Plant Irrigation and Monitoring System

ECE 445 Design Document - Fall 2023

Team #08
Kevin Le, John Burns, Carlos Toledo
Professor: Olga Mironenko
TA: Sainath Barbhai

Table of Contents

1 Introduction

1.1 Problem:

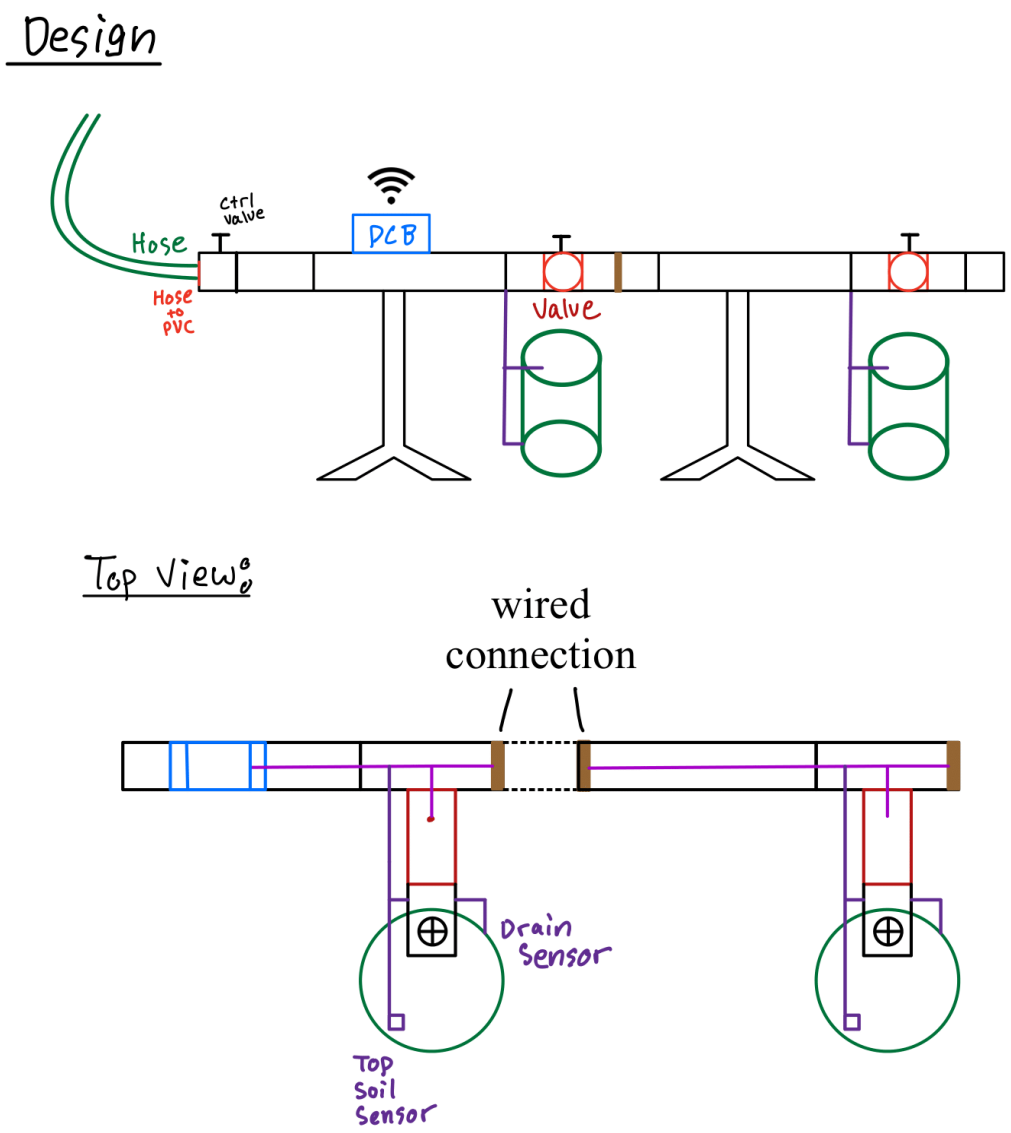
Gardening is a skill that takes a lot of intensive care and effort as each individual plant has its respective living condition it must meet. These living conditions such as required sunlight, minimum amount of water, and climate vary from plant to plant and it can be very difficult to be attentive to all these details in keeping your plants in the best possible condition as we are occupied with our busy lives or simply lack the skill. Watering outdoor plants can be very tedious and a task often forgotten.

