MEMORANDUM

To: Dr. Espinoza-Wade, Department of Mechanical Engineering, Cal Poly SLO

erwade@calpoly.edu

From: Terak Hornik — thornik@calpoly.edu

Owen Clobes* — oclobes@calpoly.edu

Date: 6/14/2024

RE: **ME 507 Term Project - Autonomous plant watering robot**

*Owen was on the project at the beginning but dropped the class but volunteered to continue in a lower capacity for the last and most critical weeks.

Introduction:

This project as implied by the name is a robot designed to take care of up to 8 small plants with no human aid.* This will be accomplished via a single axis moving watering spout. The basic idea is that all 8 (now 4) plants are placed in built in saucers and the initial values are inputted into the code. Then once activated the system will monitor the moisture levels in each pot and when the levels drop below the defined threshold the system will move the waterspout to the appropriate location and slowly water the plant while checking the moisture levels. Given an adequate water supply and assuming it is placed in a location with adequate natural light this system should be able to keep 8 (now 4) plants alive indefinitely.

Overall the project did not work as intended as some key components broke the night before the demo. While it is not definitively clear it does seem that the project is pretty close to being fully operational. After finals calm down and my schedule gets a bit of an opening I do intend to complete this project. I want to say right off the bat thank you to Jack Miller as without his help I would never have gotten this far.

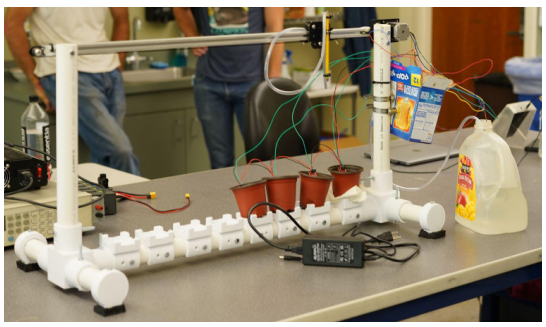
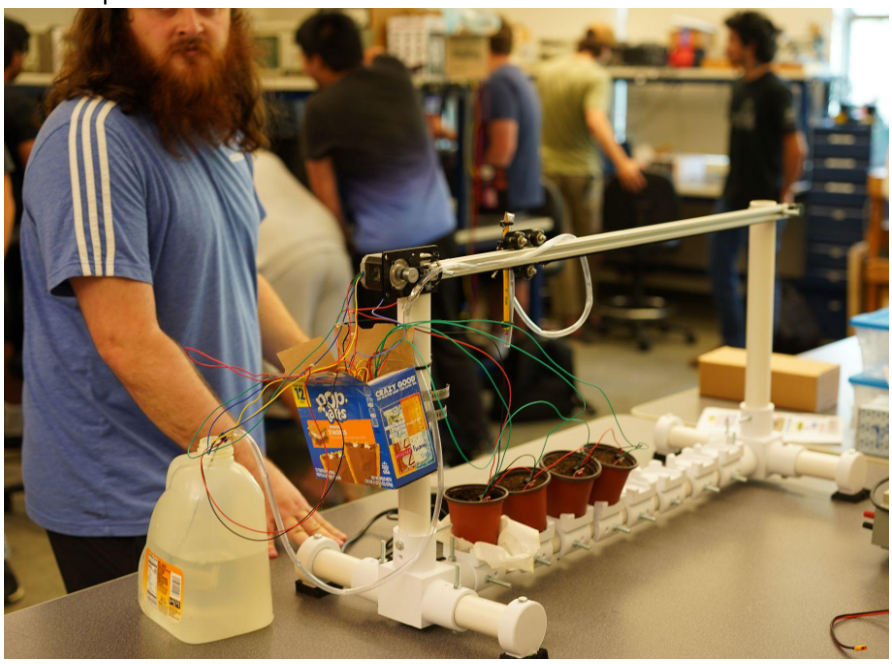


Table of Contents

Introduction: **1**
Scope & Requirements: **3**
Requirement # 1: 3
Requirement # 2: 3
Requirement # 3: 3
Requirement # 4: 3
Requirement # 5: 4
Requirement # 6: 4
Requirement # 7: 5
Hardware (Mechanical): **6**
Owen's Portion of Mechanical Hardware (Assembly) 6
Terak's Portion of Mechanical Hardware (CAD, Printing, & Debugging) 6
Hardware (Electrical): **9**
Power circuitry 10
Motor circuitry 11
Controller circuitry 13
Sensors 13
Actuators 14
Software Implementation: **15**
Modeling: **16**
Novelty: **17**
Project Evolution: **17**
Appendices: **20**
Additional photos: 20
Additional 'Rubric Specific' Comments: 39



Scope & Requirements:

Regarding the project scope I have met all the defined requirements given.** I will define the requirements then explain and justify the previous statement on an individual basis. The given requirements (paraphrased from the initially posted project guidelines***) are as follows:

1. A custom PCB designed around an MCU
2. 2 or more actuators
3. 2 or more unique sensors
4. Some sort of closed loop control
5. A wireless E-stop
6. No custom machined parts
7. Comply with Asomov’s 3 laws of robotics

Requirement # 1:

This requirement was the one thing waved for me when Owen dropped the class. That being said I do have most of a custom designed PCB done. So in some senses I did do this but it just didn't make it into the final project because Charlie wanted to prioritize getting it working over the PCB. I will refer to the custom PCB that I almost finished throughout this memo.

Requirement # 2:

This one is very objective and straightforward, I have 2 unique actuators. One is the stepper motor and the other is the water pump.

Requirement # 3:

Again this is rather straightforward and objective, I have 9 (later updated to 5) sensors total and 2 unique types. 8 (later updated to 4) of the 9 (later updated to 5) sensors are the moisture sensors and the last one is the limit switch.

Requirement # 4:

There is closed loop control. To easily define this I will use figure 1 below. For my system the input is the power provided to the water pump and the output is a moisture

reading on a given plant. When the system adds a bit of water then re-measures the moisture level that is the equivalent of going around the loop shown in figure 1. Since control can only be implemented one way (once watered the pot can not be actively dried) time delays are used to prevent any significant overshoot from the targeted value.

