ECE 445

Spring 2023

SENIOR DESIGN LABORATORY

FINAL REPORT

"Don't Kill My Plant" Habit Tracker

Team 28

Ben Wei (btwei2), DK Ehiribe (ehiribe2), Zade Lobo (zlobo3)

TA: Selva Subramaniam

Professor: Viktor Gruev

Abstract

We are trying to solve a problem that has plagued people for ages: breaking bad habits and adopting good ones. Our solution involves keeping a person's habits in check with something they can be emotionally attached to. Over the course of the semester, we built "Don't Kill My Plant Habit Tracker", an electronic enclosure that holds a user's favorite plant. Paired with a companion app on the user's phone, they will use the interface to keep track of their habits. While they continue their habits, the plant will be watered and be allowed to stay alive. However when the user doesn't keep up with their good habits, the enclosure will slowly start to kill the plant through dehydration, suffocation, and blocking out light. Though we ran into multiple fabrication, coding, and electronic challenges throughout the semester, we created a project that we are proud of.

TABLE OF CONTENTS

INTRODUCTION **4**
- Problem 4
- Solution 4
- Visual Aid 5
- High-Level Requirements 5
- High-Level Justifications 5
- Block Diagram 6
- Subsystem Overview 6
  - Power Subsystem 6
  - Application Interface Subsystem 6
  - Plant Enclosure Subsystem 7
  - Microcontroller Subsystem 7
  - Irrigation Subsystem 7

DESIGN **8**
- 3D CAD Model 8
- Circuit Schematic 8
- PCB Design 9
- App & Web Server Design 10
- Fabrication and Integration 10

COSTS & SCHEDULE **11**
- Cost Analysis 11
- Project Schedule 12

DESIGN VERIFICATIONS & CONSIDERATIONS **14**

CONCLUSION **15**
- Accomplishments 15
- Uncertainties 15
- Ethical Considerations 15
- Future Work 16

REFERENCES **17**

APPENDIX A (Requirements and Verifications) **18**

INTRODUCTION

Problem

The problem for this project was one that has plagued people for ages: breaking bad habits and adopting good ones. Even though humans may want to change these habits, they usually lack the willpower to do so. Common solutions for this problem include smartphone tracking apps and physical devices that track physical habits. These solutions are great for tracking, but most of them can be circumvented easily and don't hold people accountable for their actions. In addition, any positive reinforcement methods that they use are minor and are not effective.

Behavioral therapy has been promising in the field of medicine recently, and it is usually used to address traits in a person that might be self-detrimental or harmful for a person to have. Being able to understand what a person values and adapting their thought processes to fit what needs to change can benefit a person in the long term. Bringing behavioral therapy to fixing habits has been tried in the past, but not to as great of an extent as needed.

Solution

This project solves the problem by adding a physical component to digital habits. “Don’t Kill My Plant” is a habit tracking solution that brings a plant into the equation. If a user keeps up with their habits, our solution will maintain their plant with regular watering. But, if the user fails to complete their habits, their plant will slowly die. In this case, the plant will be denied light, water, and air. Dark curtains, which block light, will be a visual reminder to maintain personal habits.

This solution taps into multiple concepts from psychology, which makes it an effective solution. The plant acts as both positive and negative reinforcement in its maintenance and slow killing. It is also a visual reminder, intended to sit on a deck or somewhere frequently seen. And finally symbolically, the plant represents personal growth; put work in, and both you and the plant will grow. These psychological principles make this an effective solution to the problem of personal habit building.

