

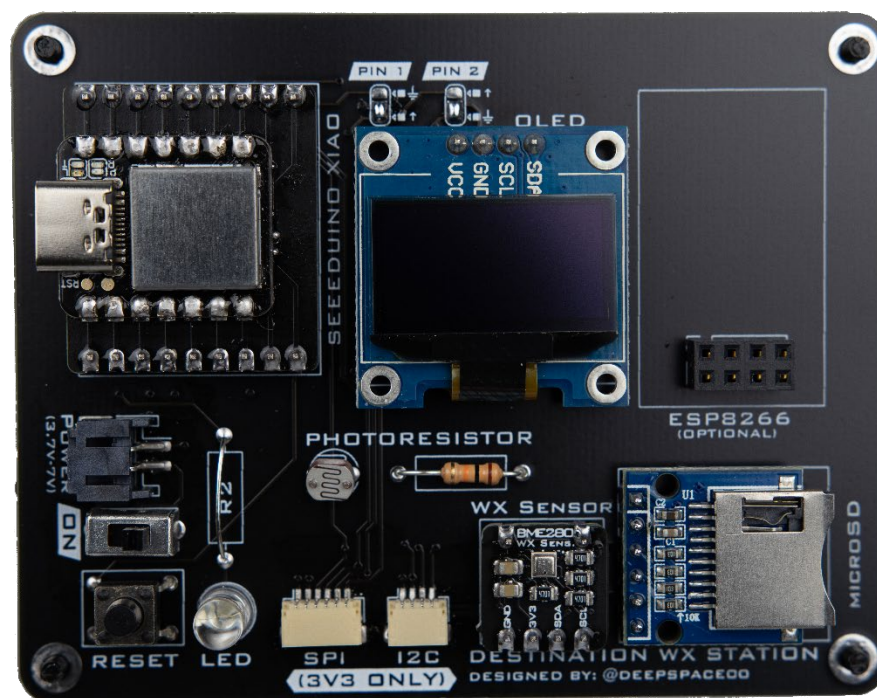


SOUTH CAROLINA GOVERNOR'S SCHOOL FOR

**SCIENCE+
MATHEMATICS**



DESTINATION WEATHER STATION REVISION 3 (SOLDERLESS) CURRICULUM





CONTENTS

Introduction	3
1. Destination Weather Station	4
2. DS WX. STA. Overview	4
2.1 Seeedunio XIAO	4
2.2 OLED Display	4
2.3 Light Sensor	4
2.4 BME280 Weather Sensor	5
2.5 microSD Card Module	5
2.6 LED	5
3. Getting Started	5
Coding Example 1: Hello World!	6
Coding Example 2: Blink	8
Coding Example 3: OLED Test	10
Coding Example 4: Light Sensor	12
Coding Example 5: SD Card Test	14
Coding Example 6: BME280 Test	17
Coding Example 7: Full Test	20
4. Uploading Code	26
4.1 Arduino IDE	26
4.2 UF2 Bootloader	26
Assembly Instructions	27
Step 1:	28
Step 2:	28
Step 3:	29
Troubleshooting Tips:	30
Weather Station Teacher Checklist	32
Destination SPACE Software Setup	33
Step 1:	34
Step 2:	34
Step 3:	35
Kit Components	36



INTRODUCTION

This document contains all documents needed for the 2022 GSSM CREATEng camps. In this document is an introduction to the weather station with a brief overview of each of the major components. Also contained in this document are several coding examples for the students. The main ones to focus on are Hello World! and Final Test. Followed by these are upload instructions, assembly instructions with troubleshooting guide, a checklist for instructors, software setup guide, and a kit components sheet. The only documents students should need are the coding examples used and the assembly instructions with troubleshooting guide. Each instructor should have or should have access to this entire document so they can familiarize themselves with the platform and the material being taught, as well as a reference incase a student needs assistance. Coding example videos can be found in the link below. Additionally, this material as well as program files can be found in our GitHub repositories linked below.



Videos: https://www.youtube.com/watch?v=XvsUvmIJXvo&list=PLfRPz5F_Jd0_xJJGgX0tyzGxaJ-VCzVwo

GitHub: <https://github.com/Destination-SPACE/Weather-Station-r3>

Program Files: <https://github.com/Destination-SPACE/Weather-Station-r3-Software/tree/main/examples/Arduino>



1. DESTINATION WEATHER STATION

Destination Weather Station is a low-cost weather sensing platform designed to engage students with remote sensing, electronics design, programming, and data analytics. This tool can be used to introduce these skills in an exciting, hands-on way, sparking scientific exploration and discovery. Destination Weather Station R3 is the third generation of platforms offered to help teach these areas of STEM to students and to inspire them to learn more.

This platform is our solderless version of our R3 kit where all components can be plugged directly into the weather station. The focus of this solderless platform is to emphasis introduction to Arduino (C++) coding language, remote sensing, and data analytics. This document contains an overview of the R3 kit, coding examples, and remote sensing activities.



2. DS WX. STA. OVERVIEW

Destination Weather Station is outfitted with many sensors, and components for students to learn about remote sensing.

The major components on this kit are to follow:

- a) Seeeduino XIAO microcontroller
- b) 128x64px OLED display
- c) Light sensor (photoresistor)
- d) BME280 humidity, pressure, and temperature sensor
- e) microSD card module
- f) LED

2.1 SEEDUNIO XIAO


The Seeeduino XIAO is a type of Arduino microcontroller. This microcontroller is powerful enough to control the OLED display, the microSD card module, and the sensors. The main advantage of using this over a standard Arduino microcontroller such as an Arduino UNO, Nano, or Mega is it is more powerful than the UNO and Nano, and it costs less than a Mega. This allows students to have a great experience with Arduino, without any of the limitations. Another advantage is that the XIAO can be programmed by dragging and dropping code files directly onto the XIAO without needing to use the Arduino IDE. This is beneficial for scenarios where a student is having issues getting code to upload, or there is an issue with the computer.