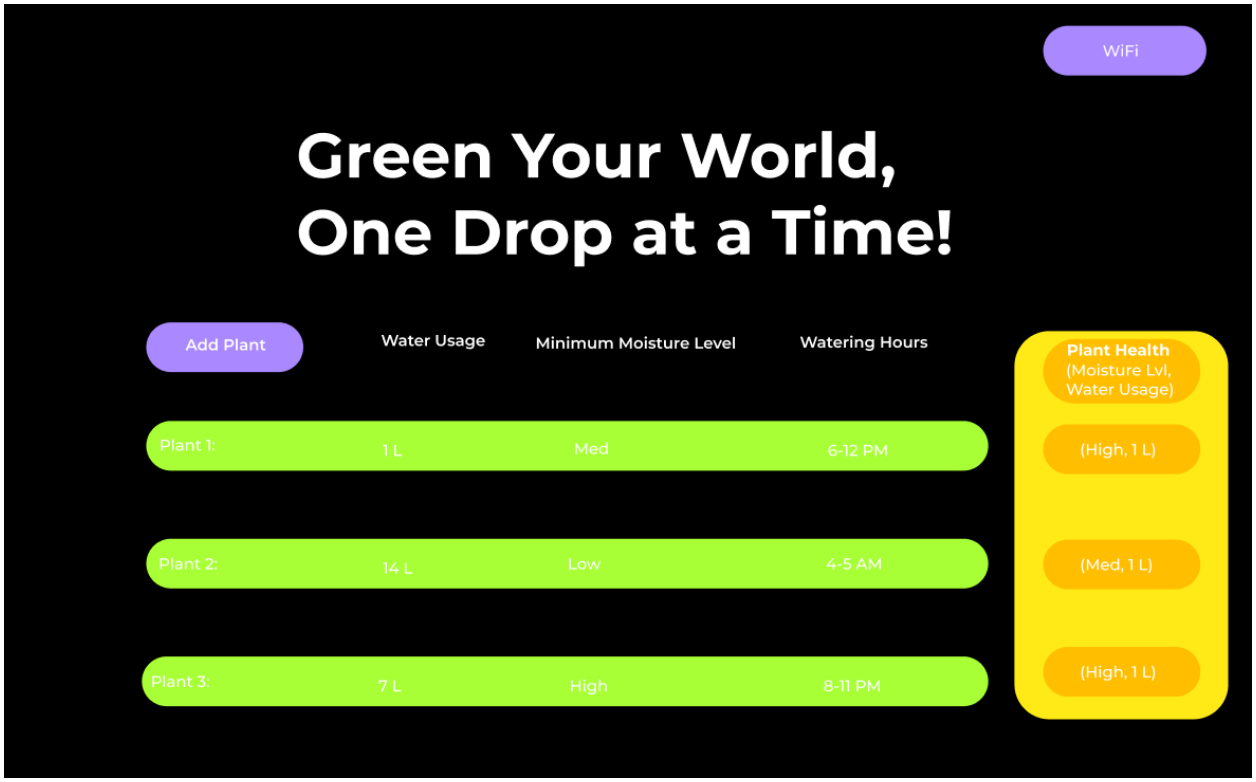
1.2 Solution:

Our solution to this is to micro-manage the watering aspect of home gardening, taking input from soil sensors to form a smart irrigation system. This system will help the user monitor a single plant or more. In terms of current competitors on the market, other similar products are limited to the number of plants that can be monitored and require a water pump. This system will be modular and can be linked together to build a larger system. Other systems only measure the moisture content within the first couple inches of the surface and do not connect directly to a water hose. Our solution will water the plant till the whole pot is moist and fully watered. Using Solenoid valves in connection to a garden hose for irrigation, a single plant can be configured to have a minimal moisture level, providing the most desirable conditions for your plant, or a connect system can be created via daisy chaining.

There is a similar project from Spring 2023 but there are significant differences. The Project i and referring to is the "Don't Kill My Plant" Habit Tracker. Their project converts phone habits to watering / environmental changes. Our project aims to care for the plant in an outdoor setting and allows for multiple plants to be taken care of. Another similar project is DIY Plantify from Spring 2023. This project moves plants away from light if it is too intense and tests moisture levels based on weight. Again, very different from our water irrigation system.

