A low cost DIY oil IPM for your crops

An emulsified vegetable oil spray can smother mites and soft-bodied insects and can suppress powdery mildew if you actually coat the target. Soybean oil has the strongest evidence. Corn oil works too, and blending the two offers some advantages. In the following article I tell you how to prepare such a spray as well as some of the scientific evidence showing how it works.



Corn oil, one of the main components of this IPM spray

Why combine soybean and corn oil?

- **Fatty acid profiles differ.** Soybean oil is richer in unsaturated fatty acids (linoleic, linolenic), while corn oil contains more oleic and palmitic. That mix can change the viscosity and spreading behavior on leaves.
- **Broader efficacy.** Soybean oil has strong data against powdery mildew, mites, and whiteflies (1) (2) (3). Corn oil has been validated in cucumber mildew trials (5). Using both hedges against variability between pests and

crops.
- **Physical properties.** Mixed oils can emulsify more easily and form finer droplets than a single oil, which may improve coverage and reduce visible residues.

Why use both Tween 20 and Tween 80?

- **Hydrophilic balance.** Tween 20 (polyoxyethylene sorbitan monolaurate) is more hydrophilic, while Tween 80 (polyoxyethylene sorbitan monooleate) is more lipophilic. Together, they stabilize emulsions of mixed triglyceride oils better than either one alone.
- **Reduced creaming/separation.** A dual-Tween system forms smaller, more stable droplets that resist breaking apart. This means the concentrate stays uniform longer and the spray deposits more evenly on foliage (4).

Step 1. Prepare the concentrate

Mix in a clean container:

- **Soybean oil:** 200 mL per liter (~760 mL per US gallon)
- **Corn oil:** 200 mL per liter (~760 mL per US gallon)
- **Tween 20:** 10 mL per liter (~38 mL per gallon)
- **Tween 80:** 10 mL per liter (~38 mL per gallon)
- Fill with clean water to reach 1 L (or 1 gal).

Mix for at least 30 minutes, ensure it is uniform. Always mix well before use. This is the concentrate: **20% soybean oil, 20% corn oil, 1% Tween 20, 1% Tween 80**.

Step 2. Dilute for spraying

For foliar application:

- **Dilution rate:** Add ~20mL of concentrate per liter of water (~75 mL per US gallon of water). If pests are present you can increase the rate up to 32mL/L (~120mL/gal).
- **Note on coverage:** Coverage is critical for this spray to work as it only kills insects on contact or prevents PM by building an oil film on the leaf that prevents spore germination. Without full coverage effectiveness will drop.

This produces a **0.8% oil spray** with **0.02% Tween 20 and 0.02% Tween 80** in the final spray solution. Mix well before use.

Shelf life considerations

- **Concentrate:** A freshly prepared concentrate can stay stable for several weeks if kept sealed, cool, and out of light. Always shake well before use, since some slow separation can occur.
- **Diluted spray:** Once mixed with water, use the spray the same day. Emulsions can separate within 12-24 hours, and microbial growth in water can destabilize the mix. Discard leftovers rather than storing diluted spray.
- **Indicators of instability:** Layering, large oil droplets, or visible separation mean the emulsion is breaking, don’t spray that on plants without mixing well again.

Why it works

Soybean oil sprays at 2% suppressed powdery mildew on roses and tomatoes (1), reduced spider mites by 97-99% (2), and

deterred whiteflies (3). Corn oil added control of cucumber mildews (5). Tweens stabilize and spread the oils (4).

Bottom line

- **Concentrate:** 200 mL soybean oil + 200 mL corn oil + 10 mL Tween 20 + 10 mL Tween 80 per liter (or 760 mL + 760 mL + 38 mL + 38 mL per gallon), topped up with water.
- **Spray dilution:** 75 mL concentrate per gallon of water.
- **Final spray:** 0.8% oil, 0.02% Tween 20, 0.02% Tween 80.
- **Shelf life:** Weeks for concentrate (if stored sealed, cool, dark); hours for diluted spray.

This blended, dual-Tween foliar spray is a low-cost, evidence-backed way to add an oil-based control into hydroponic IPM programs.

Connecting a low cost TDR moisture content/EC/temp sensor to a NodeMCUv3

I have discussed moisture content sensors extensively in the past. I have written posts about the use of capacitive moisture sensors to measure volumetric moisture content, including how to create sensor stations and how to calibrate them. However, while capacitive moisture content sensors can be a low cost alternative for low resolution monitoring of moisture content, more precise applications require the use of higher accuracy sensors, such as Time Domain Reflectometry (TDR) sensors. In this post I am going to show you how to

connect a low cost microcontroller (NodeMCUv3) to a low cost TDR moisture content sensor. *Note, some of the product links below are amazon affiliate links, which help support this blog at no additional cost to you.*

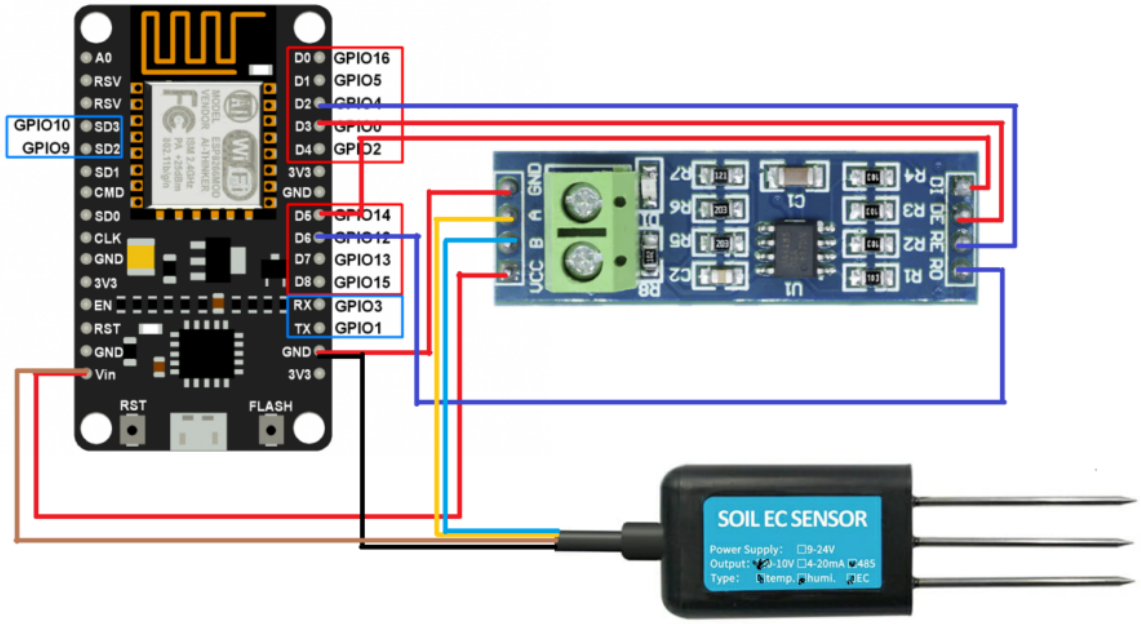


Diagram showing cable connections between moisture content sensor NodeMCUv3 and communication board.

While popular sensors like Teros-12 sensors cost hundreds of dollars, lower cost alternatives have been created by Chinese manufacturers. Using this github repository by git user Kromadg, I have been able to interface some of these low cost TDR sensors with a NodeMCUv3. The NodeMCUv3 is a very low cost microcontroller unit that you can get for less than 5 USD a piece. It is also WiFi enabled, so this project can be expanded to send data through Wifi to use in datalogging or control applications. For this project you will need the following things:

1. Micro USB cable
2. NodeMCUv3
3. THC-S RS485 sensor (Make sure to get the THC-S model)
4. TTL to RS485 communication board
5. Breadboard and jumper cables to make connections or cables and a soldering kit to make final connections.

