**Build With Us: IoT to Keep Your Plants Alive!**

Presenter: Donna Gavin

Based off a presentation by Ivy Malao, Star Washington, Yanning Qu

**Overview**

Tables will work together to build a simple device that tells you when to water a plant. Wiring together and writing code for an Arduino, LEDs, moisture sensor, you will be able to use your device in different potted plants to determine which are over or under-watered.

These instructions are divided into **three separate modules**: setting up the **Arduino**, working with the **LEDs** and working with the **moisture sensor**. The **last step** is to figure out how to use the moisture sensor readings to control the LEDs.

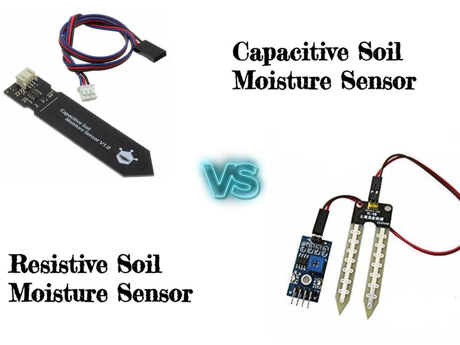
**Materials**

* Arduino Uno
* Half breadboard
* Adafruit I2C Capacitive Moisture Sensor
* 4-pin JST-PH cable
* LEDs
* Resistors
* USB-B cable
* Jumper wires

Moisture sensors include two styles: resistive sensors and capacitive sensors.

Resistive style sensors use two probes to pass current through the soil and read the resistance to get the moisture level. Wet soil causes the soil to conduct electricity better (less resistance). Corrosion of the sensor probes limits the lifespan of a resistive sensor.

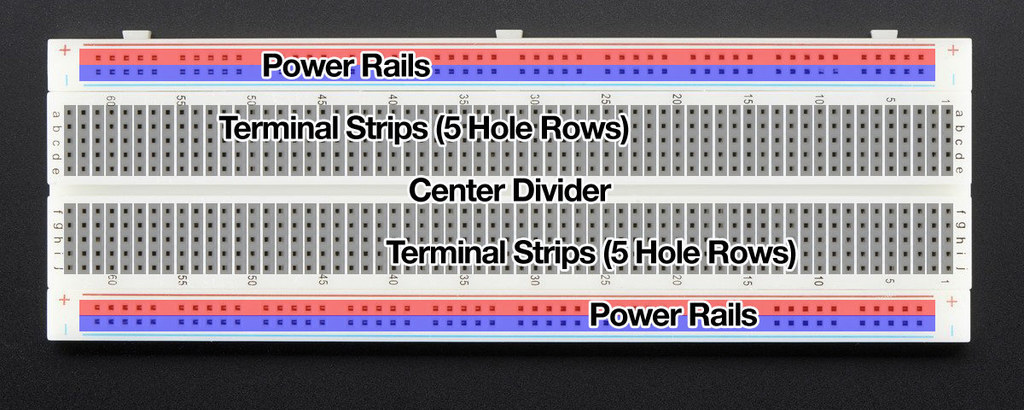
Capacitive moisture sensors use only one probe and do not have any exposed metal. They work by measuring the “dielectric” that is formed by the soil. Water is the most important factor that affects the dielectric and the sensor measures specifically the dissolved ions (electrically charged molecules). **Capacitive moisture sensors will give you a reading ranging from 200 (very dry) to 2000 (very wet).**

[](https://github.com/imalao/ghc2019_workshop/blob/master/imgs/moisture_sensors.png)

Just about every breadboard is made of three sections: Two sets of very long power rails and a large middle section that is full of 5-hole-long terminal strips.

The power rails are used to distribute the power and ground connections along the entire circuit. To help keep track of which rail is ground and which is power, connect positive / power to red (+) and connect ground to blue (-).

Put the components in the middle section, with each pin connected to a rows terminal strip.

[](https://github.com/imalao/ghc2019_workshop/blob/master/imgs/breadboard.jpeg)

**Instructions**

**Hardware and Arduino Simulator**

As each table is provided with only one set of hardware components, you can optionally use [Tinkercad](https://www.tinkercad.com/) to simulate the Arduino and hardware.

**(Module 1) Setting up the Arduino**

* Open up Arduino from the start menu.
* Click Files 🡪 Examples 🡪 Basic 🡪 Blink to open that file. See the code below.

