Compost Friend!

Created by Isaac Wellish

https://learn.adafruit.com/compost-optimization-machine

Last updated on 2024-06-03 02:27:26 PM EDT

Table of Contents

Overview 5
- Moisture
- Temperature
- Prerequisite Guides

Materials 6
- Parts Needed:
- Optional (for prototyping):

Prototyping 9

Wiring 10

CircuitPython Code 11
- Import Libraries and Initialize Values
- The Main Program Loop: Finding the Average Soil Moisture & Temperature Levels
- Determining the State of the Compost with NeoPixels
- Compost Condition Indication with If Statements
- Using The Light Sensor to Dim or Brighten NeoPixels
- Here's the full program:
- Enough Programming, Let's Test it Out!

Testing and Calibration 19
- Test it Out!
- Calibration

Soldering 22
- First, let's solder the Solar Lithium Charger
- Next we'll solder the DS18B20 temperature sensor to the CPX.
- Now we'll solder the moisture nails to the CPX.
- Soldering the DC jack

Enclosure 33

Drilling Sensor Holes 33

Mounting Overview 35

Popsicle Stick Preparation 37
- Base Stick Preparation
- Drill Holes and Remove Excess
- Offset Stick (Legs) Preparation

Mounting Continued 42
- Mounting the Battery
- Assembling the Sticks

Waterproofing 44
- DC Jack
- Circuit Playground Express and Sensors
- Solar Panel Instillation

Attaching to Bin 49
• Done!
Lessons Learned 50

Overview

So you've been composting for a bit now and things are going ok. You've been regularly turning the compost, but you have no idea if all this work is even worth it because you don't even know what's going on in there. You try to split the dry material and wet material 50/50 but some days it's rainy and others it's dry. Is it even heating up? Do you really have to stick your hand in last weeks spaghetti leftovers to tell? What if there was a way to know all the info you want to know about your compost and how to improve its health without ever touching or looking at it?

Good news! You can use a Circuit Playground Express and some extra goodies to tell you exactly what you want to know about your compost: **when to add more food scraps, more dry material, and when to turn it.**

This is a part II to the first guide (https://adafru.it/CoP) on building a compost tumbler to more easily aerate your compost. Even if you have your own tumbler, I suggest skimming through that guide to get an understanding to why we would want to "optimize (https://adafru.it/CoQ)" compost.

As mentioned in the previous guide (to compost aerobically), we want to aerate compost every so often to feed oxygen to the microbes and other organisms within the compost so they can continue to break down the food scraps. Even though a tumbler makes it easier to turn compost, we don't want to be turning it more than we have to. Furthermore, solely turning the compost isn't always going to be the answer to making our compost healthier. Here are the primary situations when we need to adjust the compost:

- When the compost is **too dry**, **add** some **food scraps** and **turn it**.
- When the compost is **too wet**, **add** some **dry** carbon **material** like leaves, wood chips, or paper and **turn it**.
- If the compost **isn't heating up**, **turn it**.

Moisture

Typically, the compost should have about half dry and half wet material within. However you can't always tell how much of each material is present within your container. Thus we can use a moisture sensor to gauge how wet or dry the compost is.

Temperature

Due to the energy the organisms within the compost breaking down the food scraps release, the compost gets HOT. At optimal state, the compost should be at 135 - 160 degrees Fahrenheit (https://adafru.it/CoR). We will use a temperature sensor to gauge how hot the compost is and adjust accordingly.

Prerequisite Guides

If you've never worked with CircuitPython, a CPX (Circuit Playground Express), or soldered, I suggest reading through these guides before continuing.

- Adafruit Circuit Playground Express (https://adafru.it/adafruit-cpx)
- Welcome to CircuitPython! (https://adafru.it/Bid)
- Collins Lab: Soldering (https://adafru.it/wsa)

Materials

Parts Needed:

1 x Circuit Playground Express — https://www.adafruit.com/product/3333
Circuit Playground Express

---

1 x Circuit Playground Bolt-On Kit — https://www.adafruit.com/product/3816
Circuit Playground Bolt-On Kit

---

1 x Black Nylon Screw and Stand-off Set – M2.5 Thread — https://www.adafruit.com/product/3299
Black Nylon Screw and Stand-off Set – M2.5 Thread

---

1 x Sugru - Black and White Pack — https://www.adafruit.com/product/437
Sugru - Black and White Pack

---

1 x USB / DC / Solar Lithium Ion/Polymer charger - v2 — https://www.adafruit.com/product/390
USB / DC / Solar Lithium Ion/Polymer charger - v2

Medium 6V 2W Solar panel - 2.0 Watt

1 x Medium 6V 2W Solar panel - 2.0 Watt — https://www.adafruit.com/product/200

Medium 6V 2W Solar panel - 2.0 Watt

---

1 x 3.5 / 1.3mm or 3.8 / 1.1mm to 5.5 / 2.1mm DC Jack Adapter Cable — https://www.adafruit.com/product/2788

3.5 / 1.3mm or 3.8 / 1.1mm to 5.5 / 2.1mm DC Jack Adapter Cable

---

1 x Small Plastic Project Enclosure - Weatherproof with Clear Top — https://www.adafruit.com/product/903

Small Plastic Project Enclosure - Weatherproof with Clear Top

---

1 x Waterproof DS18B20 Digital temperature sensor + extras — https://www.adafruit.com/product/381

Waterproof DS18B20 Digital temperature sensor + extras

---

2 x Cable Gland PG-9 size - 0.158" to 0.252" Cable Diameter - PG-9 — https://www.adafruit.com/product/761

Cable Gland PG-9 size - 0.158" to 0.252" Cable Diameter - PG-9

---

1 x Waterproof DC Power Cable Set - 5.5/2.1mm — https://www.adafruit.com/product/743

Waterproof DC Power Cable Set - 5.5/2.1mm

---

1 x Drill — https://www.homedepot.com/p/DEWALT-20-Volt-MAX-XR-Lithium-Ion-Cordless-1-2-in-Compact-Brushless-Drill-Driver-Tool-Only-DCD791B/206523957

To drill holes for enclosure and mounting

---

1 x 1/8" Drill Bit

Drilling holes for M4 Screws (enclosure mounting)

---

1 x 3/32" Drill bit

Drilling holes for M3 Screws (enclosure mounting)

---

1 x #46 Drill Bit (0.081")

Drilling holes for M2.5 Screws (enclosure mounting)

---

1 x Ruler

To measure and mark parts

For mounting components in enclosure

1 x Pack of Popsicle Sticks — https://www.target.com/p/craft-sticks-150ct-natural-wood-hand-made-modern/-/A-53085449?ref=tgt_adv_XS000000&AFID=google_pla_df&CPN
qn3F53UBUPvBoC1xAQAvD_BwE&gclsrc=aw.ds

For mounting components in enclosure

---

1 x Phillips Head Screw Driver size #0 — https://www.adafruit.com/product/424

For M4 screws

---

1 x Phillips Head Screw Driver size #00 — https://www.adafruit.com/product/424

for M3 and M2.5 screws

---

2 x Nails

For moisture sensing though capacitive touch. Any old nail will do

---

1 x Heat Shrink Pack — https://www.adafruit.com/product/344

Heat Shrink Pack

---

1 x Soldering Iron

Soldering Iron

---

1 x Solder Wire — https://www.adafruit.com/product/1886

Solder Wire

---

1 x Lighter or matches

For shrinking heat shrink

---

1 x 22 - 28 Gauge Stranded Wire — https://www.adafruit.com/product/1877

22 - 28 Gauge Stranded Wire

---

1 x Table clamp

For drilling mounting holes in popsicle sticks

---

1 x Compost Bin

A bin to store compost in and install enclosure and panel on.

Optional (for prototyping):

1 x Small Alligator Clip Test Lead (set of 12) — https://www.adafruit.com/product/1008

Small Alligator Clip Test Lead (set of 12)

Half-size breadboard

1 x Half-size breadboard — https://www.adafruit.com/product/64

Half-size breadboard

---

1 x Breadboarding wire bundle — https://www.adafruit.com/product/153

Breadboarding jumper wires

---

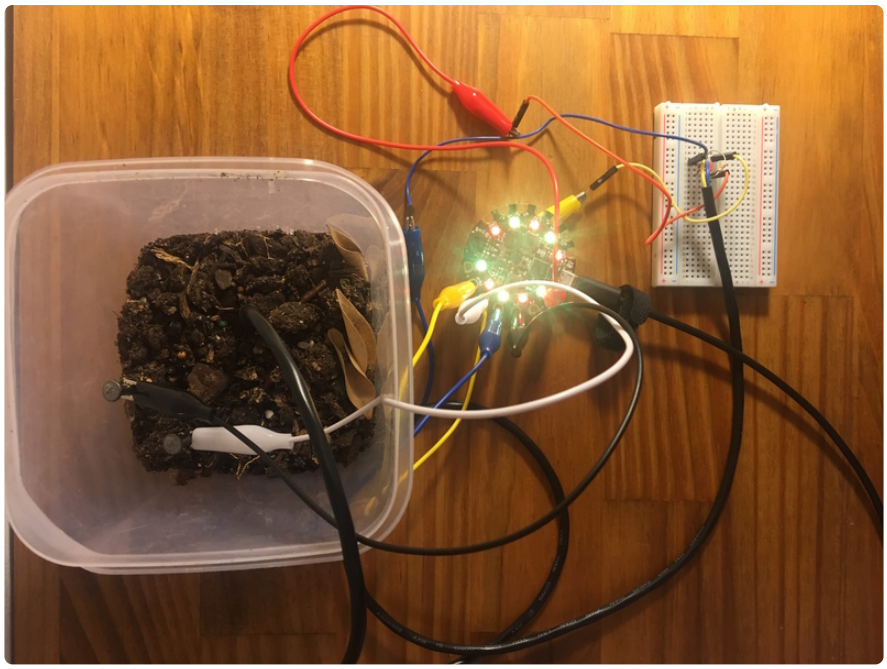
1 x A small container with some soil or dirt

For testing sensors

Prototyping

Before we build this thing, we want to test it out to make sure everything is working properly. We'll use:

- A container and some soil or dirt to act as the compost so we can measure and calibrate temperature and moisture levels
- A CPX to hook up our sensors and send the readings to a computer
- A bread board to wire the 4.7K Ohm resistor that is needed to use the temperature sensor
- The waterproof DS18B20 digital temperature sensor
- Two nails for moisture sensing
- Alligator clips
- Breadboard wires
- A micro usb cable
- A computer
- Mu editor for writing code (https://adafru.it/Bid) and receiving sensor values

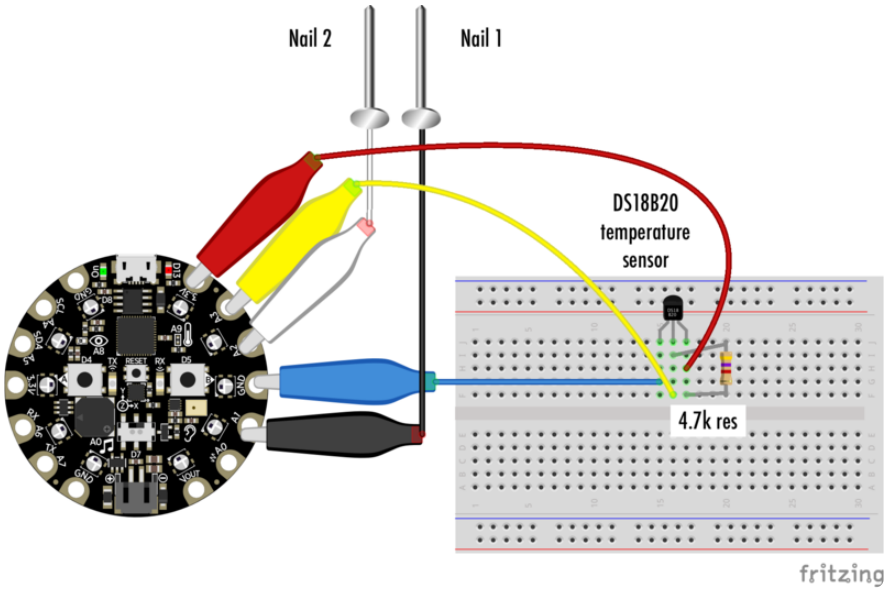


Wiring

First let's wire up the DS18B20 temperature sensor. We need to add the 4.7 KΩ pull-up resistor that comes with the sensor to the signal line to ensure the board can read the sensor. We'll use a small breadboard, some breadboard jumpers and alligator clips to wire it up.

- **Blue wire of the sensor** to **CPX ground**.
- **Yellow wire of the sensor** to **CPX A3**.
- **Red wire of the sensor** to **CPX 3.3V.**
- **4.7 KΩ resistor** connected to both the **yellow wire** (data) and **red wire** (3.3V).

For the moisture sensors we'll use **two nails** and **two alligator clips** connected to **A1** and **A2** on the CPX.



I had some help from the awesome "Using DS18B20 Temperature Sensor with CircuitPython (https://adafru.it/CoS)" guide for the wiring and code. Feel free to refer to that guide for extended info on the sensor.

