How to Make a Sea Level rise and Climate Data Logger

By: Matthew Freeman

Climate change has altered the environment we live in from global temperatures to sea levels rising. Tracking the levels in which this is happening is very important if we wish to target and combat the effects. The problem that arises is that traditional data loggers designed to track various weather conditions and tide levels are expensive. This means that less sensors are able to be implemented and less data is collected from affected areas. This lack of data has large, long-term implications on our knowledge of what is happening. This project aims to make cheap and simple data loggers that can help solve this gap in data collection.

Components

DHT 22 (Temperature and Humidity Sensor), Grove-Light Sensor, Grove-Digital Switch(P), Grove-Ultrasonic Distance Sensor, Grove-LED (Blue and Green), Anemometer, Arduino Uno, SD shield, Grove-Base Sheild

Measures

Light levels, temperature, humidity levels, distance of tides, wind speed

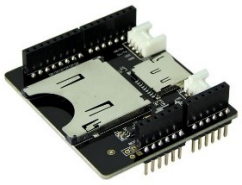
Highlights

Cost effective casing and parts
Minimal skills with soldering required
Power tools aren’t required but make it easier to make
Software is open-source and well documented for user convenience

Supplies

1. Arduino Uno Price: $27.60
   - https://www.seeedstudio.com/Arduino-Uno-Rev3-p-2995.html ---

2. SD Card Shield Price $13.90
   - https://www.seeedstudio.com/SD-card-shield-p-492.html

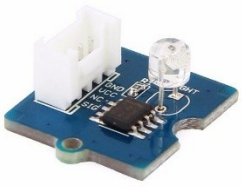


   -
3. Grove-Base Sheild Price $3.50
   - https://www.seeedstudio.com/Base-Shield-V2.html

   -
4. DHT22 Price: $6.95
   - https://www.seeedstudio.com/Grove-Temperature-Humidity-Sensor-Pro-AM2302-DHT22.html

   -
5. Grove-Light Sensor Price: $1.99

- https://www.seeedstudio.com/Grove-Light-Sensor-v1-2-LS06-S-phototransistor.html



-

6. Grove-Ultrasonic Distance Sensor Price: $3.95
   - https://www.seeedstudio.com/Grove-Ultrasonic-Distance-Sensor.html

-

7. Grove LED Price: $2.10
   - https://www.seeedstudio.com/Grove-Green-LED.html

-

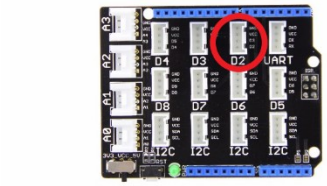
8. Anemometer Price: $32.29
   - https://www.amazon.com/Monitoring-Anemometer-Wind-Measuring-Replacement-Meteorological/dp/B0CWNC9ZYQ/ref=sims_dp_d_dex_ai_speed_loc_mtl_v5_t1_d_sccl_3_1/147-3067127-2747617?pd_rd_w=LZYxc&content-id=amzn1.sym.281550a9-05fa-4fa0-a033-b1923adca8ef&pf_rd_p=281550a9-05fa-4fa0-a033-b1923adca8ef&pf_rd_r=PY7AQ5D2RGKEJNZ57PAT&pd_rd_wg=y9eCr&pd_rd_r=7d6f98d6-f2b1-48ea-b86a-0680e318dad1&pd_rd_i=B0CWNC9ZYQ&psc=1

-

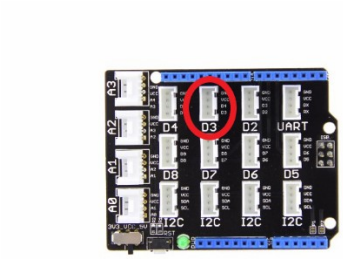
9. Grove - Universal 4 Pin Buckled 50cm Cable (5 PCs Pack) Price: $3.50
    - https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-50cm-Cable-5-PCs-Pack.html
    - 

Assembly

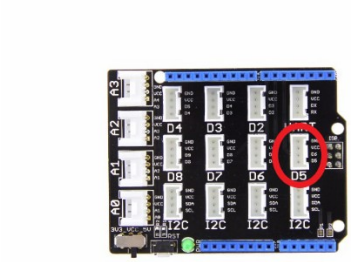
1) Place SD Shield on the Arduino UNO ensuring all pins are pushed in
2) Place Grove-Base Shield on-top of the SD Shield ensuring all pins are pushed in
    a. Note: The three boards should now be stacked on top of each other
3) Using a Grove 4 Pin Wire, attach the DHT22 sensor to port 2 on the Grove-Base Shield



