THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF CHEMICAL ENGINEERING

LOW-COST LABORATORY EQUIPMENT MONITORING AND AUTOMATION

MICHAEL JOSEPH COVER
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Chemical Engineering
with honors in Chemical Engineering

Reviewed and approved* by the following:

Wayne Curtis
Professor of Chemical Engineering
Thesis Supervisor

Ali Borhan
Professor of Chemical Engineering
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Lab automation is a common technique utilized in both industrial and research capacities in order to optimize the productivity of a process and minimize labor costs. While major companies tend to outsource automation, research labs have the potential to introduce more specific and less expensive soft circuit technology connected to the internet in order to provide an immediate status update on major equipment throughout the lab. The purpose of this project is to tether sensors and Arduino technology to vital processes in Dr. Wayne Curtis's lab, which were fiscally enabled by the grants provided by the Defense Advanced Research Projects Agency (DARPA) and National Science Foundation (NSF) Basic Research to Enable Agricultural Development (BREAD). After comparing the practical use of several microprocessors, the Arduino ethernet shield (AES) was chosen as the optimal platform for the transmission of sensor data to the internet. Several sensors were selected and extensively tested: moisture sensors for reinforcing watering procedures, thermocouples for monitoring freezer temperatures, and light sensors for ensuring the fidelity of overhead lighting above plant tissue and algae cultures. Though the AES was able to transmit data within the local network, inevitably this thesis was unsuccessful in locating a method to transmit sensor data outside of the local network. Through the implementation of the AES, after the times of social distancing to avoid spreading COVID-19, future equipment operational fidelity within CurtisLab will be enhanced through the monitoring offered by the AES-generated webservers. Ironically, the COVID-19 restrictions on research activity emphasized the utility of this thesis work. If it had been in place, it would have provided invaluable monitoring during research shut-down.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my honors advisor, Dr. Ali Borhan, for his continued support and guidance during my time here – through difficult classes and navigating different research labs. I also want to thank Dr. Wayne Curtis, my thesis advisor, for all the knowledge and surplus of fun he has bestowed upon me throughout my time in his research lab. I am especially grateful for Natalie Thompson, my graduate student mentor. Without the guidance of Dr. Curtis and Natalie, this thesis would never have been written. The aid received from the entirety of Dr. Curtis's lab group has been vital to my progress in his lab, especially the foundational efforts input by Dr. Curtis's sons and former lab members, Matthew and Brandon Curtis. Many thanks to all of you. Finally, I would like to acknowledge the unconditional support that my friends (shipmates and otherwise) and family provided me with to get here.

Chapter 1

Background and Introduction

Lab automation has emerged over the past few decades as an innovative solution for optimizing both the quality and quantity of project results – from that of large-scale industrial processes through smaller research laboratories. While the implementation varies based on the scope of the project under consideration, the core methodologies remain constant. Automation relies heavily on the core principles of industrial engineering, so regardless of what kind of process or operation is being conducted, the approach of this idea can benefit any kind of project. For this reason, novel concepts in automation – applied to a specific process – can augment automation across disciplines and in any kind of lab size or structure. In addition, the rapidly increasing availability of cheap microcontrollers and processors has dramatically facilitated the incorporation of automation practices into the everyday practices of a laboratory. Therefore, a brief synopsis of both industrial and at-home automation will be expanded upon in order to develop a perspective for the integration of lab automation in a research setting.

In an industrial capacity, lab automation has the potential to save companies fortunes through the refinement of processes and the monitoring and management of key equipment. Large enough companies have the funds and the resources to fully define and expand upon many related aspects of lab automation, whereas a research-based laboratory most likely has neither the broad expertise nor the financial motivation to implement automation. Automation will only be effective for research laboratories if it can integrate easily and be highly adaptable to changing research goals. However, as noted previously, automation can be applied even loosely to any

process – as long as there is an existing operational structure to refine. Within the industrial scope, lab automation generally consists of four major domains: (1) application of sensors and related equipment for monitoring equipment, (2) managing the existing research or process, (3) autonomous testing of equipment in order to further refine the setup and automation process, and (4) thorough statistical analyses of the data collected by the automation equipment (Rosso et al., 2019). This project focuses on the first of the four aforementioned key components of laboratory automation.

The application of sensors to monitor and manage equipment is vital to ensuring the fidelity of processes. However, a sensor in and of itself accomplishes nothing if it lacks the means to communicate key operational variables with relevant personnel. Sensors combined with a programmable logic controller (PLC), also known as a microcontroller, highlight the ever-expanding role of the Internet of Things (IoT) in the everyday life of common technology (Thramboulidis, 2015). With the right microcontroller along with the appropriate sensors and their respective motor components, the paradigm of the industrial world can be reflected in laboratory automation techniques.

Continually, the effects of progressive automation on a commercial process can be enormously beneficial, including those of specific interest to Dr. Curtis's lab for plant propagation. Specifically, in a single *pilot* plant factory – constructed with automated management of the Norway conifer embryo development – has a capacity of over one million propagated plants per year. The overall capacity in Sweden to produce tree seedlings remains within the millions. Thus, automation on an industrial scale has the potential to assist in the mass production of a highly cultured product (Egertsdotter et al., 2019). It is important to note here, though, that the technology used in cultivating conifer embryos was developed by a third-party

source. This then explains one of the primary differences between automation for industry and automation for research. Automation applied to commercial purposes generally rely on outsourcing the technology and the labor for the implementation. The expensive nature of this endeavor does not undermine the lucrative benefits for industry on a large scale, and thus it is often easier for major corporations to rely on the skills and innovations of other companies rather than invest heavily in researching an area where the company lacks expertise.

Overall, though, this process is still quite similar to the application of lab automation practices in a research lab setting. Neither the operational content nor the parties involved affect the fundamental utility of the automation goals. The application of this automation backbone is indifferent to the nature of the process; this support system can generally be applied to any process with optimizable equipment. Within a research lab, generally the automated circuitry and software are tailored more closely to the needs of the lab, and entail a much higher degree of flexibility. Outsourcing automation development is not only beyond financial consideration for a smaller research lab, but it constrains the intellectual development of lab members. It therefore is likely in the future that research-based labs will increasingly rely on low-cost, user-friendly, and highly adaptable PLCs in order to increase the quality of their research results. Along with these microcontrollers, research labs utilize readily available sensors for ensuring equipment fidelity surrounding experiments within the lab. The microcontroller paired with these sensors provides the backbone for both monitoring and managing – if the sensors contain components which can be actuated remotely – the process machinery. In addition to sensors, the application of simple switches and other actuators allows for lab users to remotely manipulate lab equipment and enhance experiment execution to obtain higher-quality results.

Having explained the general practice for automation in a research lab, the importance of applying lab automation to the highly productive and vast research within CurtisLab can be explained. More sophisticated automation strategies could be applied for the regulation of time-lapse photography, bioreactor monitoring and control, and automated plant growth studies. However, the primary need for equipment fidelity requires simply the monitoring and verification for the day-to-day operation of the photosynthetic lighting system, freezers, and soil-watering techniques. This has been particularly true for the circumstances of Dr. Curtis's lab during my tenure as an undergraduate student in the lab. The laboratory has been moved twice in the past several years, with the most recent move in Fall 2019 resulting in dealing with the typical *new-building* issues where there were multiple small floods, several power failures, and unfamiliarity with lighting timers that resulted in multiple, significant laboratory problems each week. The loss of a single -80 °F freezer (whose backup alarm battery also failed) resulted in over $10,000 of lost molecular biology enzymes, and countless loss of quality in frozen samples that will not likely ever be recovered. Therefore, in an attempt to prevent future equipment malfunctions from severely impeding research and progress from other lab members, this project focuses on the development of regulated technology that can ensure the fidelity of various lights, freezers, and soil moisture content of plants throughout the lab.

Chapter 2

Selection of Equipment

Before proceeding, it must be stated that the foundation of this project rests on the thorough lab automation research that former lab member Brandon Curtis provided, which included purchasing much of the microcontroller and microprocessor equipment (including newer and more innovative technology). This equipment was relied upon in order to fully realize the scope of this project. Additionally, the basis for the implementation of lab automation in this thesis relies on the work of Matthew Curtis – the development and automation of temporary immersion bioreactors – who performed this work in Dr. Curtis's lab before my own tenure within the laboratory (Curtis, 2013).

Another motivator for developing an alternative approach to lab automation was the strain placed on the lab during several mass movements: from Fenske building to temporary space in Greenberg Building when Fenske was torn down, and from Greenberg back to the Chemical and Biomedical Engineering Building after the new building was constructed. These movements dramatically altered several crucial variables involved in lab automation, most notably internet availability and restrictions imposed by enhanced internet security. As an example, Matthew Curtis's implemented automation – with the assistance of the software LabVIEW – involved daisy chaining Zigbee controllers in order to bypass the internet firewall and transmit time-lapse videos to lab members outside of the local network (Curtis, 2013).