The above diagram shows you how to connect the sensor, TTL-to-RS485 communication board and the NodeMCUv3. You will also want to make sure you install the ESP Software serial library in your Arduino IDE, as the normal Software Serial library won't work. You can do this by downloading the zipped library from github and then using the Sketch->Include Library menu option. Once you do so, you can upload the following code into your NodeMCUv3.

```
#include <SoftwareSerial.h>
#include <Wire.h>

// This code is a modification of the code found here (https://github.com/kromadg/soil-sensor)

#define RE D2
#define DE D3

const byte hum_temp_ec[8] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x03, 0x05, 0xCB};
byte sensorResponse[12] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte sensor_values[11];

SoftwareSerial mod(D6, D5); // RX, TX

void setup() {
    Serial.begin(115200);
    pinMode(RE, OUTPUT);
    pinMode(DE, OUTPUT);
    digitalWrite(RE, LOW);
    digitalWrite(DE, LOW);
    delay(1000);
    mod.begin(4800);
    delay(100);
}

void loop() {
    /************** Soil EC Reading ******************/
    digitalWrite(DE, HIGH);
    digitalWrite(RE, HIGH);
```

```
    memset(sensor_values, 0, sizeof(sensor_values));
    delay(100);
    if (mod.write(hum_temp_ec, sizeof(hum_temp_ec)) == 8) {
        digitalWrite(DE, LOW);
        digitalWrite(RE, LOW);
        for (byte i = 0; i < 12; i++) {
            sensorResponse[i] = mod.read();
            yield();
        }
    }

    delay(250);

    // get sensor response data
     float soil_hum = 0.1 * int(sensorResponse[3] << 8 |
sensorResponse[4]);
     float soil_temp = 0.1 * int(sensorResponse[5] << 8 |
sensorResponse[6]);
       int soil_ec = int(sensorResponse[7] << 8 |
sensorResponse[8]);

      /************** Calculations and sensor corrections
*************/

    float as_read_ec = soil_ec;

     // This equation was obtained from calibration using
distilled water and a 1.1178mS/cm solution.
    soil_ec = 1.93*soil_ec - 270.8;
    soil_ec = soil_ec/(1.0+0.019*(soil_temp-25));

     // soil_temp was left the same because the Teros and
chinese sensor values are similar

    // quadratic aproximation
     // the teros bulk_permittivity was calculated from the
teros temperature, teros bulk ec and teros pwec by Hilhorst
2000 model
    float soil_apparent_dieletric_constant = 1.3088 + 0.1439 *
soil_hum + 0.0076 * soil_hum * soil_hum;
```

```
                float    soil_bulk_permittivity    =
soil_apparent_dieletric_constant;    ///  Hammed  2015
(apparent_dieletric_constant is the real part of permittivity)
    float soil_pore_permittivity = 80.3 - 0.37 * (soil_temp -
20); /// same as water 80.3 and corrected for temperature

    // converting bulk EC to pore water EC
    float soil_pw_ec;
    if (soil_bulk_permittivity > 4.1)
         soil_pw_ec = ((soil_pore_permittivity * soil_ec) /
(soil_bulk_permittivity - 4.1) / 1000); /// from Hilhorst
2000.
    else
        soil_pw_ec = 0;

    Serial.print("Humidity:");
    Serial.print(soil_hum);
    Serial.print(",");
    Serial.print("Temperature:");
    Serial.print(soil_temp);
    Serial.print(",");
    Serial.print("EC:");
    Serial.print(soil_ec);
    Serial.print(",");
    Serial.print("READEC:");
    Serial.print(as_read_ec);
    Serial.print(",");
    Serial.print("pwEC:");
    Serial.print(soil_pw_ec);
    Serial.print(",");
    Serial.print("soil_bulk_permittivity:");
    Serial.println(soil_bulk_permittivity);
    delay(5000);
}
```

Note that RE and DE are not placed on digital pins 2 and 3, as other pins in the NodeMCUv3 carry out other functions and the board will not initialize if it has the RS485-to-TTL communicator connected through those pins. The R0 and RI pins are connected to digital pins D5 and D6, this is because in the NodeMCUv3 pins D7 and D8 are used in serial communication

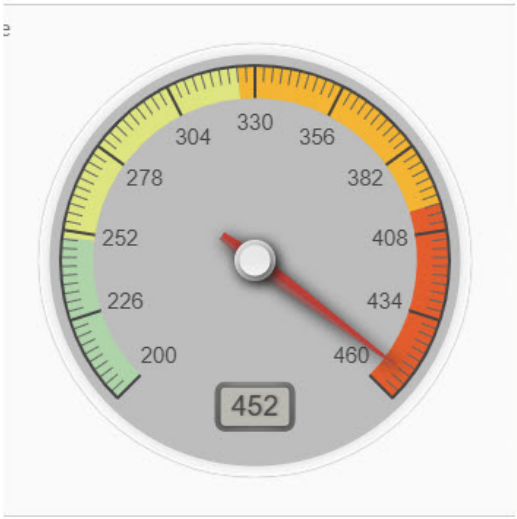
by the Serial swap command and therefore create conflicts if you use them with SoftwareSerial. The above digital pin distribution is one of the few that works well. Note that connecting RE or DE to digital pin 4 also works, but this means the blue LED on the NodeMCUv3 is powered on every time there is serial communication, a potentially undesirable effect if you're interested in battery powering the device.

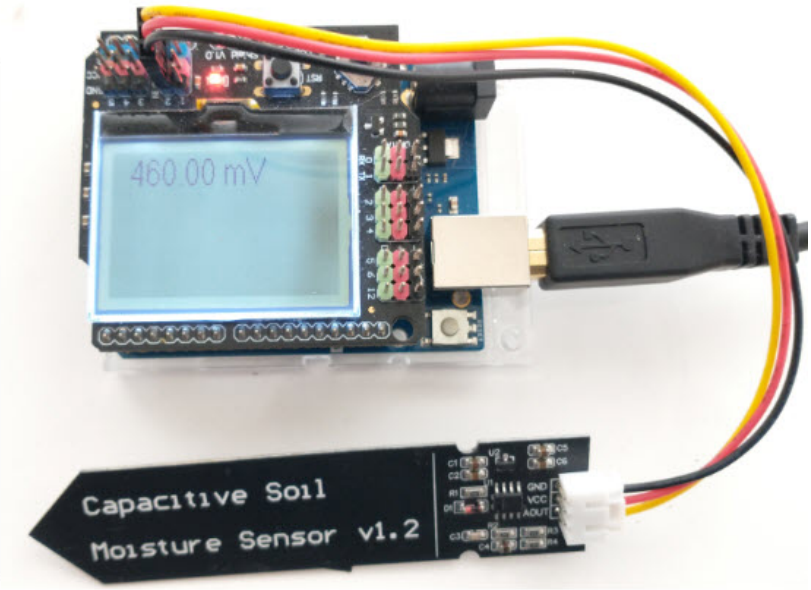
The board should now be printing all the measurements on your serial connection, so you should be able to see the readings through the Serial Monitor in the Arduino IDE. In the future I will be sharing how to expand this code to include WiFi and MQTT communication with a MyCodo server.

If you use this code please share your experience in the comments below!

Arduino hydroponics, how to build a sensor station with an online dashboard

In a previous post about Arduino hydroponics, I talked about some of the simplest projects you could build with Arduinos. We also talked about how you could steadily advance towards more complex projects, if you started with the right boards and shields. In this post, I am going to show you how to build a simple sensor station that measures media moisture and is also connected to a free dashboard platform (flespi). The Arduino will take and display readings from the sensor and transmit them over the internet, where we will be able to monitor them using a custom-made dashboard. **This project requires no proto-boards or soldering skills.**





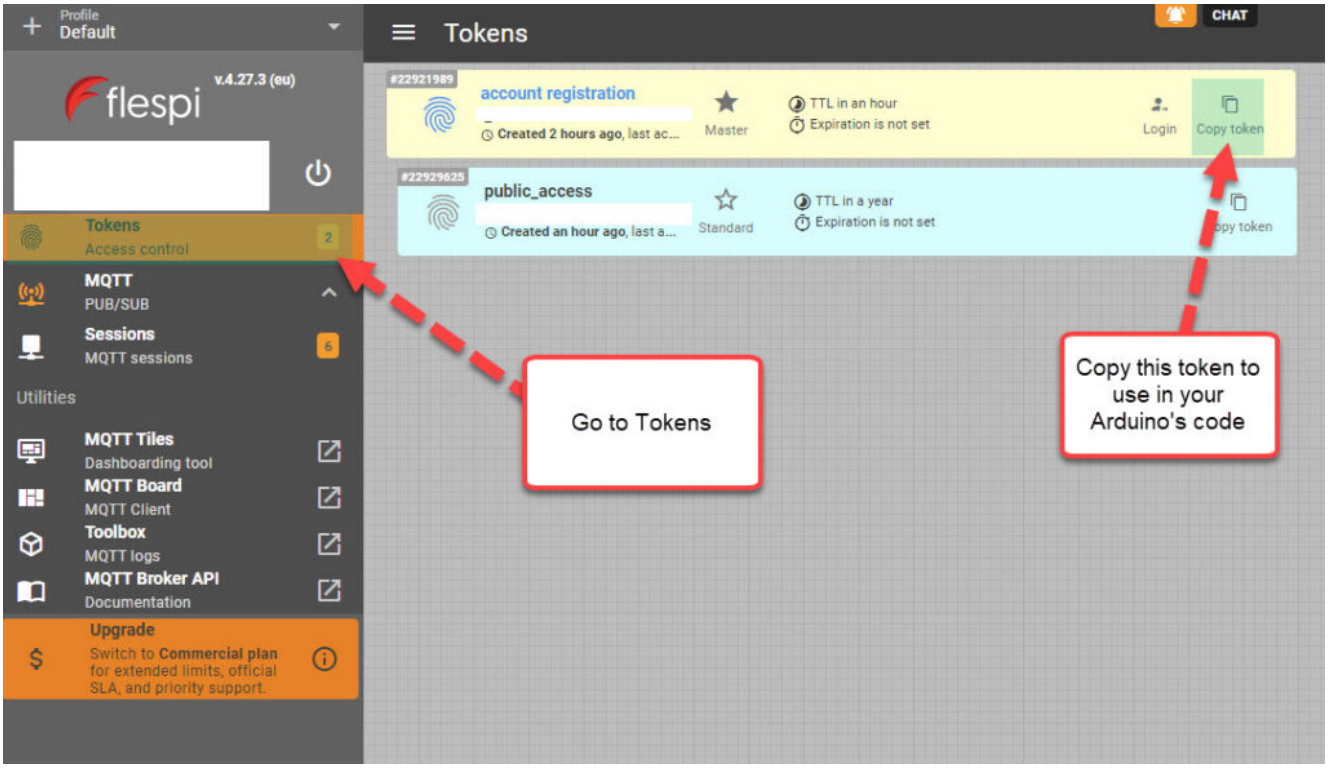
An Arduino Wifi Rev2 connected to a moisture sensor, transmitting readings to an MQTT server hosted by flespi that generates an online dashboard

The idea of this project is to provide you with a simple start to the world of Arduino hydroponics and IoT interfacing. Although the project is quite simple, you can use it as a base to build on. You can add more sensors, improve the display, create more complicated dashboards, etc.

What you will need

For this build, we are going to use an Arduino Wifi Rev2 and an LCD shield from DFRobot. For our sensor, we are going to be using these low-cost capacitive moisture sensors. This sample project uses only one sensor, but you can connect up to five sensors to the LCD shield. Since this project is going to be connected to the internet, it requires access to an internet-connected WiFi network.