1.3 Visual Aid





1.4 High Level Requirements

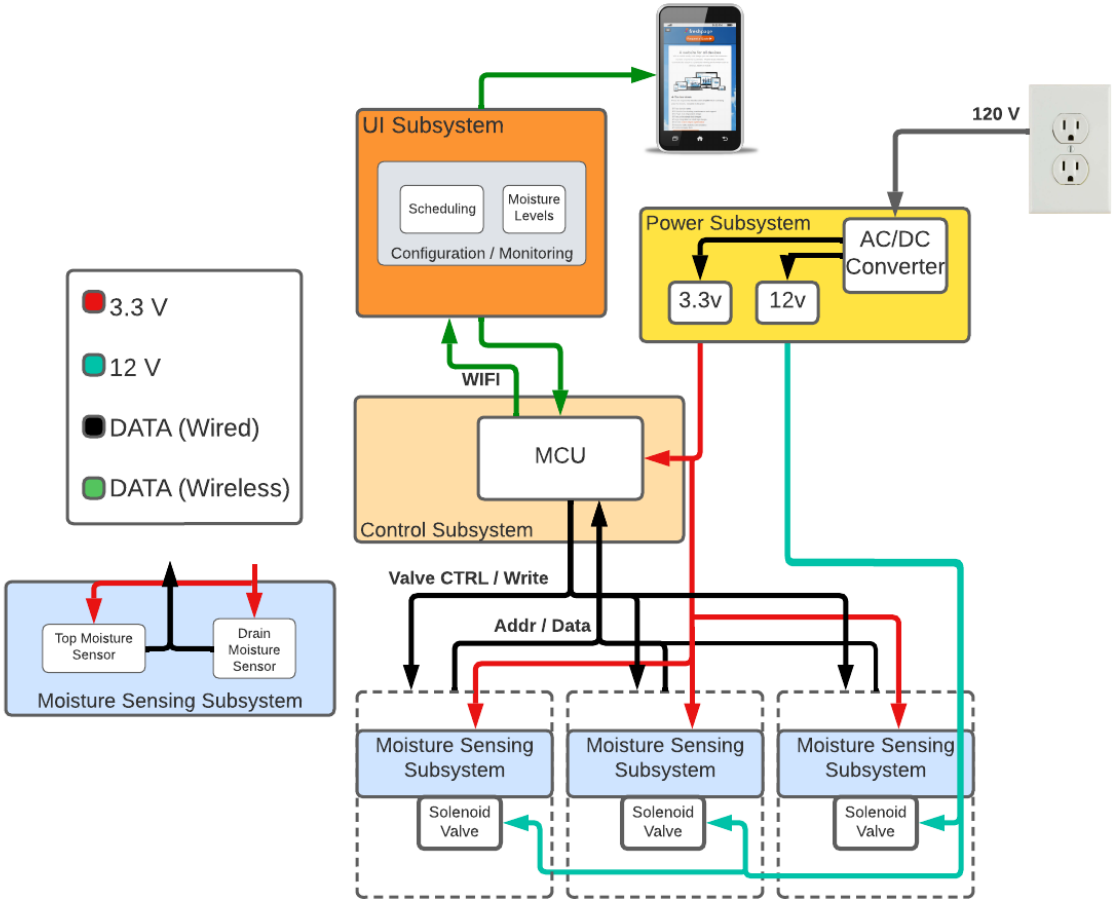
1. Requirement 1: The system controls moisture levels in individual plants, performing standard watering on schedule until drainage occurs, and continually monitors and waters plants if moisture deviates by more than ± 10% from user-defined values.

2. Requirement 2:The system starts watering within ± 15 minutes of the user-defined time, will only water within permissible windows to ± 15 minute accuracy, and stays within ± 5% of the user-defined daily water usage limit.

3. Requirement 3: The system can control multiple plants at once via master-slave communication; each slave operates independently and is not influenced by the state of other slaves.

2 Design

2.1 Block Diagram



2.2 Functional Overview

2.2.1 UI Subsystem:

This subsystem is the user interface through which parameters can be set for operation of the system. It will communicate via bluetooth or WiFi with the Control subsystem and will relay input information from the user about permissible water usage, permissible watering hours, and the minimum moisture levels for each individual plant. Additionally, the subsystem must be able to receive information from the control subsystem about the moisture level of each plant and the water consumption information. The subsystem will also be able to save each plant's configuration information to save effort on repeated plants. To build the Web-App, we will use a Javascript library called React.js, as well as react-bootstrap and CSS for the styling of the web page for a smoother user experience. We will use React Hooks such as useState and useEffect to store the necessary parameters inputted by the user and to dynamically render new information after refresh. To save the config data, we will use GraphQL to generate our own data type as each plant has its four unique parameters and with Express.js, we will use that as our backend server to connect to our frontend client and communicate with HTTP requests. Specifically, the HTTP Post request will allow us to send the user input data to the backend to be stored in a JSON file so that we can reuse this data later. To communicate with the MCU via bluetooth/wifi, we will first need to connect the ESP32-S3-WROOM to the PCB and make sure all the power and ground connections are proper. Then, we need to use the built-in WiFi.h library which allows us to use the ESP32 as a station. This means the ESP32 can behave as a client/server and handle requests to other devices connected to the network. We are going to use HTTP endpoints to process these requests and a library called ESPAsyncWebServer for server functionality. For client functionality, we are going to use the HTTPClient library for HTTP requests to the web-app. By doing this, we can have a connection with our user interface web-app and send requests over between the two for communication. If the WiFi.h library does not work, another option is to use the espressif board package, which should generate the same result. We can test this simply by outputting signals from our mobile device to turn on and off LEDs on a breadboard. To test this for our system, we can enter the parameters and see if our slave system waters the plant at the specific times inputted. Problems with wifi/bluetooth connection is secure communication where data needs to be protected from tampering. The solution to this is to Initially have the ESP32 act as an Access Point where it behaves like a WiFi network, like a router, and we can connect our laptop to it. This mode allows us to set our own encrypted SSID and password directly on the ESP32 instead of locally on someone's laptop, which can easily be breached. We then switch the ESP32 to Station mode which will allow it to send and receive data to and from the user interface. We store the WiFi credentials onto our ESP32 because the chip already has an encrypted EEPROM on it, which provides non-volatile data memory that is highly secure and can easily encrypt sensitive data.