CircuitPython Code

Are you new to using CircuitPython? No worries, there is a full getting started guide here. (https://adafru.it/CoT)

To edit the CircuitPython code and receive realtime data from our sensors in the REPL, Adafruit suggests using the Mu Editor. You can learn about Mu and installation in this tutorial. (https://adafru.it/BHi)

Open up the Mu editor or an editor of your choice with a REPL. (REPL = read-evaluate-print-loop, and it's what we'll need to use to access the values of our moisture and temperature levels so we can calibrate our sensors).

Import Libraries and Initialize Values

We'll begin the program by importing the necessary libraries and initialize the variables we'll be using.

If you think you might be missing a library, you'll be covered by installing the whole library package. Find out how in the CircuitPython Essentials Guide on CircuitPlayground Libraries. (https://adafru.it/ABU)

- **DRY_VALUE** and **WET_VALUE** should be calibrated and will be slightly different for everyone. We'll talk about how to calibrate this later.

- **tempThreshhold** is in Celcius and is the **minimum** temperature needed for ideal compost conditions. Feel free to change this to room temperature when testing the prototype to make sure everything works. Don't forget to change back to 43!

```
# Author: Isaac Wellish
# Code adapted from Tony Dicola's CircuitPython code on using the DS18x20
temperature sensor-
# as well as John Park's CircuitPython code on determining soil moisture from nails

from adafruit_onewire.bus import OneWireBus
from adafruit_ds18x20 import DS18X20
import time
import board
import simpleio
import touchio
import neopixel
import analogio
from simpleio import map_range


#Initialize neopixels
pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=.1)

#set variables for capacitive touch inputs, later used for soil moisture variables
touch = touchio.TouchIn(board.A1)
touch2 = touchio.TouchIn(board.A2)

DRY_VALUE = 3100  # calibrate this by hand!
WET_VALUE = 4000  # calibrate this by hand!
tempThreshhold = 43 #celius temperature of threshold for ideal compost temperature


# Initialize one-wire bus on board pin A3.
ow_bus = OneWireBus(board.A3)

# Scan for sensors and grab the first one found.
ds18 = DS18X20(ow_bus, ow_bus.scan()[0])

#Initialize the light senor on board to use for neopixel brightness later
light = analogio.AnalogIn(board.LIGHT)
```

The Main Program Loop: Finding the Average Soil Moisture & Temperature Levels

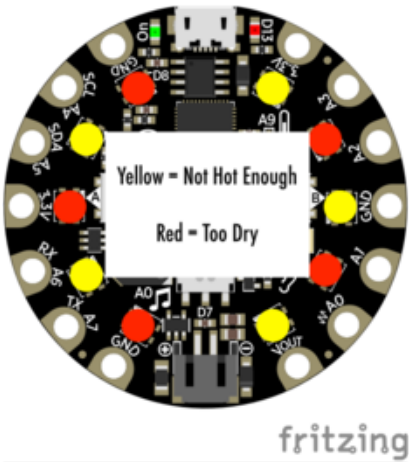
- To have a more accurate representation of what the moisture level is in the compost, we place the nails in separate locations in the compost and take the average of both moisture readings.
- Print average moisture and temperature levels.

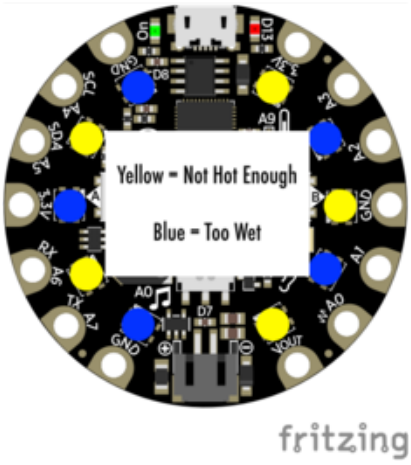
```
# Main loop
while True:

    ###SOIL MOISTURE READINGS

    #set variables for capacitive touch inputs for nails to take in soil moisture
levels
    value_A1 = touch.raw_value
    value_A2 = touch2.raw_value
```

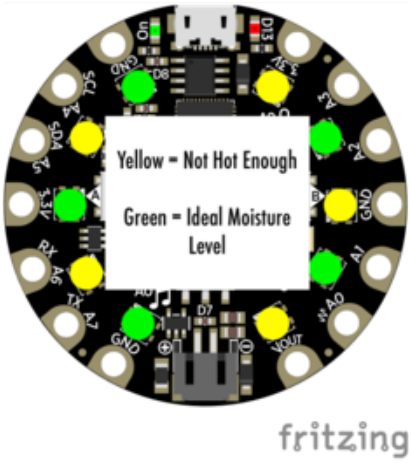
```
#take the average of both moisture levels
avgMoist = value_A1 + value_A2 / 2
print("Moisture level:",(avgMoist,))



###TEMPERATURE READINGS

#variable for temperature
compostTemp = ds18.temperature

#print the temperature
print('Temperature: {0:0.3f}C'.format(compostTemp))
```

Determining the State of the Compost with NeoPixels

- We will be using the on board NeoPixels as visual feedback for the information the temperature and moisture sensors are giving us. The colors let us know how to best help the compost.
- Ideally we would throw in a wifi module and make the project Internet of Things friendly to be able to see the real sensor data without having to look at a box on the compost bin but I wanted to keep this project a little more simple. I will most likely be adding IoT functionality to this project later!





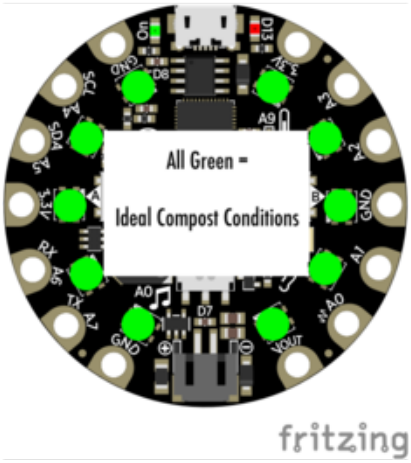


When the compost is **not hot enough and too dry**, the colors **yellow and red** will be displayed alternating. This color combination means you must **add food scraps to counterbalance the dryness**. You must **also turn the compost** to mix in the new food scraps as well as give the organisms inside some oxygen so they can better breakdown the compost, releasing heat and increasing the temperature.

When the compost is **not hot enough and too wet**, the colors **yellow and blue** will be displayed alternating. This color combination means you must add dry carbon-based material like leaves and wood chips to counterbalance the wet stuff. You must **turn the compost** as well for the same reasons as above.

When the compost is **not hot enough but at the right moisture level** (so close!), the colors **yellow and green** will be displayed alternating. This color combination means the moisture is at the ideal level however the compost isn't hot enough. **Give the compost a couple turns** to bring some oxygen to those hungry organisms in there!

When the compost is at the ideal temperature level, regardless of moisture, **all the NeoPixels will be green.** We have ideal conditions! This means **you don't have to do anything!**



Compost Condition Indication with If Statements

```
###IF STATEMENTS TO DETERMINE STATE OF COMPOST


    #RED &amp; YELLOW = TOO COLD &amp; TOO DRY
    if((compostTemp&lt;tempThreshhold) and (avgMoist&lt;DRY_VALUE)):
        pixels[0] = (255,0,0) # red
        pixels[1] = (255,255,0) #yellow
        pixels[2] = (255,0,0)
        pixels[3] = (255,255,0)
        pixels[4] = (255,0,0)
        pixels[5] = (255,255,0)
        pixels[6] = (255,0,0)
        pixels[7] = (255,255,0)
        pixels[8] = (255,0,0)
        pixels[9] = (255,255,0)

        print("Not hot enough, too dry")

    #BLUE &amp; YELLOW = TOO COLD &amp; TOO WET
    elif((compostTemp&lt;tempThreshhold) and (avgMoist&gt;WET_VALUE)):
        pixels[0] = (0,0,255) # blue
        pixels[1] = (255,255,0) #yellow
        pixels[2] = (0,0,255)
        pixels[3] = (255,255,0)
        pixels[4] = (0,0,255)
        pixels[5] = (255,255,0)
        pixels[6] = (0,0,255)
        pixels[7] = (255,255,0)
        pixels[8] = (0,0,255)
        pixels[9] = (255,255,0)
        print("Not hot enough, too wet")

    #GREEN &amp; YELLOW = TOO COLD &amp; MOISTURE LEVEL OPTIMUM
    elif((compostTemp&lt;tempThreshhold) and (avgMoist &gt;DRY_VALUE and
avgMoist&lt;WET_VALUE)):
        pixels[0] = (0,255,0) # green
        pixels[1] = (255,255,0) #yellow
        pixels[2] = (0,255,0)
        pixels[3] = (255,255,0)
        pixels[4] = (0,255,0)
        pixels[5] = (255,255,0)
        pixels[6] = (0,255,0)
        pixels[7] = (255,255,0)
        pixels[8] = (0,255,0)
        pixels[9] = (255,255,0)
```

```
        print("Not hot enough, right moisture level")

    #ALL GREEN = COMPOST AT OPTIMUM TEMPERATURE &amp;amp; MOISTURE
    elif(compostTemp&amp;gt;tempThreshhold):
        pixels.fill((0,255,0))# green
        print("Compost Ready")
```

Using The Light Sensor to Dim or Brighten NeoPixels

- To **decrease the amount of power needed** for running this program on the CPX (Circuit Playground Express) and make it easier on the eyes to see the NeoPixel colors, we will use the on board light sensor to **brighten the NeoPixels when it's brighter out and dim the NeoPixels when it's darker out**.
- We'll use a mapping function to map the range of the light sensor to the range of NeoPixel brightness.
- We then use that mapped value as the level of brightness for the NeoPixels to show on the board.
- Lastly we pause the program for three seconds to save power. We don't need to be updating the program and sensor data any more than this. If you'd like to save power even more, feel free to change this amount of time!

*It would make sense to show the NeoPixels ("pixels.show()") at the end of the program but for some reason the NeoPixels would only update correctly if the show() function was executed **before** updating the brightness level. This works fine with a three second pause in between while loop runs but will become more and more faulty once the update time is greatly increased. The brightness should update well enough as long as the update time is less than a minute or so. Please let us know if you have figured out a better fix!

```
###LIGHTING CONFIGURATION

    #print value of light sensor
    print((light.value,))

    #map light snesor range to neopixel brightness range
    peak = map_range(light.value, 2000, 62000, 0.01, 0.3)


    #print neopixel brightness levels
    print(peak)

    #show neopixels
    pixels.show()

    #update neopixel brightness based on level of exposed light
    pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=peak)

    #pause for three seconds
    time.sleep(3)
```

```
###END PROGRAM
```

Here's the full program:

```
# SPDX-FileCopyrightText: 2018 Isaac Wellish for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Author: Isaac Wellish
# Code adapted from Tony Dicola's CircuitPython code using the DS18x20 temp sensor
# as well as John Park's CircuitPython code determining soil moisture from nails

import time
from adafruit_onewire.bus import OneWireBus
from adafruit_ds18x20 import DS18X20
import board
import touchio
import neopixel
import analogio
from simpleio import map_range

# Initialize neopixels
pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=.1)

# set variables for capacitive touch inputs, later used for soil moisture variables
touch = touchio.TouchIn(board.A1)
touch2 = touchio.TouchIn(board.A2)

DRY_VALUE = 3100  # calibrate this by hand!
WET_VALUE = 4000  # calibrate this by hand!
tempThreshhold = 43 #celius temperature of threshold for ideal compost temperature

# Initialize one-wire bus on board pin A3.
ow_bus = OneWireBus(board.A3)

# Scan for sensors and grab the first one found.
ds18 = DS18X20(ow_bus, ow_bus.scan()[0])

# Initialize the light senor on board to use for neopixel brightness later
light = analogio.AnalogIn(board.LIGHT)

# Main loop
while True:

    # SOIL MOISTURE READINGS

    # set capacitive touch inputs for nails to take in soil moisture levels
    value_A1 = touch.raw_value
    value_A2 = touch2.raw_value

    # take the average of both moisture levels
    avgMoist = value_A1 + value_A2 / 2
    print("Moisture level:",(avgMoist,))

    # TEMPERATURE READINGS

    # variable for temperature
    compostTemp = ds18.temperature

    # print the temperature
    print('Temperature: {0:0.3f}C'.format(compostTemp))

    # IF STATEMENTS TO DETERMINE STATE OF COMPOST
```

```
    # RED & YELLOW = TOO COLD & TOO DRY
    if((compostTemp<tempThreshhold) and (avgMoist<DRY_VALUE)):
        pixels[0] = (255,0,0)    # red
        pixels[1] = (255,255,0)  # yellow
        pixels[2] = (255,0,0)
        pixels[3] = (255,255,0)
        pixels[4] = (255,0,0)
        pixels[5] = (255,255,0)
        pixels[6] = (255,0,0)
        pixels[7] = (255,255,0)
        pixels[8] = (255,0,0)
        pixels[9] = (255,255,0)

        print("Not hot enough, too dry")

    # BLUE & YELLOW = TOO COLD & TOO WET
    elif((compostTemp<tempThreshhold) and (avgMoist>WET_VALUE)):
        pixels[0] = (0,0,255)    # blue
        pixels[1] = (255,255,0)  # yellow
        pixels[2] = (0,0,255)
        pixels[3] = (255,255,0)
        pixels[4] = (0,0,255)
        pixels[5] = (255,255,0)
        pixels[6] = (0,0,255)
        pixels[7] = (255,255,0)
        pixels[8] = (0,0,255)
        pixels[9] = (255,255,0)
        print("Not hot enough, too wet")

    # GREEN & YELLOW = TOO COLD & MOISTURE LEVEL OPTIMUM
    elif((compostTemp<tempThreshhold) and (avgMoist >DRY_VALUE and
avgMoist<WET_VALUE)):
        pixels[0] = (0,255,0)    # green
        pixels[1] = (255,255,0)  # yellow
        pixels[2] = (0,255,0)
        pixels[3] = (255,255,0)
        pixels[4] = (0,255,0)
        pixels[5] = (255,255,0)
        pixels[6] = (0,255,0)
        pixels[7] = (255,255,0)
        pixels[8] = (0,255,0)
        pixels[9] = (255,255,0)
        print("Not hot enough, right moisture level")

    # ALL GREEN = COMPOST AT OPTIMUM TEMPERATURE & MOISTURE
    elif compostTemp > tempThreshhold:
        pixels.fill((0,255,0))  # green
        print("Compost Ready")

    # LIGHTING CONFIGURATION

    # print value of light sensor
    print((light.value,))

    # map light snesor range to neopixel brightness range
    peak = map_range(light.value, 2000, 62000, 0.01, 0.3)

    # print neopixel brightness levels
    print(peak)

    # show neopixels
    pixels.show()

    # update neopixel brightness based on level of exposed light
    pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=peak)

    # pause for three seconds
    time.sleep(3)
```

```
# END PROGRAM
```

Enough Programming, Let's Test it Out!