Visual Aid

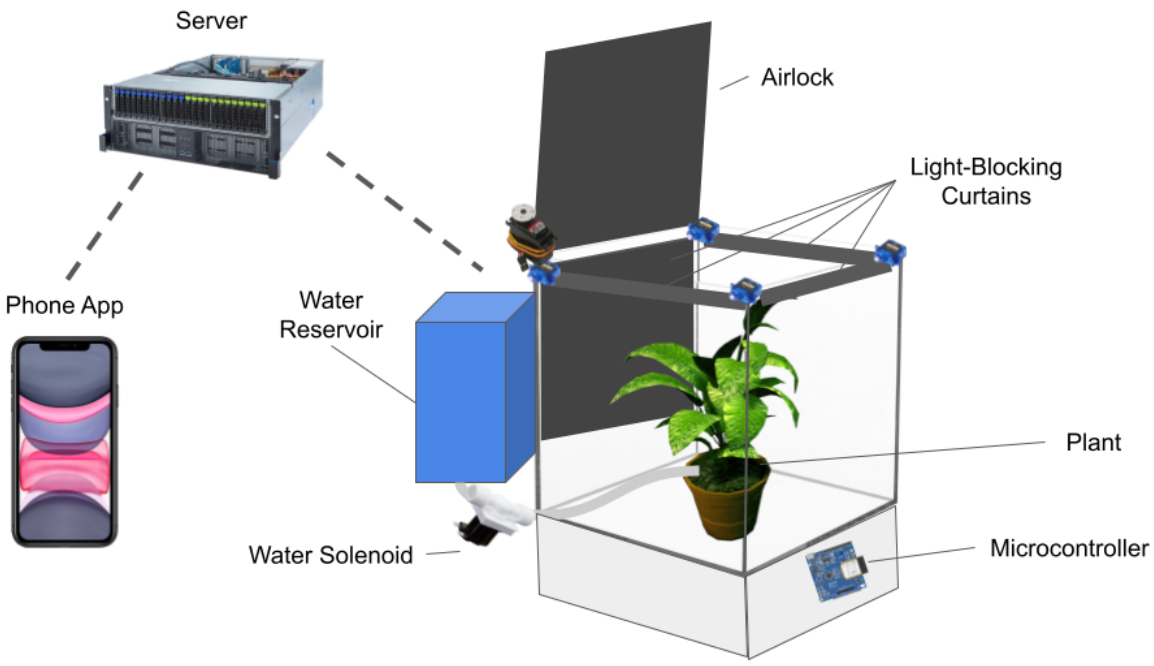

Figure 1. The visual depiction of our solution, showing the app, server, and enclosure.

High-Level Requirements

- The application interface is able to facilitate habit tracking for the user and send this information to the physical device
- The created device system is capable of keeping a potted plant alive and killing a potted plant according to the data passed to it.
- The design has a modular design, allowing for more plants-enclosure addons to track more habits.

High-Level Justifications

Since this problem is based in psychology, the solution is too. The application requirement calls for the user to habit track, which is an effective method of building habits. The enclosure, with the ability to provide for and kill the plant, creates a material consequence. And finally, modularity offers additional material consequences; more plants means greater obligation to keep up with habits. These three requirements effectively address the problem of habit tracking.