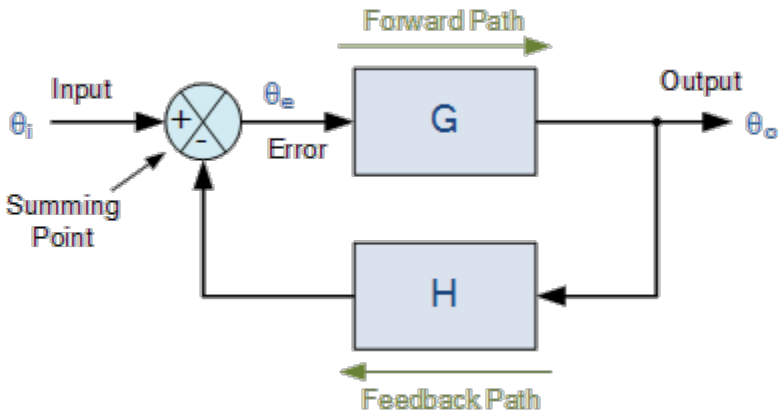


Figure 1: General example of closed loop control.

Requirement # 5:

This could be interpreted a few ways. On the one hand the project was constantly stopped (due to it not being operational) so in a sense I did this too well.

Using a more serious interpretation I had a built in (and thus wireless) E-stop that activated along with a red LED when there was a problem (A moisture measurement outside of the range of values that makes sense or too many attempts made to water a plant without it being watered). So there wasn't a traditional dead man's switch but I think this makes more sense for this type of project. If the project was a robot that walks around the traditional approach would make sense but for this it would mean that the robotic plant caring system would only work while a human held down a button and that kind of defeats the purpose.

Requirement # 6:

Again this is rather straightforward and objective, I have no custom machined parts. As was suggested in the initial project guidelines document I made all the custom parts needed in SolidWorks and FDM 3D printed them on my personal FDM 3D printer.

Requirement # 7:

The 1st part of the 1st law is met as this robot can not harm a person. However, the 2nd part of the first law is not met as it will not do anything to prevent a person from coming to harm. The 1st part of the 2nd law is pretty clearly obeyed as it is a simple enough device that it will without question obey any orders given via its code. However it fails the 2nd part of the 2nd law as it would still follow orders given even if the orders would harm a person or allow a person to come to harm. The 3rd law is more questionable. It will not actively act in self preservation but on the other hand it is rather robust and will not endanger itself actively. Overall I feel my robot has sufficiently met this requirement enough to consider it completely met for practical purposes.

Also I want to note that for some reason the last (and possibly most important) of Assamov's laws of robotics was omitted here. It is the 0th law: a robot may not harm humanity or through inaction allow humanity to come to harm. And the associated changes to the 1st, 2nd, and 3rd laws.

* Humans will still need to do the following:
    - Make sure the water reservoir isn't empty.
    - Check for the red error LED light signaling that something is wrong
    - Define initial values (location of the pot and minimum moisture percentage desired for the plant)

** There has been some small adjustment of the requirements due to Owen dropping the class.

*** I am omitting requirements that don't apply to this project at all such as things that apply to mobile robots or batteries.

Hardware (Mechanical):

Owen's Portion of Mechanical Hardware (Assembly)

The frame of the assembly was constructed mainly from Schedule 40 1.25 in PVC pipe and custom 3D printed parts. The PVC was cut to length with a skill saw. Holes for fasteners were drilled into the PVC using a ¼" drill bit and components were fastened together with ¼"x20x3.5" hex bolts. The beam along the top of the frame is a 1" 8020 aluminum extrusion used for mounting the belt pulley and the water tube carriage.

The 8 pot holders were 3D printed and fastened along the bottom beam of the frame. The initial holes in the 3D printed parts were slightly too small for the selected bolts and were drilled out with the same ¼" drill bit.

Note: this section and only this section was written by Owen. The rest was written by Terak alone.

Terak's Portion of Mechanical Hardware (CAD, Printing, & Debugging)

Using Solidworks I was able to make part files for each of the 4 custom parts to be FDM printed. The 4 parts are as follows: 'PVC 4 connector' that forms the joins at the bottom corners, 'PVC to Al' that forms the top corners, that is placed on the ends of the support legs and supports the bottom of the pots on the lower PVC rail. 2 PVC 4 connectors were printed taking ~1.25 days each, 4 PVC caps were printed taking ~1 day together, 2 PVC to Als were printed with the PVC caps, and 8 PVC to plant adapters were printed taking ~2.3 days together. Regarding the slicing, extra walls were chosen (4 instead of the standard 2) to provide some bending resistance while allowing for a little bit of compressibility and a lower infill density (~25% density and 0.28mm layer height was used for all prints).

I used the same 2 global variables in each file: 'OD' for the outer diameter of the PVC (I began these files before we finalized a dimension for this) and 'clear' from the clearance between the OD and the ID designed to fit over it thus allowing for a slight clearance fit. The final value for OD ended up being 1.66in which is the OD value stated by Home Depot for 1.25in PVC pipes. After a few test fits the final value for clear was 0.05in allowing for a snug but still easy to work with fit.