Additionally, you will also need a free flespi account. Go to the flespi page and create an account before you continue with the project. You should select the MQTT option when creating your account since the project uses the MQTT protocol for transmission. After logging into your account, copy the token shown on the “Tokens” page, as you will need it to set up the code.



Copy the token from the “Tokens” menu in flespi

Libraries and code

This project uses the U8g2, ArduinoMQTTClient and WiFiNINA libraries. You should install them before attempting to run the code. The code below is all you need for the project. Make sure you edit the code to input your WiFi SSID, password, and Flespi token, before uploading it to your Arduino. This also assumes you will connect the moisture sensor to the analogue 2 port of your Arduino. You should change the ANALOG_PORT variable to point to the correct port if needed.

```
#include <Arduino.h>
#include <U8g2lib.h>
#include <WiFiNINA.h>
#include <ArduinoMqttClient.h>
#include <SPI.h>

#define SECRET_SSID "enter your wifi ssid here"
#define SECRET_PASS "enter your password here"
#define FLESPI_TOKEN "enter your flespi token here"
#define ANALOG_PORT  A2
```

```
#define MQTT_BROKER    "mqtt.flespi.io"
#define MQTT_PORT      1883

U8G2_ST7565_NHD_C12864_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/
13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
float capacitance;
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

// checks connection to wifi network and flespi MQTT server
void check_connection()
{
  if (!mqttClient.connected()) {
    WiFi.end();
    WiFi.begin(SECRET_SSID, SECRET_PASS);
    delay(10000);
    mqttClient.setUsernamePassword(FLESPI_TOKEN, "");
    if (!mqttClient.connect(MQTT_BROKER, MQTT_PORT)) {
      Serial.print("MQTT connection failed! Error code = ");
      Serial.println(mqttClient.connectError());
      delay(100);
    }
  }
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(4, OUTPUT);
  Serial.begin(9600);
  analogReference(DEFAULT);
  check_connection();
}

void loop() {

  String moisture_string;
  check_connection();

  // read moisture sensor, since this is a wifiRev2 we need to
set the reference to VDD
  analogReference(VDD);
```

```
  capacitance = analogRead(ANALOG_PORT);
  // form the string we will display on the Arduino LCD screen
  moisture_string = String(capacitance) + " mV";
  Serial.println(moisture_string);
  // send moisture sensor reading to flespi
  mqttClient.beginMessage("MOISTURE1");
  mqttClient.print(capacitance);
  mqttClient.endMessage();

  // the LCD screen only works if I reinitialize it on every
loop
   // I also need to reset the analogReference for it to
properly work
  analogReference(DEFAULT);
  u8g2.begin();
  u8g2.setFont(u8g2_font_crox3h_tf);
  u8g2.clearBuffer();          // clear the internal memory
   u8g2.drawStr(10,15,moisture_string.c_str());   // write
something to the internal memory
  u8g2.sendBuffer();          // transfer internal memory to
the display

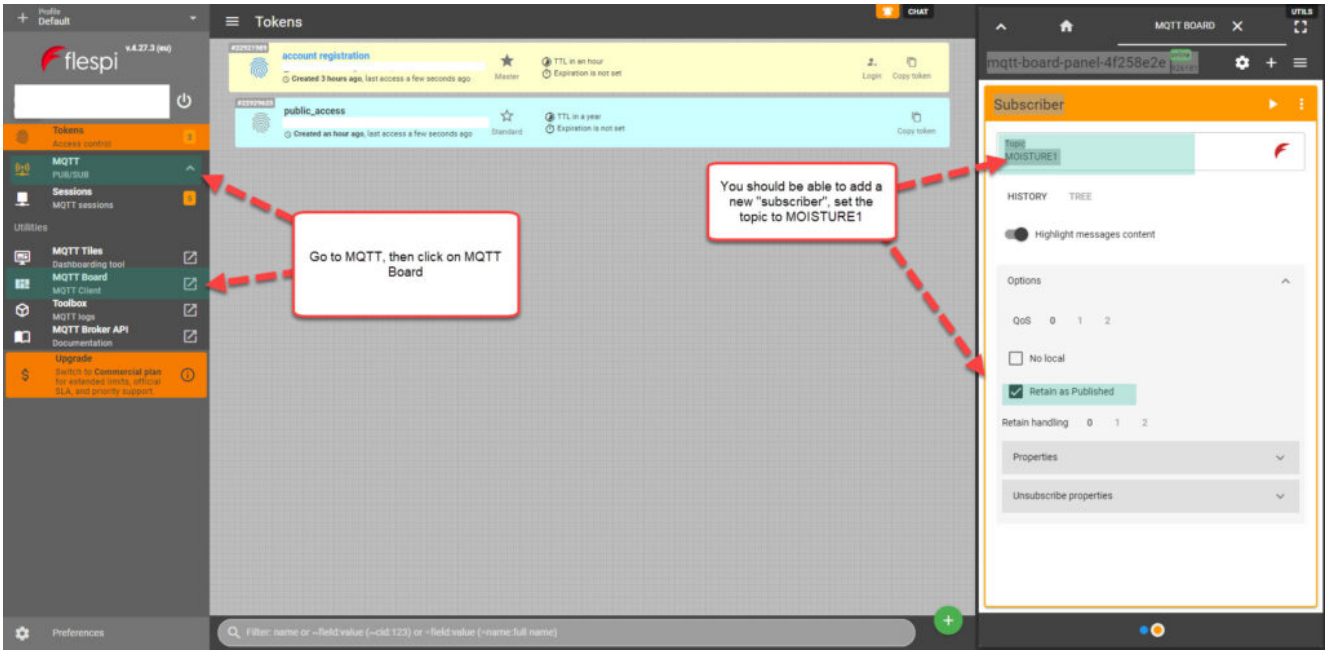
  delay(5000);
}
```

Your Arduino should now connect to the internet, take a reading from the moisture sensor, display it on the LCD shield and send it to flespi for recording. Note that the display of the data on the LCD shield is quite rudimentary. This is because I didn’t optimize the font or play too much with the interface. However, this code should provide you with a good template if you want to refine the display.

Configure Flespi

The next step is to configure flespi to record and display our readings. First, click the MQTT option to the left and then go into the “MQTT Board” (click the button, no the arrow that opens up a new page). Here, you will be able to add a new subscriber. A “subscriber” is an instance that listens to MQTT

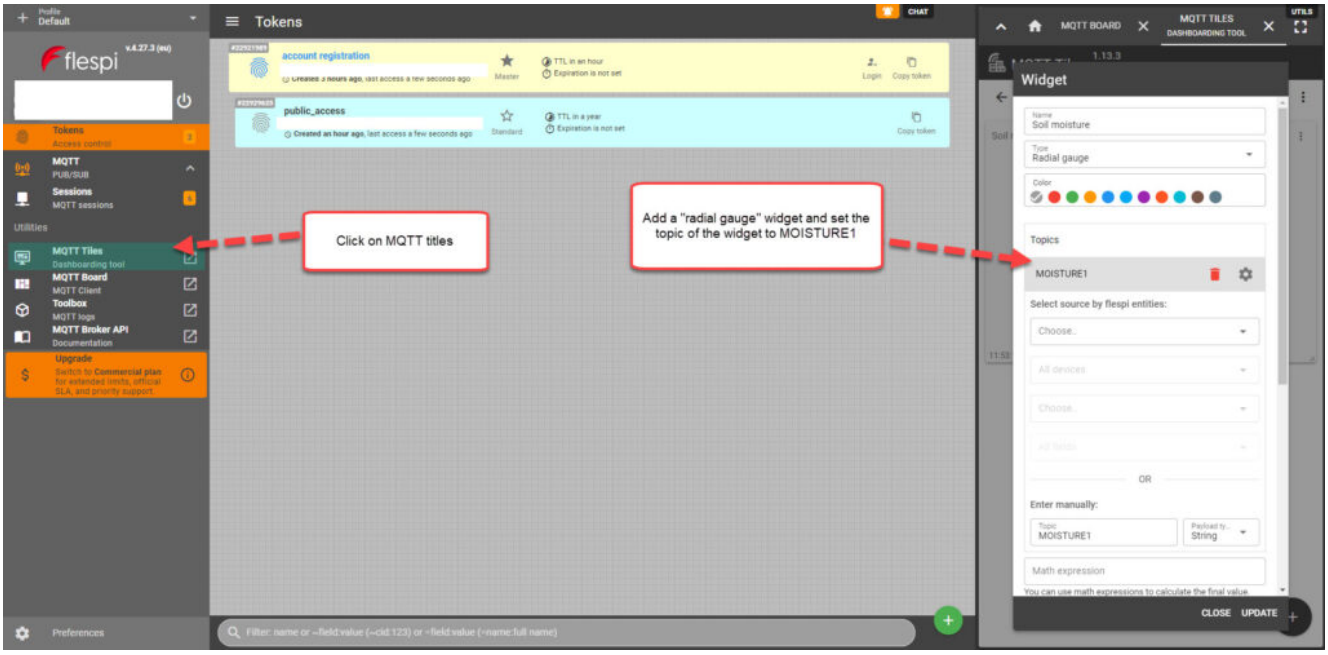
messages being published and “MOISTURE1” is the topic that our Arduino will be publishing messages to. If you want to publish data for multiple sensors, you should give each sensor its own topic, then add one flespi subscriber for each sensor.



Go to flespi and create a new “subscriber”, set the topic to MOISTURE1

Create the Dashboard

The last step, is to use the “MQTT Titles” menu to create a dashboard. I added a gauge widget to a new dashboard, and then set the topic of it to MOISTURE1, so that its data is updated with our MQTT messages. I set the minimum value to 200; the maximum value to 460; and the low, mid, and high levels to 250, 325, and 400 respectively.