// the setup function runs once when you press reset or power the board

void setup()

{

// initialize digital pin LED\_BUILTIN as an output.

pinMode(LED\_BUILTIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop()

{

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

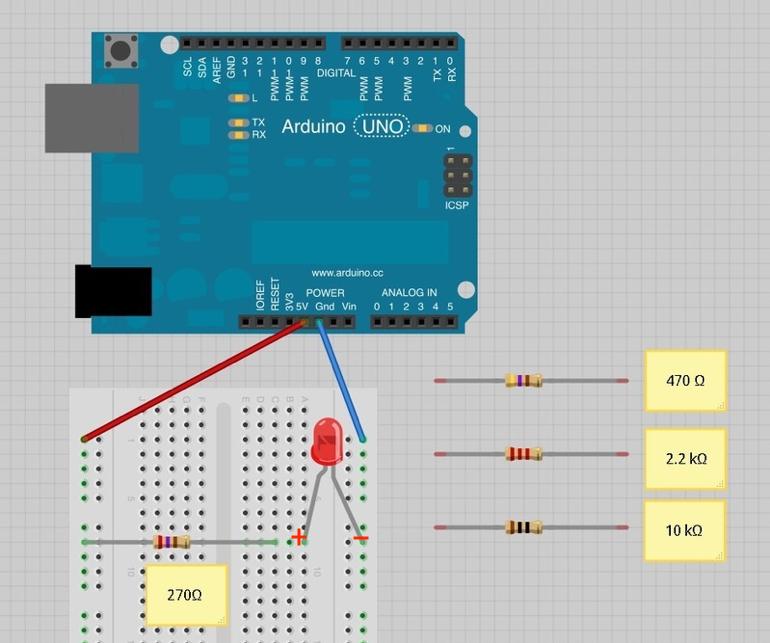
digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

* Connect your Arduino to your laptop using the USB-B cable.
* Click on Tools 🡪 Board 🡪 Arduino Uno
* Click on Tools 🡪 Port 🡪 COMx (Arduino Uno), where x will be a number.
* **Upload and save** your code by clicking the arrow at the top
* Your Arduino's light should now be flashing!

**(Module 2) LEDs**

[](https://github.com/imalao/ghc2019_workshop/blob/master/imgs/led_pinout.png)

* Connect your stripboard as shown above using the jumper wires, any resistor, and LED. The Arduino will provide power if you plug it in with the USB-B cable. LEDs have a positive and negative leads and will not light if they are the wrong way around. The positive lead is longer.
* With the 270 Ohm resistor, the LED should be quite bright. You can use any of the provided resistors.
* Now attach the LED to a digital output pin of the Arduino so that you can control the LED with code. Move the jumper wire connected to power (the 5V pin) to one of the digital pins (e.g. 13). This should be the red wire in the diagram.
* Blink the LEDs using the following code sample – simply add the line that is in bold in your blink code, as shown on the next page.

/\*

Blink

Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.

\*/

// Add this code – this represents the digital pin connected to your LED

**int led = 13;**

// the setup routine runs once when you press reset:

void setup()

{

// initialize the digital pin as an output.

pinMode(led, OUTPUT);

}

// the loop routine runs over and over again forever:

void loop()

{

// initialize the digital pin as an output.

digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(led, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

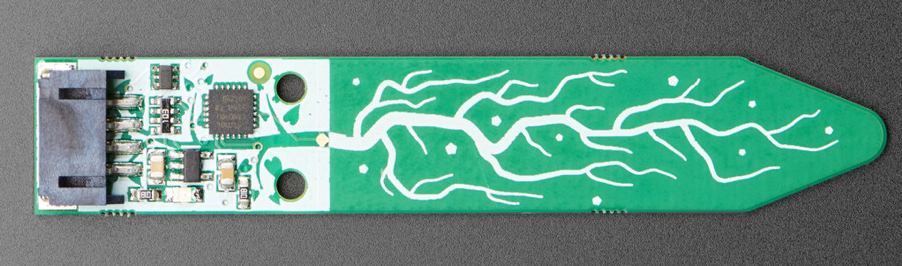
}

* Connect your Arduino to your laptop using the USB-B cable.
* **Upload and save** your code by clicking the arrow at the top.
* Your LED should now be flashing!

**(Module 3) Moisture Sensor**

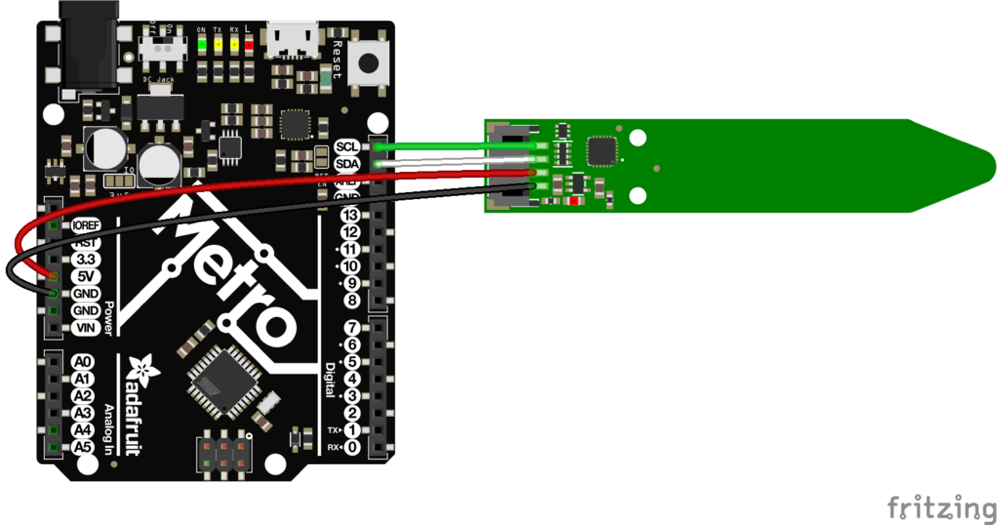
**Hardware**

Connect the moisture sensor to the Arduino using the 4-pin JST-PH cable. JST is just the manufacturer’s name and PH is the series of cable.

