UPPSALA UNIVERSITET

1TE774 Elektronikprojekt - SmartGrowSystems

By: Max Fridh, Johan Haglöf, Linus Määttä, Haza Newman, Leo Niemöller

13 Jan - 2024

Abstract

This report details the development of a 'Smart Grow System,' integrating analog and digital electronics in order to achieve a self-sustaining growth environment, specifically targeted for plants. Utilizing horticulture LEDs (HLEDs) and a pneumatic pump watering system, this project aims to mimic a natural environment for said plants using both analog and digital electronics.

This project followed a strict project development plan, including a detailed timeline, group-meetings and task allocation within the group. The technical implementation is built on using key components such as HLEDs, Arduino, MagI³C-LDHM LED, soil moisture sensor and various electronics components and circuits.

The results stand as proof of the project's success, showcasing a fully functional ‘Smart Grow System’. A 3D-printed modular enclosure, mobile app, self-controlling watering and cooling system as well as established plant growth. Suggestions for future projects within the targeted field are, to list a few, avoiding PCB design errors, improving or accounting for 3D printing accuracy and increasing the implementation of the app.

Table of contents

Introduction

Purpose

This project's core purpose is to engineer and deploy a ‘Smart Grow System’, that harmoniously integrates analog and digital electronics in order to facilitate the growth of simple plants and spices. SmartGrowSystems utilizes horticulture LEDs (“HLEDs”) in different colors and wavelengths along with a pneumatic pump watering system to mimic a natural environment. This allows for the plant to grow regardless of its surrounding conditions. This endeavor aims to leverage simulation software for electronic circuit analysis, facilitating seamless data acquisition from sensors, precise data management using a microcontroller, and the merging of analog and digital methodologies. As aspiring electrical engineers, our primary objective is to comprehend and apply the intricacies of circuit design, signal acquisition, manipulation, and control within the context of our Smart Grow System. This pursuit not only forms the cornerstone of our educational journey but also equips us with indispensable skills to innovate and contribute to the dynamic landscape of electrical engineering.

Goal

The project's primary objective is to harness analog signals to control physical devices using both analog and digital signals. A secondary aim is to analyze the project's findings and draw insightful conclusions.

Demarcations

Temperature and humidity control and growth monitoring via a camera were purposely omitted from our initially proposed project plan to maintain focus on essential components like HLED circuitry, Arduino-controlled systems, and 3D enclosure construction. This decision aimed to streamline our objectives and align with the project's specified scope.

Background

In recent times, indoor plant cultivation has surged, serving horticulture and food production needs [1]. Electronic devices, incorporating intricate electrical components such as amplifiers, filters, and sensors, play a crucial role in revolutionizing indoor farming practices. These devices have found uses in both large scale industrial production to drive large outputs of plants for nourishment and decoration, as well as modest uses for the private consumer. This technology orchestrates and allows the creation of controlled environments by managing light, temperature, humidity, and nutrients. Within these systems, analog and digital signals serve as fundamental elements. While digital methods dominate for their precision, analog control remains vital for the many needed adjustments in indoor farming setups. This blend enables precise environmental control crucial for plant development, marking the synergy between analog and digital realms as pivotal for optimal growth conditions in indoor farming systems. The amalgamation of electronics and agriculture signals a promising future, aiming to enhance crop yields, resource efficiency, and sustainability in farming practices.

Method and Theory

Method

Our approach to developing the Smart Grow System involved a systematic process that combined research, collaborative efforts, and methodical construction. The project was divided into distinct phases, each contributing to the system's comprehensive design and functionality.

1. **Objective Identification:**
    - Our project aims to achieve the following key objectives:
        - Design and implement an HLED circuit tailored for horticulture needs, ensuring optimal plant growth conditions based on plant stage.
        - Integrate various components controlled by Arduino microcontrollers, including the fan, moisture sensor, and water pump, to automate and regulate the environmental parameters necessary for plant cultivation.
        - Fabricate a compact and minimal protective enclosure using 3D printing technology to house and safeguard the entire system, ensuring durability and efficient functionality in diverse environments.
2. **Project Plan:**

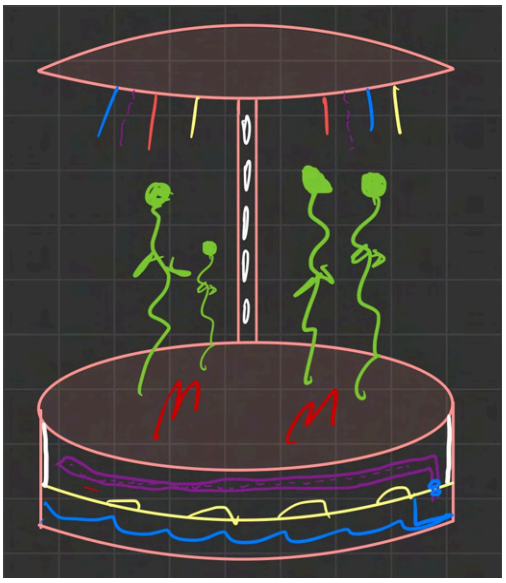


Figure 1: Initial draft of the whole project, which was used in the project plan.

    - A previously submitted and accepted project plan detailed the SmartGrowSystems initiative, outlining objectives, division of responsibilities among team members, identified components, and the proposed timeline. The plan highlighted sequential phases from research and prototyping to system testing and final presentation, with the goal to adhere to the budget of 1999:- SEK. Initial draft in the project plan is illustrated in Figure 1.

3. **Task allocation:**
    - Assigned specific technical areas to team members based on their expertise: 3D Design, Cooling System, Arduino Uno R4, Self-Watering System, and Horticulture LED Grow Lights.
4. **Phased Approach:**
    - *Research and Knowledge Gathering:*
        - Conducted extensive research to gather essential information and understand system requirements.
        - Utilized simulation tools like LTSpice and Fusion 360 for design and simulation.
    - *Design and Simulation:*
        - Developed intricate circuit designs and conducted simulations to finalize component specifications.
    - *Circuit Construction and Programming:*
        - Assembled test circuits on a breadboard to ensure proper functionality and correct wiring.
        - Components were rigorously tested to ensure seamless integration.
        - Write the Arduino code that will control and automate the system.
        - Design an interactive smartphone interface.
    - *Troubleshooting and Implementation:*
        - Resolved issues and implemented functional code to control the circuit, primarily in the laboratory setting at Ångström Laboratory, Building 7k.
5. **Collaborative Efforts:**
    - Despite each team member being focused on their specific area of expertise, each member contributed to programming, research, component organization, modeling, implementation, and report writing.
    - Regular discussions and ad-hoc meetings were held to share information, explain design decisions, address potential issues, and explore alternative solutions.
6. **Milestones and Timeline:**
    - Week 1-2: Research and finalization of component specifications including the initial design phase of the 3D designed enclosure.
    - Week 3: HLED circuit prototype development, Arduino-controlled fan system implementation, and 3D print design finalization.
    - Week 4: Capacitive moisture sensor integration, water pump system design, and testing.
    - Week 5-6: System assembly, troubleshooting, and comprehensive testing.
    - Week 7 (December 20, 2023): Final presentation and demonstration of the fully functional Smart Grow System.
    - Week 8-9: Writing the report.
7. **Location:**
    - The majority of this work took place in the laboratory at the Ångström Laboratory in Building 7.

Block Diagram

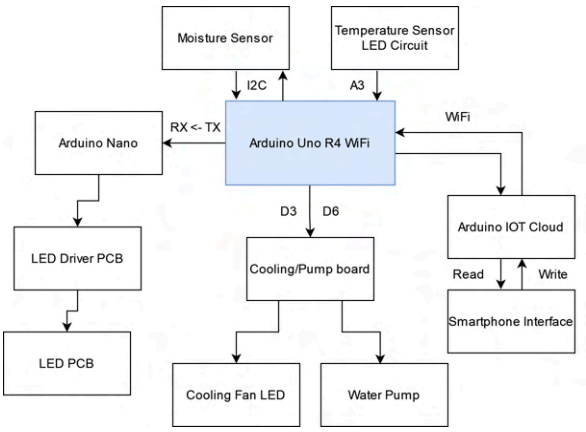


Figure 2: A block diagram illustrating the whole system.

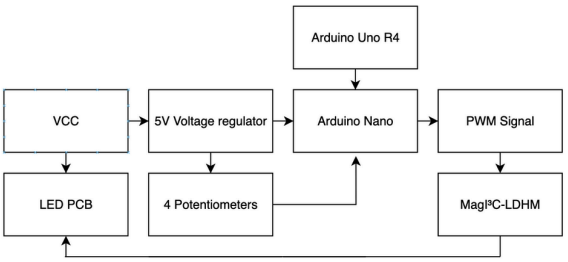


Figure 3: A flowchart of the how the HLED grow light system is connected along with the other individual subcomponents

Figure 2 is an overarching block diagram illustrating how all the different compartments are connected. The HLED system is illustrated in a more detailed diagram in figure 3.

- **Arduino UNO R4 Wifi:**
  - Works as the central controller of the system and contains the vast majority of Arduino code. Receives and processes signals from sensors and transmits data to all the other compartments in the project.
  - It communicates with the Arduino IOT Cloud via WiFi, and the cloud is able to READ/WRITE variables in the Arduino code set by the Smartphone Interface.
  - Sends information to the Arduino Nano through Serial communication (TX→RX)
- **Watering System**
  - Pump board contains a custom made transistor switch circuit that turns on the pump after receiving a digital signal (D6) from the Arduino UNO.
  - The automatic water system is reliant on the moisture reading from the Moisture sensor (I2C). The pump will activate when it dips below a set moisture level from the Smartphone interface.
- **Cooling system**
  - The temperature sensor is located on the LED circuit board. Readings from the sensor (A3) are used to create a fan curve for the 5V-cooling fan, in order to lower the noise levels while giving sufficient cooling.
  - The Arduino sends a digital PWM signal (D3) to the Cooling circuit, a custom made transistor switch circuit. The PWM signal will dictate the speed (RPM) of the 5V cooling fan.
- **HLED Grow Light System**
  - Consists of two custom made PCB boards designed in Fusion 360.
  - The LED circuit consists of 16 high powered horticulture HLEDs. They are separated by color and connected in 4 separate series circuits.
  - The other board functions as the LED driver PCB. It houses the Arduino Nano that takes in analogue signals from 4 separate potentiometers which value dictate the PWM signal out from the Arduino Nano. Each PWM signal has the capability to change the brightness of each individual color of the LED.
- **3D enclosure**
  - The whole system will be encapsulated in a custom design made in Autodesk Fusion 360. Everything will be 3D printed with PLA-plastic.

Theory and components

Arduino

Arduino, an open-source microcontroller platform, is valued for its user-friendly and versatile design, featuring embedded processing capabilities and multiple input/output pins. This enables users to interface with various physical components, making it a crucial tool for prototyping and creating projects spanning from basic LED applications to advanced Internet of Things (IoT) innovations, suitable for both newcomers and experienced electronics and embedded system engineers [2]. Comes in various sizes with ranging functionality depending on its use case and environment.

MagI³C-LDHM LED Step Down High Current Module

The MagI³C-LDHM LED Step Down High Current Module, engineered by Würth Elektronik, stands as a compact and efficient power module tailored for high-current LEDs. Its significance lies in providing precise and stable current output, offering the capability to be pulse-width modulation (PWM) controlled. This feature enhances its versatility, enabling seamless integration into applications requiring PWM steering for optimized LED performance in diverse lighting scenarios.

Soil moisture sensor capacitive I2C

The Soil Moisture Sensor Capacitive I2C by Electrokit is a superior alternative to traditional resistive soil sensors. The probe consists of two metal sheets being fully encased in resin, avoiding oxidation issues seen in resistive sensors with fully exposed metal. These metal sheets will form an electric field that induces a capacitance. Because water has a higher dielectric constant than air, the presence of water will increase the induced capacitance. [3] With its capacitive touch measurements and I2C connectivity to the Arduino, it offers moisture readings ranging from very dry to very wet without soldering, simplifying connection to various microcontrollers. It was connected to the Arduino Uno using a JST 4-PH cable.

WL-SMDC SMT Horticulture HLED

The WL-SMDC SMT Mono-color Ceramic LED Waterclear, developed by Würth Elektronik, is a compact and efficient ceramic LED module designed for mono-color applications. Operating within a voltage range of 2.8V to 3.6V, this LED module delivers reliable light output while maintaining a wattage output of 2.2 watts suitable for the intended application. With its specific current requirements of 350 mA and controllable pulse-width modulation (PWM) capability, it seamlessly integrates into various lighting scenarios, optimizing LED performance as per specific requirements. Will be referred to as “HLED” as it is specifically designed for horticulture purposes.

Linear Voltage Regulator

The LM7805CT Fairchild Linear Voltage Regulator is a reliable and efficient voltage regulation component designed to provide a steady 5-volt output. Operating within specified voltage and current parameters, this regulator ensures a consistent voltage output from a higher input voltage source, making it suitable for numerous electronic applications.

PWM

PWM (Pulse Width Modulation) is a modulation technique that can be used to control the brightness of LED lamps or even the speed of motors by rapidly switching the signal on and off, creating an average effect on the load [4]. The average voltage (V_avg) is defined as δ*V_max (Duty cycle * Voltage). A PWM signal consists of high and low voltages and possesses amplitude (A), time period (T_d), and pulse width (δ represents the "on" or "high" time as a percentage). The control of the PWM signal is based on the time period and pulse width. The duty cycle is calculated using the time period and pulse width with the following equation: D = δ/T_d [4]. In this experiment the PWM signal is used to control how fast the DC fan spins.

Bipolar junction transistor

A Bipolar Junction Transistor (BJT) is a voltage-controlled transistor with three terminals: Base, Collector, and Emitter [5]. Control is achieved by adjusting the base terminal voltage to meet or exceed the threshold voltage. BJTs can act as switches, and manipulating the base voltage ensures the collector current operates in the active region. Saturated, the transistor achieves its peak collector current, making it desirable for powering loads connected to the collector terminal. In the cutoff region, with minimal current flow, the load remains unpowered. In our experiment, by modulating the base voltage to keep the transistor in saturation, it effectively serves as a switch to power the load [5].

OP-Amp

An Operation Amplifier, or OP-Amp for short, is an active electronic component, mainly consisting of multiple transistors, allowing the amplification of voltage signals up to the supply voltage [6]. The OP-Amp has both an inverting (negative), and non-inverting (positive) input. In order to amplify the signal, the signal needs a feedback loop, essentially connecting the output to one of the inputs (typically the inverting input) with intermediary passive components such as resistors and capacitors [6]. In this experiment, the OP-Amp is used to amplify the voltage signal from the temperature sensor to use the full range of 0-5 volts that the Arduino analog inputs uses [2].

Temperature sensor

The LM60 temperature sensor is an active electronic component that accurately measures temperature and converts it into an analog voltage signal. Known for its high accuracy and a measurement range from -40 to 120 degrees Celsius, it offers a reliable solution [7]. Differentiating between linear and non-linear aspects in temperature sensors is essential. A linear sensor, like the LM60, establishes a direct relationship between voltage output and temperature increase $y = kx + m$. In contrast, non-linear sensors may exhibit

characteristics such as $y = kx^2 - x$, especially within specific temperature ranges. To operate, the temperature sensor requires a 5-volt power supply, with three pins—two for power supply and one for signal output [7].

Rectifier diode

A rectifier diode is an electronic component that allows the flow of electric current in only one direction while blocking it in the opposite direction. Its primary function is to convert alternating current (AC) into direct current (DC) by allowing the positive half of the AC waveform to pass through while blocking the negative half. This allows unidirectional flow of electrical current for proper operation [4]. The diode has a cathode (negative end), and an anode (positive end). In this experiment, a rectifier diode is coupled in parallel to the fan (inductive load), in order to force current in the direction from anode to cathode.

Pump

A pump facilitates the movement of liquids against gravity, typically featuring suction and output ends. Pumps, historically operated manually, utilized force to create pressure differences for liquid transfer. Nowadays, with motorized components, pumps can be fully automated, employing electrical principles like batteries and motors to control the piston. In essence, pumps create pressure differences, either through manual or electrical means, enabling fluid movement upstream [8].

Cooling fan

A cooling fan dissipates heat from electronic devices in confined spaces using a small motor and blades. The motor generates a rotating magnetic field through direct current, inducing motion in the rotor. When powered on, the blades create airflow, pulling in ambient air to cool heat-producing components. This process transfers heat to cooler surroundings, promoting thermal equilibrium and expelling the accumulated heat.

Arduino IoT Cloud

This software is offered by Arduino and offers a diverse range of features for IoT applications. It includes a built-in IDE, the capability to design interactive smartphone interfaces, facilitates Arduino connectivity to Cloud servers via WiFi, and enables the creation of READ/WRITE variables that automatically update through the cloud, making them easily implementable in Arduino code [9].

Materials

Materials			
HLED Components:	**HLED Cooling System:**	**Water Pump System:**	**Miscellaneous:**
16 Various 2.2W HLED, 3.2V forward voltage, 350 mA	LM60 temperature sensor	Submersible water pump 3V	Arduino Uno R4
MagI³C-LDHM LED Step Down High Current Module	80mm computer DC fan (5V)	Silicon tubes	Wires, connectors, etc.
48V power supply, 1.46A	BJT transistor 2N3904	BJT transistor 2N3904	Drill
A custom designed HLED PCB	Rectifier (one-way diode)	Rectifier (one-way diode)	Metal filtration 120 size mesh
A custom designed HLED Driver PCB	OP-amp LM741cm	Project board (70x90mm)	Weight scale
Arduino Nano Everyday	Project board (70x90mm)	Various Ω Resistor	
Wurth Terminal blocks Series 2141 (2-pin)	Various Ω Resistor		**Moisture Sensor:**
Potentiometers PTV09A	Terminal blocks	**3D Printing:**	Soil moisture sensor capacitive I2C
SMD Capacitors 2.2µF, rated for 50V		PLA Filament	
LM7805CT Fairchild Linear Voltage Regulators			

Figure 4: Overview of all the materials used for this project

Technical Implementation

See appendix.

Results

3D-Printed Modular Enclosure, Mobile Application, and Arduino/PCB Integration:

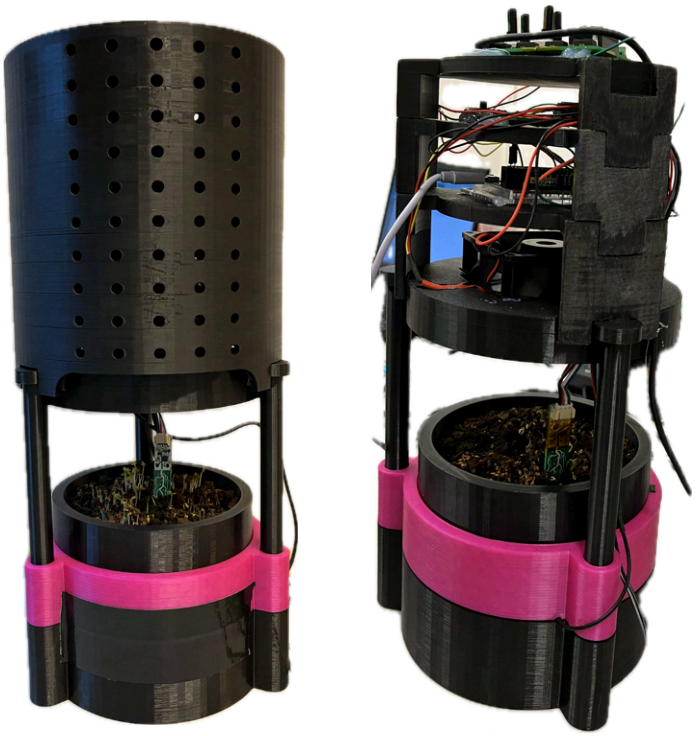


Figure 5: Two pictures of the completed SmartGrowSystem with cover (left), and a picture of the completed SmartGrowSystem, without the top cover (right).

Figure 6: A picture of the SmartGrowSystem with HLED PWM's brightness set to 50% from above (left) and a picture demonstrating the HLED PWM's at 50% brightness from underneath (right).

Video demonstration of Water Pump via app

The 3D-printed modular enclosure stood as a foundation for ensuring structural integrity, adjustability and safeguarding components within the Smart Grow System. Its cutting-edge

design enabled swift reconfiguration, allowing quick alterations, streamlined maintenance, and potential system modifications. Beyond structural support this enclosure effectively shielded delicate electronic components, allowed for proper cooling via adequate airflow and enhanced system stability and longevity.

The Arduino IOT cloud app

Simultaneously, the integration of a dedicated mobile application empowered personalized control and real-time monitoring of system components. This user-friendly app facilitated seamless adjustment of HLED intensity, fan speed, watering schedules and soil moisture levels. Real-time monitoring coupled with comprehensive data visualization including prior moisture levels, provided instant updates and insights, facilitating informed decision-making for optimized plant growth conditions.

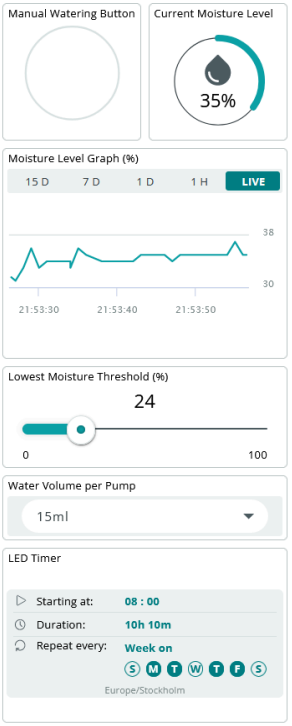


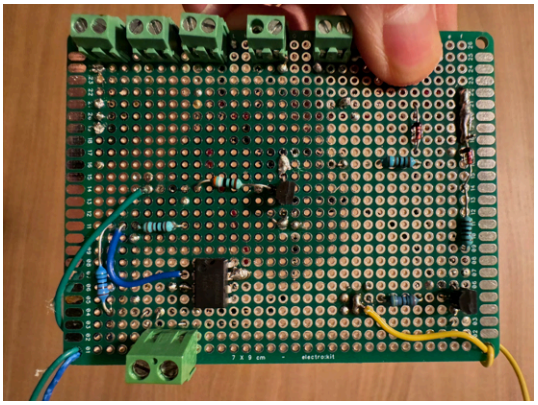
Figure 7: Showing the Arduino interface

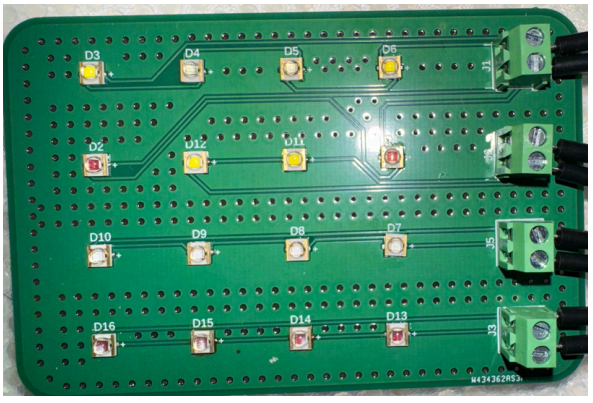
The mobile app seamlessly integrated with the Smart Grow System, utilizing Arduino IOT Cloud services, enhancing functionality, providing precise control, deeper system understanding, and an enriched user experience.

Furthermore, the sophisticated HLED lighting system, comprising 2 distinct PCBs, was pivotal to the system's functionality. The HLED array PCB housed specialized HLEDs tailored for horticultural needs, emitting precise spectrums optimal for plant growth stages, simulating natural sunlight conditions crucial for photosynthesis and growth.

PCB design and implementation:

The synchronized operation of these PCBs facilitated dynamic and adaptable lighting, providing the right intensity and wavelengths crucial for different growth stages. The HLED driver PCB efficiently regulates the power supply to the HLEDs, enabling precise adjustments in light intensity and cycles, aligning with various plant species' needs. The meticulously designed PCBs ensured efficient power utilization and increased heat dissipation by using a thermal optimization method requiring 2 separate copper layers that are connected by many vias. As such the design assures long-term durability, when combined with the fan system, enhancing the entire's system's reliability and sustainability.





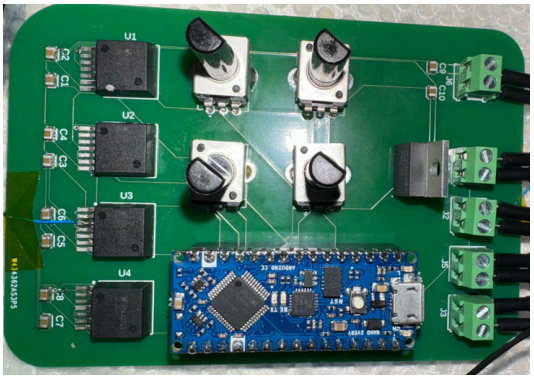


Figure 8: A photo of the breadboard circuit and the fundamental components used in both the cooling system and the self-watering system (top left), A picture showing the HLED PCB (top right), and a picture showing the LED driver board (bottom left).

Pump Measurements:

In order to measure the water flow of the submersible pump, a linear equation that corresponds to how much volume is being pumped after how long the pump has been activated needs to be made. All the data points specified from the testing were put in Excel spreadsheets in order to create a trendline of all the averages at specified time intervals. All the raw data and equation for the trendline can be found in Figure 21 below.

Delay (ms)	Test 1(ml)	Test 2(ml)	Test 3(ml)	Test 4(ml)	Average(ml)
500	4	3	3	3	3,25
1000	15	15	15	15	15
1500	25	26	27	26	26
2000	37	38	36	36	36,75
2500	50	51	53	50	51
3000	63	61	59	60	60,75
3500	73	72	76	72	73,25
4000	77	82	77	83	79,75
4500	100	96	97	92	96,25
5000	110	107	105	102	106

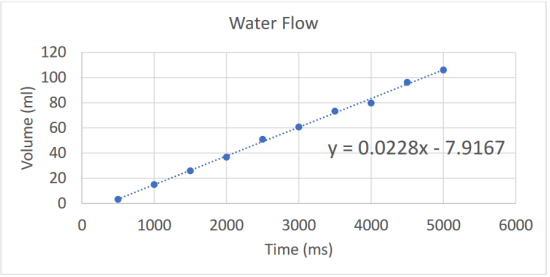


Figure 9: Table that contains all the test data from the Pump Water Output Measurements with corresponding trendline and expression for the line equation.

The generated linear equation is $V = 0.0228t - 7.9167$, where V is the volume of water and t is how long the pump has been activated. When V is negative it can be seen as when the silicone tube is being filled with water before it reaches the holes on the other end. For this project, however, it's more important to find how long the activation period needs to be for the pump to give a specified volume of water. So the function needs to be rearranged in regards to time.

$$V = 0.0228t - 7.9167 \Leftrightarrow t = (V + 7.9167)/0.0228 \Leftrightarrow t(ms) = 43.85 * V(ml) + 347$$

This function is simply used in the wait() function in order to dictate how many ms the pump stays activated. The value of V(ml) is dictated by a Cloud Variable that can be changed on the Dashboard to any desired water volume.

Pump and Fan Control Mechanisms:

Integral to the Smart Grow System's functionality were the controlled mechanisms for the fan and pump, each utilizing distinct control techniques.

The system featured a fan regulated via Pulse Width Modulation (PWM), controlled by a temperature sensor. This setup employed the temperature sensor to monitor environmental conditions, sending input to the Arduino. The Arduino, acting as the control unit, adjusted the fan's speed using PWM signals based on the temperature variations detected. This dynamic control allowed the fan to adapt its speed correspondingly, ensuring optimal cooling in response to changing environmental conditions within the growth environment.

Additionally, a Bipolar Junction Transistor (BJT) transistor switch regulated the water pump. The control mechanism involved connecting the transistor's base pin to a digital output pin on the Arduino, integrated with an appropriate resistor for proper voltage regulation. The BJT transistor, functioning as a switch, enabled precise and controlled watering of the plants within the system, allowing the Arduino to activate and deactivate the water pump as necessary.

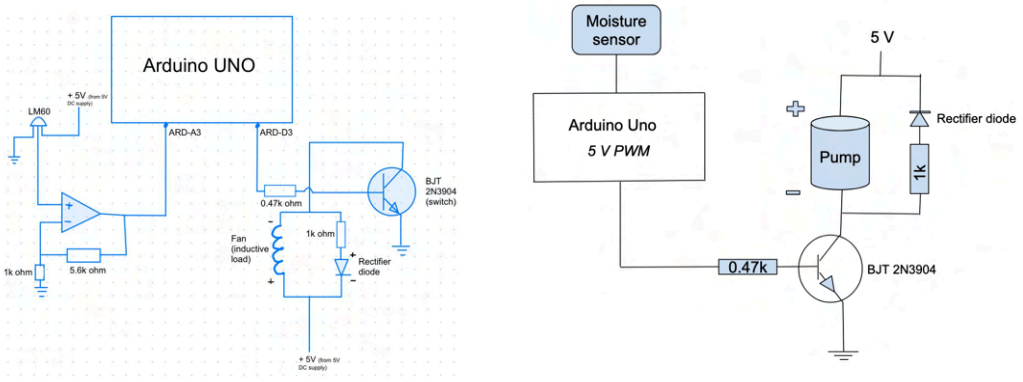


Figure 10: The circuit diagrams for the fan circuit (left), and the diagram for the pump circuit (right). The pump circuit has an identical setup to the fan circuit but without the analog temperature sensor part.

Analysis and discussion

The experiment aimed to integrate analog electronic and digital components into a system that allowed automated/manual control of watering and lighting. This was done by combining a pump and soil sensor system along with an HLED system that was cooled by controlling a DC fan's speed based on temperature.

The project faced a number of hardships along the way. These were mainly caused by supply issues, stemming from the desired components being out of stock as well as long lead times. The design of the housing did meet requirements from the start, however due to expansions having to be made, for example, having to add another top layer to house the various circuit boards, some of the parts had to be re-printed. Due to the slow nature of 3D printing technology, this led to the final assembly process being delayed. It was also realized that the circuits for the cooling system, watering system and HLED driver could have been

housed on a single PCB. The system would also have been simplified by using a single Arduino instead of two. In combination, these would consequently minimize the overall footprint of the system.

Other components were deemed to not meet requirements after testing, causing new parts to be ordered and further delays. This included a 12V fan, which did not receive enough current to spin at a fast enough speed to satisfactorily cool the HLED board. Other ad-hoc component changes included the voltage regulators on the HLED driver board, which did not have a quite low current output and a low max voltage, so it could not supply the potentiometers nor Arduinos with 5V. This was later solved by using a separate 5V power supply. In the future a DC to DC convertor should be used with the right voltage and current requirements.

After assembly, it also became apparent that more components than just the HLED board would benefit from cooling, such as the MagI³C-LDHM LED Step Down High Current Module that reached a peak temperature of 85 degrees celsius. A solution to this could be to expand the cooling system to add a second fan at the very top and modify the top cover to allow for airflow.

The 3D design revealed potential areas for improvement. Enhancements were identified, including providing more space between the fan and the printed material for improved airflow. Additionally, it became evident that enlarging the radius of the top cover would enhance mechanical movement. To address these issues, adjustments were made by grinding the outer sides of the lids and the inner part of the cover to achieve functional dimensions. This process, although time and energy-consuming, successfully resolved the identified issues.

The Smart Grow System featured sophisticated HLED systems facilitated by two distinct PCBs. One PCB hosted specialized HLEDs tailored for indoor plant growth, while the other served as the HLED driver, enabling the simulation of natural sunlight conditions. Additionally, a user-friendly mobile app allowed remote system control, enabling adjustments to HLED intensity, manual or automated watering schedules and soil moisture levels for optimized plant growth. The app proved to be a useful and user friendly way to interact with the system that allowed the user to have a fully customized experience to potentially use manual control in cases where the automatic controls could not properly control the plant environment parameters. Ideally, this would work from however far away the user is from the SmartGrowSystems. Due to the limitations of the WIFI connection however, this is range limited, if not applying a mes net or intranet VPN solution between the device and wifi signal the Arduino is connected to.

Advanced control mechanisms were integrated for the fan and pump regulation, utilizing PWM for fan speed regulation based on temperature and a BJT transistor switch for pump control. Although the fan objective was not met, due to challenges in precisely aligning temperature and fan speed. Despite achieving the desired fan speed regulation via a PWM signal from the Arduino, challenges emerged due to code inaccuracies, hardware issues, linear equation problems, and the absence of a calibrated temperature scale. Consequently, the fan functioned strictly as either on or off during the demo. Testing the temperature control part of the fan control circuit did however yield voltage and current readings in line with what

would be expected of a functioning OP - AMP circuit, lending to the fact that the main hindrance was most likely due to an erroneous arduino program.

With more time it is likely that this issue can be remedied and that the fan and pumps can be programmed to automatically adjust using the arduino according to values received by the various sensors. Or perhaps, rather than using a temperature sensor, another solution could be to set the PWM speed manually in the App. That way the speed could be adjusted if the fan is too loud.

Conclusions and recommendations

This experiment demonstrates practical applications despite its fundamental nature. There are numerous scenarios where environmental triggers activate and control systems independently, mimicking real-world applications. The versatility of such systems extends to countless other applications, emphasizing the importance of understanding fundamental principles to grasp complex systems. To improve this experiment, enhancing the app's functionality stands as a key goal. The updated app should enable users to adjust each row of HLEDs individually via PWM signals, eliminating the need for manual/physical potentiometer adjustments. Otherwise, instead of using a linear voltage regulator, which gets quite warm due to energy losses, a DC to DC converter should be used as the fan itself would overload the voltage regulator.

Additionally, allowing fan speed adjustments via the app using PWM signals for precise control would enhance user convenience. Furthermore, refining the accuracy of 3D printing is crucial. Ensuring the correct scaling of 3D parts is essential for seamless integration within the system, as well as being more generous with the space between 3D printed moving parts. Additionally, parts that needed to be water tight should have been printed with higher quality and more filling to avoid tiny leaks. Finally, avoiding mismatched labels on the PCB design is vital to prevent manual errors during the soldering process, optimizing efficiency during assembly and minimizing unnecessary work.

SmartGrowSystems is a concept that can be expanded upon with additional automatic and manual control. One can imagine many features that could be of interest, such as dispensing nutrients to the soil, using a camera to monitor the plant health in real time or adding thresholds to warn the user when certain parameters fall below or above a desired limit, as a safeguard if automatic controls seize. The app could also be updated to allow for presets of different plants, such as basil, cress or chili. Evidently though, this would require an immense amount of testing to determine the proper values and parameters. Overall, SmartGrowSystems meets the requirements set in the brief, and seamlessly allows for all encompassing manual control of the plant environment with many automatic controls already being implemented.

References

[1] P. Chatakul et al., "Interior plants: Trends, species, and their benefits," *Building and Environment*, vol. 222, pp. 109325, Aug. 15, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0360132322005583.
[2] UpSkill Learning - "Arduino: Programming Arduino - Beginners Guide To Get Started With Internet Of Things." Arduino Programming Book, Arduino Programming for IOT Projects, Arduino Guide for Engineers, Arduino Board Series, 2016. [Book]
[3] Anders Hagnestål (with Magnus Berg), “Elektroteknikens grunder III (1TE669): Elektromagnetism för elektroingenjörer–föreläsningsanteckningar”, 2017 revision by Magnus Berg. [Compendium]
[4] M. Rizani Rusli, et al., "Pulse Width Modulation (PWM) and Pulse Amplitude Modulation (PAM) Technique for
Medium-Speed BLDCM in Electric Vehicle Application," 2018 International Seminar on Application for Technology of
Information and Communication, 2018, pp. 87-92, doi: 10.1109/ISEMANTIC.2018.8549816. [Book]
[5] Molin. Bengt, “Analog elektronik”, 2022, third version. [Book]
[6] R. Keim, “Basic amplifier configurations: The non-inverting amplifier - video tutorial,” All About Circuits, 23-Aug-2020. [Online]. Available: https://www.allaboutcircuits.com/video-tutorials/basic-amplifier-configurations-non-inverting-amplifier/.
[7] “LM60 data sheet”, Texas Instruments, May 2004, revised August 2017. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm60.pdf?ts=1697701348032&ref_url=https%253A%252F%252Fwww.google.com%252F
[8] ESDU, "Radial, Mixed and Axial Flow Pumps," Engineering Sciences Data Unit, TM, Issue: 2007-01, Issued: October 1980, with Amendments A and B in June 2003. Endorsed by The Institution of Mechanical Engineers and The Institution of Chemical Engineers. IHS GROUP Company. Observe Copyright. For current status, contact ESDU.
[8] Pelonis, Sam, “ The difference between AC Fans & DC Fans”, Pelonis Technologies, 29-May-2019. [online]. Available: https://www.pelonistechnologies.com/blog/advantages-and-disadvantages-of-ac-fans-and-dc-fans
[9] Arduino. "Arduino IoT Cloud." Arduino Documentation. Available: https://docs.arduino.cc/arduino-cloud.

Appendix

- App demonstration and suggested App improvements

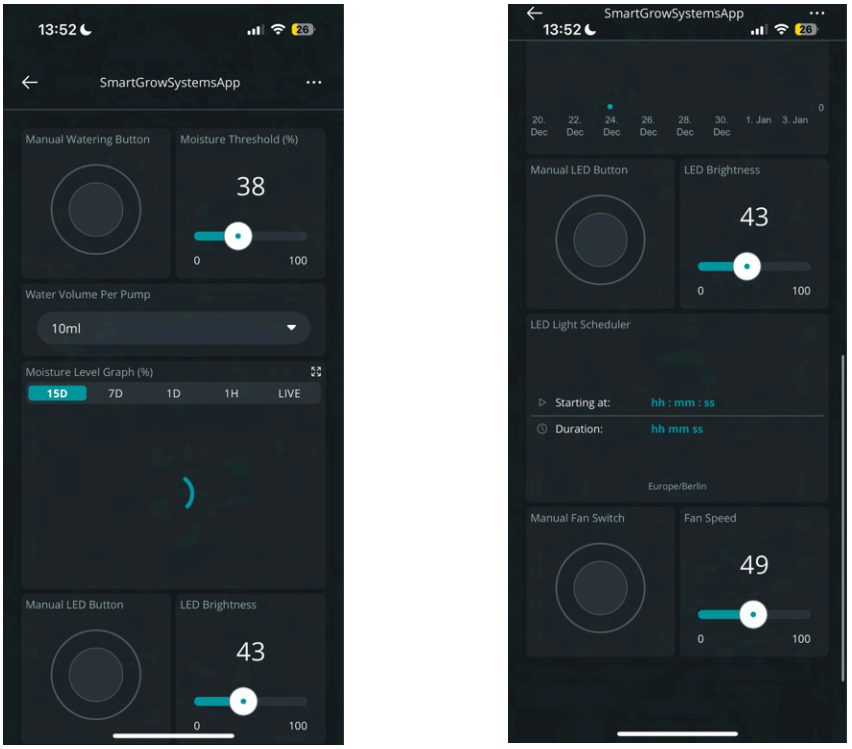


Figure 11: Screenshots of the improved version of the app running via iOT Remote on iOS

- Technical Implementation Overview

The project's technical implementation revolves around integrating diverse systems to develop an automated smart pot. The central focus requires a functioning HLED system, a soil moisture sensor, water pump, and lies in crafting a temperature-regulated system employing an Arduino microcontroller to manage a DC fan's speed. This begins with a temperature-sensitive sensor generating a voltage signal, which is then amplified by an operational amplifier (OP-Amp) before reaching the Arduino's analog inputs. Fusion 360 software contributes significantly by aiding in the precise design of 3D-printed enclosure parts and creating layouts for the HLED driver and HLED PCBs. This meticulous planning ensures optimal functionality within the HLED PCBs, seamlessly merging mechanical and electronic aspects. The Arduino's programmed algorithm interprets analog signals, converting them into digital points and adjusting the fan's speed based on temperature measurements. Specific thresholds are set in the code to correspond to ambient temperature and variances between reference and actual readings. Concurrently, the project manages the Horticulture LEDs through PWM signals from an Arduino Nano, enabling dynamic control over light cycles that simulate sunlight conditions for optimal plant growth. The whole system is then automated with variables which can be controlled through a smartphone interface created with Arduino IOT, that communicates with the system via wifi.

This integrated setup, involving 3D Design, Cooling System, Arduino Uno R4, Self-Watering System, and Horticulture LED Grow Lights, culminates in a comprehensive automated plant care solution.

Step-by-Step Instructions for the 3D system

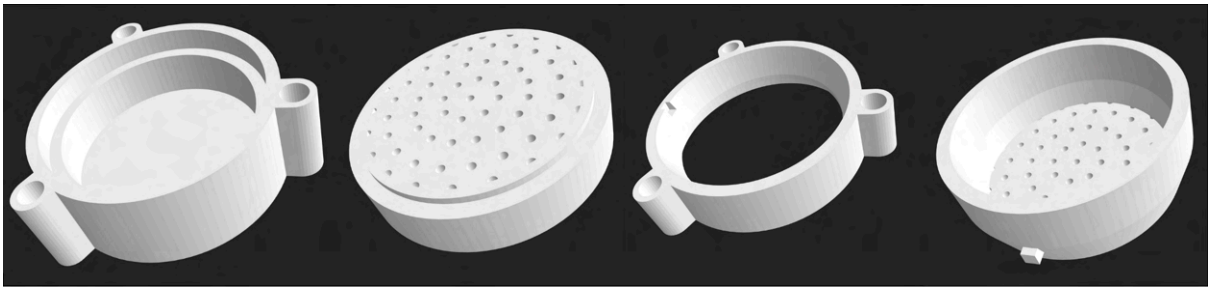


Figure 12: Photos of the 3D-designed components used the lower half of the smart pot that contains the pump and water and growth medium.

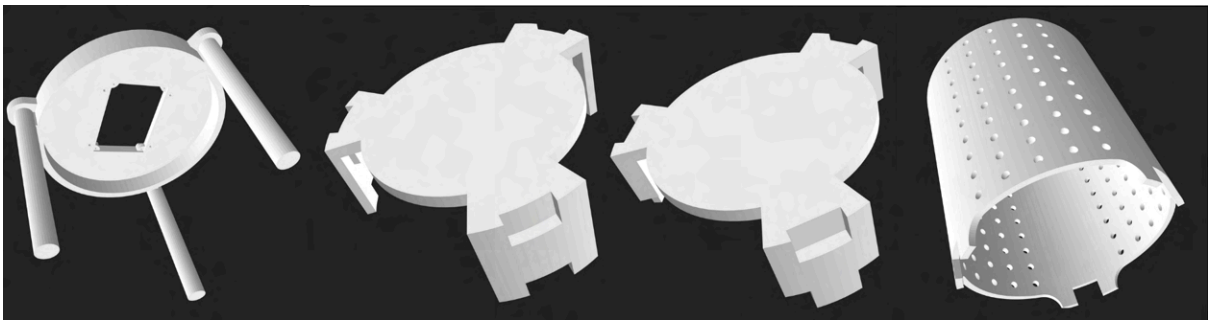


Figure 13: Photos of the 3D-designed components used on the upper half of the smart pot that contains the majority of the electronics.

Designing Modular Pot and Protective Housing in Autodesk Fusion 360:

1. **Open Autodesk Fusion 360:** Launch the software and start a new project for designing the modular pot.
2. **Create the Pot Base:**
    - Start with a cylindrical shape as the base of the pot. Adjust the dimensions based on the desired size and scale for the plant.
    - Add features for water drainage at the bottom of the pot to prevent waterlogging.
3. **Design Modular Pieces:**
    - Create modular components that can move up and down. This can be achieved by incorporating sliders, rails, or threaded connections to allow adjustability.
    - Ensure the modularity allows quick changes and easy scaling based on plant growth.
4. **Integrate Protective Housing:**
    - Design compartments or enclosures within the pot to accommodate the circuit boards. Create slots, holders, or compartments that securely hold the boards in place.
    - Ensure adequate protection from moisture, dust, or accidental damage.
5. **Incorporate Spaces for Components:**

- Allocate specific areas within the pot for accommodating water reservoirs, LEDs for plant growth, and provisions for airflow.
- Design channels or compartments for water storage and circulation, ensuring proper drainage and ventilation for airflow.

6. **Finalize Design and Validate:**
   - Refine the design, ensuring all components fit together seamlessly and function as intended.
   - Validate the design by simulating movements of the modular parts and checking for the functionality of protective housing for circuit boards.

Printing with Creality Ender:

1. **Prepare 3D Printing File:**
   - Export the finalized design as an STL or compatible file format suitable for 3D printing.
2. **Slice the Model:**
   - Use slicing software compatible with Creality Ender to prepare the 3D model for printing. Adjust settings for scaling, layer height, infill density and supports as needed.
3. **Load Filament and Start Printing:**
   - Load the desired filament into the Creality Ender 3D printer.
   - Begin the printing process, ensuring the printer is calibrated and settings are optimized for quality and precision.
4. **Monitor and Post-Processing:**
   - Monitor the printing process to ensure proper adhesion and layering.
   - After printing is complete, remove supports and perform any necessary post-processing like sanding or trimming to refine the printed parts.
5. **Assemble and Test:**
   - Once all components are printed, assemble the modular pot and check the functionality of moving parts, circuit board placement, water reservoir, LED placement, and airflow provisions.

Schematic, project board and Step-by-Step Instructions for the cooling system

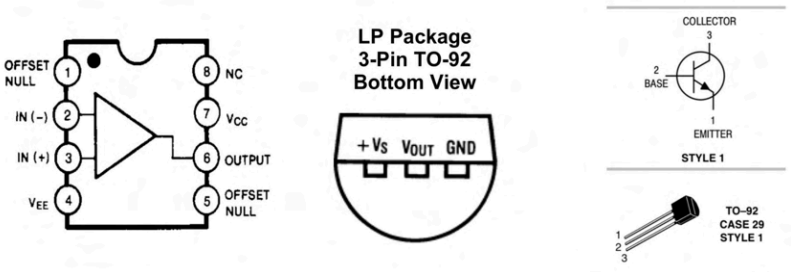


Figure 14: A photo of the pin configurations for the LM741cn, LM60 and 2N3094 components (in order as shown).

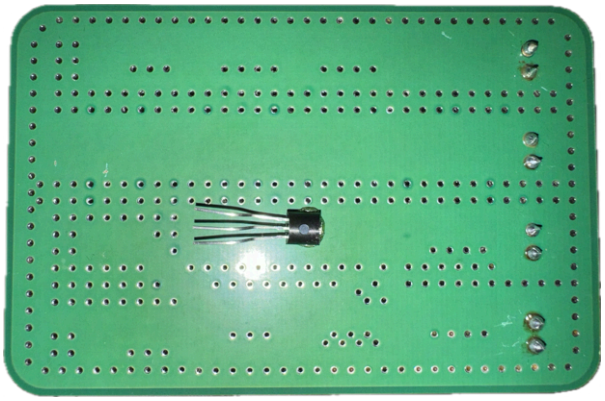


Figure 15: A photo of the pin configurations for the LM741cn, LM60 and 2N3094 components (in order as shown).

Circuit Board Assembly Steps:

1. **Glue LM60 Sensor:**
    - Attach the LM60 temperature sensor to the back of the circuit board as depicted in Figure 6.
2. **Connect LM60 Temperature Sensor:**
    - Far-right pin → Ground (GND).
    - Far-left pin → +5V power supply.
    - Center pin → Pin 3 of the LM741cn (OP-Amp).
3. **Connect Power Sources to LM741cn (OP-Amp):**
    - +5V DC voltage → Pin 7 of LM741cn.
    - -5V DC voltage → Pin 4 of LM741cn.
4. **Connect Resistors to the OP-Amp:**
    - Connect the left side of a 5.6k resistor to pin 2 of LM741cn.
    - Connect the right side of the same 5.6k resistor to pin 6 of LM741cn.
    - Connect the left side of a 1k resistor to pin 2 of LM741cn.
    - Connect the right side of the same 1k resistor to ground.
5. **Connect BJT-Transistor:**
    - Base (pin 2) → Digital PWM pin 3 on the Arduino.
    - Emitter (pin 1) → Ground (GND).
    - Collector (pin 3) → DC fan and positive side of a rectifier diode.
    - Connect the negative side of the rectifier diode to the other side of the DC fan, completing the circuit.
6. **Connect 5V Connector:**
    - Red positive wire → - Cathode of the rectifier diode.
    - Black negative wire → Ground (GND).
7. **Connect Arduino to a PC:**
    - Use a USB cable to power the Arduino and upload the programmed code onto the Arduino (Refer to the Appendix for the code).

Temperature Calibration and Linear Equation:

1. **Create a Linear Equation for the Temperature Sensor:**

- Use a calibrated temperature reader to measure and document the voltage levels corresponding to three distinct temperatures:
  - Room temperature (Note the voltage).
  - Slightly above room temperature (Note the voltage).
  - Much warmer than room temperature (Note the voltage).
- Create a linear equation using the LM60 sensor's voltage outputs and the respective temperatures measured.

Step-by-Step Instructions for the Arduino Uno R4

Creating Full Custom Control with Arduino IOT Cloud:

1. **Setup Arduino IOT Cloud:**
   - Register and set up an account on the Arduino IOT Cloud platform.
   - Create a new project and configure devices for interaction with the Arduino Uno.
2. **Create Cloud Variables and a Dashboard in Arduino IOT**
   - Cloud Variables is a feature in which you can add READ/WRITE variables to your Arduino code that automatically gets updated via wifi.
   - The Dashboard is a design tool in which you can create an interactive Desktop or Smartphone interface in which you can monitor or change Cloud Variables.
3. **Implement Manual Watering Button:**
   - Create a boolean Cloud Variable that dictates when the manual watering should initiate.
   - Make a push button on the Dashboard that turns the Cloud Variable to “True”.
   - Implement the Cloud Variable in the code, and whenever its boolean expression is “True”, initiate the manual watering process once.
4. **Monitor Moisture Levels with Real-Time Updates:**
   - Convert the signal from the moisture sensor into a more accessible/readable value. Make the sensor's highest capped value 100% and the 0% reading is taken when the sensor is in dry air.
   - Implement a Cloud Variable that returns the average of ten separate readings from the moisture sensors. This is done in order to prevent interference and noise. This Moisture(%) reading is used in all other automations.
   - Implement a percentage gauge on the Dashboard that returns the Moisture(%) reading in real time.
   - Create a plot chart on the Dashboard that can show the Moisture(%) reading up to 15 days into the past. Great way to evaluate the status of the plant.
5. **Threshold for Automatic Water Pump Activation**
   - Create a 0-100% slider on the Dashboard that WRITES the value of a new Cloud Variable which acts as an adjustable Threshold for when that plants get automatically watered.
   - Implement code that compares the current moisture level with the Threshold once every hour. If it’s below the set value, the automated watering process will be initiated. Doing the comparison once an hour adds leeway in case issues arise.

- Create a dropdown menu on the Dashboard that determines the volume of water which should be sent during each pump.

6. **Implement LED Timer Scheduler and Manual ON/OFF switch:**
   - In order to control the LED-lights via WiFi, the Arduino Uno needs to be able to communicate with the Arduino Nano. This is done through Serial communication where the Uno TX port (transmitter) is connected to the Nano RX port (receiver). Sending signals works by simply using Serial.Write() and Serial.Read().
   - Create a scheduler on the Dashboard that can adjust when the lights are on. This can be done daily, weekly or monthly. Send this boolean state to Arduino Nano via previously mentioned Serial communication.
   - Implement a button to the Dashboard and Arduino code that manually activates the boolean state to “True”. Send this signal via Serial bus to turn off or on the LEDs.
7. **Create Fan Control:**
   - Implement a slider that corresponds to the PWM signal being sent from the Arduino to the cooling fans mounted on the LEDs.
8. **Communication Between Arduino Nano and Cloud:**
   - Ensure the Arduino Nano is connected to the Arduino IOT Cloud
   - Establish bidirectional communication for data exchange and system control between the Arduino Nano and cloud platform.
9. **Testing and Validation:**
   - Test each functionality thoroughly to ensure proper operation and responsiveness to commands from the Arduino IOT Cloud interface.
   - Validate data accuracy, real-time updates, scheduling functionalities, and system variable changes via WiFi.

Schematic, Breadboard and Step-by-Step Instructions for the pump system

Figure 16: A photo of the Soil Moisture Sensor Capacitive I2C and the pump used (not to scale).

1. **Circuit Setup:**
   - Connect the Soil Moisture Sensor Capacitive I2C to the Arduino following the sensor's datasheet or pinout diagram.

    - The sensor has power (VCC), ground (GND), and data (SDA, SCL) pins, which should be connected to the corresponding Arduino pins (5V, GND, SDA, SCL).
2. **Design the Transistor Switch Circuit:**
    - Connect the transistor's base pin to a digital output pin on the Arduino with an appropriate transistor in between the base and arduino.
    - Connect the emitter pin of the transistor to the ground (GND) of the Arduino.
    - Connect the collector pin of the transistor to one terminal of the pump.
    - Connect the other terminal of the pump to the positive supply voltage (VCC).
3. **Implement Circuit Protection:**
    - Place a rectifier diode in parallel with the pump, ensuring the cathode (marked end) connects to the positive supply and the anode to the pump terminal.
    - Use appropriate resistors in the circuit for current limiting or voltage division, considering the transistor's requirements and the pump's specifications.
4. **Coding Arduino for Moisture-Based Pump Control:**
    - Write Arduino code to read data from the Soil Moisture Sensor Capacitive I2C.
    - Set up a condition in the code to determine the moisture level threshold at which the pump should activate or deactivate.
    - Use the digital output pin connected to the transistor as a control signal to activate or deactivate the pump based on the moisture level readings.
5. **Creating a Linear Equation for Pump Water Output Measurement:**
    - **Setup:** Connect the water pump to the Arduino to measure water output accurately over time. Submerge the water pump in a water container and connect the silicone tube. Put the other end of the silicone tube in an empty container on top of a weight scale.
    - **Data Collection:** Activate the water pump at specified time intervals and measure the volume of water with the scale. Four readings were taken at each interval.
    - **Record Results:** Note time intervals set on Arduino and corresponding water quantities accurately.
    - **Plot Data Points:** Plot collected data on a graph (time vs. water output).
    - **Linear Regression:** Use Microsoft Excel to find the trend line representing the relationship between time and water output.
    - **Equation:** Formulate a linear equation ($y = mx + c$) from the regression analysis.
    - **Usage:** Implement this equation in Arduino code to control pump operation based on desired water quantity.
    - **Implementation:** Calculate pump activation duration using the equation for precise water delivery.
6. **Assembly and Testing:**
    - Connect the circuit components as per the designed schematic and upload the Arduino code.
    - Power the Arduino and the pump circuit.
    - Test the system by adjusting soil moisture levels around the sensor to verify if the pump activates and deactivates as intended based on the programmed moisture threshold.

- Measure the volume of water and verify if they correspond to previous measured calculations.

7. **Safety Precautions:**
   - Ensure correct wiring and polarity to prevent damage to the components.
   - Avoid exceeding the specified voltage and current limits of the pump to prevent overheating or damage.

Schematic, PCBs and Step-by-Step Instructions for the Horticulture LED grow lights system

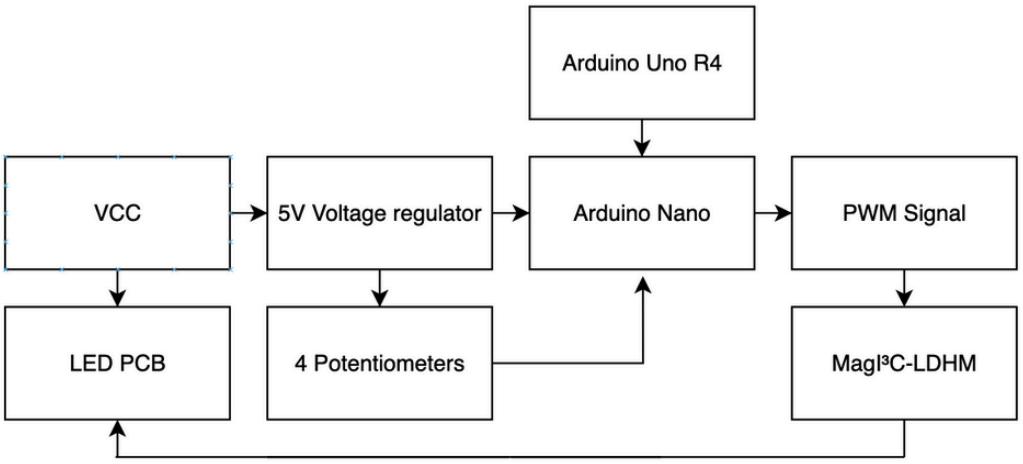


Figure 3: A flowchart of the how the HLED grow light system is connected along with the other individual subcomponents

PCB 1 - HLED Array:

1. **Design in Fusion 360:** Create a PCB layout with 16 HLEDs arranged in a series configuration. Ensure the PCB dimensions accommodate the HLEDs' specifications: 2.2W power, 3.2V forward voltage, 350 mA current.
2. **Routing and Connection:** Ensure proper trace routing to connect the HLEDs in series, considering appropriate spacing and trace widths for current handling.
3. **Testing and Simulation:** Simulate the design in Fusion 360 for functionality and potential issues. Verify the PCB layout's compatibility with the HLED specifications.
4. **Assembly and Testing:**
   a. Connect the circuit components as per the designed schematic.
   b. Power circuits individually with no more than 10V-12V and 1A.
   c. Test the system by adjusting voltage and amperage to see if the HLEDs dim with adjustments.
5. **Safety Precautions:**
   a. Ensure correct wiring and polarity to prevent damage to the components.
   b. Avoid exceeding the specified voltage and current limits of the HLEDs to prevent overheating or damage.
   c. Do not overuse without a fan.

PCB 2 - Power Regulation and Control:

1. **Design in Fusion 360:** Create a second PCB layout for power regulation and Arduino Nano control. Design it to receive a VCC signal and convert it to a stable 5V output using a linear regulator (LM7805CT).
2. **Arduino Nano Integration:** Include components and pin connectors to interface the regulated 5V signal to an Arduino Nano. Consider the pin connections between the Arduino Nano and Uno R4.
3. **Potentiometer Integration:** Include four potentiometers (PTV09A) that receive the 5V signal. Connect these potentiometers to Arduino Nano pins A1-A4 for analog signal processing.
4. **PWM Signal Generation:** Develop code for the Arduino Nano to convert analog signals from the potentiometers to PWM signals suitable for HLED control. Consider the pin connections that support PWM signals only.
5. **Magic Module Integration:** Interface the output from the Arduino Nano to the MagI³C-LDHM LED Step Down High Current Modules connected to the HLED PCB. Ensure correct voltage levels for HLED operation.
6. **Additional Components:** Incorporate terminal blocks (Series 2141) for convenient wiring and decoupling capacitors (2.2µF) to ensure stability in the voltage regulation circuit.
7. **Power Supply Consideration:** Design the PCB to be compatible with a 48V power supply, drawing 1.46A for efficient HLED operation.
8. **Connect HLED PCB to the HLED Driver PCB:** Connect a separately custom-designed HLED PCB to the HLED Driver PCB, ensuring compatibility and connection points with the HLED PCB and other components.
9. **Validation and Assembly:** Validate the PCB designs, ensuring proper connections, voltage regulation, and Arduino control functionality. Proceed with assembly once validated. Connect the circuit components as per the designed schematic and upload the Arduino code. Power the Arduino and connect your voltage source.

- **Arduino Nano Everyday Code used in Arduino IDE**

```
// Define analog sensor pins
const int sensorPin1 = A1;  // Pin A1 connected to Sensor 1
const int sensorPin2 = A2;  // Pin A2 connected to Sensor 2
const int sensorPin3 = A3;  // Pin A3 connected to Sensor 3
const int sensorPin4 = A4;  // Pin A4 connected to Sensor 4

// Builtin LED
const int indicatorLedPin = LED_BUILTIN;  // Onboard LED pin


void setup() {
 Serial1.begin(115200); // Initialize serial1 at 115200 baud rate with a 1s delay to get is started
 delay(1000);

 pinMode(3, OUTPUT);   // Pin 3 supports PWM, LED 1
 pinMode(5, OUTPUT);   // Pin 5 supports PWM, LED 2
 pinMode(6, OUTPUT);  // Pin 6 supports PWM, LED 3
 pinMode(10, OUTPUT);  // Pin 10 supports PWM, LED 4

 pinMode(indicatorLedPin, OUTPUT); // Set indicator LED pin as OUTPUT
}

int led_status = 0;

void loop() {
 if (Serial1.available() > 0) { // Check if data is available to read from serial
   char command = Serial1.read();
    if (command == '1'){
     led_status = 1;
     digitalWrite(indicatorLedPin, HIGH); // Turn on the LED indicator
    }

    else if (command == '0'){
     led_status = 0;
     digitalWrite(indicatorLedPin, LOW); // Turn off the LED indicator
    }
```

```
    }

    if (led_status == 1){

    // Read analog sensor values
    int sensorValue1 = analogRead(sensorPin1);  // Read value from Sensor 1
    int sensorValue2 = analogRead(sensorPin2);  // Read value from Sensor 2
    int sensorValue3 = analogRead(sensorPin3);  // Read value from Sensor 3
    int sensorValue4 = analogRead(sensorPin4);  // Read value from Sensor 4

    // Map sensor values to PWM range and control LED brightness
    analogWrite(3, map(sensorValue1, 0, 1023, 0, 255));  // Sensor 1 -> pin 6 (LED 1)
    analogWrite(5, map(sensorValue2, 0, 1023, 0, 255));  // Sensor 2 -> pin 7 (LED 2)
    analogWrite(6, map(sensorValue3, 0, 1023, 0, 255)); // Sensor 3 -> pin 10 (LED 3)
    analogWrite(10, map(sensorValue4, 0, 1023, 0, 255)); // Sensor 4 -> pin 11 (LED 4)
    }

    else {
     digitalWrite(3, LOW); // Turn off light connected to pin 5
    digitalWrite(5, LOW); // Turn off light connected to pin 6
    digitalWrite(6, LOW); // Turn off light connected to pin 9
    digitalWrite(10, LOW); // Turn off light connected to pin 10
    }

    delay(10); // Small delay between sensor readings and LED updates
   }
```

Figure 17: Code written in Arduino IDE for the Arduino Uno R4 Wifi Code used in Arduino IDE

- **Arduino Uno R4 Wifi Code used in Arduino IDE**

```
#include "arduino_secrets.h"
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/58e2d3d4-3f02-476b-a03a-761da3c58c3e

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  float moisture;
  int cutoff;
  int flow;
  CloudSchedule timer;
  bool watering;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include "Adafruit_seesaw.h"
#include "RTC.h"

// Assigns the moisture sensor the name ss through the I2C ports
Adafruit_seesaw ss;

// Assign the pump on pin 6
int pump = 6;

// Assign analog readings for temp sensor
const int analogRealValuePin = A3;

// Assign fan control pin
const int fanControlPin = 3;
```

```
void setup() {

//Prevent the digital pin from having an output during bootup. This caused issues initially
 pinMode(pump, INPUT_PULLUP);
 // Initialize serial and wait for port to open:
 Serial.begin(115200);
// This delay gives the chance to wait for a Serial Monitor without blocking if none is found
 delay(1000);
// Initialize Serial1 which is needed to communicate through TX -> RX serial bus
 Serial1.begin(115200);
 delay(1000);

// Assign fan control pin
 pinMode(fanControlPin, OUTPUT);

// For testing, will make arduino LED turn on in order to test code
 pinMode(LED_BUILTIN, OUTPUT);

// Setups and starts the Moisture Sensor
 if (!ss.begin(0x36)) {
   Serial.println("ERROR! seesaw not found");
   while(1) delay(1);
 } else {
   Serial.print("seesaw started! version: ");
   Serial.println(ss.getVersion(), HEX);
 }
 // Start the internal Real Time Clock.
 RTC.begin();

// Used to setup the date/time. Hide this whenever the time doesn't need to be updated.
startTime(day,month,year,hour,minutes,seconds)
//  RTCTime startTime(10, Month::DECEMBER, 2023, 23, 18, 00, DayOfWeek::MONDAY,
SaveLight::SAVING_TIME_ACTIVE);
//  RTC.setTime(startTime);
  // Defined in thingProperties.h
```

```
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
   /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
  */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  //Make Moisture 100 on start so it won't water the plant on startup
  moisture = 100;
  // Assign Pump pin as output
  pinMode(pump, OUTPUT);
 }

 // Highest and lowest range for the moisture sensor
 float max_capread = 1015 ;
 float min_capread = 300;

 // Variables used to only make the program run once every hour, value will represent the last hour plant was watered. Initial value outsie of range to triffer on startup
 int last_hour_control = 100;

 // Variable that stores the previous change in LED lights. Used so it won't send Serial signals every loop to the arduino NANO
 char last_led_stage = '9';

 // "button" variable that activates code in the void loop after the IOT Cloud manual watering button has been pushed
 int button = 0;

 // Dictates whenever the fan is on or off
 int led_status = 0;
```

```
// Value that creates a moisture average
float moist_average = 0;

// This variable keeps track of how many samples are taken for the average. Will reset at 10.
float number_average = 0;

//Value that keeps track of the last second (initially outside range)
//int last_second = 61;

void loop() {
// Used to update changes from the cloud
 ArduinoCloud.update();
 // Function that adds the correct wait time for the pump to pump equal volume of water as the "flow" value from the wifi interface
 int flow_delay = 347 + (43.85 * flow);

// Manual watering that activates the pump for "flow_delay" set of ms after pressing the button in the app
 if (button >= 2) {
   digitalWrite(pump, HIGH);
   digitalWrite(LED_BUILTIN,HIGH);
   Serial.println("Pump has been manually activated");
   delay(flow_delay);
   digitalWrite(pump, LOW);
   digitalWrite(LED_BUILTIN,LOW);
   button = 0;
 }

// Take reading from the moisture sensor. Temperature won't be needed for this project
//  float tempC = ss.getTemp();
 uint16_t capread = ss.touchRead(0);

// currentTime takes the values from the RTC clock and updates each loop
 RTCTime currentTime;
 RTC.getTime(currentTime);

// Create a percentage value from the moisture sensor
```

```
 float capread_percentage = ((capread - min_capread) / (max_capread-min_capread)) * 100 ;

//Below is the code for the cooling fan for the LEDs
//Turns off the fans when the LED is off
 if (led_status == 0) {
   analogWrite(fanControlPin, 0);
 }

//Turns on the fan when the LED is on
 if (led_status == 1) {
// Values from sensor
   float realValue = analogRead(analogRealValuePin);

// Convert analog reading to voltages
   float realVoltage = (realValue / 1023.0) * 5.0;
 // Convert voltages to temperatures using 11.5x + 18.7
   float realValueTemp = (11.5 * realVoltage)+18.7;

// Alternative interpitation of the temperature
   float temperature = (160*realVoltage -657);

// Assign setpointValueTemp, limit for when fans will kick it up a notch
   float setpointValueTemp = 30;

// Sees if the temperature is bigger than "setpointValueTemp"
   float tempDifference = realValueTemp - (setpointValueTemp);
 // Below is the range in order to control the fan speed
   if (tempDifference <= 1) {
     analogWrite(fanControlPin, 255);
   }

   else if (tempDifference > 1) {
     analogWrite(fanControlPin, 255);
   }

// Can be used for monitoring the fan system, good for troubleshooting
//    Serial.println(analogRealValuePin);
```

```
//    Serial.println(realValue);
//    Serial.println(realVoltage);
//    Serial.println(realValueTemp);
//    Serial.print("Temperature:"); Serial.println(temperature);
//    Serial.println("_____________");
//    Serial.println(tempDifference);
 }

// Reset the sensor if it crashes, value is outside of the reading range. Sensor gets stuck on really high value when it crashes
 if (capread > 2000) {
   Serial.println("Moisture sensor has encountered an error and is restarting");
   ss.begin(0x36);
   delay(5000);
 }

/*
// Overwrite the cloud moisture value with 10 reading average from the sensor. 10 plot average over 10 seconds. Won't capture readings outside of the 100% range.
 if (capread_percentage <= 100 && currentTime.getSeconds() != last_second) {
   moist_average = moist_average + capread_percentage;
   last_second = currentTime.getSeconds();
   if (currentTime.getSeconds() % 10 ==0) {
     int average = moist_average/10;
     if (average <= 100) {
       moisture = average;
     moist_average = 0;
     }
   }
 }
*/
// Make a 10 plot average from the moisture sensor to initiate the moisture level
 if (capread_percentage <= 100) {
   moist_average = moist_average + capread_percentage;
   number_average = number_average +1;
   Serial.println(moist_average);
   Serial.println(number_average);
```

```
    if (number_average == 10){
      moisture = (moist_average / 10);
      number_average = 0;
      moist_average = 0;
    }
  }

  // Below is the automatic watering code. Has a timer that looks at the moisture value every hour and compares it to the "cutoff" value from the cloud.
  // Will excetute the loop once an hour, and update the "last_hour_control"
  if (currentTime.getHour() % 1 == 0 && last_hour_control != currentTime.getHour()) {
    last_hour_control = currentTime.getHour();
  // Sees if the moisture sensor has dipped below the assigned cutoff value from the cloud
    if (moisture < float(cutoff)) {

  // This is the code that will initiate the pump, which water quantity is dictated by the "flow_delay" cloud variable
      digitalWrite(pump, HIGH);
      Serial.println("Pump has been automatically activated");
      delay(flow_delay);
      digitalWrite(pump, LOW);
    }
  }



  // Whenever the timer is active, the LEDs will turn on and vice versa. Sends a serial signal to NANO through the TX->RX pins to update its state
  if (timer.isActive() == 1 && last_led_stage != '1') {
    Serial1.write('1');
    led_status = 1;
    Serial.println("LED automatically turned on from timer");
    digitalWrite(LED_BUILTIN,HIGH);
    last_led_stage = '1';
  }
  else if (timer.isActive() == 0 && last_led_stage != '0'){
    Serial1.write('0');
```

```
    led_status = 0;
    Serial.println("LED automatically turned off from timer");
    digitalWrite(LED_BUILTIN,LOW);
    last_led_stage = '0';
  }
  // Print values from the system in order for monetering
  //  Serial.print("Temperature: "); Serial.print(tempC); Serial.println("C");
  //  Serial.print("Capacitive: "); Serial.println(capread);
  //  Serial.print("Moisture(%): "); Serial.print(capread_percentage); Serial.println("%");
  //  Serial.print("Time: "); Serial.print(currentTime.getHour()); Serial.print(":");
  Serial.println(currentTime.getMinutes());
  // Delays the whole systen for 1 second
  delay(1000);
  }

 // Below are all the cloud variables that activates everytime there is a change from the IOT cloud

 void onWateringChange()  {
 // Manual watering will activate in the loop when "button" variable ha been released, other words button
 >= 2
  button = button + 1;
 }


 void onCutoffChange()  {
  // Add your code here to act upon Cutoff change
 }

 void onTimerChange()  {
 // Code that activates when there is a change to the timer
  if (timer.isActive()) {
    Serial1.write('1');
    led_status = 1;
    Serial.println("The LED has been turned on manually");
    digitalWrite(LED_BUILTIN,HIGH);

  }
```

```
    else {
      Serial1.write('0');
      led_status = 0;
      Serial.println("The LED has been turned off manually");
      digitalWrite(LED_BUILTIN,LOW);
    }

  }

  void onFlowChange()  {
  // Updates the "flow_delay" whenever changed
  //  int flow_delay = 347 + (43.85 * flow);
  }
```

Figure 18: Code written in Arduino IDE for the Arduino Uno R4 Wifi Code used in Arduino IDE