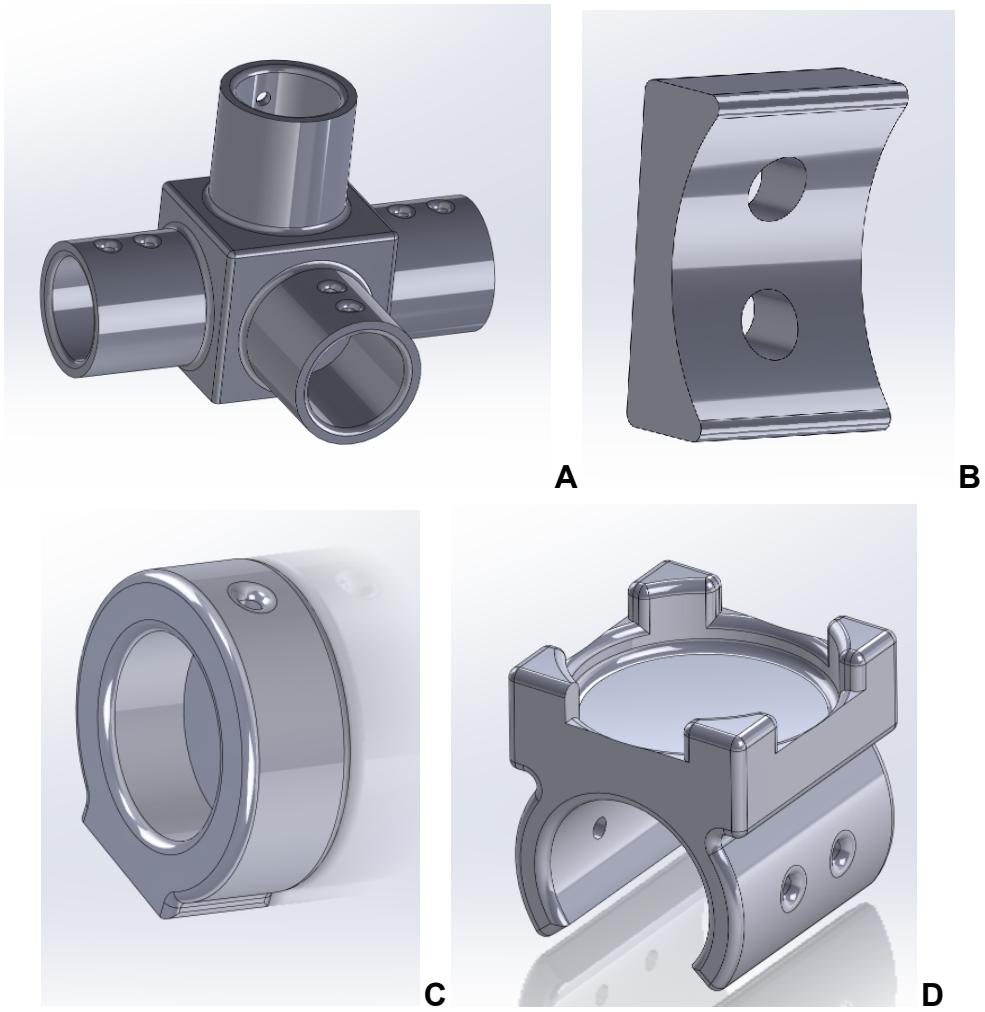


Figure 2: Solidworks screenshots of all CAD files. 2A is ‘PVC 4 connector’, 2B is ‘PVC to Al’, 2C is ‘PVC cap’, and 2D is ‘PVC to plant adapter’.

In the final build the parts in figure 2 parts A and D worked great. Part C needed an extension of the flat surface to make it work as the bolts going through the vertical holes on the part shown in part A were larger than expected and caused the whole device to rock back and forth. This was fixed by hot gluing a piece of foam on the bottom surfaces (this can be seen in the appendix of pictures). As for the adapter shown in figure 2B I decided to scrap it at the last minute as Owen had only drilled a single hole (rather than 2) in the aluminum for the mounting bolts and it was too late to re-design and re-print a model with one centered hole.

When Owen brought in the ‘finished’ build there was a lot left to do on the mechanical hardware side of things. This included the following: attaching the motor, dealing with the ends of the belt, adjusting the thickness of the belt, attaching the belt tensioner, repositioning the aluminum attachment so the tensioner would actually fit,

attaching the limit switch, adjusting the feet so the frame sits flat, attaching the part that slides along the aluminum when the motor spins, mounting the motor attachment plate, attacking the motor, attaching the pop tart box to house the electronics, locating/attaching the water reservoir, attaching the water tube so that the location could move without kinking (this was never successfully finished), attaching the limit switch, and a few more smaller details.

Lastly, I will mention some of the last minute hardware debugging. When the belt was unspooled it became clear that it was slightly too wide so it was cut lengthwise with a pair of scissors so that it would fit. Also the little crimped end caps that came with it seem to have disappeared so using some of the excess I was able to develop a simple solution: poke 2 holes in the belt an inch or so apart and use a wire to tie the holes together in order to create a loop. Then to keep it intact I twisted the ends of the wire together, soldered the twisted ends then added some hot glue to create a mechanically reliable structure. This cornered a bit when I first thought of it but after some informal testing with some scrap pieces of belt I found it worked very well. SImilarly there was a list minute problem with the limit switch attachment. Luckily, I noticed that it fit rather snugly in the side slot in the aluminum. So by wiring it up in advance and then press fitting it into the slot and hot gluing it in place it was reliably secured and ready for use.

Hardware (Electrical):

The main electrical hardware used was an array of moisture sensors, a stepper motor, a water pump, and a ‘blackpill’ microcontroller with its associated development board. Other more minor things include the following: transistors used as switches, LEDs used to indicate various statuses, a motor driver IC, a power management IC (this was removed towards the end of the project, I will leave it here as it did play a role in the project development), and a larger number of passive components. I conceptually broke these into several semi-discrete categories: power circuitry, motor circuitry, controller circuitry, sensors, and actuators. The full electrical hardware (everything except the limit switch, motor driver and other peripherals) can be seen below in figure 3.