[](https://github.com/imalao/ghc2019_workshop/blob/master/imgs/adafruit_moisture_sensor.png)

Looking at the image above, from top to bottom, the pinout is

1. **I2C SCL**: Clock Line used to synchronize all data transfers over the I2C bus.
2. **I2C SDA**: Data Line
3. **Vin**: 3-5V DC. Has a regulator which supplies 5V to the 5V pin/node
4. **GND**: power and logic ground

[](https://github.com/imalao/ghc2019_workshop/blob/master/imgs/moisture_sensor_pinout.png)

* Connect **Vin Red Wire to the power supply**. Use the same voltage that the microcontroller logic is based off of. For most Arduino's, that is 5V.
* Connect **GND Black Wire to common power/data ground**
* Connect the **SCL Green Wire pin to the I2C clock SCL pin** on your Arduino. On an UNO Arduino, this is also known as A5.
* Connect the **SDA White Wire pin to the I2C data SDA pin** on your Arduino. On an UNO Arduino, this is also known as A4.

**Software**

* After setting up your Arduino (Module 1), install the Adafruit libraries
  + Click on **Libraries** on the left bar
  + Click on **Library Manager**, the blue button on the top.
  + Search for **Adafruit\_seesaw** and click on Install to download the library. Choose Include Adafruit Seesaw Library.
  + Go to **Favorites**, mouse over the Adafruit Seesaw Library, and click **Include** to include the Adafruit seesaw libraries.
* Go to this link to download the file: <http://bit.ly/3Ybkmax>

Adafruit\_seesaw ss;

// to communicate with the computer, use one of the following rates and set the serial\_baud // variable to the value:

// 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.

uint16\_t serial\_baud = 0; // Set the serial\_baud rate

uint16\_t seesaw\_I2C\_address = 0x36; // Adafruit seesaw I2C address

// the setup routine runs once when you press reset:

void setup()

{

// initialize the digital pin as an output.

Serial.begin(serial\_baud);

Serial.println("Seesaw Soil Sensor example");

if (!ss.begin(seesaw\_I2C\_address))

{

Serial.println("ERROR! seesaw not found");

while (1);

}

else

{

Serial.print("Seesaw started! version: ");

Serial.println(ss.getVersion(), HEX);

}

}

// the loop routine runs over and over again forever:

void loop()

{

float tempC = ss.getTemp(); // temperature reading

uint16\_t capread = ss.touchRead(0); // capacitive reading

Serial.print("Temperature: "); Serial.print(tempC);

Serial.println("\*C");

Serial.print("Capacitive: "); Serial.println(capread);

delay(1000); // wait for a second

}

* Connect your Arduino to your laptop using the USB-B cable.
* **Upload and save** your code by clicking the arrow at the top. If
* Click **Tools 🡪Serial Monitor** to read the serial console output.
* Open up the serial console at the baud rate you set in serial\_baud to see the temperature and capacitive reading. **Baud** is a measurement of bit rate.

1. Try touching the body of the sensor to make the moisture measurement go up. The temperature reading is only on the chip itself but gives an idea of the room’s temperature.

**Last Step!**

Connect the breadboard to the Arduino if it isn’t connected already. Positive should go to from the 5V on the arduino to the + on the breadboard, in the same row as the resister. The negative should go from the – on the breadboard in the same row as the LED and connect to a GRD on the Arduino.

Modify and combine your LED code and moisture sensor code to control your LEDs based on the moisture or temperature reading! For example, you can blink your LEDs at different rates or turn on different colored LEDs based on your measurements.

To do this, you will need to add an if/else statement to the moisture sensor code, that turns on the LED when the capread value is less than 500 – meaning that the soil is dry and needs to be watered. In the void loop() function, just after the following statement:

uint16\_t capread = ss.touchRead(0); // capacitive reading

Add an if/else statement in this format:

void loop() {

// see https://adafruit.github.io/Adafruit\_Seesaw/html/class\_adafruit\_\_seesaw.html for documentation

float tempC = ss.getTemp(); // temperature reading

uint16\_t capread = ss.touchRead(0); // capacitive reading

if (capread > 500)

{

digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)

delay(500); // wait for a second

}

else

{

digitalWrite(led, LOW); // turn the LED off by making the voltage LOW

}

Serial.print("Temperature: "); Serial.print(tempC);

Serial.println("\*C");

Serial.print("Capacitive: "); Serial.println(capread);

delay(1000); // wait for a second

}

This should turn off the LED when you are touching the moisture sensor. When you let go of the moisture sensor, the LED should turn back on.