Though this experiment began simply based on my own love for coding and solving problems with computer logic, it quickly evolved into an experiment in lab automation with a considerable impact on the lab members and overall lab around the applied sensors. However, an idea has no basis without the specific sensors and the computing power needed to transmit sensor

output to the user. Touring the lab and understanding the operational parameters – due largely to the guidance of the knowledgeable and capable lab members – was essential to gaining an understanding of what would benefit the most from applying sensors to monitor equipment output. These critical and wonderful persons are listed after my own name in **Table 1**. Natalie Thompson was my graduate student mentor, and Dr. Wayne Curtis was the primary investigator supervising this project.

Table 1. Contact list for experiment

First Name	*Last Name*	*Email*
Mr. Michael	Cover	Mkc5535@psu.edu
Ms. Natalie	Thompson	Nst31@psu.edu
Dr. Wayne	Curtis	Wrc2@psu.edu

Through this process, the most ideal microprocessor and appropriate sensors were selected in order to implement a support monitoring structure throughout key lab components. Once selected, thorough testing was performed on the selected moisture, temperature, and light sensors in order to fully understand their capabilities. The ideal operational limits were derived from the sensors, and these boundaries were applied logically via the microprocessor code in order to safeguard the selected equipment. To begin, a microcontroller/microprocessor pair was selected.

Internet-Capable Device

A method for connecting the sensors to the internet was essential for data transmission to be carried out from the sensors to the lab members outside of the lab. The importance of discussing this matter cannot be understated, as all of these IoT devices discussed in the following paragraphs have considerable potential for implementation in lab automation. Their services and parameters vary considerably, and thus have a broad area for applications in a research-based setting. Six devices were considered for implementation for this project: PARTICLE Electron (around $70.00 for an Electron kit), PARTICLE Photon ($20.00 per device), Raspberry Pi ($70.00-$100.00 for a kit), Arduino Wi-Fi Shield ($11.99), Arduino Bluetooth Shield (around $13.00), and the Arduino Ethernet Shield (AES) for $11.99 per device. All of the aforementioned Arduino shields require a base Arduino microcontroller, which can be purchased for $15.00 to $20.00.

The PARTICLE Electron – a cellular-based transmitting device which can transmit sensor data to users through their cell phone – was one of the most viable options, but it was inevitably removed from consideration because of the limitations inherent to its transmission. The cellular signal within the laboratory itself was not especially reliable, and the minor monthly cost was still more than the free transmission offered by the other platforms ($2.99 per Electron per month). In addition, most importantly, all of the other options (except the Bluetooth shield) had the potential to communicate to users through an Application Program Interface (API), which in its essence was by far the most preferable transmission method. An API in this context (described further in Chapter 4) would allow the internet signal received by the microcontroller/microprocessor to output many different kinds of messages (emails, smart-phone application push notifications, tweets, et cetera) to any person using these previously mentioned

services. For any lab with excellent cellular signal and no need for API implementation, the PARTICLE Electron would be an excellent choice.

Further, the PARTICLE Photon was also heavily considered for its application in this project. PARTICLE is considered innovative for its user-friendly and sleek method of integration. Not only does the PARTICLE Photon package include an API within the code – all of which can be downloaded through a computer's command prompt – but this device is capable of connecting to most internet networks with ease. Because PARTICLE tech is considerably newer than the Arduino Uno boards/shields, the related processing chip power attached to the PLC is also considerably greater. Although this sleek model is theoretically sound, the additional required keys of WPA2 enterprise networks became an insurmountable boundary (please refer to the Appendix for more details on the troubleshooting process for connecting the PARTICLE Photon). However, as this company and technology continues to develop, the application of this device in a WPA2 enterprise network has enormous potential. If a local server could be developed within a laboratory going forward, this method would have been the obvious choice.

All of the other device interfaces rely on Arduino microcontrollers in order to process data input from sensors. Many of these were available within CurtisLab before the start of this project thanks to Brandon Curtis. The Raspberry Pi was also considered. This was the only option considered which contributed a high-powered microprocessor, rather than only a PLC. A microprocessor is essentially a small computer, with the ability to compute significantly faster than a microcontroller. Whereas a microcontroller is severely limited in its computing power, and can only run one program at a time, a Raspberry Pi microprocessor has a much greater capacity to run program code and can run several programs at once. In an application where these discrepancies constitute the difference between meeting standards and not, the Raspberry

Pi would be the obvious choice in a setting not relying on a WPA2 Enterprise network. However, similar to the implementation of the PARTICLE Photon, the WPA2 Enterprise connectivity encryption sophistication prevented further development of this method.

Finally, the Arduino shield products were considered for this project. The Arduino Wi-Fi Shield offered the ability to wirelessly connect sensors to the internet, but this model unfortunately was retired several years before the start of this project. In addition, it is likely that this wireless transmission method (similar to both the Raspberry Pi and the PARTICLE Photon) would have run into issues with connecting to a WPA2 Enterprise network. Further, research was conducted to determine the contextual significance of applying the Arduino Bluetooth Shield. Inevitably, the need to transmit equipment operational data to lab users outside of the small radius inherent to Bluetooth devices removed this device from consideration.

The AES was studied thoroughly, and chosen as the Internet-capable device for several reasons. Firstly, the relative limitations of the Arduino Uno processing power were not deemed significant here – it could effectively execute the hybrid code combining webserver functions and the thermocouple, light sensor, and moisture sensor programs. Secondly, and most importantly, the Ethernet Shield could overcome the WPA2 Enterprise encryption limitations which severely impeded the other platforms. Although this method is not wireless, it only requires a unique Media Access Control (MAC) and Internet Protocol (IP) address for the AES to function as desired. Thirdly, as was considered earlier, the Arduino Uno (coupled with the internet capability of the Ethernet shield) had the potential to connect to a third-party API in order to transmit sensor data to lab members outside of the local network. The preferred API method was abandoned due to their API servers being shut down, but this will be discussed in

full in Chapter 5. Having chosen the PLC and the internet-capable unit, the experiments were then conducted.

Soil Moisture Sensor

The need to monitor the moisture level in plants demanded the selection of a soil moisture sensor. To understand raw data from the soil moisture sensors, the science behind the equipment was investigated. These sensors based on the following equation:

$$k = \frac{C_m}{C_a} \qquad (2.1),$$

where C_m is the measured capacitance of the soil/medium which the sensor resides, C_a is the capacitance without the medium/of the air, and k is the dielectric constant. This is how moisture is quantified, by calculating the dielectric constant, or the dielectric permittivity, of the plant soil.

The plants in CurtisLab vary considerably, but consist of cabbage, Dioscorea, transgenic tomato plants and Arabidopsis plants. They are located in plastic pots (including sewer pipes for groups of cabbage plants), which allowed for easy access of any of the soil-moisture sensors in consideration. However, a limiting factor was the mode of data transmission that the sensors relied on. Many of the soil moisture sensors were developed by companies that could communicate to customers' cell phones through integrated wireless chips. This increased the ease of implementation, but certainly did not overlap with the mission of developing relatively cheap automation throughout the lab with the same platform. The whole purpose of this project was to develop the means to transmit data to lab members through basic, soft-circuit equipment.

To reduce cost, simple dual-prong sensors – HiLetgo LM393 Soil Moisture Detect Sensors – were chosen that could easily attach to a microprocessor. These sensors are available in packs of five from HiLetgo for $7.50. Once this dual-prong moisture sensor was chosen, the proper wiring was investigated and implemented (as shown in **Figure 1**).

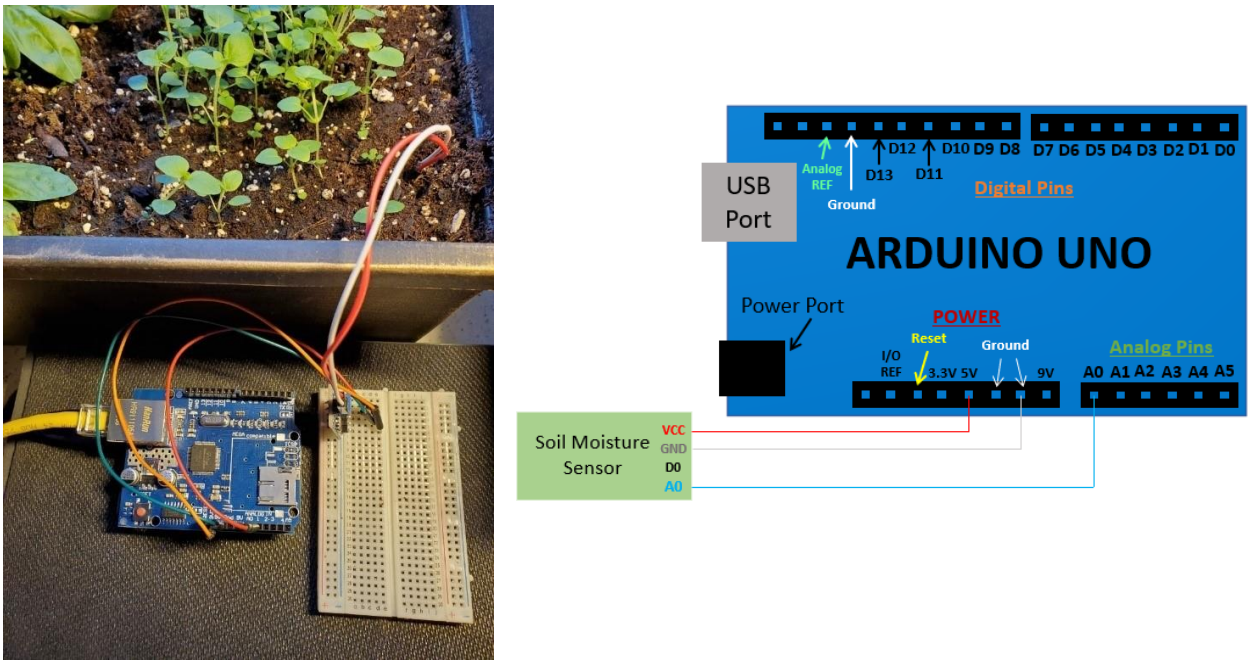

Figure 1. Picture and wiring diagram of the soil moisture sensor.