Use the MQTT titles menu to add widgets to a new dashboard

After you finish creating the dashboard, you can then use the “Get link” button, which looks like a link from a chain next to your dashboard’s title. You will need to create an additional token in the “Tokens” menu so that you can use it for the sharing of the dashboard. After you generate the link, it should be publicly available for anyone who is interested. This is the link to the dashboard I created.

Conclusion

You can create a simple and expandable sensor station using an Arduino Wifi Rev2, a capacitive moisture sensor, and an LCD shield. This station can be connected to the internet via Wifi and send its data to flespi, which allows us to create free online dashboards. You can expand on this sensor station by adding more moisture sensors or any other Gravity shield compatible sensors, such as a BME280 sensor for temperature, humidity, and atmospheric pressure readings.

How to make an organic hydroponic nutrient solution

Hydroponic nutrients are usually made with synthetic chemicals that come from industrial processes. While these chemicals are usually of a higher purity than those mined or obtained from animal or vegetable resources, it also means that these products contain no microbes or bio-stimulants and their origin implies they cannot be used in organically certified growing operations. Growers who want a more organic approach might still want to use hydroponic solutions, but traditional hydroponic fertilizers cannot be used due to the fact that they lack many of the traits desired in an organic fertilizer. In this post, I will show you how you can create a complete hydroponic solution from scratch using only OMRI-approved raw materials.



This seal is given to products that have been approved by the OMRI organization, which certifies which products can be used in organic culture

OMRI nutrient sources

A complete hydroponic solution should provide all substances that are necessary for plant growth. This means we need to

provide nitrogen, phosphorus, potassium, magnesium, calcium, sulfur, iron, zinc, boron, copper, molybdenum, and manganese. Furthermore, we need to ensure that all of these nutrients are provided in forms that are available for the plants. This means we need to find sources that contain all the elements we need and then create a process that makes all of these nutrients adequately bioavailable. The following are the nutrient sources that we will be using, all of them are OMRI listed:

Please note the amazon links below are referral links. This means that I get a small commission when you choose to buy the products through these links, at no extra cost to you.

- Bark compost
- Solubor
- Copper Sulfate
- Corn Steep Liquor
- Ferti-Nitro Plus
- Iron Sulfate
- Magnesium Sulfate
- Manganese Sulfate
- Potassium Sulfate
- Seabird Guano
- Zinc Sulfate

Mixing the solution

This solution cannot be created in a concentrated form. This means we will be preparing a solution that will be fed directly to plants. However, since many of the inputs contain a lot of insoluble materials – due to their origin – there will need to be a filtration process in the end. This filtration step is necessary if you want to avoid problems dealing with the clogging of irrigation lines, in case you want to feed this into a regular irrigation system. If you want to hand water directly, then you can avoid this

filtration step.

Since the solution is not concentrated, the amounts to be weighed can be small for some of the materials. For this reason, I advise you to prepare at least 100 gallons of solution, so that you don’t require to weigh very small amounts of material. This will help keep the errors due to measurements low. To make this preparation you will need the following materials:

- A tank that can hold 100 gallons
- A flow meter to measure water flow
- A scale that can weight +/-0.01g max 500g
- An air pump rated for at least 100 gallons of water
- Air stones to diffuse air

To prepare the solution (100 gallons), follow these steps:

1. Add 50 gallons of water using the flow meter. Ideally use RO water, but you can use tap water as well if that is not possible.
2. Weigh and add all the ingredients per the table below.
3. Add another 50 gallons of water using the flow meter.
4. Place the air pump inside the solution and switch it on.
5. Maintain constant aeration for at least 15 days. Do not use it before this time has passed.
6. After 15 days have passed, filter the solution to use in irrigation lines or use directly to hand water. Keep air flowing through the solution even after the 15 days have passed.
7. The solution might also become basic during this process, if necessary, you can bring the pH of the solution down with citric acid before watering plants.

Bark compost	190
Solubor	0.65
Copper sulfate	0.15

Corn Steep Liquor	330
Ferti-Nitro Plus	220
Iron Sulfate	4
Magnesium sulfate	190
Manganese Sulfate	1
Potassium Sulfate	136
Seabird Guano	265
Zinc Sulfate	0.10

Table of ingredients to weigh. Masses are in grams.

The reason for the long wait

Plants ideally require nitrate in order to grow, the above inputs do not contain nitrate in appreciable amounts but mainly organic nitrogen sources. In this and this previous posts, you can learn more about organic nitrogen and why it is not ideal to use this in an unprocessed manner in a hydroponic crop. When you irrigate with organic nitrogen, most of the nitrogen will go unused and significant time will need to pass in the root zone for it to become available. The organic nitrogen decomposition process can also destabilize the pH of the root zone, making it harder for plants to properly absorb nutrients. By carrying out this process outside of the root zone, we make it easier on the plants, as we feed a pre-digested solution that is rich in available nutrients and microbes. The Seabird Guano and Bark compost, both provide the microbe inoculations necessary for the nitrogen decomposition process to take place. Oxygen, which we continuously pump into the solution, is also key to this process. The CSL and the Ferti-Nitro Plus will provide the organic nitrogen sources that will be decomposed.

This solution also contains a significant amount of amino acids. Although most of these amino acids will be converted into more readily absorbable nitrate through the digestion

process, a small amount will be left undigested, which will lock onto the heavy metal ions. This will help prevent precipitation issues and provide the plant with organically derived chelates.

Also note that no specific molybdenum input is included. This is because it is present as an impurity in the corn steep liquor at a high enough concentration, so its explicit addition is not required.

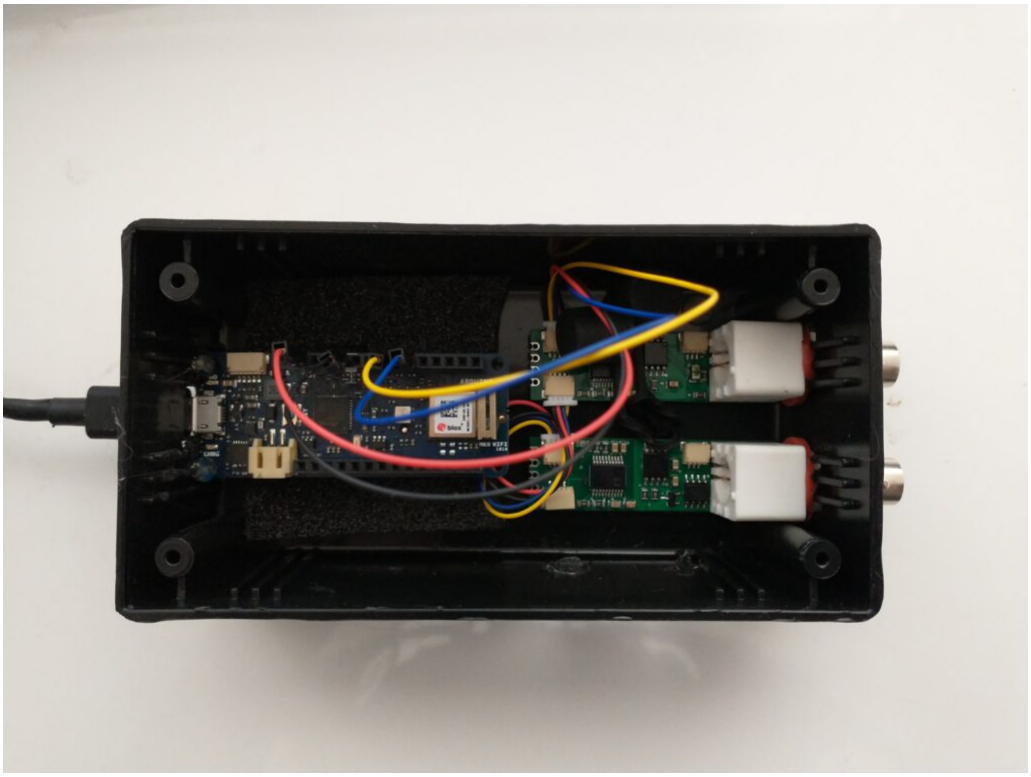
Conclusion

The above solution should fully replace a traditional hydroponic solution, using only OMRI-approved materials. The final concentrations of nutrients should be spot on for the healthy development of most small and large plants. The solution will also contain a lot of microbes and bio-stimulants, which will also help plant growth. Of course, the final character of the solution will depend on the temperature of the digestion, the amount of aeration present, and the nature of the inputs used (as OMRI inputs have a significant amount of variability due to their sourcing). It might take a few tries to adjust this process to your particular conditions. Note that the above solution is intended to be used with soilless media that has not been amended, as it should provide all nutrients required for plant growth.

Did you prepare the above solution? Leave a comment telling us about your experience!

Creating a pH/EC wireless sensing station for MyCodo using an Arduino MKR Wifi 1010

There are multiple open-source projects available online for the creation of pH/EC sensing stations for hydroponics. However, all of the ones I have found use a single Arduino or Raspberry Pi to perform the measurements and store any data, making them unsuitable for applications where more flexibility is needed. For example, a facility using multiple different reservoir tanks for nutrient storage might require multiple pH/EC sensing stations, and single-board wired setups would be unable to accommodate this without a lot of additional development. In this post, I am going to show you a simple pH/EC sensing station I built with an Arduino MKR Wifi 1010 that can communicate with a MyCodo server using the MQTT protocol. Multiple sensing stations could be built and all of them can communicate with the same MyCodo server.



My Arduino MKR wifi 1010 based sensing station, using uFire pH and EC boards in a small project box.

