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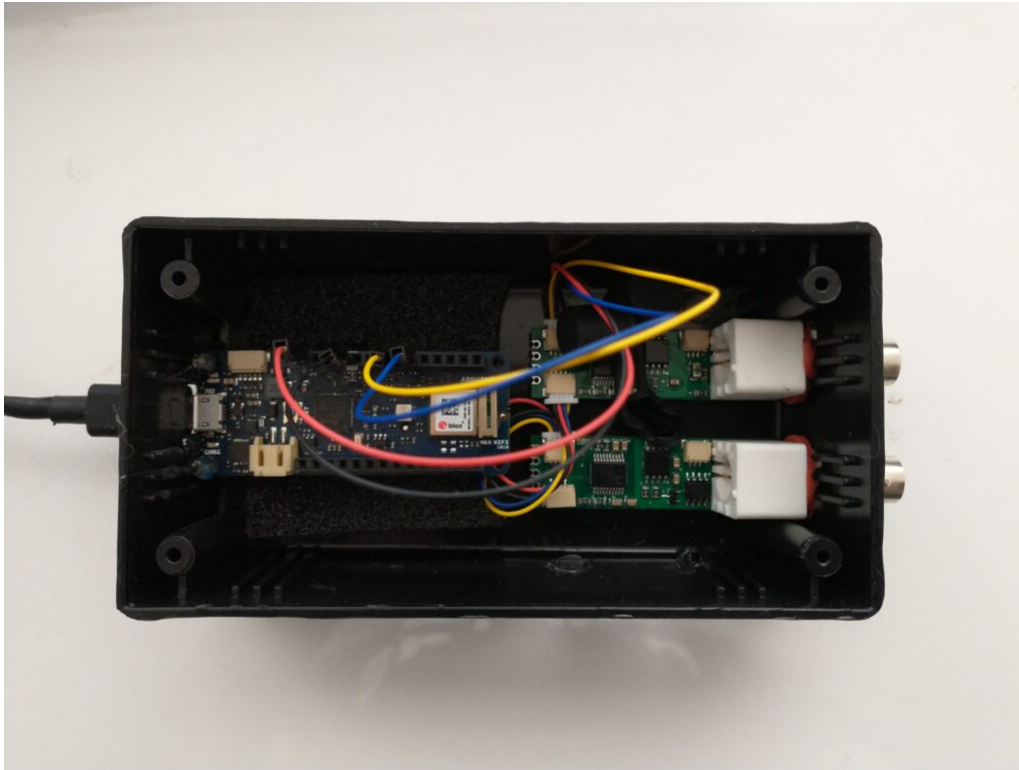
My Arduino MKR wifi 1010 based sensing station, using uFire pH and EC boards in a small project box.

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.

A simple cheatsheet for macro nutrient additions in hydroponics

In hydroponic growing, we are often faced with the need to adjust the nutrient concentrations of a fertilizer reservoir or foliar spray directly, in order to increase the quantity of some nutrient by a specific amount. Although you can use a program like HydroBuddy in order to quickly calculate these

values, it is often the case that these calculations need to be done in the field or in a growing environment, and a computer to calculate things is not at hand. For this reason, I have created a small "cheat sheet" that you can use in order to figure out the amounts of salts that you would need to add to a solution to increase any of the macronutrients by 10 ppm.

Salt Name	ppm	Element	ppm	Element	g/L	g/gal
Calcium nitrate (ag grade)	10	N (NO3-)	13.19	Ca	0.0694	0.2629
MAP	10	N (NH4+)	22.1	P	0.0821	0.3108
Ammonium Sulfate	10	N (NH4+)	11.4	S	0.0472	0.1785
Gypsum	10	Ca	7.99	S	0.0430	0.1626
Calcium Chloride	10	Ca	17.69	Cl	0.0277	0.1048
Magnesium Nitrate Hexahydrate	10	N (NO3-)	8.67	Mg	0.0915	0.3463
Epsom Salt	10	Mg	13.19	S	0.1014	0.3839
Magnesium Chloride	10	Mg	29.16	Cl	0.0392	0.1483
AgSil 16H	10	Si	10.9	K	0.0411	0.1554
MKP	10	P	12.62	K	0.0439	0.1663
Potassium Nitrate	10	N (NO3-)	27.87	K	0.0730	0.2763
Potassium Sulfate	10	K	4.10	S	0.0223	0.0844
Potassium Chloride	10	K	9.067	Cl	0.0191	0.0722

Cheatsheet for macronutrient additions in hydroponics

With the above cheatsheet, you can quickly evaluate some of the most common options you would have to increase all the different macronutrients in a hydroponic or foliar solution by 10 ppm and which secondary elemental contributions you would get from these additions. For example, if you add 0.0694g/L of Calcium Nitrate, this would add 10ppm of Nitrogen as nitrate plus 13.19ppm of Calcium. Careful consideration of secondary contributions need to be taken into account, especially when using salts that contain elements that can be toxic, such as

chlorides.

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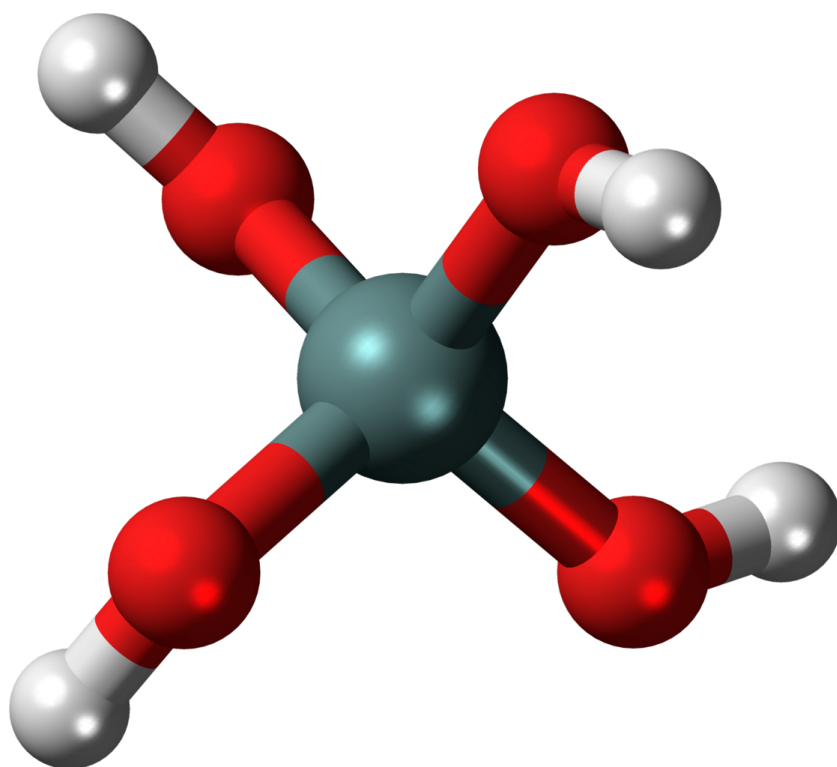
Model representation of orthosilicic acid, also called mono-silicic acid.