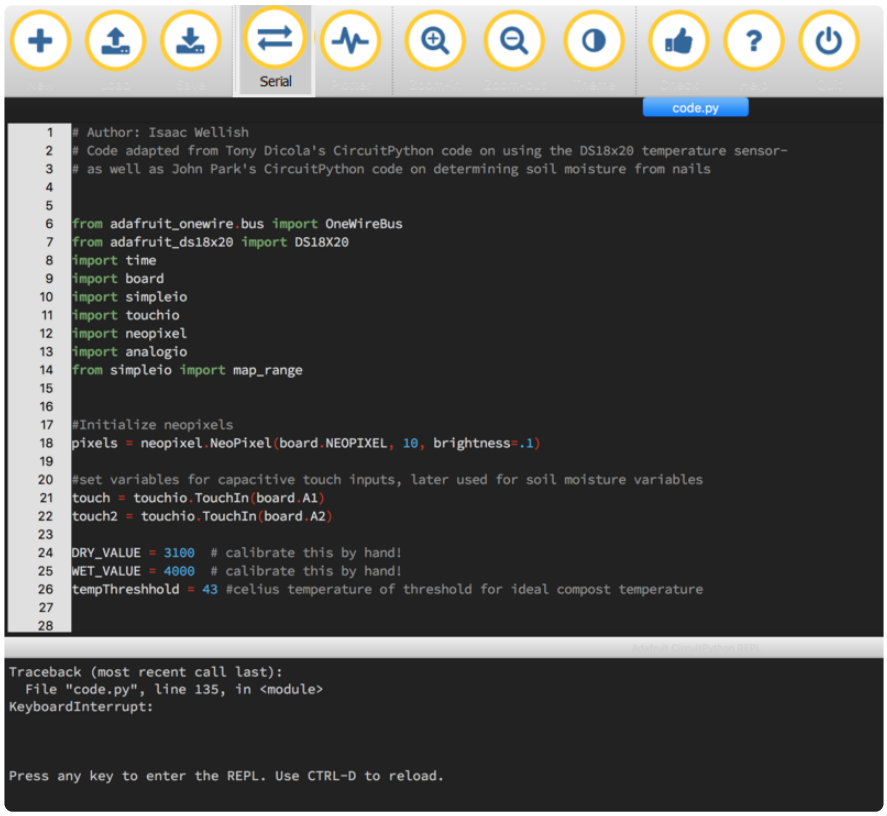
Testing and Calibration

Now that we have everything hooked up and the code ready to go, let's test it out.

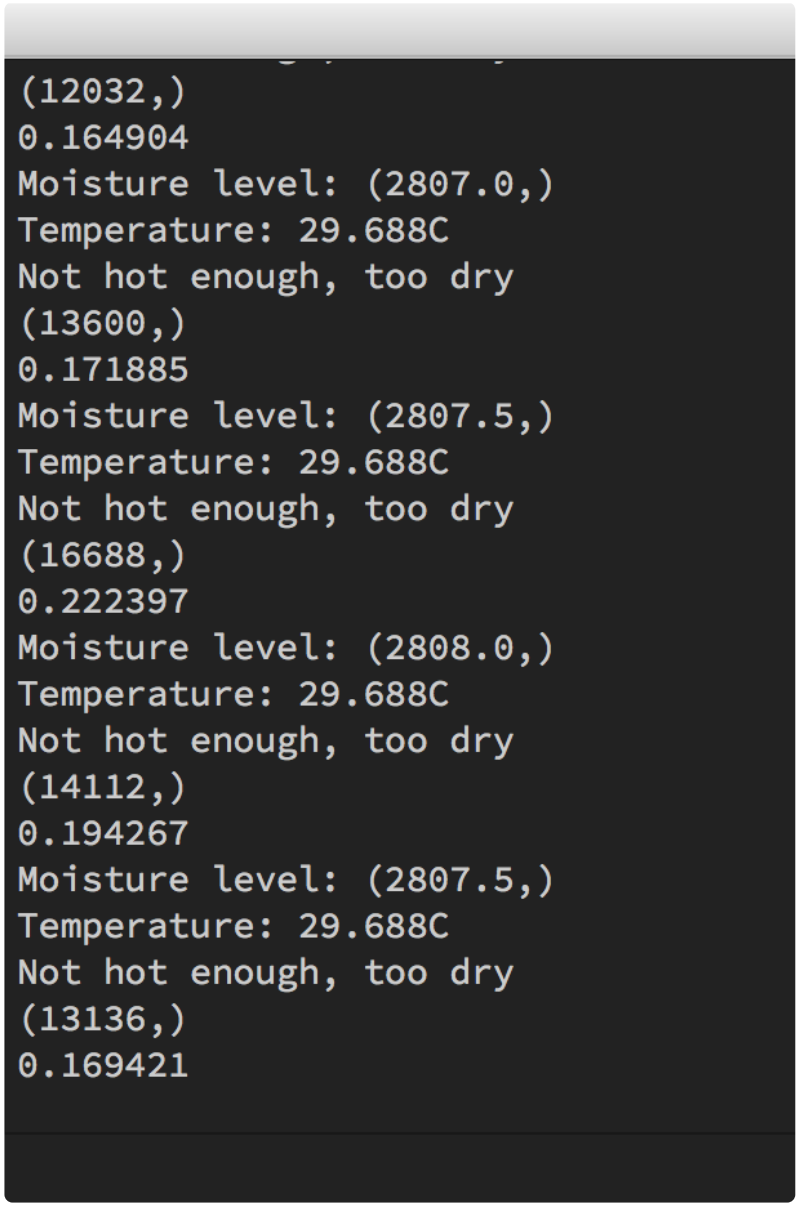
- Save the CircuitPython file as "code.py" on your computer.
- Plug the Circuit Playground Express into your computer.
- Drag the "code.py" file onto your CIRCUITPY drive that should show up in your operating system file explorer/finder program.

Test it Out!

- Open up "code.py" from the Circuit Playground Express on Mu
- Click the "Serial" button on the top tool bar in Mu 1.0.0 or greater
- Hit CTRL + D on your keyboard to re-load the code and start receiving sensor values.



- You should start to see the sensor values being printed in the REPL.
- Do these values make sense? Are the lower or higher than what you want?
- It was a hot day for me so the temperature being at 29.7 C made sense.
- Try wrapping your hand in the temperature sensor and watch the values increase.
- Try touching the nails and see what happens to the moisture values.



Calibration

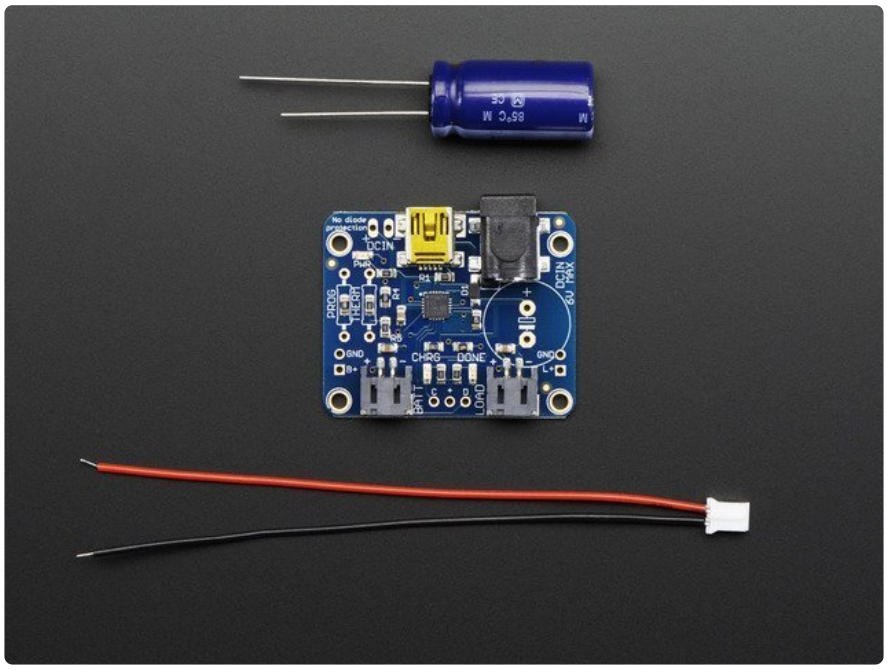
Use the cup of dirt from before with some water in it to calibrate your moisture sensors (aka nails). Put enough water in the soil so that the soil is at a consistency of a damp sponge. Try changing the "dry" and "wet" moisture values at the beginning of the program so that the NeoPixels turn green when the nails are in the ideal moisture environment you have created.

```
DRY_VALUE = 3100  # calibrate this by hand!
WET_VALUE = 4000  # calibrate this by hand!
tempThreshhold = 43 #celius temperature of threshold for ideal compost temperature
```

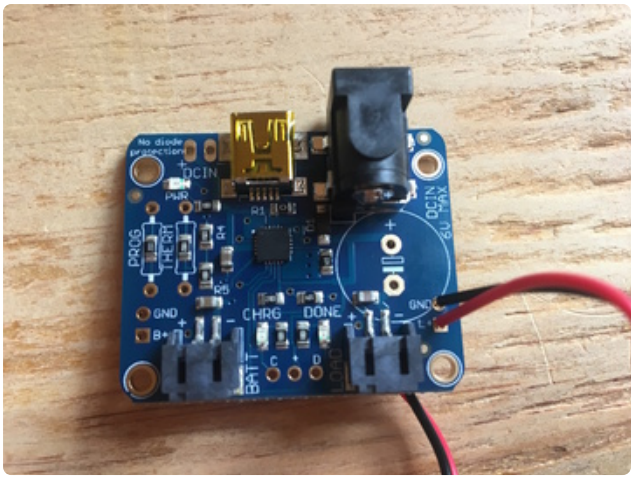
Once the sensors are calibrated, the wiring and code are good to go. It's time to start the build!