This project makes use of the small pH/EC boards provided by uFire, which have a lower cost compared to those provided by companies like Atlas, but do have adequate electrical isolation to avoid problems in readings when multiple electrodes are put in the same solution. This is a substantial improvement over other low-cost boards where using multiple probes can cause heavy electrical noise and interference. In order to build this project you will require the following materials:

Note, some of the links below are amazon affiliate links. This means that I get a small commission if you purchase through these links at absolutely no extra cost to you. The links to other websites are not affiliate links.

1. Arduino MKR Wifi 1010
2. uFire pH probe
3. uFire EC probe
4. A rugged pH probe with a VNC connector
5. An rugged EC probe with a VNC connector
6. Two Qwiic-to-Qwiic connectors

7. One Qwiic-to-male connector
8. A project box to put everything inside (optional)
9. A micro USB cable

The code for the project is shown below:

```
#include <uFire_EC.h>
#include <uFire_pH.h>
#include <WiFiNINA.h>
#include <ArduinoMqttClient.h>

#define SECRET_SSID "ENTER WIFI SSID HERE"
#define SECRET_PASS "ENTER WIFI PASSWORD HERE"

//calibration solutions used
#define PH_HIGH_SOLUTION_PH 7.0
#define PH_LOW_SOLUTION_PH  4.0
#define EC_HIGH_SOLUTION_EC 10.0
#define EC_LOW_SOLUTION_EC  1.0
#define CALIBRATION_TEMP    20.0

// topics for the mqtt sensors
// Make sure all stations have different topics
#define EC_TOPIC       "EC1"
#define PH_TOPIC       "PH1"
#define CALIB_TOPIC    "CALIB1"
#define MQTT_BROKER    "ENTER MQTT SERVER IP HERE"
#define MQTT_PORT      1883

int status = WL_IDLE_STATUS;     // the Wifi radio's status
String message;

uFire_pH ph;
uFire_EC ec;
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

void check_connection()
{
  if (!mqttClient.connected()) {
    WiFi.end();
    status = WiFi.begin(SECRET_SSID, SECRET_PASS);
```

```
    delay(10000);
    if (!mqttClient.connect(MQTT_BROKER, MQTT_PORT)) {
      Serial.print("MQTT connection failed! Error code = ");
      Serial.println(mqttClient.connectError());
      delay(100);
    }
    mqttClient.subscribe(CALIB_TOPIC);
  }
}

void setup()
{
  Serial.begin(9600);
  while (!Serial);

  // connect to wifi and mqtt broker
  check_connection();
  // coorectly initialize the uFire sensors
  // note the Wire.begin() statement is critical
  Wire.begin();
  ec.begin();
  ph.begin();
}

void loop()
{
  // mqtt keep alive
  mqttClient.poll();

  // read messages
  message = "";
  while (mqttClient.available()) {
      message += (char)mqttClient.read();
    }

  // execute calibration if requested
  Serial.println(message);
              if       (message       ==       "EC1_HIGH")
ec.calibrateProbeHigh(EC_HIGH_SOLUTION_EC, CALIBRATION_TEMP);
               if       (message       ==       "EC1_LOW")
ec.calibrateProbeLow(EC_LOW_SOLUTION_EC, CALIBRATION_TEMP);
```

```
            if      (message      ==      "PH1_HIGH")
ph.calibrateProbeHigh(PH_HIGH_SOLUTION_PH);
            if      (message      ==      "PH1_LOW")
ph.calibrateProbeLow(PH_LOW_SOLUTION_PH);

  // Measure EC
  ec.measureEC();
  Serial.println((String) "mS/cm: " + ec.mS);

  // Measure pH
  ph.measurepH();
  Serial.println((String) "pH: " + ph.pH);

  // Ensure the wifi and mqtt connections are alive
  check_connection();

  // post EC to MQTT server
  mqttClient.beginMessage(EC_TOPIC);
  mqttClient.print(ec.mS);
  mqttClient.endMessage();

  // post pH to MQTT server
  mqttClient.beginMessage(PH_TOPIC);
  mqttClient.print(ph.pH);
  mqttClient.endMessage();

  // ensure sensors are not probed too frequently
  delay(1000);

}
```

Once you get all the materials you should first assemble the components. Connect the pH and EC board together using the Qwiic-to-Qwiic connector, then use the Qwiic-to-male connector to hook up one of these boards to the Arduino (doesn’t matter which one). Connect the black cable to ground, red cable to 5V, blue cable to SDA, and yellow cable to SCL. Set up your board according to the instructions in the Arduino MKR wifi 1010 getting started page, modify the code above to properly include information about your wifi network, calibration solutions, and MQTT server, then upload the code. The Arduino

will connect to your Wifi and MQTT servers and automatically reconnect when there are connection issues.

The above code will also post the readings of the pH and EC sensors to topics PH1 and EC1 respectively if you add an input in MyCodo to capture these readings you should be able to store them and take control actions using the MyCodo interface. Additionally, the Arduino code will respond to calibration requests published to the topic "CALIB1". For example, if you want to calibrate your EC sensor with a two-point calibration method with a standard solution with an EC of 10mS/cm, you would put the electrode in the calibration solution, then send the message "EC1_HIGH" to the CALIB1 topic and the Arduino will perform the task as requested. The code assumes you will want to do 2 point calibrations for both EC and pH, with the calibration events triggered by EC1_HIGH, EC1_LOW, PH1_HIGH, and PH1_LOW. Note that the definition of the EC and pH values of the calibration solutions should be changed to the solutions you will be using within the code. The high/low values in the code, as is, are 10mS/cm|1mS/cm for EC and 7|4 for pH.

How to make your own stabilized mono-silicic acid for use in hydroponics

Please follow this link, for an updated an easier process for the synthesis of mono/ortho-silicic acid.

During the past several years, there have been a lot of "mono-silicic acid" products being marketed for their use in hydroponic culture. These differ from the traditional

potassium silicate based products in that they are very acidic in their concentrated form and are stable in solution for longer periods of time at the pH values used in hydroponics. While a hydroponic nutrient solution that has potassium silicate added to it and the pH adjusted to 5.5-6.5 will generally see extensive polymerization of the silicon-containing molecules within 24 hours, these stabilized mono-silicic acid products will be stable for far longer periods of time. They are therefore ideal for use in recirculating systems, where potassium silicate additions can be less effective.

If you watched my youtube video on silicon in hydroponics, you'll know that the most common way to produce these stabilized products is quite complicated and involves the use of silicon chloride and very specific reaction conditions. These are unavailable to hydroponic growers, reason why it is not easy to produce a home-made version of choline stabilized ortho-silicic acid (ch-OSA). However, choline is not the only reagent that can be used to stabilize silicic acid in solution, and research in industry has shown us that it is actually possible to form stabilized silicic acid products starting from potassium silicate itself.

Model representation of orthosilicic acid, also called mono-silicic acid.

This patent describes how to prepare mono-silicic acid solutions that are stabilized by carnitine and several other additives, in the region from 0.01 to 8% silicic acid by weight. The great thing about this process is that we can start from potassium silicate, which is readily available. The concentration of Si obtained will be significantly lower than what is possible when generating ch-OSA – where solutions can reach 40% mono-silicic acid – but the fact that we can prepare it from readily available materials might compensate for this to some extent. **It is also worth noting that this process**

comes from an unexpired patent, so it should not be used commercially without licensing the technology from the owner of the intellectual property.

Extrapolating from the contents of the patent and the examples given, we can come up with a process to brew our own mono-silicic acid at an 8% concentration. Here are the things you will need:

Note the amazon links below are affiliate links. This means that, if you choose to purchase through these links, I get a commission for your purchase, at no extra cost to you.

1. Potassium silicate (at least 32% K as K_2O)
2. Carnitine hydrochloride
3. Phosphoric acid (85%)
4. Propylene glycol
5. Distilled water
6. Scale to weight the materials (precision of at least +/- 0.1g, max at least 500g)

To prepare around 425g of stabilized mono-silicic acid, you could follow this process.

Before continuing please make sure you understand what you're doing. Wear adequate eye and body protection, carry this out in a place with enough ventilation and make sure you read the material safety data sheet (MSDS) of all the materials used. These instructions are provided for educational purposes only, follow them at *your own risk.*

1. Add 10g of carnitine hydrochloride to a clean 1000mL beaker
2. Add 65g of distilled water to the mix.
3. Stir until all the carnitine hydrochloride dissolves
4. Add 10g of propylene glycol.
5. Add 240g of 85% phosphoric acid.
6. Put the mixture on an ice bath with ample ice.
7. Wait for 15 minutes, so that the mixture cools down.

8. During the course of an hour, s*lowly* add 125g of potassium silicate to the mixture with constant stirring. Add more ice to the ice bath if needed to keep the solution cool. Note that predissolving the silicate in 150mL of distilled water and adding it as a liquid makes this process easier, although KOH additions might be required to complete its dissolution.