Block Diagram

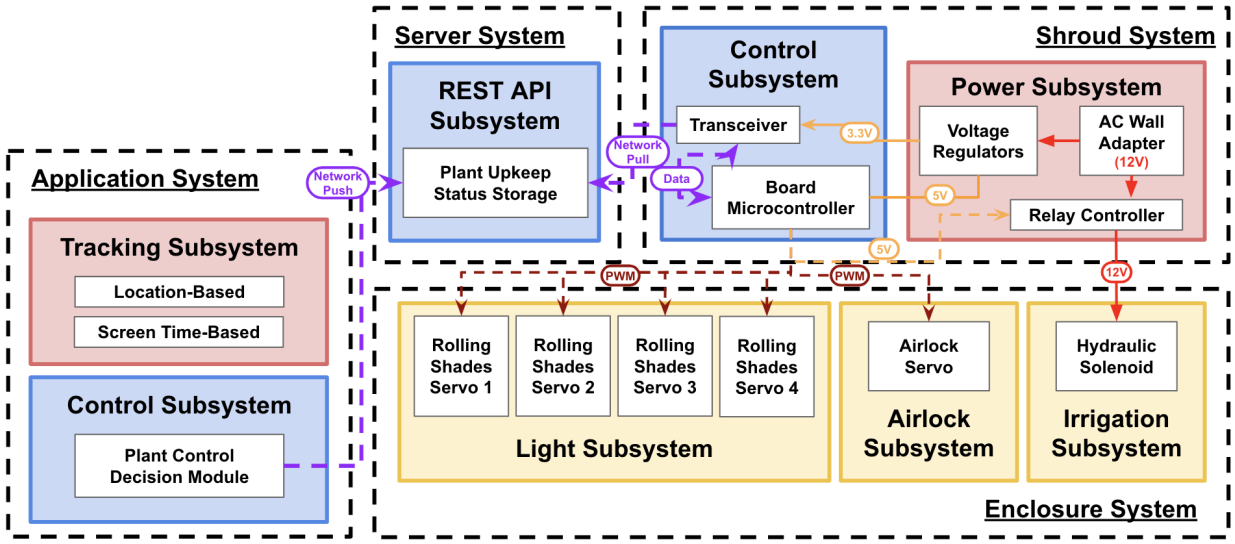

Figure 2. This is the block diagram of our project. This details the implementation of the app, server, and enclosure systems.

This top-level design is split into five subsystems: power, enclosure, microcontroller, irrigation, and application. The airlock and light systems are combined into one enclosure subsystem. This is because they rely on the same inputs and internal mechanisms.

Subsystem Overview

Power Subsystem

This module should interface between power from the wall input and each of the subsystems. This means converting the 12V input to both 5V and 3.3V for other systems.

Application Interface Subsystem

The application interface is a phone application that will offer multiple ways to track habit forming, including location, screentime, and message and call tracking. This allows the application to pick up on habits such as going to the gym, avoiding a coffee shop, spending too much time on social media, or messaging your family.

Plant Enclosure Subsystem

The plant enclosure is a box with an airtight lid in order to create an isolated environment for a plant. The box itself will contain transparent walls with a method for blocking light out (either electronic tint or rolling window shades. The box will have an airtight lid that can be electronically opened and closed by the microcontroller. This lid and walls are implemented with servos.

Microcontroller Subsystem

The microcontroller system will use an ATmega328-P to control the other subsystems. A ESP8266 chip will also be used to add wireless capabilities to the system. This system will regulate the functions of the other subsystems. For each enclosure, the microcontroller will fetch a binary signal from a server and manage systems accordingly. This means, for example, upon receiving a ‘0’ signal for enclosure 0, all subsystems should maintain the plant: regularly water, open curtains, and open airlock. The opposite is true for a signal of ‘1.’

Irrigation Subsystem

The irrigation subsystem will be controlled by the microcontroller in order to routinely water the plant through the included water solenoid. The reservoir outside the plant enclosure will allow the user to input water for irrigation, but the actual water delivery will be controlled through hydraulic tubing piping it inside the system.

DESIGN

3D CAD Model

In order to test how the subsystems would work together, we decided to put together a 3D CAD of the system together in Autodesk Inventor. In order to make everything exactly to the specification, we employed some neat tricks and tactics.

In order to work out how the curtains would deploy to block light, we deemed that the easiest way was to use rolling shades that wrapped around a shaft to roll up. Because of this, the shaft needed to be secured on two ends: to a motor and to a bearing to hold it up. The mounts for the motors are all custom designs so it would hold the motor, but it would also hold the shaft coming in from the other corner.