Requirements	Verification

• Can be launched successfully locally on a Web-App with intended UI-Subsystem and working backend.
  ○ UI-Subsystem loads up and user can click on buttons
  ○ Once user submits parameters, the data will be saved
	• A webpage with the intended arguments and buttons to add plants and configuration settings should pop up, ensuring frontend works properly
	• To ensure backend functionality, users should click on configured settings. If the setting loads in properly, then backend functionality works. If nothing happens, the backend server is not running properly

• Can communicate via bluetooth/WiFi with the control subsystem. ○ Confirm that the LED on the ESP32 lights up	• A webpage with the intended arguments and buttons to add plants and configuration settings should pop up, ensuring frontend works properly • User clicks on “Test WiFi” button
• Can take in user parameters and communicate them to the control unit. ○ Exact same parameters show up as a data type in graphql server when query button is pressed	• A webpage with the intended arguments and buttons to add plants and configuration settings should pop up, ensuring frontend works properly • To ensure saved configuration settings functionality, user clicks on “add new plant” button then fills in the four parameters: plant name, water usage, watering hours, and the minimum moisture level • User clicks “save” button • To ensure proper functionality, the solenoid should allow water to flow to the specific plant only at the specific watering hours. Confirm that this is true as no water should be allocated outside these times.
• Can receive plant moisture and water usage information from the master control unit and display it in a legible manner to the user. ○ The expected parameters show up on UI-Subsystem	• A webpage with the intended arguments and buttons to add plants and configuration settings should pop up, ensuring frontend works properly • User clicks on “add new plant” button then fills in the four parameters: plant name, water usage, watering hours, and the minimum moisture level • User attaches moisture sensor to the top of the soil of the plant • On the UI-Subsystem, a reading of the plant moisture and water usage information is displayed as “LOW/MED/HIGH” indicating moisture level and the water usage in liters. Confirm that these values show up and the parameter does not show up as empty

2.2.2 Control Subsystem:

This subsystem is a microcontroller which acts as the master control of the system. It will communicate via bluetooth/WiFi with the UI subsystem to take in the user's parameters. Also, it will receive wired data from the moisture sensing subsystems which encodes the state of each plant's two sensors. Using the information from the state of the plant and the restrictions from the user's inputs on whether watering is allowed or not, it will send out a signal to open or close the valve for each individual plant. Master to slave wired communication will follow a basic Read/Write protocol. The master will cycle through every address in the system and will first have the Write signal low and perform SPI communication with the slave and make its determination of whether to open or close that slave's valve. Then it will raise the Write signal as well as output the open/close signal on the specified wire for a set amount of time before cycling to the next address.

A detailed explanation of the master-slave communication protocol can be found in Appendix 1.

Requirements	Verification
1. When the master pcb is in the WIFI_CONFIG mode upon first startup, the master will act as an access point for the user to input WIFI credentials. a. **AP Name:** PLANT_SYS b. BLUE LED = WIFI_CONFIG	● turn on system, press reset, grab your phone and search for open WIFI networks ● connect to the Access Point (AP) named PLANT_SYS with password: config23 ● Follow URL: url_placeholder on your phone to interact with the system ● a prompt will appear to turn on and off an onboard Blue LED to verify connection. There will also be two textboxes to enter the name of your WIFI network and WIFI password. ● press enter to save password
2. wifi credentials are saved for the main system to connect t o WIFI a. PLANT_SYS = DNE b. Green LED = WIFI Connected	● confirm that your network connection to PLANT_SYS no longer exists ● a Green LED will be turned on to confirm a successful WIFI connection
● The MCU can connect to the main UI system and relay plant information ○ Website Status GET request shows Active ○ POST request appears on front end	● plants added to the system will show up on the front end, only if a plant exists ● data sent to the front end will cycle through by addresses and can be printed out in terminal for testing ● configuration details will be pulled and saved from the front end ● confirm by printing received data from UI

