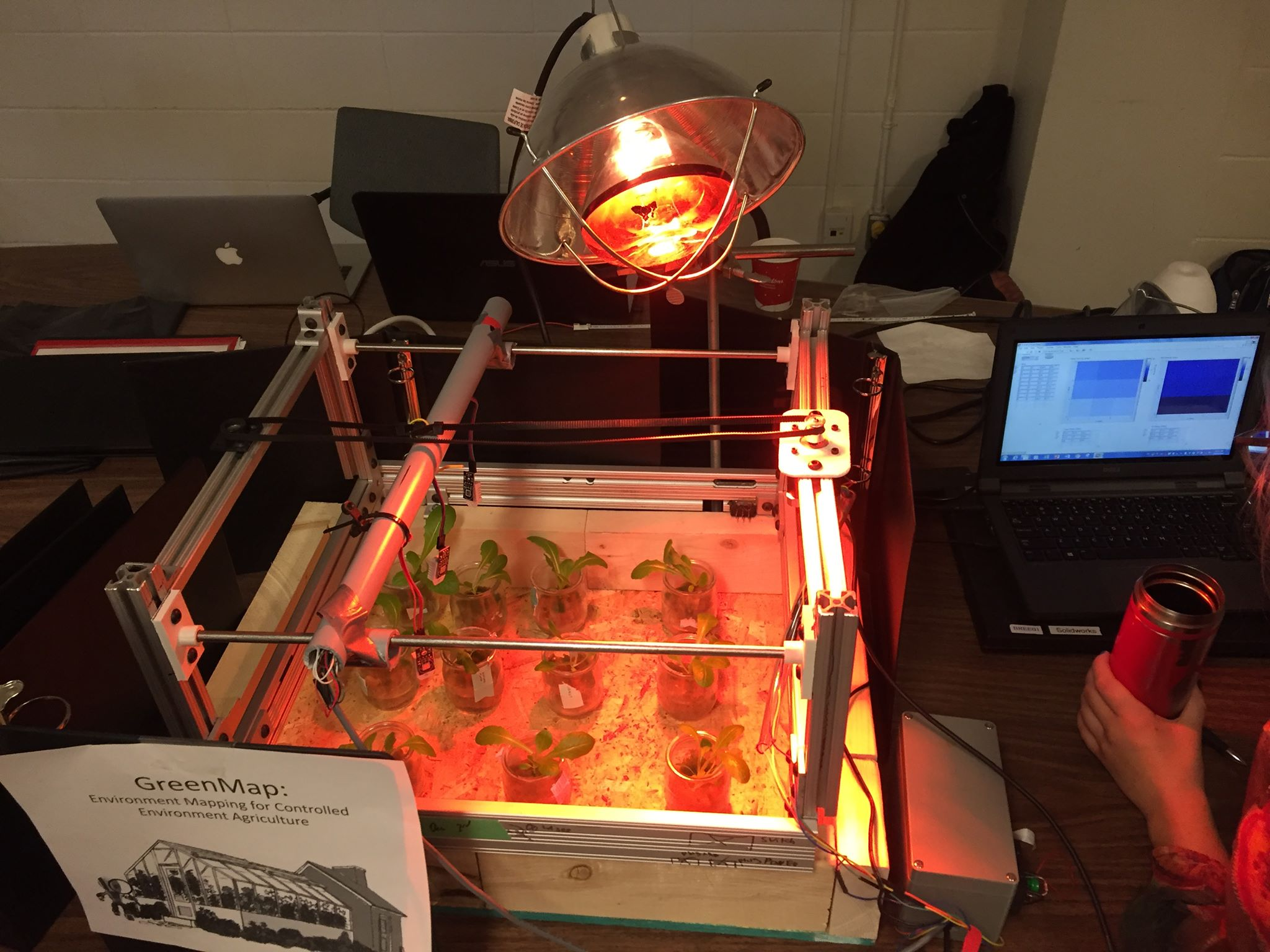
**GREEN MAP**

*Digital Mapping for Controlled Environment Agriculture*



Instrumentation & Control

McGill University

*DL , SM, PO, RW, JW*

Table of Contents

[**Introduction**](#_snz4qdez4m7v) **3**

[**Methods**](#_5oygbpueocww) **4**

[Sensors](#_sdikpqje95mx) 4

[Mechanical Structure & Wiring](#_3p65bn44iacl) 5

[2.2.1 Calibration](#_c83mfwvda1ad) 6

[Arduino Code](#_2i5iuxuyyknm)  6

[Labview Code](#_jru5s59x466e) 8

[**3. Results**](#_6ya85up5l0ge) **11**

[**4. Analysis**](#_17loto5du0v) **13**

[**5. Discussion**](#_2k5ut7oo588r) **13**

[**6. Conclusion**](#_y657tzvwwmc1) **15**

**7. References 16**

**8.Appendices 18**

# Introduction

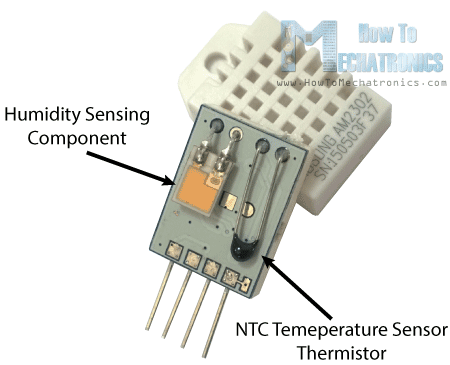
In order to meet food demand in Quebec, the agricultural sector is shifting towards year-round local food production in greenhouses. In Montreal alone, the main greenhouse tomato producer titled “Les Serres Sagami” (owned by Stephen Roy) produces roughly 240,000 kg on a weekly basis (Mark Cardwell, 2015). What is most remarkable is that Les Serres Sagami is able to maintain production over the winter season. Mr. Roy attributes his success both in production and quality to the advancement in technology. For heating alone, he was able to reduce the greenhouse consumption from 400,000 to 15,000 cubic meters of natural gas annually (Mark Cardwell, 2015). Following this trend, this projects purpose is to develop a LabView and Arduino code to monitor and display sensory data about relative humidity and temperature in a greenhouse quickly and for an affordable price.

Hydroponic systems are the preferred method to produce at a large scale in greenhouses. However, to have a profitable return on investment, these systems must be large, which increases the incidence of key growth parameters being skewed from their optimal values. Additionally, to increase biomass production, there must be an increase in light output which often leads to a local increase in temperature in a certain area of the system, and without proper ventilation the formation of hotspots can be found within the hydroponic system. Hotspots can lead to plant death if left unattended and also leads to unwanted pathogen growth. To fix hotspots and increase yield and uniformity, one must know where these points of stagnation are, which requires extensive data mapping of temperature, relative humidity, light intensity, CO₂ concentration, air speed, etc. Achieving accurate information for the entire system would require a large number of sensors, which in turn greatly increases the cost associated with indoor hydroponic systems with artificial lighting.

Following the above mentioned statement, this paper discusses the digital mapping of a controlled environment agricultural system a group of bioresource engineer undergrad and graduate students developed. This scaled down prototype consolidates data on a X and Y scale, which is processed using LabView and an Arduino code to plot a 2D heat map based on humidity and temperature readings from a set of DHT22 sensors, where X and Y will be the dimension of the surface mapped. The first demonstration of this method has proven the concept of this tool to be relatively cost effective, and provides an live feed data of the system. (Arduino Project Hub, 2018)

# Methods

## Sensors

This system uses 3 DHT22 sensors, which measure temperature and relative humidity (RH). These sensors are comprised of a resistive humidity sensor, a NTC thermistor and an integrated circuit (IC). The temperature and measurement range are respectively, -40-125 °C ± 0.5 °C and 1-100% ± 2-10% (Dejan, n.d.). The temperature range over a 24-hour period in a greenhouse ideally range between about 18 °C to 24 °C with humidity ranging from about 50-70%. So, DHT22 sensors will comfortably cover the environment range in question. These sensors were purchased from ABRA Electronics, and cost about $10 each, making them very affordable for such a project.