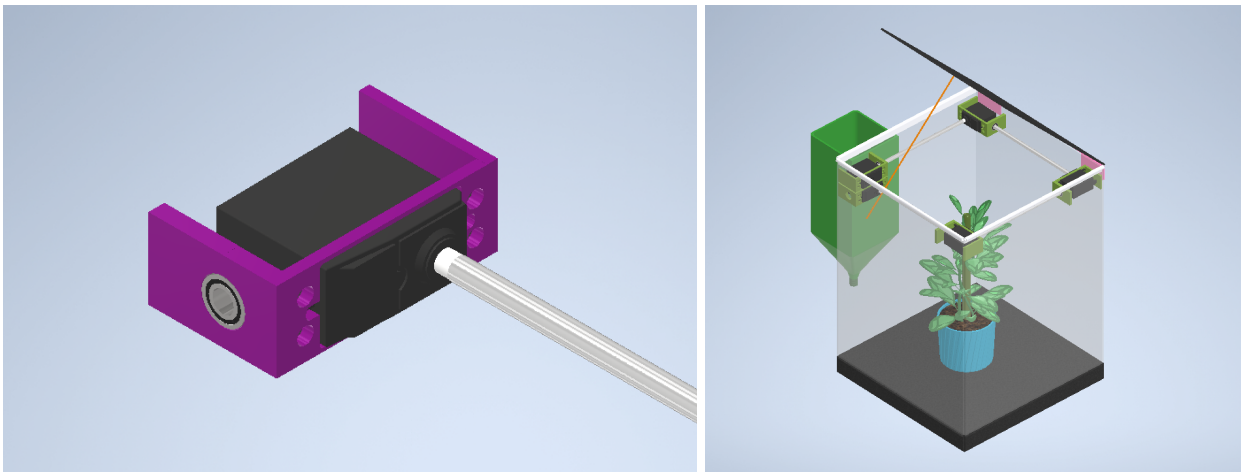

Figure 3. On the left, a CAD diagram is shown for a servo mount. On the right, a fully assembled enclosure is shown in CAD.

The model altogether uses five servo motors: four to drive the curtains and one to prop the lid for airflow. In addition, the water solenoid would be used to feed the plant water.

Circuit Schematic

When creating the circuit schematic, there are few considerations we kept in mind.

- When wiring the microcontroller, we wired it to only read and write digital signals. In addition, we used an external clock for it.
- The relay does not need a flyback diode because there is no MOSFET that can be damaged from the reverse current.
- The ESP8266 needs to be wired in reverse due to the orientation of the pins on the board.

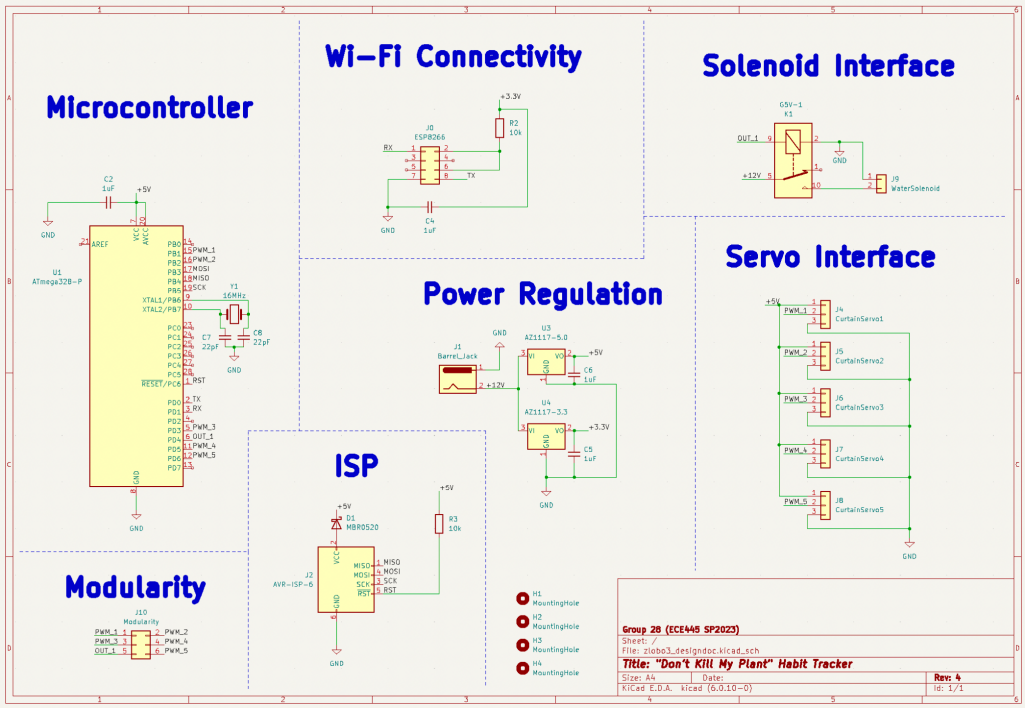

Figure 4. This diagram shows the final design for our circuit schematic.