4) Using a Grove 4 Pin Wire, attach the Grove-Digital Switch(P) to port 3 on the Grove-Base Shield

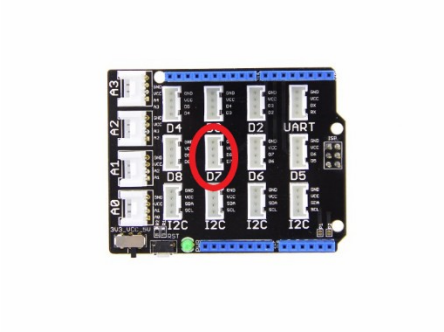


5) Using a Grove 4 Pin Wire, attach the Grove-Distance Sensor to port 5 on the Grove-Base Shield



6) Using a Grove 4 Pin Wire, attach the Grove-LED (Blue) to port 6 on the Grove-Base Shield

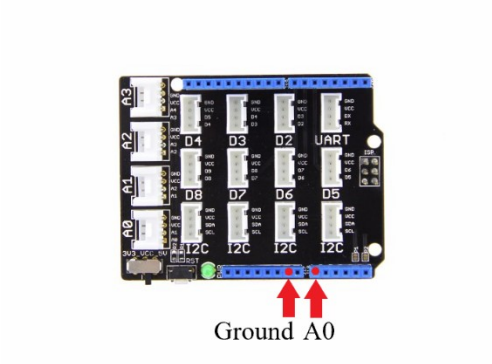
7) Using a Grove 4 Pin Wire, attach the Grove-LED (Green) to port 7 on the Grove-Base Shield



   a. *Note: for LED modules from grove, the orange screw piece can be rotated to dim or brighten the LED

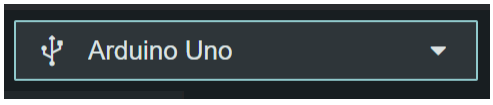
8) Using a Grove 4 Pin Wire, attach the Grove-Light Sensor to port A1 on the Grove-Base Shield

9) For the Anemometer, some steps are required to prepare the sensor for use.
    a. Using wire cutters, cut the Anemometer's wire to the desired length
    b. Strip the outer black coating ~2 inches to expose a black wire and a red wire
    c. Strip both wires ~0.5 inches
    d. Twist both wires separately to connect the individual wires into one solid wire
    e. Locate the pin slots on the Arduino UNO (located below the slots labeled I2C
    f. Plug the red wire into the slot labeled A0
    g. Plug the black wire into the slot labeled GND



10) Use the wire provided with the Arduino UNO to connect it to your computer
11) Download ARDUINO IDE for uploading and modifying code
    a. The following link is for downloading ARDUINO IDE: https://www.arduino.cc/en/software
    b.

12) Download the attached code
13) After opening the code in ARDUINO IDE, in the top left corner, select the ARDUINO UNO board



14) To the left of the board selection, press the arrow facing the right to upload the code to the board. 
15) Happy data logging!!!

Code

Pre-Setup

```
1  #include "Grove_Temperature_And_Humidity_Sensor.h"          //Temperature and Humidity library
2  #include "Ultrasonic.h"                                    //Ultrasonic Ranger library
3  #include <SD.h>                                            //SD library
```

These three lines of code include the libraries necessary to run different sensors from the data logger. They contain extra code and instructions for the sensor that allows them to run properly on your Arduino device without having to write it all out yourself.

```
8  const int chipSelect = 4;                                  //SD pin
9  int led[] = {6, 7};                                        //LED Order: Blue, Green
10 int anne = A0; int light = A1;                             //Annemometer pin and Light Sensor pin
11 int readingMinutes = 1; int readingTime;                   //changing reading time from minute to milliseconds
12 int readTime = readingMinutes * 60000;                     //Converting read time from minutes to milliseconds
13 int debug = 50;                                            //Debug delay time
14 int lightReading; int lightPercent;                        //Light sensor variables
15 int windCount; int windSpeed;                              //Annemometer variables
16
17 File myFile;                                               //Creating file state for SD
18
19 Ultrasonic ultrasonic(5);                                  //Range finder pin
20 long RangeInCenti;                                        //Range in centimeters for ultasonic ranger
21
22 DHT dht(DHTPIN,DHTTYPE);                                  //DHT Setup
23 float tempval;                                            //Decimal for temperature
24 float humidity;                                           //Decimal for humidity value
```