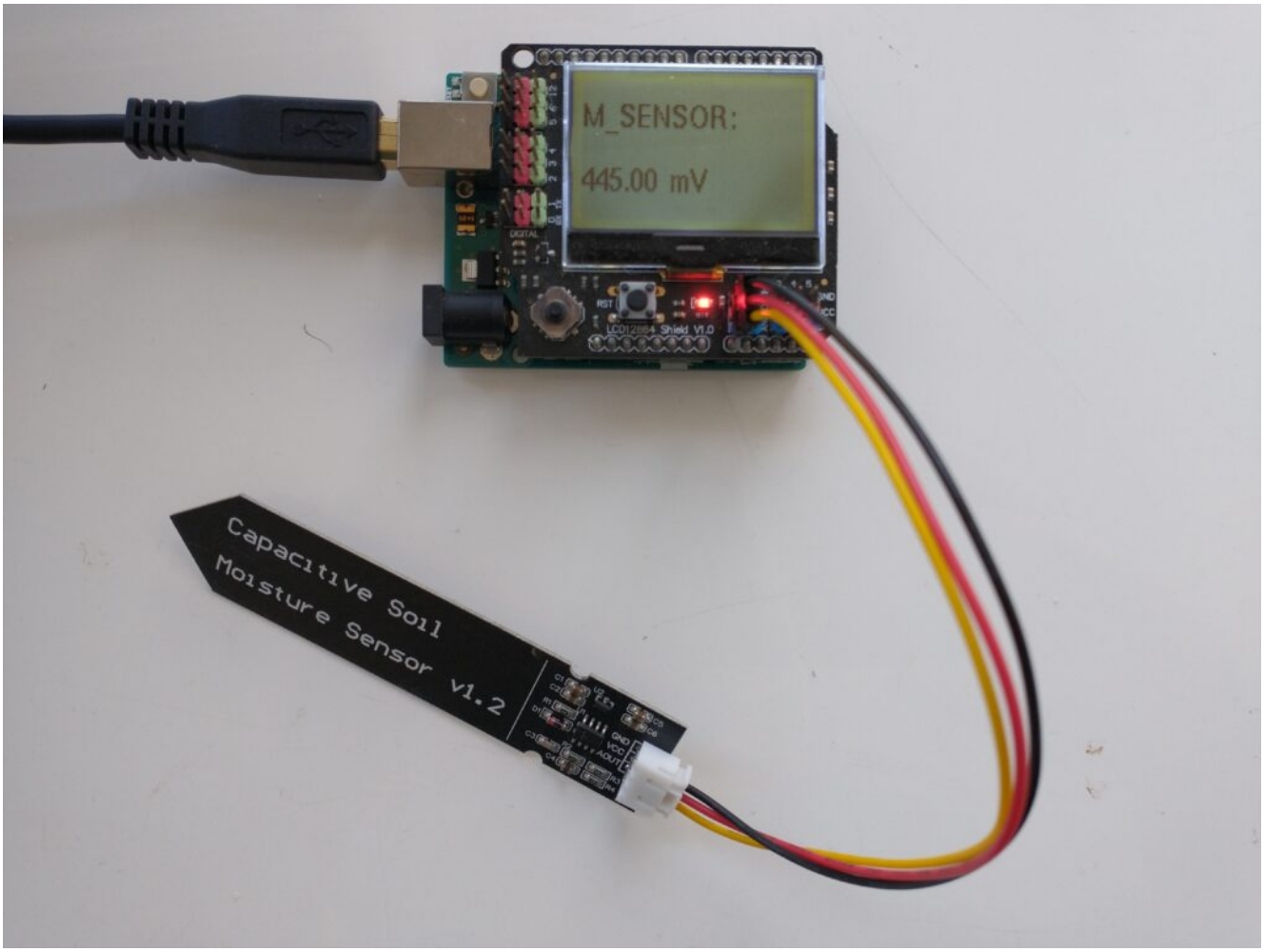
After this, you should be left with an acidic, completely translucid, carnitine and propylene glycol stabilized mono-silicic acid solution that should be around 7-8% w/w of Si as elemental Si. If there's any precipitate in the solution then the stabilization process did not go well and the silicic acid formed polymerized into silica. **This solution should then be used at around 1g/gal, which will provide ~18-20ppm of Si as elemental Si in your hydroponic solution**. When using this solution,. the silicon present at the pH used in hydroponics should be much more stable than when derived from direct addition of potassium silicate.

If you go through the above process, leave a comment and let us know how it went.

Calibrating a capacitive moisture/water content sensor for hydroponics

As I've mentioned multiple times in my blog, moisture sensing is one of the most important measurements in a hydroponic crop that uses a significant amount of media. It allows you to properly time irrigations and avoid over or under watering your plants. Capacitive sensors are the lowest cost initial

approach to adequate moisture monitoring of your media. In this post, we are going to learn how to use an Arduino with a gravity shield and a low-cost capacitive moisture sensor in order to accurately monitor the water saturation of our media.



A capacitive sensor plugged into an LCD shield and an Arduino UNO. The measurements are very easy to track with this setup.

An analogue capacitive moisture sensor like this one does not expose any metallic parts to the media and can be used for the monitoring of moisture content. This sensor is powered by 3.5-5V and gives you a voltage signal that is proportional to the dielectric constant of the media it is in. As the dielectric constant of media changes with the presence of water, so does the signal of the sensor. However, no specific voltage corresponds to a specific water content measurement by definition, so we need to calibrate the sensor in order to interpret the voltage values we read from it. In order to get readings from this sensor, I use an Arduino UNO, the above-

linked capacitive sensor, and an LCD shield from dfrobot that I can use to easily get the device readings. The arduino code used for this device is also shared below.

```
#include <U8glib.h>

#define XCOL_SET 0
#define XCOL_SET_UNITS 50

U8GLIB_NHD_C12864 u8g(13, 11, 10, 9, 8);
float capacitance;

void draw() {

  u8g.setFont(u8g_font_helvB10);
  u8g.drawStr(0,21,"M_SENSOR:");
  u8g.setPrintPos(XCOL_SET,51);
  u8g.print(capacitance);
  u8g.drawStr( XCOL_SET_UNITS,51,"mV" );
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(4, OUTPUT);
  Serial.begin(115200);
  analogReference(DEFAULT);
  Serial.println("MOISTURE");

  u8g.setContrast(0);
  u8g.setRot180();
}

void loop() {

  draw();
  capacitance = analogRead(1);
  Serial.println(capacitance);

  u8g.firstPage();
    do  {
      draw();
```

```
    }
      while( u8g.nextPage() );

  delay(1000);
}
```

The calibration of a moisture sensor usually requires the creation of what soil scientists call a “water retention” curve, which is a curve where you plot the sensor’s signal as a function of a fixed volume or weight of water added to the media. However, this approach involves the use of many different containers and the addition of water to the media followed by oven drying steps in order to accurately determine how much moisture was actually in the media for every measurement carried out.

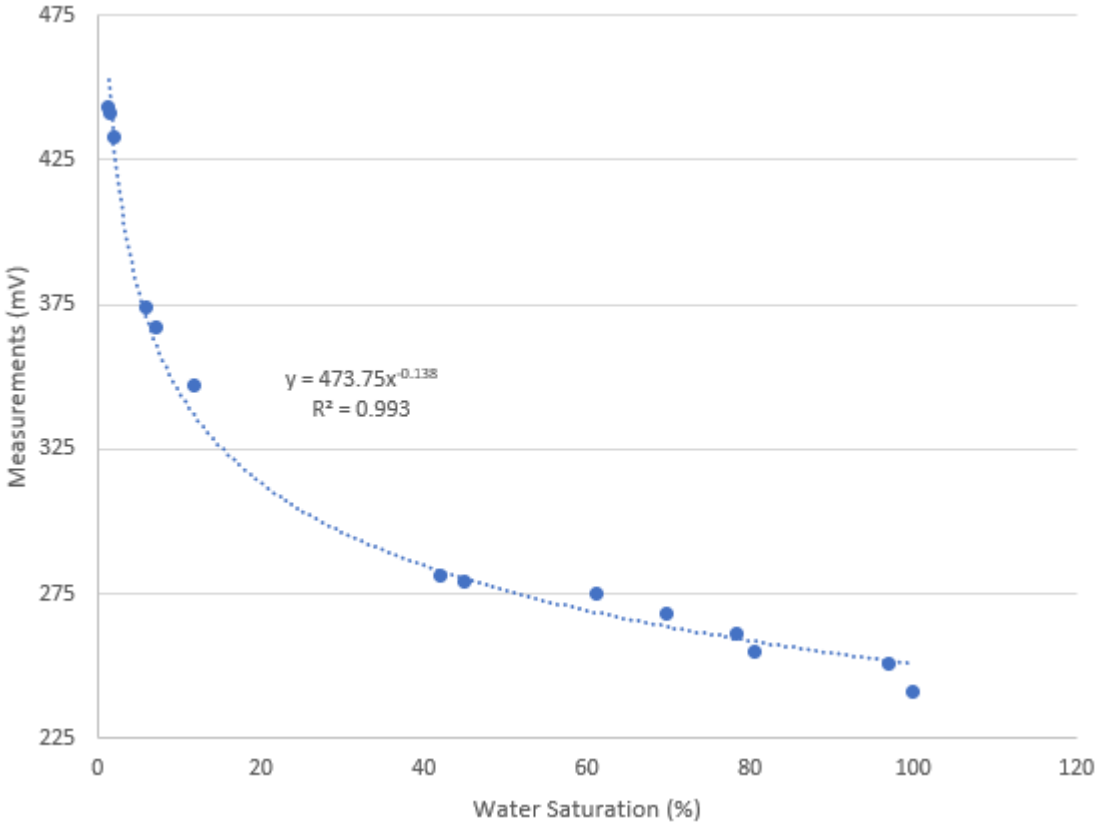
In order to do this procedure in an easier manner, losing as little accuracy as possible, I have created a calibration procedure that makes use of natural drying and only requires one single oven drying step. The procedure is as follows:

1. Heat the media that will be used for an experiment in an oven at 110°C (this drives out all water).
2. Wait for the media to cool to room temperature.
3. Fill the container that will be used for the test (this can plastic but has to have holes at the bottom). Put the sensor in the media, make sure the sensor is driven into the media until the top line is reached.
4. Take a reading, this is the “completely dry” media reading.
5. Disconnect the sensor from the Arduino.
6. Weigh the container+media+sensor. This will be the “dry weight”.
7. Take the sensor out.
8. Add water to the media until there is ample runoff coming out of the bottom.
9. Wait until no more runoff comes off.
10. Put the sensor in the media, making sure you drive it in

until the top line is reached. The sensor won’t be taken out for the remainder of the experiment.