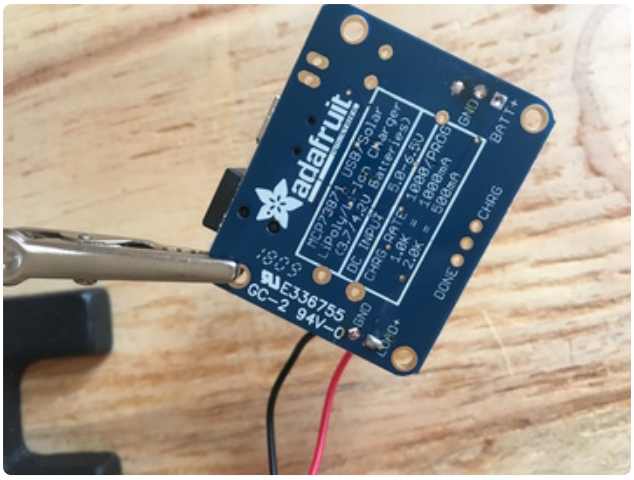
Soldering

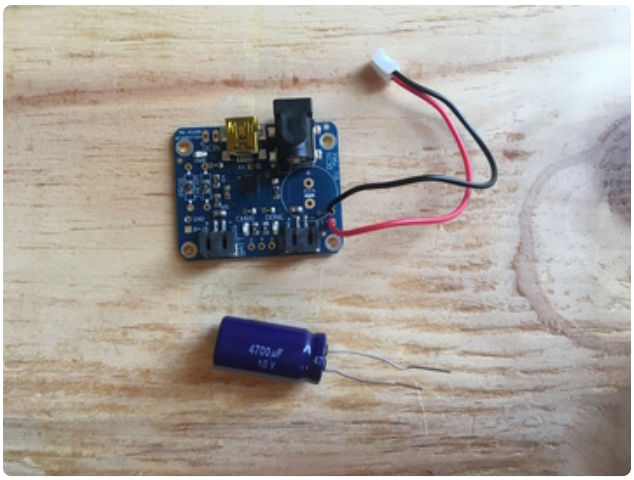
First, let's solder the Solar Lithium Charger (http://adafru.it/390)



This handy component will be used to regulate solar power from the solar panel and safely charge the lithium battery. The lithium battery will be powering the CPX. The reason the capacitor is so large is that it is needed to stabilize the solar panel as the current can be quite high. For extensive information on the Solar Lithium Charger check out the guide. (https://adafru.it/CfO)



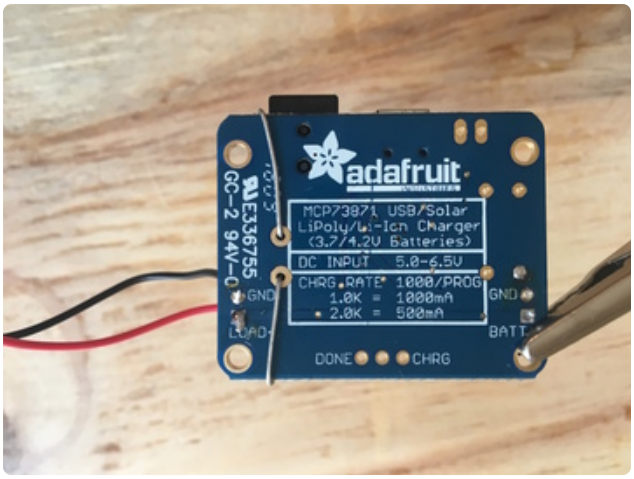


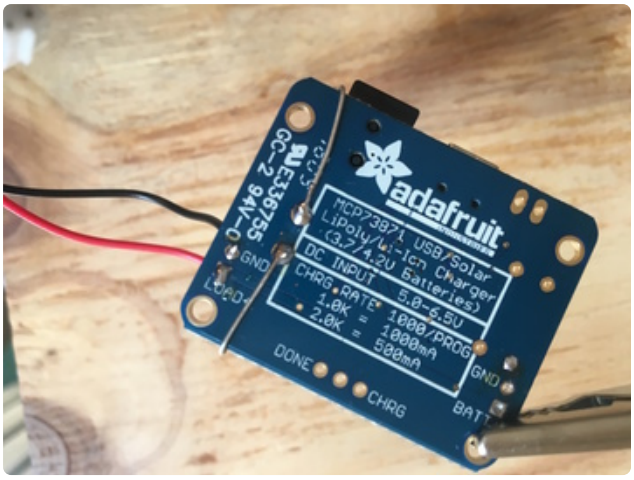


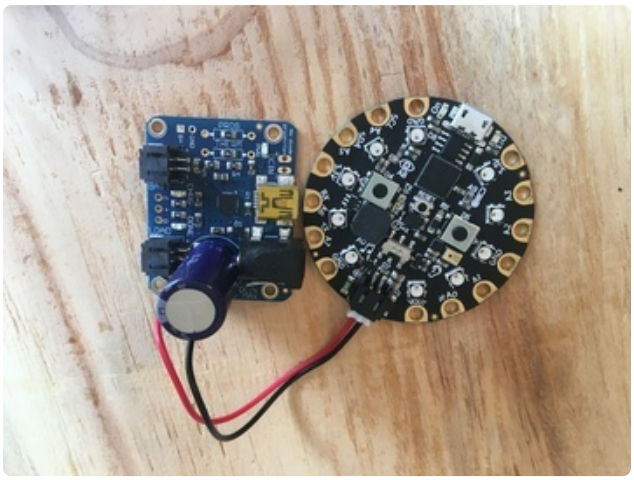
First let's solder on the red and black wired JST connector to the `"LOAD"` side of the charger.

Next we'll solder on the capacitor. **Make sure the longer leg is put through the positive hole.**

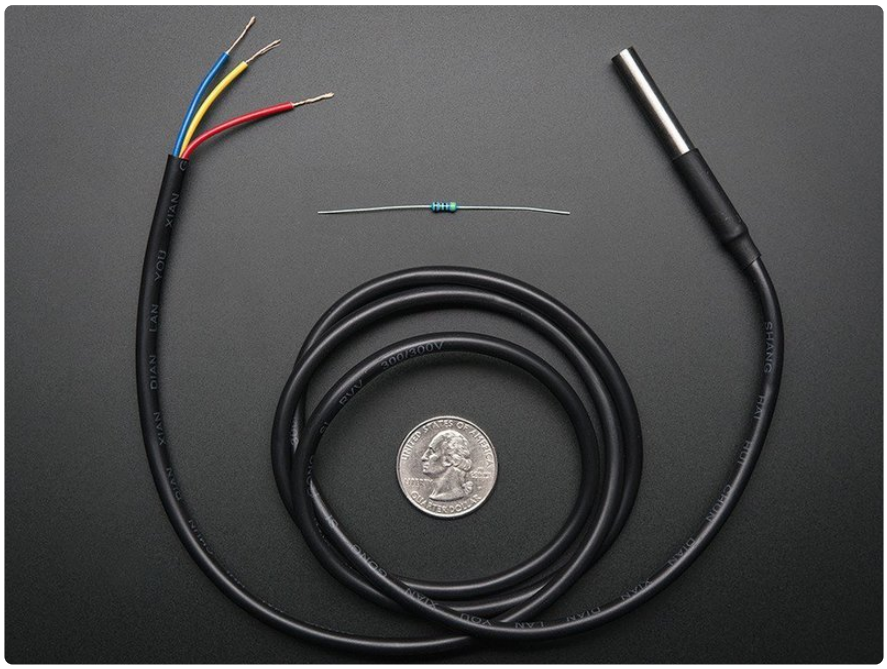
The JST connector will eventually be plugged into the CPX to power it.

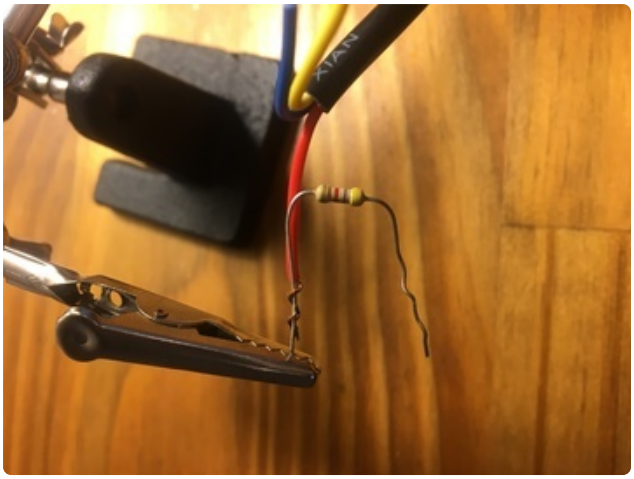

adafruit
MCP73871 USB/Solar
LiPoly/Li-Ion CHarger
(3.7/4.2V Batteries)
DC INPUT 5.0-6.5V
CHRG.RATE 1000/PROG
1.0K = 1000mA
2.0K = 500mA
GND
BATT
DONE
CHRG
LOAD-
E336755
GC-2 94V-0


adafruit
MCP73871 USB/Solar
LiPoly/Li-Ion CHarger
(3.7/4.2V Batteries)
DC INPUT 5.0-6.5V
CHRG.RATE 1000/PROG
1.0K = 1000mA
2.0K = 500mA
GND
BATT
DONE
CHRG
LOAD-
E336755
GC-2 94V-0

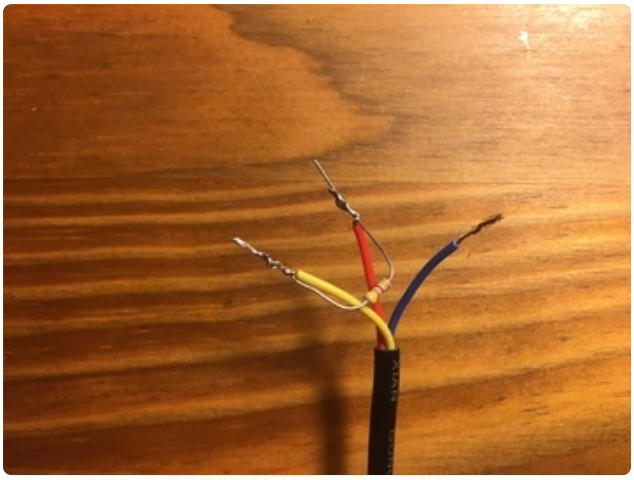


Next we'll solder the DS18B20 temperature sensor to the CPX.



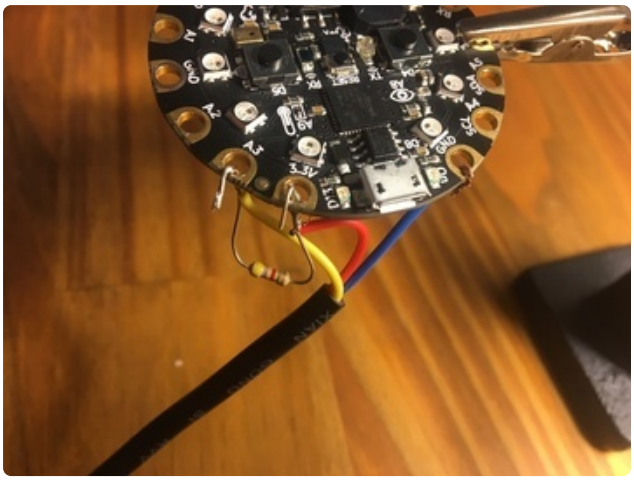


We need the `4.7K` resistor to be between the `power (red)` and `data (yellow)` lines. We solder the resistor on accordingly.



Next wrap the blue line around the `GND` on the CPX. Specifically the `GND` next to the micro usb port.

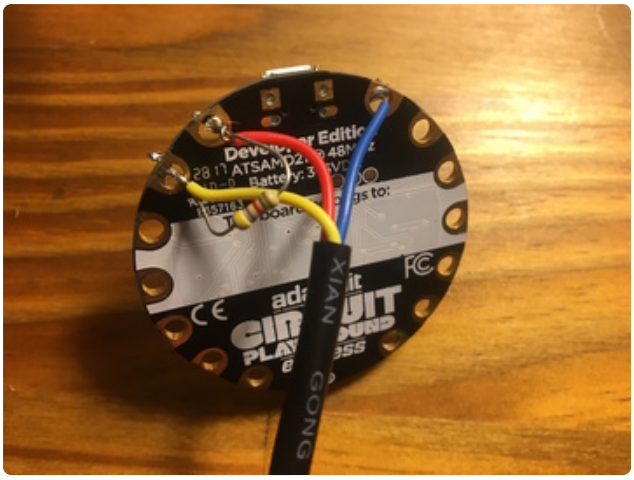
Wrap the red line around the `3.3V` specifically the one on the other side of the `GND` we just wired.



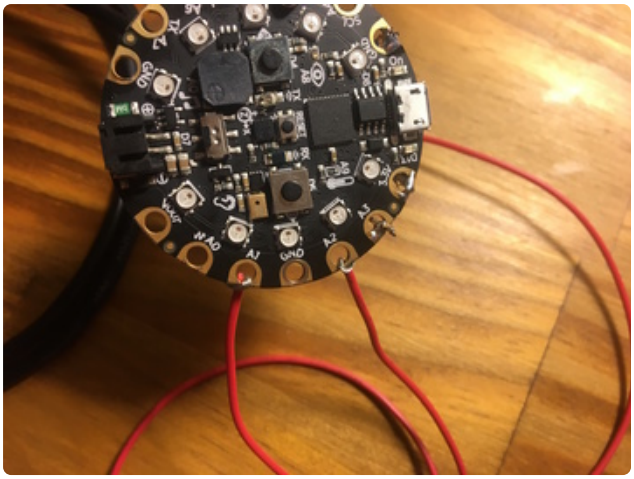
Wrap the yellow line on `A3` input.

Solder the wires in place and cut off the excess resistor / wire.

Find further information on the DS18B20 temperature sensor here. (https://adafru.it/CoV)

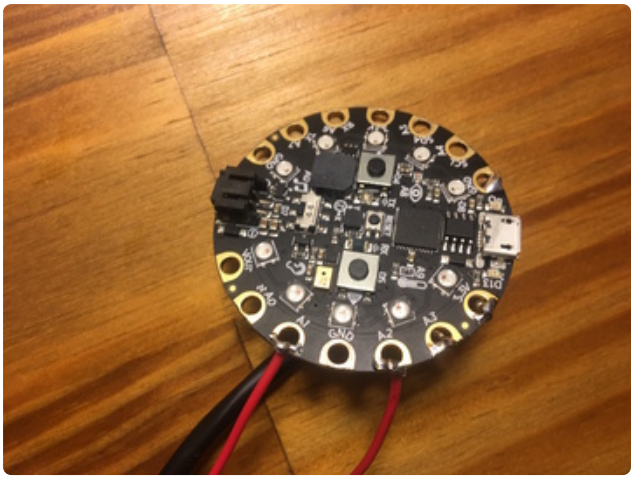


Now we'll solder the moisture nails to the CPX.