This line of code identifies the key integers used in the code. Lines 8, 9, 10, and 19 all identify the pin numbers that each sensor is attached to on the Grove-Base shield. The remaining integers are variables that are used when data logging is actually occurring. Each integer has a descriptive name given to it. The sensors will assign a number to these integers whenever data logging is occurring. We can then call on these names afterwards to print to an excel file and they will print the value given to them to the file. Line 11 and 12 are used to determine how much time needs to pass between readings.

```
26 void blink(int blinkNumber, int ledNumber) {               //Function for blinking LEDs for diagnostic purposes
27   digitalWrite(led[0],LOW);
28   for(int i=0; i < blinkNumber; i++) {                     //count up from zero to number of blink times
29     digitalWrite(led[0], LOW);                             //Turns of diagnostic LED
30     digitalWrite(led[ledNumber], HIGH); delay(1000);       //Turns LED on set amount of times
31     digitalWrite(led[ledNumber],LOW); delay(1000);         //Turns LED off set amount of times
32   }
33 }
```

This is a function to identify if any errors occur at any time with the data logging device. The blue LED will blink a certain amount of times depending on what the error is so that we can visually see where the problem is without having to unhook everything.

Setup

```
35  void setup() {
36    Serial.begin(9600);                                   //Initialize Serial
37      while(!Serial);                                     //Hold until Serial begins
38
39    for (int p = 0; p < 3; p++) {                         //Initialize LED pins
40      pinMode(led[p], OUTPUT);                            //Sets LED pins as outputs
41    } delay(debug);                                       //Debounce delay
42
43    pinMode(anne,OUTPUT); delay(debug);                   //Annemometer initialization
44    dht.begin(); delay(debug);                            //Initialize Temp and Humidity sensor
45    pinMode(light,INPUT); delay(debug);                   //Initialize Light Sensor
```

This is the beginning of the setup process. The setup process is used to make sure that all processes necessary for data logging are running as well as define to the system what each sensor is supposed to do. For example, lines 39-41 define the LEDs as being outputs. This means that the system knows that we intend to send power to them whenever we need to use them. Lines 43-45 define the various sensors that are utilized on the data logger.

```
47    Serial.print("Initializing SD card...");              //Print to serial that SD is starting
48    digitalWrite(led[0],HIGH);                            //Blue LED means that setup is beginning for device
49      delay(debug);                                       //Debounce delay
50    if (!SD.begin(chipSelect)) {                          //Starting SD
51      Serial.println("initialization failed.");           //Print to serial that SD failed
52      blink(2,0);                                         //2 Blue blinks for SD card startup failure
53      while(!SD.begin(chipSelect));
54    }
55    Serial.println("initialization done."); delay(debug); //Print to serial that SD starup is done
```

This section of code is used to initialize the attached SD card. The code will not continue if the SD card isn't working to ensure that we will be able to save any data we take.

```
57  myFile = SD.open("sealevel.csv", FILE_WRITE); delay(debug);     //creates file named sea_level.csv meaning its a commas deleted excel
58
59  if (myFile) {
60    Serial.println("Initializing Headers");                       //Print to serial that headers are being made
61      myFile.print("time_min,"); myFile.print("temp,"); myFile.print("humidity_percent,");
62      myFile.print("wind_count,"); myFile.print("wind_speed_mph,"); myFile.print("light_raw,"); myFile.print("light_percent,");
63      myFile.println("tide_distance_cm,"); myFile.close();
64  delay(debug);                                                   //Debounce delay
65    Serial.println("Headers written.");                           //print to serial if the file was composed correctly
66  } else {
67    Serial.println("Failed to open file for writing headers.");   //Serial print if file is writing properly
68      blink(4,0);                                                 //4 Blue blinks for file creation errors
69      while(!myFile);
70  }
71  delay(debug);                                                   //Debounce delay
72    Serial.println("Setup is complete");                          //Print to serial that everything is completed
73      digitalWrite(led[0],LOW);                                   //Turn off Blue LED to show setup has stopped
74      digitalWrite(led[1],HIGH);                                  //Turn on Green LED to show everthing is ready
75      delay(5000);                                                //Delay 5 seconds
76      digitalWrite(led[1],LOW);                                   //Turn off Green LED to save power
77  delay(debug);                                                   //Debounce delay
78  }
```

This section of code creates a file on the SD card for us to save all of our data too. Line 57 creates the files name. Line 61-63 labels the headings for each file. Line 67-69 will give an error code if the files are not created and labeled properly. Line 72-76 will only run if everything in setup has occurred correctly. The blue LED will turn off and the Green LED will turn on for 5 seconds before any readings will take place.