[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$_2$O)](#)
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.

HydroBuddy coming to Android, free and open source!

The [Hydrobuddy open source hydroponic nutrient calculator](#) – which has been used extensively by both professional and amateur growers for the past 11 years – is finally making the leap to the Android platform. This is a big leap, as many growers – especially in developing countries – lack access to a PC but have easy access to Android phones. Thanks to the effort of the [LAMW development team](#) – who made the development of Android apps using Lazarus possible – I have been able to recode and port HydroBuddy to mobile and will release it for free and with NO ads within the next couple of weeks within the Google Playstore. The [source code](#) is already available on github and will be updated as the application development

process continues.
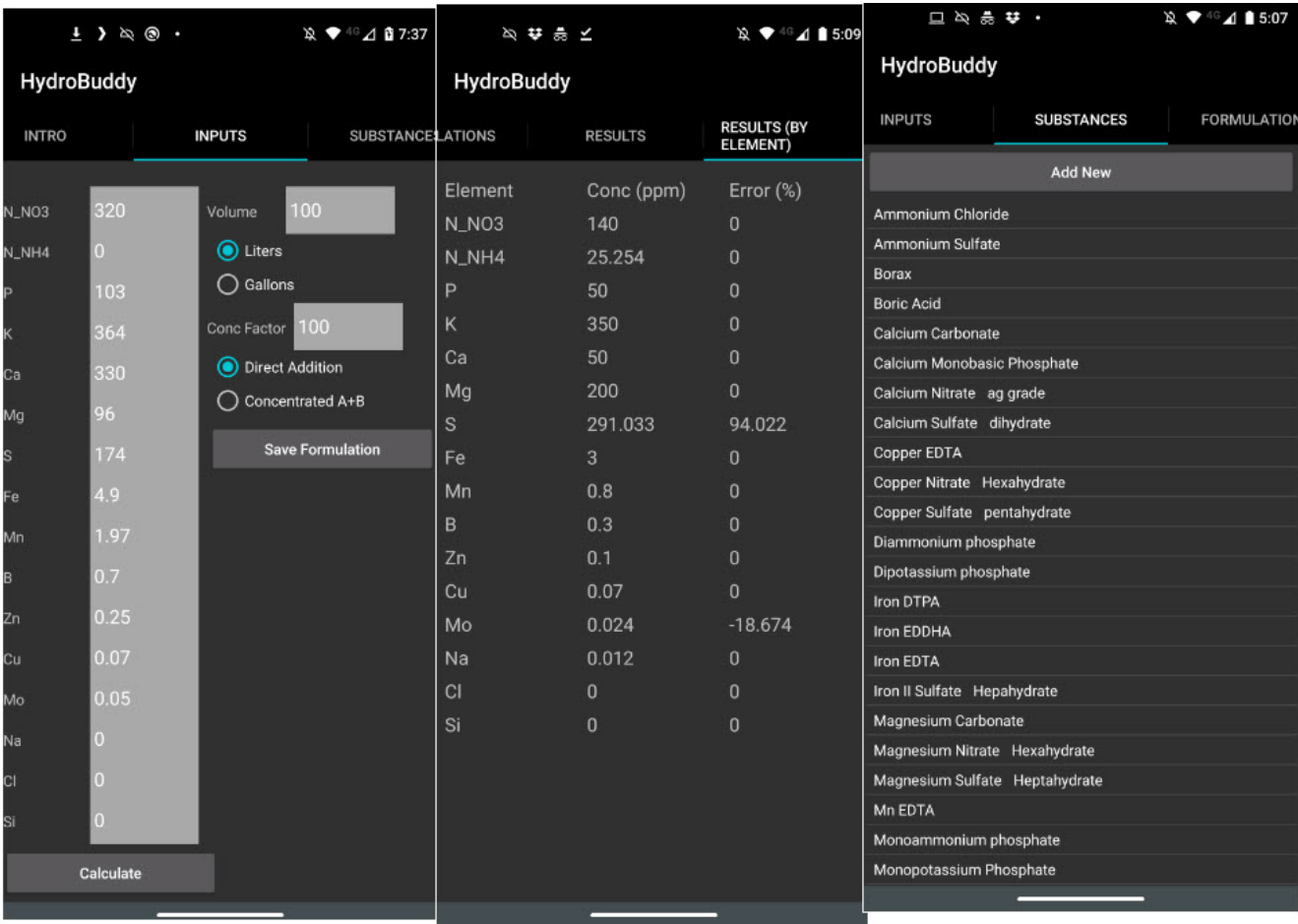
Some screenshots of the current testing release of the HydroBuddy Android app.

The HydroBuddy Android application will include most of the features present in the desktop application. It will include all the same substances and formulations that are included in the desktop application and will allow you to perform the same types of calculations you do on the regular application. It is however true that some functionality will be missing initially. The "Copy Commercial Nutrient Formulation" section has not been implemented yet, neither have the "Water Quality Parameters", EC estimations, or calculation of instrumental errors. The "Concentration from weights" and "set weight" calculations are also missing at this point in time.

Right now, the HydroBuddy application is in its internal testing stage, which means that only people who I add to the testing group are able to download and use the application. **If**

you want to participate in this testing phase, you can download HydroBuddy in your Android device through [this link](#). If you participate in this process, please share meaningful feedback about the application with me. *This feedback can be left as a comment in this post as well.*

The port to Android is expected to be a stepping stone towards HydroBuddy v2.0, which will include a complete recoding of several portions of the calculator. This will be done in order to implement a more flexible database structure that allows for more effective saving and sharing of the inputs and outputs of the program. The aim of this is to allow growers of all origins: hobbyists, industry professionals, or researchers, to exchange complete calculation sets and allow for a much more profound use of the calculator as a community-building tool.