The working principle of both temperature and humidity will be outlined. Resistive humidity sensors are used which measure the electrical conductivity, or resistance, across a non-metal substance between two electrodes (Dejan, n.d.). As the humidity changes in the environment, the non-metal substance will absorb or release water vapor, affecting its conductivity of the material. This material is often a solid polyelectrolyte or conductive polymer (Anusha, 2017). The measured resistance is outputted to the IC, which processes the data and sends the digital output to the microcontroller, in our case an Arduino Uno. The relationship between the RH of the environment and the resistivity will vary based on the material, but generally there is an inverse exponential relationship between the resistance and RH (Fujimoto, Maeda. 2000).

The NTC thermistor used to measure temperature has a similar working principle to that of the RH sensor described above. They are made of ceramic or polymer, whose resistance changes in relation to the ambient temperature (ResistorGuide, n.d.). NTC stands for ‘negative sensor coefficient,’ indicating the resistance decreases as the temperature increases, which can be seen on any characteristic NTC curve. This characteristic curve is non-linear and can be approximated with a variety of equations, although the most precise is the Steinhart-Hart equation. Like the aforementioned humidity sensor, within the DHT22 sensor, the resistance is measured and the IC then processes this data and sends the digital data to the microcontroller (Dejan, n.d.).

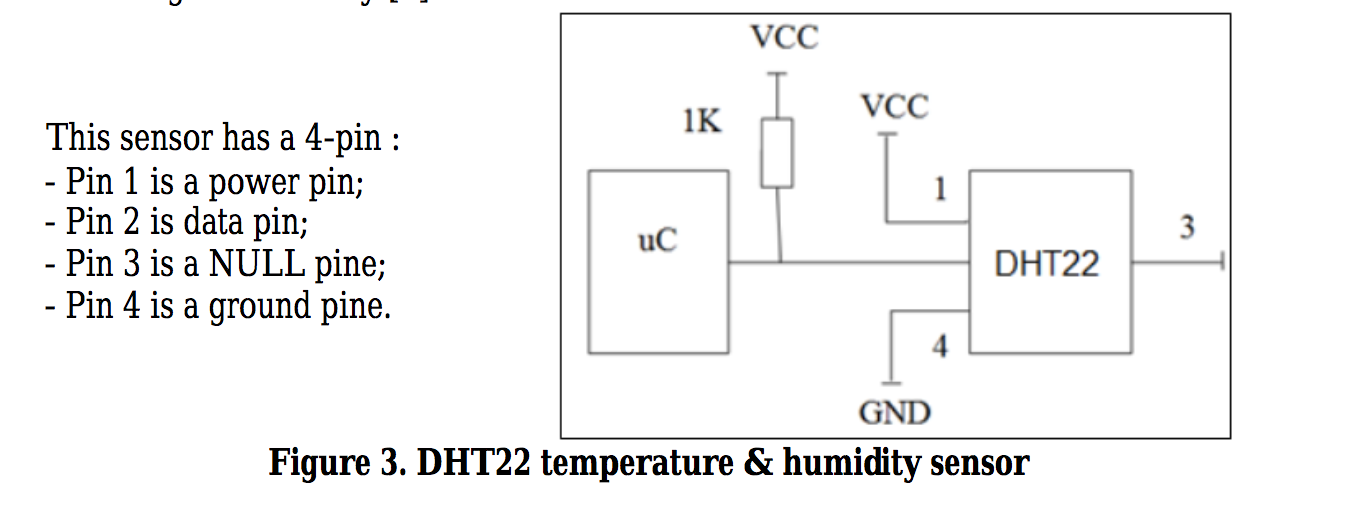
DHT22 sensors are used often for their low cost and generally good accuracy. The maximum sampling rate of the sensor is 0.5 Hz, yielding 1 sample every 2 seconds, which should especially be taken into account whenever data is taken within short time windows. In this system, the arduino is used to communicate the data between the sensors and output them to LabView, where it is mapped in 2-D. The arduino also transposes this data into an array to ease the communication with labview.