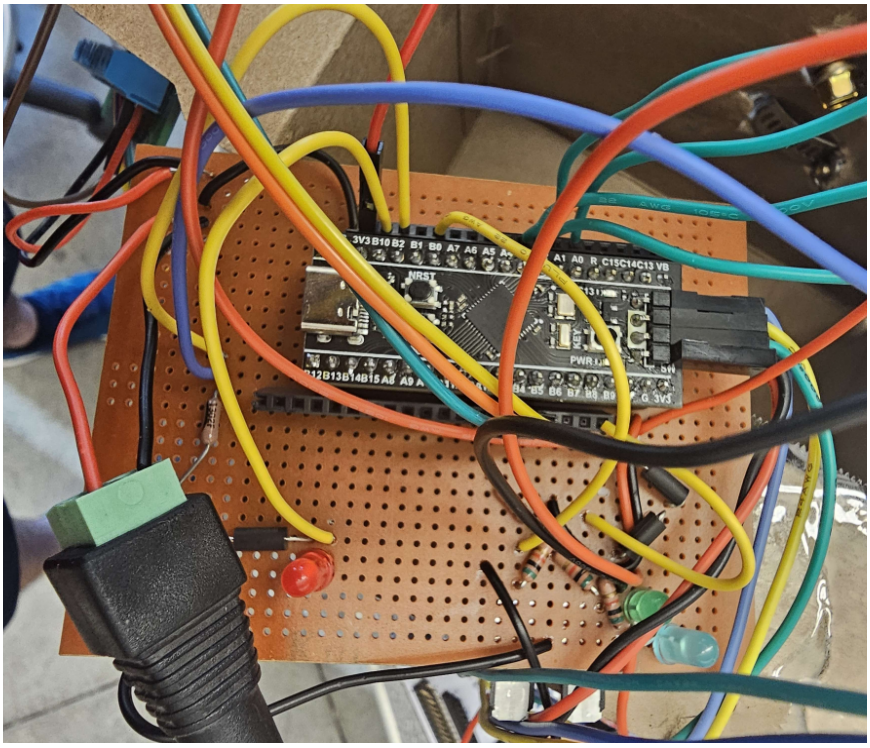


Figure 3: Final circuitry used in final implementation.

Power circuitry

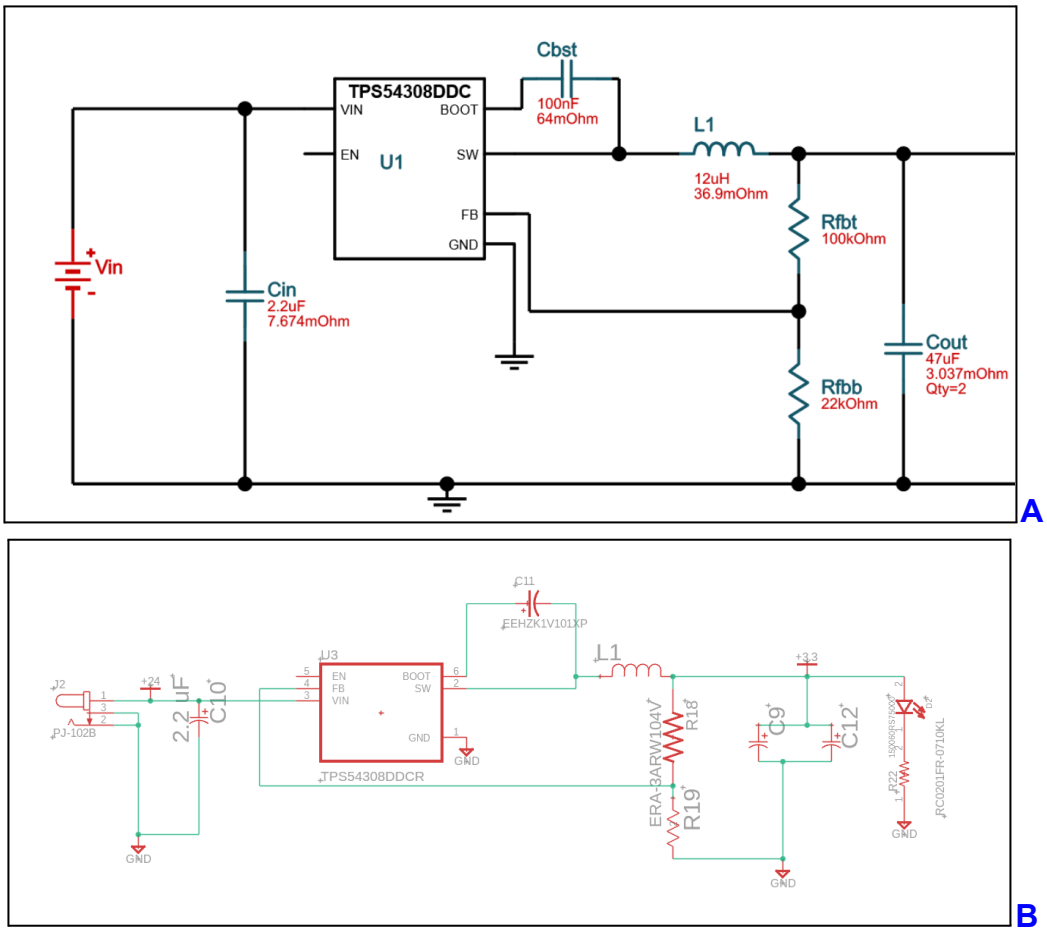
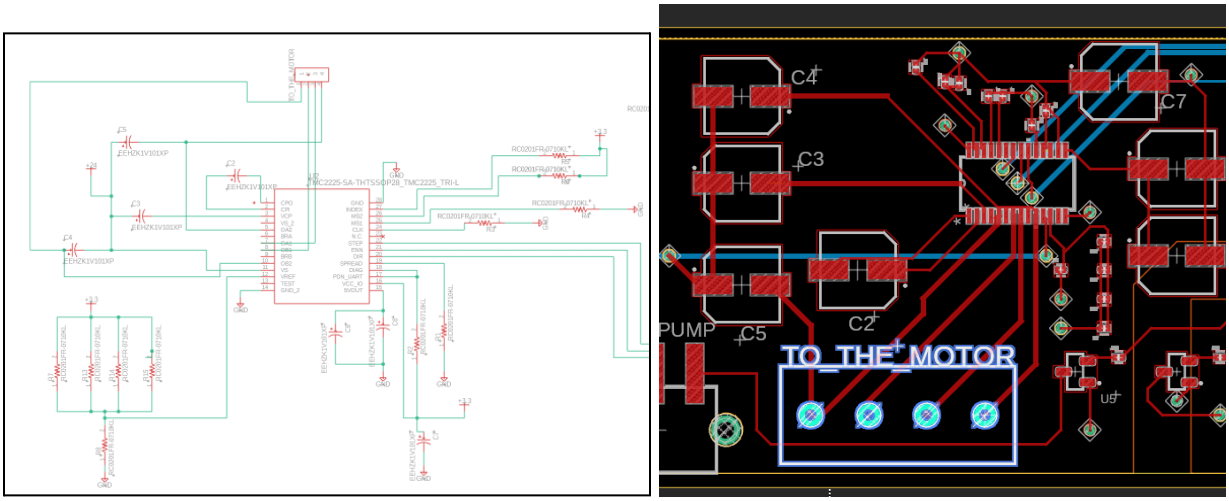


Figure 4: Power circuit schematic.
4A shows the schematic generated online. 4B shows the fusion schematic I made based on the schematic in 2A.