● The MCU can cycle addresses for each moisture sensing subsystem ○ HEX LED connected to the main board shows addresses cycling from 0-USER_PLANT_COUNT	● After initiating the irrigation system, a 7 segment LED will show the current address in the cycle. ● ensure the changing addresses are increasing over time and cycling values
● The system can perform serial communication between the moisture sensing subsystems and control subsystem. ○ Users can see different moisture levels appear on the UI subsystem when a moisture sensor is moved between dry and wet soil.	● after connecting at least 1 moisture sensing subsystem and indicate the number of slave boards connected ● confirm the MCU is cycling through the number of moisture sensing subsystems provided by user ● confirm moisture sensor data is being accepted and matches the appropriate moisture sensing subsystem (have 1 dry and 1 water to debug, dry or wet drain sensor) using the UI subsystem
● The MCU can send control signals to the moisture sensing subsystem ○ The user defines a plant's minimal moisture level to be at the mid range value, placing the sensor in dry soil results in the plant's valve opening, placing the sensor in water results in the valve closing.	● add a plant on the UI subsystem to be at a mid range value ● place the moisture sensor in a dry pot of soil or in the air to confirm 'dryness', and the valve will open ● place the moisture sensor in water or hydrated soil, the valve will close

2.2.3 Moisture Sensing Subsystem:

This subsystem will be replicated n times for n plants and contains the sensors and valve for its individual plant. It will have a sensor at the bottom of the plant to detect whether water is draining through the bottom of the plant or not, and it will have a moisture sensor within the top few inches of the plant's soil. It will contain the solenoid valve which is positioned to allow/disallow water flow to the plant. It will contain digital logic to implement the correct control/communication of the system. At a high level, it will have its unique address saved on chip and will compare with the address data lines to determine if it is being communicated with by the master or not. If it is, and the Write signal is low, it will allow data from its two sensors to be transmitted via SPI protocol to the Control Subsystem. If the Write signal is high, it will store the valve control signal value into a flip flop. The output of this flip flop is used as the control of whether the solenoid valve is open or closed.

Requirements	Verification
● The moisture sensing subsystem can communicate data to the master control unit within +/- 10% accuracy ○ Data which appears on the UI subsystem for the status of each plant is within +/- 10% of a voltmeter reading of the plant' sensor output	● Use system in normal operation with moisture sensors in plant soil. ● Use a voltmeter with negative probe at a plant's top level moisture sensor's ground pin and positive on the sensor's output pin. ● Read the voltmeter's measurement. ● Compare with the value on the UI subsystem for the plant in question and ensure the value is within +/- 10%.
● Each moisture sensing subsystem operates independently and is not influenced by other plants. ○ Switching between dry and wet environments on one plant opens and closes that plant's valve while not affecting the valve of any other plants.	● Use system in normal operation with moisture sensors in plant soil. ● Select one moisture sensing subsystem, use the UI subsystem to set the minimum moisture level to the mid-range value. ● Alternate placing the moisture sensor in question between a pot of completely dry soil and a container of water. ● This specific unit should have its valve open when in the dry soil and close while in the water; the other units should not be affected by this switching.

2.2.5 Tolerance Analysis

The most critical component of our system is getting accurate moisture data from the plants to the master control subsystem. If we are not getting reasonably correct data, our control decisions may be often incorrect and thus the system may operate seemingly randomly. We will aim to have the data received by the microcontroller to have less than 10% error from the actual value read by the sensor.
Our sensors and our 8 bit ADC will both run on a 3.3V supply voltage. However, online reading suggests that the max output of the sensor will be 80% of Vcc = 3.3V * .8 = 2.64V. So our analog sensor output will have an operating range of 0 - 2.64V.
Our 8 bit DAC sourced at 3.3V will detect levels every 3.3V / (2^8) = 0.129V step. Sampling an analog signal with an 8 bit DAC introduces quantization error while going from the “infinite precision” analog signal to the 8 bit precision DAC. Effectively, our DAC will map input [0, 0.129V) to 0V, [.129, .258V) to 0.129V, etc. The quantization error can be considered the actual analog signal minus the voltage it is mapped to. Thus, we can see that the max quantization error is one voltage step- here, 0.129V (ie, worst case is the actual signal is o[n] + 0.128999999V which gets mapped to o[n] where o[n] is the output of the DAC). Our max error in the signal will therefore be .129V / 2.64V = 0.0489 = 4.89%, comfortably within our desired 10% error margin for communication of the moisture data.