## Mechanical Structure & Wiring

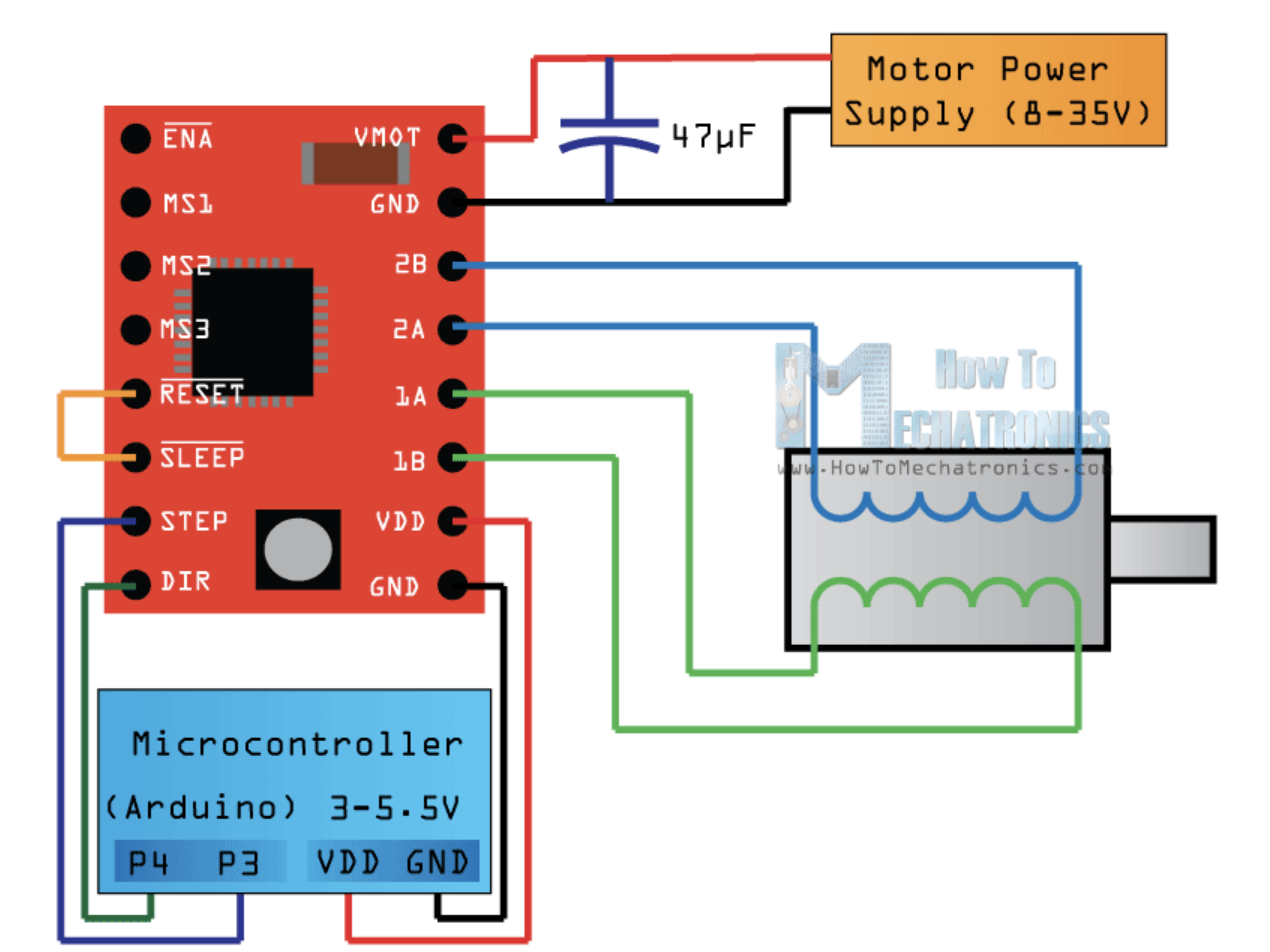
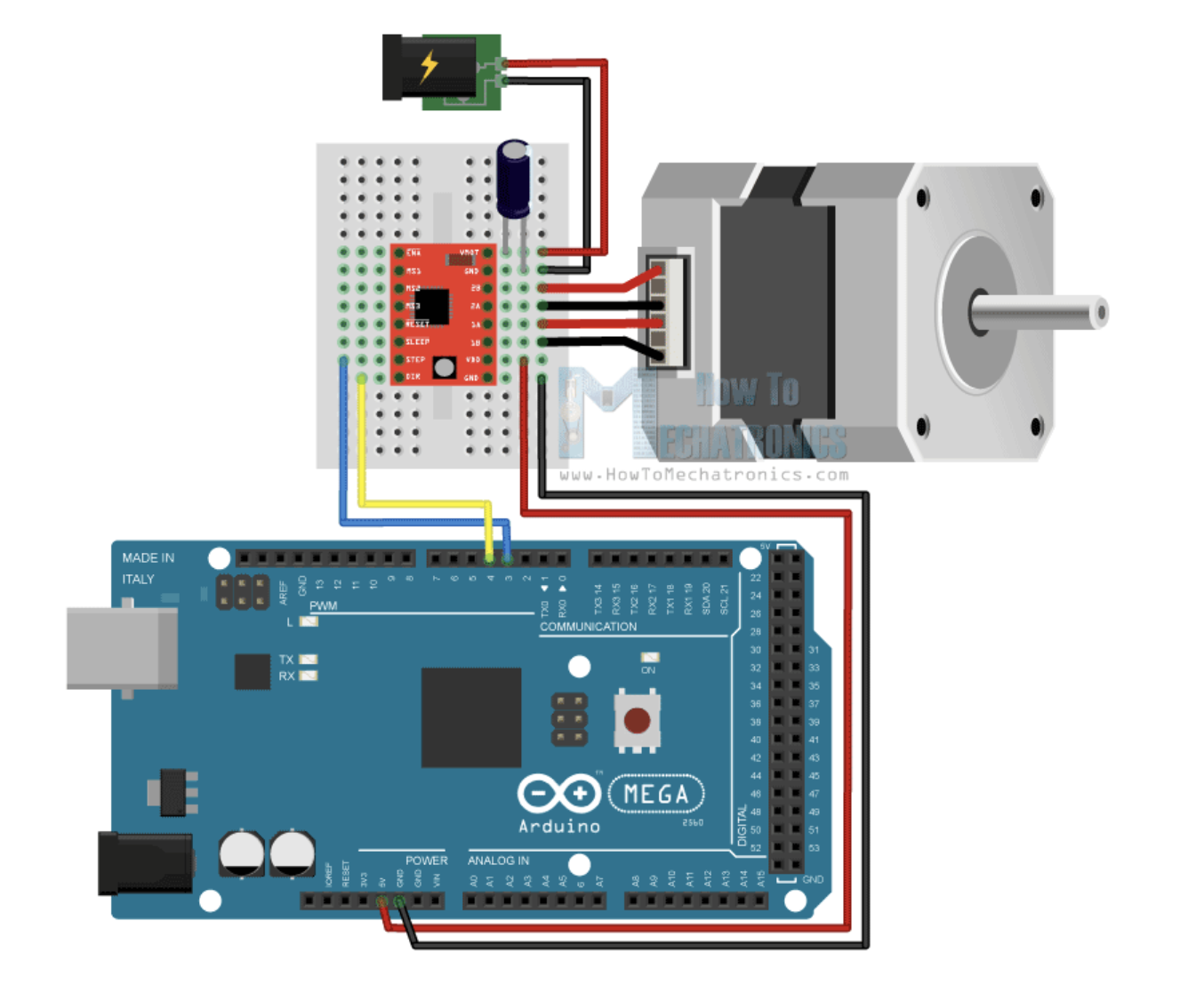
The physical design of this prototype starts with a wooden base frame made from plywood, cedar and screws. The dimensions are 2 feet by 1.5 feet with a ¼ in thickness plywood frame. The cedar sideboards are 2.5 inch thick to ensure maximal stability. In order to obtain a sweeping motion over our system, 1 inch steel bars were constructed in a square frame (1.5 x 1 feet) 3 fashion at both ends and connected together by ¼ steel roads (1.5 feet long) . The frame and steel road was attached and secured in place using nuts and bolts.

The GreenMap runs on a track which is powered by a stepper motor and timing belt. The sensors were attached to three points across a pvc pipe which was secured to the timing belt, allowing the sensors to sweep across the system. We made the sensors stop at intervals of 10 cm across the track so that they could take a reading, the time of reading depending on what we needed the sensors to do and is described below. The GreenMap sensor system is activated button activated, one push initializes the motor and sensors which move to three locations across the system and return to its initial position.

The connection and wiring for the DHT22 can be seen in figure 1. While the wiring for the motor to the arduino can be seen from figure 3. A microstepping drive was used due to its built in translator, which made the operation of the motor far easier, and the wiring for the microstepping drive can be seen from figure 2.



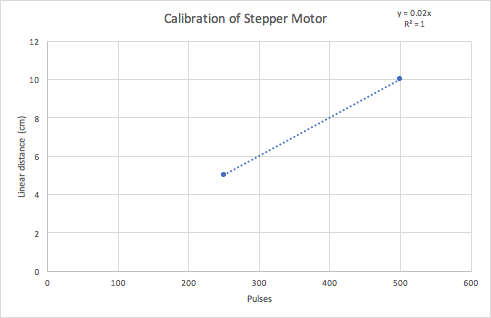
*Figure 1: DHT22 Wiring*



*Figure 2: Wiring of A4988 driver Figure 3: Wiring of stepper motor to arduino*

### 2.2.1 Calibration

Calibration of the stepper motor was performed by inputting different pulse values into the motor and measuring the resulting distance the sensor bar moved. This resulted in a calibration curve with a slope of 0.02 and y-intercept of zero.



*Figure 4 Calibration curve of a linear movement induced by a stepper motor*

## Arduino Code

The DHT22 sensors and their movement were programmed with an Arduino Uno board, the full code is available in appendix.

Since the project aim was to prove the concept of a data acquisition tool for hydroponic system, this code is not very flexible. Many parameters are set constant, to reduce the chances of error and simplify the code. Some functions were added to help troubleshooting the code, their operation is conditional upon the user making them true prior to running the code. This code reads 3 DHT22 sensors, activates a stepper motor and outputs the serial data with the position of the temperature and humidity readings.

### 2.3.1 Libraries

Firstly, libraries must be imported, the most important of which is the “DHT.h” library, which allows the arduino to read from DHT22 sensors. For this code, extra libraries were included, like the “RTClib.h” for potential time keeping of the measures using a Real Time Clock, and the “Wire.h” library, which is necessary for many sensors to establish I2C communication.

### 2.3.2 Definitions of pins and variables

Next the global variables to be used within the code and the corresponding pins on the Arduino board are defined. The LOG\_INTERVAL is used to set the delay between the readings. It is set as an unsigned long, because when using a delay function, the delay given is interpreted as an integer, which is limited to the value 32,767, therefore limiting the delay to 32,7 seconds. When using an unsigned long, the variable size is extended and stores 32 bits, giving a maximum value of 4,294,967,295 which we considered large enough for the delay we expect to use.

Since this project uses 3 sensors, it is important that each of them gets a unique output, to avoid over writing previous output from another sensor. These 3 sensors must be initialized before setup.

## 2.3.3 Setup

In the setup part of the code, the pins are defined as inputs or outputs and the pins relevant to the stepper motor are set to low, to avoid the motor running during setup. The PRINT\_LABEL condition is an example of a troubleshoot function, it would print a label in the serial monitor so that the user could tell if the sensors are reading properly.

## 2.3.4 Void Loop

The trigger to the sensors taking the readings and moving in the systems is a button, which would change the state of the program from 0, which is inactive to 1, which is active moves the motor and takes readings.