Sadly there is no iOS version planned, since I do not own, or plan to own, any Apple devices, and therefore cannot program/compile/test code in this platform. However, the licensing terms of both the desktop and mobile versions of the application do allow anyone to port and publish the application in iOS, provided it is non-commercial and released for free, with no ads and under the same licensing terms (GPLv2).

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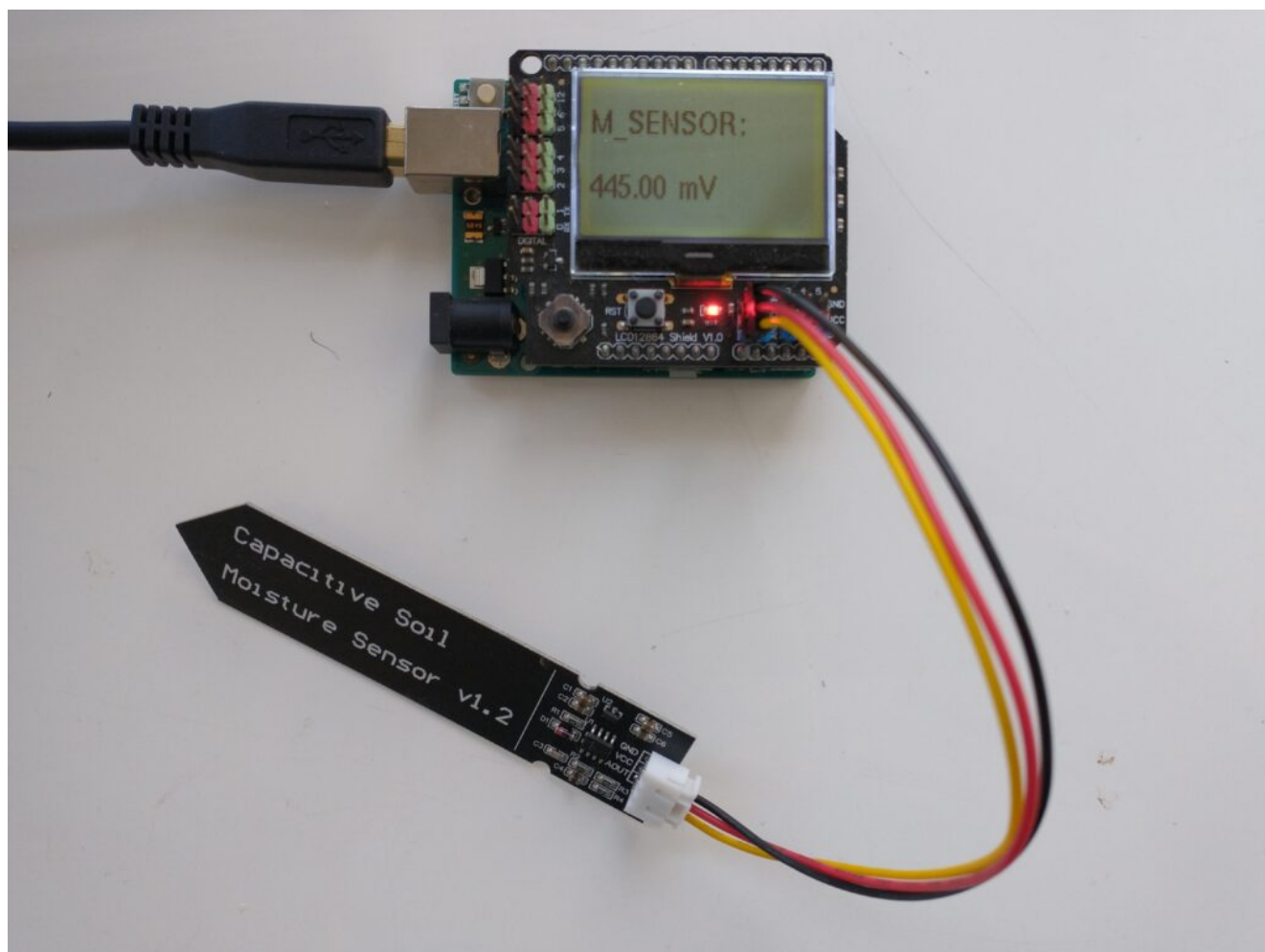
A capacitive sensor plugged into an LCD shield and an Arduino UNO. The measurements are very easy to track with this setup.

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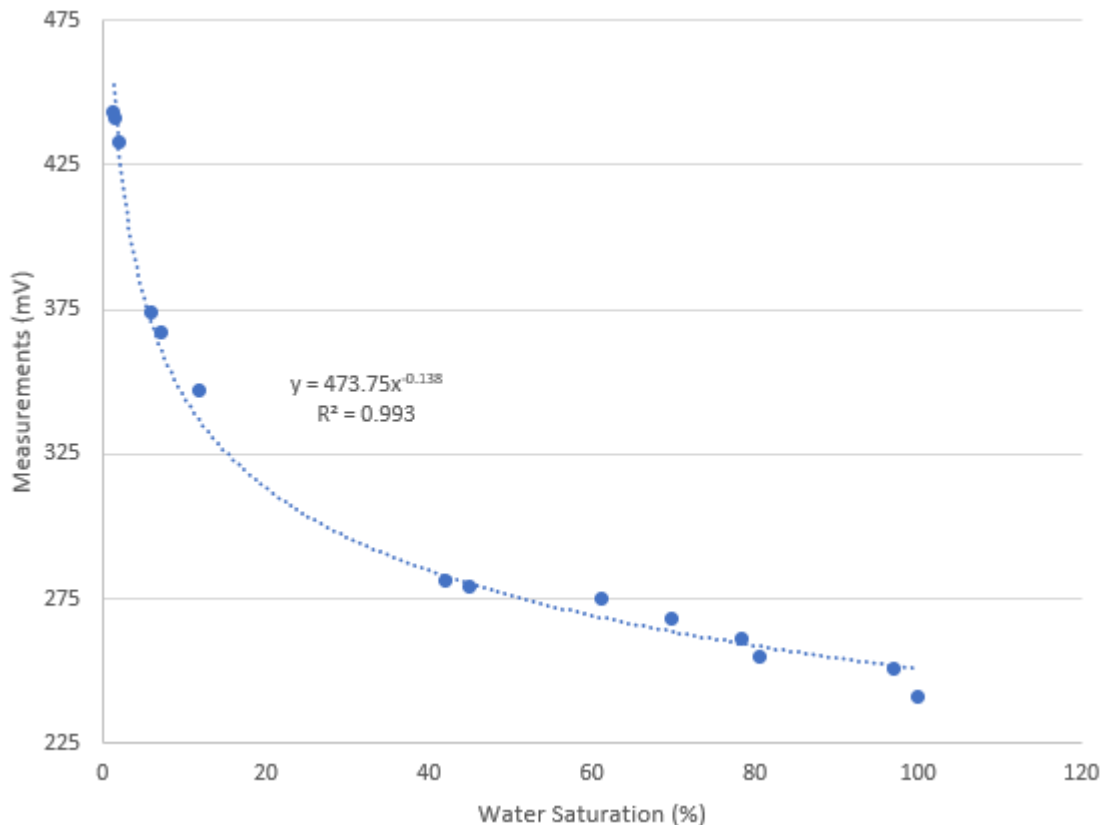
In order to do this procedure in an easier manner, losing as little accuracy as possible, I have created a calibration procedure that makes use of natural drying and only requires one single oven drying step. The procedure is as follows:

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.

The chart shows Measurements (mV) on the y-axis (from 225 to 475) versus Water Saturation (%) on the x-axis (from 0 to 120), with the fitted equation:

$$y = 473.75x^{-0.138}$$
$$R^2 = 0.993$$

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.

Properly positioning temperature and humidity sensors in a hydroponic growing environment

Temperature and humidity are two key variables you need to measure. They are important because they determine how your plants will transport water, and transpiration controls a lot of processes, with nutrient transport being one of the most important ones. However, the value of these variables in a growing environment – being that a greenhouse or a grow room – can change substantially depending on where they are measured. It is therefore critical to know where to place sensors and how to interpret their readings based on their location. In this post, we are going to discuss where it is best to measure these variables and what consequences it could have if these values are not measured properly.
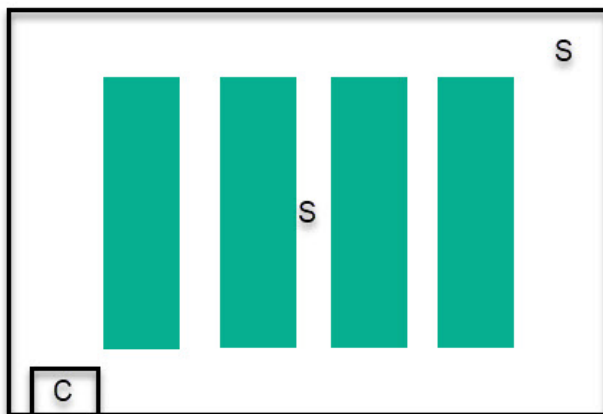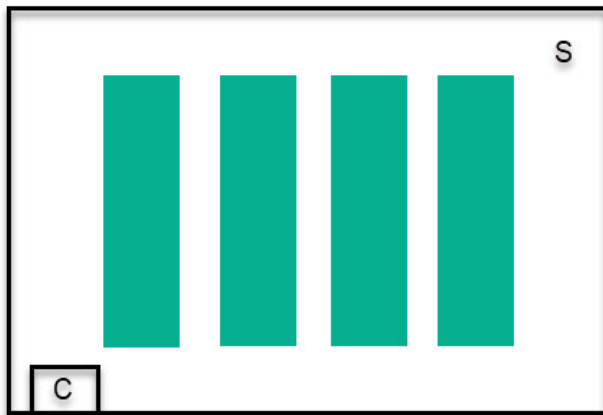
Sensor placement relative to a control source (AC, humidifier/dehumidifier) for one or two sensors. Note that this setup assumes good circulation throughout the room, including middle of canopy.

Let's start with the worst possible case, you only have one set of sensors and you need to control your environment with it. *In this case, place your temperature and humidity sensors at canopy height, as far away as possible from both the AC ducts and the humidifier/dehumidifier, make sure the sensors are hanging in the air and not stuck to a wall or tubing.* Then, make sure you use a hot wire anemometer to verify that your air movement speed is at least 0.3m/s across the entire room. This setup ensures that the worst controlled part of the room is at the correct value and it also attempts to minimize the gradient created from the control sources to the sensor using a good amount of air circulation. It is not perfect though and significantly different "climate zones" will be created close to and far away from the climate control devices.

The above setup can be used effectively in small growing environments, but can be problematic as both the number of plants and the size of the growing environment increases. At this point, using a single set of sensors is not an option if adequate climate control is desired. In these cases, multiple sensors need to be placed to ensure that climate control is being done properly. When using multiple sensors, place the second sensor at the place with the lowest air circulation inside the room, at canopy height, which is usually in the middle of the room, then place subsequence sensors as far away from either this or the first sensor in sequence. When doing climate control, the system needs to ensure all of the sensors remain within a "safety band", making sure no sensor becomes too cold/hot, humid/dry, during control cycles.
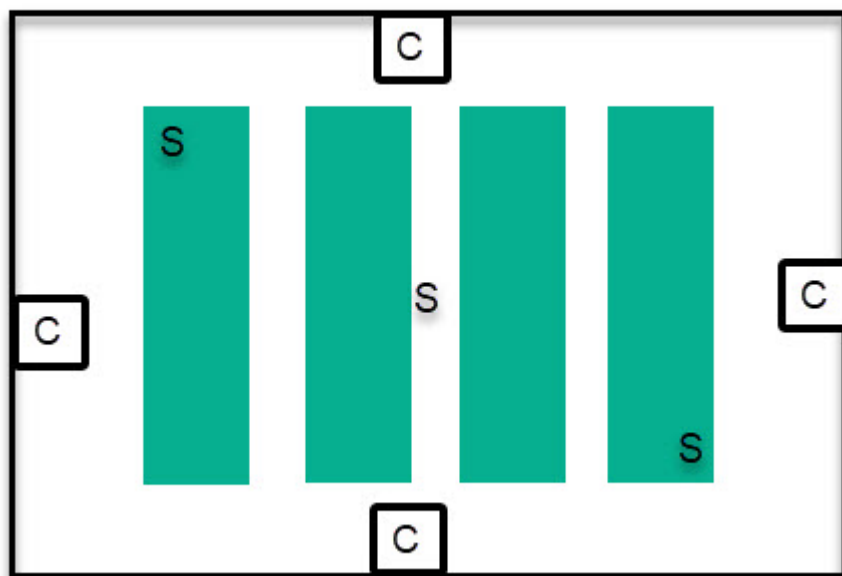
Sensor placement for multiple control sources. Sensors are placed in order trying to always be as far away from sensors as possible but within the plant canopy.