2.2 OLED DISPLAY

The OLED display included in this kit has a pixel area of 128x64px. This can be used to display data readings from the sensors, or to display text.

2.3 LIGHT SENSOR



The light sensor in this kit is a standard photoresistor. The photoresistor changes resistance dependent on the amount of light present. The brighter the light the lower the resistance. This can be used to approximate how bright the ambient light inside a room is, or how bright it is outside. This is the same sensor you may be familiar with in night lights that turn on in the dark.

2.4 BME280 WEATHER SENSOR

The BME280 weather sensor can measure temperature, pressure, and humidity. This sensor has a broad temperature range, so it can even be used outside in the winter, or on the hottest summer days.

2.5 MICROSD CARD MODULE

This module allows students to record their data to a microSD card. This can then be plugged into the computer where the data can be analyzed. The data format used is a CSV file, which is compatible with Microsoft Excel and Google Sheets.

2.6 LED

This is a standard LED. It can be programmed to turn on and off.

3. GETTING STARTED

Provided are assembly instructions that should be passed out to students. The typical amount of time required to assemble the kit is about 10 minutes. A background on what each device/sensor does can be taught, but students, particular younger ones, find it difficult to keep up.

Before students power on their weather stations ensure that all of the devices are plugged in the correct orientation. There should be ample detail in the instructions indicating which way they should face as well as indications on the weather station itself, but there is nothing preventing students from plugging them in backwards. If a device is plugged in backwards, it could potentially break the entire kit.

Once kits are fully assembled students are free to plug their weather stations into their computers and begin coding examples and activities. Not every example or activity must be completed, but a variety should be offered. Jumping straight to the fully integrated code example can be confusing, the earlier examples are intended to familiarize students with the functions and context of each sensor before everything is fully integrated. Same can be said with the remote sensing activities.

The following coding examples and remote sensing activities can be printed out or provided as PDFs to students. Each example has an estimated time of for completion, background describing the intent of the activity, steps for completing it, and a worksheet for students to complete when done. Turn-key code is provided at the end of each example.



CODING EXAMPLE 1: HELLO WORLD!


In this example you will learn how to upload code to your Destination Weather Station and learn how to use the Serial Monitor. This is an iconic example and is usually the first thing you test when learning a new programming language.

OBJECTIVE: Print Hello World! on the Arduino Serial Monitor

To get started open Arduino IDE on your computer. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

`void setup()` is used to define setup conditions for the program.




To setup the serial monitor we need to define a bitrate in the `void setup()` function. To do this we use the function `Serial.begin()`. Inside of the parenthesis is where the bitrate is declared. In this example we will use 9600.

`void loop()` is used as the main body of the code. The microcontroller will repeat this loop as long as the Arduino is powered on and uninterrupted.

Inside this loop is where we will have the Arduino print Hello World to the serial monitor. To do this we need to tell it what we want to print. To do this we use the `Serial.print()` function. In this example we will use a slightly different function which will print what's inside the parenthesis on a new line each time, instead of immediately following the previously printed thing. This is defined as `Serial.println()`.

Next we need to figure out what we want to be printed. If you only type what you want to be printed in the parenthesis Arduino will think you are trying to call a variable called Hello World!, which is not what we want. What we want to print is what's called a String. A String is a set of characters that are interpreted literally, meaning what you want is what is what you get. To do this we put double quotes around our text such as "Hello World!". All that needs to be done next is to combine the two to print Hello World!. We can do this by typing `Serial.println("Hello World");`.

The last thing we need to do is add a delay. We add a delay because otherwise the Arduino will try to print Hello World! as fast as it can. We do this by calling the `delay()` function. Inside the parenthesis is where we define how long we want it to wait in milliseconds. In this example we will use 500. We can do this by typing `delay(500);`.



Now let's put all of it together. In the `setup` function define the bitrate, and in the `loop` function define the `print` command and the `delay` command.



EXAMPLE:

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  Serial.println("Hello World!");  
  delay(500);  
}
```



Once you have uploaded the code to your Weather Station you can open the Serial Monitor. To do this in Arduino IDE go to Tools on the top navigation bar and then click on Serial Monitor. This should open the Serial Monitor.

You can change what is printed to the serial monitor by changing what is in the "" and change the delay by changing the 500. You can also add more than one **Serial.print()** by writing it again on a new line.



CODING EXAMPLE 2: BLINK


In this example you will learn how to use the LED on the Destination Weather Station.

OBJECTIVE: Make the LED on the weather station flash.

To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

On the first line we need to define which pin the LED is connected to on the Arduino. We do this so that the Arduino know where it is connected. This can be done by using `#define`. To define the LED pin on the Arduino as pin 0 we type `#define LED 0`.



Inside `void setup()` we need to define what the LED is. The Arduino does not know if LED is an input or an output. We define this by using the `pinMode()` function. Inside of this we define the LED as an output by typing `pinMode(LED, OUTPUT);`.

Inside `void loop()` is where we tell the LED what to do. We do this by using the `digitalWrite` function. We can either set the LED as high (on) or low (off). To do this we type `digitalWrite(LED, HIGH);` or `digitalWrite(LED, LOW);`. Whenever you set the LED as high you should always pair it with a low after a delay so the LED is not always on.

In this example code your Arduino to flash the LED twice with 100ms between when its on and off and then follow it with a delay of 500ms before the code loops back to the top. This will appear as the LED flashing twice.



EXAMPLE:

```
#define LED 0

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(100);
  digitalWrite(LED, LOW);
  delay(100);
  digitalWrite(LED, HIGH);
  delay(100);
  digitalWrite(LED, LOW);
  delay(500);
}
```



CODING EXAMPLE 3: OLED TEST

In this example you will learn how to use the OLED on your Destination Weather Station and learn how to use Arduino libraries.

OBJECTIVE: Print Hello World! on OLED display

To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

To get started with using the OLED display on your Destination Weather Station you need to call a few libraries. Libraries are extra functions built by a 3rd party which make coding easier. We will be using libraries to control the OLED display and for the communication protocol with the OLED. To include libraries we will use `#include`.

To do this type:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Next we need to define the parameters of the OLED display, such as the resolution and the reset.

To do this type:

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
```

The next step is to combine the parameters above so the Arduino knows how to format the display.