Loop

```
80  void loop() {
81
82    delay(readTime);
83
84    readingTime += readingMinutes;                                //Increases the reading time by however long the reading minutes was set to
85
86    lightReading = analogAverage(10, light);                      //function to read light average
87    lightPercent = map(lightReading, 0, 800, 0, 100);             //Maps light level to a percentage based system
88
89    windCount = analogAverage(10, anne);                          //Averages the count of 10 windspeed readings
90    windSpeed = windCount/2.2993;                                 //Calibration to convert counts from annemometer to windspeed in MPH
91
92    RangeInCenti = ultrasonic.MeasureInCentimeters();             //Reads ultrasonic sensors and gives a distance
93
94    tempval = averageDHT(10, dht.readTemperature());              //Function to read average temp
95    humidity = averageDHT(10, dht.readHumidity());                //Function to read average humidity
96
97    check();                                                      //checks to make sure sensors are working properly
98
```

This code represents the beginning of the loop. The loop is what will continuously run over and over again until the device is turned off. Line 82 will wait for a certain amount of time specified in the pre-setup before readings take place. After that time is up, each sensor will take a reading, and store its value to the integers we defined in the pre-setup. The readings, like on line 86, 89, 94, 95,and 97, do rely on custom functions which are shown later on in the additional functions section.

```
99   myFile = SD.open("sealevel.csv", FILE_WRITE);                //creates file named sea_level.csv meaning its a commas deleted excel
100
101  if (myFile) {                                                //Adds to myFile if open
102    myFile.print(readingTime); myFile.print(","); myFile.print(tempval); myFile.print(","); myFile.print(humidity); myFile.print(",");
103    myFile.print(windCount); myFile.print(","); myFile.print(windSpeed); myFile.print(","); myFile.print(lightReading); myFile.print(",");
104    myFile.print(lightPercent); myFile.print(","); myFile.println(RangeInCenti); myFile.close();
105    Serial.println("Data Recorded");
106  }
107  blink(1,1);                                                  //Blinks green once to show data has been taken
108 }
```

This part of the loop is where we store all of the data that we collect. Line 99 opens the file we previously made. Lines 102-105, store the values for each sensor that was just collected. A comma is also printed between each value as that is used to define that the next value needs to be placed in the next column in the excel spreadsheet.

Additional Functions

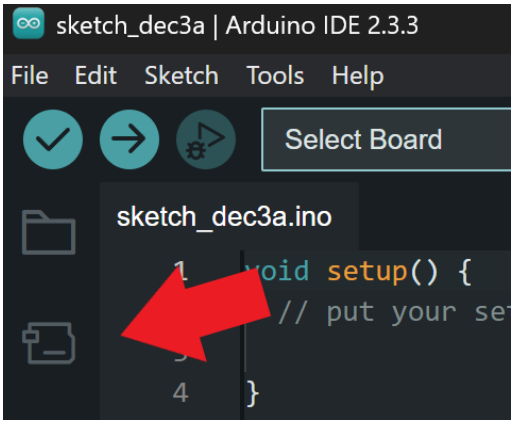
```
110  float averageDHT(int numReadings, int sensor) {              //Average Temp and Humidity Readings
111    float total = 0.0;                                         //Reset total to 0.0
112    for(int i=0; i<numReadings; i++){                          //Count up from 0 to 1 below read count
113      total += sensor;                                         //Read Sensor
114      delay(debug);                                            //Delay between readings
115    } delay(debug);                                            //Debounce delay
116    return total/numReadings;                                  //Return Avereged Humidity Readings
117  }
118
119  int analogAverage(int numReadings, int sensor) {             //Average Analog Readings
120    int total = 0;                                             //Reset total to 0
121    for(int i=0; i<numReadings; i++)  {                        //Count up from 0 to 1 below read count
122      total += analogRead(sensor);                             //Read Light Sensor
123      delay(debug);                                            //Delay between readings
124    } delay(debug);                                            //Debounce delay
125    return total/numReadings;                                  //Return Averaged Light Readings
126  }
```

Lines 110-117 are a custom function used for the DHT sensor. The number of times you want a reading to take place and from either the temperature or humidity sensor can be specified. The values from each reading are then averaged together and then returned as the new value for whatever integer you assign the function to. Lines 119-126 are a custom function to average the readings from analog sensors. The same logic from the first averaging function applies here as well.