When you implement a sensor system like this, you will realize pretty quickly that climate becomes very hard to control in a larger room when there is only one source of control (one AC, one humidifier, and one dehumidifier) because gradients become too big for effective control, so it takes too long for the AC

to be able to properly control the room while ensuring all sensors remain within proper boundaries. In this case, it becomes necessary to add multiple sources of control, so that the extent of gradients within the room can be minimized. This means adding multiple ducts for the output of an AC, multiple humidifier/dehumidifiers, etc.

In these cases, sources of control are placed outside of the plant canopy to avoid plants being exposed to the flows from these control sources (which would expose them to very cold/hot/dry/humid air). Then the sensors need to be placed within the plant canopy, starting from as far away as possible from all sources of control — usually the middle of the canopy — and then to the corners of the growing environment.

Note that the control algorithm needs to ensure all of these sensors are within the proper control band and not attempt to control the average reading of these sensors. If you try to use the average of sensors to control a room, you might be left with a room where two extremes are present and the control system believes everything is ok while these extremes are maintained. The median is a better way to control a room, but it only becomes useful when 5 or more sensors are used. If only 2 or 3 sensors are used, ensuring all of these sensors are within adequate bands is fundamental to ensure that the room will have a lower chance of having humidity/temperature microclimates that will be detrimental to plant growth.

Making the most out of your hydroponic setup's logged

sensor and control data

If you have a hydroponic crop with a data logging and automated control solution, you probably have a lot of sensor and control data recorded that could be useful to take your crop's results to the next level. In this post, I am going to talk about some things that you could be doing with these stored data. You will see how the usage of this data opens up many possibilities and that even implementing the most basic of these suggestions could lead to important improvements in your understanding of your crop and its results.
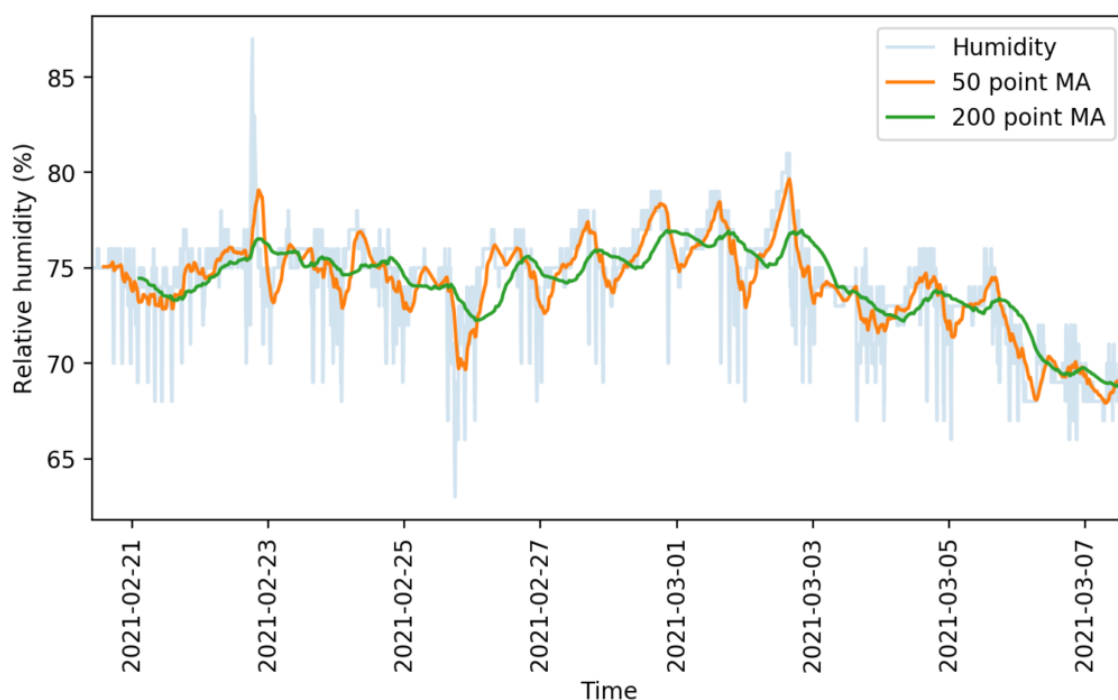

Use of different moving average to smooth out sensor readings.

The lowest hanging fruit to take advantage of logged data is to be able to download the data and put it into a database structure that you can properly query and search. Most data logging solutions record the data in either very simple structures, like csv files, or non-relational databases — like MongoDB — which are rather limited and do not allow for the degree of versatility that a true relational database engine offers. Having the data in a properly built database will allow you to start using it in a creative way. For example,

with the data in a proper database, it becomes possible to create a custom data visualization that can help you understand what's going on inside your growing environment.

The images in this post show you some examples of this. The first one shows a simple example where a rather noisy humidity sensor is smoothed out using different moving averages, these averages can then be used to implement more effective control algorithms. The second image shows a detailed map of the temperature and humidity values experienced in a room, colored by the hour of the day. We can use this plot to easily locate where problematic times and VPD conditions might be, just by looking at when extreme readings happen. This behavior would be harder to observe and diagnose on a regular VPD Vs Time plot. Regular data logging web interfaces and platforms will not allow you to create plots of this sort, which is why putting your data into a proper DB and manipulating it to create custom visualizations can be very powerful.
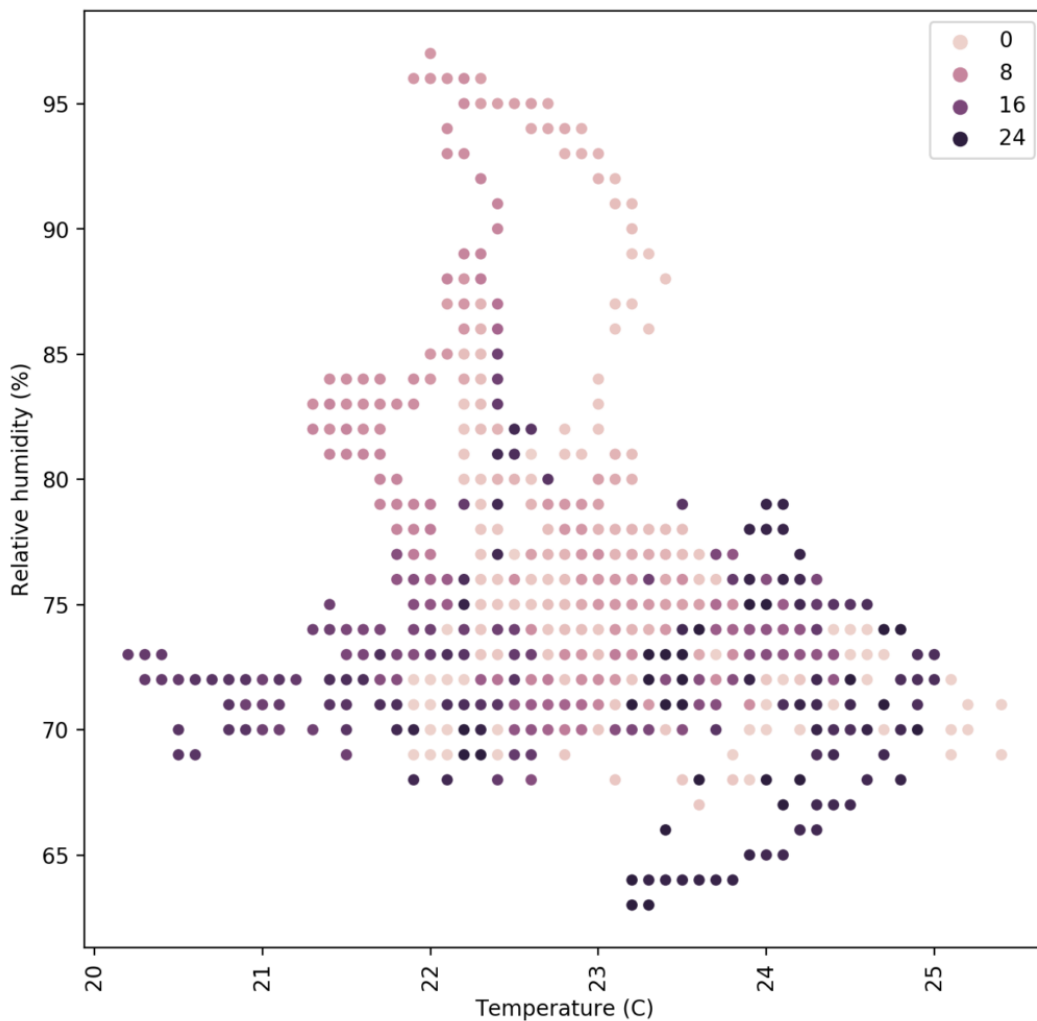
Relative Humidity Vs Temperature map colored as a function of the hour of the day for a growing room being constantly monitored

The most powerful uses of the data come into play when you actually piece together your control and sensor data. Say you have an AC system coupled with a temperature sensor but you have a lot of other temperature and humidity readings and you also know the age of your plants at each point in time. Using this, you can create an advanced control algorithm where a system will use all of this additional information to know when to trigger AC systems and dehumidifiers to control the environment. Having a lot of logged data from a set point control system is a great starting point to train a reinforcement learning algorithm for climate control, since we know which control actions were taken at each point in time and we know the effect these had. Implementing such control mechanisms can lead to control systems that avoid spikes in

humidity and temperature across light on/off cycles, greatly smoothing out the environmental transitions for your crops.

Finally, there is also the potential to improve yields by gathering detailed mappings of yield data in a room and relating these yields with environmental sensors. If you have several different sensors in a room and you know the yield that you obtained on a per-plant basis, then you can create a map of all the yields in a room in order to see if there are important disparities in your yields because of differences in local humidity, temperature, light or air circulation levels. This can lead to important insights that can help better adjust climate conditions for the entire grow room. If multiple rooms are available, the information about environmental sensor data can be related to yields in order to stir all rooms towards more favorable conditions.

For example, after analyzing yield and temperature data from multiple growing cycles of one of my customers, we realized that the greenhouse with the lowest temperature standard deviations between sensors was giving the best yields, we then implemented better control algorithms on the other greenhouses to prevent this from happening, obtaining significantly better results across the board after that.

Data is a treasure. If you have been recording judicious sensor, climate control, and yield data through time, you're probably sitting on a gold mine that you haven't exploited yet. If you're interested in using my help to do so, please consider [booking an hour of consultation](#) time with me so that we can discuss your needs and how we could leverage your data to improve your growing results.

Commercial sensor and data logging solutions for hydroponics

On a previous post, I discussed a very interesting open-source sensor/data logging alternative for Hydroponics called [MyCodo](#), which offers a lot of features and flexibility for those growers with the time and skills necessary to implement their own sensor and data logging setup. However, many growers don't have the time to do this on their own — or the time and willingness to hire someone to do it for them — and all they want is a solution that "just works" out of the box and that fits most of their data logging needs. In this post I am going to talk about three commercial solutions — in no particular order — that I've had experience with along with some of the advantages and disadvantages that each one offers you. **Note that this post has not been sponsored by any of these brands**. *The statements below represent my opinions on the matter and the facts, to the best of my knowledge. I recommend you contact each company to ask specific questions pertinent to your needs.*

[Growtronix](#). This company offers a complete solution for monitoring and automation of hydroponic crops. Their sensors are hooked through cabled connections and they support a wide array of analogue sensors, both sold by them and by third parties. As long as a sensor can work on a 3.5-5V input and give an analogue reading, it can be installed in a growtronix setup. Their web interface is user-friendly, it allows you to

view sensor readings and create control schemes using simple if logic statements. They have also shared the source code of their web interface with some of my customers in the past, so if you would like to customize things beyond their base web application, I'm sure you could figure it out if you have the time and programming skills. Growtronix support — per the experience of the customers I have you have used it — has been stellar.

There are however some downsides to using growtronix. Since everything is cabled you will need to lay cables across your rooms if you want to hook up multiple sensors within them. The system lacks support for third party i2c sensors, meaning that you can only connect analogue sensors and will miss on some interesting third-party sensor offerings. The data is also stored in a non-relational mongoDB implementation, which means that querying data and doing complicated data analysis will not be easy with them. Their control algorithm technology is also rather simple, to the best of my knowledge they do not offer more advanced control mechanisms beyond the if logic statements they allow the users to program.

[Agrowtek](#). Similar to Growtronix, they also offer a complete monitoring and automation solution for hydroponic crops. However, they offer their own touchscreen computers to connect to their sensors, dosing pumps, and relay modules, so they do not have a dedicated web interface for their sensors that is hosted on any computer but you must purchase their own. Their "GrowControl" panels will hook with normal ethernet cables to any of the sensors they offer and you will be able to program all the behavior of the sensors and the relays from these stations. Their main advantage is easy setup, everything easily hooks up and you can then program things within the

GrowControl panels to fit whatever simple control needs you might have. You can probably setup 200 sensors/relays in a day to control an entire facility using this setup. Their custom computer also gives you more stability, meaning crashes of the system are rare (according to the customers I have who have used them). From the three companies discussed in this post, this is also the only one to offer nutrient injection systems in their offering.

However, one big limitation of this company is how closed the ecosystem is. You have absolutely no ability to hook up third-party sensors and sadly their offering lacks some important and basic sensors for a medium to large scale hydroponic setup, specifically water content and water potential sensors. You are also becoming reliant on the availability of support from them and – if the company went under – it would be very hard for you to be able to fix or find replacements for their sensors or their control panels. Their control algorithms are also fairly simple and are limited to basic if-logic, similar to the Growtronix system. Data is also not logged into any database but as basic csv files, which means substantial effort will be needed to perform advanced data analysis tasks.

[SmartBeeControllers](). This company also offers a complete automation and monitoring solution for your hydroponic crop. Their main differentiating factor relative to the last two is that sensor stations connect wirelessly to your computer, allowing you to place sensors throughout your facility without having to set up cables through the entire place. Their sensor stations can hook up to a large number of sensors so, for example, you can use a water content station to hook up six of their capacitive water content sensors. They also require a computer server with the web software to communicate with – alike Growtronix – and their software has a focus on simplicity. In this case, control options are even more limited than in other cases, with basically only simple set-

point logic available to control relays (to the best of my knowledge).

The SmartBee ecosystem is also quite limited and offers no pH/EC/ORP sensors or water potential sensors (tensiometers). You have no ability to hook up third-party sensors as well, meaning you're stuck with this offering if you use them. Because of the wireless nature of communications, sensor readings and their stability can also be compromised due to excessive electromagnetic noise, which can be particularly problematic in a short room that has a lot of HPS ballasts. It is also true that in the past (2-3 years ago) their support seemed to have problems, with several complaints about their response time online. I do not know if their technical support has improved so I would advise you to seek recent opinions about it on social media if you're considering them for purchase. The people I know who used them didn't need to contact support, so I cannot comment on this aspect from my customers' experience.

The above are three commercially available data logging systems for hydroponics. All of them should be easy to hook up and should provide you with basic data logging and control capabilities for your grow. In my opinion, the most complete one is Growtronix, given the ability to add third-party sensors — even if only analogue ones — and the quality of their sensors and web application software. However, if controlling the nutrient injection process electronically is important for your situation, then Agrowtek might be a better solution. None of them however provide advanced control mechanisms — like reinforcement learning-based climate control — and none of them provide access to all sensors that would be desirable, so a custom DIY setup might be best if these features are very important to you.

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.
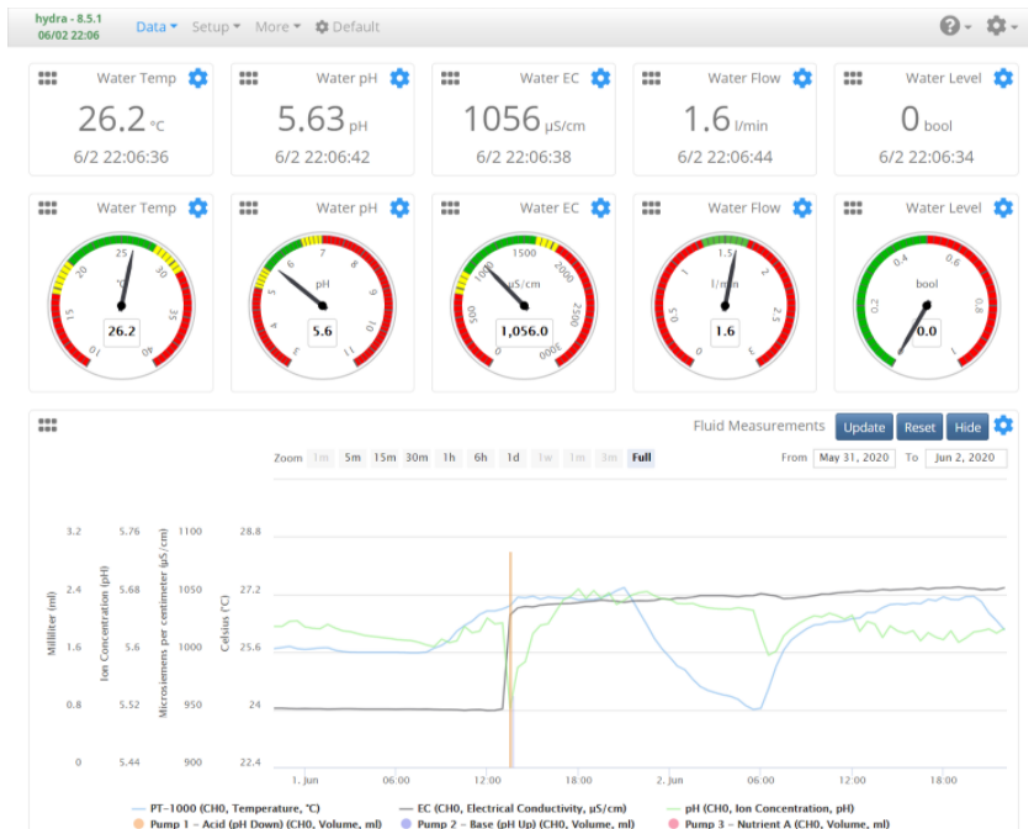
Sample control panel image taken from the MyCodo github project website