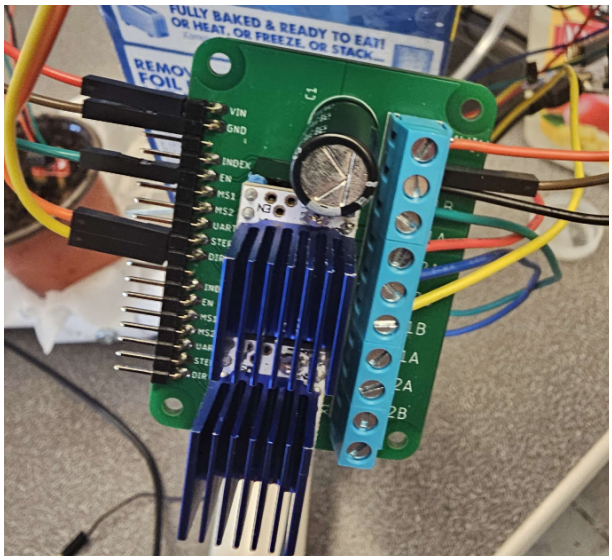
The power circuitry was centered around the TPS54308DDC IC which helpfully steps down the voltage from the incoming 24V to the more desirable 3.3V needed for this application. In essence this is what allowed me to use the 24V supply from the wall that the stepper motor needs to function while also allowing me to power the smaller things that would be destroyed by that voltage. However, there was one key problem with this: the IC was very small (about 1x2x2 mm). My hands shake with an amplitude of more than 1 mm so my plan of soldering wires onto the surface mount leads then attaching the wires to my perf board was a problem. I asked Dr. Wade for advice and he suggested using tweezers and a magnifying glass. Since that didn't help with the problem of my hands shaking I asked if I could have a friend do this part of the soldering and I was told no. When I asked Charlie in lab he suggested I use a larger IC he had laying around. Unfortunately, that didn't go anywhere as that IC needed a particular inductor that was not on hand. Charlie then suggested I simply leave the USB-C port of

the blackpill plugged in as my 3V3 supply and to just plug in that and the 24V source in parallel so that is what I ended up doing despite my hard work in the development of this power circuitry.

Motor circuitry



A **B**



C

Figure 5: Motor circuit initial schematic and physical layout.
5A shows the schematic in fusion where the bulk of the relevant things for this category are located. 5B shows the physical layout of some of these components on the PCB editor. 5C shows the final implementation using a driver Charlie provided.

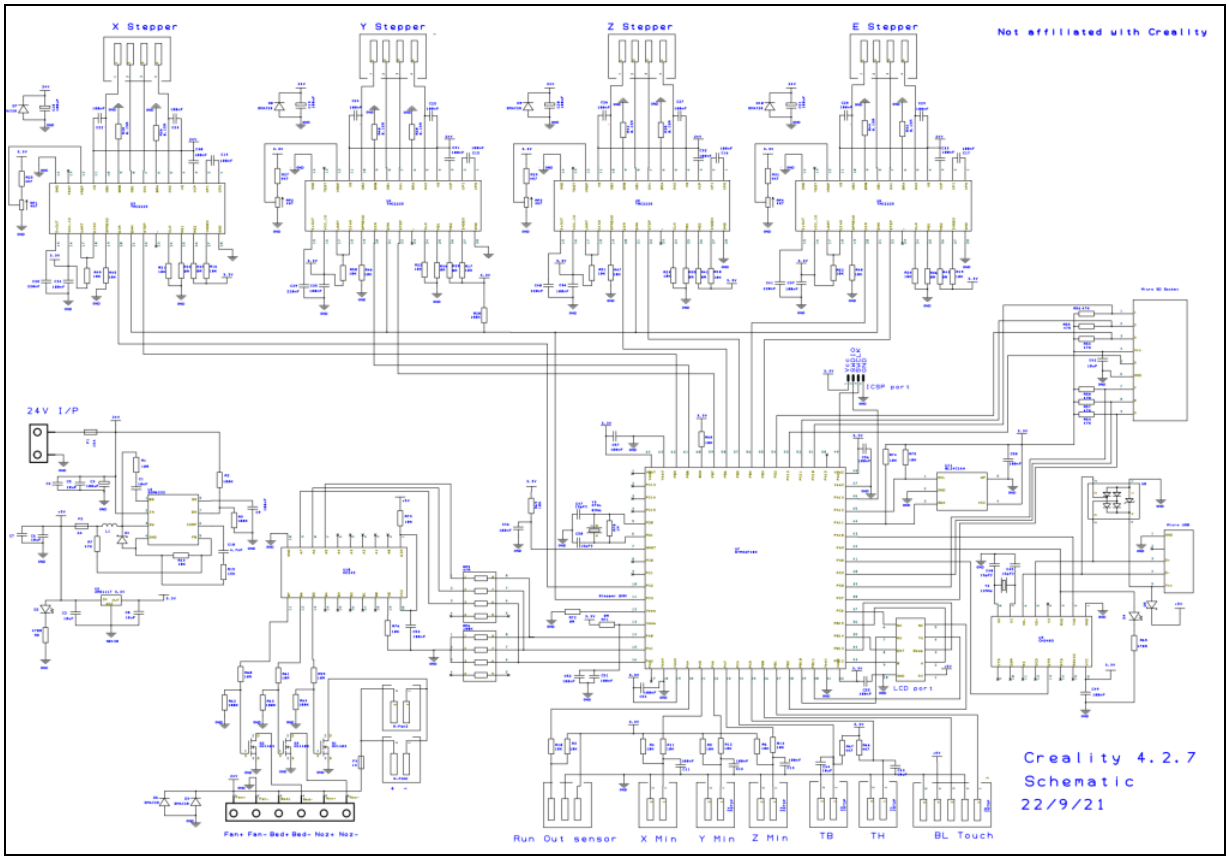


Figure 6: Motor driving circuit for similar application.
This figure shows the schematic for the Creality 3D printer motherboard. Importantly the stepper motor I am using is the exact same one that many Creality 3D printers use. Note: My 3D printer that was used to print things for this project has this motherboard.