Cut **two 12"** of pieces of **22 - 28 gauge stranded wire**.

Remove about **0.5"** of wire wrap on each end.

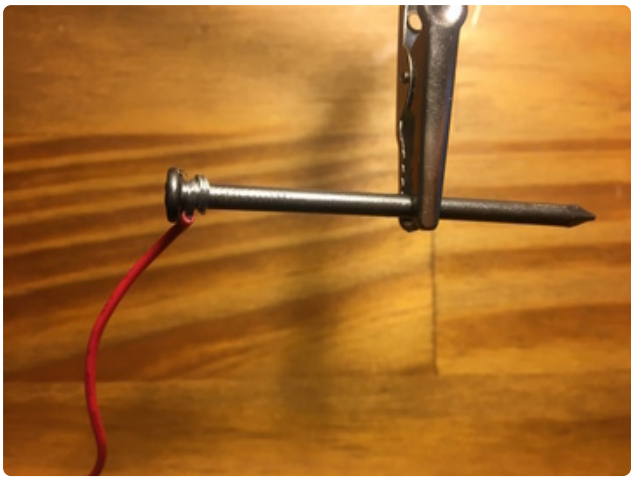


Wrap the end of one wire to `A1` on the CPX and the other to `A2`.

Solder the wires in place.

Wrap the other sides to the two nails.

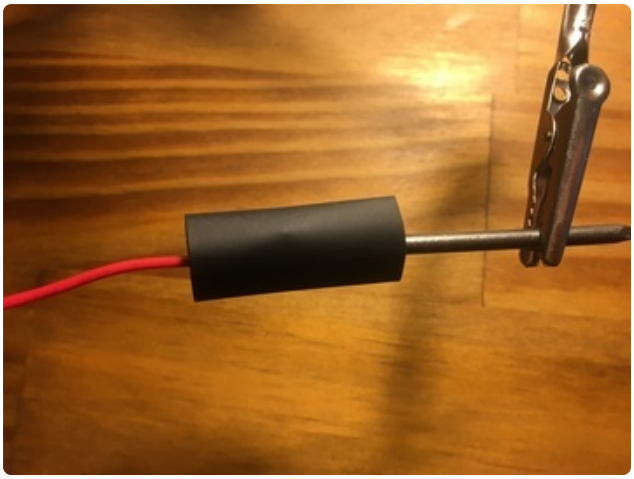
Solder them to the nails.

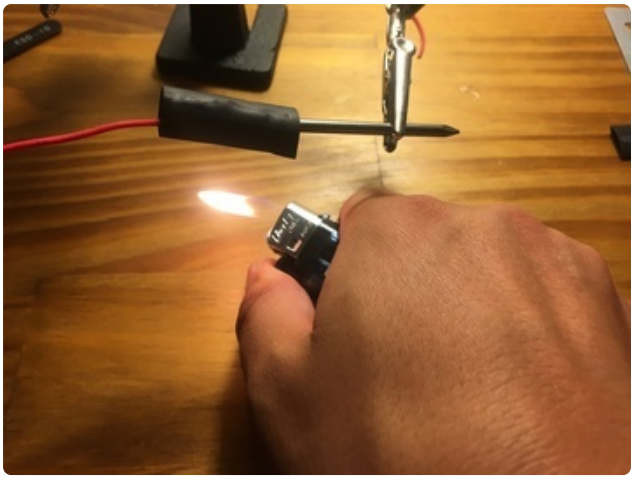


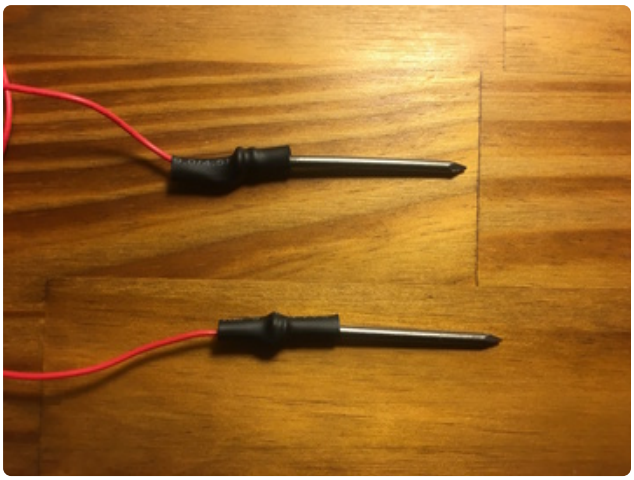
Cut **two 1"** pieces of **3/8" heat shrink**

Slide them over the the nail until the nail tops are in the center of the heat shrink.

Use a lighter, match or other heat source like the hot body of a soldering iron (not the tip) to activate the heat shrink.



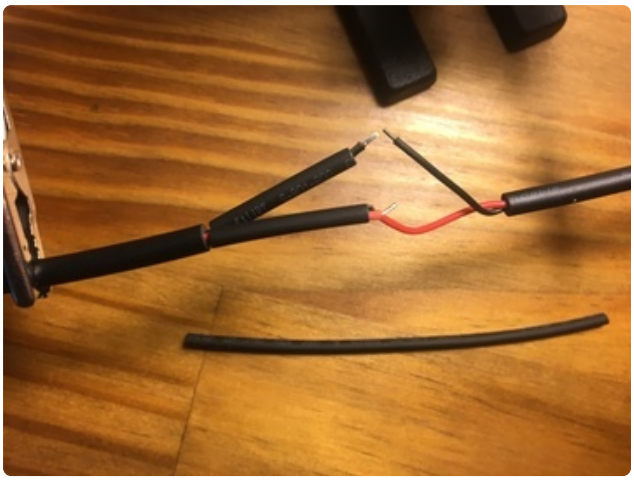




Connect the CPX to your computer to make sure the sensors are working and that all the connections are good before we move on.

Soldering the DC jack

In order to keep the power line from the solar panel, as well as the enclosure waterproof, we will have to place a cable gland though the DC lines before we solder them together. We must do this because the gland does not fit over the thick part of the jack.



Take one of the cable glands, unscrew the smaller piece off and keep it close by (we'll need it later). Slide the gland over the male DC jack end.

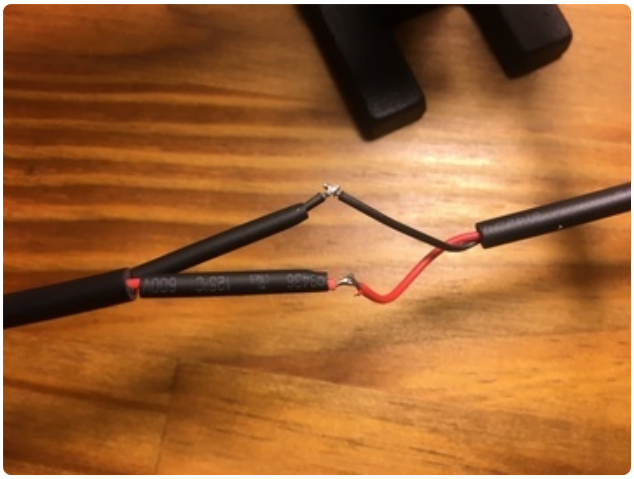
Cut a **3"** piece of **1/4" heat shrink** and slide it over the female DC jack end.

Next cut **two 1.5"** pieces of **3/32" heat shrink** and slide each on over the `ground (black)` and `power (red)` wires.

Solder the male and female DC jack ends together (black to black and red to red).

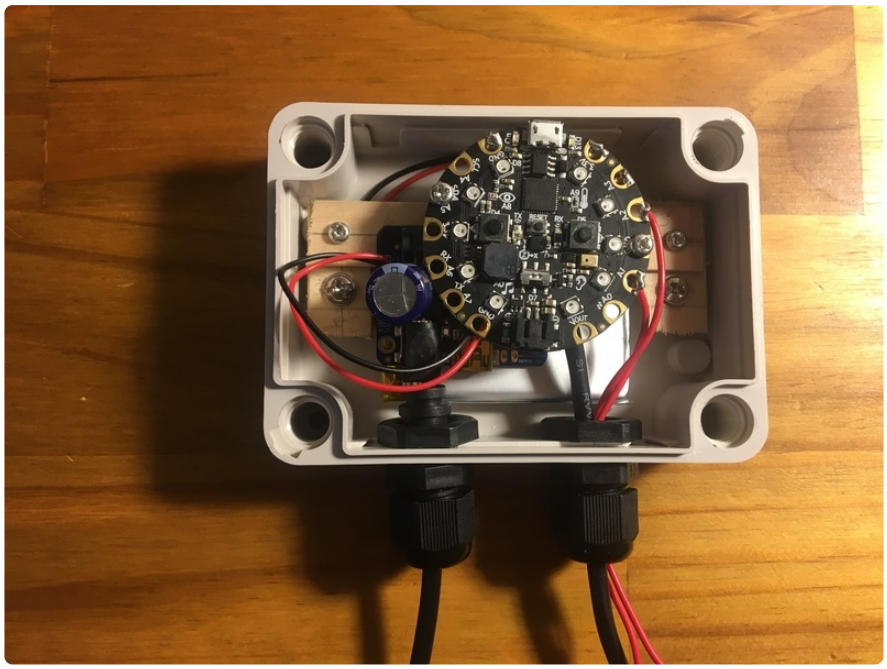
Slide the smaller heat shrink over the ground and power. Activate heat shrink with a heat source.

Slide the larger heat shrink over the wires and smaller heat shrink. Activate heat shrink with a heat source.



Now that we have things soldered up, we'll have to mount it all to the enclosure!

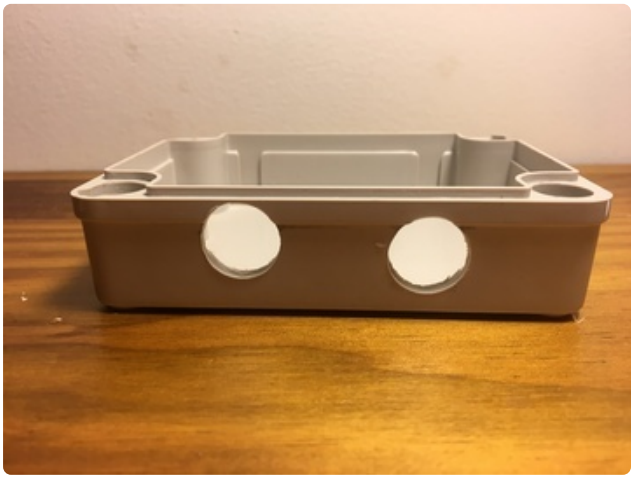
Enclosure



We need a study enclosure and mounting system to hold our electrical components in place. It's also necessary that no water be able to get into the enclosure. We'll be using a small water proof enclosure along with some popsicle sticks and various screws to accomplish this.

Drilling Sensor Holes

The enclosure we're using is great for drilling holes. We'll need our sensors and solar panel to connect to the components in the box but be outside of it. These holes will allow this to happen. Later we'll waterproof the holes.



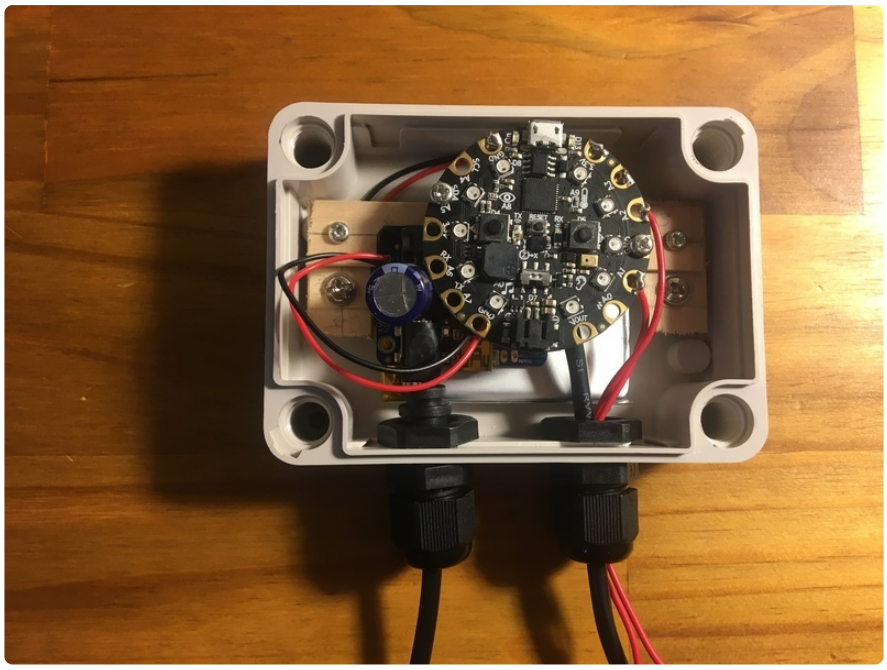
On the left side of the enclosure (orientation doesn't matter), draw a large X with a pencil **1 & 3/8" from the left side** and **1/2" from the top** of the opaque half of the enclosure. This hole will be for the solar panel DC cable and needs a bit more height (because of the thickness of the jack end) than the hole we'll draw from the sensors next.

Use a **5/8" drill bit** and drill a hole through the center of the X you drew.