PCB Design

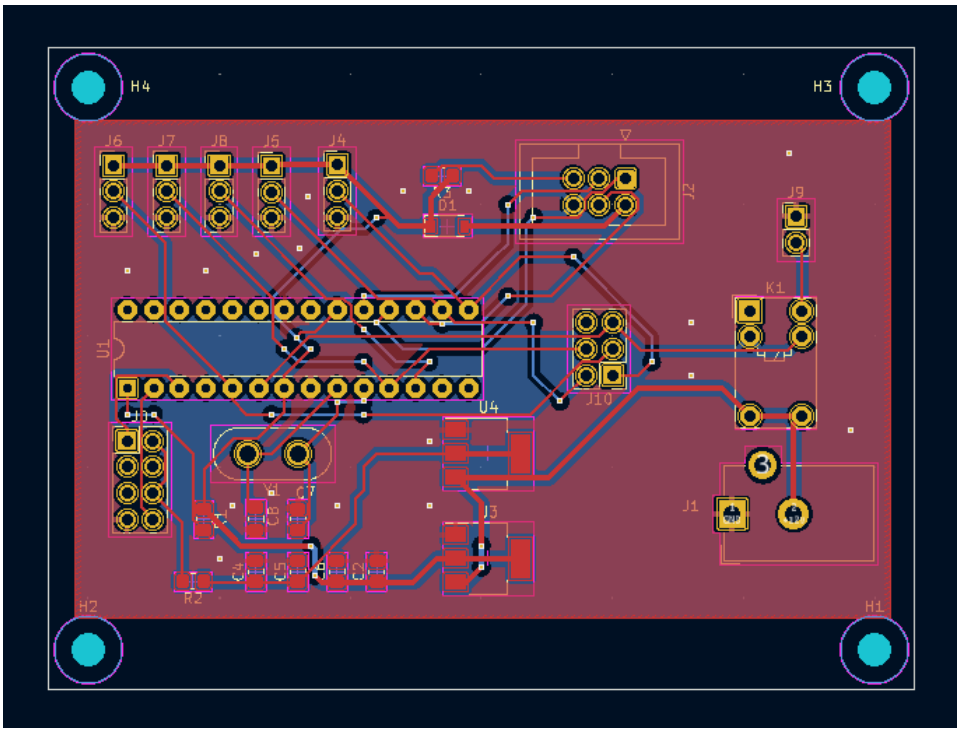

Figure 5. This diagram shows the footprint and routing for the final PCB.

App & Web Server Design

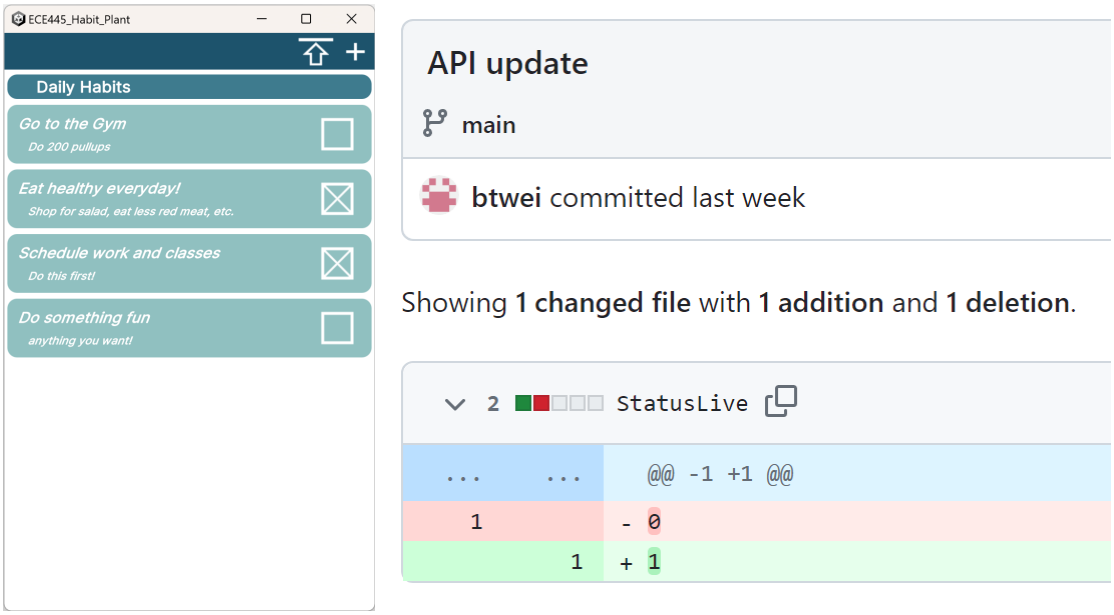

Figure 6. Final application on the left with four example habits. On the right is a screenshot of the Github page after an API update.

The application subsystem was created in Unity. This app was built in three sections: user interface (UI), base functionality, and server calls. UI involved creating custom scalable elements for the window, pop ups, and habit elements. Base functionality involved adding functions to buttons and providing backend habit management. Server calls involved pushing data to the server via http requests.

Instead of using a standalone web server, Github was used instead. This allowed for easy verification of the server state, which sped up our verification processes. A challenge with the Github API is that the 'location' of the server page changes after each update. So, before writing data, first the page location needs to be found (via a GET request), and then data is written (via a PUT request).

Fabrication and Integration

The fabrication went according to the 3D CAD document created. The enclosure was secured together with silicone epoxy to make it watertight and airtight. The 3D printed mounts were also attached via silicone epoxy, and holes were drilled and sealed for cable management and the irrigation subsystem.

COSTS & SCHEDULE

Cost Analysis

Itm	Qty Unit	Qty	Qty/Pck	Amt/Pck	Pcks Prchsd	Qty Rcvd	Amt
Ball Bearings	Number	4	10	$7.99	1	10	$7.99
Black Acrylic	Number	3	1	$12.60	3	3	$37.80
Vinyl Gasket	Feet	4	17	$6.93	1	17	$6.93
Steel Rod	Inches	42	48	$6.38	1	48	$6.38
Clear Polycarbonate	Number	4	1	$10.24	4	4	$40.96
Silicone Sealant	Number	2	2	$9.56	1	2	$9.56
Blackout Fabric	Yards	1	1	$7.99	1	1	$7.99
ATmega328-P	Chip	1	1	$3.25	1	1	$3.25
MBR0520	Component	1	1	$0.02	1	1	$0.02
Barrel Jack	Component	1	1	$1.09	1	1	$1.09
AZ1117-5.0	Component	1	1	$0.40	1	1	$0.40
AZ1117-3.3	Component	1	1	$0.40	1	1	$0.40
G5V-1	Component	1	1	$2.31	1	1	$2.31
ESP8266	Component	1	1	$7.50	1	1	$7.50
Pin Headers	Component	20	1	$0.35	20	20	$7.00
10K Ohm Resistors	Component	3	1	$0.10	3	3	$0.30

1 uF Capacitors	Component	5	1	$0.10	5	5	$0.50
0.1 uF Capacitors	Component	1	1	$0.36	1	1	$0.36
900-00008	Component	5	1	$18.95	5	5	$94.76
AC Power Adapter	Adapter	1	1	$10.88	1	1	$10.88
997 Water Solenoid	Component	1	1	$6.95	1	1	$6.95

In addition to the cost analysis above, we are also including a time-cost for employment of the people working on this project. We assume that 3 students are working on this project at a pay of $41 per hour. With a total of 40 hours working on the project and an overhead of 2.5 times the hourly rate, we arrive at a labor cost of **$12,300**.

Subtotal	$253.33
Expected Tax	$22.80
Expected Total	$276.13
Total Including Labor	$12,576.13

Project Schedule

Week	Task	Person
2/20-2/26	Circuit Schematic Design	Zade
	Team Contract	All
	Design Document	All
2/27-3/05	Finalize PCB Design	All

	Order PCB	
	Order COTS Parts	
3/06-3/12	Develop Mobile App	Ben
	Collect Parts	Dike
	Fabricate Enclosure	Zade
3/13-3/19	Spring Break	
3/20-3/26	Solder PCB	Dike
	Develop Mobile App	Ben
	Fabricate Enclosure	Zade
3/27-4/02	Solder PCB	Dike
	Fabricate Enclosure	Zade
	Finish Mobile App	Ben
4/03-4/09	Finalize PCB	Dike
	Wiring of Enclosure	Zade
	Write Code for Board	Ben
4/10-4/16	Integration and Debugging	All
4/17-4/23	Mock Demonstration	All
	Presentation Rough Draft	
4/24-4/30	Demonstration	All
	Presentation Final Draft	
5/01-5/04	Final Presentation	All
	Final Paper	

DESIGN VERIFICATIONS & CONSIDERATIONS

The largest design constraint was implementing the fluid solenoid. Solenoids inherently require larger voltage sources due to them containing strong inductors for operation. The best solenoid for fluids that we could find operated at 12 volts, which was much larger than the 3.3 volts that we had initially decided on with operation of an ESP32.

An ESP32 chip operates at a maximum of 3.3 volts, causing there to be multiple problems for our use case. The first major problem occurs with the operation of servo motors, which require three connections: 5 volts, a PWM signal, and ground. Our initial thought was to include another regulator for components that needed 5 volts. However, this doesn't fix our second problem, which was the operation of the solenoid. The inductor required a 12 volt digital signal for ideal operation, which the ESP32 could definitely not provide. The alternative solution was to include a transistor with a diode to get 12 volts directly from the power supply, but operating a solenoid with a transistor is not an ideal solution due to the high current and effects of the transistor. We decided to use a relay to operate the solenoid, but most hobbyist relays only work with 5 volt signals and above.

This prompted our switch to using an ATmega328-P chipset, commonly found on boards like the Arduino Uno and Arduino Nano. The chipset can be powered with as low as 1.8 volts, but can use up to 5 volts for its digital logic. With the use of this microcontroller, we downgraded from the ESP32 to the ESP8266 for wireless communication because we only needed the transceiver. This not only allows us to power the servo motors much more easily, but it also allows the inclusion of a relay for use with the solenoid.

CONCLUSION

Accomplishments

By the end of the project, we were able to successfully meet most of our high level requirements that we assigned to it. Our final enclosure had everything that we had hoped for, except with a debug button instead of signals being sent over Wi-Fi. Our application interface was also very clean, allowing for users to make habits, keep track of them, and upload them to our server.

One of our largest accomplishments was debugging electronics. Our group had little to no knowledge of how to debug electronic components since many of our ECE classes were taken online through the COVID pandemic. Through this class, we were able to get a better grasp on how to use tools like the multimeter and the oscilloscope to debug our project and understand what was going wrong. In addition, we learned more about interfacing with hardware and software through electronics. This knowledge was crucial, and we learned a lot throughout the build process.

Uncertainties

In the end, our project remained not entirely finished due to issues with our ESP8266 breakout chip. We spent a lot of time debugging the chip, and we attribute our problems to be improper communication between the microcontroller and the chip. The cause of this could be many reasons such as incorrect firmware on the chip, signals being too different in voltage, or even incorrect PCB setup. While we didn't have time to completely debug the issue, we were at least able to get the rest of the project to work despite these setbacks.

Ethical Considerations

A potential safety concern that may arise in this project is the exposure of certain parts of our system to water delivered through the irrigation subsystem. We were able to address this issue by ensuring that the plant enclosure system is properly sealed and can be properly drained to avoid leakage into other components. We planned to be mindful of the IEEE Code of Ethics 7.8.I.5 as we received continuous feedback through the development of this project and we used this feedback to improve our project and continue producing honest data. Relating to the IEEE Code of Ethics 7.8.I.1, we also planned to be mindful of the safety of the public and the environment by ensuring that nothing other than a plant is placed into the airtight plant enclosure system because this system can cause harm to the object it encloses.

Future Work

In the future, if work continues on the project, we would like to continue debugging the issue integrating the ESP8266 breakout chip to pull down the data. We would also like to secure the PCB better to the bottom of the system. This will allow for less accessible cables and fewer dangling wires.

Past what we have stated as the goal of this project, looking into sending data back to the application interface from the enclosure would prove useful to monitor the state of the plant and the levels in the reservoir from the user's app. In addition, we would like to monitor the plan in more ways than just feeding it water. We can use sensors to monitor light, soil pH, soil moisture, and air quality.

REFERENCES

"IEEE code of ethics," Institute of Electrical and Electronics Engineers (IEEE), Jun-2020. [Online]. https://www.ieee.org/about/corporate/governance/p7-8.html. [visited on 09-Feb-2023].

"Atmega328 Pinout," *Atmega328 pinout*. [Online]. https://www.learningaboutelectronics.com/Articles/Atmega328-pinout.php. [visited on 21-Feb-2023].

