BIOENG 13/2351 Hydruino Final Report

Group 15: Yu-Hsuan (Teddy) Chao, Lauren Grice, Jordan Weaver

1. Introduction

We developed a one-plant hydroponic chamber with an Arduino and Labview to control water level, light exposure, temperature, and relative humidity within the chamber. Hydroponic systems can conserve water and nutrients, prevent pests, increase crop yields, and speed up crop growth. This prototype system seeks to address a major drawback of hydroponics - the constant monitoring and adjusting of the system to maintain optimal growing conditions. A sketch of the system is shown in Figure 1.

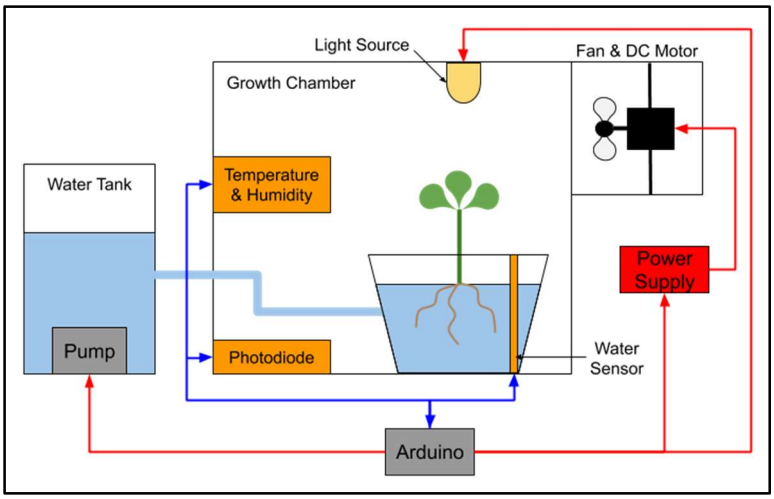

Figure 1. Diagram of the hydroponic chamber

The Labview code takes in Arduino sensor data, user set points, and the current time. It uses internal models of the system and returns control outputs to the Arduino. For example, the program only adds supplemental light during an intended light schedule, rather than whenever the photodiode registers dark. Figure 2 gives a conceptual flowchart of the program.

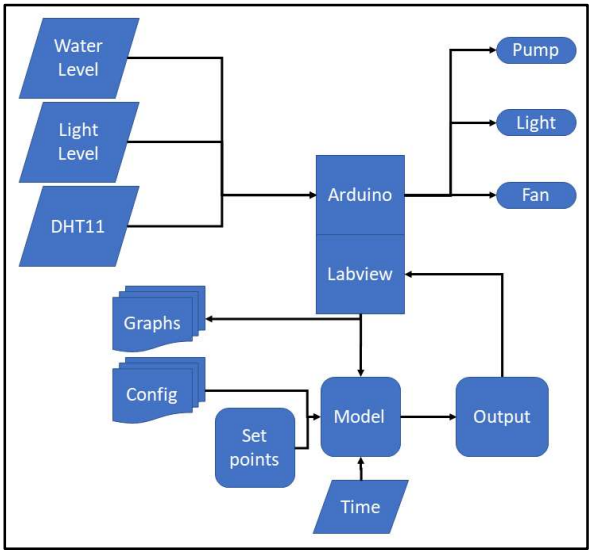

Figure 2. Flow chart representing input and output between subsystems.

Initial Goal	Achieved
Maintain Water Level	50% - needs Relay, Pump, reservoir, and piping
Maintain Temperature	100%
Maintain Humidity	100%
Maintain Light	75% - needs Relay and Grow Light
Operate Continuously without PC	0% - Would require laptop running Labview
Use DHT11 + Custom Firmware	100%
3D Print Something	100% - Motor Mounting Bracket

Table 1. Project Goals and Achievement Progress.

2. METHODS

2.1 Hardware

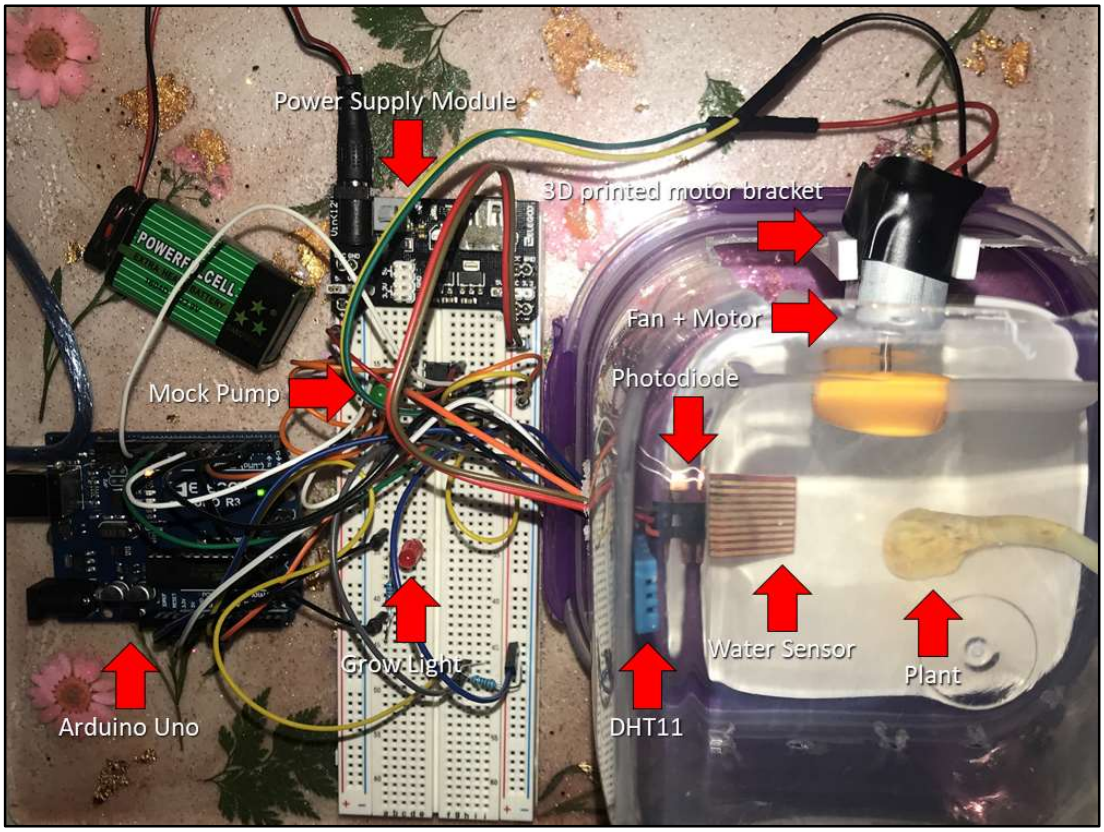

Figure 3. Final System Configuration.

2.1.1 DHT11 Temperature and Humidity Sensor

The DHT11 senses temperature (via a bimetallic thermistor) and humidity (via a capacitive humidity sensor). A bimetallic thermistor is a junction of two metals, which changes output voltage based on the temperature of the junction. A capacitive humidity sensor is a hygroscopic dielectric material between two electrodes, which changes capacitance based on ambient humidity. This sensor was placed inside the Growth Chamber. All air is exchanged with the fan to read the environment temperature and humidity once per hour and when the program is started. At all other times, the DHT11 reads the Growth Chamber's conditions. The DHT11 was wired according to Figure 4.

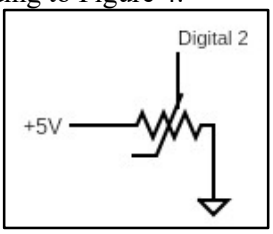

Figure 4. DHT11 wiring Diagram.

2.1.2 Photodiode

A photodiode senses light exposure via a photoactive layer, returning an analog output. The photodiode was placed inside the growth chamber, near the plant's leaves, and wired according to Figure 5.

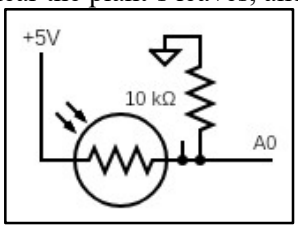

Figure 5. Photoresistor wiring diagram.

2.1.3 Water Level Sensor

This sensor measures the resistance between Power Traces and Sensor Traces to detect a water level, giving an analog output voltage. It was placed in the plant's water reservoir and wired according to Figure 6.

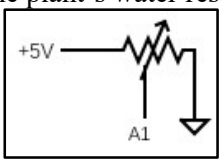

Figure 6. Water level sensor wiring diagram.

2.1.4 Fan, Motor, Controller, and 3D Printed Motor Bracket

Since light exposure and open water will elevate temperature and humidity respectively, a fan and DC motor were used as a ventilation system for the Growth Chamber. A motor bracket (Figure 3) was designed in Blender 2.82, sliced in Ultimaker Cura 4.8.0, and 3D printed with PLA on a modified Creality Ender 3 printer. To allow control of the motor from the Arduino, an L293D H-bridge integrated circuit was used, wired according to Figure 7.

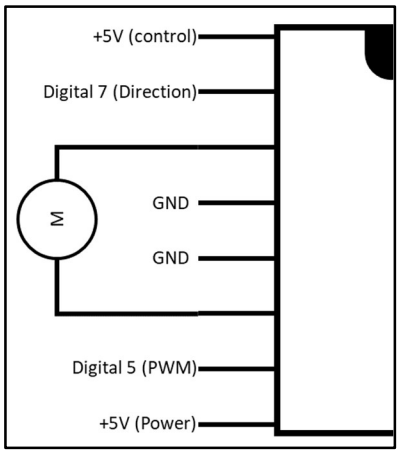

Figure 7. L293D H-Bridge Integrated Circuit Wiring Schematic.

2.1.5 Grow Light

A typical Grow Light has a power draw upwards of 400 Watts. For this project, a single LED was chosen as a stand-in for a relay, line-level power source, and light. As this mock component will not impact the plant's growth, it was wired directly on the prototype board. The LED was wired according to Figure 8.

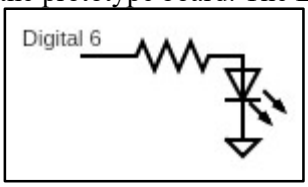

Figure 8. LED Wiring Diagram.

2.1.6 Arduino Uno

In full scale applications, an Arduino would either communicate a Labview installation remotely via WiFi or Bluetooth, or have an Arduino-only control scheme equivalent to the Labview code directly on the microcontroller. For the prototype system, communication and power over a serial port was sufficient. Additionally, the Arduino Uno was used to flash a Sanguino bootloader to the Ender 3 printer and upgrade it to Marlin 2.0.7 firmware. This prevented a case of thermal runaway due to a faulty thermistor.

2.1.7 Power Supply Module

Since the Fan and Motor have a current draw above the Arduino's output, a separate power supply was required. The Power Supply Module provided 5 V DC to four rails of the prototype board via 9V DC battery or barrel-jack DC converter (Figure 3). Many components drew power from these rails for wiring convenience.

2.1.8 Mock Pump

Common water pumps run on 12 V DC. As this is beyond the capacity of the Power Supply Module, a green LED was used as a stand in for the water pump. As this component has no current impact, it was wired directly on the prototype board. In the future, the Mock Pump will be replaced with a relay, dedicated power supply, and real water pump using the Water Level Sensor's output to maintain a constant volume of water in the reservoir. The Mock Pump was wired identically to the Fan and Motor on the second channel of the L293D H-bridge integrated circuit.

2.2 Labview Code

Figure 9 shows the breakdown of virtual instruments (VIs) included in the project.

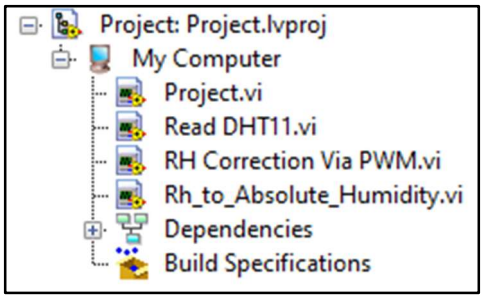

Figure 9. The structure of the project .lvproj file.

2.2.1 Read DHT11.vi

The LINX example for the DHT11 does not function. Instead, a custom firmware from Arafa Microsys [1] was used on the Arduino Uno in conjunction with a custom LINX command (Tutorial at [2]) in Labview to measure temperature and humidity. The Sub-VI calls LINX Custom Command #0 to run on the Arduino firmware, waiting up to 30 seconds for the command to return an output array. This long delay is to give the Arduino time to finish any pending operations and read the DHT11, which may take up to 15 seconds at times. The Arduino recognizes this command number, reads the DHT11 at a fixed port, and returns an array with a reading for both Temperature in Celsius and Relative Humidity. The array is split into the Relative Humidity and Temperature readings, cast to unsigned 8-bit integer types as required by the custom function, then returned from the Sub-VI. "Read DHT11" is shown in Figure 10.

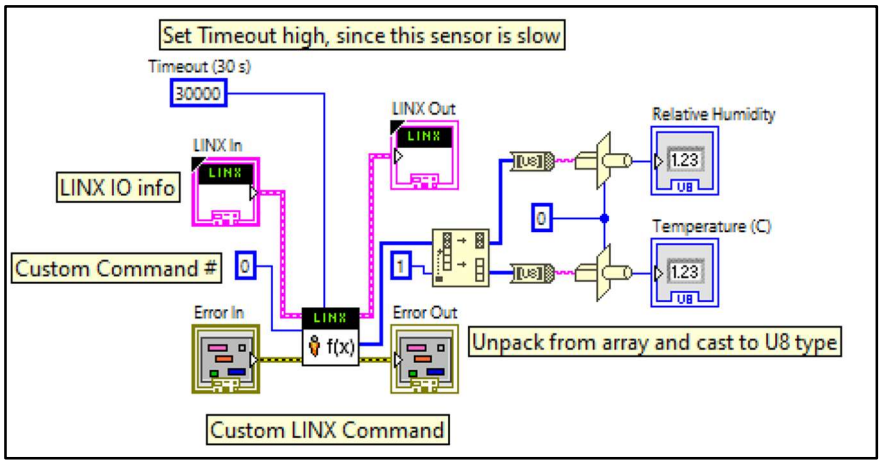

Figure 10. "Read DHT11" Sub-VI.

2.2.2 RH Correction Via PWM.vi

One of the main environmental controls is the Growth Chamber's Relative Humidity (RH), measured as a percent of the water vapor held in the air divided by the maximum amount that could be held at the given temperature. RH depends on the Absolute Humidity (AH) and the current temperature. Since both RH and temperature change when the fan is run, an iterative solving method was necessary. The system's and environment's RH and temperature and the maximum allowed RH are the inputs to this Sub-VI. The system's AH is calculated outside of the iterative solver, since this value will not change. Each iteration of the solver loop "tests" one additional second of ventilation through 6 distinct steps before outputting the number of seconds required for RH correction. These steps are;

1. Calculate the new Growth Chamber temperature after the *i*th second of ventilation
2. Convert the Growth Chamber's RH to AH based on 1.
3. Calculate the AH after the *i*th second of ventilation.
4. Convert back to Growth Chamber RH.

5. Compare 4 to the Maximum RH.
6. Stop the loop if predicted RH < Max RH, the loop has reached 30 iterations, or Stop is pressed

The Labview code for RH Correction Via PWM is given in Figure 11.

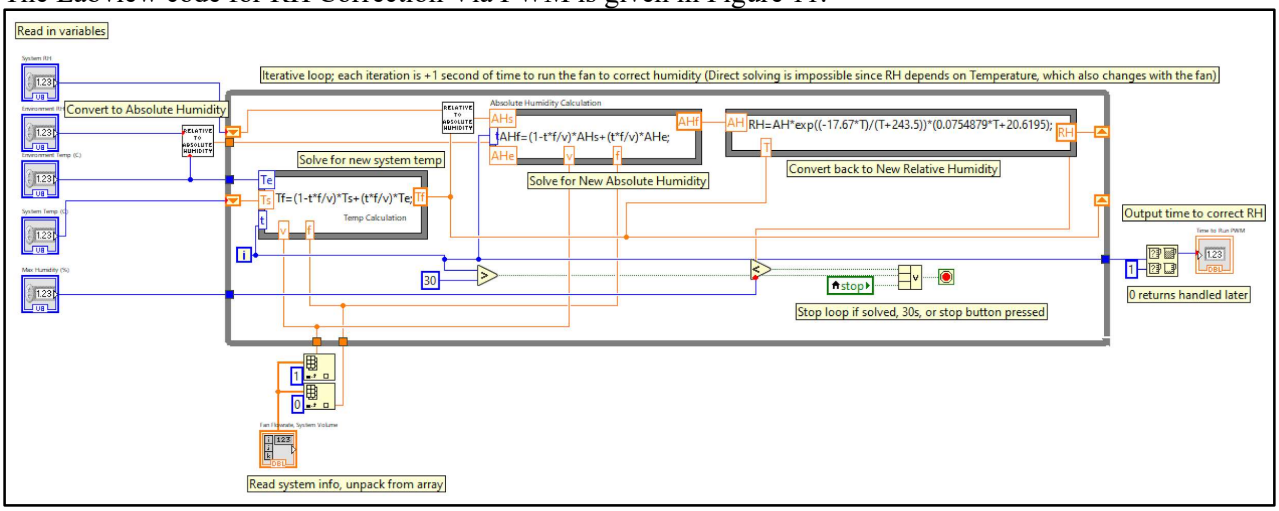

Figure 11. "RH Correction Via PWM" Sub-VI.

2.2.3 Rh_to_Absolute_Humidity.vi

This Sub-VI converts Relative Humidity to Absolute Humidity using the equation shown in Figure 12. It requires input of the Growth Chamber's current Temperature (C) and Relative humidity (%).

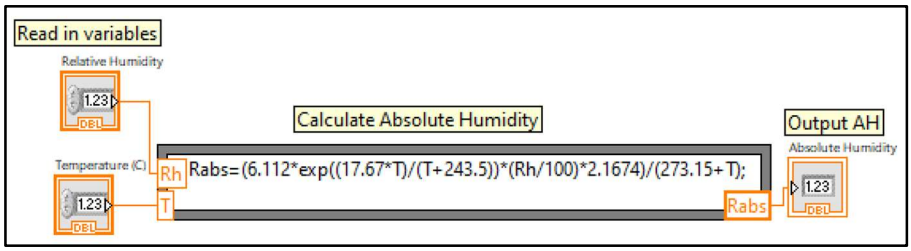

Figure 12. "RH to Absolute Humidity" Sub-VI.

2.2.4 Main VI: Project.vi

The control loop for this project is large; as such, only selected portions will be directly highlighted in the interest of report length. The full code is available at github.com/TheSilverhead/LabView. The User Interface during a run is shown in Figure 13.

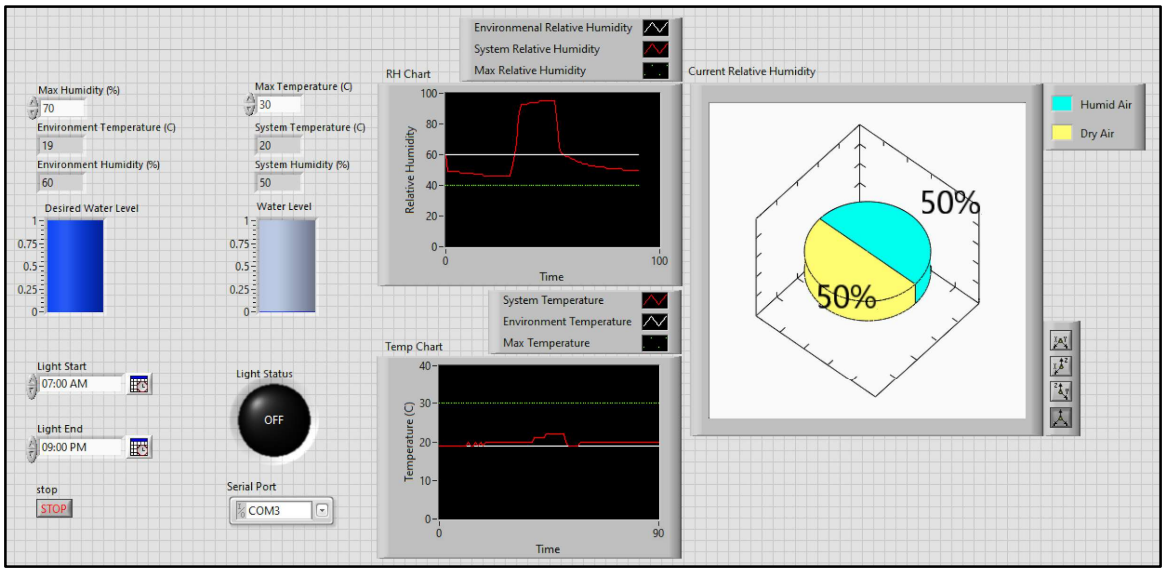

Figure 13. "Project" VI User Interface. Shows a relative humidity spike and correction via ventilation. The user supplies the desired Maximum Relative Humidity (to prevent mold growth), Max Temperature (based on plant tolerance), Desired Water Level (based on root length), Light Start/Light End time, and the COM port the Arduino is available on. Right: strip charts of Environment, System, and Maximum Temperature and Relative Humidity, and a visualization of the current Relative Humidity.

The default Max Humidity (70%) is intended as a level which would prevent mold or mildew growth within the chamber. The Max Temperature is an estimate of plant tolerances, and the Light Hours is roughly the period of sun exposure for May, a typical time to plant. The Water Level defaults to 100%, or a full pot, since young plants will have shorter roots.

The prototype may vary in Growth Chamber size, fan flow rate, and other physical variables when constructed at different scales or by different users. Instead of hard-coding these variables, a Configuration file is programmatically created, written with default values if created, then read into the program as variables. This is done before the COM port is opened, so that a user can generate the file without the need for an attached Arduino.

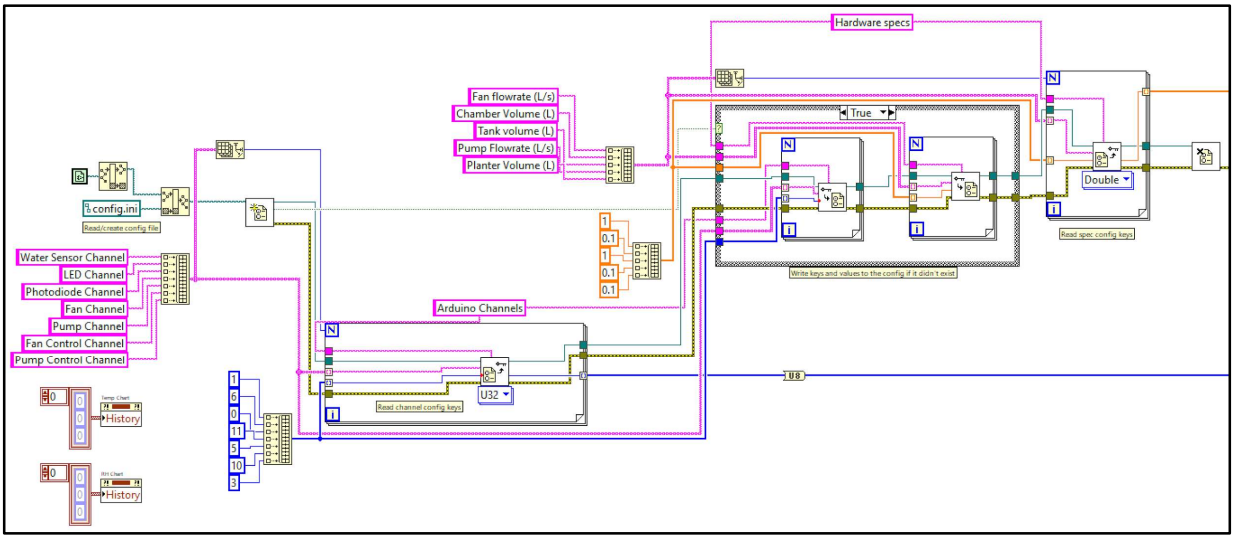

Figure 14. Configuration File Reading, Creation, Writing, and Chart Clearing.

The Grow Light is turned on only when it is Within Light Hours (Between Light On and Light Off), and when the photodiode reads less than 50% of its maximum. Figure 14 gives the logic of checking for "Within Light Hours" and whether it is currently on the hour (XX:00) or Iteration 0, which signals a full air exchange and DHT11 reading from the environment.

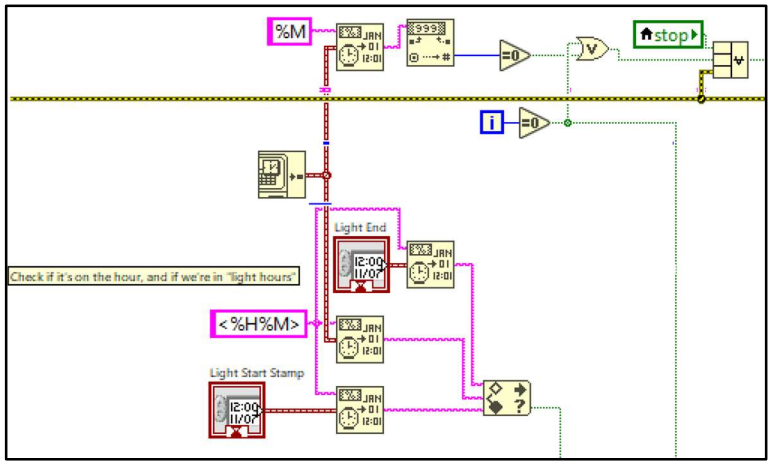

Figure 15. Checking for iteration 0, on the hour, and within Light Hours. Extra wires removed for clarity.

When the logic from Figure 15 returns True, Figure 16 is run. This sets the motor channel to the correct direction and the PWM duty cycle to 100%. The For Loop runs for *N* iterations, each lasting 1 second, as a simple timer, where *N* is 1.5*(Volume of Growth Chamber / Volumetric Flow Rate of Fan), rounded. The duty cycle is returned to 0%. The DHT11 is then read, and the User Interface updated. The same Set, Wait, and Return logic is followed whenever the Fan or Pump are run.

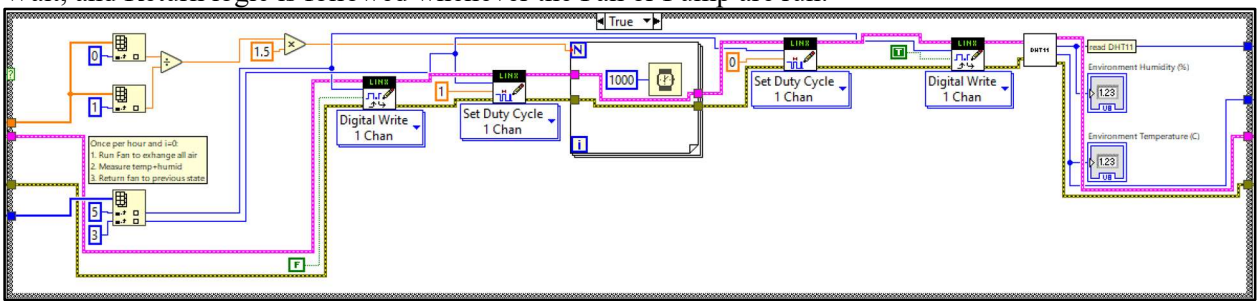

Figure 16. Hourly and Iteration 0 air exchange and DHT11 reading for environment Temperature (C) and Relative Humidity (%).

The control loop only exits when the user presses a Stop button, as this is meant to be a continuous use program. Figure 17 shows the logic of safe shutdown.

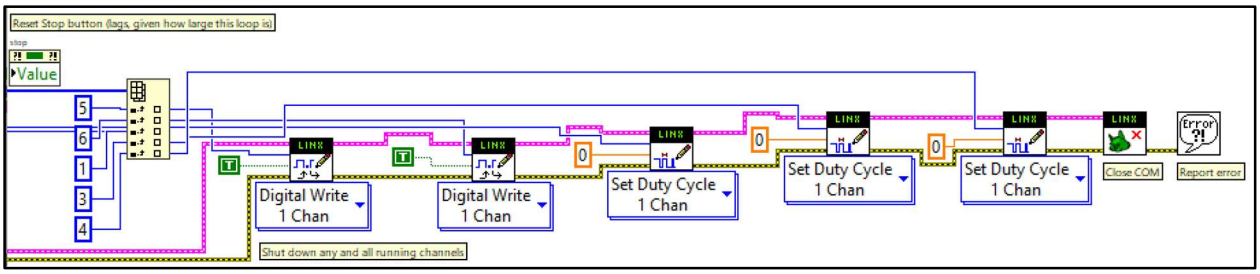

Figure 17. Shutdown sequence, putting all channels in a safe state before closing the COM port.

3. Functional Testing

Lacking water in the system and the desire to move a computer to a window for long periods of time to grow a sprouted garlic clove limited Jordan's functional testing to the Grow Light and Fan. Both tests can be seen in Figure 13. When the system was turned on in a dimly lit room, during Light Hours, the Grow Light and its indicator on the User Interface turned on. A flashlight shone on the photodiode turned off the Grow Light after several seconds.

The Environment Temperature/Humidity and RH/Temp Corrections were tested by running the code and allowing the Growth Chamber to stabilize. Hot, humid air was introduced into the Growth Chamber until the DHT11's readings triggered the Fan to run. The system corrected more slowly than expected; this was due to moisture condensing on the interior surfaces of the Growth Chamber and a lower than expected Fan flow rate. As a result of this test, a 3D printed fan replacement has been added to Future Work.

4. Future Work

This prototype system has several obvious areas for future work. In no particular order,

- Use a relay and line-level voltage to control a real Grow Light (400 W+)
- Add a relay, power supply, water pump, and external water reservoir
- Use a better Fan (3D printed replacement possible)
- Calibrate photodiode based on light exposure from actual sunlight
- Add pH, Total Dissolved Solids (TDS) meters and correction methods
- Translate the code to Arduino-only, or implement remote Labview connection
- Permanently wire and enclose the system

5. References

[1] Arafa Microsys
https://www.youtube.com/watch?v=DGTbRQR-x_4
[2] LINX Custom Command Tutorial
https://www.labviewmakerhub.com/doku.php?id=learn:tutorials:libraries:linx:misc:adding_custom_command