To view data obtained from the sensor and the Arduino microcontroller, code was developed in the Arduino IDE software (Appendix B). Developing this code was essential to learning the basics of analog pins within the world of C/C++ language Arduino code. After the data was inputted, the serial monitor – the basic display monitor of Arduino –outputted the data generated within the soil moisture code (**Figure 2**).

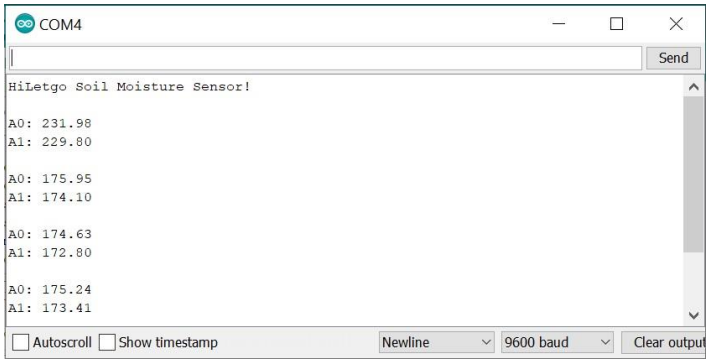

Figure 2: Arduino serial monitor display of soil moisture sensor data.

As can be seen in **Figure 2**, these analog values are not specific to any metric; they are dimensionless. Therefore, in cases where the units acquired from the sensors need to be exact, there is a need for 'conversion' code that can extract values corresponding to a given unit from the raw, generic analog values (range from 0 to 1023, or 2^{10} different output values). This range of analog values – typical of a 1024-bit microcontroller – corresponds to the voltage outputted by the sensor. A returned analog value of 0 indicates that the sensor returns 0 V, and an analog value of 1023 means that the sensor returned 5V, the maximum possible returned voltage. This range of voltage outputs (0-5V), combined with the resolution of the given sensor, determines the sensitivity of the range in a given application. Though these outputs are unitless, the consistent output here was sufficient for future measurement and correlation to soil moisture content. Regardless of the units supplied, lab members would have to determine what ranges of any scale correspond to desired watering levels. Thus, because raw analog output is acceptable here, the basic methodology for reading information from sensors and displaying them in a readily accessible manner was determined. It is important to note that soil permittivity will remain nearly constant for nearly all applications, whereas the appropriate ranges determined for both temperature and light will vary a considerable amount based on the needs of a specific research lab.

Temperature Sensor

For temperature, there are many available models that can communicate with a microcontroller such as the Arduino Uno. First, an understanding of the science was developed before the sensor was chosen. The thermoelectric effect explains how a change in voltage can be converted to a temperature change, and how a measured temperature change can be converted to a voltage. All thermocouples rely on the thermoelectric effect to function, but because different types of thermocouples contain different metals, the math involved likewise varies. Several types of commonly available thermocouples were considered here: J, K, and T-type. Of these, only K-type thermocouples had inexpensive and user-friendly models available (the MAX6675 model, sold by HiLetgo) for which the wiring was extremely straight forward. Continually, thermistors – which output temperature-dependent resistances – were also considered here. Thermistors are not commonly soldered together with resistors like the K-Type thermocouple; using them would either require a breadboard (mobility lessened greatly) or lab members to solder them together. Further, using thermistors with Arduino tech requires converting inputted resistances to temperatures. Thus, to simplify the implementation, thermistors were ruled out.

The MAX6675 K-Type thermocouple – available in a pack of two for $12.99 – was chosen for its ergonomic wiring setup and inexpensive nature. In order to work effectively, a K-Type thermocouple depends on a circuit containing two different alloys: chromel and alumel. The voltage difference in a thermocouple is generated at the ‘hot’ point/measuring point where the dissimilar metals are welded together. The electron densities of the different metals vary at different temperatures. The voltage difference is then measured at the ‘cold’ junction where the wires end and do not touch. The differential temperature (a thermocouple does not measure absolute temperature) between the ‘hot’/measuring point and the ‘cold’/connection point is

determined from the voltage difference between the two dissimilar metals. A cold-junction compensator ensures that cold junction temperature (generally at ambient temperature) does not affect the measured result. Now, a K-Type thermocouple is characterized based on the following equation:

$$T_{ref} = \frac{1}{A+W(BC+W^2)} - 273.15 \qquad (2.2),$$

where T_{ref} is the converted temperature given in Celsius. W is the natural log of the resistance measured in the circuit of the two dissimilar metals, and A (1.28E-3), B (0.24E-3) and C (9.27E-8) are constants.

Fortunately, the MAX6675 thermocouple has an Arduino library that is largely "plug and play" with circuitry that converts generic analog input into temperature values – for both Fahrenheit and Celsius. This type of sensor is increasingly available as dedicated chip technology provides inexpensive signal conditioning that replaces entire circuit boards of the interfacing circuitry of the past. As these units are commonly understood by all lab members, this output is vastly preferred over general analog output. After connecting the necessary wiring to operate the MAX6675 thermocouple (wiring shown in **Figure 3**), these temperature sensors were then placed adjacent to a freezer in Dr. Curtis's lab to enhance its operational fidelity.

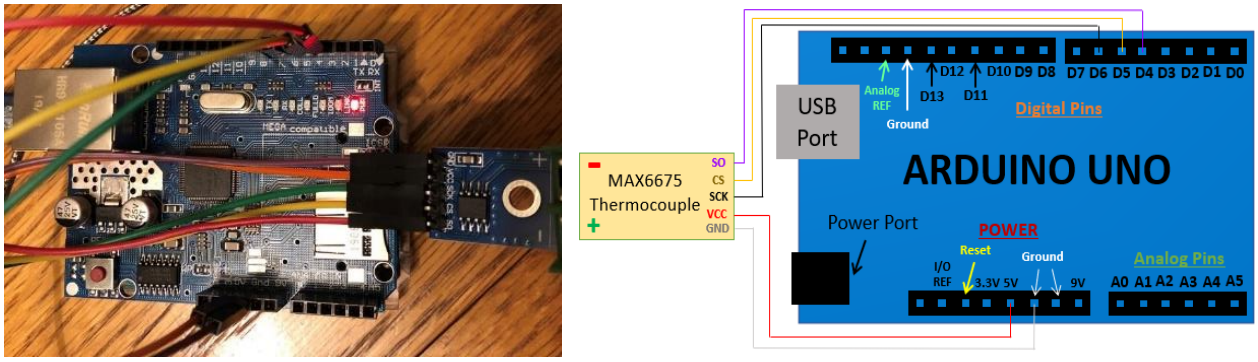

Figure 3. Picture and wiring diagram of the MAX6675 k-type thermocouple circuit.

The MAX6675 thermocouple program (code shown in Appendix B) was run and temperatures in Celsius and Fahrenheit were outputted. This can be seen in **Figure 4**.

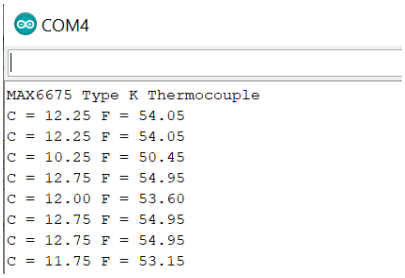

Figure 4. Serial monitor display of MAX6675 thermocouple output.

Light Sensor

A light sensor was chosen due to several important considerations. Although other light sensors have the potential to measure the intensity of different bands of wavelengths of light – made possible by technologies such as those utilizing an inverse LED sensor (Tsai et al., 2019) – this was deemed unnecessary for simply determining if an overhead light was on or off. The BH1750 light sensor was chosen for its ability to accurately measure the overall light intensity of its surroundings. Light sensors often rely on photoresistors, or light-dependent resistors, but the BH1750 light sensor relies on a more sensitive component (photodiode) to function. Based on the following equation (**Equation 2.3**),

$$R_\lambda = \frac{I_P}{P} \qquad (2.3),$$

photodiode responsivity (R_λ) can be understood. Within this equation (**Equation 2.3**), I_P is the photocurrent generated by a light source, and P is the incident light power, where P is equal to the photon flux multiplied by the energy of a photon. Overall, the BH1750 returns a value with units of lux (a measurement of luminous emittance that relates to human perception of

light), which is equal to a lumen (base unit for luminous intensity) over an area. The BH1750, once soldered appropriately, was connected appropriately in a circuit (**Figure 5**).