On the right side of the enclosure draw a large X **1 & 3/8" from the right side** and **5/8" from the top**. This hole will be used for the sensors.

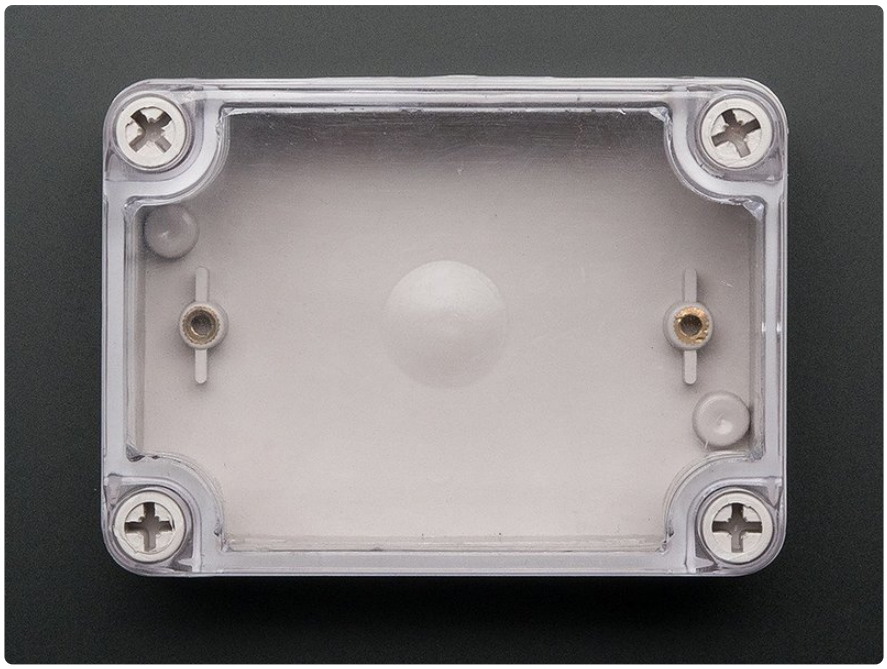
Use the same **5/8" drill bit** to drill a hole.

Mounting Overview

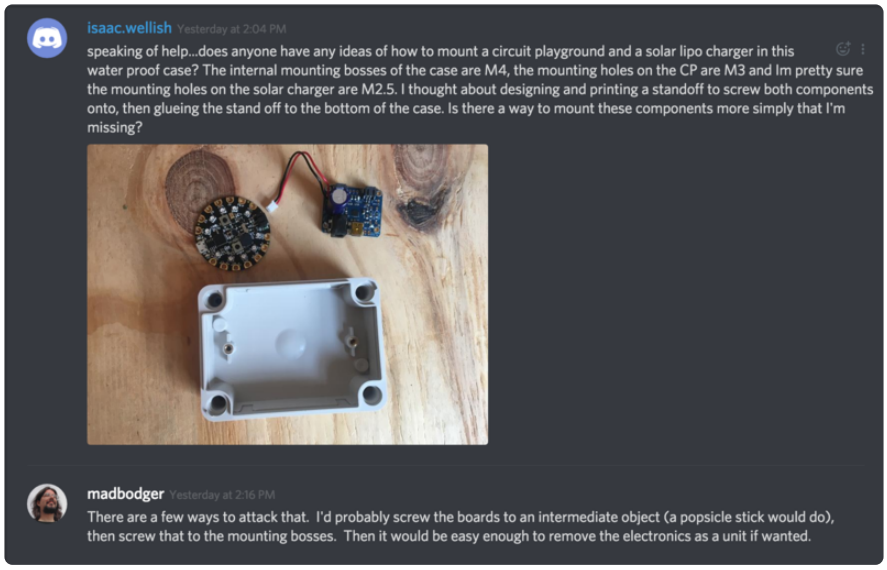


To mount the electronics to the enclosure we'll be using... popsicle sticks! Yes you heard me right! You can find them at your local dollar store.

These sticks work well with the built in mounting holes in the enclosure. They are inexpensive and plentiful. If you mess up it's ok just grab another! Sometimes called "craft sticks", these are actually a bit thicker than popsicle sticks and should measure **5/8" by 5 & 1/2"**

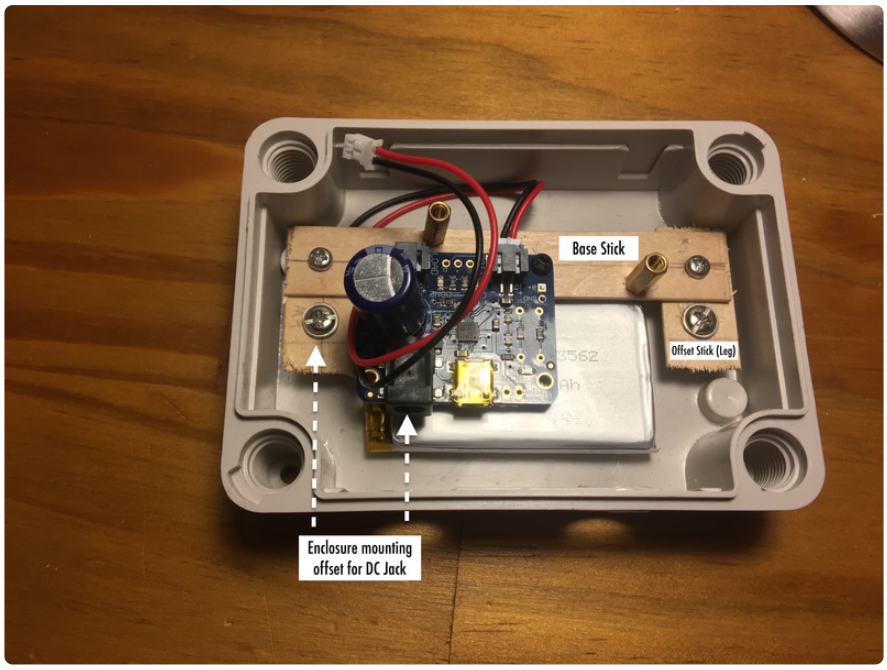


When struggling deeply in figuring out how to securely mount the electronics for this project, I went to the Adafruit Discord () and posted about my problem in the #projecthelp channel. The popsicle stick suggestion was given to me by user "madboger" in a matter of MINUTES after posting, and turned out to be a life saver! More info on the Adafruit Discord. (https://adafru.it/xKc)



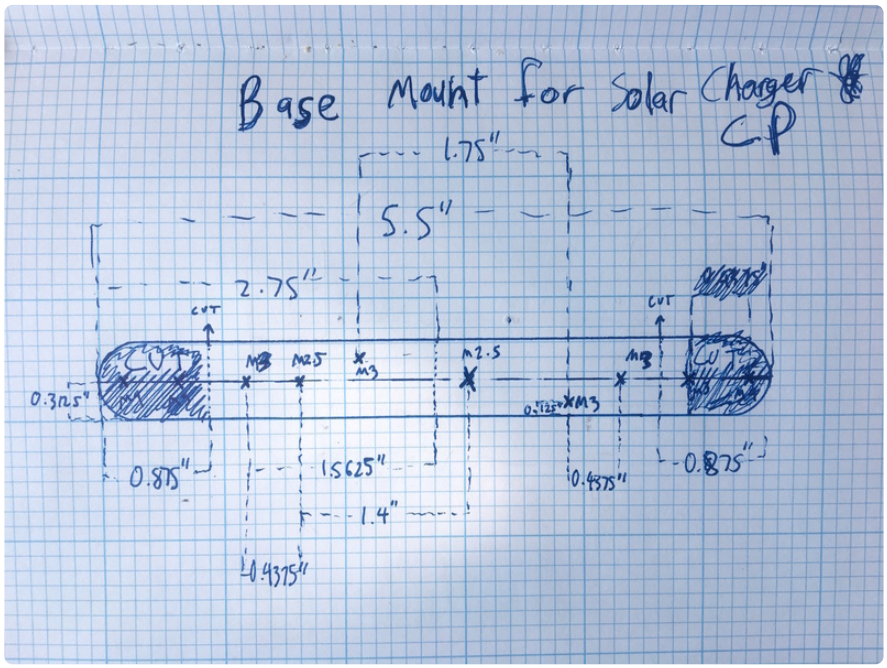
If you ever have problems with a project, don't hesitate to head over to the Adafruit Discord! It saved this project!

Popsicle Stick Preparation

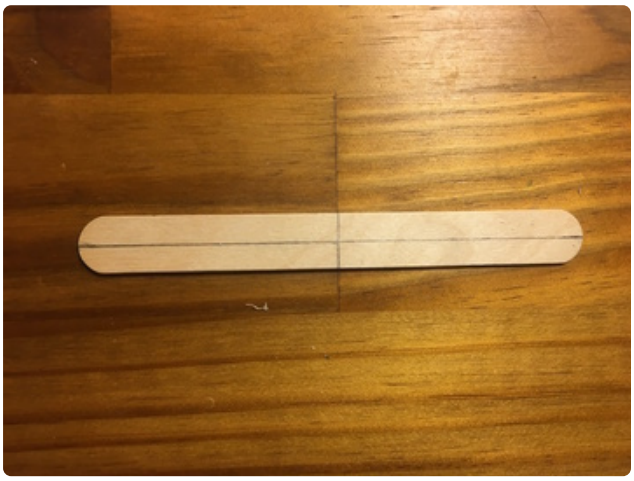


We'll need two popsicle sticks to make the mounting unit for the enclosure. One stick will be the base that holds the electronic components, the other will be to make two legs to offset the base enough so the DC jack from the solar panel can reach the solar charging unit in the enclosure correctly.

Base Stick Preparation

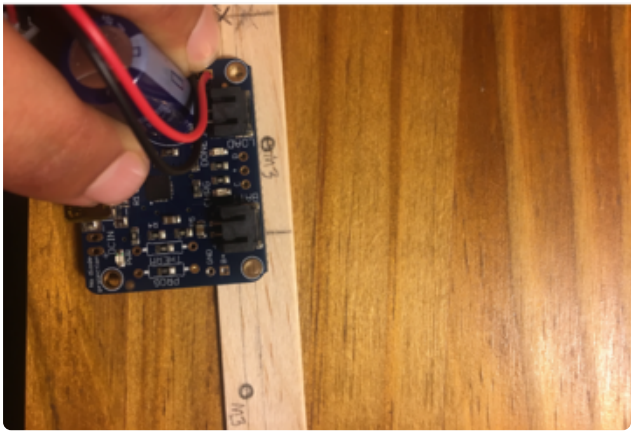


Find the master dimension sheet for the base stick above. Each time you see **M2.5** and **M3**, these represent different sized holes needed for specific mounting screws. The solar charger unit will use the **M2.5** screws and the CPX and one side of the offset legs will use the **M3** screws.

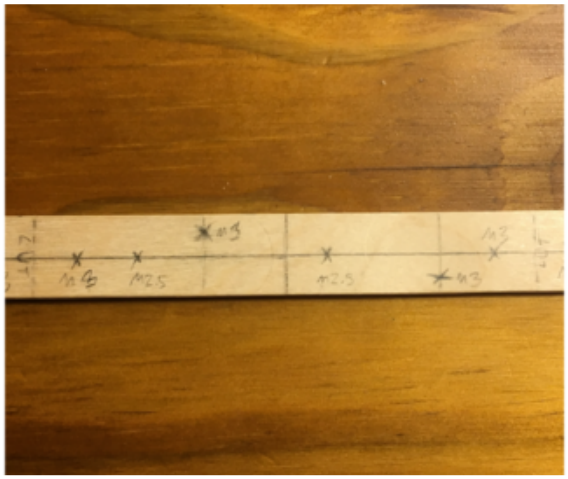


Take one of the popsicle sticks and draw a line though the center length wise at **5/16"** and width wise at **2 & 3/4"**.

Next draw an X **1.5625" or 1 & 9/16"** to the left and right of the center of the stick. Label these with an **M3**. These will connect to the offset legs later.



Now draw an X **0.4375" or 7/16"** to the left of the right side **M3** X we just drew and **1/8"** from the bottom edge of the stick. From this X draw another **1 & 3/4"** to the left and **1/8"** from the top edge of the stick. Label these **M3** as well. They will be for the CPX.



Lastly draw an X **0.4375" or 7/16"** to the right of **M3** X we drew on the left hand side. Now take the solar charger unit and hold it so that the capacitor is on the left side and with the left mounting hole over the X we just drew, mark the stick through the right side hole while making sure the unit is aligned parallel with the stick. edge. This will close to guarantee that the holes are the right distance to mount the unit.

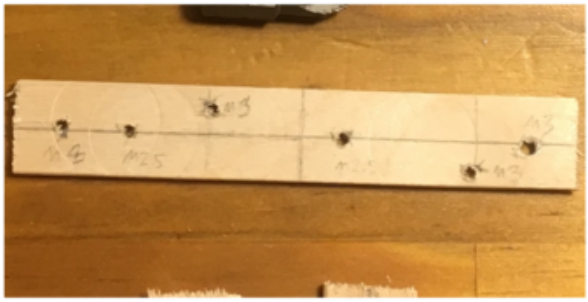
Drill Holes and Remove Excess

Use a clamp to hold the popsicle stick in place when drilling. Drill into stick on top of a table with some scrap wood or similar material underneath as to not damage the table.

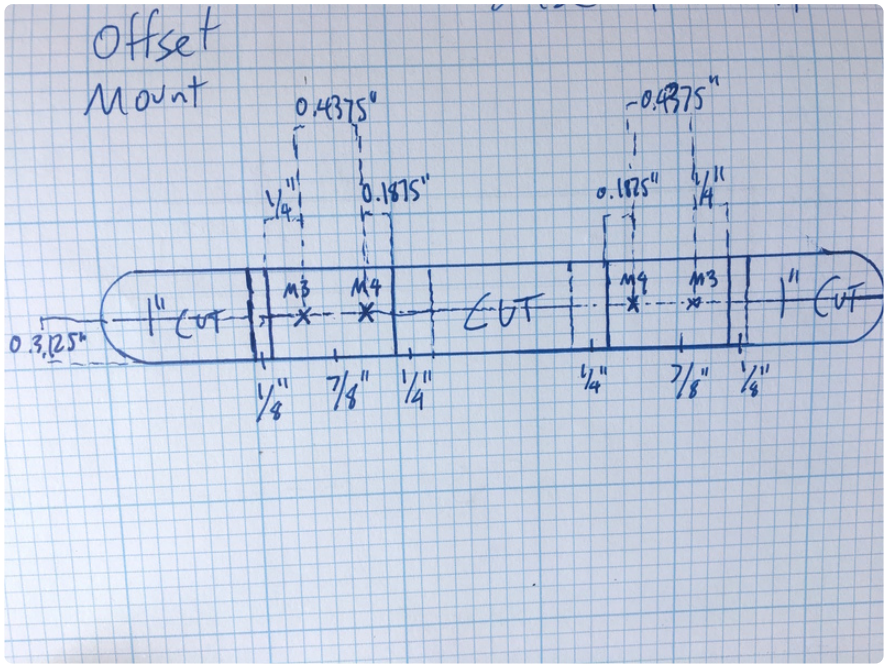
Use a **3/32"** drill bit to drill holes for all **M3** labeled Xs.

Use a **#46** drill bit **(0.081")** for all **M2.5** labeled Xs.

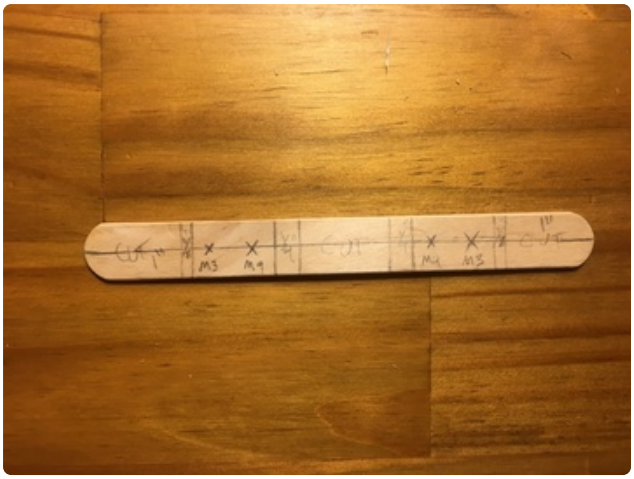
To remove excess on ends, hold stick over table edge so the cut line is aligned with edge. Hold one hand on stick as close to edge as possible. Use the other to slowly apply more and more pressure on the stick end until the edge bends off. Pop off the excess.

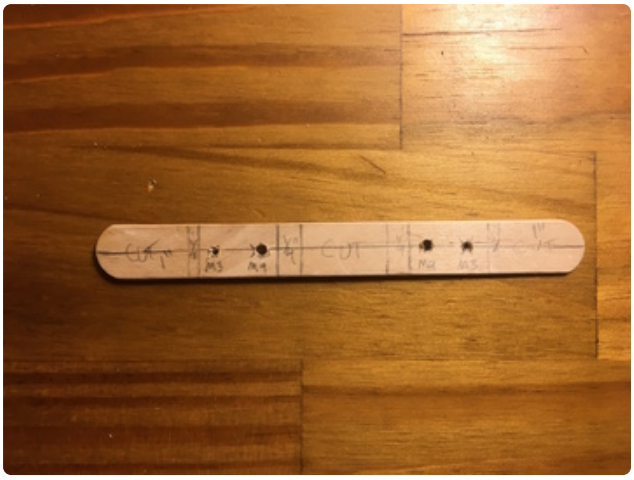


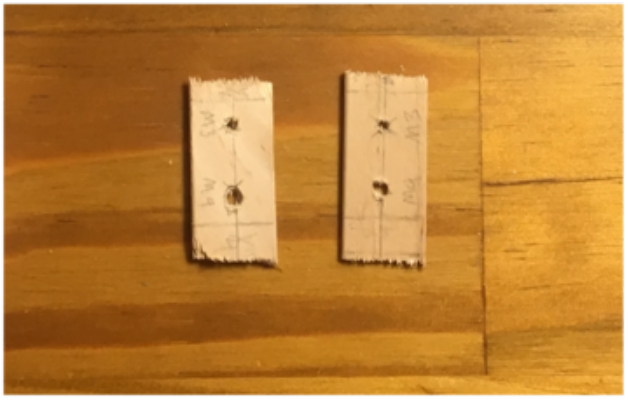
Offset Stick (Legs) Preparation



Find the master dimension sheet for the offset mounting stick above. The **M4** screws will be used to mount to the enclosure internal mounting holes. The **M3** screws will be used to attach the offset legs to the base. The reason for the various spacing in between each of the leg cuts is to give the stick enough room so that the holes do not strip when the excess is removed.







Take the other popsicle stick and draw a line through the stick length wise at **5/16"**.

Draw a line **1"** from the left side and **1"** from the right. These will later be used to remove excess.

Draw another line **1/8"** to the right and left of the lines you just drew.

Mark an X **1/4"** to the right and left of these lines and label **M3**.

Mark an X **0.4375" or 7/16"** to the right and left of the most recently drawn Xs and label **M4**.

Draw another line **7/8"** inches from the right and left of the last lines we drew. Draw a line **1/4"** to the right and left of these lines.

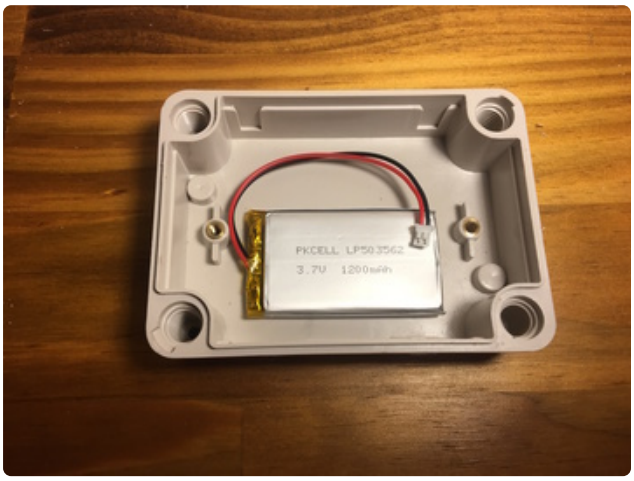
Use a **1/8"** drill bit to drill holes in the **M4** Xs.

Use a **3/32"** drill bit to drill holes in the **M3** Xs.

Remove the excess along the cut lines.

Mounting Continued

Mounting the Battery

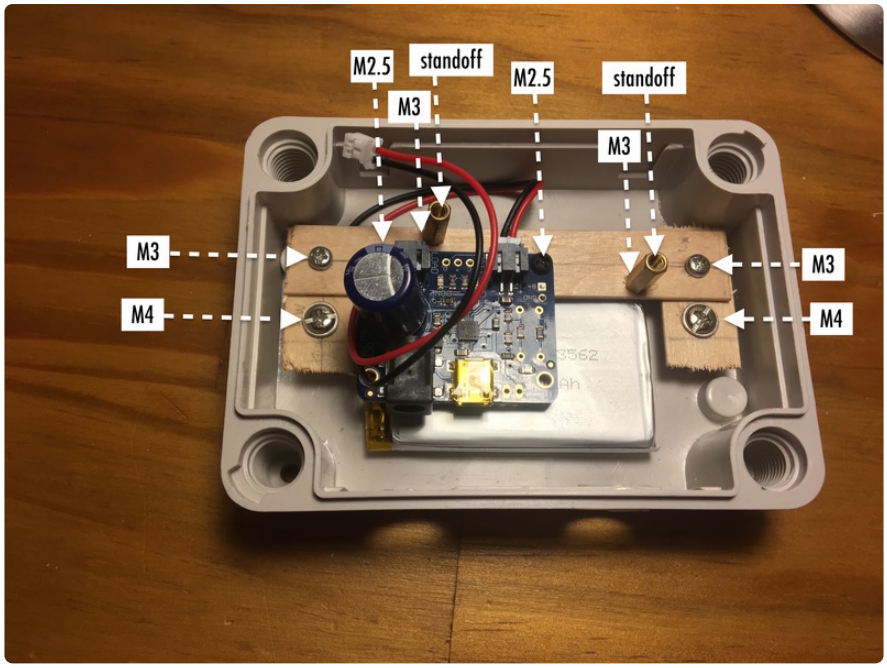


Apply Sugru (http://adafru.it/437), double-sided tape or super glue to one side of the LiPo battery.

Place the battery on the bottom of the enclosure apply pressure to mount in place.

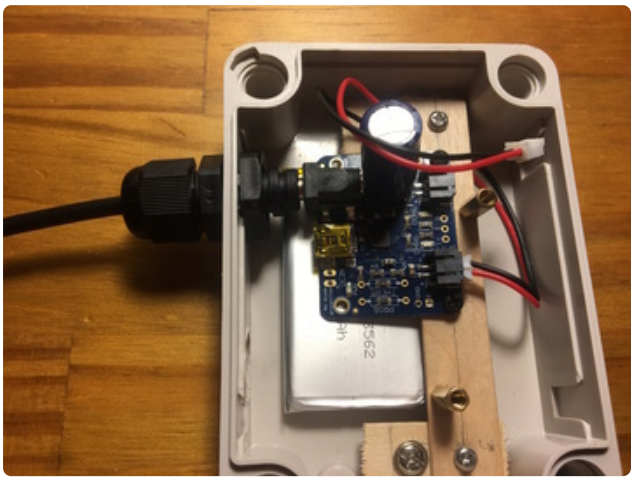
Assembling the Sticks

- Connect two **M3** screws from the ends of the base stick to the **M3** sides of the offset sticks.
- Screw in two more **M3** screws **from the bottom of the base stick.**
- Twist on the stand offs to these screws.
- Screw on the **M4** screws to the enclosure base.
- Screw on the **M2.5** screws through the LiPo charger unit and onto the base stick.
- Plug the battery into the `"BATT"` JST connector on the LiPo charger.



Waterproofing

DC Jack

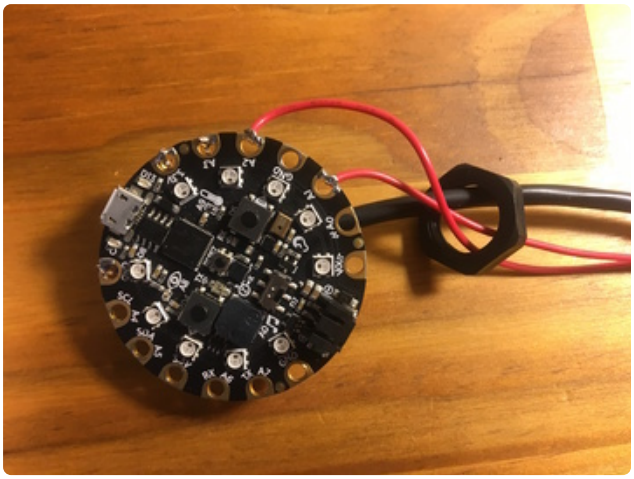


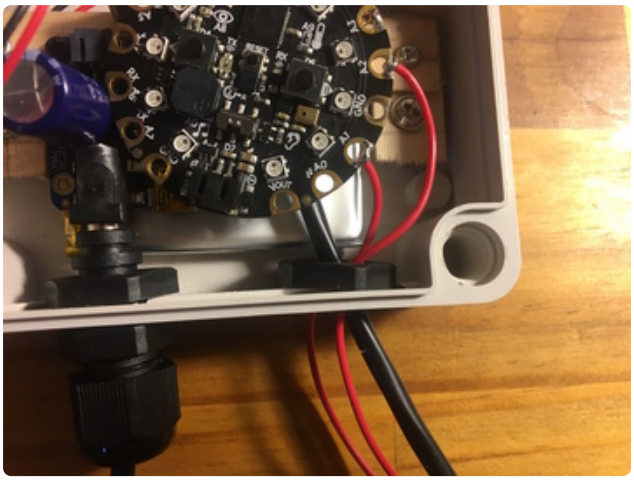
Take the DC connector we soldered from before with the cable gland loosely attached and slide it through the drilled hole on the left side of the enclosure.

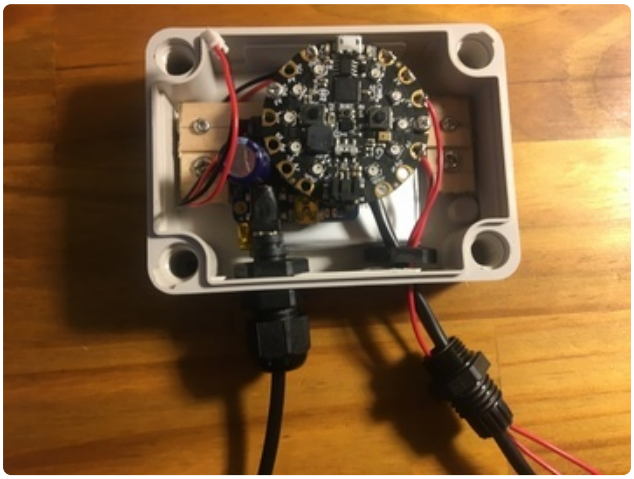
Make sure to slide the small plastic twist component through the other side of the DC jack before plugging it into the solar LiPo unit.

Screw the entire gable gland together. It is now waterproof.

Circuit Playground Express and Sensors







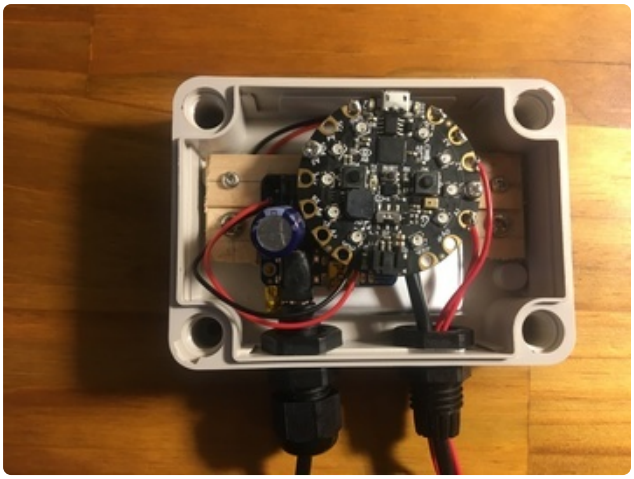
With the other cable gland, unscrew the small plastic piece and slide it through the sensors.

Slide the sensors through the other drilled hole in the enclosure.

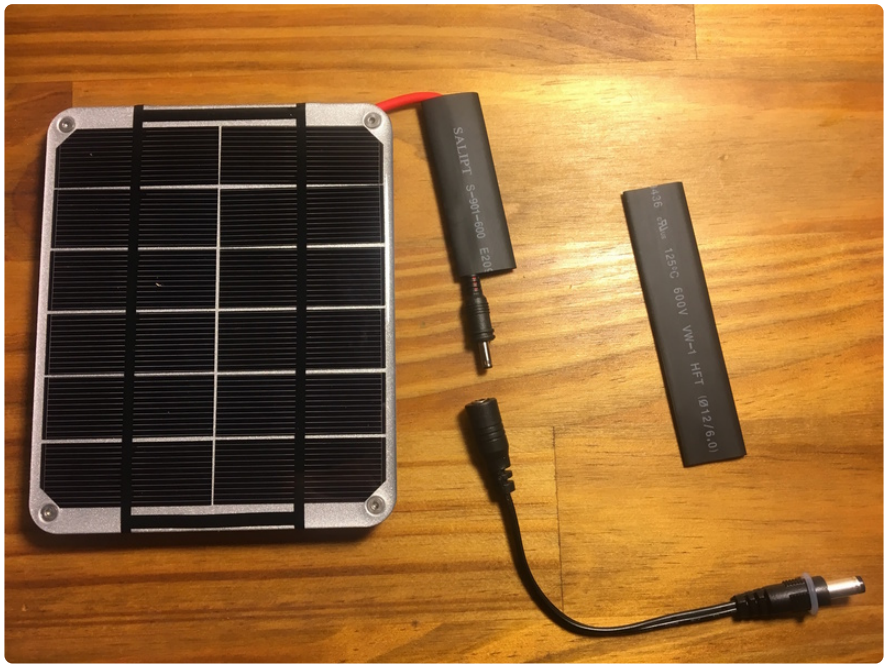
Slide the other part of the cable gland through the sensors.

Screw the cable gland together.

Screw the **M3** screws over the CPX into the standoffs.

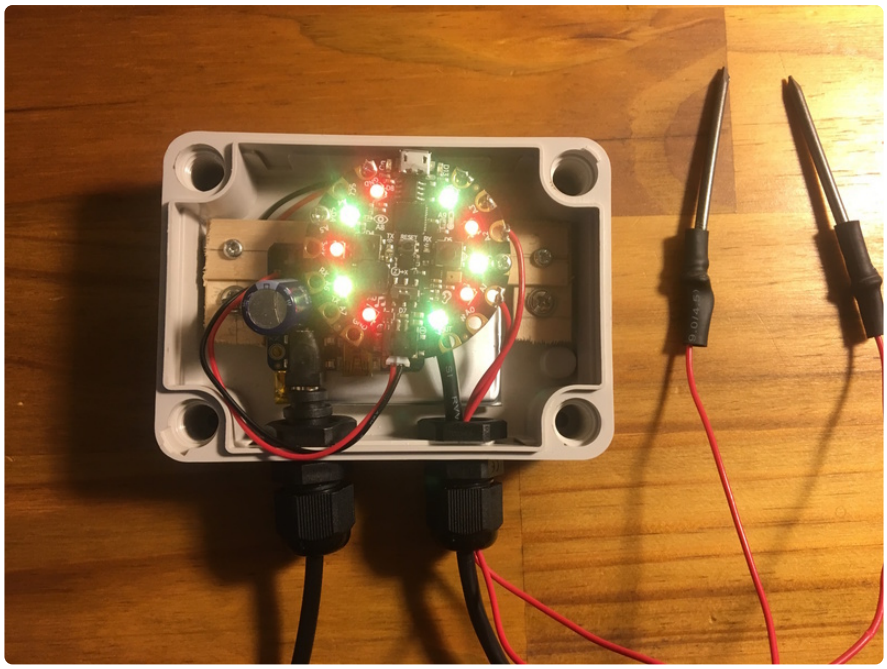


Solar Panel Instillation

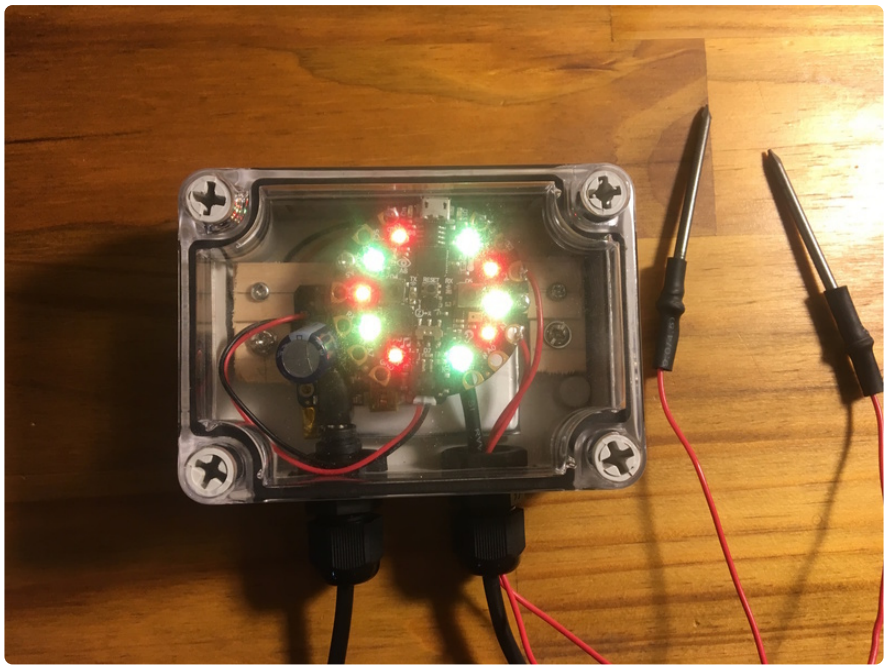


Cut one **2"** piece of **3/8"** heat shrink.
Connect the solar panel DC jack with the DC adapter cable.
Pull the heat shrink over the connected cables and activate.
Connect the adapter to the DC jack from the enclosure and screw on the plastic piece keeping the O-ring up against the female end to keep out moisture.

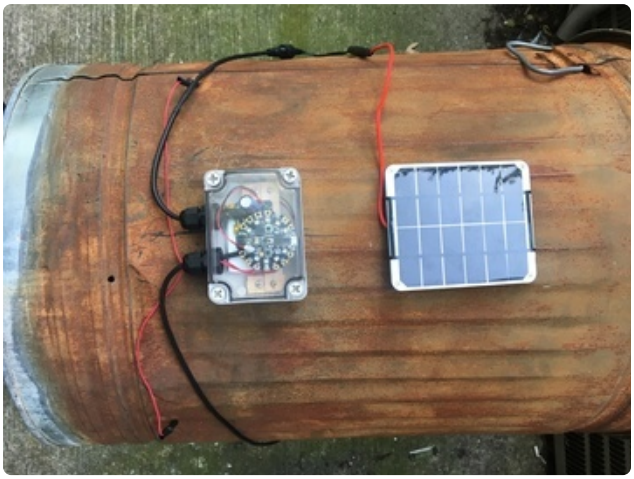
Plug the JST connector from the LiPo charger into the CPX.

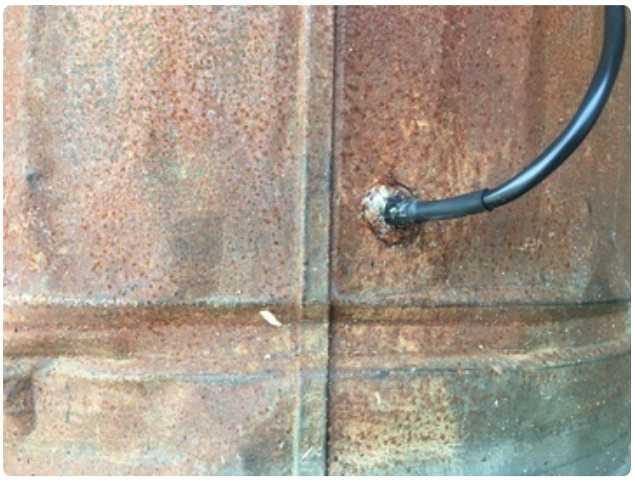


Screw in the 4 plastic enclosure screws and we are in business!



Attaching to Bin





Place enclosure and solar panel in desired locations on bin.

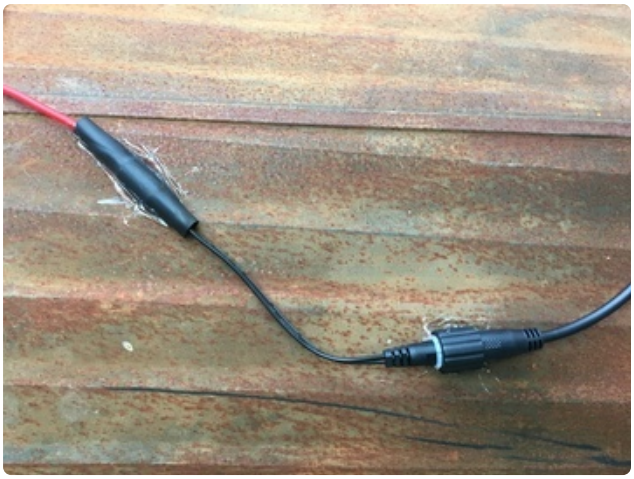
Put sensors in holes that are far away from each other. If you don't already have holes in your bin you may need to make some.

Use a glue gun or some sort of mounting medium to mount enclosure and panel in place.

Glue sensors in place.

Glue various cables so they do not dangle.

Re-attach JST connector to CPX if unplugged.



When placing the bin outside, try a place where sunlight can directly come in contact with the solar panel.

*Consider angling the solar panel towards the sky before installing. This will ensure more direct sunlit hitting the panel. **20 - 40 degrees** up from the current position would be a good range.

Done!

Lessons Learned

This project was quite a rollercoaster. I learned so much in the process. Like:

- DISCORD IS YOUR FRIEND.

- Popsicle sticks are a cheap way to build a sturdy mounting system inside an enclosure but... popsicle sticks are cheap. Popsicle sticks will break a lot. Take a break from looking at popsicle sticks after this project.
- Using a table or object you can drill into helps tremendously when drilling with a clamp.
- It's better to leave inches as fractions than decimals as most rulers are measured in 1/16s of an inch.
- Don’t use the wrong size drill bit...
- Drill before cutting off popsicle stick excess. The holes have a tendency to strip if you don't.
- Make sure dimensions are RIGHT before continuing a build.
- Using nails as moisture sensors works OK but I'm looking forward to an improved moisture sensor Adafruit may or may not be working on.
- Composting is cool. Turning your compost bin into a bionic solar powered bot is COOOL.

This project was a proof of concept that tracking the temperature and moisture of compost can help tremendously to improve the health of the compost. In the future I would like to add an IoT component to this project to be able to track temperature and moisture remotely through Adafruit IO. The next version may also have facial features... Until then, stay tuned, and stay composting!