Overall this turned out to be one of the more complex aspects of the project. This is because the motor doesn't interact directly with the MCU but instead has a driver (the TMC2225 IC) that acts as an intermediary. Speaking generally the MCU tells the driver 3 things: ‘DIR’, ‘ENN’ and ’STEP’. As I interpret them they individually represent 3 key parameters needed for the motor to know what the MCU wants it to do. ENN enables the motor to run, DIR sets a direction to turn and STEP sets the distance to turn. Using these 3 inputs the driver can operate the motor as desired. ENN and DIR simply take a single digital input so the appropriate pin can simply be set or reset as needed. For STEP it moves one step each time the pin is set. So this means that a sort of PWM signal can be used where each rising edge corresponds to a single step taken and the pulse width corresponds to the speed the motor moves at. For this project I used a pulse width of 1 millisecond (for a half cycle) which corresponds to about a quarter of an inch per second (0.246 [in/s]). While this is rather slow for many applications, for the purposes of this project it is totally acceptable. If needed the variable assigned to the half cycle delay can be altered to adjust this speed to any value desired.

Controller circuitry

The primary part of the controller circuitry is the blackpill that controls everything. Beyond that there were 2 transistor switches (turning on and off the pump and sensors respectively), 3 LEDs (indicating power is working, an error has occurred and the sensors are on respectively). And several passive components. As can be seen in figure 3 the blackpill pins were accessible due to the female ports I attached along each side of the microcontroller.

Regarding the ST link I had some issues getting the debugger to work using the board I was working with so I (with help from Jack) used one of the ME405 kits to rig up a way to debug my code as can be seen below in figure 7. This allowed me to debug some initial issues with my code. Most of this was simple things like typos but before my blackpill died I was able to begin the process of calibrating the moisture sensors. During this (it is unclear if this was before or after the death of the blackpill) the results suggested the higher moisture would result in a lower moisture reading to the MCU so I updated my code accordingly. Now I think this may have been a fluke caused by the semi-functional blackpill.

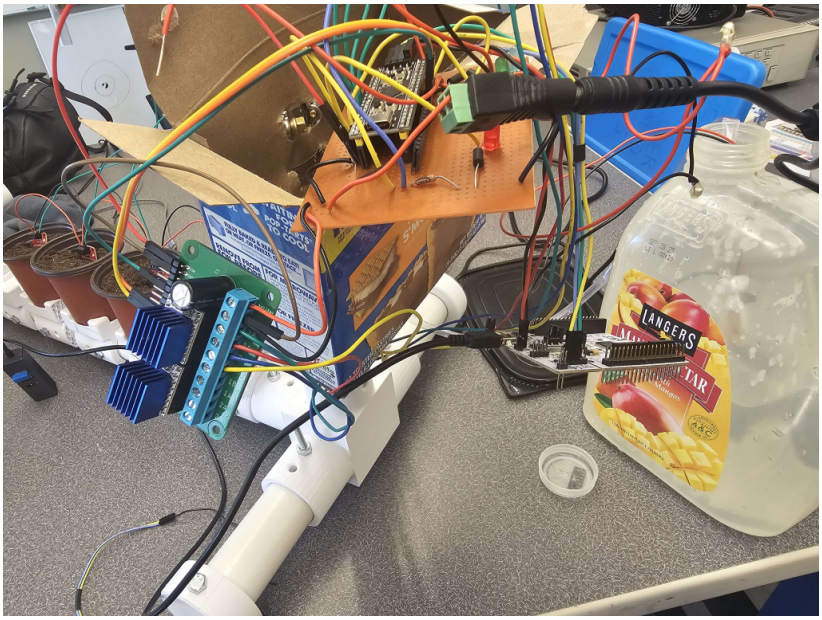


Figure 7: ST link setup

Sensors

The 2 unique sensors I am using are limit switches (LINK) and moisture sensors (LINK). Figure 8 below shows what they look like. Their function is very straightforward so I won't discuss in detail, but I will say that the moisture sensors detect the moisture in

the soil and the limit switches allow the stepper motor to find the zero point on the rail. Also it is worth noting that the moisture sensors are recommended to be kept unpowered unless a measurement is being taken so as to minimize corrosion effects.

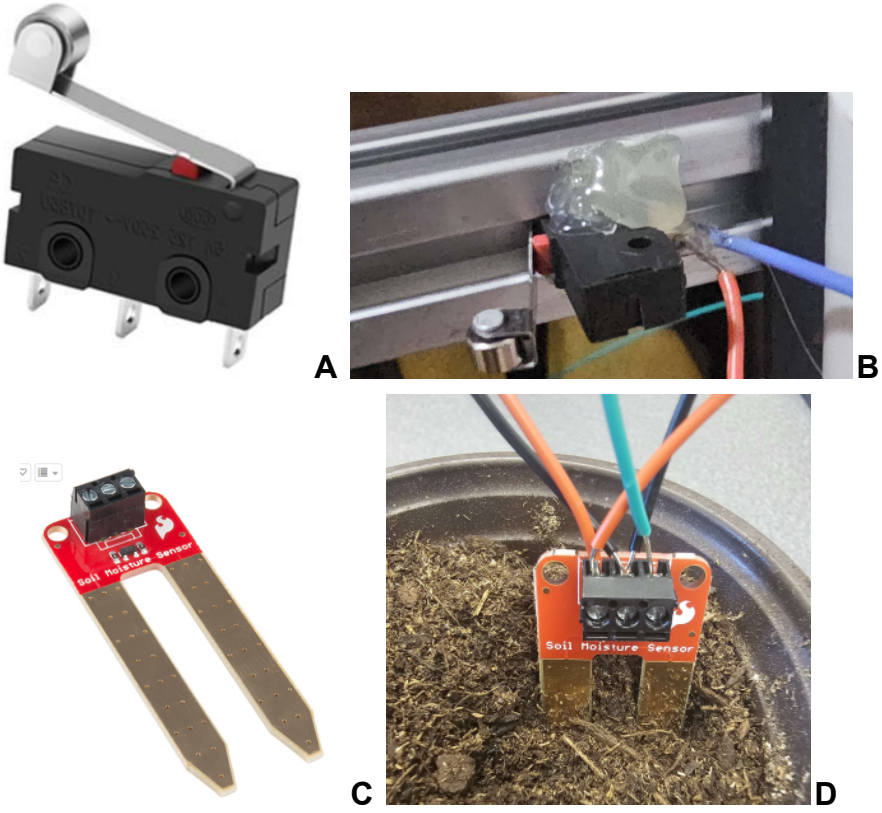


Figure 8: sensors. 8A shows the limit switch, 8B shows the limit switch on the final design, 8C the moisture sensors and 8D shows the moisture sensors on the final design.