B. Tymrak, "Programming an AVR ATmega328P with an Arduino," *Programming an AVR atmega328p with an Arduino*. [Online]. https://www.brennantymrak.com/articles/programming-avr-with-arduino. [visited on 22-Feb-2023].

"Transistors, relays, and controlling high-current loads," *ITP Physical Computing*. [Online]. https://itp.nyu.edu/physcomp/lessons/electronics/transistors-relays-and-controlling-high-current-loads/. [visited on 20-Feb-2023].

APPENDIX A (Requirements and Verifications)

Power Subsystem Requirements	**Verification**	**Verification Status (Y/N)**
-Must convert 12VDC to 5VDC	-Check with a multimeter that 12VDC is received from the wall adapter -Check with a multimeter that 5VDC +/-20% is being outputted -Check with a multimeter that at least 0.8mA is being outputted	Y
-Must convert 12VDC to 3VDC	-Check with a multimeter that 12VDC is received from the wall adapter -Check with a multimeter that 3.3VDC +/-10% is being outputted -Check with a multimeter that at least 0.8mA is being outputted	Y
App Interface Subsystem Requirements	**Verification**	**Verification Status (Y/N)**
-The application includes basic user configurable habit tracking that leverages phone features such as location info	-Check that the user can input a habit to be tracked -Observe that there is a clear display of whether habits are kept or not	Y
-The application should be able to post binary signals that correspond to each plant enclosure: 0 for habit kept, 1 for a habit broken	-Use a test program to force signals of all 0 and then all 1. Verify that the server signals are updated accordingly. -Use the actual program to test that the habit tracking translates into updated server signals.	Y
Plant Enclosure Subsystem Requirements	**Verification**	
-The shades can fully roll and unroll through the use of servos.	-First, check that the curtains are mechanically able to be deployed (no snags or other issues)	Y

	-Next, verify that 5V+/-20% are being received as power -Next, verify that the control signals are correct through the use of an oscilloscope. This involves using a test program to generate component specified PWM signals. -Finally, verify that the system responds appropriately to the test signals. Observe that the servos both fully roll and unroll the shades.	
-The airtight lid can fully close and open through the use of a servo	-First, check that the airlock is mechanically able to function -Next, verify that 5V+/-20% are being received as power -Next verify the correctness of the control signal. Use an oscilloscope with a test program, and check that the component specified PWM signals are being generated. -Finally, observe that the airtight lid can be fully closed and opened through the use of these control signals	Y
Microcontroller Subsystem Requirements	**Verification**	**Verification Status (Y/N)**
-Binary signals are successfully fetched from the server	-Use the application to toggle each enclosure's binary signals -Observe that the curtain signals activate accordingly with an oscilloscope (which verifies that wireless signals are received).	N
-Servo subsystems are managed through control signals, opening blinds or closing them and opening the airlock or closing it based on the corresponding binary signal	-First, use a test program to internally force a signal transition from 0 (don't kill) to 1 (kill) and vice versa (say, transition every given interval of time), this should not depend on testing the wireless capabilities. -Use an oscilloscope to verify the servo specific PWM signals. Check that the width is	Y

	correct for 0->1 transitions and 1->0 transitions.	
-Irrigation subsystems are managed through control signals, watering regularly or withholding water based on the corresponding binary signal	-Test both binary enclosure signals individually via a test program (which forces a binary signal) For a signal of 1 (kill): -Use a multimeter to check that the irrigation control output is never raised to logical 1 For a signal of 0 (don't kill): -Use a multimeter to check that the irrigation control output is regularly raised to logical 1 for a set amount of time *For example, every hour, the signal is raised for 30 seconds (or a shorter interval for testing)	Y
Irrigation Subsystem Requirements	**Verification**	**Verification Status (Y/N)**
-This subsystem must water the plant when it receives a logical 1 signal and not otherwise	-First, check that the 12V input is functioning with a multimeter -Next, verify the input signals are correct using a multimeter and a test program. The logical high signal should be 5V +/- 10%. The logical low signal should be 0.5V +/- 100%. -Finally, verify that 12V are passed to the solenoid only upon receiving a logical high signal. Again, use a multimeter to verify that 12V are being passed through.	Y