So when the state is 1, the operations begins. The sensors move from their rest position (theoretically out of the system) to the first reading position (which is set to 0). The sequence is the following, using user defined functions: move, wait, read, log position and data.

As mentioned before, the code is hard-coded and inflexible. To read at 3 different points the operations were repeated 3 times and the restart function places the sensors to their original position. The positions are all define within the code as 0, 1 and 2.

This loop is running continuously, and best practice is to not stop it completely using delay functions. For that reason, we initially used a function to count time between readings using the “millis” function, but this strategy sometime lead to a skip of the readings, so we chose to use the delay functions instead with which we never skipped a reading. Hence the commented out “if” function surrounding every reading sequence.

The Serial Output had to be edited so that it could be read efficiently with LabView. Basically each line has the following outline : X position, Y position, Temperature, Relative humidity. The X position is relevant to the position set by the stepper motor and the Y position relates to a unique sensor on the moving stick, each of the sensors being at a different position on the Y axis.

## 2.3.5 Definition of the timing of sensors

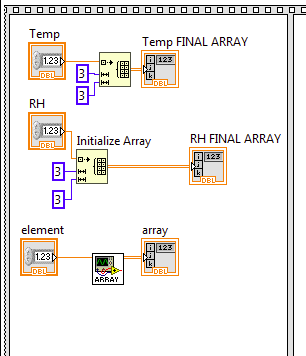
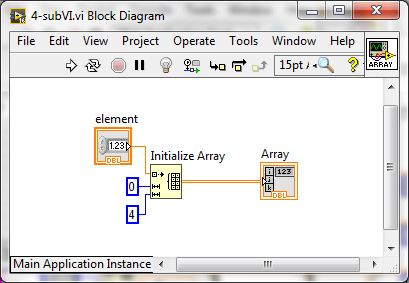
The DHT22 as a response time of 5 seconds in forced air convection. Hence allowing the sensor stand still for at least 5 seconds would allow an acceptable reading. For that reason the delay between a measure was set to 7 seconds. (Aosong Electronics, 2018)

Having the sensors be still at the measuring point was key for this tool to have accurate measures of temperature and relative humidity. However, having a longer time delay in between measurements reduced the frequency of serial data input, which would cause timeout error in LabView if the delay was longer than 7 seconds, and 5 seconds for the first delay.

## Main VI Application

The LabVIEW code used in this project has the goal to (1) read input serial data from an Arduino, (2) transform that data into two separate arrays (for Temperature and Relative Humidity) and (3) produce three output intensity graphs. The LabVIEW code was created as follows

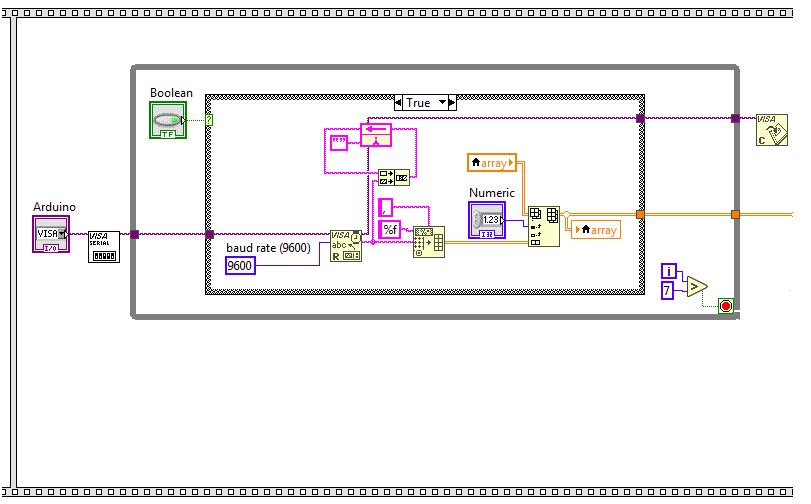
Section 1: Initializing Arrays

*Figure 5.1 Initializing Arrays in Front Panel Figure 5.2 SubVI Front Panel*

* In the first frame of the Flat Sequence Structure, three arrays are created. The “Temp FINAL ARRAY” and “RH FINAL ARRAY” are 3x3 arrays which will eventually hold Temperature and Relative Humidity data, respectively, and are created using the “Initialize Array” function.
* The “array” is the 0x4 array that will hold *all* the 4 types of incoming data from Arduino: x-location, y-location, temperature, relative humidity. This array is created using a **sub-VI Routine** which initializes a 0x4 array. The array has 0 rows since these will be populated in Section 2 and initializing *with* rows results in the final array having excess rows.
* These initial creations of the arrays to be used later ensures that arrays are erased of previous information when new trials are run.

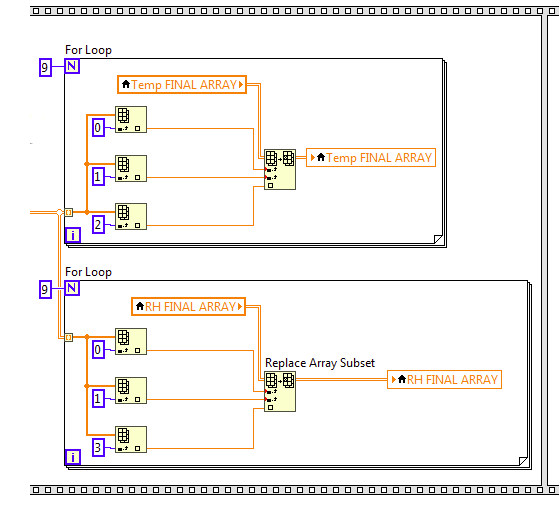
Section 2: Read Input Serial Data from an Arduino



*Figure 6 Reading Serial Data in Front Panel*

* In the first part of the second frame of the Flat Sequence Structure, input serial data is read from arduino and set to a matrix.
* The beginning section of this code was taken from lab 3. This section opens the Arduino serial port (selected manually on the Block Diagram), reads the serial information at a baud rate of 9600, and closes the Visa port after the While Loop is completed.
* The While Loop allows the computer to accept data until the scan of the sensors has been completed, or for 7 iterations. Technically, it should be accepting data for 3 iterations, however the time delay often resulted in problems and through trial and error we found 7 iterations allowed LabVIEW to read the correct amount of data from Arduino: not too much, and not too little.
* The Case Structure allows the user to specify if the reading should occur or not, through a Boolean switch.
  + The “true” case was coded using Nasir (2014) as a reference. The serial data from the Arduino is concatenated, then this data is separated based on commas, and compiled into an array. The incoming data is in sets of 3 rows and is added onto the array with the “Replace Array Subset” function
  + When the case is “false”, the output array is simply blank.
* After the While Loop has run through, the final “array” is sent to create two seperate arrays

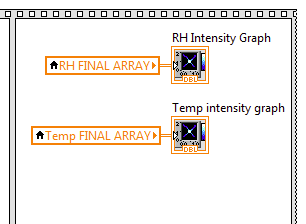
Section 3: Create Two Separate Arrays



*Figure 7 Creating Two Seperate Arrays in Front Panel*

* In the second part of the second frame of the Flat Sequence Structure, “array” data is split and sent to create seperate arrays for temperature and relative humidity.
* A For Loop is created for each, so that the array receives only 9 data points (since each is a 3x3 matrix). In each For Loop, the “Index Array” tool is used to select the 0th (x-location), 1st (y-location), 2nd (Temperature) *or* 3rd (Relative Humidity) columns of “array”.
* Using the “Replace Array Subset” tool, the column index is replaced by the x-location (0th column of “array”), the row index is replace by the y-location (1st column of “array”), and the actual value to show in the array is replaced by either temperature values or relative humidity values.

Section 4: Create Two Intensity Graphs



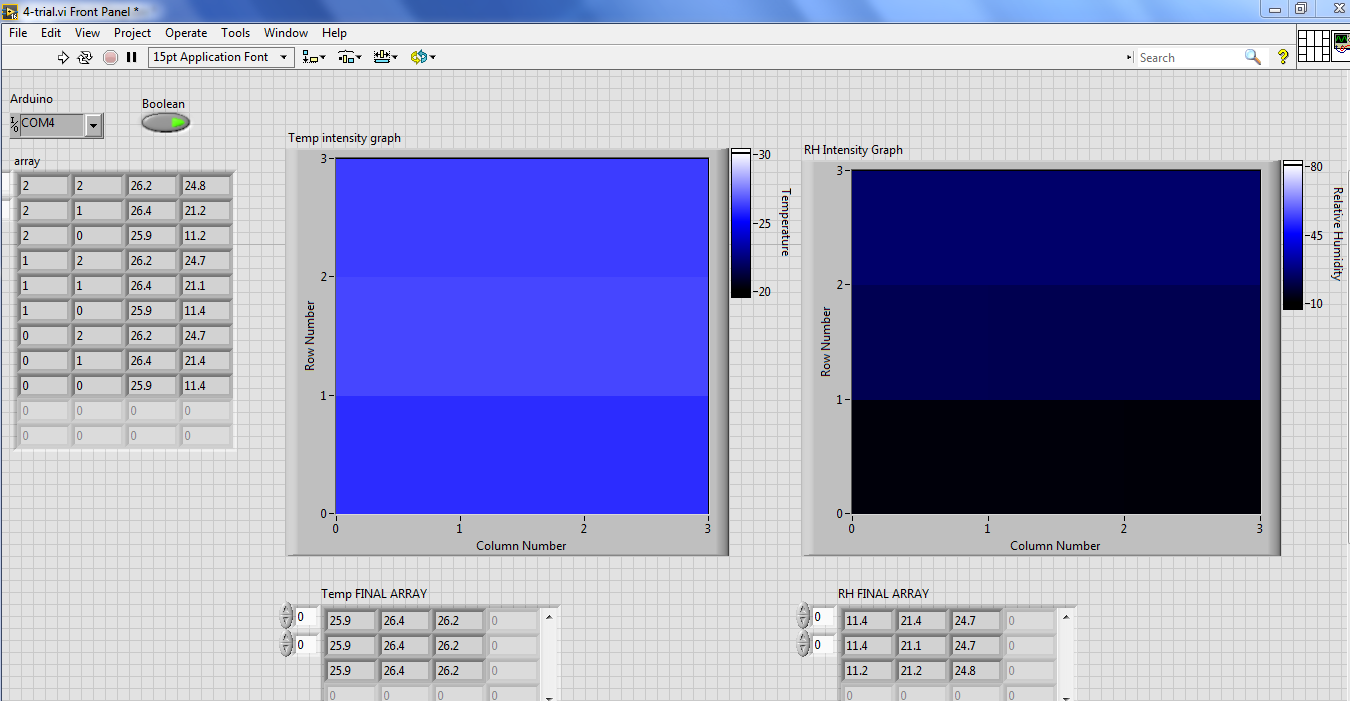
*Figure 8 Create Intensity Graphs in Front Panel*

* Finally, in the third frame of the Flat Sequence Structure, the two intensity graphs are created
* Each intensity graph simply accepts the array data from either “Temp FINAL ARRAY” or “RH FINAL ARRAY” and outputs a graph of different colours in each block of the matrix. The final scale and axes labels were changed manually on the Block Diagram. The output graphs display higher values in white and lower values in dark blue.

The final LabVIEW code, although very difficult to create, is rather simple which is more user-friendly and easier to understand, therefore better. A complete image of the Block Diagram and Front Panel are available in Appendix.

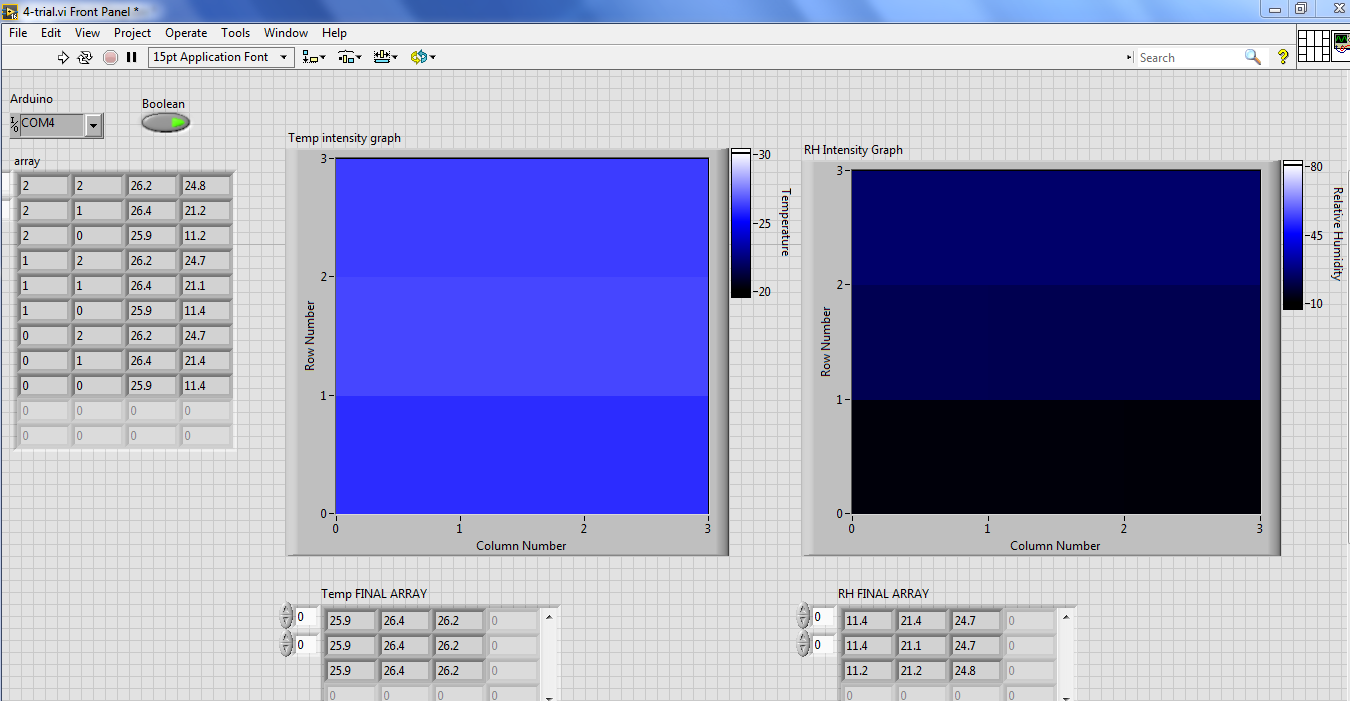
# 3. Results

As can be seen in the figure 9 to the right, the output from the arduino can be read as follows: X\_position, Y\_Position, Temperature, Humidity. Each location of the sweep outputted three lines of results, yielding a 9 X 4 array, to represent the 9 quadrants being examined.

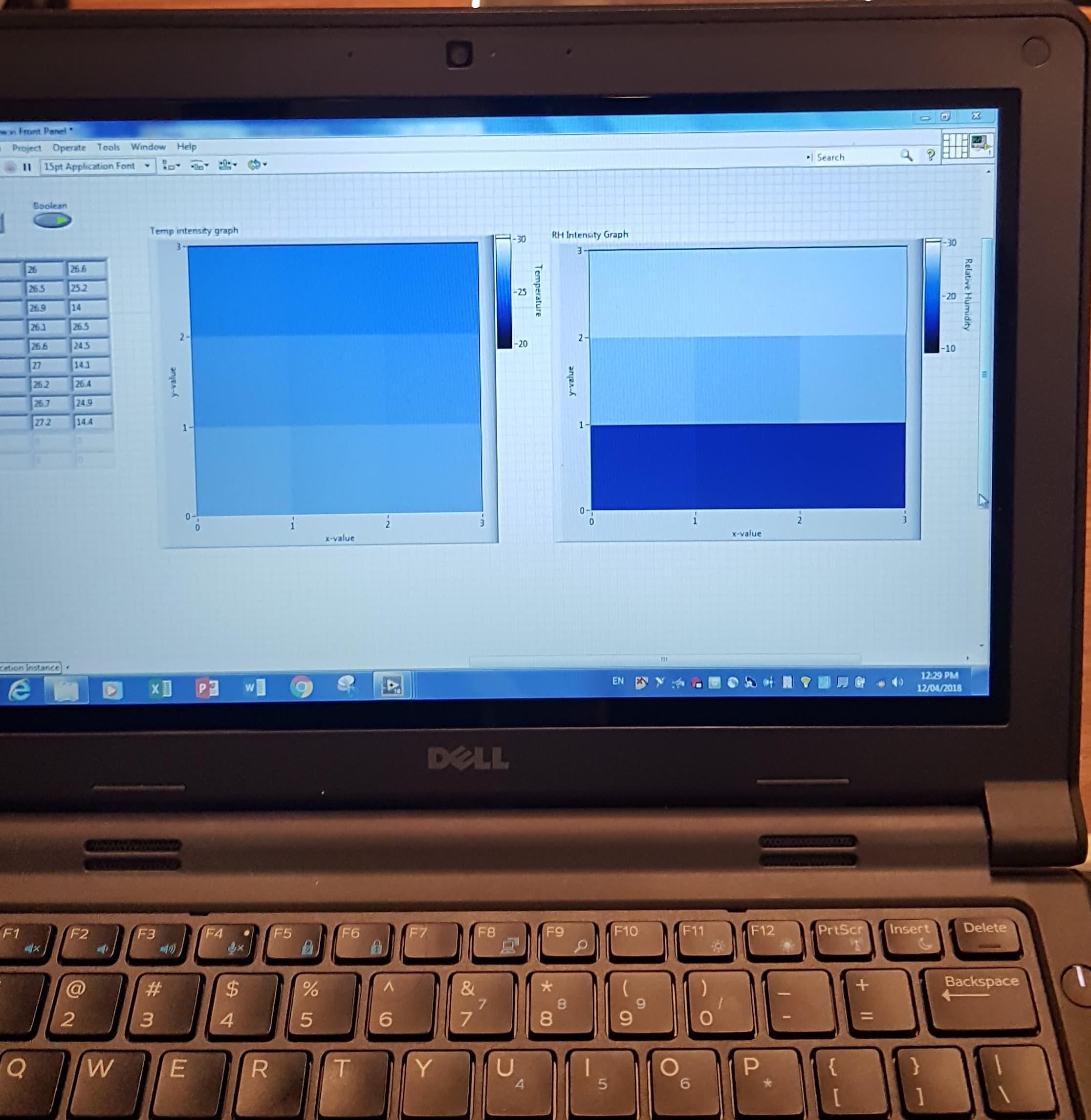


*Figure 9: Results in output array*

The produced maps can be seen below, in figures 10 and 11. The map on the left represents temperature, which that on the right RH.The darker the color, the lower the variable being measured at that particular location. The maps in figure XX was after the heat lamp was turned off and the temperature in the system was allowed to equalize. In figure XX, more variation can be seen as the heat lamp had been on.

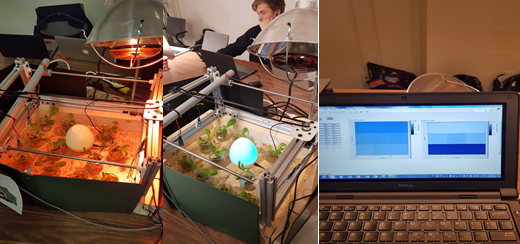


*Figure 10: Screenshot of mapped 2-D output, without heat lamp.*



*Figure 11: Photo of mapped 2-D output, with heat lamp*

During the demonstration, a humidifier and heat lamp were used to simulate different conditions in the demonstration hydroponic system.



# *Figure 12 System with the heat light on, humidifier and mapping output in LabView*

# 4. Analysis

We used a heat lamp to make one point hotter than the other points, however after approximately 5 minutes we noticed that the difference between the other sensors measurements and that of the area being heated up by the lamp were relatively the same with variations no greater than .5°C . Similar results also occurred for the humidity sensors, we placed a humidifier under one sensor so that sensor would show higher values for relative humidity then the others, once again after around 5 minutes the other sensors were giving readings similar to that of the sensor above the humidifier, which gave a reading around 30 % RH . We noticed that there was usually wide variation between the humidity sensors sometimes by more than 10% RH. This could be due to the nature and of the sensor measurement and it could also be due to the fact that the RH sensors were of cheaper value .

The data collected is as as precise as the sensors which is ±0.5 °C for the temperature rating ± 2-10% for the relative humidity. Calibration could be used to improve the quality and precision, and remove any unwanted noise in the data. This would be a future point of study for this project.

# 5. Discussion

The GreenMap worked on developing a heat and humidity map of our model hydroponic system. Although heat and humidity levels are very important for keeping a plant healthy there are a few more additions that would make the Green Map an ideal addition to the greenhouse/ hydroponics industry. Including a wavelength sensitive diode into our system would allow for mapping of the various wavelengths of light that the plants would be receiving. These operate on the p-n junction theory and work on a range of 450-950nm (First Sensor, 2018). These sensors would help make the growing of the plants for efficient as some plants react better to different wavelengths of light then others.

Another addition that would be useful to the system would be anemometers. One thing we noticed with our system was that once the heat lamp was heating up one spot of our model greenhouse long enough then the rest of the area would be around that same temperature. With air speed sensors we would be able to measure the flow of the air and make another map of the air is being distributed throughout the greenhouse. When placing the humidifier in our model greenhouse it was assumed that the humidity sensor above the humidifier would give the highest moisture content. This was not the case when the walls of the system were open, because ventilation would disperse the humidity. However, when the walls of the system were blocked, the humidity readings did show higher humidity where the humidifier was located. Thus if the tool could give a reading on air movement, the humidity sensors with the temperature and humidity sensors would create a much better map for our greenhouse as it would gives us the all important flows of air current throughout the system allowing us to care for the plants more effectively.

Scalability is a main concern for such a system to be successfully implemented in biomass production facilities. Ideally, the code would be less restrictive and allow a sensing of its position in the system, allowing for easier scaling. Another aspect of scalability is the physical system. To use this identical mechanical system would require large motors, and could also impede flow of traffic due to the bar sweeping over many rows. This could be remedied by using individual motors for each sensor and have them run on their own mechanical track, or just produce a 1-D map, across the length of the greenhouse. This would not result in as much information as the 2-D system, but would give far more information than the currently used stationary sensor.

One important note is that we adhered completely to our projects proposal. We proposed a system that would use labview to plot a 2D heat map and humidity map by using DH22 sensors with an Arduino uno. It also had the two signal inputs relative humidity and temperature and a head a control mechanism that would broom over the hydroponic system by using a track and a motor. It is imperative that in the engineering world a project is delivered exactly as the client was expecting to maintain the level of professionalism that is required, as this ensures that the client will be willing to continue to do business with you.

# 6. Conclusion

The GreenMap provides a novel aspect to environmental control in greenhouse and other controlled environment agriculture. Currently, few stationary sensors are used to represent the large, dynamic environment in question and this setup could undoubtedly under represent regions within the system. This design provides a proof of concept, showing that with a click of a button environmental data from a large region can be retrieved, and displayed in a user-friendly visual manner.

This system holds potential to be scaled up both in size and complexity of data acquired. Realistically, to apply this at a commercial level there would not be a ‘one size fits all system,’ due to the unique nature of most greenhouses. Although, Our prototype system shows the potential of environment mapping concept, which can be applied to nearly any controlled growth system. The GreenMap is a tool to bring modern farmers even more comfortably into the 21st century.

# 7. References

Bogdan, Mihai. (2016). How to Use the DHT22 Sensor for Measuring Temperature and Humidity with the Arduino Board. ACTA Universitatis Cibiniensis. 68. 10.1515/aucts-2016-0005.

Mark Cardwell. (2015). A time of transformation in Quebec's greenhouse industry. Retrieved from: https://montrealgazette.com/business/a-time-of-transformation-in-quebecs-greenhouse-industry

Nasir, S. Z. (2014, October 22). Display Serial Data on Real Time Graph in Labview. Retrieved December 4, 2018, from [https://www.theengineeringprojects.com/2014/10/ display- serial-data-on-real-time-graph-in-labview.html](https://www.theengineeringprojects.com/2014/10/display-serial-data-on-real-time-graph-in-labview.html)

First Sensor. (2018). Wavelength-sensitive diodes (WS). Retrieved from [https:// www.first-sensor.com/ en/products/optical-sensors/ detectors/wavelength-sensitive-diodes-ws/](https://www.first-sensor.com/en/products/optical-sensors/detectors/wavelength-sensitive-diodes-ws/)

Dejan. (n.d.). DHT11 & DHT22 Sensor Temperature and Humidity Tutorial. Retrieved December 3, 2018, from https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/

Dejan. (n.d.). How To Control Stepper Motor with A4988 Driver and Arduino. Retrieved, December 3, 2018, from https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/?fbclid=IwAR1D1h4QKjSYGWBY-9N5zUxcC6QsrascAtYvgBE\_zfDxYPOxRQicMPNtQBY

Anusha. (2017, June 7). Humidity Sensor - Types and Working Principle. Retrieved from <https://www.electronicshub.org/humidity-sensor-types-working-principle/#Resistive_Humidity_Sensors_Electrical_Conductivity_Sensors>

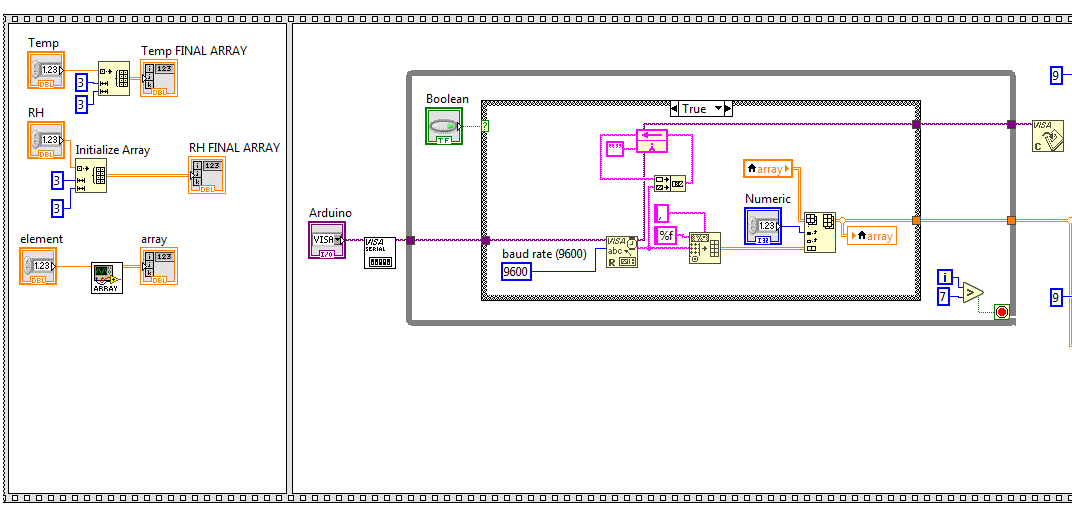
Fujimoto, H., & Maeda, T. (2000). Fluorinated Pitch. In Advanced Inorganic Fluorides (pp. 591–623). Elsevier. <https://doi.org/10.1016/b978-044472002-3/50019-3>

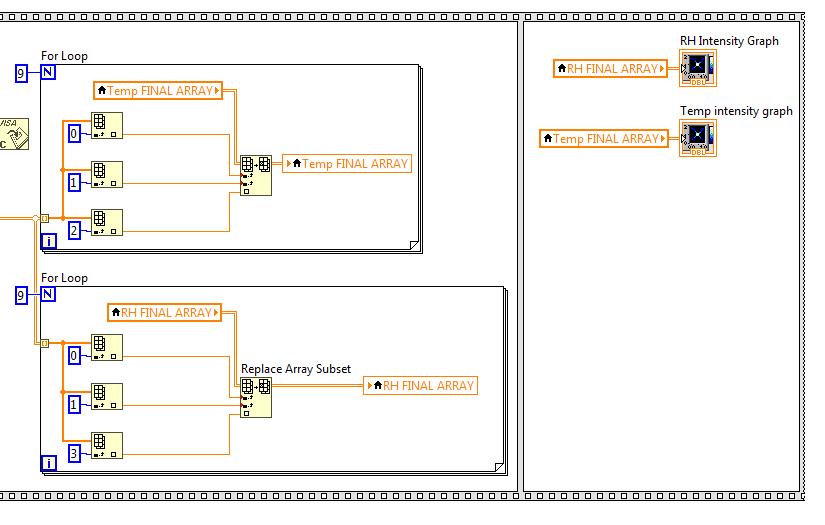
ResistorGuide. (n.d.). NTC thermistor » Resistor Guide. Retrieved from http://www.resistorguide.com/ntc-thermistor/

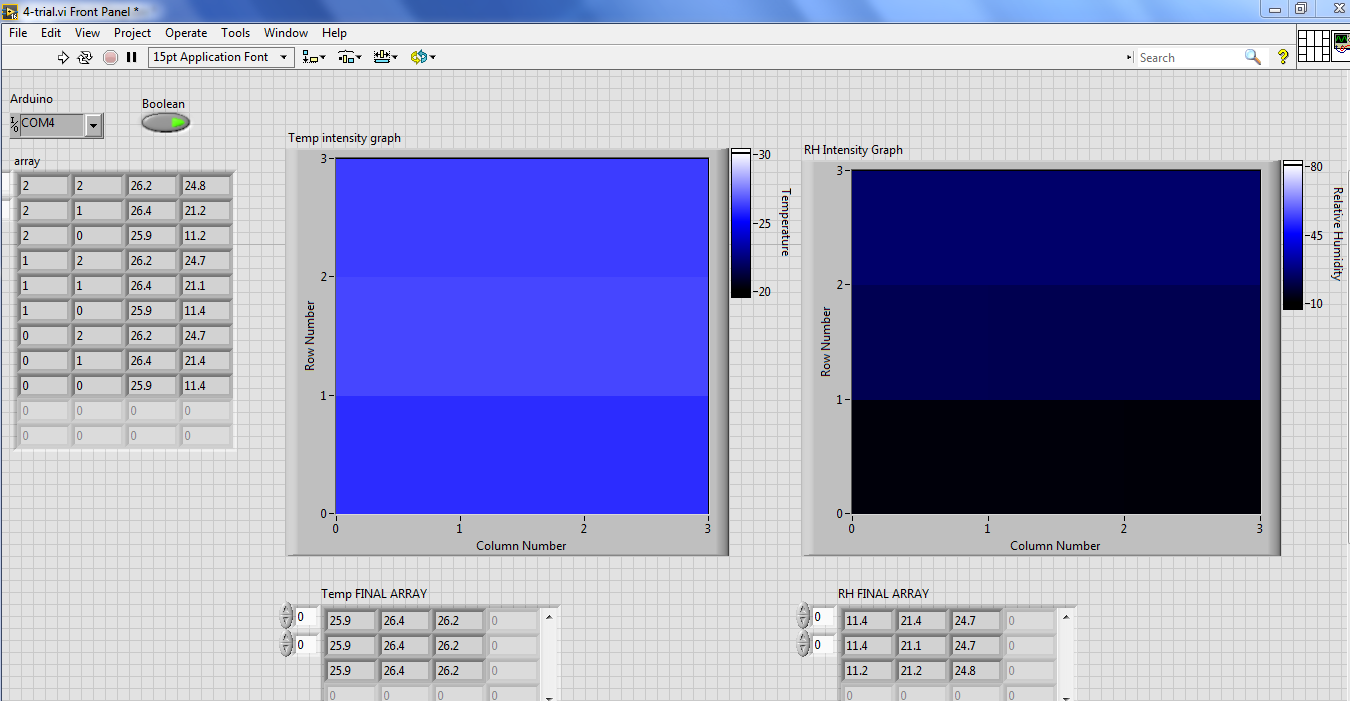
Dejan. (n.d.). How To Control Stepper Motor with A4988 Driver and Arduino. Retrieved, December 3, 2018, from https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/?fbclid=IwAR1D1h4QKjSYGWBY-9N5zUxcC6QsrascAtYvgBE\_zfDxYPOxRQicMPNtQBY

# 8. Appendices

## Final LabVIEW Front Panel & Block Diagram







# Arduino code

// Code for the TEMP and RH mapping with moving sensors

// Include Libraries

#include <SD.h>

#include <SPI.h>

#include <Wire.h>

#include "RTClib.h"

#include "DHT.h"

// RK previous version would #include <dht.h>

// Sampling Frequency

// how many milliseconds between grabbing data and logging it. 1000 ms is once a second

unsigned long LOG\_INTERVAL = 7000; //mills between entries (reduce to take more/faster data)

unsigned long LOG\_INTERVAL\_1 = 5000;

// how many milliseconds before writing the logged data permanently to disk

// set it to the LOG\_INTERVAL to write each time (safest and fastest)

// set it to 10\*LOG\_INTERVAL to write all data every 10 datareads, you could lose up to

// the last 10 reads if power is lost but it uses less power and is much faster!

unsigned long SYNC\_INTERVAL = 10\*LOG\_INTERVAL; // mills between calls to flush() - to write data to the card

uint32\_t syncTime = 0; // time of last sync()

//const int DECISION\_INTERVAL = 1000; // subject to change, Interval between checking conditions

#define PRINT\_LABEL 0 // Print labels in serial output

#define LOG\_DATA 1 //

#define POSITION\_CHECK 0 //

// Uncomment whatever type you're using!

//#define DHTTYPE DHT11 // DHT 11

#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

// define position values at start

int x\_position=0;

int y\_position=0;

// define pins

const int DHTPIN1 = 2;

const int DHTPIN2 = 3;

const int DHTPIN3 = 4;

const int START = 5;

const int RESET = 6;

const int greenLEDpin = 7;

const int startpin = 8;

const int chipSelect = 9;

const int stepPin = 10;

const int dirPin = 11;

// define the number of DHT to use

const int dhtcount = 3;

boolean TempState = true;

boolean HumdState = true;

// define x\_count for stepper movement 500 is 10 cm,

int x\_1=250;

int x\_2=500;

int x\_restart=1250;

// define state to operate

int state=0; // 0 not runing, 0 is runing

int TempInReading;

int HumdInReading;

// Define the sensors output, each sensor is unique

float TEMP0;

float TEMP1;

float TEMP2;

float RH0;

float RH1;

float RH2;

// Initialize DHT sensor.

// Note that older versions of this library took an optional third parameter to

// tweak the timings for faster processors. This parameter is no longer needed

// as the current DHT reading algorithm adjusts itself to work on faster procs.

DHT dht0(DHTPIN1, DHTTYPE);

DHT dht1(DHTPIN2, DHTTYPE);

DHT dht2(DHTPIN3, DHTTYPE);

void setup() {

// Define Pin type

Serial.begin(9600);

pinMode(DHTPIN1,INPUT);

pinMode(DHTPIN2,INPUT);

pinMode(DHTPIN3,INPUT);

pinMode(startpin, INPUT);

// use debugging LEDs if needed, could be readings indicator

pinMode(greenLEDpin, OUTPUT);

pinMode(stepPin,OUTPUT);

pinMode(dirPin,OUTPUT);

digitalWrite(stepPin,LOW);

digitalWrite(dirPin,LOW);

// connect to RTC, if needed

Wire.begin();

#if PRINT\_LABEL

Serial.println("temp\_0,temp\_1,temp\_2,rh\_0,rh\_1,rh\_2");

#endif

// DL made 3 DHT, 1 for each sensor

dht0.begin();

dht1.begin();

dht2.begin();

}

// dht1.begin(); this was included in void set up at one point...

//// STUFF TO ENTER POSITION

//char rx\_byte = 0;

//String rx\_str = "";

//boolean not\_number = false;

//int result;

void loop() {

// // delay for the amount of time we want between readings

digitalWrite(greenLEDpin, HIGH);

// Button for start signal

if (digitalRead(startpin)== HIGH){

state=1;

}

// Start the brooming

while (state == 1){

MOVE\_1();

x\_position=0;

// if ((millis() - syncTime) < SYNC\_INTERVAL){

// syncTime = millis();

delay(LOG\_INTERVAL\_1);

READ\_SENSORS();

LOG\_WRITE();

// }

MOVE\_2();

x\_position=1;

// if ((millis() - syncTime) < SYNC\_INTERVAL){

// syncTime = millis();

delay(LOG\_INTERVAL);

READ\_SENSORS();

LOG\_WRITE();

//}

MOVE\_2();

x\_position=2;

// if ((millis() - syncTime) < SYNC\_INTERVAL){

// syncTime = millis();

delay(LOG\_INTERVAL);

READ\_SENSORS();

LOG\_WRITE();

// }

RESTART ();

state=0;

break;

}

#if POSITION\_CHECK

Serial.print(x\_position);

Serial.print(", ");

Serial.print(y\_position);

#endif

// Now we write data to computer via serial ! Don't sync too often, let time for sensor response to be stable

}

// Read Sensors loop

void READ\_SENSORS() {

// Get DHT22 Info

// First Sensor

TEMP0 = dht0.readTemperature();

RH0 = dht0.readHumidity();

// Second Sensor

TEMP1 = dht1.readTemperature();

RH1 = dht1.readHumidity();

// Third Sensor

TEMP2 = dht2.readTemperature();

RH2 = dht2.readHumidity();

}

// Log Loop

void LOG\_WRITE() {

y\_position=0;

Serial.print(x\_position);

Serial.print(", ");

Serial.print(y\_position);

Serial.print(", ");

Serial.print(TEMP0);

Serial.print(", ");

Serial.print(RH0);

Serial.println();

y\_position++;

Serial.print(x\_position);

Serial.print(", ");

Serial.print(y\_position);

Serial.print(", ");

Serial.print(TEMP1);

Serial.print(", ");

Serial.print(RH1);

Serial.println();

y\_position++;

Serial.print(x\_position);

Serial.print(", ");

Serial.print(y\_position);

Serial.print(", ");

Serial.print(TEMP2);

Serial.print(", ");

Serial.print(RH2);

Serial.println();

}

void ENTER\_STATE (){

if (Serial.available() > 0) { // is a character available?

state = Serial.parseInt();

Serial.println(x\_2);

}

}

void MOVE\_1 (){

digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction

for(int x = 0; x < x\_1; x++) {

digitalWrite(stepPin,HIGH);

delayMicroseconds(500);

digitalWrite(stepPin,LOW);

delayMicroseconds(500);

}

delay(1000); // One second delay

}

void MOVE\_2 (){

digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction

for(int x = 0; x < x\_2; x++) {

digitalWrite(stepPin,HIGH);

delayMicroseconds(500);

digitalWrite(stepPin,LOW);

delayMicroseconds(500);

}

delay(1000); // One second delay

}

void RESTART (){

digitalWrite(dirPin,LOW); //Changes the rotations direction

// Makes 400 pulses for making two full cycle rotation

for(int x = 0; x < x\_restart; x++) {

digitalWrite(stepPin,HIGH);

delayMicroseconds(500);

digitalWrite(stepPin,LOW);

delayMicroseconds(500);

}

delay(1000);

}