11. Connect the sensor and take a reading. Take note of this value.
12. Disconnect the sensor.
13. Weigh the container+media+sensor. Take note of this value. The first reading you take is the “max saturation” weight.
14. Repeat steps 11 to 13 every 1-6 hours (time is not very important as long as you gather enough data points) until you reach close to the dry weight. This can take several days depending on the media used. The more points you measure, the more accurate your curve will be.

After carrying out this procedure, you can create a curve like the one shown below. You can use the difference between each weight and the dry weight divided by the difference between the weight at max saturation and the dry weight in order to calculate the water saturation percentage. As you can see, the curves for these sensors are fairly linear in the 40-100% moisture range for the media I tested, while below 40% the regime changes and the measurement increases exponentially until it reached the “dry weight” sensor value. The entire curve can be described with a power-law equation. This behavior will not be the same for all media tested, reason why it’s very important for you to calibrate the sensor for the specific media you want to use.



Calibration curve for a capacitive moisture sensor. In this case, the media was a mixture of 50% river sand and 50% rice husks.

Once you have your sensor calibrated you can know what a measurement in mV implies in terms of water saturation for a given media type. This allows you to time your irrigations at a given water saturation % effectively. Which water saturation percentage might be better, depends on the properties of your media and how the water potential changes as a function of the water saturation. However, this allows you to experiment with irrigations at different water saturation % values and figure out exactly where you need to water so that your plants are not under or overwatered.

It is also worth noting that the above sensor is probably not rugged enough for use in a hydroponic setup without a bit more hardening. In order to use these sensors in practical applications, you should apply conformal coating to the electronics at the top of the sensor and then use shrink tubing in order to fully protect these electronics from the elements.

MyCodo: an open-source solution for control, data logging and visualization

There are sadly not a lot of open source solutions for data logging, visualization, and control that have all the features required to be decently expandable and allow for different use cases. Most open source solutions have been developed by individuals for their particular needs. The consequence of this is that the hardware is very specific and difficult to expand on and the software has been written to be hard-coded to the hardware, making true wide use by the DIY community hard. However, MyCodo – a project that was shared with me by a reader of the blog – seems to get rid of this paradigm, creating an open-source implementation that is truly expandable and that offers most of the features anyone would want in a truly flexible DIY setup. In this post, we will talk about this project, what it offers and how it could be expanded for hydroponic grows of all scales.

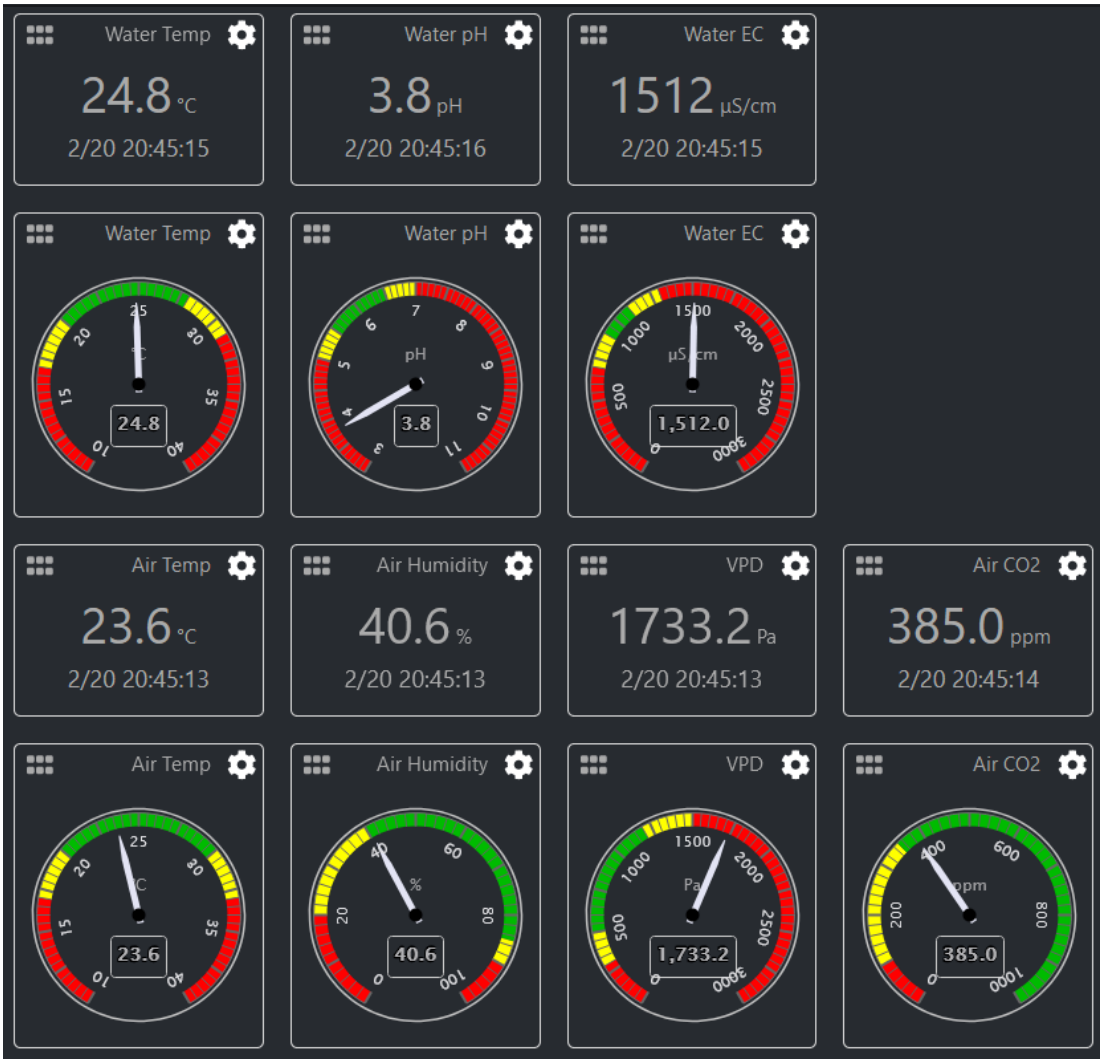


Sample control panel image taken from the MyCodo github project website

MyCodo is centered around a Raspberry Pi as its main computing hub. Once you install it, the project creates a web interface in the Raspberry Pi that you can use to manage inputs and the control actions that are derived from them. Most of the inputs that are supported by the MyCodo implementation are designed to be directly connected to the Raspberry Pi, such that the Pi acts both as the computing brains and the sensor/control hub for the implementation. When used in this way, only the Raspberry Pi is required, with whichever sensors and relays you want to add to it. This can already be powerful but has the problem that the Raspberry Pi is directly in the middle of the sensing/control environment and the entire implementation could be vulnerable to catastrophic failure due to interactions with the environment (say water getting on the Raspberry Pi).

Thankfully, the developer(s) of the MyCodo implementation had the vision to implement input/output options to use MQTT subscribe/publish mechanics. The MQTT protocol is a messaging

system where a device in a network can listen to or publish to different “topics”. So you can have an Arduino that publishes messages under the topic “HumiditySensor1” that contain the humidity value measured at each point in time and you can have it at the same time subscribe to a topic called “HumidityControl” and when it receives an “on” message in this topic it turns on a dehumidifier. You can then use MyCodo to “listen” to the humidity sensor messages, execute its own control algorithms on it, and publish the adequate control action to the “HumidityControl” topic whenever it thinks that a dehumidifier needs to be turned on. This is the way in which my custom-built Arduino/Raspberry PI control implementation generally works.



Another sample MyCodo panel

MyCodo, therefore, has a lot of flexibility that is not shared by any other open-source implementations, at least among the

ones I have found, for environmental control. Although there are no MQTT sensor stations implemented that I could find for the MyCodo, it should be fairly straightforward to build these sensing/control stations using Arduinos and the MQTT protocol and it should then be easy to add these stations to the MyCodo so that they can benefit from the system's control interfaces. In a system like this – with independent MQTT enabled sensor/control stations – you can control small or large facilities and not depend on the use of a single raspberry pi to do the entire setup. This means you could use the MyCodo to control different rooms and be able to have a centralized sensing setup for all your needs.

I have decided to give MyCodo a try for my latest hydroponic system. You should expect some videos about this in my youtube channel along with a github repository containing the code for the sensing and control stations that I am going to build in order to use Arduinos for turning relays on/off and send sensor readings. A Raspberry Pi will be used as the central control hub for the project, hosting the MyCodo webserver and code.