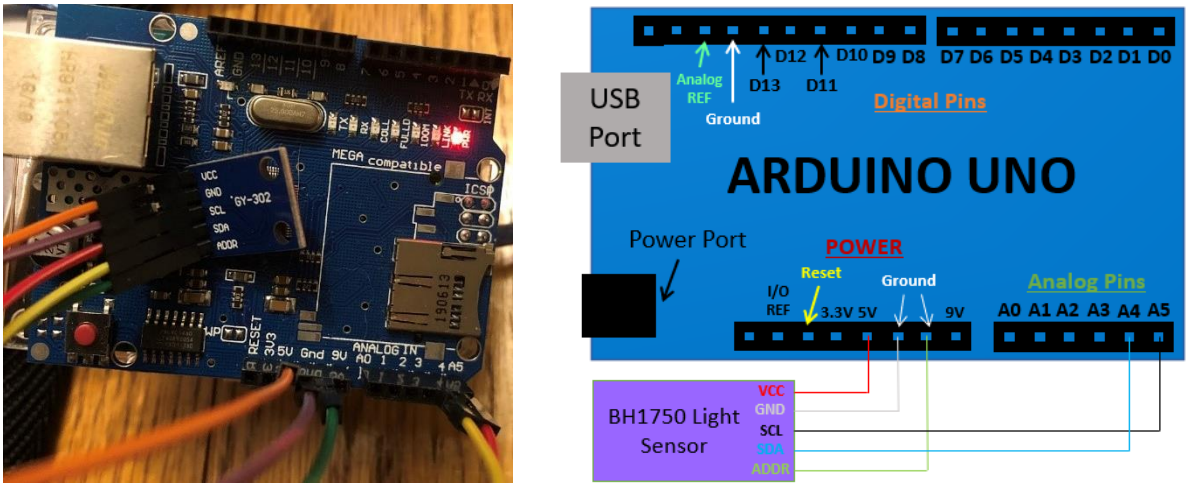

Figure 5. Picture and wiring diagram of the BH1750 light sensor circuit.

The output generated from this sensor (code shown in Appendix B) can be seen in the sample data taken from the Arduino serial monitor (**Figure 6**). As with the thermal sensor, there is an on-board signal conditioning that interfaces to dedicated code that interprets the voltage outputs to a calibrated signal. This major change from historic do-it-yourself circuits – built from a combination of photodiodes, resistors, and capacitors – has greatly simplified the use of low-cost PLCs for applications in an environmental setting.

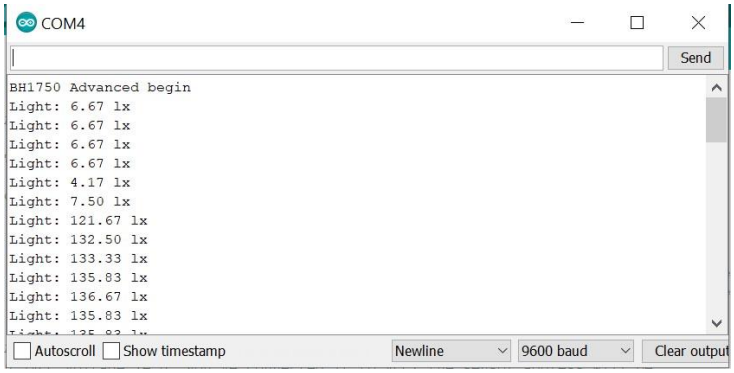

Figure 6. Serial monitor display for the BH1750 light sensor.

Chapter 3

Testing the Sensors

Once the key circuitry and processing equipment were selected, testing for the capabilities and limitations of all of the sensors was conducted. These experiments were essential for the determination of sensor operational boundaries (upper and lower bounds to signal danger for all related equipment) which were implemented within the webserver code (Appendix B). Since the sensors varied widely with sensitivity, consistency, and response time, it was essential that the sensors be studied thoroughly before utilizing them as safety nets for the lab equipment/supplies which they were designated to monitor.

Moisture Sensor Capabilities

The moisture sensor was studied both under static and dynamic conditions in an effort to more thoroughly understand these sensors. These tests were carried out so that the moisture sensors could be vetted for their application in ensuring proper watering of potted plants or more controlled execution of drought studies. During static experimentation where the soil moisture content remained constant, the time to reach steady state for the sensors was determined to be about ten minutes (displayed in **Figure 7**).

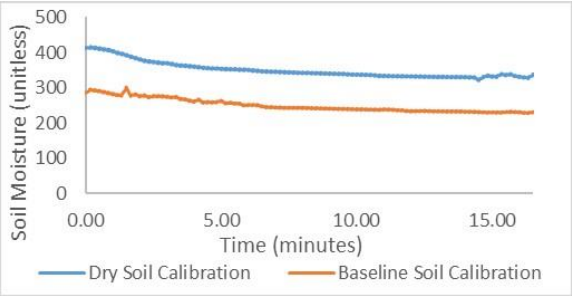

Figure 7. Determination of time to steady state for the moisture sensors.

Though equilibrium was only a consideration during the startup of the sensors, it was a crucial consideration for determining the reliability of the output. Once time to steady state was determined, static unitless soil moisture values were approximated for four conditions: overwatered soil (254.3), normally watered soil (259.7), dry soil (316.3), and extremely dry soil (764.5). These average values were determined from the moisture profiles demonstrated in **Figure 8**.

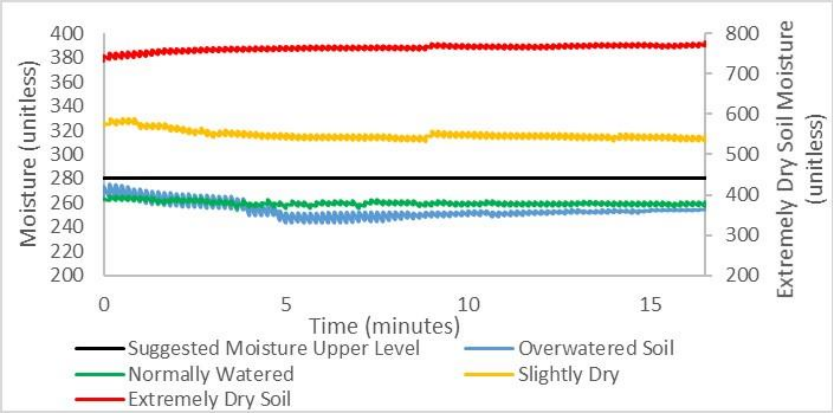

Figure 8. Calculation of upper soil permittivity bound.

These median values were determined by running Arduino programs that collected sensor data until it provided a stable output at the qualitative watering levels. From the static testing, it was clear that the soil permittivity did not exhibit a large enough difference between overwatering (254.3) and normal watering (259.7) iterations to differentiate them adequately. The sensors were simply not as precise within this range of soil moisture values. Admittedly, this measurement is based on qualitative measurements obtained from observation, and complete characterization of this behavior will require correlating the sensor output much more closely with soil moisture content. The natural deviations of the sensor were simply too great to establish a lower soil permittivity bound in order to inform lab members when certain plants were

overwatered. Alternatively, a scale/weight sensor could be utilized for determining this upper range of excess water.

However, the observed difference in the soil dielectric constant displayed a strong dependence between the normal level and both the dry and extremely dry soil measurements. For this reason, a soil permittivity upper bound was established at 280.0. The selection of this upper bound value was significant because this was the minimum value for which the monitoring code would inform the user that a given plant was too dry. In retrospect, it was quite subjective to describe what 'normal' and 'dry' conditions refer to, but even quantitative soil permittivity values are quite foreign to most people. Notably, when water is added to a pot of known size and soil quantity, the water constant is directly calculable, such that this approach to facilitating water when the pot becomes dry can be combined with simple calculations and experience to provide for consistent plant watering. Thus, it was clear that these conditions needed to somehow be described more fully when applying this for actual plants within lab. Unfortunately, the COVID-19 epidemic prevented further testing in lab to appropriately adjust the upper bound soil dielectric constant value and provide for a mass balance-based modeling approach to provide for consistent watering based on the more accurate low soil moisture range.

Further, a dynamic test was run in order to determine the response time of the sensor when water was added to the soil while recording data. The results from this trial can be seen in **Figure 9**.

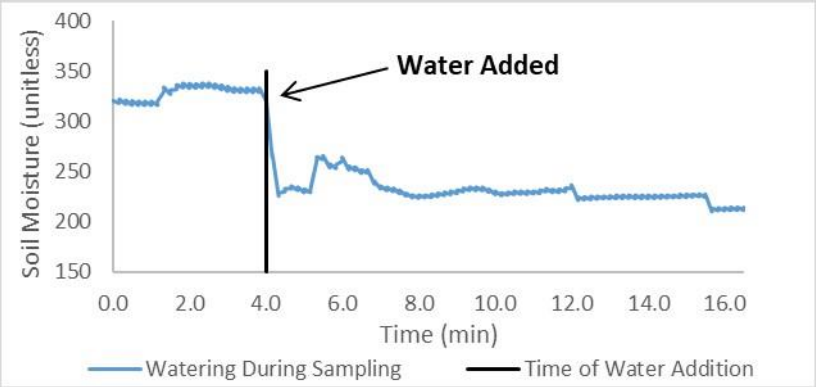

Figure 9. Determination of soil moisture response time to watering.

Relatively dry soil was monitored regularly with the soil moisture sensor until four minutes into the experiment, at which point water was added. As would be expected for an electrical-based sensor, there is a rapid change in soil conductance with watering. This large rapid drop in soil conductivity occurs within seconds, and invariably depends on the relative location of the watering relative to the probe (watering from above in this case). From there, the data clearly shows that the sensor fluctuates considerably until near the ten-minute mark (six minutes) where a quasi-steady state of the soil moisture sensor is achieved. A noteworthy comment: the watering in this case brought the sensor into the 'saturated' region where it was less accurate. Nonetheless, it became clear that sufficient water was added as a basis for faulty reporting in the event of inadequate water supply.

Although part of this time delay was inherently due to the gravity-driven permeation of water through the soil, the soil moisture sensor certainly was not instantaneous in adjusting to changing water levels (as was determined during the calibration trials). Soil is not a homogenous medium, so the distribution of water within the soil matrix will invariably occur over a period of minutes. This response is exhibited by the sensor, which continues to decline after several minutes. In the future, it is recommended that future iterations of this experiment are done on soil

that is then characterized by the soil moisture content to advance future drought studies performed in this lab.

Thermocouple Operation

The MAX6675 thermocouple was subjected to various testing in order to develop a good understanding of this temperature sensor's ability. This was essential because the thermocouples in consideration were to be used to monitor and support the operational fidelity of all the cooling units and an elevated temperature incubator within the laboratory. Both static and dynamic tests were conducted. First, a baseline trial was performed in order to record fluctuations of the thermocouple when exposed to the air at room temperature. The averaged room temperature (15.63 °C / 60.14 °F) was then calculated. A second trial was then conducted to monitor a cooling unit's temperature. To prepare for this trial, minimal insulation – tin foil around the thermocouple and cloth around the tin foil – was placed around the thermocouple, and twenty minutes was given for the temperature profile to adjust after this change. Next, the averaged fridge surface temperature was determined (15.19 °C / 59.34 °F). Both the room temperature and fridge surface temperature profiles were recorded through Arduino data collection and are displayed in **Figure 10**, where the response time of the thermocouple was also determined.

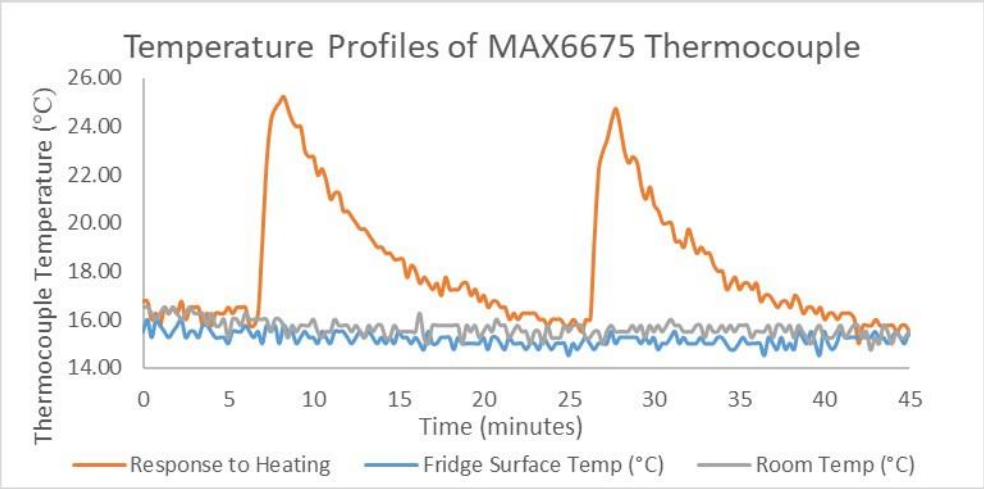

Figure 10. Testing of thermocouple to evaluate an upper temperature bound.

The intent of implementing this sensor was to monitor the temperature of freezers used in lab and to determine quantitatively, through thermocouple output, when the freezer had shut off and its contents were warming to a dangerous level. The temperature differential between room temperature and the fridge surface temperature was only 0.44 °C / 0.79 °F. This was insufficient to adequately set a 'danger' temperature level (warmer than freezer surface temperature) to use for monitoring the freezer surface temperature. Thus, this method for safeguarding freezers by monitoring them with a thermocouple failed. This conclusion was not terribly surprising, as the insulation of these units was designed to minimize thermal losses, and maintain a surface temperature close to ambient temperature. The obvious solution being the placement of probes inside the unit (which should be possibly with minimal disruption of the seals – although some cryo-freezers are designed to produce a vacuum on cooling which could be problematic).

On the other hand, the response time of the thermocouple was effectively determined, as shown in **Figure 10**. After warming the thermocouple for 1.5 minutes (from 16.00 °C to 25.00 °C), it took about fifteen minutes for the sensor to return to room temperature. When carrying out future testing with the thermocouple, it will be important to keep this delay time in mind.

In the future, more extensive use of insulation could be tested as a means by which to utilize thermal profiles from equipment to the exterior by manipulating this surface temperature. The thermocouple attached to the fridge surface inevitably equilibrated to a temperature quite similar to room temperature due to a lack of sufficient insulation. As can be observed below (**Figure 11**), the effect of additional insulation – recommending at least one inch of foam insulation for similar applications – could considerably lower the thermocouple temperature and improve the capability of a thermocouple for safeguarding a freezer within a research lab.

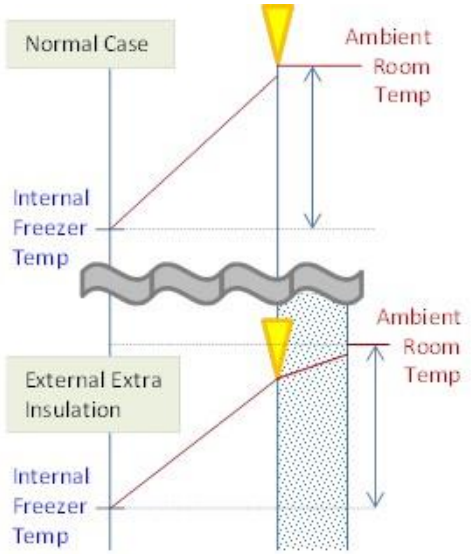

Figure 11. Expected effect of insulating a thermocouple in future applications.

Light Sensor Function

In an effort to protect many of the plants growing in CurtisLab from overhead lights turning off unintentionally, or being left on all night, BH1750 light sensors were studied thoroughly in an effort to develop a lower bound lux level which would indicate that the lights had turned off erroneously. In addition, response time and consistency of the sensors were carefully observed.

During static testing of the BH1750 light sensor, all three iterations attested to the extreme level of consistency that this light sensor exhibited. The light intensity conditions monitored were as follows (in increasing order of intensity): dark (sensor shielded from all light), six feet of horizontal separation between sensor and overhead light, and directly beneath the overhead light. The results from these iterations can be seen in the following figure (**Figure 12**).

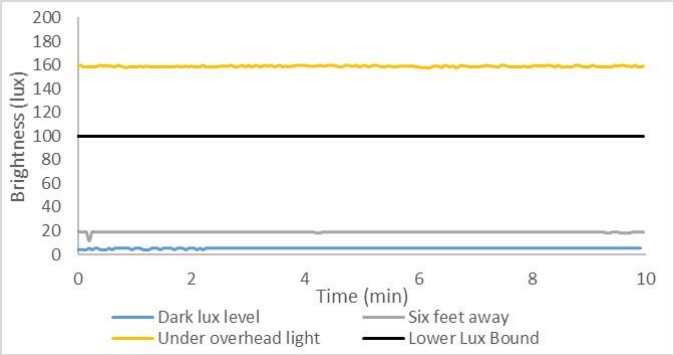

Figure 12. Determination of lower brightness bound.

From these three trials, a lower brightness bound (100 lux) is proposed for the use of the sensor in ambient room lighting changes. Considering the extreme consistency of the sensor at different lux levels, and the fact that this lower brightness bound is considerably different from the average lux of all other iterations, the lower light level bound was successfully determined in these trials. Notably, many of the lights in Dr. Curtis's laboratory (in the Chemical and Biomedical Engineering Building) utilize motion sensors. As a result, when there are after hours/late night efforts, the overhead lighting may be unavoidably turned on. A logging of this activity would be possible, where for some plants this is extremely important as the period of darkness can affect things like the flowering or germination of seeds.

Further, the response time of this light sensor was tested, and the results are shown in **Figure 13**.

