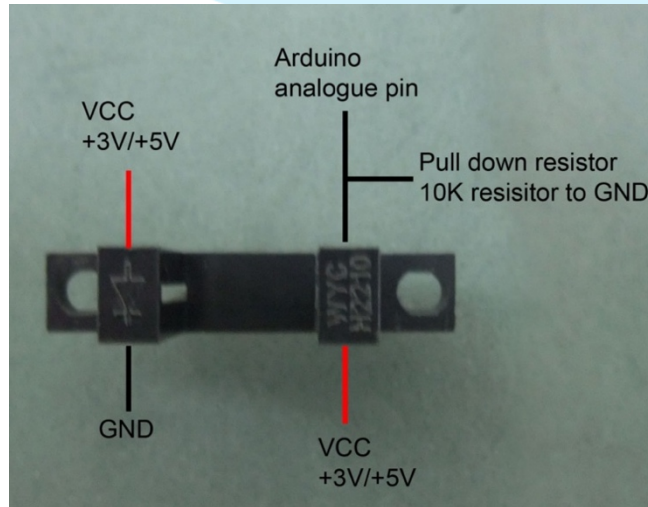


Made with  Fritzing.org

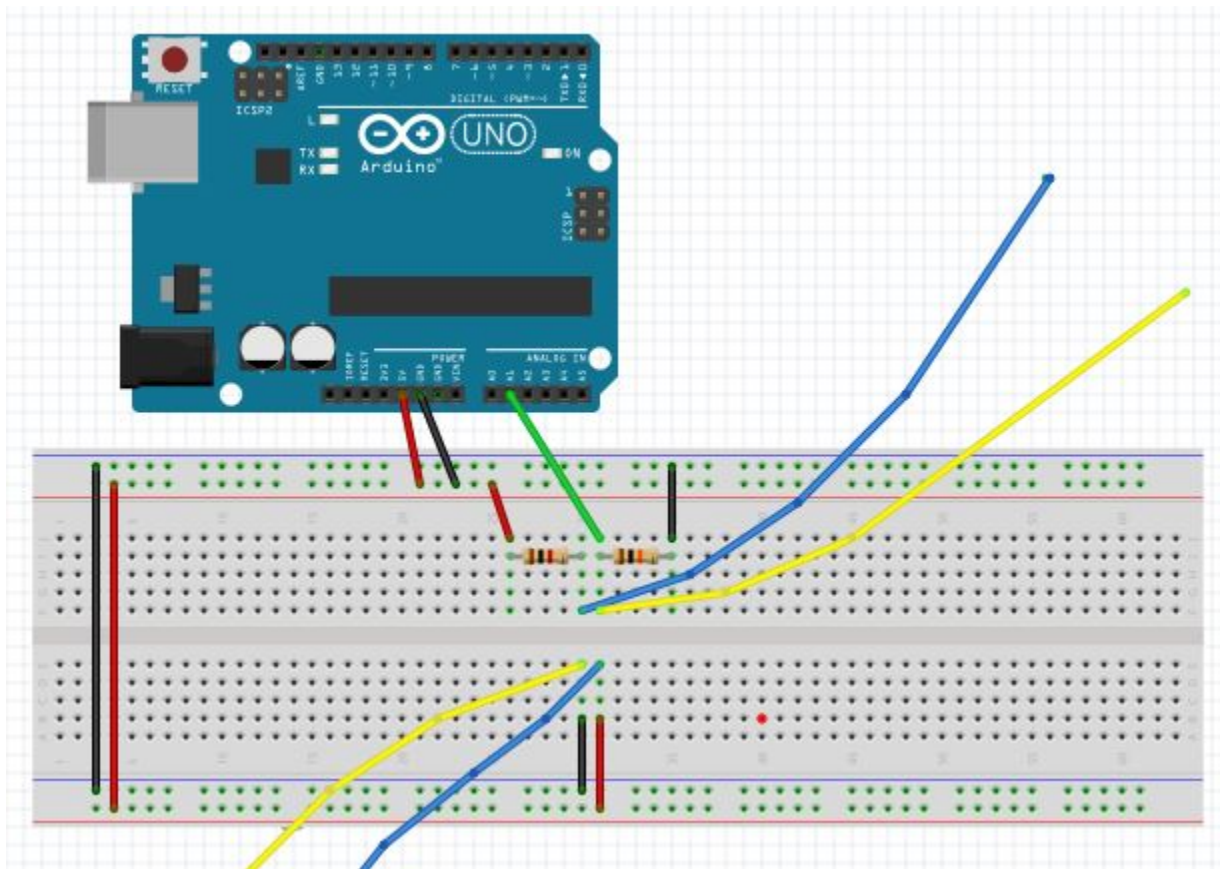
slice of the encoder wheel) the output reads HIGH. While this is an analog output, the values read from the Serial Monitor will be higher or lower. Here is how to wire your system with the Arduino: R2 is a 1K Ω resistor, and R1 is a 10K Ω resistor. For a more intuitive schematic of the setup, refer to the picture below:



As depicted above, try wiring your breadboard with a spare photo interrupter first, to test the system. Here is how the photo interrupter is connected to the rest of the system:



Once you are comfortable with this setup, replace the sensor with the wires leading out of the anemometer structure, so that it looks more like this:



Test for functionality by copy pasting the code below into the Arduino IDE. This code outputs an analog value into the Serial Monitor from the photo interrupter input.

```

/* Analog Read Interrupter
* -----
*/

int PhotoPin = A2; // select the input pin for the interrupter
int val = 0; // variable to store the value coming from the sensor

void setup()
{
  Serial.begin(115200); // set up Serial library at 115200 bps
}

void loop()
{
  val = analogRead(PhotoPin); // read the value from the sensor
  //if (val < 300){
  // Serial.println("BLOCKED");
  // }
  //else {
  Serial.println(val); // print the sensor value to the serial monitor
  delay(1);
  // }
}

```

Try spinning the shaft and see what happens to the outputs in the Serial Monitor. The homemade anemometer output, as seen when this code is implemented, is random analog numbers spit out from the photo interrupter: meaningless without calibration. We calibrated our device in a wind tunnel. If any alternative designs or structural components were integrated in your design (different shaped cups, etc), it is recommended you make your own calibration curve by blowing air from a fan at various speeds, and using a handheld anemometer to correlate wind speed with encoder output values. For reference, the calibration for our device is

$$y = 0.03x + 0.6622$$

In which y is wind speed in m/s, and x is anemometer rotation speed in rad/s.

However, this is not complete. To use this calibration, we must convert the analog output given to us in the serial monitor to a rotation speed in radians per second. This is done by finding the frequency of HIGH point output, with the following code:

```

/* Analog Read Interrupter
* -----
WITH CALIBRATION
*/

int PhotoPin = A2; // select the input pin for the interrupter
int HP = 0; //High Points in photo interrupt output
int outputs[500];

```

```

int val = 0;

void setup() {
  Serial.begin(115200); // set up Serial library at 115200 bps
}
void loop() {
  //GET DATA (0.5 sec)
  //take a 0.5 second sample of data, load data into a vector to store
  for (int i = 0; i < 500; i++){
    int val = analogRead(PhotoPin); // read the value from the sensor
    outputs[i] = val;
    delay(1);
  }
  //FIND MAX VALUES IN DATA, GET HP VALUE
  for (int j = 1; j < 500; j++) {
    //find the high points; if a point is greater than the points
    //around it and if its higher than 950, its a HP.
    if (outputs[j] > 950 && outputs[j] > outputs[j-1] && outputs[j] >
    outputs[j+1]){
      HP++;
    }
  }
  //CALCULATE AND PRINT VELOCITY
  int x = HP*0.131;
  int rads = x/0.5; //[rad/s]
  // v = sqrt(2gh), but use the calibration v = 0.03(rads) + 0.6622
  int velocity = 0.03*rads + 0.6622;
  Serial.println(val); // print the sensor value to the serial monitor
  Serial.println(velocity);
  delay(6000); //delay 1 min before taking another 10 s sample
}

```

NOTE: This code is buggy. You may need to make some alterations to make it work.

And that's it! If you got this far, you now have a working standalone anemometer. Further iterations of this manual will include separate mounting instructions, as well as hardware and software integration with the rest of the system.

Additional Resources:

Here are some additional resources we found helpful when this part of the project.

These are a few home-built anemometers similar to our own:

<http://www.hackersbench.com/Projects/anemometer/anemometer3.html>

http://www.raphnet.net/divers/anemometre/anemometer_en.php

This explains more about how to wire a photo interrupter:

<http://www.martyncurrey.com/connecting-an-photo-interrupter-to-an-arduino/>

Appendix A: Wiring Diagrams

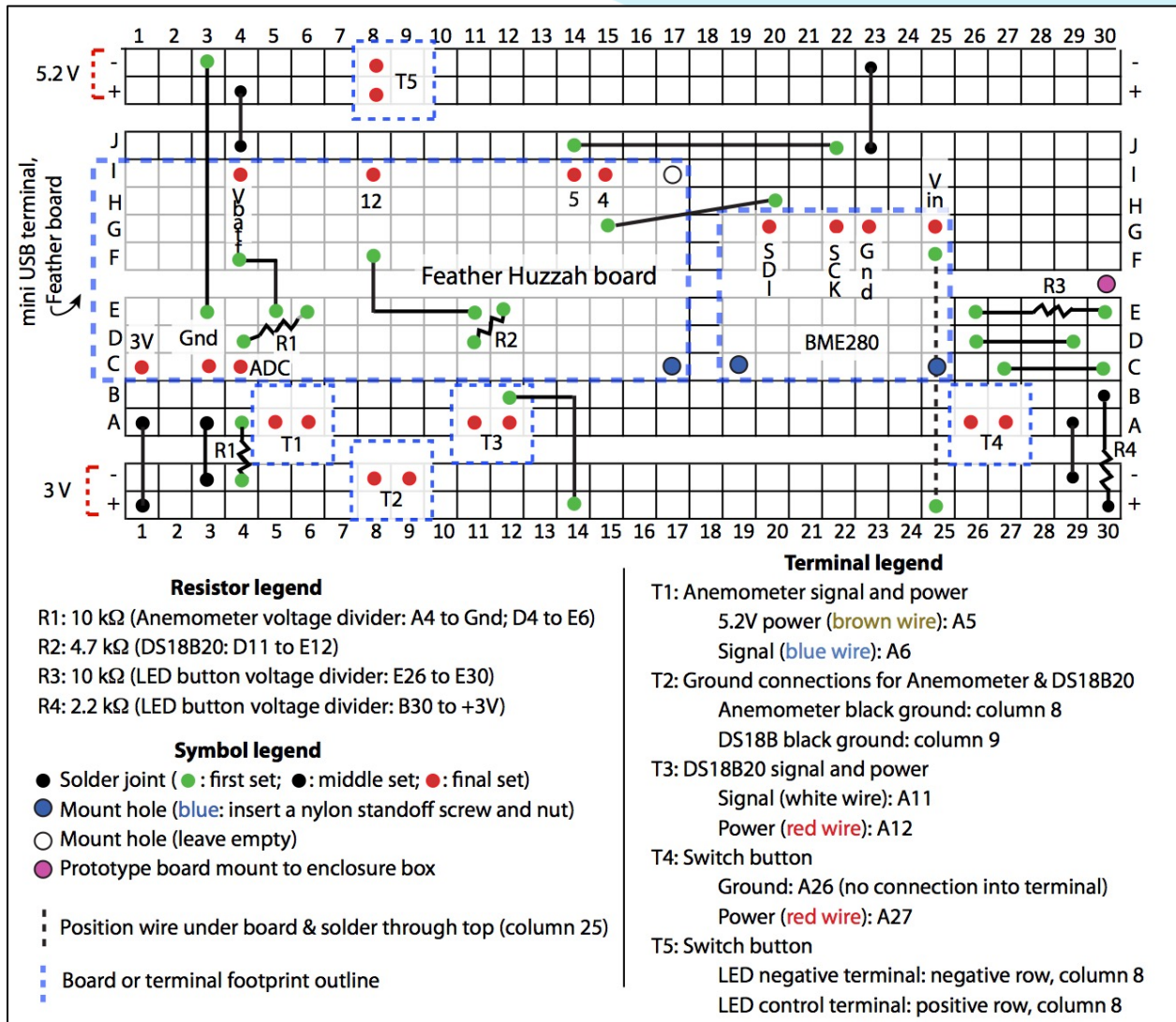


Figure A1: The wiring layout for the perma-proto board (replicate of Figure 3).

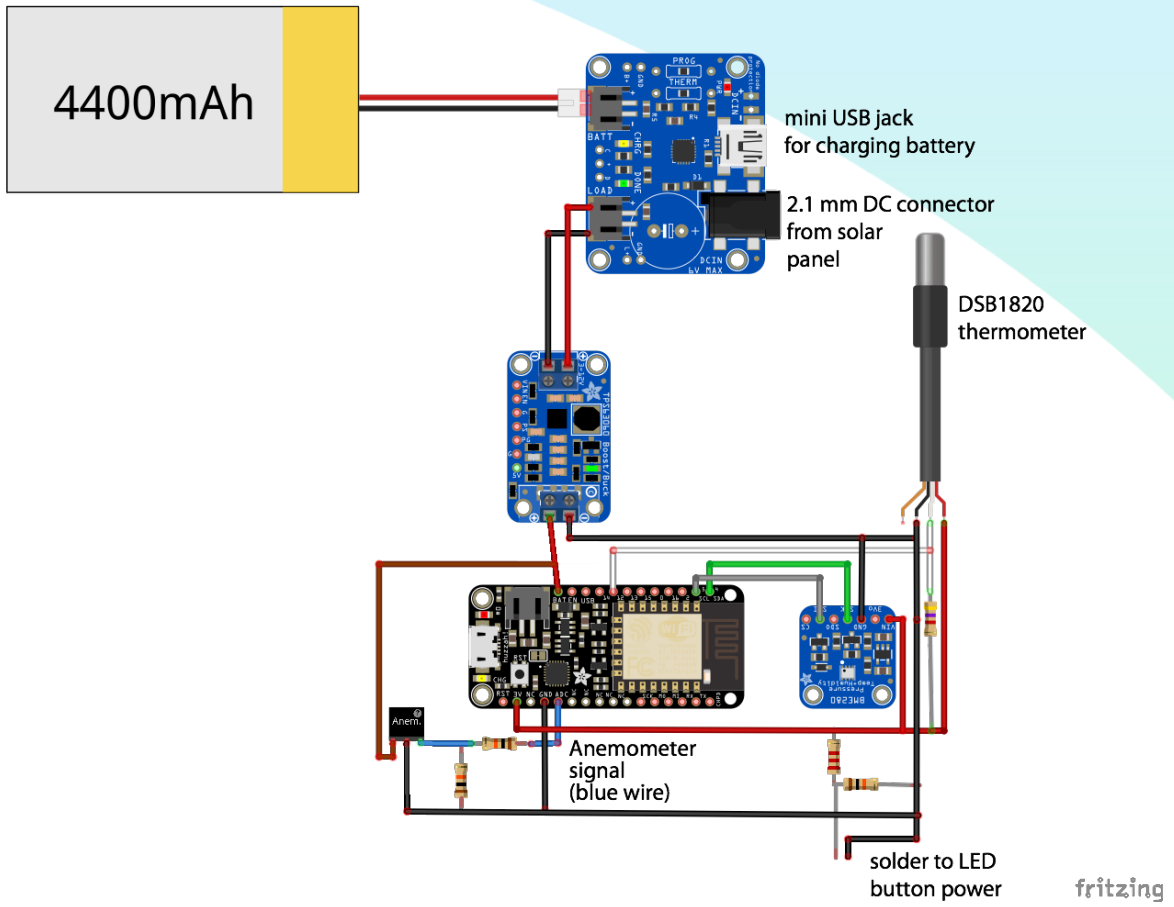


Figure A2: The connection layout for all electrical components of the weather station (replicate of Figure 4).

Appendix B: Weather Station Arduino Code

The code that you should upload to the weather station, via the Arduino IDE, is called "GLOBE_WeatherStationCode.ino". Open Arduino IDE, create a new sketch, and copy and paste the following code into the new sketch. Make sure to update everything that's highlighted with a red background, based off your Wifi network and ThingSpeak account!

```
/*
  GLOBE_WeatherStationCode.ino
  Caleb Farny, Mission Earth
  Thingspeak API Upload for wifi output from Huzzah, BME280, DS18B20 thermometer, and
  Anemometer
*/
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280_GLOBE.h>
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <ThingSpeak.h>

// I2C integration: gpio 5 & 4 = BME280 SDA & SCL
// available pins: 12, 13, 14, 16
// don't use gpio 0,2,15

#define BME_SDA 5
#define BME_SCL 4

#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI
unsigned long delayTime;

const int analogInPin = A0;
float ADCmeas = 0.0;
float Windspeed = 0;
float Windspeed2 = 0; // corrected wind speed for non-zero ADC offset
float celsius = 0;

OneWire ds(12); // on pin 12

WiFiClient client;
//
```

```

const char* ssid = "wifissid";
const char* password = "wifipassword";

//Server Information
unsigned long myChannelNumber = xxx;
const char * myWriteAPIKey = "xxx";

const char* host = "api.thingspeak.com";

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  delay(100);
  Serial.println(F("GLOBE Weather Station"));

  bool status;

  // default settings
  status = bme.begin();
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  } else
    Serial.println(F("BME280 sensor identified"));
}

delayTime = 3000;

Serial.println();

//delay(100);

// Now we connect to a WiFi network: comment out to ++ for no wifi
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

ThingSpeak.begin(client);

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

```



```

Serial.print("");
Serial.print("WiFi connected");
Serial.println();
Serial.print("IP address: ");
Serial.println();
Serial.print(WiFi.localIP());
Serial.println();

int value = 0;
delay(5000);
++value;

Serial.print("connecting to ");
Serial.println(host);
Serial.println();
// ++ wifi end
}

void loop() {
  // put your main code here, to run repeatedly:
  if (client.connect(host, 80)) {
    bool status;
    byte i;

    status = bme.begin();
    if (!status) {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      while (1);
    }
    delay(1000);

    // grab first reading
    DSTemp();
    float temp = celsius;
    float humidity = bme.readHumidity();
    float pressure = bme.readPressure();
    float wspeed = Windspeed;
    float wspeed2 = Windspeed2;

    for ( i = 1; i <100; i++) {
      DSTemp();
      temp = celsius +temp;
      humidity = bme.readHumidity()+humidity;

```

```

    pressure = bme.readPressure()+pressure;
    Wind();
    wspeed = Windspeed + wspeed;
    wspeed2 = Windspeed2 + wspeed2;
}

float tempxmt = temp / 100;
float humxmt = humidity / 100;
float pressxmt = pressure / 100 / 100; // account for avg'ing and for unit conversion
float wspeedxmt = wspeed / 100;
float wspeedxmt2 = wspeed2 / 100;

// check averaging
Serial.print("Avg Temperature = ");
Serial.print(tempxmt);
Serial.println(" *C");

Serial.print("Avg pressure = ");
Serial.print(pressxmt);
Serial.println(" hPa");

Serial.print("Avg humidity = ");
Serial.print(humxmt);
Serial.println(" %");

Serial.print("Avg wind spd = "); // wind speed without correcting for ADC offset
Serial.print(wspeedxmt);
Serial.println(" mph");

Serial.print("ADC value ");
Serial.print(ADCmeas);
Serial.println(" steps");

Serial.print("Avg wind spd, corrected = "); // corrected wind speed for non-zero ADC offset
Serial.print(wspeedxmt2); // remember: WindSpeed2 = WindSpeed if Oadc = 0 (baseline case).
Serial.println(" mph");
    delay(delayTime);

    ThingSpeak.setField(1,tempxmt);
    ThingSpeak.setField(2,humxmt);
    ThingSpeak.setField(3,pressxmt);
    ThingSpeak.setField(4,wspeedxmt2); // remember: WindSpeed2 = WindSpeed if Oadc = 0
(baseline case).

```

```

    // Write the fields that you've set all at once.
    ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

// had to subtract 2 min 15 sec to compensate for operations & avg'ing = 120 + 15 = 135000;
1800-135 = 1650 000
    Serial.print("Posting to Thingspeak");
    Serial.println();

    Serial.println("Done Posting to Thingspeak");
    // wait and then post again
    delay(1650000); // Update every 30 min; ThingSpeak will only accept updates every 15
seconds. Arduino clock: 1 = 1 ms

    // ++++++ wifi end
    }
}

//Sensor Function Outside loop
void printValues() {
    Serial.print("Temperature = ");
    Serial.print(bme.readTemperature());
    Serial.println(" *C");

    Serial.print("Pressure = ");
    Serial.print(bme.readPressure() / 100);
    Serial.println(" hPa");

    Serial.print("Humidity = ");
    Serial.print(bme.readHumidity());
    Serial.println(" %");
}

void DSTemp(){
    byte i;
    byte present = 0;
    byte type_s;
    byte data[12];
    byte addr[8];
    //float celsius, fahrenheit;
    //

    //
    if ( !ds.search(addr) ) {
        Serial.println("No more addresses.");
    }
}

```

```

Serial.println();
ds.reset_search();
delay(250);
return;
}

if (OneWire::crc8(addr, 7) != addr[7]) {
  Serial.println("CRC is not valid!");
  return;
}

  type_s = 0; // chip is DS18B20

ds.reset();
ds.select(addr);
ds.write(0x44, 1); // start conversion, with parasite power on at the end

delay(1000); // maybe 750ms is enough, maybe not
// // we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Read Scratchpad

for ( i = 0; i < 9; i++) { // we need 9 bytes
  data[i] = ds.read();
}
// Convert the data to actual temperature
// because the result is a 16 bit signed integer, it should
// be stored to an "int16_t" type, which is always 16 bits
// even when compiled on a 32 bit processor.
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  // Serial.println("9 bit res?");
  raw = raw << 3; // 9 bit resolution default
  if (data[7] == 0x10) {
    // "count remain" gives full 12 bit resolution
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
}
// it's the latter case:
else {
  // Serial.println("looks like it's else");
  byte cfg = (data[4] & 0x60);

```

```

// at lower res, the low bits are undefined, so let's zero them
if (cfg == 0x00) raw = raw & ~7; // 9 bit resolution, 93.75 ms
else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
//// default is 12 bit resolution, 750 ms conversion time
}
celsius = (float)raw / 16.0;
// fahrenheit = celsius * 1.8 + 32.0;

if ( !ds.search(addr)) {
  ds.reset_search();
  delay(250);
  return;
}
}

//Windspeed function
void Wind() {
  float Oanem = 0.4; // anemometer offset (V)
  float Kanem = 1.6 / 32.4; // anem sensitivity (V/ m/s)
  float Kvd = 0.5; // voltage divider to match ADC 1 V max input
  float Oadc = 0.0 // compensates for ADC offset (V)
  // Huzzah ADC: n = 10 bits, VFS = 1 V; --> 1024 vertical steps

  ADCmeas = analogRead(analogInPin); // number of digital steps that need to be converted to a
  voltage via Vres

  Windspeed = (ADCmeas / (Kvd * Kanem * 1024) - Oanem / Kanem) * 2.23694; // no ADC offset
  considered
  Windspeed2 = Windspeed; // for the case with no offset required.
  // compensate for ADC offset:
  Windspeed2 = (ADCmeas / 1024 - Oadc - Kvd*Oanem) / (Kvd * Kanem) * 2.23694;

  // 2.23694 mph = 1 m/s

  if (Windspeed2 < 0) {
    Windspeed2 = 0;
  }
}
}

```

Appendix C: Video library

Bundle 1: Intro, power management, buck/boost (12:13 duration) <https://youtu.be/lfMBLsC3xTA>

- Intro project & intro to board
- 2.2k resistor & therm overview: 3:13
- 2.2k resistor rear mount: 4:30
- Therm & resistor finish, capacitor: 4:58
- Final review of power management board, buck/boost overview: 7:14
- Buck/boost: 11:00

Bundle 2: Perma-proto overview & BME280 prep (11:26) <https://youtu.be/NbnqFafLHxU>

- Proto board overview
- Layout 1: 3:39
- Layout 2: 5:20
- BME280 headers, part 1: 9:41
- BME280 headers, part 2: 10:48

Bundle 3: Jumper wire, Feather Huzzah, terminal layout (11:54) <https://youtu.be/5YchgJTYrB8>

- Layout 1:
- Layout 2: 2:49
- Layout (bottom) 3: 4:33
- Feather board: 6:17
- Terminals: 10:06

Bundle 4: Mounting & assembly (10:14) <https://youtu.be/ScX1iTSYWii>

- Completed board, overview
- BME280 mount: 2:47
- BME280 – board solder: 4:37
- Feather + BME280 mounting: 5:28
- Buck/boost mounting & inter-board cable connections: 6:16
- Assembly: 9:29

Bundle 5: LED button integration (16:03) <https://youtu.be/2d3HB5x8ny8>

- Board final assembly + LED button overview
- LED wiring: 6:45
- LED wire connections to board: 13:32

Bundle 6: Sensor connections to board (13:39) <https://youtu.be/aD8hUcEXeKM>

- Sensor cable identification
- Completed sensor connections, box mounting: 3:23
- Electronics box final packaging: 5:39

Bundle 7: Tripod hardware mounting & assembly (16:12) <https://youtu.be/itRORff3Mqc>

- Solar panel mounting
- Anemometer, temp sensor, radiation shield mount: 4:31
- Electronics box waterproofing: 11:14
- Electronics box mounting: 14:08

Appendix D: STEM Exercises

High School Coding Lesson: If...else statements

In coding, as in life, we make decisions based on what the situations around us are. If your friend is going to the store and asks if you want something, you might tell them to get you a Diet Coke, but if they don't have that, get you a Mountain Dew instead. This is a real-life example of what coders call, an if...else statement. With this statement, we can have the computer output two different things depending on what the input is.

In the case of the weather station, we have to use an if...else statement in order to get an output from our anemometer. As you might have learned, when the anemometer is moved by the wind, it produces a voltage. The range of this voltage is from 0.4V-2V. At 0.4V, the wind speed is 0m/s and at 2V, the wind speed is 34m/s. We need to create an if...else statement so that any voltage reading under 0.4V gives an output of 0m/s and any other voltage will get plugged into an equation to calculate the wind speed. The equation for this calculation is:

$$(V - V_{min}) * Wind_{max} / (V_{max} - V_{min}) = \text{Wind speed}$$

Using everyday language, write out an if...else statement that would give us the wind speed.

If _____

else _____

So now we're going to make your statement into coding language. Use the following table of coding language to translate your statement:

The input voltage the anemometer gets	sensorVoltage
The voltage that equals 0m/s	voltageMin
The wind speed calculation result	windSpeed
The maximum wind speed we can record	windSpeedMax
The maximum voltage we can input	voltageMax
After writing the condition, return goes before what you tell the computer to output for that condition	return
Less than or equal to, multiply, divide, subtract, equals	<=, *, /, -, =

```
if _____:  
    return _____;  
else: _____ ( _____ ) _____ ( _____ )  
    return _____
```

Answer:

```
if sensorVoltage <= voltageMin:  
    return windSpeed = 0;  
else: windSpeed = (sensorVoltage - voltageMin)*windSpeedMax/(voltageMax  
- voltageMin)  
    return windSpeed
```

Power & Renewable Energy Calculations

What do we need to power the lights, or your cellphones, or your TV at home?

Electricity! Write what you think electricity is, and how we know how much of it we need?

Everything that uses electricity requires a certain amount of power to operate. We calculate power by using the equation: **$P=VI$** .

P is our power, **V** is the voltage used, and **I** is the current in amps (A) or (mA). (1000mA = 1A) Power is measured in watts (W). In our weather station, need to determine how much power our Huzzah board is going to use so that we can buy a battery and solar panel big enough to power it.

The Huzzah board requires 5V to operate. When the Huzzah is on but not doing anything, it uses 250mA (0.25A) every hour. Every 30 min, the Huzzah takes a measurement of the weather and when it does this, it uses more current: 500mA (0.5A) every hour. If the board is taking measurements for 4.5 min (roughly every hour) and is not doing anything for the remaining 55.5min, how much current does the board use in an hour?

$$\left(\frac{4.5}{60} \times 500mA\right) + \left(\frac{55.5}{60} \times 250mA\right) = 268.75mA/hr = 0.269A$$

Using what you solved for the current, how much power does the Pi use in an hour?

$$P = 5V \times 0.269A = 1.3W$$

Our weather station uses solar energy to produce electricity instead of the electricity from our power outlets. If it costs \$0.01 for each watt we get from our power outlets, how much money are we saving every day by using solar power? How much money per year?

$$\text{Cost/day} = \$0.01/W \times 1.3W/hr \times 24hrs = \$0.322$$

$$\text{Cost/year} = \$0.322/day \times 365days/year = \$117.7$$

In this case, our solar panel saves us almost \$118 per year, and compared to our phones, lights and TV, our weather station uses very low power. Solar energy can save households thousands of dollars a year!

Appendix D: Manufacturer Guides

Adafruit supplies two guidance documents that may be helpful to reference, one for the Huzzah breakout board and another for the solar power circuit. These are both 40-page documents, and are posted separately. Alternately they can be downloaded by visiting:

Feather Huzzah board:

<https://learn.adafruit.com/adafruit-feather-huzzah-esp8266>

Solar power circuit:

<https://cdn-learn.adafruit.com/downloads/pdf/usb-dc-and-solar-lipoly-charger.pdf>

Davis Instruments supplies a similar document that should ship with the radiation shield, but can also be downloaded by visiting:

https://www.davisnet.com/product_documents/weather/manuals/07395-093_IM_07714.pdf