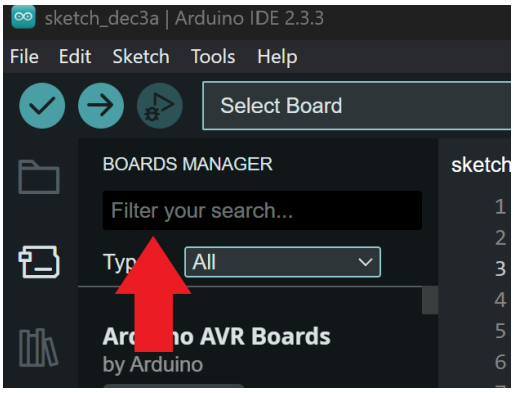
```
void check() {                                  //Checks to see if the Light sensor is reading properly
  if (lightReading < 0 || lightReading > 1023)  {   //Check light sensor value
    lightReading = -1; lightPercent = -1;       //Change light values to -1 to show error
  } delay(debug);                               //Debounce delay
  if (windCount < 0 || windCount > 1023)  {     //Check wind sensor value
    windCount = -2; windSpeed = -2;             //Change wind values to -1 to show error
  } delay(debug);                               //Debounce delay
  if (RangeInCenti < 0 || RangeInCenti > 400) {     //Check distance value
    RangeInCenti = -3;                          //Change distance value to -1 to show error
  } delay(debug);                               //Debounce delay
  if (humidity < 0 || humidity > 100)  {        //Check humidity value
    humidity = -4.0;                            //Change humidity value to -1 to show error
  } delay(debug);                               //Debounce delay
  if (tempval < -40 || tempval > 80)  {         //Check temperature value
    tempval = -5.0;                             //Change temperature value to -1 to show error
  } delay(debug);                               //Debounce delay
  return;                                       //Returns from check
}

```

This function is used to determine that all sensors are working and if not changes their sensor value to a value that isn't normally possible so it can be easily seen in the excel file. Each sensor is given a unique number so that it can be easily identified. The light sensor is -1, the wind speed sensor is -2, the distance sensor is -3, the humidity sensor is -4, and the temperature sensor is -5.

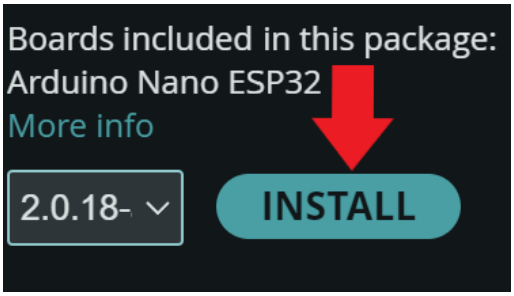
Installing Libraries



To start click on the library install tab as see here.



Then search for the libraries that you want to install.



Then click the install button and you're done.

The libraries we used here are as follows:

- Grove_Temperature_And_Humidity_Sensor.h
- Ultrasonic.h
- SD.h

Testing

Once you have completed assembling the data logger, it is important to test it before moving on to designing a housing for it and implementing it in the field. To do so, plug the Arduino into your computer and allow it to run for a couple minutes to allow multiple data points to be taken. I suggest lowering the reading time to 1 minute and then running the logger for ten minutes. Afterwards, unplug the device and check the excel file on the SD card. Ensure that none of the sensors are giving a negative number error code which would mean they are not working properly. If they are giving a negative number, first ensure that all wires are attached correctly. If they are looking to see if the sensor has any visible damage.

With this excel file of initial results, you can also test to see if the data logger sensor are accurate. If you have another method of measuring the different data points, like a separate thermometer, etc., you can then change the data points from the data logger to be more accurate.

Once these initial test are done, you can replace the SD card in the data logger and begin actually taking data. For deploying the logger in a field, the most important aspects are powering it and keeping it watertight. For powering, I suggest using a 9v battery as they are cheap and easy to implement with the Arduino unit using the included 9v battery wire. For the device housing, cheap plastic Tupperware containers work well for housing. Try to get ones that market themselves as waterproof. These will typically have a rubber ring around where the lid goes to keep an airtight seal. Holes can be drilled in the container to allow the sensors to come out and silicone can be used to cover up those holes afterwards to maintain that waterproof effect.

Congratulations, you’re now ready to start data logging. This device measure wind speed, temperature, humidity, light levels, and distance. This makes it a really good device for measuring the change in tides for local waters. Happy logging!