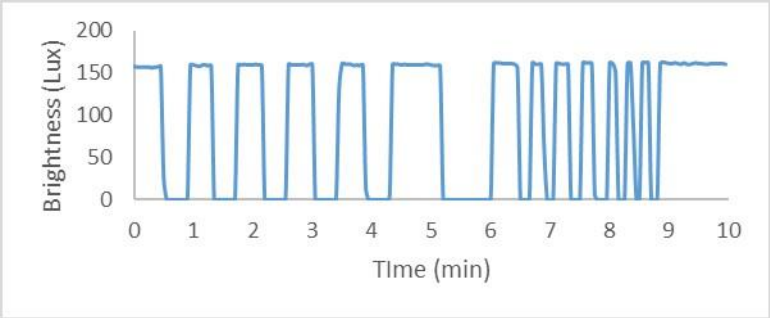

Figure 13. Response time of light sensor at varying intervals.

For the first four minutes, the overhead light was turned off and on every thirty seconds. For the next two minutes: every 1 minute. Then the light was turned off and back on again at increasingly smaller time intervals. From the above figure (**Figure 13**), it is clear that the chosen light sensor has essentially no response time delay. Considering a technology based on a photodiode and associated speeds of electrical circuits, some capacitance is invariably built into the sensor. It was opportune to verify the rapid response. Unlike thermal responses which require the transport of thermal energy, these measurements of light emittance were nearly instantaneous.

Chapter 4

Detailed Explanation of the Microprocessor-Internet Interface

In order to develop a baseline understanding of how a device transmits and receives data through the internal-external server interface, a handful of terms and concepts must be explored. Before approaching this interface, the methods for communicating within the local server must be explained in order to develop an understanding of how the local router/server labels and 'talks' to the microprocessors used within this project. Further, the means by which a device can access an external server is then critical to understanding the key role and application of an Internet-capable microprocessor.

How Devices Communicate within the Local Server

Without properly identifying all devices connected to a particular wireless network, an Internet Service Provider (ISP) router could not possibly communicate with these devices. It is for this basic reason that all devices contain unique MAC and IP addresses. The router itself also embodies several connection points that enable communication to occur. When manually connecting a new device to a server – which will most likely be the case for connecting a PLC and its Internet-capable companion – the connection requires four important sets of numbers: the device MAC address, the device IP address, the router IP address, and the network gateway.

The first of these, the MAC address, is a six-byte serial code which is given to a device, generally during manufacturing. Unexpectedly, some AES shields and other microprocessors are not assigned a MAC address. All of the AES devices used within the scope of this project were not assigned a MAC address during manufacturing, but it was quite easy to designate unique MAC addresses to these boards within the Arduino IDE code. It is important to note here that two devices on a network cannot operate with the same MAC address. For this reason, the AES or another similar microprocessor will fail to connect to the server if it does not receive a unique address. Once assigned, though, MAC addresses are

not designed to change. Further, an ISP router does not label devices with MAC addresses, but it does assign local IP addresses to each device.

An internal/local IP address is a thirty-two-bit number assigned by the router identifying a device on the Internet. It can and does frequently change when a device is disconnected from the network, but remains the same while a device is connected to the router. Notably, a router has both an internal and an external IP address. It is the internal address which communicates with devices on the local network, and this local network encapsulates only the devices which are in physical proximity and are connected to it. IP addresses can be broken down separately into static and dynamic categories – both of which may characterize both internal and external IPs.

A static IP address is one that does not change, and generally provides for better upload and download speeds, as well as improved geolocation services. On the other hand, utilizing the same IP address poses a greater security risk and is less cost effective for the server. Hackers and other security threats can retrieve information about a device from its IP address, and thus can more easily track down a device if it always uses the same address. The security of sensor data is not an especially great security risk within CurtisLab, but it is something that should be considered before deciding to manually override the IP designation process and assign a static IP address to a microprocessor.

For a network – especially ones with more traffic – a process called Dynamic Host Configuration Protocol (DHCP) is utilized to assign dynamic IP addresses to different devices using the server. This way, IP addresses are leased when needed, and can be terminated when a device disconnects from the network. By using a DHCP, a network is capable of having many more devices/users than allowable IP addresses. Suffice to say, the AES or other microprocessor will inevitably be given a dynamic IP address while communicating with the server. To improve the dependability of the Internet-capable device, the router – at least on a home server, likely not for a WPA2 Enterprise server – can be modified through the Internet to always assign the same IP address to a certain device. This varies with ISPs, but generally can be done by typing in the router's IP address into a web browser and modifying the router's IP distribution

settings from there. This can be an extremely useful tool to improve code execution while troubleshooting sensor and webserver problems.

The fourth and final connection component is a gateway, which is a node at which data transmissions halt while entering or coming back from other networks. The default gateway of a network usually exhibits the following IP address: 192.168.1.1 (this can facilitate webserver code execution). This node exists within the modem/router supplying the server, in addition to any hub which collects data from a variety of different devices. The gateway of a server is the ISP that allows for access to the entire Internet. When a server acts as the gateway, it also acts as a firewall and a proxy server. A firewall protects unwanted outside entities from entering the private network (especially through undesignated ports – which will be described in full in section 4.2), and a proxy server that mediates internet traffic flow between a device and the website the user browses.

Additionally, within the scope of the local network resides the basic output of the AES webserver programs. The AES relies on two forms of output: messages located on the serial monitor within the Arduino IDE software, and information which appears on the AES-generated webpage through Hypertext Markup Language (HTML) code. Both of these monitors will only contain information while the Arduino program is running. Information outputted onto the serial monitor is only accessible on the computer which runs the Arduino IDE software. Alternatively, the HTML webpage generated by the AES – or other Internet-capable device – can be viewed by any devices within the local network.

This collection of information describing the local network forms a sound foundation for explaining the process by which a router transmits information to and receives information from an external server.

How the Local Server Communicates with External Servers

A router has both an internal/local and an external/global IP address. The external, dynamic IP address attributed to a router by the ISP is what allows all devices within a local network to receive and transmit information, through mail/messaging or accessing websites or otherwise. This process, called Network Address Translation (NAT), is what allows many internal devices within a local network to use the Internet under the guise of a single external IP address (**Figure 14**).

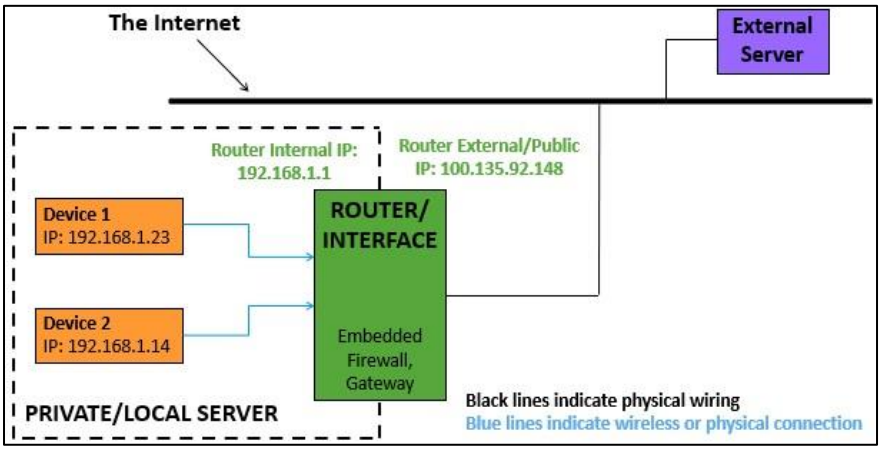

Figure 14. Inter-server communication through the Internet.

A NAT router's global IP connects to the Wide Area Network (WAN) interface, the side of the internal/external interface that communicates with the global Internet. Most external networks are configured in the same manner as a local network (**Figure 14**), and are connected to each other through physical cables.

Within the realm of laboratory automation, an Internet-capable microprocessor has considerably greater utility when it is possible to access and manipulate sensors attached to the microprocessor while outside of the local network within the laboratory. Although, the firewall of local servers generally prevents external devices from accessing the computer services. This can be overcome by two main methods: the use of an API or port forwarding.

An Application Program Interface, or API, is a software which acts as an intermediate source between two different applications, on the same or different servers. Generally, APIs are used to allow for

one software to rely on the functions/pre-existing structure from another software, without requiring development of unnecessary code to accomplish certain tasks. The API does not provide all of the code/information from one software to another, but rather supplies data (created by the programmers) needed to accomplish the given task. In the case of lab automation, one of the most useful instances of an API is one which pairs the webserver program from the AES with various messaging services. Within the local network, the Internet-capable device can connect to the API server through a Hypertext Transfer Protocol (HTTP) request and transmit/receive data from lab members. This functionality allows the AES to communicate with outside servers in a way that is not possible simply from the utility of the AES-generated webpage.

Additionally, port forwarding provides another potential method by which a local microprocessor can communicate outside of the local network. Port forwarding enables remote computers/devices to connect directly to a specific device within a private Local-Area Network (LAN). Generally, the firewall of a private network allows local devices to connect with servers outside of the local network (designated with the external IP address of the router), but it rejects requests from external entities to establish connections with devices on the local server. Devices are able to connect to the internet through their 'Public' IP address – the external IP address of the router – but their internal, private IP address remains protected in this exchange. One of the most common applications of port forwarding – which is the desired application for this project – is enabling a device to transmit a public HTTP server within a private LAN.

A server has 65,535 ports, which corresponds to the maximum allowable number that a sixteen-bit number can represent. The server reserves all port numbers below 1024 and above 49,150, so these cannot be utilized via port forwarding. The most common, allowed ports include: 23 for Telnet (transmits unencrypted text messages), 67 and 68 for DHCP, 80 for HTTP web servers and web pages, 143 for Internet Message Access Protocol (IMAP), and 443 for HTTP Secure (HTTPS). In order to establish a connection between devices on separate servers, a router administrator can reconfigure the gateways'

operating system to allow for one port number on the gateway to be exclusively used for communicating with a service on the private network. Unfortunately, this generally involves either weakening the firewall or deactivating it completely while the AES or other Internet-capable device is running. On a home network, this approach most likely imparts minimal risk onto the devices on the network. On the other hand, if a project relies on the LAN of a University or other large institution, then this security risk would be too great to consider this approach.

Having explained the details involved for connecting an Internet-capable device to both local and remote servers, the results of the AES webserver experiments can be understood to a much greater extent.

Chapter 5

Results and Impact of the Project

The outbreak of COVID-19 and the subsequent lockdown of the Chemical and Biomedical Engineering Building prevented the majority of personnel from entering the building; subsequent implementation of the tested sensors and AES devices within the laboratory would have been conducted under different circumstances. Fortunately, the webserver programs were completed and tested in a home environment. The results indicated a successful future transition of all the equipment to provide low-cost monitoring and even data logging in the lab environment.

Establishing a connection to the router was a prerequisite for running any webserver code. Without a connection to a client and to the Internet, any of the safeguarding code (for which the operational bounds were established from Chapter 3) would fail to communicate anything to lab members monitoring the sensors from within the lab. This step required establishing a connection to the Internet for a device which previously contained neither a MAC address nor an IP address. Fortunately, scanning a network for all devices (which is essential for determining which MAC and IP addresses are not taken) was executed through several command prompt commands: “arp -a” and “ipconfig.” It was then possible to establish a connection to the internet through a webserver program run on the AES. Once the program was uploaded onto the Arduino board from a computer with the necessary Arduino IDE coding software, the AES successfully connected to the local network and waited until a client connected to the webserver in order to output sensor data. The program then waited until a client was established, and this was done by inputting the AES IP address into a webserver. It is important to note here that while waiting for a lab member to type the Arduino IP address into a web browser while on the

local network, the AES does nothing. Basic webserver client output is shown in **Figure 15**. Simple webserver code is shown in Appendix B.

Figure 15. Basic webserver program HTML output.

The webserver program code was written for each sensor described in Chapter 2 by combining the code of individual sensor programs with that of the Arduino default webserver program. This code enabled data transmission from the sensors to the local server. Conditional statements were included in the code so that error messages could be outputted to the user (through the webserver) if sensor data indicated that equipment was operating under undesirable circumstances. Due to the social distancing practices occurring during the COVID-19 epidemic, all of the webserver experiments were executed in a home setting. An example of a specific webserver application is shown for the MAX6675 thermocouple below (**Figure 16**).

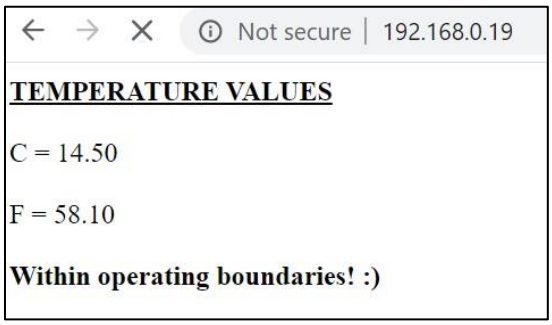

Figure 16. Sample HTML output of thermocouple webserver program.

One problem remained: to realize how to transmit data from the local network to the external Internet so that researchers could access the sensor data from anywhere. Without a

means to transmit data outside the network, the information being transmitted by the sensors would only be visible by users on the local network.

Two possibilities were explored fully: the use of an API and a method called port forwarding. The API was the preferred method for this project, as it was the most common technique for transmitting sensor information outside the network. Most Arduino forums (which compile issues for Arduino home projects) encouraged users to utilize the Pushingbox API in order to transmit AES data outside the local network. However, once the Pushingbox API code was executed, it was determined that the network of this specific API was shut down. This API code stalled indefinitely, and when the 'current' IP address of the Pushingbox router was entered into a web browser – which is publicly available – the results claimed that the IP address for this router was not functional (**Figure 17**).

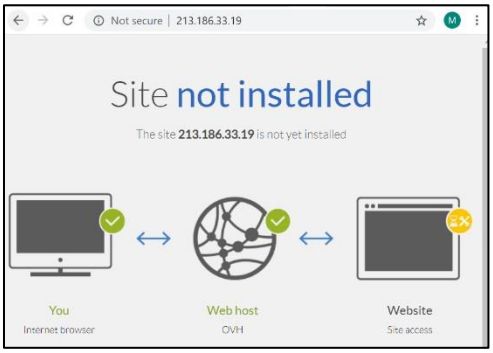

Figure 17. Indication of inactive status for Pushingbox API server.

The developers of Pushingbox did not respond to formal inquiries. Since no other renowned API services could be located, this method was unfortunately discarded. In the future, contacting the developers of this API again and potentially reaching out to other companies would be extremely useful for transmitting data from the sensors to outside of the lab.

Port forwarding was studied as a possible alternative for locating sensor information from outside the local server. Unfortunately, brief research into the port forwarding method revealed a

fatal flaw to this approach: this approach required severely downgrading or disabling the network firewall while the AES was in operation. A router constantly transmits data from the local to the external server, and vice versa. This is the reason why a router has both a local and an external IP address. However, for security reasons, routers are usually preset to only allow data to be transmitted into or out of the local server through several enabled ports – as explained in Chapter 4. Transmitting AES data outside the local network would require assigning a unique port to the operations of the shield, and then disabling the firewall on the router in order to allow it to forward information from the local to the external server through this unique port. Thus, this project was thus far unsuccessful in its ambitions to select an appropriate method for transmitting AES data from the local network to the world wide web. However, for projects at home or those being carried out in laboratories which have a unique local server, the possibility of disabling the firewall (assuming this would pose no great security threat to the lab members by doing so) would be a much more feasible option.

Chapter 6

Conclusion and Future Expansion of Existing Methods

The testing of all the equipment in this lab resulted in both successes and failures from the goals stated at the beginning of this project. The moisture and light sensors were tested successfully, and operational boundaries were established for both of them so that they could be implemented successfully in the lab. However, the soil moisture sensors were not calibrated to meaningful operating conditions, and this must be pursued in the future of this project. Also, the webserver programs for the AES were successfully executed. The current webserver results allow CurtisLab members to visualize sensor output from anywhere within the local network of the Chemical and Biomedical Engineering Building. Additionally, a lab member could remotely log into a computer from outside the lab to visualize sensor output. On the other hand, the MAX6675 thermocouples require further testing in order to utilize them to protect cooling units within the laboratory, and the attempts to transmit AES data beyond the local network were unsuccessful.

Regardless of the results of this experiment up until this point, the future of this implementation of this project has tremendous potential to assist all CurtisLab members in their research. Once this period of national social distancing ends, all of this equipment monitoring can be applied to the laboratory. Additionally, the soil moisture sensor could serve a potentially vital role, on a quantitative basis, in calibrating the watering requirements of *Nicotiana benthamiana*. This plant, even when watered daily, rapidly wilts due to its sensitivity to soil moisture changes (Senthil-Kumar et al., 2007). In order to achieve absolute calibration for the BH1750 light sensors, it will be necessary in the future to compare output from the BH1750 to that of the LI-COR pyranometer which is readily accessible in CurtisLab. By either placing

probes inside incubators/freezers or manipulating the thermocouple thermal gradients (adding thick insulation around them), these sensors could effectively enhance the operational fidelity of the freezers within the laboratory. Continually, future research into APIs and port forwarding could possibly reveal an alternative method for transmitting AES data outside of the local network, which would allow for continuous support of equipment (rather than being limited to safe checks only when lab members are physically present).

Beyond the equipment and processes tested during this project so far, many possibilities exist for expanding upon the foundation established here. For one, pressure and humidity sensors could be tested and applied via the same basic lab automation practices conducted within this project and by many others before. Also, the lab could invest in SD cards for the AES units, as this would allow for these units to conduct extensive data logging experiments. PLCs are extremely limited in their ability to process things (RAM/dynamic memory is considerably less powerful than that of a microprocessor), and an SD card would enable them to function better in experiments where logging and analyzing data over long periods of time is desired. In addition, obtaining SD cards would enable the lab to perform experiments based in artificially intelligent Arduino code. The beauty of running neural networks for key variables is that they can learn from data collected and provide input for lab personnel to better conduct these data collection experiments (Hinton & Salakhutdinov, 2006). All in all, the implementation of these considerations has the potential to augment lab performance and intellectual growth within Dr. Wayne Curtis's lab.

Appendix A

Troubleshooting the PARTICLE Photon

The PARTICLE connection protocol, embedded within the command prompt, ultimately led to the same dead end – regardless of which pathway was taken. After updating and uploading the proper code onto the PARTICLE Photon, the program would prompt the user to select an authentication type, EAP (Extensible Authentication Protocol) or TLS (Transport-Layer Security). The network of the Pennsylvania State University relies on an EAP-TTLS authentication type. When selecting either authentication type, each method would inevitably request that the user inputted the CA (Certificate Authority) Certificate of the network, in PEM (Privacy Enhanced Mail) format – also known as sixty-four bit/base64 x.509 format. However, the only means to input the PEM format was through Notepad (which the PARTICLE program would prompt to open).

Onward: a CA Certificate is used to validate the legitimacy of the gateway for a device, which is needed to connect to the network. After receiving the base64 CA Certificate from the Penn State IT staff (which generally remains encoded), this text file was loaded into the Notepad application. However, there was no clear way to upload this Notepad file into the PARTICLE application. After many failed attempts, I consulted the PARTICLE staff regarding this issue.

After discussing this with PARTICLE engineering staff, it was learned that the Photon platform simply was not designed for WPA2 Enterprise networks due to the fact that their Cypress WICED networking code had bugs and was closed source. Because the software was closed source, the developers physically could not modify it to enable Photons to connect consistently with WPA2 Enterprise.

Appendix B

Arduino Sensor Programs

Soil Moisture Sensor Code

```
#define SensorPin A0
#define SensPin A1
float sensorValue = 0;
float sensVal2 = 0;
int num = 1;
void setup() {
 Serial.begin(9600);
 Serial.println("HiLetgo Soil Moisture Sensor!\n");
}
void loop() {
 for (int i = 0; i < 100; i++)
 {
   sensorValue = sensorValue + analogRead(SensorPin);
   sensVal2 += analogRead(SensPin);
   delay(1);
 }
 sensorValue = sensorValue/100.0;
 sensVal2 = sensVal2/100.0;
 Serial.print("A0: ");
```

```
 Serial.println(sensorValue);
 Serial.print("A1: ");
 Serial.println(sensVal2);
 Serial.println();
 delay(3000);
          }
```

Thermocouple Code

```
#include <max6675.h>
int soPin = 4;// SO = Serial Out
int csPin = 5;// CS = Chip Select
int scPin = 6;// SC = Serial Clock
//Creates a thermocouple type which relies on the max6675 library functions
MAX6675 tCoup1(scPin, csPin, soPin);
void setup() {
  Serial.begin(9600);// initialize serial monitor with 9600 baud
  Serial.println("MAX6675 Type K Thermocouple");
}
void loop() {
  // Outputs current temperature
   Serial.print("C = ");
   float Cels = tCoup1.readCelsius() - 7.0;
   Serial.print(Cels);
   Serial.print(" F = ");
```

```
  Serial.println((Cels*1.8 + 32.0));
  delay(5000);
        }
```

Light Sensor Code

```
//Wire library allows you to communicate with I2C devices
#include <Wire.h>
#include <BH1750.h>
//Declaring an object of the BH1750 type
BH1750 lightMeter;
void setup(){
  Serial.begin(9600);
  //Initializing the I2C bus
  Wire.begin();
  lightMeter.begin();
  Serial.println("BH1750 Test");
}
void loop() {
  float lux = lightMeter.readLightLevel();
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lux");
  delay(3000);
}
```

Ethernet Shield Webserver Code

```
include <SPI.h>
#include <Ethernet.h>
//unique mac address for your shield
byte mac[] = {
  0xAB, 0xAF, 0x69, 0xAE, 0x3A, 0xC4
};
// Initialize the Ethernet server library
EthernetServer server(80);
void setup() {
  Ethernet.init(10);  // Most Arduino shields
  Serial.begin(9600);
  while (!Serial) {
    ; //does nothing forever if the serial monitor cannot connect
  }
  Serial.println("Ethernet WebServer Example");
  // start the Ethernet connection and the server:
  //Ethernet.begin() might only work if you input the shield IP address, in the form
  //{192,168,1,X}
  Ethernet.begin(mac);
  // Check for Ethernet hardware present
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
    while (true) {
      delay(1);
```

```
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
  }
  // start the server
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
```

```
          //Connection closed after completion of the response
          client.println("Connection: close");
          //refreshes the page every 5 seconds automatically
          client.println("Refresh: 5");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin. Should write in MAX6675/BH1750/moisture sensor code here with conditional statements to monitor proposed upper/lower bounds
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
```

```
        currentLineIsBlank = false;
      }
    }
  }
  // give the web browser time to receive the data
  delay(1);
  // close the connection:
  client.stop();
  Serial.println("client disconnected");
 }
}
```

BIBLIOGRAPHY

[1] Curtis, M. S. (2013). Novel temporary immersion bioreactor allows the manipulation of headspace composition to improve plant tissue propagation. In *Electronic Theses for Schreyer Honors College*. https://honors.libraries.psu.edu/files/final_submissions/2009

[2] Egertsdotter, U., Ahmad, I., & Clapham, D. (2019). Automation and scale up of somatic embryogenesis for commercial plant production, with emphasis on conifers. In *Frontiers in Plant Science*. https://doi.org/10.3389/fpls.2019.00109

[3] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*. https://doi.org/10.1126/science.1127647

[4] Rosso, V., Albrecht, J., Roberts, F., & Janey, J. M. (2019). Uniting laboratory automation, DoE data, and modeling techniques to accelerate chemical process development. *Reaction Chemistry and Engineering*. https://doi.org/10.1039/c9re00079h

[5] Senthil-Kumar, M., Govind, G., Kang, L., Mysore, K. S., & Udayakumar, M. (2007). Functional characterization of Nicotiana benthamiana homologs of peanut water deficit-induced genes by virus-induced gene silencing. *Planta*. https://doi.org/10.1007/s00425-006-0367-0

[6] Thramboulidis, K. (2015). A cyber-physical system-based approach for industrial automation systems. *Computers in Industry*. https://doi.org/10.1016/j.compind.2015.04.006

[7] Tsai, H. Y., Su, F. C., Chou, C. H., Lin, Y. H., Huang, K. C., Yang, Y. J. J., Kuo, L. W., Liao, L. De, & Yu, H. S. (2019). Wearable Inverse Light-Emitting Diode Sensor for Measuring Light Intensity at Specific Wavelengths in Light Therapy. *IEEE Transactions on Instrumentation and Measurement*. https://doi.org/10.1109/TIM.2019.2899444

ACADEMIC VITA

MICHAEL JOSEPH COVER

OBJECTIVE

To summarize my achievements as a chemical engineering student at the Pennsylvania State University in the Schreyer Honors College.

EDUCATION

- Bachelor of Science, Chemical Engineering
- The Pennsylvania State University, University Park, PA
- Schreyer Honors College
- Expected Graduation: May 2020

RELEVANT COURSES

- Engineering Design; Chemical Engineering Material Balances, Thermodynamics, Phase Equilibria, Fluids, Heat Transfer, Process Safety and Senior Plant Design

WORK EXPERIENCE

Busser, Whitford Country Club, Exton, PA (May 2014 – August 2016)

- Utilized organizational skills to ensure success of banquets and dinners.
- Recruited and trained multiple employees which went on to become valuable assets to the Whitford Country Club
- In absence of pool manager: carried on his responsibilities, and efficiently coordinated opening and closing procedures, leading a team of peers.
- Six years of Spanish before college allowed me to communicate more effectively with employees whose first language is Spanish

Midshipman, Department of Defense (August 2016 – Present)

- Mandatory workouts once weekly, and drill practice twice weekly from Fall 2016 – Spring 2018
- Tutored other midshipmen for Mathematics, Physics, Chemistry, and English from Spring 2017 – Fall 2018
- Served active duty from May 17th, 2017 – June 17th, 2017 while at Career Orientation and Training for Midshipmen (CORTRAMID) in San Diego, CA
- Chosen as the sophomore midshipman of the Fall 2017 semester
- Active duty in June 2018 in Jacksonville, Florida.
    - Learned what chemicals are used and how to deploy them to extinguish all types of fires

COMPUTER SKILLS

	C	C++	Java
	SolidWorks	Excel	Mathematica
	Fluent	Arduino	Raspberry Pi

ACTIVITIES & AWARDS

Member, National Honor Society (Spring 2015)
National NROTC Scholarship (2015)
Dean's List (Fall 2016-Spring 2019)
Navy ROTC Vice President (Fall 2019)
IM Frisbee Captain (Fall 2017 and Fall 2018)
Participant of the PNC Leadership Development Center (Spring 2018)
President of the Penn State Nuclear Navy Club (Spring 2018-Fall 2018)