2.3 Cost and Schedule

2.3.1 Cost Analysis:

[Budget Spreadsheet]

2.3.2 Schedule

Week	Task	Person	Due
9/25	1. add initial references 2. update schedule once dev boards are ordered 3. define slave design elements 4. define master design elements 5. initial pcb designs 6. buy parts / finish parts list 7. start a safety manual	1. Carlos 2. ALL 3. John 4. John 5. Carlos 6. ALL 7. Carlos	Design Review Sign-up closes Design Document 11:59p
10/2	1. prototype backend 2. Code wifi within microcontroller 3. prototype moisture sensor + valve 4. refine pcb designs	1. Carlos 2. Kevin 3. John 4. ALL	Design Review 8:00a - 6:00p PCB Review 4:00p - 6:00p
10/9	1. submit first pcb designs a. master and slave 2. update safety manual	1. ALL 2. ALL	1st Round PCBway Orders 4:45pm Teamwork Evaluation I 11:59p
10/16	1. review/refine pcb designs 2. Validate slave hardware design	1. ALL 2. John	2nd Round PCBway Orders 4:45pm
10/23	1. last chance to review/refine pcb designs	1. ALL	3rd Round PCBway Orders 4:45pm Individual progress reports 11:59pm
10/30	1. Frontend UI 2. Backend 3. Program ESP32 for WiFi 4. Implement communication protocol	1. Kevin 2. Carlos, Kevin 3. Carlos, Kevin 4. John	
11/6	1. Program ESP32 for WiFi	1. Carlos, Kevin	
11/13 (Mock Demo)			

11/20 (Fall Break)			
11/27 (Final Demo)			
12/4 (Final Presentation)			

Kevin: Frontend UI, Backend, Coding ESP32 using Arduino for WiFi
John: Slave hardware design, master-slave communication protocol design
Carlos: ESP32 onboard Configuration, Backend, PCB design

3 Ethics and Safety

Safety Standards: Following the IEEE guideline on the safety of our project, it is designed to be easy to use and compatible. It is safe to use and will not have the ability to cause property damage (IEEE Code of Ethics 7.8.9) **[1]**. Any high voltage components will be properly sealed to protect against the elements, to ensure user safety and prevent property damage. Informing the user how to use the device properly will be crucial to ensure safety and functionality.

Water Usage Regulations: In compliance with state regulations (e.g., Illinois watering restrictions), implementing features in our project to help prevent excessive water usage during specific times and seasons. Any rules or regulations on a state and local will be encouraged in the system (ACM 2.3) **[2]**. The user will be notified of these restrictions via push notification.

- Illinois: No watering is allowed between 10 a.m. - 4 p.m. in all areas during the period from May 15 through September 15. **[3]**
- Champaign Water Restriction levels (Voluntary,Mandatory,Emergency) will be monitored and notify users to limit water use. **[4]**

Intellectual Property and Attribution: When developing unique Wi-Fi technology, it's important to credit others' work appropriately and ensure that our project respects existing patents and intellectual property rights. Properly citing and respecting the work of others helps maintain ethical standards (ACM Code 1.5). **[2]**

Privacy Concerns: Ensuring the privacy of users' data and information transmitted over the Wi-Fi network is crucial. Following privacy standards of ACM Code 1.6 **[2]**, each user has a right to privacy and privacy standards will be followed to ensure information accessible via the

internet is protected. This will be done by safeguarding any of the user's personal information by following industry level privacy practices in our code.

4 Citations

[1] "IEEE Code of Ethics." IEEE (Institute of Electrical and Electronics Engineers). https://ieee.org/about/corporate/governance/p7-8.html

[2] "ACM Code of Ethics and Professional Conduct." *ACM (Association for Computing Machinery)*. 2018. https://www.acm.org/code-of-ethics

[3] "Naturescape Blog: Regional Water Restrictions." *Naturescape Blog | Regional Water Restrictions.* 2015. www.naturescapedesigninc.com/regional-watering-restrictions.html#:~:text=It%20is%20unlawful%20for%20any,plants%2C%20or%20any%20other%20vegetation

[4] "Model Water Use Restriction Ordinance." *Champaign County Regional Planning Commission*, CCRPC , 2013, https://ccrpc.org/documents/model-water-use-restriction-ordinance/