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.
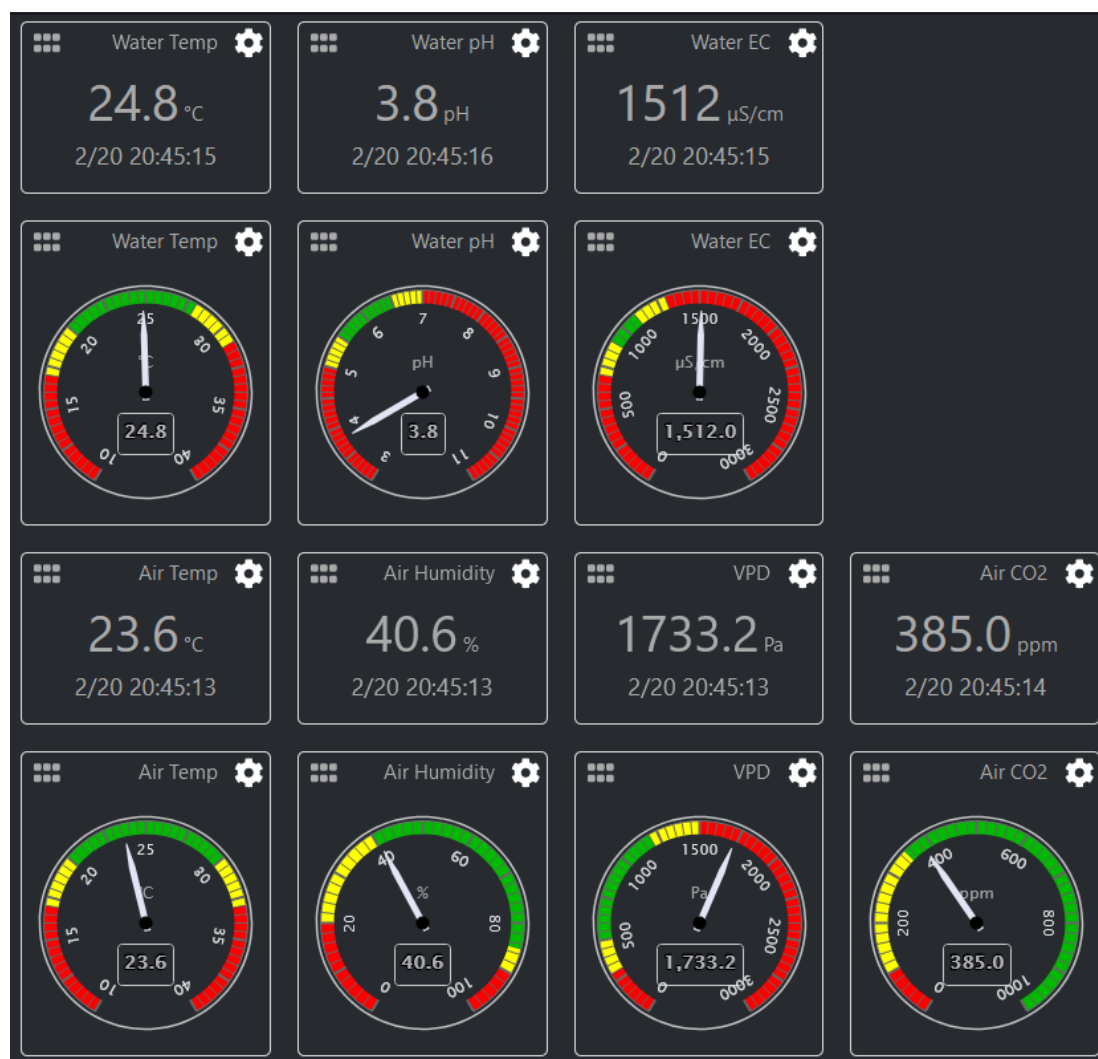
Another sample MyCodo panel

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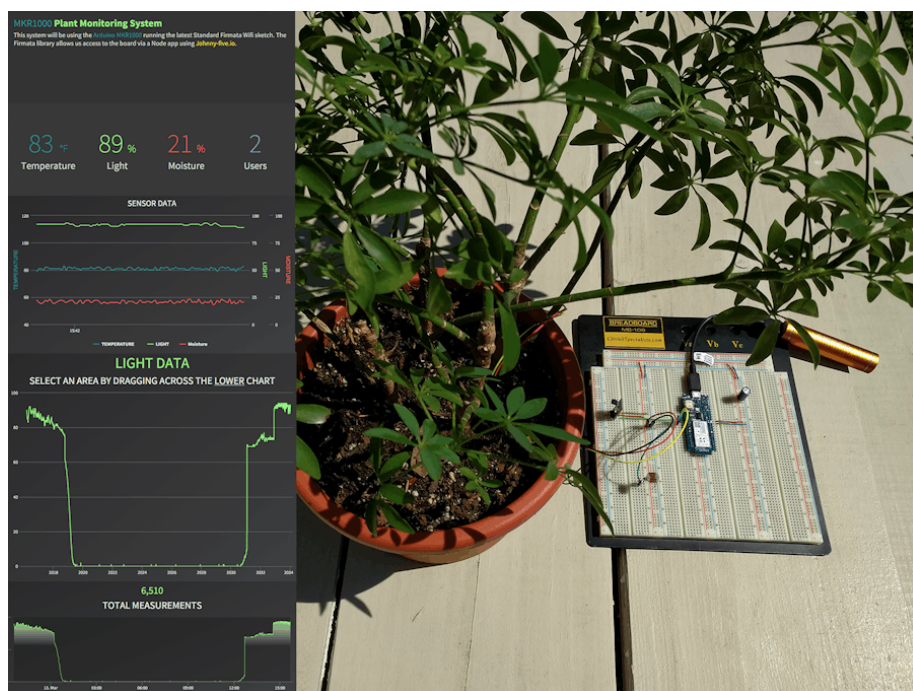
A simple plant monitoring custom built system. Read more here.

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.