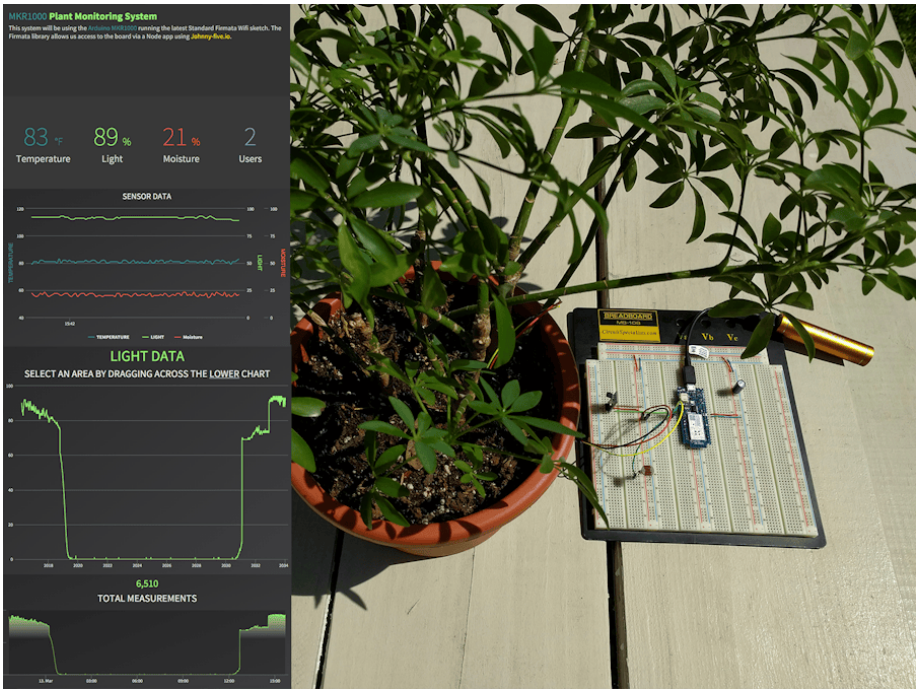
Pros and cons of building your own sensor and data logging system in hydroponics

If you've read my blog before, you know how important data logging is to having a successful hydroponic crop. Data allows you to monitor and tune the different variables in your grow, which allows you to give your plants the perfect environment through their entire growing cycle. However, deciding how to

do this is not simple, you need to decide if you're going to go with a company that sells some pre-made data-logging solution or you need to build everything yourself. In this post, I'm going to talk about several pros and cons of building your own data logging system for your hydroponic crop.

Pros

You have control over everything. The most important pro when building your own data logging solution is that you have total and absolute control over all aspects of it. If you want to support some type of sensors or have your data stored a certain way, there is nothing preventing you from doing this except your own skills and imagination. If you want to support an obscure messaging protocol, wireless transmission system, etc, it is all up to you. You won't be limited by the management decisions of an external company and you will be able to build a system that perfectly caters to your needs.



A simple plant monitoring custom built system. Read more here.

You will be able to leverage low-cost hardware. When building your own system you will be able to get all the parts

yourself. This means you will be able to substantially reduce costs. Of course, you're incurring the important cost of your time but the hardware itself will be low cost and once you implement the basic setup you will be able to connect new rooms and build new logging stations for a fraction of the cost of buying one commercially.

Take advantage of new hardware quickly. As new technologies for monitoring environmental variables are invented or the desire to control new variables comes into play, your ability to fully control your setup will allow you to take advantage of new hardware that comes into the scene while companies will usually be very slow to respond to such changes.

A much deeper understanding. When you build all the monitoring setup yourself, you will create a lot of understanding about how the sensors work, how each one of them is calibrated, how data is transmitted, stored, etc. If you build your own monitoring setup you will gain a much deeper understanding than somebody who just buys an off-the-shelf product.

No need for patchwork approaches. When you decide to get a commercial solution for data logging, one of the issues that comes along is that you will get the setup from a company that supports some types of sensors but you will often face challenges if you want a sensor outside this offering. This will usually mean buying a setup that includes that sensor from a completely different company, measuring some variables with one system and some others with another system.

Cons

No one to support it. The biggest drawback of building things yourself – or hiring someone to build a custom system for you – is that you will have no one to help you debug your system when things go wrong. You will also have limited ability to delegate this work, as your highly custom system will demand somebody with a high level of skill to become familiar with it

and operate it with the same level of proficiency as you do. A custom solution means all of this responsibility will fall on the shoulders of those who developed the system.



A custom built data logging system to read EC/pH/ORP. Read more here.

Limited by your knowledge. Although it is true that you will get a pretty deep understanding of the things you decide to incorporate into your system, you will also be very limited in the design and implementation of your system because of your particular limitations as an individual. A big company that develops a data logging system will have dozens of people working on it, and all of their experience will go into the decisions that were made in the sensor and software implementations. This can mean better sensor choices are made, more robust communication protocols are used, etc.

Not built for sharing. Custom-built systems usually have the problem that they are built with poor documentation. Sharing

is normally not the priority and people will prefer to build “fast and dirty” in order to get things done. This means that the code is usually poorly commented and of a lower quality than what you get from a product that comes from a business. Although some people who build custom software that they intend to release as open-source implementations will often go to great lengths to provide great code quality this is rarely the case when the intention is not to make everything open source.

Big overhauls are a big problem. Since your custom building efforts will usually rely on one or two individuals, bad decisions that are made at the beginning of a project will carry a big toll during the entire life of the system. Poor decisions will be hard to overcome, as a lot of work will be needed to overhaul these “built from scratch” systemA big business with large teams will make fewer poor decision and those mistakes will be found out and fixed faster.

Messy hardware that often breaks easily. Due to the fact that people who build DIY implementations will go for rapid prototyping and functionality over robustness, sensor and data logging setups built in this manner will usually lack the roughness of commercial implementations. While a business dedicated to data logging wants to build systems with adequate sensor housing, and durability for transport, with customer satisfaction in mind, a person who builds this for him or herself might be ok with having a lot of exposed boards and cables. Overall DIY setups are therefore less robust, more likely to break, and more likely to suffer from electrical issues like poorly grounded circuitry.

Hopefully, the above pros and cons give you a useful idea of what you’re gaining and losing when you decide to build your own custom-built data logging system for hydroponics. While you will usually get much more flexible, lower cost, cohesive and personalized setups from custom building, this will usually come at the cost of higher support costs in time,

lower reliability, lower build quality, and compromises in quality depending on where your strengths as a builder/coder are. For small setups, it is usually a no-brainer to go with a custom setup – because of how much you learn from doing this and how much you can experiment – while for larger setups careful consideration of the above cons is important.

Five common mistakes people make when formulating hydroponic nutrients

It is not very difficult to create a basic DIY hydroponic formulation; the raw salts are available at a very low cost, and the target concentrations for the different nutrients can be found online. My nutrient calculator – HydroBuddy – contains large amounts of pre-made formulations in its database that you can use as a base for your first custom hydroponic endeavors. However, there are some common mistakes that are made when formulating hydroponic nutrients that can seriously hurt your chances of success when creating a hydroponic recipe of your own. In this post I will be going through the 5 mistakes I see most often and tell you why these can seriously hurt your chances of success.

Failing to account for the water that will be used. A very common mistake when formulating nutrients is to ignore the composition of the water that you will be using and how your hydroponic formulation needs to account for that. If your water contains a lot of calcium or magnesium then you will need to adjust your formulation to use less of these nutrients. It is also important not to trust an analysis

report from your water company but to do a water analysis yourself, since water analysis reports from your water company might not be up to date or might not cover the exact water source your water is coming from. It is also important to do several analyses per year in order to account for variations in the water composition due to temperature (which can be big). Other substances, such as carbonates and silicates also need to be taken into account in your formulation as these will affect the pH and chemical behavior of your hydroponic solution.



Failing to account for substances needed to adjust the pH of the hydroponic solution. When a hydroponic solution is prepared, the pH of the solution will often need to be adjusted to a pH that is within an acceptable range in hydroponics (often 5.8-6.2). This is commonly achieved by adding acid since when tap/well water is used, a substantial amount of carbonates and/or silicates will need to be neutralized. Depending on the salt choices made for the recipe, adjustments could still be needed even if RO water is used. Since these adjustments most commonly use phosphoric acid, not accounting for them can often cause solutions to become very P rich with time, causing problems with the

absorption of other nutrients, especially Zn and Cu. A nutrient formulation should account for the pH corrections that will be required and properly adjust the concentration of nutrients so that they will reach the proper targets considering these additions.

Iron is chelated but manganese is not. It is quite common in hydroponics for people to formulate nutrients where Fe is chelated with EDTA and/or DTPA but manganese sources are not chelated at all, often added from sulfates. Since manganese has a high affinity for these chelating agents as well, it will take some of these chelating agents from the Fe and then cause Fe phosphates to precipitate in concentrated solutions. To avoid this problem, many nutrient solutions in A/B configurations that do not chelate their Mn will have the Fe in the A solution and then the other micronutrients in the B solution. This can be problematic as it implies the Fe/other micro ratios will change if different stages with different A/B proportions are used through the crop cycle. In order to avoid this issue, always make sure all the micronutrients are chelated.

Not properly considering the ammonium/nitrate ratio. Nitrogen coming from nitrate and nitrogen coming from ammonium are completely different chemically and absorbed very differently by plants. While plants can live with solutions with concentrations of nitrogen coming from nitrate as high as 200-250ppm, they will face substantial toxicity issues with solutions that contain ammonium at only a fraction of this concentration. It is therefore quite important to ensure that you’re adding the proper sources of nitrogen and that the ratio of ammonium to nitrate is in the ideal range for the plants that you’re growing. When in doubt, plants can survive quite well with only nitrogen from nitrate, so you can completely eliminate any additional sources of ammonium. Note that urea, provides nitrogen that is converted to nitrogen from ammonium, so avoid using urea as a fertilizer in

hydroponic.

Not considering the media composition and contributions. When growing in hydroponic systems, the media can play a significant role in providing nutrients to the hydroponic crop and different media types will provide nutrients very differently. A saturated media extract (SME) analysis will give you an idea of what the media can contribute and you can therefore adjust your nutrient solution to account for some of the things that the media will be putting into the solution. There are sadly no broad rules of thumb for this as the contributions from the media will depend on how the media was pretreated and how/if it was amended. It will often be the case that untreated coco will require formulations with significantly lower K, while buffered/treated coco might not require this. Some peat moss providers also heavily amend their media with dolomite/limestone, which substantially changes Ca/Mg requirements, as the root system