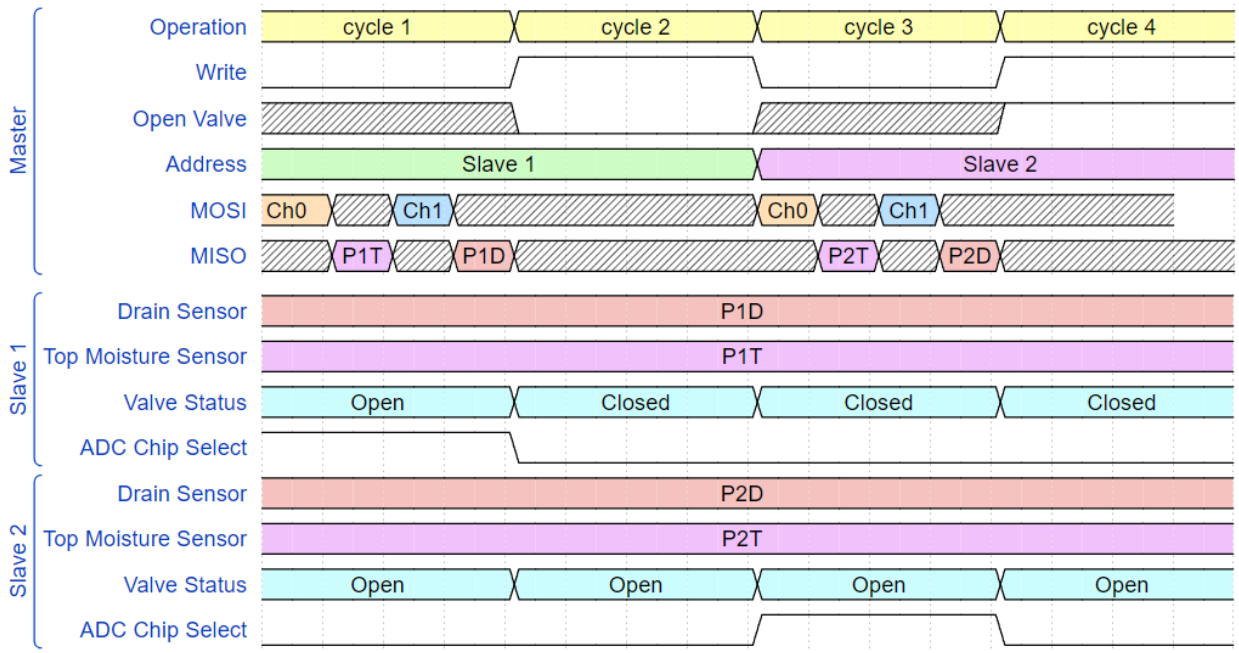
Appendix 1

In the effort to make the system as scalable as possible, we will be using digital hardware to control the slaves instead of a microcontroller per slave. To implement this, we will be using a custom communication protocol outlined below. The system will operate by using digital logic to implement the following pseudocode for each slave:

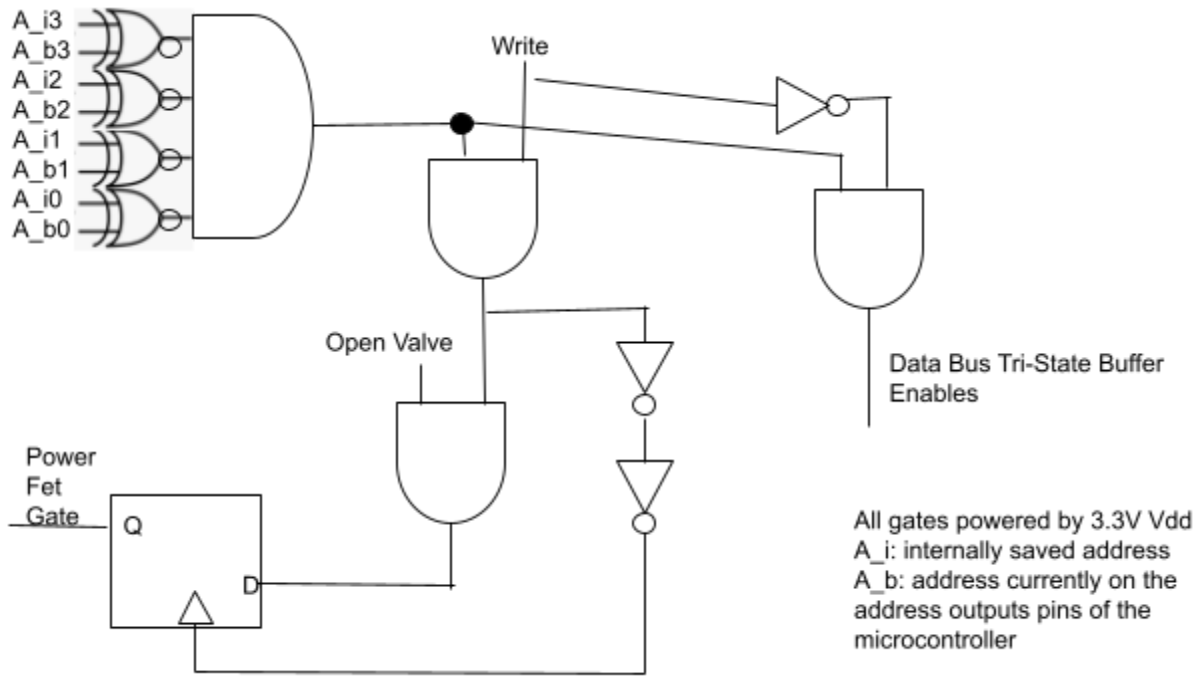
```
if (address == myAddress):
        if (Write == 0):
                myChipSelect <- 1
        if (Write == 1):
                myValveState <- Open Valve
```

The chip select signal will be used to initiate communication between the master and slave via SPI communication protocol. Since we want to actually perform reads from two different channels (the top level moisture sensor and the drain sensor) during our overall read operation, each read operation will contain a brief "untoggle" where the control unit sets the address to a reserved non-operational value (ie 1111) which will ensure all slaves chip select signals are 0. The address will then be set back to the slave's actual address and a new SPI conversation will

be had, this time with Data In instructing the ADC to output data from the second channel. This was not shown on the waveform as it would require going down to the SPI clock level and get needlessly messy.
The valve state will be stored in a Flip Flop clocked by the AND of the address comparator and Write signal, so it will update when the master is writing to the particular slave and hold its value constant otherwise.
The master will be responsible for raising and lowering the Write signal, cycling through the slaves' addresses, performing SPI communication protocol, reading the data on the buses when the Write signal is low, using the bus data and the user's inputs to make a decision on opening or closing the particular slave's valve, and outputting that decision when the Write signal is high. We will have to take care of the specific timings of when the master cycles, reads information, and outputs information after consideration of the worst case set-up and hold times of the digital hardware. A high-level waveform is shown on the next page.



Credit to Aliaksei Chapyzhenka (github: https://github.com/drom) for the waveform creator.



Slave digital logic overview

ECE 445 Parts List

https://www.digikey.com/

Item	Price	Digikey PN	Description	Comments
ESP32-S3-WROOM	$6.40	1965-ESP32-S3-\	MCU	
PLASTIC WATER SOLENOID VALVE	$6.95	1528-2003-ND	Water Valve	
SEN-13322	$6.50	1568-1360-ND	Moisture Sensor	
RPT-6003	$21.46	1866-4095-ND	AC/DC Converter: 3.3V, 5V, 12V, 40W	
ESP32-C6-DEVKITC-1-N8		1965-ESP32-C6-	Dev Board	
ADC081C021CIMKX/N	$1.28	296-ADC081C0	adc	
CD74HC366E	$0.87	296-33070-5-NE	tri state buffer	
SN74LVC1G175DCKR	$0.39	296-16998-2-NE	flip flop	
SN74HCS86PWR	$0.35	296-SN74HCS8	xor	
SN74LVC1G14DCKR	$0.29	296-11608-2-NE	inverter	
SN74HCS21PWR	$0.35	296-SN74HCS2	4 input and	
SN74LVC1G08DCKR	$0.33	296-11602-2-NE	2 input and	
IRLB8721PBF	$1.07	IRLB8721PBF-NE	Power mosfet	

Master Cost:	$27.86
Slave Count:	2
Total Slave Cost:	$55.88
Total Parts Cost:	**$112.30**
Machine Cost:	$43.92
Parts + Wage Total:	$21,503.14

Materials	Price	Link	Description
Liquid Electrical Tape Spray	$7.49	https://www.ha	Circuit Waterproof Spray
1/2 in. PVC Tee	$0.64	https://www.ho	T PVC connection hose->pipe->T->(valve + next slave)
1/2 in. x 3/4 in. F-Adapter Fitting	$1.27	https://www.ho	Valve to PVC connection
1/2 in. PVC Plug	$1.69	https://www.ho	Plug end of Tee tread allows for expansion
1/2 in. PVC M-Adapter Fitting	$0.75	https://www.ho	For expansion with PVC pipe
1/2 in. PVC Hose Adapter	$3.90	https://www.ho	Hose->pipe adapter
1/2 in. x 10 ft. PVC	$4.18	https://www.ho	pipe
All-Purpose Fast Setting Clear Cement	$3.98	https://www.wa	pvc glue
Quick Connect	$0.50	https://bc-robot	wire connection to power valves
Material Total:	$28.56		

Labor Cost

Avg Salary For ECE:	$98,472.50
Estimated Hourly Wage:	$47.34
Estimated hrs to Complete:	60
Team Members:	3
Total Wage Costs:	$21,303.00
Fabricator Hourly Wage:	$21.96
Fabricator Hours:	2
Machine Cost:	$43.92