Actuators

The 2 unique actuators I am using are a stepper motor (LINK) and a water pump (LINK). Figure 9 below shows what they look like. Their functionally is also very straightforward so I won't discuss in detail, but I will say that the stepper motor moves the watering outlet to the appropriate location and the pump provides the water to the plant.

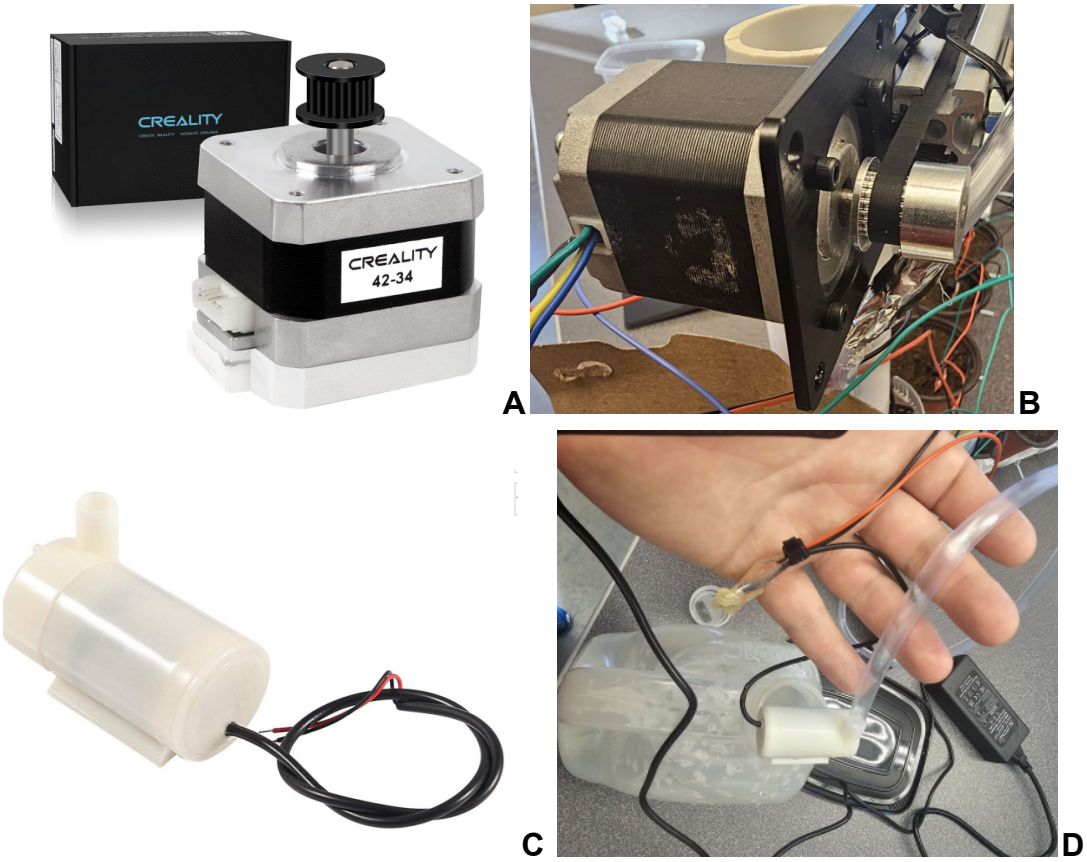


Figure 9: Actuators. 9A shows the stepper motor, 9B the stepper motor on the final design, 9C shows the water pump, and 9D the water pump on the final design.

Software Implementation:

My code was intentionally written on a single .c file. This was because it was easier for me to manage one file as I was working alone. I will now proceed to describe how the structure of the code functions from a very high level perspective via a bullet point list. For more detailed information see my attached source code.

- User defined values are inputted including the following: desired moisture levels for each individual plant (both a minimum and desired overshoot value), the distance in inches from home (the location of the limit switch) to the center of each plant, a wet calibration reading, a dry calibration reading and various delay constants.
- In main (but before the while loop) the following things happen:
    - The distance in inches to each plant is translated into a distance in units of steps of the motor.

    - The moisture minimum and overshoot values are converted from a percentage of a mV reading.
    - Several arrays are formed.
- In main (inside the while loop) the following things happen:
    - All plant moisture values are measured and stored.
    - For each plant:
        - Moisture readings are checked for basic coherence.
        - Moisture readings are checked against the desired values.
        - The watering counter is set to 0.
    - If plant(s) need water then the following things happen:
        - The motor is brought back to home.
        - The motor is brought to the appropriate location of the plant that needs water.
        - The pump is briefly turned on to provide a little water.
        - The moisture levels are measured and checked again.
        - The variables indicating whether watering needs to take place are updated.
        - The watering counter is incremented and checked against a maximum value.
    - A delay occurs so the plants aren't checked with unnecessary frequency.

Modeling:

There was no mathematical or physical modeling that I would personally consider advanced enough to merit significant description in this memo. So, I will briefly list the various models used:

- ➢ Explicit two way mapping between motor steps and distance. At the end of the day it came out to 80 steps per mm.
- ➢ Implicit one way mapping between pump power and water provided. This was never finalized as I needed the rest of the project to function in order to empirically test this.
- ➢ Implicit one way mapping between watering a plant and its moisture levels
- ➢ Explicit one way mapping between the mV signal given by the moisture sensors and the moisture percentage of the soil. This was intended to be defined by some calibration measurements of fully dry and fully saturated soil then the other values could be adjusted appropriately.