To do this type:

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

Now we can move on to `void setup()`. Set the bitrate to 57600. Still within this function we need to define more formatting for the display

To do this type:

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.setTextColor(WHITE);
```



```
display.setTextSize(1);  
display.clearDisplay();  
Wire.begin();
```

Next we can tell the Arduino what to print on the display. To do this we will first clear the display using the function above, set the cursor to 0,0 using `display.setCursor(0,0);`, print Hello World! using `display.print("Hello World!");`, then to print this on the display use `display.display();`.

EXAMPLE:



```
#include <Wire.h>  
#include <Adafruit_GFX.h>  
#include <Adafruit_SSD1306.h>  
#define SCREEN_WIDTH 128  
#define SCREEN_HEIGHT 64  
#define OLED_RESET -1  
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);  
void setup() {  
    Serial.begin(57600);  
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  
    display.setTextColor(WHITE);  
    display.setTextSize(1);  
    display.clearDisplay();  
    Wire.begin();  
}  
void loop() {  
    display.clearDisplay();  
    display.setCursor(0,0);  
    display.print("Hello World!");  
    display.display();  
}
```



CODING EXAMPLE 4: LIGHT SENSOR

In this example you will learn how to use the light sensor on your Destination Weather Station.

OBJECTIVE: Print light sensor data on OLED display

To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

Begin with the same format as the previous example to make the OLED display work. Above `void setup()` the light sensor pin needs to be defined similarly to the LED in example 2. We will define the light sensor as `#define lightSensor A2` because it is an analog device, unlike the LED which is digital.

Next we need to define the data from the light sensor as an integer. To do this we will use the `int` variable. To do this we will use `int lightData;`

The next modifications that need to be made to the OLED example is in `void loop()`. In the first line inside this loop we will record the reading from the light sensor as `lightData` using the `analogRead` function. We will do this by typing `lightData = analogRead(lightSensor);`.

Next we will replace Hello World! with a header for the light sensor. We will use `"Light Sensor:"`. Now we will add a second line for the data reading. To do this set the cursor to 0,10 on the next line. Then print the data to the display using `display.print(lightData);`. The final step is to add a delay of 100ms at the end of the loop.


EXAMPLE:

```
#include <Wire.h>

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define lightSensor A2
```

```
int lightData;
```




```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

```
void setup() {  
    Serial.begin(57600);  
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  
    display.setTextColor(WHITE);  
    display.setTextSize(1);  
    display.clearDisplay();  
    Wire.begin();  
}
```



```
void loop() {  
    lightData = analogRead(lightSensor);  
    display.clearDisplay();  
    display.setCursor(0,0);  
    display.print("Light Sensor:");  
    display.setCursor(0,10);  
    display.print(lightData);  
    display.display();  
    delay(100);  
}
```



CODING EXAMPLE 5: SD CARD TEST

In this example you will test the functionality of the microSD card module by reading and writing to it.

OBJECTIVE: Test the microSD card module

To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

Begin by including the SPI and SD libraries. Next we are going to use the `File` function to define the variable we want to use for the SD card, we will call it `myFile`.

In the setup loop we need to set the bitrate to 9600. Next we will use a while loop to wait until the serial is ready to be used. We will do this by typing `while (!Serial);`. Next we will print `"Initializing SD card..."` to the serial monitor so that we know that it has started. Now we will add an If function for if the SD card does not get initialized. We will do this by typing `if(!SD.begin(3))` and inside the if function we will print `"initialization failed!"` incase it fails and add an infinite while loop so it does not continue. This can be done by typing `while(1);`.

Next we will call the `myFile` function and open/create a text file by typing `myFile = SD.open("test.txt", FILE_WRITE);`. We will then use another If function for if the SD card is available. We will do this by using `if(myFile)`. Inside the If function we will print `"Writing to test.txt..."` on the serial monitor and then write to the SD card. We can do this by using `myFile.println("testing 1, 2, 3.");`. Next we will close the file by typing `myFile.close();`. Then we print `"done."` on the serial monitor. We will then use an Else function for if the SD card can not open the text file. We can do this by printing the error message `"error opening test.txt"` inside the else function.

The last step is to read from the SD card. We can do this by using the `myFile` function and telling it to open the text file by typing `myFile = SD.open("test.txt");`. Then we will add an If function for the `myFile` function and inside of it we will print `"test.txt:"` to the serial monitor. Next we will add a while loop to print all the data from the SD card by doing `while(myFile.available())`. Inside of the while loop we will print to the serial monitor the data by using `Serial.write(myFile.read());`. Then we will close the file again. Lastly we will add another Else function and print `"error opening test.txt"` to the serial monitor.

EXAMPLE:

```
#include <SPI.h>

#include <SD.h>

File myFile;

void setup() {

  Serial.begin(9600);

  while (!Serial);

  Serial.print("Initializing SD card...");

  if (!SD.begin(3)){

    Serial.println("initialization failed!");

    while (1);

  }

  Serial.println("initialization done.");

  myFile = SD.open("test.txt", FILE_WRITE);

  if (myFile){

    Serial.print("Writing to test.txt...");

    myFile.println("testing 1, 2, 3.");

    myFile.close();

    Serial.println("done.");

  }

  else{



    Serial.println("error opening test.txt");

  }

  myFile = SD.open("test.txt");

  if (myFile){

    Serial.println("test.txt:");
```



```
while (myFile.available()) {  
    Serial.write(myFile.read());  
}  
myFile.close();  
}  
else{  
    Serial.println("error opening test.txt");  
}  
}  
void loop() {  
}
```


CODING EXAMPLE 6: BME280 TEST

In this example you will print the data from the weather sensor to the OLED display.

OBJECTIVE: Print weather sensor data to OLED

To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

We will begin by including the Wire, Adafruit_GFX, AdafruitSSD1306, and Adafruit_BME280 libraries. Next define the parameters for the OLED. Next type `Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);` and then `Adafruit_BME280 bme;` Then we will create float variables for the sensor measurements. We will call these `tempData`, `presData`, and `humData`.

Inside the setup loop set the bitrate to 57600. Next define the formatting for the display the same as in previous examples. Then start the I2C bus by using `Wire.begin()`; . Next we will add an If function for if the weather sensor is not detected it will display a message on the OLED. We will do this by typing


```
if(!bme.begin()){  
    display.print("BME failure");  
    display.display();  
    while(1);  
}
```

In the void loop we will begin by taking the measurements from the sensor. For temperature type `tempData = bme.readTemperature()`; . For pressure type `presData = bme.readPressure() / 100.00;` . We are dividing it by 100 to convert it from Pascals to hectopascals. For humidity type `humData = bme.readHumidity()`;


Next we will print the data to the OLED similar to the method for the light sensor. On the first line of the OLED print `"DS Weather Station"`. On the second line print `"Temperature: "` followed by the temperature data by typing `display.print(tempData, 2);` . This rounds it to 2 decimal places. Next print `" C"`. Do this for the pressure and humidity data as well, swapping the words in the strings and the variable makes as necessary. Lastly add a delay of 500 ms.

EXAMPLE:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BME280.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Adafruit_BME280 bme;
float tempData;
float presData;
float humData;
void setup() {
  Serial.begin(57600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.clearDisplay();
  Wire.begin();
  if(!bme.begin()){
    display.print("BME failure");
    display.display();
    while(1);
  }
}
```



```
void loop() {  
  tempData = bme.readTemperature();  
  presData = bme.readPressure() / 100.00;  
  humData = bme.readHumidity();  
  
  display.clearDisplay();  
  display.setCursor(0,0);  
  display.print("DS Weather Station");  
  
  display.setCursor(0, 20);  
  display.print("Temperature: ");  
  display.print(tempData, 2);  
  display.print(" C");  
  
  display.setCursor(0, 30);  
  display.print("Pressure: ");  
  display.print(presData, 2);  
  display.print(" hPa");  
  
  display.setCursor(0, 40);  
  display.print("Humidity: ");  
  display.print(humData, 2);  
  display.print(" %");  
  display.display();  
  
  delay(500);  
}
```



CODING EXAMPLE 7: FULL TEST

In this example you will fully integrate everything you have learned and make the weather sensor, light sensor, LED, OLED display, and microSD card work together.

OBJECTIVE: Fully integrate every component on the weather Station


To get started open Arduino IDE on your computer and open a new sketch by going to File > New. A blank sketch should open which only includes `void setup()` and `void loop()`.

STRUCTURE:

Begin by including the Wire, SPI, SD, Adafruit_GFX, Adafruit_SSD1306, and Adafruit_BME280 libraries. Define the OLED parameters and the LED, sdModule, and light sensor pin numbers. Next type `Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);` and then `Adafruit_BME280 bme;`. Add float variables for the time, temperature, pressure, and humidity. Add an integer variable for the light sensor. Next declare the variable name for the SD card `myFile`.


In the setup loop make the LED an output then set the bitrate to 57600. Next define the formatting for the OLED. Start the I2C bus. Just like in the previous example, add a failure If function for if the weather sensor fails, and make one for the SD card as well. Next define the file name for the SD card as `datalog.csv`. Then add headers for the SD card so you know where it resets. It should look like the lines below.

```
if(myFile){
    myFile.print("-----");
    myFile.print(",");
    myFile.print("-----");
    myFile.print(",");
    myFile.print("-----");
    myFile.print(",");
    myFile.print("-----");
    myFile.print(",");
    myFile.println("-----");
    myFile.close();
}
```



In the void loop start by setting the LED high. Next call the time variable and set it equal to the current time in milliseconds and divide it by 1000.00. It should look like this `timeS = millis() / 1000.00;`. Collect the measurements for the weather sensor and the light sensor like in the previous examples. Open the CSV file by using `myFile = SD.open("datalog.csv", FILE_WRITE);`. Next write the measurements to the SD card. Use the `myFile.print()` function to do this. In-between each measurement print a comma (",") so the CSV file works correctly. On the last measurement make it `myFile.println()` and then close the file. Next print the data to the OLED like in the previous examples. Then set the LED low. Lastly add a 500 ms delay.


EXAMPLE:




```
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BME280.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define LED 0
#define sdModule 3
#define lightSensor A2
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Adafruit_BME280 bme;
```



```
float timeS;
float tempData;
float presData;
```



```
float humData;
int lightData;

File myFile;

void setup() {
  pinMode(LED, OUTPUT);


  Serial.begin(57600);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.clearDisplay();

  Wire.begin();

  if(!bme.begin()){
    display.print("BME failure");
    display.display();
    while(1);
  }
  if (!SD.begin(sdModule)){
    display.print("SD failure");
    display.display();
    while(1);
  }
}
```





```


myFile = SD.open("datalog.csv", FILE_WRITE);
if(myFile){
  myFile.print("-----");
  myFile.print(",");
  myFile.print("-----");
  myFile.print(",");
  myFile.print("-----");
  myFile.print(",");
  myFile.print("-----");
  myFile.print(",");
  myFile.println("-----");
  myFile.close();
}
}


void loop() {
  digitalWrite(LED, HIGH);

  timeS = millis() / 1000.00;
  tempData = bme.readTemperature();
  presData = bme.readPressure() / 100.00;
  humData = bme.readHumidity();
  lightData = analogRead(lightSensor);

  myFile = SD.open("datalog.csv", FILE_WRITE);
  if(myFile){
    myFile.print(timeS, 2);

```









```
myFile.print(",");
myFile.print(tempData, 2);
myFile.print(",");
myFile.print(presData, 2);
myFile.print(",");
myFile.print(humData, 2);
myFile.print(",");
myFile.println(lightData);
myFile.close();
}
display.clearDisplay();
display.setCursor(0,0);
display.print("DS Weather Station");

display.setCursor(0, 20);
display.print("Temperature: ");
display.print(tempData, 2);
display.print(" C");

display.setCursor(0, 30);
display.print("Pressure: ");
display.print(presData, 2);
display.print(" hPa");

display.setCursor(0, 40);
display.print("Humidity: ");
display.print(humData, 2);
```







```
display.print(" %");

display.setCursor(0,50);
display.print("Light Level: ");
display.print(lightData);
display.display();

digitalWrite(LED, LOW);



delay(500);
}
```



4. UPLOADING CODE

There are two main ways to upload code to the Destination Weather Station. The first way is using Arduino IDE and using the upload button. This is the preferred way as the students will be able to write or modify their own code. The second method is uploading precompiled code using the UF2 bootloader built into the Seeeduino XIAO. This method is intended for if the computer is having difficulties uploading the code through Arduino IDE. The downside to using this is that you cannot modify the pre-compiled code. Below are more details for both of these methods.

4.1 ARDUINO IDE

To upload code to the Destination Weather Station using Arduino IDE begin by plugging the weather station into the computer. Next open up the program you want to upload. Click the blue checkmark  to verify the sketch. If the verification fails check the code for errors. Next go to **Tools** and ensure the Seeeduino XIAO is selected under **Board:** and make sure that a COM Port is selected. Next Click upload  to upload the code to the weather station. This process can take some time. The file manager may open during this process. It will automatically close.

4.2 UF2 BOOTLOADER

To upload a .UF2 file to the Destination Weather Station begin by plugging it into the computer. Next double click the Reset button on the weather station. This will open up the file manager. Next drag the desired .UF2 code into the folder just like on a flash drive that just opened. The file manager will then close automatically.

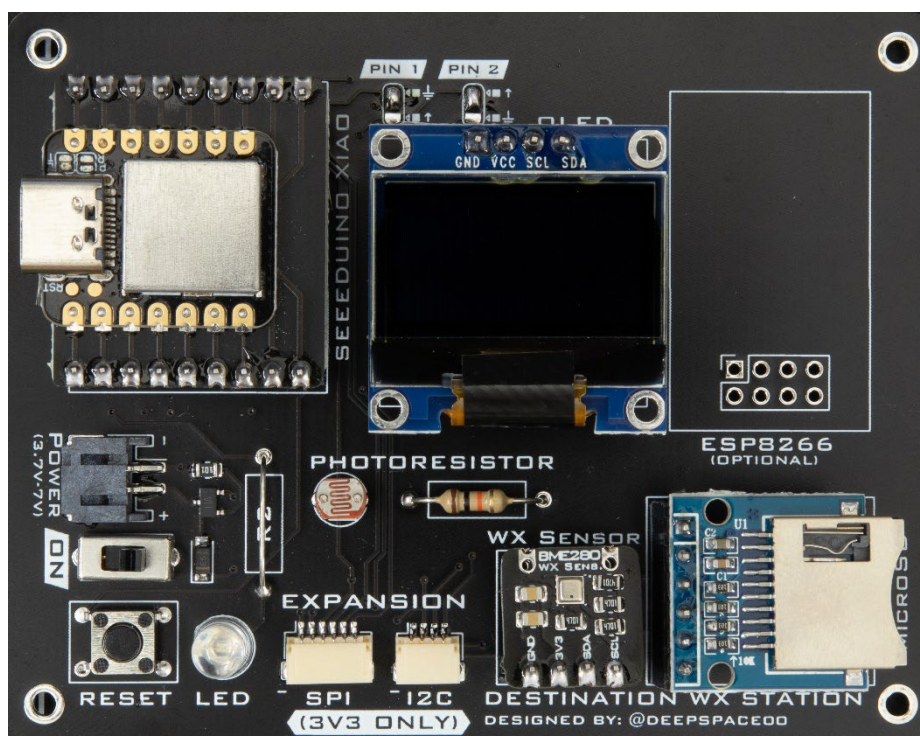


SOUTH CAROLINA GOVERNOR'S SCHOOL FOR
**SCIENCE+
MATHEMATICS**



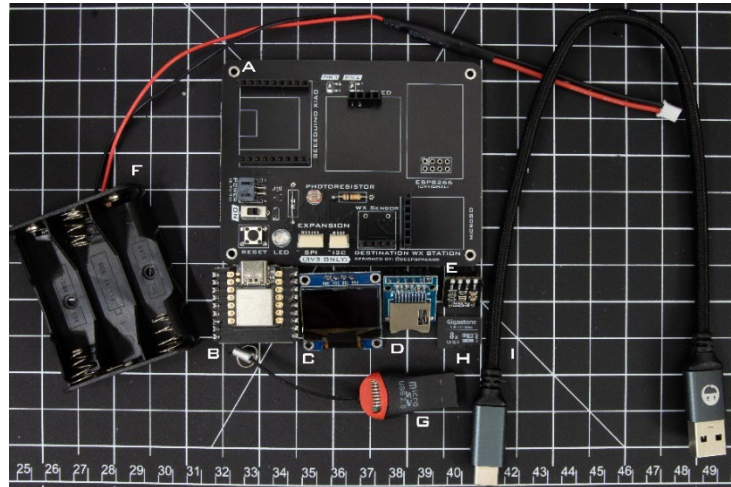
DESTINATION WEATHER STATION REVISION 3 (SOLDERLESS)

ASSEMBLY INSTRUCTIONS



STEP 1:

LAYOUT ALL PARTS IN THE KIT AS SHOWN BELOW



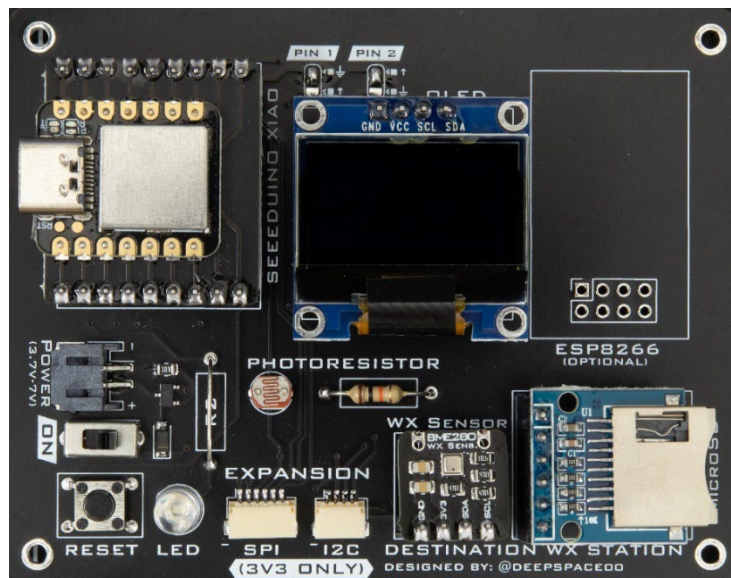
A. CIRCUIT BOARD
B. SEEDUINO XIAO
C. OLED DISPLAY

D. MICROSD MODULE
E. WEATHER SENSOR
F. BATTERY PACK

G. MICROSD ADAPTER
H. MICROSD CARD
I. USB-C CABLE

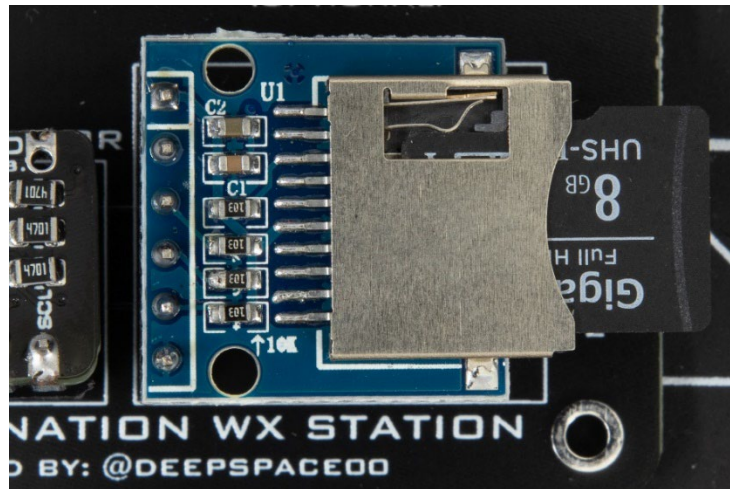
STEP 2:

CONNECT THE SEEDUINO XIAO, OLED DISPLAY, WEATHER SENSOR, AND THE MICROSD MODULE TO THE CIRCUIT BOARD. YOUR WEATHER STATION SHOULD LOOK LIKE THE IMAGE BELOW.



STEP 3:

INSERT THE MICROSD CARD INTO THE MICROSD MODULE WITH THE WORDS FACING UP AND THE GOLD CONTACTS GOING IN. THE MICROSD CARD WILL NOT GO IN ALL THE WAY. IT WILL ONLY GO IN AS FAR AS THE IMAGE SHOWN BELOW.

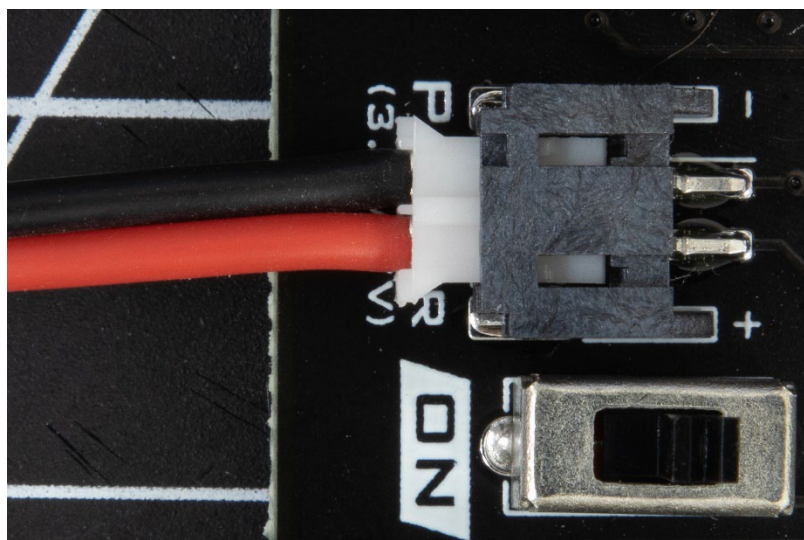



TROUBLESHOOTING TIPS:

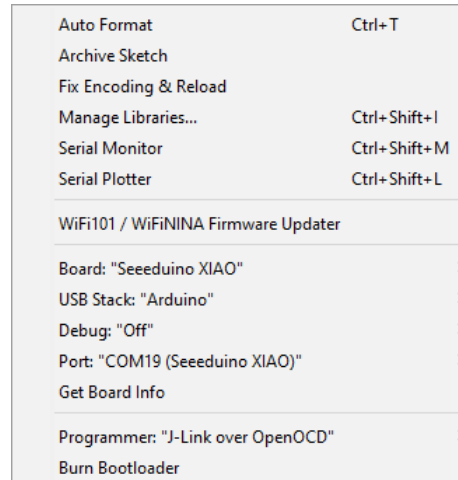
1. MAKE SURE THAT THE MICROSD CARD IS PLUGGED INTO THE WEATHER STATION BEFORE TRYING TO COLLECT DATA
2. WHEN INSERTING THE MICROSD CARD INTO THE MICROSD CARD ADAPTER IT WILL ONLY GO IN AS FAR AS THE IMAGE SHOWN BELOW. IF YOU ARE HAVING DIFFICULTIES INSERTING IT, TRY ROCKING IT SIDE TO SIDE AND INSERTING IT AGAIN.



3. WHEN PLUGGING IN THE BATTERY PACK INTO THE WEATHER STATION, THE CONNECTOR SHOULD ONLY FIT ONE WAY. THE LITTLE BUMP IN THE MIDDLE SHOULD BE FACING UP. REFER TO THE IMAGE BELOW.



- 
4. THE POWER SWITCH ONLY WORKS WHEN THE WEATHER STATION IS NOT PLUGGED INTO THE COMPUTER.
5. WHEN UPLOADING CODE TO THE WEATHER STATION MAKE SURE THAT THE FOLLOWING PARAMETERS ARE SIMILAR TO THE ONES BELOW (NOTE: YOUR COM PORT MAY BE DIFFERENT FROM THE ONE BELOW)



6. IF **SEEDUINO XIAO** IS NOT SHOWING UP, GO TO **FILE > PREFERENCES** AND ENSURE THE BOARD MANAGER URL IS ENTERED AND THE **VERIFY CODE AFTER UPLOAD** BUTTON IS UNCHECKED



SOUTH CAROLINA GOVERNOR'S SCHOOL FOR
SCIENCE+
MATHEMATICS



WEATHER STATION TEACHER CHECKLIST

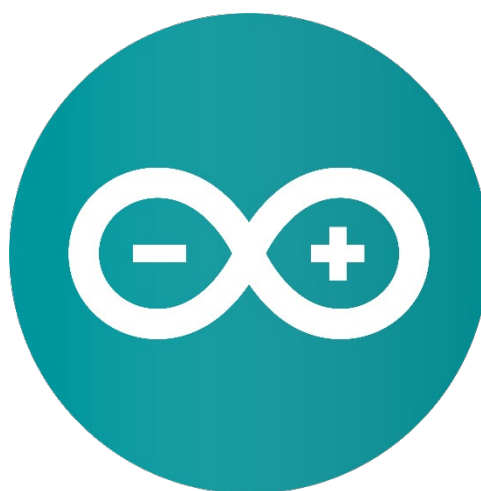
1. Assemble Weather station with students. Make sure parts are oriented correctly.
2. Have students open Arduino IDE on their computers. Have students open up File > Preferences. Make sure the check for updates box is **unchecked** and the board manager URL is entered
3. Have students go to Tools and make sure that the Seeed SAMD boards are an option
4. Have students complete Hello World example and upload Full Test code
5. Make sure weather stations work with Full Test code
6. Collect data outside



SOUTH CAROLINA GOVERNOR'S SCHOOL FOR
**SCIENCE+
MATHEMATICS**



DESTINATION SPACE SOFTWARE SETUP ARDUINO IDE

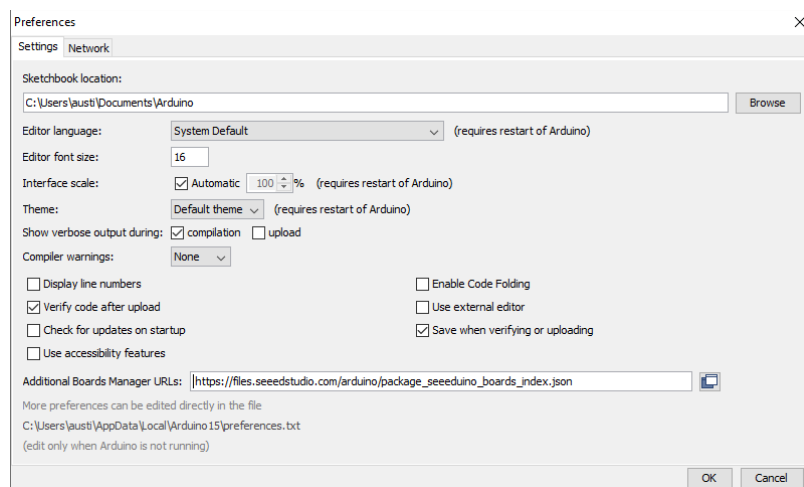


STEP 1:

DOWNLOAD THE LATEST VERSION OF ARDUINO IDE FROM
<https://www.arduino.cc/en/software>

STEP 2:

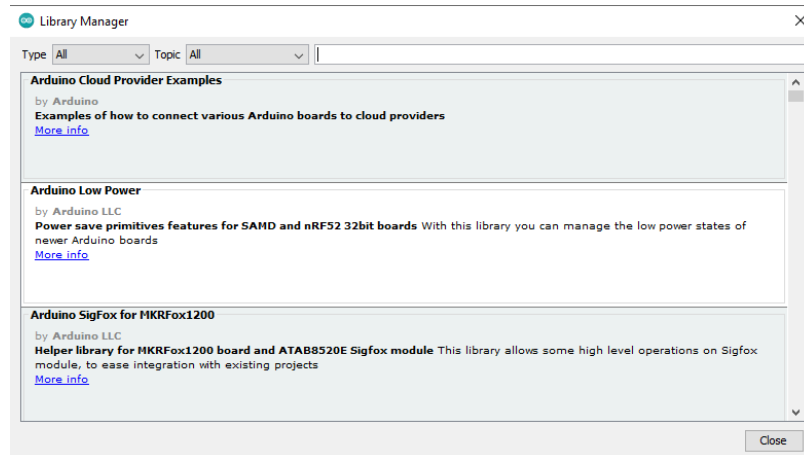
- OPEN ARDUINO IDE
- GO TO **FILE > PREFERENCES**
- UNCHECK THE **CHECK FOR UPDATES ON STARTUP** BUTTON
- IN THE TEXT BOX NEXT TO **ADDITIONAL BOARD MANAGER URLS** COPY AND PASTE THE FOLLOWING URL AND CLICK THE **OK** BUTTON
https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json



- WAIT FOR EVERYTHING TO FINALIZE
- GO TO **TOOLS > BOARD: > BOARD MANAGER...**
- IN THE SEARCH BOX ENTER **SEEED SAMD BOARDS**
- CLICK **INSTALL**

STEP 3:

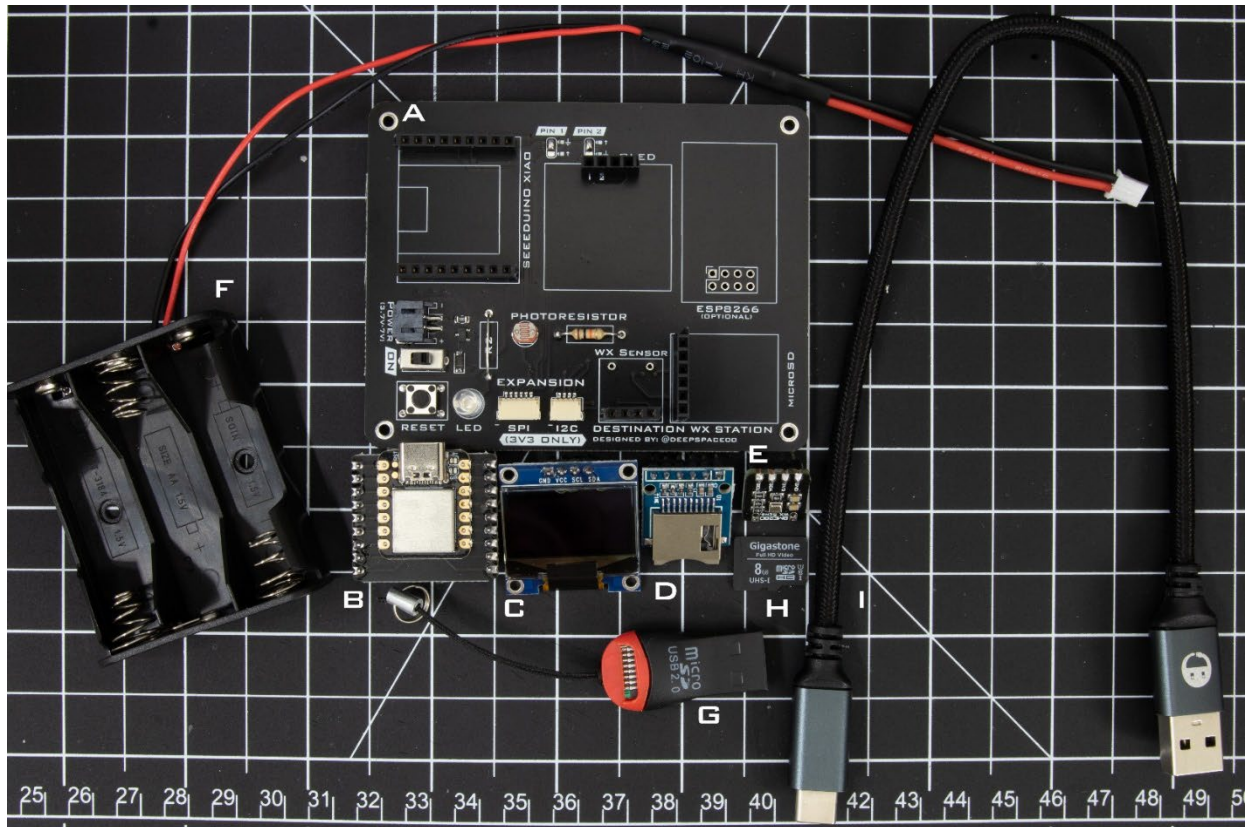
- GO TO **SKETCH > INCLUDE LIBRARIES > MANAGE LIBRARIES...**



- IN THE SEARCH BOX ENTER **ADAFRUIT SSD1306**
- **CLICK INSTALL**
- IN THE SEARCH BOX ENTER **ADAFRUIT GFX**
- **CLICK INSTALL**
- IN THE SEARCH BOX ENTER **ADAFRUIT BME280 LIBRARY**
- **CLICK INSTALL**
- IN THE SEARCH BOX ENTER **ADAFRUIT UNIFIED SENSOR**
- **CLICK INSTALL**

KIT COMPONENTS

Make sure each kit contains the following before packing up



- A. CIRCUIT BOARD
- B. SEEDUINO XIAO
- C. OLED DISPLAY
- D. MICROSD MODULE
- E. WEATHER SENSOR
- F. BATTERY PACK
- G. MICROSD ADAPTER
- H. MICROSD CARD
- I. USB-C CABLE
- J. 3X AA BATTERIES