Novelty:

Novelty can be thought of here in 2 distinct senses. 1 being uniqueness compared to my peers in ME 507 this quarter and 2 being uniqueness compared to existing designs in general.

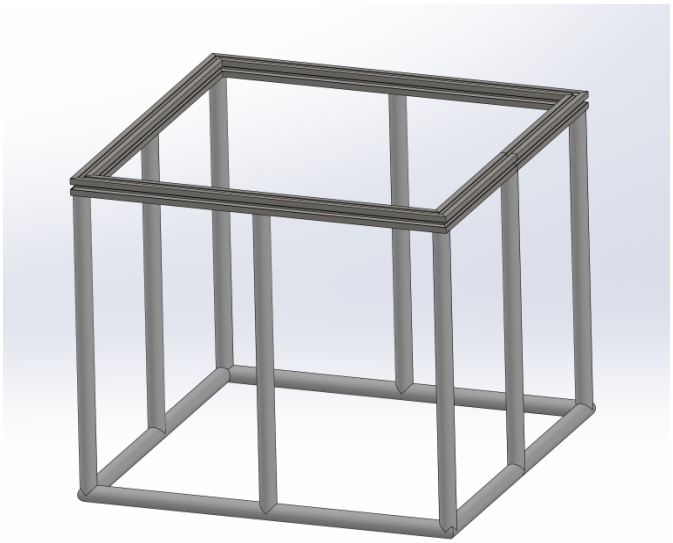
In the 1st sense (based on my admittedly limited understanding of what everyone else is doing) this project is unique in that it is stationary (very few projects had this characteristic), has no battery (very few projects had this characteristic), it was primarily developed by only one person (zero other projects had this characteristic), and runs on a comparatively longer timescale (I believe my time scale and time constants were in fact the largest in the class). Another novel aspect of this project is that it works with a (non-human) living system.

In the 2nd sense (based on my admittedly very limited understanding of what exists in the world in this particular niche) this project is most unique in that it is made by me. Additionally, it is designed so that it can be expanded without too much difficulty, it takes plants into its structure (rather then a small device being attached to an existing plant as seems to be common with this type of device), it is self contained, it uses off the shelf 3D printer parts (such as the aluminum extrusion or stepper motor), and that the plants can be treated individually and thus watered according to individual needs of various species of plants.

Project Evolution:

Evolution (in the colloquial sense rather than the technical sense like that used in biology) is a great word to describe the development of this project. As the situation and as my interpretation of the situation changed the project changed to align with it. Up until the last day this process was ongoing. Figure 10 below shows the original sketch of what the project will look like and then what it actually looked like at the end to illustrate how different they are.

A



B

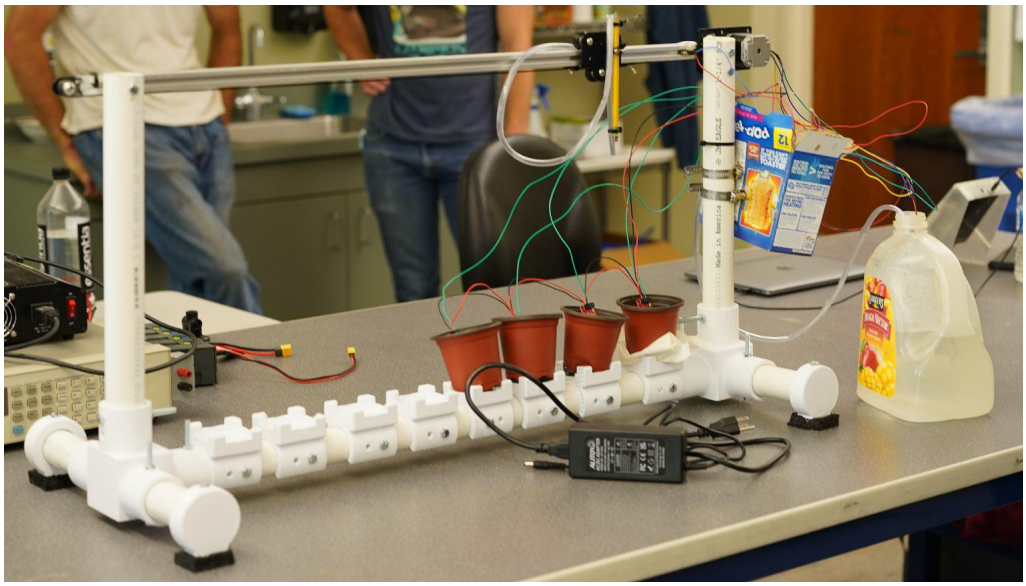


Figure 10: Original project sketch and final project result. 10A shows the sketch submitted in the original project proposal. 10B shows the project at my presentation during finals week.

The first major instance of this project evolving to fit a different situation was the change from 2 axis to 1 axis which was due to the fact that originally we had thought that there would be more time allocated to the project, less lab and homework assignments and that the design would be more difficult to implement then we original thought.

Lots of things had to be ‘taken in stride’ with this project including the following: switching to a perf board after all the work designing the custom PCB, switching to a different type of perf board after all of the work organizing my circuitry on the other type,

losing my partner week 8, losing 4 of 8 moisture sensors the day before the project demo, Owen's minimal hardware assembly, parts (like the belt or wire connections to the stepper motor) not fitting, changing from surface mount to through hole parts, switching the motor driver circuitry, switching the power circuitry or a USB cable, and much more. Sometimes it felt like more work went into things that ended up not being part of the final project than the things that did, but it was more or less universally taken calmly and my focus was readjusted to the next task.

Some of these chances were not predictable ahead of time (ie: Owen dropping the class or losing 4 sensors) but some were (ie: changing to the 2nt type of perf board or the belt not fitting). I think that one of my biggest mistakes during this project was not being proactive here where I could have been. For example if I had test fitted the belt the day it arrived it would have saved me some significant last minute stress and allowed me to focus on other things at the critical moment. Overall I just wish I had more time to work on this so I could have made a working robot, but that can still happen over the summer.

Appendices:

Additional photos: