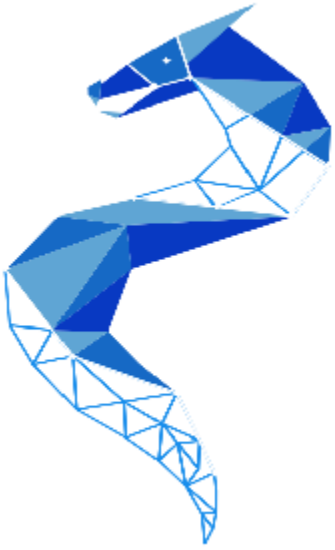
Senior Design 1: Final Report
3D Data Projection Grid
ECE Department



Group 27

Dallas Moody - Computer Engineer
Sean Jacob - Computer Engineer
Tomas Yepez - Electrical Engineer
Trevor Chesley - Computer Engineer

Reviewer Committee

Dr. Mike Borowczak ECE
Dr. Linwood Jones ECE
Sumanta Pattanaik CS

Advisor

Dr. Sahawneh ECE

Table of Contents

Chapter 1 - Executive Summary

Data visualization has been improving and evolving for a while now. We now have so many different kinds of tables, charts, and applications for our viewing convenience. We even have physical representations of data. Our goal is to make a three-dimensional data grid that shows data on the x, y and z plane.This is where the idea of a 3D data projection grid came from. Draco Labs, along with Dr. Mike proposed this idea to my group and said that it's a prototype and we have creative freedom.

This project will have many uses in the realms of education, topography, and data analysis. We hope to be able to use this project as a way to make it easier for everyone to visualize three-dimensional data without needing to scroll and drag around a screen to see. Some implementations are topography to show the different heights of a chunk of land and basic bar charts with multiple variables. We also talk about the different applications and uses of our project in chapter 2 with our motivation and background section as to why we chose to do this project and what our motivation was behind getting this to work and what it would improve in day-to-day life.

Our system will be composed of an 8x8 grid of LEDs in two dimensions. These LEDs will all be addressable and connected via daisy chain while staying in parallel to reduce power consumption. We will be adding epoxy towers as our three-dimensional aspect to show us a vertical view. Each LED has its own tower that can be raised to 4 different levels of height.The LED will then shine through the epoxy tower to make the tower light up. We will use 8 motors (1 per row of LEDs) with 8 gears on each motor rod (1 per column of LEDs) to raise up the tower. We have a bearing under each gear so the gear will only raise up the tower when a solenoid is active.

This report shows our planning and process of making this project. We start with the motivation and background of why we want to do this project, then we move onto research and selection of the parts needed for this to work. We then talk about our constraints and limitations that come with the nature of this design. We then show our design and schematics for each of our components and how they will be implemented into the design as well as how they will work together. Following will be the documentation of our fabrication and prototype testing. In our prototype testing, we didn't use a PCB, rather we just used a breadboard and wires to make sure our code worked and our thought process of the design would be achievable. Then we show our documentation of the bill of materials, our distribution of work, and our project milestones. Finally, we have documentation regarding our references and copyright permissions.

Chapter 2 - Project Description

2.1 Motivation and Background

In a data-driven world, 3D data visualization has become a problem in data analysis. In almost every field, there is a data analyst staring at graphs and interpreting all day. Trying to interpret data in three dimensions is difficult on a two-dimensional screen. We haven't had a consistent breakthrough of a system that can generate 3D data physically.

Inspiration for the project comes from netlogo that designs a 2D array of pixels to show, for example, cows and sheep grazing in a field of grass and dirt. We want to make that more immersive by using the motorized system to extend the LEDs view to make visualizing the data/picture in 3D. We also take inspiration from the children's toy lightsabers that extend out and shrink back in as our way to make the height changing system work.

With an addition of a rectangular prism shaped grid of LEDs, we would be able to not only see data in three dimensions but walk around and physically point, touch and explain the data to many people so everyone has a better understanding of said data. Everyone has their own way of learning and understanding new material and hands-on is a very popular one. Having this physical display of data could help these people who need hands on to understand what they're looking at.

Another application is in the school system. This system could help teachers better explain their content to the children, again, in a more physical hands-on way. With the addressable LEDs, the system can also be used not just for data visualization but in any way you can think of from light shows to decoration to complex data visualization. Especially with the common use of LEDs in households with kids asking for LED strips all the time, a new system or breakthrough to the third dimension would be a milestone in technological advancement that will be remembered forever.

While we do have static displays of data like blocks and styrofoam, it doesn't have the ability to be changed dynamically to accommodate different data sets and/or types of data displayed. With the projection grid, we can switch the display from a bar graph to a line chart to even just a random combination of lights. Also, with the motorized actuation system, we can visualize moving in real time so we can see the changes in data as they happen.

With the use of LEDs instead of physical objects as a diagram, we can see data change with the turning on and off of LEDs. An example could be showing a 3D pie chart or showing the topology of different areas in the world. Another example of the usefulness of this project in other areas of study.

2.2 Features and Functionality

The function of the 3D projection grid is to be able to display data in 3D. Using an 8x8x4 grid, data will be able to be displayed using red, green, blue, and white LEDs. The colors of the LEDs will not be fixed, meaning each LED will be able to display each of those colors when desired. Examples of data that could be displayed with this grid are mathematical functions, atoms and bonds in 3D, electric/magnetic fields, and more. With a 3D printed base housing the electronic components such as the battery, the microcontroller, and any other components needed to make it function, the LEDs will extend up from the base in order to simulate whatever data is trying to be visualized.

The static matrix aspect of the 3D grid will allow us to represent the 3D grid of the LEDs. Using an array, each LED will be able to be mapped to a specific coordinate in the LED grid. This type of mapping will allow each LED to be controlled independently from another in terms of brightness, color, or on/off state. Since the matrix is static, the size is fixed at compile time, meaning predictable memory usage and efficient indexing. Efficient indexing is important here to be able to access each LED with its coordinates without having the microcontroller search through. The static dataset playback target is 1 matrix/minute. With each matrix describing the on/off state, brightness, and color state of each LED, multiple matrices can be loaded into the grid to be cycled through to display a completely different data set every minute.

An intuitive feature that may be implemented into the 3D grid is modular expansion. What this means is the user will be able to take an identical copy of the 3D grid and connect it to the first one. Given this connection, the user can decide to operate the second, connected grid as a separate grid to create a larger composite grid. Expansion is an important feature to be able to display larger data or display the same data to a larger group of people. When two grids are connected, the number of LEDs doubles, meaning the ability to display the same data as with one grid, in greater detail. The connection of two grids is designed to automatically recognize the connection and synchronize animations. Using a “plug and play” style of connectivity, these 3D grids could theoretically be connected infinitely to be able to make larger and larger grids to increase resolution, configure different layouts, and scale the display.

The 3D grid is intended for indoor use only. Its electronics, motor system, and LEDs are designed for controlled environments. In outdoor lighting, the LED lights will likely not perform as they would indoors due to the lack of contrast, diminishing the effect of the grid. The grid will unlikely be waterproof meaning it would need to stay indoors at all times to avoid any rain damage.

Having a marketable product is crucial to be able to sell it. The 3D grid is designed with customer feedback in mind as well as taking into consideration other similar products on the market. A crucial part of this project that plays into the marketability role is the modular extension.

2.3 Related Work

One of the main drivers for this project was the website netlogoweb. When viewing different models on this site, certain datasets felt lacking in how they represented and displayed the data. For example datasets on animal activity in a region are represented by boxes, squares, and little icons representing different animals. This can vary significantly from how the region actually looks, however, and adding a height element to work alongside color would add to the understanding of how the region is laid out. Along with netlogo there exists other 2D data modeling services which all have the same goal as the 3D projection grid. These include Repast, MASON, and GAMA, with GAMA also being focused on geographic modeling.

Although there doesn't seem to be a retail product on the market for sale to consumers, there are other renditions of the 3D grid that can display data. When looking on YouTube, videos can be found of similar iterations of the 3D grid. Most of these grids are built using very small LED bulbs. There are examples of the LEDs hanging down from a console unit (the power, PCB, MCU housing, and more) hooked onto the ceiling, but also examples of the console unit being the base with the LEDs extending from the base up. Moreover, the grids online have the ability to change the display in real time. They show how the grids can display sine waves propagating in real time, they can show a sort of firework animation, and much more with very fast refresh rates. In other words, all of these renditions are very similar to each other with no significant innovation from one to the next.

The main notion that makes our proposed product different from those in the "market" already, is the height control aspect. In our proposed project, we will be using a telescoping effect with the motorized actuation. Similar to how a telescope can extend forward with tubes that nest inside one another, we will be using that to display data in 3D. The motor feature will be how our 3D grid displays data of different heights. Instead of lighting up LEDs sequentially above one another, we will be raising the height of the grid physically.

Another aspect of our project that cannot be found elsewhere on the internet, is the inter-module connectivity. Each project seen online has a stand-alone project with a console unit and LEDs that extend from it. Nothing else. Our project is different because we will be able to take an identical 3D grid, and connect it to the original grid to display larger data. The two modules/grids will be able to communicate with each other to display data on a larger grid. The connection should be seamless and easy for anyone to connect and disconnect.

Lastly, we intend our grid to be very easy to use. The entire idea behind the grid is to be able to display different data. The user will input a dataset to be displayed (matrix) and output the result immediately. A goal of this grid is to be able to shuffle through different data sets, essentially the grid refreshes to show different data than what was previously shown, at least 1 dataset per minute, with a goal of 10 datasets per minute.

2.4 Overview of Project Goals

Basic Goals

- Create an 8×8 LED(64 LEDs) grid capable of displaying 2D matrices.
- Create a motorized system with the ability to represent data as different heights, to work in conjunction with the LED grid.
- Enable simultaneous visualization of datasets combining both color (LEDs) and height (motor).
- Implement 4 discrete height levels per square (resulting in an 8×8×4 matrix)
- Design and build opaque squares to diffuse light and represent height levels.
- Create a stable base that ensures proper LED spacing/mounting
- Be able to display 8x8x4 datasets with a refresh rate of at least 1 dataset per minute.
- Establish proper power distribution for all 64 LEDs
- Develop a microcontroller-based system to control and address LEDs.
- Design a PCB capable of routing power to the LED and height control systems, along with aiding in communication with the MCU and data storage.
- Coordinate motors and LEDs through a main microcontroller to synchronize color and height.
- Provide a human-computer interface (HCI) for uploading and interacting with data.
- Preset visualization modes(topographical, density, heat maps)
- Have a storage system able to store different static 3D datasets.
- Working communication interface between software and hardware
- Basic safety system

Advanced Goals

- Expand to include 4 additional height levels resulting in 8 discrete height levels per square (8×8×8 matrix).
- Be able to display 8x8x4 datasets with a refresh rate of at least 5 datasets per minute.
- Develop advanced data visualization modes (e.g., real-time updates, animated transitions).
- Optimize for efficiency and responsiveness of motors and LEDs.
- Allow motors to work for both extension and retraction
- Allow for scalable expansion beyond 8×8 (e.g., larger grids).

Stretch Goals

- Be able to display 8x8x4 datasets with a refresh rate of at least 10 datasets per minute.
- Explore interactive controls (touch, gesture, or web-based interaction).
- Add dynamic dataset switching or streaming visualization.

Basic Objectives

- Create an 8x8 LED grid used for data visualization with the ability to display static datasets.
- Build an 8x8 motorized system on top of an LED grid with each having the ability to display up to 4 different heights.
- Have an MCU connected to an 8x8 LED grid with the ability to address each LED with RGBW+ colors.
- Have the ability for the MCU to store real time data on the different heights of each square with the ability to control each height.
- Have connectors on top of LEDs to allow for more modules to be connected
- Use MCU to store state information, log events, and be able to reset from errors in operation.
- Have a memory storage device able to store multiple static datasets that the MCU can draw upon.
- Include the ability to upload data to the memory module via a connection to an external source.
- Include the ability for the MCU to detect when a new module is connected and adjust the LED grid to display the proper dimensions.
- With the motorized system, control the height of the z-axis to increase the height dimension by at least 4 levels.
- Have all components attached to a 3D printed housing.
- Store all power on housing and be able to sustain operation for more than 1.5 hours.

Advanced Objectives

- Expand motorized system to include 4 more height levels available by increasing square height and motor functionality, resulting in 8 discrete levels
- Be able to refresh the LED grid to display up to 5 different data sets a minute by increasing performance of system
- Live simulation of data, real-time interactive control, and reactive element to possibly incorporate game-like system by enhancing system complexity
- Store all power on housing and sustain operation for a minimum of 2 hours.
- 2-way motorized system to extend and retract levels without user assistance needed
- Achieve modular expansion, allowing grids of 8x16, 16x16, etc. with full capabilities

Stretch Objectives

- Be able to refresh the LED grid to display up to 10 different data sets a minute by increasing performance of system
- Implementation of interactive control methods, allowing for touch, gesturing, or web-based remote interaction
- Add dynamic switching, allowing for the potential of streaming visualization through cycling of datasets

Requirement Specifications

- The grid should contain LEDs that are *all* addressable. The LEDs will be addressed individually via an array in the software. This provides a design where there is no searching or lookup tables, just direct access to each LED.
- Base dimensions of the data visualization grid will be 8x8 LED grid centered around a base
- Base will be some 3d printed housing that is large enough to hold all hardware needed for the projection grid
- The entire grid and housing will be relatively light, with most of the weight coming from the motorized system
- Refresh rate 1 matrix/minute (goal 5/minute). Each matrix will correspond with a "design" or display of data. The goal is to be able to swap from one display to another at a rate of *at least* one display per minute with an optimal goal of 5 per minute.
- Motorized height control ≥4 levels (goal 8).
- Inter-module connectivity. The 8x8x4 grid should be connectable to another identical 8x8x4 grid. This connection should be synchronous across both grids so that data can be displayed across both, essentially both grids should be communicating with each other.
- The system should be able to display a 3D visualization of data depending on the user input from external feed.
- The LED system should support the full range of RGBW color space with the ability for numerous color combinations.
- Height controlled by motors.
- The system shall expect data input from the user through the form of USB/ external feed through the use of a web app or PC.
- For a system where so many moving parts are acting in combination, it is important to ensure the voltage levels are not too high as we do not want our power consumption to be too high to maintain safety.
- Maintaining low power consumption will be important, not only for safety, but for optimizing our system performance as a whole

Parameter	Specification	Engineering Requirement

System Specifications		
Dimensions	8x8x4	The grid will be designed in an 8x8x4 configuration as specified in the project overview
Weight	<10 lbs	The modules must be lightweight. Given the equipment that will be used to build it, less than 10 pounds is reasonable
Component Specifications		
Housing	3D Printed	Using a 3D printer to construct the housing for the electric components will allow for a sleek look while keeping costs low
Power Consumption	≤60V, ≤24V System Bus, and No Mains	Low input voltage is required to ensure safe construction and operation
Matrix Display	Response Time	The grid must fully to display a new matrix in under 1 second
Demo Specifications		
Refresh Rate	≥ 1 Matrix/Minute	The grid must be able to refresh the matrix being displayed at least once every minute. The ultimate objective is to be able to refresh 10 matrices every minute
Height Control	≥ 4 Levels	The motorized height control must be able to alter the grid's topology of at least 4 levels on the z-axis.
Colors	Red, Green, Blue	Colored LEDs must be used to create contrast when displaying data

Figure 2.1 *Requirements Table*

2.5 Hardware Block Diagram

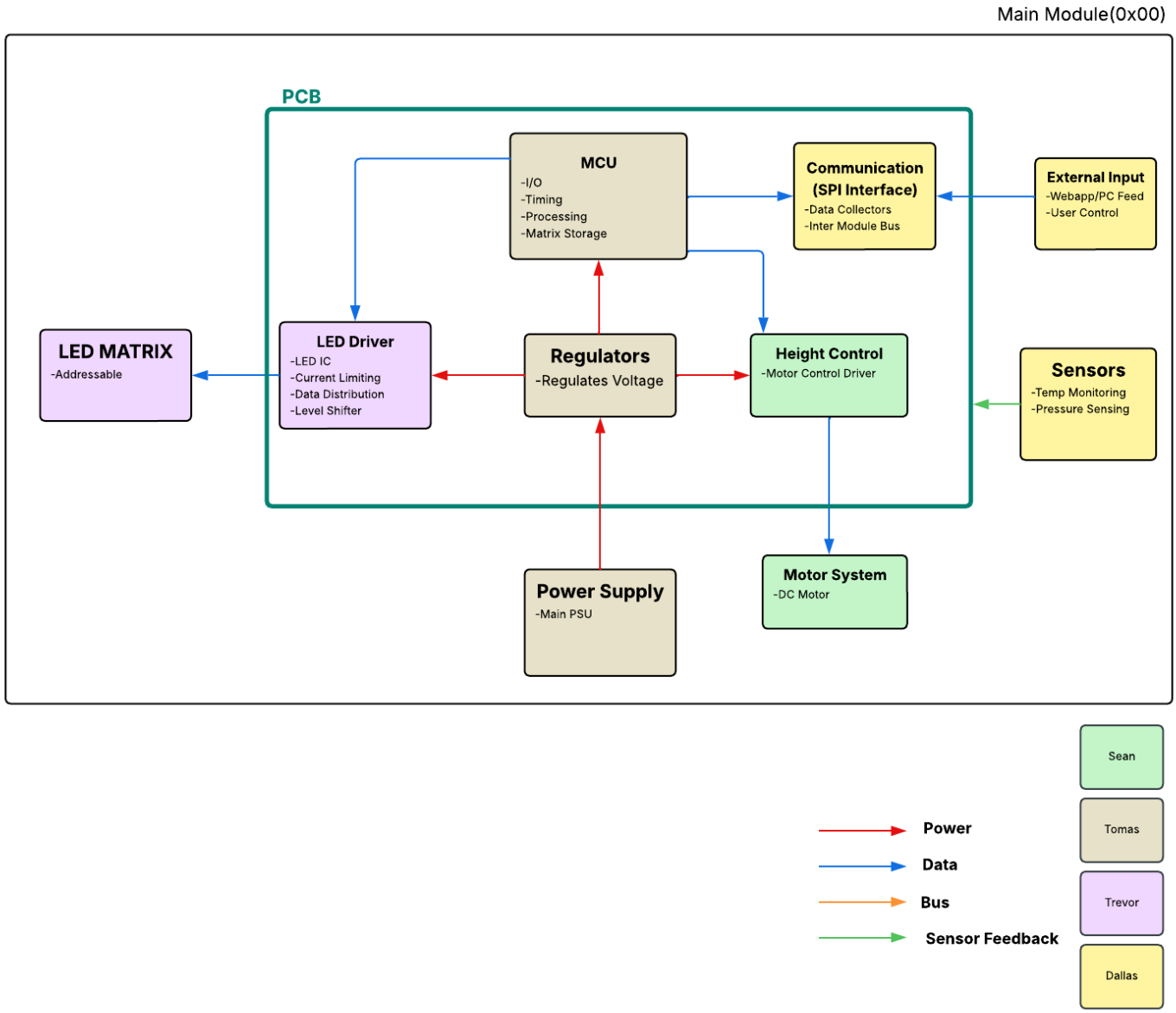


Figure 2.2 *Hardware Block Diagram*

Shown above is the hardware block diagram for the 3D projection grid. The block diagram outlines the fundamental aspects of the project. You can see the PCB section outlined in green along with all other subsystems needed for the project.

2.6 Software Flow Chart

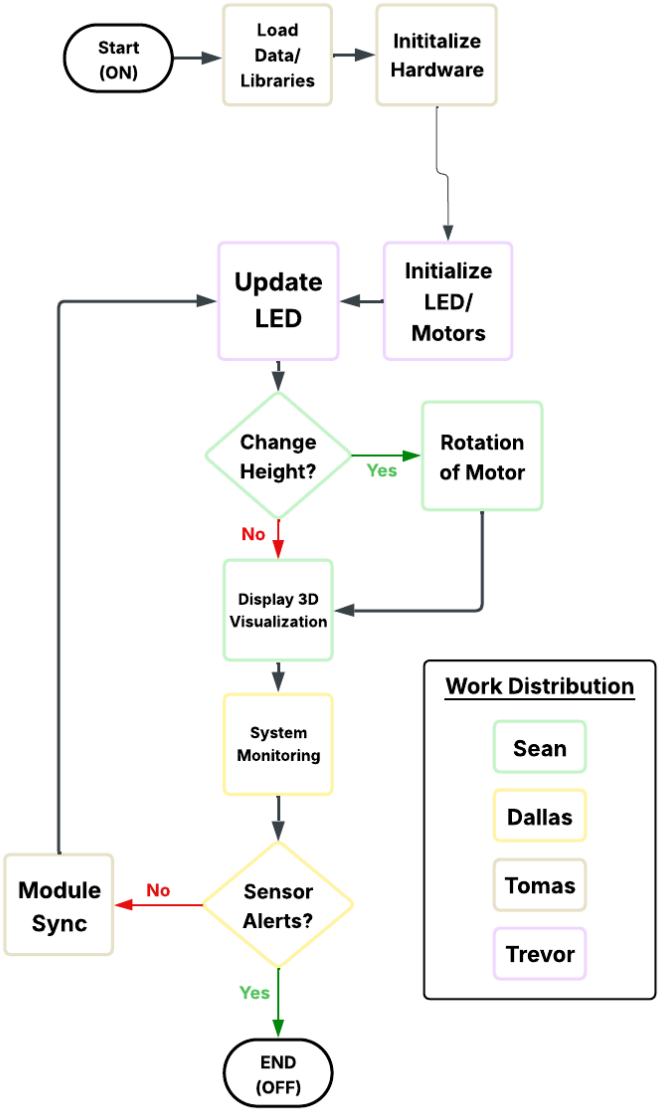


Figure 2.3 *Software Flow chart*

2.7 House of Quality

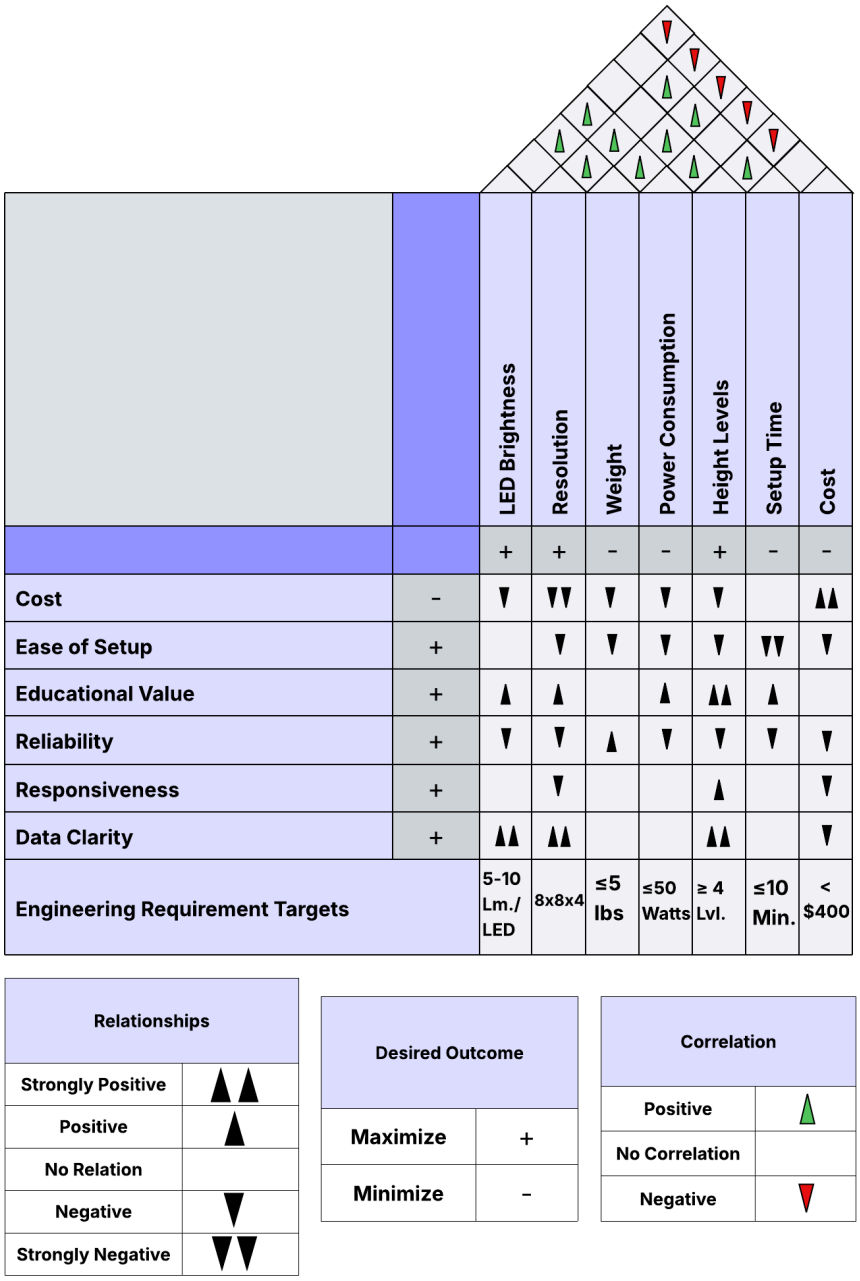


Figure 2.4 *House of Quality*

Chapter 3 - Research and Investigation

3.1 Technologies

3.1.1 MCU

3.1.1.1 MCU vs FPGA

For the programming in our project we have two different routes we could go. We can use a microcontroller unit (MCU) or a field-programmable gate array(FPGA). As with most things, there are advantages and disadvantages to both choices. An example would be that FPGAs are more powerful and precise than MCU's. There is a much smaller margin of error when using an FPGA. However, in most cases, the FPGA is a higher cost option and consumes much more power than an MCU would.

FPGAs use a lot of power because of their increased processing and usually having applications in parallel processing. MCUs are more low power coming with several different low power modes out of the box. FPGAs also aren't very user friendly as they are very complex when it comes to programming since they are more geared to the industry applications which are very niche. FPGAs also use languages like Verilog and VHDL, which are hardware description languages and make them way more difficult to work with. FPGAs also need external components to integrate UART, I2C and ADC.

MCU's have most if not all of these features already built in. This makes microcontrollers much easier to use and user friendly. They have a lot of documentation and online resources available that help with learning how to use it. MCU's also use common coding languages like C and Python which makes debugging and writing the code much easier than with verilog. With FPGA's complex hardware it's hard to find documentation that matches what you're using since they vary between vendors.

Criteria	MCU	FPGA
Power Consumption	✔	
Parallel Processing		✔
User Friendly	✔	
Debugging	✔	
Processing Power		✔
Cost	✔	

Scalability		✔
Peripherals	✔	
Documentation	✔	

Table 3.1 *MCU vs FPGA comparison table*

3.1.1.2 ESP32

The ESP32 is focused on networking apps and has built-in WiFi, Bluetooth, and other features. They have antennas and have at least 4 MB of flash memory. The processor runs a dual-core at 240 MHz and plenty of SRAM with 520 KB. You can also lower the speed to 80 MHz at minimum to lower power consumption.

The operating temperature is very low being at around 1.6 W of power consumption. It has all the peripherals that we would need already onboard like UART, SPI, I2C, PWM, ADC, DAC and 26 GPIO pins. These boards are also very cheap coming in around $10 per board.

We also have experience using the ESP in other classes so the learning curve would be pretty easy to get past with our experience. It also has a lot of online documentation to help if we come into any problems with the MCU.

The ESP supports multiple programming frameworks including Arduino IDE and ESP-IDF which allows the user to choose an environment that best suits their needs. There is also a very good community with open-source libraries that make it easy to implement common functions like web servers.

3.1.1.3 MSP430

The MSP is one of the MCU's we are most familiar with as they have been the main MCU used in most classes. These don't come with bluetooth or WiFi like the ESP however. They have 128 KB of flash memory. It also has less SRAM than the ESP with only 2 KB compared to the ESP's 520 KB.

The operating power consumption is very low being only 1 watt with 5 volts of operating voltage and 0.2 amps being the current draw. The MSP also has peripherals already onboard like UART, I2C, PWM, ADC, and DAC.

We have also had many classes using this MCU as well so the learning curve again wouldn't be too much of an issue with all of our past experience with this project.

Texas Instruments provides development tools like Code Composer Studio and MSP Launchpad that offer a good starting point for prototyping and testing. They also have a variety of variants with different configurations and peripherals allowing easy scaling based on what the project needs.

3.1.1.4 Raspberry Pi

The traditional Raspberry Pi has many potential benefits for this project but also has downsides. It honestly could be considered as a full on computer instead of an embedded system as it is a complete Linux machine, just a very small one. This makes it much easier to debug any problems we have. It also lets us choose between any programming language we want.

The 1.8 GHz quad-core CPU is plenty for our project, almost too much. This Pi also has at least 1 GB of RAM and at least 32 GB of storage capacity which is way more than enough for our project.. These have 2.4 and 5 GHz 802.11ac wireless, Bluetooth and Bluetooth Low Energy built in which may be useful to us. It may not have some peripherals but, through USB ports, we can add whatever we need.

It needs 5 V and minimum 3 A to power which is a crazy amount of power compared to the traditional embedded systems. It is capable of UART, SPI and I2C protocols through the 40-pin header GPIO pins. It does not have an ADC or DAC by default which can cause problems for our project. It is very expensive compared to the other embedded systems choices.

We don’t have as much experience with this so the learning would be a challenge.

3.1.1.5 Arduino MKR 1000

Arduino has Wi-Fi and LoRaWAN board families available which can benefit our project. The MKR Wi-Fi 1000 series boards use a 5 V power supply and operate at 3.3 V while only using 7 mA per pin used. It has a very low power consumption and supports using batteries and will charge automatically. These boards include UART, SPI, I2C, PWM, ADC and DAC, everything we would need while also having 2.4 GHz 802.11 bgn Wi-Fi, Bluetooth and Bluetooth Low Energy out of the box. With 256 KB of flash memory, 32 KB of RAM and a 48 MHz processor it’s up to standards for other MCU offerings and servers our needs just right. However, this seems to be too good to be true and you would be right. These run at about $40 which is very expensive compared to our other choices.

We don't have much experience with this MCU either and the debugging would still be difficult since we would first have to know how to use the product.

	ESP32	MSP430	Arduino MKR 1000	Raspberry Pi
Cost	$9.18	$20	$38.6	$15
WiFi	Yes	No	Yes	Yes
Bluetooth	Yes	No	Yes	Yes
UART	Yes	Yes	Yes	Yes
I2C	Yes	Yes	Yes	Yes
SPI	Yes	Yes	Yes	Yes
PWM	Yes	Yes	Yes	Yes
ADC	Yes	Yes	Yes	No
DAC	Yes	Yes	Yes	No
Flash Memory	4 MB	128 KB	256 KB	None
RAM	520 KB	2 KB	32 KB	1 GB
Operating Voltage (V)	3.6	5	3.3	5
Current Draw (A)	0.379	0.2	0.154	3
Power Consumption (W)	1.3644	1	0.5082	15

Table 3.2 *Comparison of Microcontrollers*

3.1.2 Height Control

For a projection grid where the height control is one of the most important factors, ensuring smooth and reliable transition is key. Using compressed air or motors were the first thought when attempting to come up with a method. After some research, we came up with a few potential ideas. Pneumatic cylinders, Individual Servo Motors, and Motors with Gear Racks were the options we found best-fit to handle the job.

3.1.2.1 Pneumatic cylinders

Pneumatic cylinders are a way to convert compressed air into a motion of some kind. These pneumatic systems work as mechanical actuators, generating force and producing a smooth change in position. Pneumatics often consist of a few key components to ensure proper actuation. In the case of a height control system, a cylinder containing a piston will receive air from a compressor and force forward movement of an attached rod. This works by dividing the cylinder into two sections, separated by a piston. When one section receives compressed air, it will push the piston towards the other section, pushing the rod. This idea works inversely when applying compressed air to the other side of the chamber, retracing the rod. The main idea is the conversion of compressed air's energy into mechanical force used in the motion of the system.

The extension and retraction of the rod through piston movement is not the only thing to consider when working with pneumatics. We also need to determine a consistent way to distribute the air to all the squares in the grid. The compressed air will flow throughout the system using tubing that connects through specific fittings. When dealing with a system involving many actuators, it is important we can guarantee proper air flow to the desired areas. We can use valves and regulators to ensure consistent results. The regulator helps in maintaining consistent air flow, which is crucial in a system where exactness is key. The regulator can guarantee that the square is receiving the proper amount of air flow for the desired height. Valves are used to allow or cut off air flow through an area of tubing. The valves can open(let air flow in) or close(cut air flow off).

When determining if pneumatic actuators are the right fit for a specific project, it is important to look at the trade-offs. One of the main disadvantages of a pneumatic system is the cost. The large range of components needed for pneumatics leads to a high initial cost. Having to purchase air compressors, cylinders, treatment equipment, and distribution tubing/fittings can turn out to be quite expensive. For a system with 64 actuators, we are looking at around $1000 for height control alone. Another problem with pneumatics is the precision in which we can realistically achieve. Air compression makes exact height change very difficult to achieve, especially in a system with so many moving parts. Power consumption is something else we have to worry about when working with air compression. Leaking air will cause a significant amount of waste which lowers overall efficiency of the system.

While pneumatics lacks in some areas, it excels in others. The reliability of a pneumatic system to continue to function over time is one of its main benefits. Because of the fact that pneumatic actuators don't rely on electrical components, they make a great fit for long term usage. They are able to maintain functionality in somewhat harsh conditions, making them very robust to their surroundings. In a system where seamless transition is ideal, actuators work perfectly in smooth transition of states. The flow of compressed air into an actuator, followed by movement of piston and rod, gives the feeling of a smooth build up to a desired height.

3.1.2.2 Individual Stepper Motor

When looking at other possible ways to implement the height control of our system, we found stepper motors. Having individual stepper motors for each node of the grid could provide very easy and fast control over height levels. The way stepper motors work is by dividing rotations of the motor into steps. This process allows the user to select the rotation amount to a much higher degree of precision. In our case, this step-based movement would provide a great way to raise and lower each of the nodes in our grid. The precision provided by the step motors is one of the biggest upsides to using it. Looking at the accuracy of these stepper motors, we can see a range of 3-5% non cumulative error per step. With the proper system, these motors should not miss any steps, meaning there won't be cumulative error. Sometimes, stepper motors may run into the problem of losing steps during fast rotations or under large load. This issue likely won't impact us as much due to the relatively low load/speed of our system. Precision and reliability are a huge reason for someone building this grid to use stepper motors. This along with the stepper motor's ability to hold its position, make it a very good option for the projection grid. Stepper motors are also very simple to use. Comparing the complexity of getting a stepper and a pneumatic cylinder to each raise the height is a night and day difference.

Stepper motors do not come without their own drawbacks. The main disadvantage in our case, is the price. Having 64 individual stepper motors for our 8x8 grid would see us in the range of $5-10 each. Paying 300-600 dollars for just the motors alone would likely not be feasible for our project, and this doesn't even include drivers or power supply for them. Another very big problem with stepper motors is the current draw. When holding their positions, the stepper motors will consume the full amount of current, increasing heat and lowering efficiency of the system. The last big issue with stepper motors is the potential to skip steps. Even with a low error, steps may be missed. In the case where these steps are missed, there is no protection of the system. No feedback from the motors means that skipped steps will be forgotten, which could lead to incorrect adjustments of height.

3.1.2.3 Row-Based DC Motor

DC motors allow for rotational movement that can be converted into a height adjustment. For the final method of controlling height, we are looking at a method using rows of these dc motors. Each row will be controlled by a dc motor attached to a threaded rod. This rod will be responsible for controlling all 8 nodes in the given row. Attached to the threaded rod at each node will be a gear rack that, once engaged, will be able to freely move up and down to the desired height. This method allows the user a more robust system, ensuring significant torque in a relatively small design. DC motors are good when you want a smooth and seamless change in height. Another benefit to using DC motors is the simplicity of changing speed. DC motors use PWM or changes in voltage to control speed at which the motor rotates. This is very good for our system where we favor precision over power. DC motors are also very efficient, ranging from around 70-90% energy efficiency depending on the system. With the correct additions to the system, DC motors are also very accurate. Having an encoder or some sort of feedback will ensure the precision of the motor is not compromised over time.

Some of the main concerns with DC motors, especially brushed ones, is that they wear down over time. This wear and tear on the motors could see a decrease in the lifespan of the system. For our motors specifically, it may not ever become a problem we encounter due to the short time scale this project will be in operation. Previously, we spoke about the need for added components, like encoders, to maintain precision of our system. This addition of components may add complexity to the design, not only in hardware, but software as well. This need for closed loop control will also come with additional costs. Each motor will require an encoder, which can cost anywhere from $10-30. Another issue with DC motors is the need for power when holding position. For our system, we may need to have a row unchanged holding position for a long time. Having to continuously provide power to this row will see high energy waste. The last concern with using DC motors is the added complexity of creating bidirectional movement. We will require the use of something like an H-bridge driver for each of the motors. This will add extra cost and complexity to our system.

	Pneumatic Cylinders	Stepper Motor	DC Motor
Power Consumption	Low	Moderate-High	Moderate
Complexity	Moderate	High	Moderate
Response Time	Fast (50-200ms)	Moderate (100-500ms)	Fast (50-250ms)
Output Force	High	Moderate	Moderate-High
Precision	Low (±0.5-2mm)	High (±0.01mm)	Moderate (±0.1-1mm)
Energy Efficiency	High	Low	Moderate-High
Cost	High ($50-200)	Moderate ($30-100)	Low ($15-50)

Table 3.3 *Height Control mechanism comparison*

3.1.2.4 Motors

For each rod, there needs to be a motor that drives the rotational force. This is a critical selection as the motors each need to be able to provide the torque necessary to drive the gear assemblies and push squares into their desired heights. Another aspect to be considered is the speed requirement for each rod so that the refresh rate reaches the minimum of 1 dataset per minute. A combination of factors affect how to select motors, including how much power is needed for each rod and the tradeoffs between torque and speed..

One of the most natural design choices for motors comes from deciding between an AC motor or a DC motor. Each has their own benefits, with DC usually providing more torque with less speed and AC more speed with less torque. The following sections will go into more depth on the different functions these two types of electric motors provide.

3.1.2.5 AC versus DC

Both AC and DC motors provide rotation using magnetic fields, with different kinds of motors for each power type. AC motors come in two main forms, synchronous motors and asynchronous. DC motors also vary, mainly into three categories: series, shunt, and compound. First, a look into the differences between AC and DC will be done, with further examination into which type will be best suited for driving the rod assemblies.

In AC motors, an AC current is applied to the stator, or stationary part of the motor. This creates the alternating electric fields critical to rotating the rotor, or the rotational aspect. Due to this, AC motors resist changes in its speed and require the help of a variable frequency drive (VFD) to control how fast the motor spins. VFDs add to the cost of these motors and introduce complexity to their operation. This resistance changing speed also adds benefits, however, making AC motors more resistant to changing loads not found when using DC. Resistance to load changes would benefit the 3D Projection Grid as the squares connect and disconnect from the motor driven rod. Another benefit to using AC motors comes from the lower costs associated with them. Due to lower cost in manufacturing AC motors and the widespread use of them, they are usually cheaper than DC motors. With the requirement of keeping the budget as small as possible, this is a desirable trait in motor selection. Downsides of AC motors include being less efficient than DC, due to using electromagnets instead of permanent ones. On top of this, more inefficiencies are added when considering slip in the motor. Slip is the difference between how fast the rotor spins and how fast magnetic fields are revolving. It is a crucial phenomenon to the function of AC motors resulting in producing torque, but at the cost of efficient power consumption.

DC motors on the other hand provide simplicity and functionality. Instead of electromagnets, they use permanent magnets and polarity to control the rotor. It is also very easy to control speed and direction with DC by altering the polarity and magnitude of the voltage being applied. Compared to AC motors, however, DC does not adapt to changing loads well. Care needs to be given when increasing or decreasing the load on these motors. They also have much shorter

lifespans than AC, especially brushed motors which require regular maintenance. This is less of a problem considering the lifespan of the 3D projection grid, but will ultimately help in deciding on which to choose.

	AC	DC
Torque Control	Great	Good
Speed Control	Medium	Great
Lifespan	Long	Low-Medium
Circuit Complexity	High	Low
Cost	High	Low

Table 3.4 *AC/DC Motor Comparison*

When considering the arguments for each type of motor, it becomes clear that DC motors are best suited for the 3D Projection Grid. A major factor in this decision is the requirement for both rotation directions. Each square must be able to be raised, and to be lowered, requiring both spin directions. Adding the reverse direction to AC motors is nontrivial, and adds to the complexity and cost. When comparing this to how DC motors handle reverse direction, a simple change of polarity of the input voltages, it becomes clear that the advantages of DC outweigh what functionality AC would provide. Another reason for deciding DC is that the AC motors would not be running at constant operation. The AC motors would be turning on and off which provides additional problems due to the current surge required to begin rotation. DC does not have this problem making the benefits of using it even clearer.

3.1.2.6 Permanent Magnet, Series, Shunt, and Compound DC Motors

There are four different types of brush DC motors. Brushless motors will not be considered due to the increased cost that increases longevity, which is not a constraint for the 3D grid. The four types of DC motors being considered are permanent magnet DC (PMDC) motors, DC Series motors, shunt motors, and compound DC motors.

PMDC motors function using stationary magnets positioned in the stationary stator. These magnets generate electric fields that spin the rotor via conductive poles. The more poles inside the rotor the more torque produced when the electric field is generated. PMDC motors are small, cheap, and efficient making them an attractive choice to power the rod assemblies. A major concern, however, is the amount of torque these types of DC motors can provide. Making sure they provide enough force to rotate and push the squares into place is critical.

With Series motors, a field winding is used to generate an electromagnetic field that interacts with the series connected armature winding. In this scenario, the field winding acts as the stator with the armature being the rotor. These two components are connected in series to make sure the current flowing through the field windings also flows through the armature, increasing the field strength proportional to the amount of current. This results in a high starting torque, the most desirable feature of this kind of DC motor. A main issue with series motors is how they behave under load, with the speed and torque depending heavily on the load, making it hard to predict and alter the loads on the motors. Due to this a series motor is not viable for the 3D projection grid as altering loads will be occurring, increasing complexity.

Shunt motors differ from series motors due to the field winding and armature being placed in parallel instead of series. This leads to an almost opposite behavior than series motors. Shunt motors are designed in a way to keep the current in the armature almost constant, even with varying loads. With the current being constant, the speed changes very little even when the load changes on the motor. These kinds of motors are used when constant speed is needed for varying loads. One downside is how the torque behaves on startup, with a much smaller startup torque than a series motor can provide. Shunt motors could be effective in driving the 3D projection grid, and are considered for use.

Finally, the compound DC motor is a combination of a series and shunt DC motor. Including field windings in both series and parallel, the main goal of this kind of motor is to increase starting torque, along with maintaining constant speed under varying loads. A compound DC motor does not as well as a shunt would in speed regulation, or a series would in starting torque, but balances between the two. A downside to this category of motors is increased cost that comes with the complexity and amount of components introduced. With the 3D projection grid not needing a very high starting torque, compound motors do not fit the use case as well as the other possible selections.

3.1.2.7 NFP-GM37-3429-EN Brushed Motor

The NFP GM37 Brushed Motor is a geared motor with a 37mm diameter gearbox. The NFP motor operates at either 12V or 24V[11]. This model of motor comes with an encoder that can be used to provide feedback on the motors position. This will be crucial for ensuring the motor can precisely adjust height to one of the four levels. Since the model is a geared motor, it will work very well in a low torque/speed system. This also means that we won't need to configure any torque reduction mechanisms to guarantee safe functionality. We see a moderate to high efficiency for this specific model.

One of the main benefits to using this model is its low power consumption when idling. When we were looking at the stepper motors, we saw them drawing their full current even when holding position. This is not an issue with this model of DC motor. The price of this motor is another thing that makes it desirable. Needing only 8 of them, we would only be looking at

around $150 for everything. With the lifespan of these motors being more than long enough, it would be very little cost after the initial purchase, if any.

Some of the issues seen by the NFP GM37 relate to wear over time and noise. Since the motor is brushed, we might run into some issues with the system breaking down over time. Replacement of brushes is not a large concern, as it is relatively simple and unlikely to be necessary for a good amount of time. The other issue is noise, which is something that comes with the fact it is brushed. Precautionary measures may be taken to mitigate this noise problem. For one, you can add dampening to reduce the noise around the motor. Another solution may be to add some capacitance across the motors. Another small problem with this model could be availability. This specific motor is sold by industrial suppliers, so it may be more difficult to get as opposed to a commonly used hobbyist motor.

3.1.2.8 Maxon EC 90 Flat Brushless Motor

The Maxon EC 90 Flat is a brushless motor specializing in precision. The motor itself has a 90mm diameter frame and is relatively flat, as the model name suggests. The brushless motor doesn't rely on mechanical brushes like the previous model, instead it uses electronic commutation. This makes it so control over the motor is done electronically, as opposed to mechanically. This method uses sensors and a controller to determine position and adjust rotations. This specific model of brushless motor uses Maxon's winding technology, which helps with the dynamic response of the motor. The Maxon EC 90 excels when looking at its ability to maintain efficiency and performance.

One of the main reasons to use the Maxon EC 90 is its reliability. Maxon's winding technology sees an iron-free rotor that provides a precise response. This technology also helps with common motor problems such as cogging. Oftentimes, DC motors will not rotate smoothly, but this winding technology helps eliminate that. Another benefit to using this model is its lifespan. For a motor of this type, you could expect an average lifespan exceeding 20,000 hours of continuous use. The efficiency seen by it is also very good. We see around 80-90% efficiency[12] for this brushless motor. There are many good reasons to use this model, but as we will see there is a very big problem with it.

The biggest problem with the Maxon EC 90 is its cost. Compared to other DC motors, the price of this one is significantly higher. If we wanted to purchase 8 of these motors for our grid, we'd be looking at no less than $1000. This is very significant when determining what is actually feasible. The pricing alone is enough to deter projects of smaller scale from choosing the Maxon EC 90. This motor also does not come with an encoder. This would mean we'd have to implement ourselves, adding to cost and complexity.

3.1.2.9 Pololu 37D Metal Gearmotor

The Pololu 37D Metal Gearmotor is the most widely used DC motor we've discussed so far. It has a 37mm diameter gearbox. As the name suggests, the gearbox is made from metal. The

Pololu gearmotor also comes with an encoder in the form of a 64 CPR encoder. This means 64 counts per revolution(with the potential to increase if in quadrature mode) will be used when determining position. The Pololu typically operates at 6V or 12V. Since this motor is very common among hobbyists, it has a lot of configurations based on what is needed.

When comparing the Pololu 37D to the other models, we see some things that are beneficial and some things that aren't. One benefit to this model is its build quality. Since it is a metal gearbox, we don't have to worry about the quality of it like we would with something made from plastic. Many of the qualities seen in the NFP-GM37 model are also seen here. Low idle power consumption is among these qualities. Another similarity we see is the low initial cost. These motors are not too expensive to buy initially, but do come with potential for maintenance in the future. This model does a good job at providing a simple way to control height. It relies on simple PWM to control motor function. Unlike the Maxon or NFP, this model is cheap and available. Since it is very commonly used, it should be no trouble to find one online for a decent price.

The Pololu 37D also has some drawbacks to look at. Low system efficiency is something to look at when using this model. We see around 50-60% efficiency. Compared to the NFP, we see a decrease in encoder resolution. With only 64 CPR[13], we may not have as precise of a rotation as we'd like. We also have a lot of noise in the Pololu 37D. Again, this can be helped with additional resources, but it is still something to take into consideration. Of the three, this model also has the lowest lifespan. The last problem with this model is how much heat is generated as a result of extended use. The moderate to high heat generation raises some questions about its usability in our system.

	NFP-GM37-3429-EN	Maxon EC90 Flat	Pololu 37D
Torque	Moderate	Low-Moderate	Moderate
Efficiency	Moderate	High (85%)	Moderate
Speed(RPM)	10-1600 RPM	2080-3200 RPM	10000 RPM
Complexity	Low	High	Low
Noise	Moderate	Low	Low-Moderate
Dynamic Response	Moderate	Very High	Moderate
Cost	Low ($15-20)	High ($150-250)	Moderate ($20-50)

Table 3.5 *Comparison table to DC Motors*

3.1.3 Motor Control

After deciding on a DC motor for our height control, we will need to now figure out a way to control these motors. For our specific system, bidirectional control of the DC motors is essential. We will need to find a way to reliably rotate the motor clockwise and counter-clockwise at a desired RPM. Along with the need for bidirectional control, we will need a way to control the speed of the motors. With multiple motors running simultaneously, it is crucial that we find an efficient way to do this. Upon doing research, we have decided that the best potential options are an H-bridge, a DPDT Relay, and a Solid State Relay Bridge.

3.1.3.1 H-bridge

The H-bridge is a circuit composed of 4 MOSFETs or BJTs that utilizes switching to control current flow. The 4 switching circuits are configured in an H shape, giving reason to the name H-bridge. By activating diagonal pairs of transistors, we are able to select current direction going through the motor. By switching the direction of current, we are able to switch the polarity of the voltage applied to the motor. DC motors will rotate in a certain direction based on the voltages applied to the 2 pins. This method for motor control is very commonly used for applications of a wide range.

The H-bridge has many benefits when dealing with motor control. One of the benefits is its ability to control speed and direction to a very precise degree. The H-bridge can accelerate or decelerate very smoothly, making for a seamless change in height. The H-bridge also excels at response time of switching. We can expect the control mechanism to respond in a time somewhere in the nanoseconds. It also benefits from its compact and simple design. The H-bridge is able to be used in any application, ranging from simple embedded systems to sophisticated control mechanisms. Knowing this, you can also expect it to be scalable for a wide range of power needs. Another positive of using an H-bridge is its low mechanical noise. Overall, the H-bridge is an excellent choice if you want a highly reliable, low-complexity, mechanism for DC motor control.

Although there are many pros to using the H-bridge, it does have some problems. When dealing with H-bridges at the lowest level, there is some complexity in regard to the circuitry. Understanding how switching circuits like MOSFETs or BJTs work may come with some difficulty initially. This is not likely to have too much of an effect on us due to the option of using H-bridge driver IC without needing to configure the circuit on our own. We also run the risk of a shoot-through failure. This occurs when two of the switching circuits on the same side are turned on at the same time. This could potentially destroy transistors, causing our H-bridge to be deemed useless. We also need to worry about heat dissipation when dealing with the H-bridge. In the case where our transistors are not very efficient or our PWM frequencies are high, heat dissipation problems will become evident. One of the last big issues with using an

H-bridge is the need for protective circuits alongside it. Using overcurrent or thermal protection circuits can help to prevent damage.

3.1.3.2 DPDT Relay

A Double-Pole Double-Throw(DPDT) relay is a switch with two switching circuits that can each connect a terminal to one of two outputs. It utilizes electromechanical actuation to function. The way it works is by separating 6-8 power terminals as two separate poles. This is where the double-pole from double-pole double-throw comes from. When the relay coil is energized, a magnetic field is created that will switch both poles at the same time. For forward control, we'd deenergize the coil so that the current is flowing forward through the motor. When energized, we will see the contact points switch through mechanical actuation. The terminals will now be connected in the opposite configuration of the deenergized version. This will result in the motor powering in the backward direction. The energizing and deenergizing of the relay coil will require a separate circuit used for control.

Using a DPDT relay could be a good idea if you want a simple design with the ability to handle high current. The contact based switching is very intuitive and easy to implement. Another positive is its ability to function without the need for a heatsink. The contacts don't generate much heat, so protective components are not likely to be needed. There is also very low voltage drop across the closed contact points. This means that power loss over these points is relatively low. There are also fail safes available when using a DPDT relay. We see that it is defaulted to normally-closed, meaning that it will allow flow of electricity in its resting state. This means that when it is deenergized, it can return to a closed state, and stop in the case of a failure. The DPDT relay also comes in at a relatively low cost considering the applications it can be used in. The relay also has the ability to switch many circuits at once, not being limited to use by a single motor.

The biggest disadvantage of using a DPDT relay is its lifespan. The mechanical lifespan of one of these can range anywhere from 100,000 to 1,000,000 uses depending on the strain put on it. This in itself is not terrible, but when you take into account the electrical lifespan being 10,000 to 100,000 cycles, you can see the potential problems. This short electrical lifespan comes as a result of the switching of inductive loads. These inductive loads can generate voltage spikes, damaging the circuit and decreasing lifespan. Slow switching speed is also a problem with using a DPDT relay. With the DPDT relay, there is around 10-50ms needed to make the new connection and around 5-15ms for the breaking of the previous connection. A very big problem with this method is the inability to control speed using PWM. For our projection grid, speed control isn't exactly essential, but not having it may cause some problems. There is also a phenomenon called contact bounce that we need to worry about. Multiple connections or disconnections may be caused for a period of time during the switching process. There is also a problem seen by these relays called contact arcing. This happens when electricity flows through a gap during breaking a connection. This can lead to an electrical arc and may lead to electrical noise. They are also susceptible to vibration interference from other components. Movement of

other components in our system, like that seen by the rotation of our DC motors, may lead to problems with the contact based relay.

3.1.3.3 Solid State Relay Bridge

The Solid State Relay Bridge is a solid-state circuit that utilizes 4 solid-state relays arranged in a bridge to control current flow. This control of current flow allows for the powering of electronics such as DC motors. The solid state relay bridges control of current flow allows for it to reverse polarity seen by the motor, reversing its direction. The setup of the SSR bridge is similar to that of an H-bridge, but instead of transistors, there are solid-state relay modules. Similar to the H-bridge, the SSR bridge is controlled by diagonal pairs of SSRs receiving control signals. When the first and fourth SSR are signaled, current will flow normally, and the motor will rotate forward. When the other pair of SSRs are signaled, we will see the motor rotate backward. The control signals sent to these pairs are electrically isolated from power of circuit using coupling and are typically 3-32V DC with 5-25mA of current. SSR bridges come in different types, with the option for MOSFET-based, TRIAC-based, and SCR-based.

SSR bridges do exceedingly well when it comes to lifespan. Due to the solid-state nature of the bridge, you can expect an almost endless lifespan from them. The noise associated with SSR bridges is another reason you may choose this over competitors. Both mechanically and electrically, the noise seen by these models is significantly lower than others. The switching speed of these are also very fast. Compared to mechanical relays, solid-state relays are much faster. We also do not have to worry about contact bounce. When looking at an option like the DPDT relay, we see a problem where bouncing may occur and lead to problems. This is not an issue for SSR bridges. Using an SSR bridge, we can also handle a lot higher current than if we were to use another option. Repeatability over the course of its life will remain when using an SSR bridge.

When looking to purchase an SSR bridge, you can expect to spend more than you would on a mechanical relay. This higher price point comes with the functionality of the mechanism. Using an SSR, we will need to make sure we have a heatsink. High heat generation makes this an essential addition to the system. We also see a reduced efficiency when using a solid-state relay bridge. Some of the different versions of these SSR bridges handle speed control differently. Using TRIAC-based, speed control will be very limited through PWM. Some of these SSRs also require constant current flow to remain on. This can become a problem when adding many to a system. These SSR bridges are also significantly larger than the other options, again making it difficult for a system that requires many of them.

	H-bridge	DPDT Relay	SSR Bridge
Bidirectional	Yes	Yes	Yes

Control			
Speed Control	Yes (PWM)	No	Limited
Switching Speed	Very Fast	Slow	Fast
Efficiency	High	Very High	High
Complexity	Moderate	Low	Moderate
Heat Dissipation	Moderate	Low	High
Lifespan	Long	Limited	Very Long
Protection	Built-in	External Required	Built-in
Cost	Low-Moderate ($2-15)	Low ($3-10)	High ($15-50+)

Table 3.6 *Motor controller table*

3.1.3.4 L298N H-bridge Driver

The L298N H-bridge driver is a dual full-bridge driver, coming in the form of a full driver module or IC. The L298N deals with applications of high voltage and high current. This specific model of H-bridge is able to be used to control two motors at once. The L298N has two full H-bridge circuits on a single chip. Each of the bridges built into the circuit are responsible for controlling one of the DC motors. Direction of motor rotation is determined by setting the input pin pairs to high/low or low/high. In the case where we set the first input high and the second input low, we would see the first motor go forward. If we were to reverse this, we'd see our rotation for motor one reversed. The same idea is used for the second motor, except using input three and input four. Speed control is also achieved easily for the L298N. Simple application of PWM signals can be used to control rotation speed of the motors being driven. Along with the four input pins, there's also two two enable pins. These enable pins can be used to control our motor speed. These are defaulted to full speed unless otherwise adjusted. A PWM signal of 0-255 will determine the speed at which the motor rotates while being driven by the H-bridge driver[14].

One of the main reasons to choose the L298N is its ability to control two motors simultaneously. This will mean that we will only require half the amount of drivers to fully control the motors in our system. The pricing of these H-bridge drivers is also a reason you'd select them. Typically ranging from $2-5, these fully operational modules come with a lot of functionality for very little cost. The ease of finding these modules for purchase is another reason they make a good fit. Due to the popularity of the L298N, it is easy to find from a number of online sellers. Simplicity is another thing the L298N excels at. Having a relatively simple design helps reduce circuit complexity when dealing with many at once. These are also very robust to damage or errors over a long period of use. Mistakes as a result of wear-and-tear are not likely to occur when using the L298N. When looking at implementation of these into our system, it is important to take into account other components needed with it. The L298N is great because it doesn't require external components like flyback diodes, which are already built in.

When dealing with the L298N, we need to look at some of the downsides. One big one is its efficiency. Since it is based around BJTs, the design comes with problems of high voltage drop. This results in a good amount of wasted power as heat. This heat generation is significant and must be monitored to ensure no serious problems can occur. To aid in this heat generation, it is recommended to add a heatsink. This will ensure proper heat dissipation, and will allow for full functionality at an optimal level. The simplicity of the system is an upside, but that also means the L298N lacks in some areas when compared to newer models. MOSFET designs are typically a lot more efficient than that of BJTs. These MOSFET designs also solve another problem typically associated with the BJT H-bridge, slow switching. Slower switching speeds could be noticed when dealing with the simple L298N design.

3.1.3.5 TB6612FNG Dual Motor Driver

The TB6612FNG Dual Motor Driver is a driver used for controlling DC motors. The TB6612FNG uses MOSFETs as opposed to BJTs. The switching of these MOSFETs is done by adjusting voltage levels. The TB6612FNG has 7 pins used for controlling two DC motors in our system[15]. Of these seven, four are input pins. Two of these are each used for determining direction of one of the two motors. This is similar to what we have already seen with the L298N. Where the TB6612FNG differs is with the other pins. Two PWM pins are used to control the speed of either of the motors connected. Sending signals to these pins will allow you to control speed in a much cleaner way than you would with more complex models. The TB6612FNG also has a standby pin(STBY). This will allow you to control whether the motor driver is in operation or at rest.

Compared to other models of motor drivers, the TB6612FNG has great efficiency. Low On-resistance means that power loss is minimized heavily. As a result of this, heat generation remains relatively low. The design of the TB6612FNG is also very compact, making for a good choice when size concerns are present. Like the L298N, the TB6612FNG has dual motor control capabilities. This is very useful when dealing with many motors, because it will effectively cut the amount of motor control drivers needed in half. This dual motor driver also comes at a

relatively low cost. The price per unit for one of these can range from $3 to $6. This model of driver also comes with the benefit of multiple operating modes, which is something we didn't see from the L298N. Modes like short brake, clockwise, or stop help with simple control in an intuitive design. Built-in protection is another thing that separates the TB6612FNG from other motor controllers. In the case where temperatures get too high, there is thermal shutdown and undervoltage lockout protection.

With the addition of more control, comes more complexity. Comparing the pinout of the L298N to the TB6612FNG, we see the difference in simplicity. The seven pins used by the TB6612FNG make it slightly more complex to work with. These models of motor controllers also suffer from the problem of availability. Due to them not being as widely used, there are less options for purchasing online.

3.1.3.6 DRV8871 Brushed DC Motor Driver

The DRV8871 DC Motor Driver is a MOSFET based driver used to control a single motor. The design features an 8-pin setup. When looking at motor controllers, the DRV8871 is among some of the higher end versions. It contains four N-channel MOSFETs with very low on-resistance[16]. This low on-resistance is similar to that seen by the TB6612FNG, meaning efficiency is also very high. The difference can be seen when looking at the capabilities of the two models. The high current and voltage handling means the DRV8871 can be used for a much wider range of applications. One of the reasons the DRV8871 handles high current and voltage so well is the integrated current regulation. This built-in regulation will continuously monitor the system to ensure current doesn't get too high. Options for speed control can be achieved using PWM signals on the input pins.

The DRV8871 Motor Driver differs from other models significantly when looking at the control pins. Having only two input pins, the design is very simple. Since there is only one motor being driven, we only have two input pins to worry about. When one input is high and the other is low it will drive the motor to rotate one way, and when reversing the signals it will rotate the other direction. This simple design makes it very easy to get setup and running. Speed control using PWM is also very simple when using the DRV8871. Driving a PW signal of varying frequencies will cause the motor to run at the desired speed. Protection is also built into the DRV8871 like it was with the TB6612FNG. Problems like undervoltage lockout or overcurrent are protected against when using the DRV8871. These are also built to last in harsh environments, so wear from simple use is not likely to be an issue. Like we saw with the TB6612FNG, low on-resistance helps maintain high efficiency of the system.

The biggest problem with the DRV8871 is the fact that it can only be used to control a single motor. Both the other options we have looked at have dual-motor capability, so using the DRV8871 would require purchasing twice the amount of controllers. With this, the price per unit for these drivers is almost double that of the other two models we have already looked at. Heat dissipation is another thing we have to worry about. High current flow can lead to high power generation, requiring the need for a heatsink. The simplicity of the input pins can also be looked at as one of the downsides to using it. Not having a dedicated enable pin can require some

outside the box thinking when implementing. The DRV8871 is likely more powerful than we'd need in our specific application.

	L298N	TB6612FNG	DRV8871
Switching Circuit	BJT	MOSFET	MOSFET
Heat Dissipation	Poor-Moderate	Excellent	Good
Availability	High	Moderate	Limited
Lifespan	Moderate-High	High	High
Complexity	Moderate	Low-Moderate	Low
Response Time	Moderate	Fast	Fast
Ease of Use	Easy	Moderately Easy	Moderate
Efficiency	Moderate (60-70%)	High (90-95%)	High (90-95%)
Cost	Low ($2-4)	Low-Moderate ($5-10)	Moderate ($5-12)

Table 3.7 *H-bridge comparison table*

3.1.4 Locking Mechanism

When working with our height control system, we are going to need a way to ensure the correct gear racks are being raised. We want to find a way to disengage squares that are already at the desired height while raising ones that are not. Having a single threaded rod for each row will mean that we want to ensure each node is only being raised if we have not yet reached the desired height. A couple of the ways we found include a solenoid pin, servo linkage, and electromagnetic clutch.

3.1.4.1 Solenoid-Actuated Pin

The solenoid pin actuator is the most simple of all the locking mechanisms we are going to look at. It works because of the properties of the electromagnetic coil built into it. When powered, electricity will flow through the coil, creating a magnetic field, and pulling the pin to the coil. The solenoid pin has a spring on the side it rests on when powered off. This spring is able to

make sure the pin stops when being magnetized as well as return it back to its resting position once powered off. This whole system works because of the pins' attraction to the magnetic field when powered on. You can imagine its function to be very similar to that of a lock on a door. Instead of locking the door physically you'd power it on, and when powered off, we can open that same door.

This mechanism is very simple to not only understand, but to use as well. The main concerns of this include proper alignment, sizing, and decomposition. Alignment in a system where parts are constantly in motion may prove to be difficult. Having the gear rack and threaded rod disengage at whatever height will require alignment is taken care of properly. The sizing will also be important considering we do not want the pin to be too large to not fit, but also too small to where the locking is not sturdy. Constant use of the solenoid pin will also come with the troubles of wear and tear. This may not be initially evident, but over time it will definitely play a factor.

The solenoid pin excels in areas like speed, ease of use, and fail-safes. The switching of states seen by the pin happens in milliseconds, meaning that we won't have to wait a significant amount of time for disengagements to occur. As soon as it is powered off, the magnetic field instantly disappears, causing the spring to push the pin back in very little time. The pin solenoid pin will automatically extend and retract when powered on and off which is good because that means no complex setup. The mechanics of the solenoid pin have great fail-safes in the way that when they are powered on they extend and when powered off they immediately retract. There is no need to worry about partial extension because of this fact.

3.1.4.2 Servo-Actuated Linkage

The servo-actuated linkage is another method researched to help with locking the gear rack in place when not disengaged. The way it works is a servo motor rotates a shaft that holds some sort of lever. The rotation of the lever will engage or disengage the mechanism. The lever is a type of mechanical linkage that will help us decide when to allow the gear rack to move and when not to. Some of the issues with this method include complexity, cost, speed, and ease of use. The actual placement of the level will need to be engineered in a more thoughtful way as a result of the shape of it. This method also requires the use of a motor along with the linkage. This will cost around $10 and will be needed for each of the modules. Another problem is the speed at which this change will take place. The cheaper servo motors will often take longer than we'd want from a locking mechanism. The last problem arises when looking at the difficulty of use. There is not a simple way to make the linkage open and close the connection with the gear rack, programming will be required.

The benefits to using this type of mechanism are its precision and adjustability. The use of servo motors allow for us to make exact changes to turning radius to ensure it moves exactly as expected. Not having to rely on binary states of on or off will allow for a lot more control of our system. The other benefit, adjustability, ties in with precision. Being able to adjust exactly how much rotation we get can allow us to perform tasks that wouldn't be otherwise possible with other mechanisms.

3.1.4.3 Electromagnetic Clutch

The electromagnetic clutch is a method of locking in which electromagnetic force is used to engage or disengage the gear rack. The basic construction consists of a rotor connected to a motor that will spin while the motor is on and an armature which will only spin when the clutch is in an engaged state. There is also an electromagnetic coil that will create the electromagnetic field when powered on. When disengaged, springs will push the armature away from the rotor, leaving a small air gap. This gap will stop the movement of gear due to the lack of friction. Some of the troubles associated with this method are heat, power consumption, lock stability, and wear and tear. Long term usage will see problems with maintaining acceptable temperature, which can lead to reduced lifespan or even shutdown. Power consumption is a similar issue seen. Generating an electromagnetic field for a long period of time can use a lot of power and overtime it adds up. This can be anywhere from 10-40W depending on length of usage. Wear and tear will also be seen from extended use. Due to its requirement for friction to move the components together, these surfaces can break down to the point where they don't function properly. The last issue we see with this method is the lock stability. Under a large load, the locking mechanism may exhibit slipping and that could be fatal for the system.

The electromagnetic clutch is a good method if we were worried about the stress on the gear rack. The gradual buildup of torque means that it won't be putting too much pressure on the gear of our system. One of the flaws of the clutch was its lock stability. This instability also helps us in the case of protecting our components. Since it may slip, it will ensure that the components connected won't be destroyed and the motor won't stall. With this method we are also provided more variation in the speed at which the system will perform. The last benefit to this method is the ability to keep the motor spinning while also disengaging the gear rack. Since the motor is able to spin while there is no friction between rotor and armature, we can continue movement through it to other modules while disengaged.

	Solenoid-Actuated Pin	Servo Actuated Linkage	Electromagnetic Clutch
Speed	Very High (10-50ms)	Moderate (200-500ms)	Fast (50-150ms)
Power Consumption	Moderate	Low	High
Complexity	Low	Moderate-High	Moderate
Precision	High	Very High	High

Size	Small	Medium-Large	Small
Cost	Moderate ($20-100)	Moderate ($30-100)	High ($100-200)

Table 3.8 *Locking Mechanism comparison table*

3.1.4.4 Magnet-Schultz G TC A 070 X43 A01

The Magnet-Schultz linear solenoid is a DC high-performance solenoid. This model has 7 frame sizes(40-100mm in units of 10mm) and multiple stroke sizes for each. The Magnet-Schultz solenoid is high-performing, which is very useful in applications where reliable actuation is essential. The solenoid operates at 24V DC. The Magnet-Schultz delivers 63N of continuous force at the start and the end of the stroke with potential to reach up to 200N. It is a heavy-duty, industrial grade, solenoid actuator that specializes in high force applications.

The Magnet-Schultz solenoid makes a good choice for our system due to its reliable and powerful actuation. The solenoid being high-performance ensures that we can safely engage/disengage without worry of constant failure. Its large size and heavy weight mean that it will be robust to wear and tear over time. Its lifespan is another thing to take into consideration. The Magnet-Schultz is able to actuate for at least twice as long as rival solenoids on the market, with a range of 2-10 million cycles before breakdown may occur. This model does very well at providing steady and strong force. As mentioned before, 63N of continuous force is seen, with much higher values seen intermittently. This solenoid is also rated for class F insulation, meaning it can withstand up to 155 degrees celsius of heat. The build quality and reliability are some of the main reasons you'd choose the Magnet-Schultz solenoid.

A big problem with the Magnet-Schultz solenoid is the power consumption associated with it. When looking at the datasheet, we see the average power for all possible sizes. Looking at the 70mm specifically, we see an average power of almost 40W. This is not the only thing that concerns us. The current draw for this model is also something we need to take into consideration. The model continuously draws 1.5A of current regardless of if it's in use or not. This can become a problem with many solenoids all working together simultaneously. This all adds to the Magnet's problem with high heat generation. The response time for the Magnet-Schultz is also relatively slow. Looking at the 70mm again, we see that the actuation time for engaging is almost 160ms and the time for disengaging is around 100ms. In a system like ours, it is not essential to have fast response time, but it does help significantly. The final, and main, problem with the Magnet-Schultz solenoid is the price. You can expect to spend $100 or more on a solenoid of this level. Needing multiple for our system, this may become a significant problem.

3.1.4.5 Guardian Electric T8X16-C-12D

The Guardian Electric T8X16 solenoid is a tubular pull solenoid. The frame of the Guardian solenoid is 25.4mm(1 inch) in diameter. The stroke length for the Guardian solenoid is 16mm. The Guardian Electric operates at 12V DC with a continuous duty rating. The holding force output by the Guardian Electric varies based on which state the solenoid is in. It has a holding force of 8.5N, 17N at the stroke start, and 29N when fully engaged during continuous duty. The current draw of this specific solenoid is approximately 0.4A when power consumption is 5.3W. Mounting of the solenoid is achieved using the L brackets that come with it. The stroke of the Guardian Electric features a conical plunger tip useful for connections.

Some of the main upside to using the Guardian Electric is its reliability at a moderate cost. The solenoid is able to produce up to 30N, making it effective in the application of our system specifically. When looking to purchase the Guardian Electric, you can expect to pay around $50 per unit. This is a little expensive for a solenoid, but its functionality makes it worth the higher price point. We see a relatively low current draw from the system, only amounting to around 424mA during continuous duty. The Guardian Electric also excels at response time for actuation. The solenoid can expect to respond as fast as 30ms. This solenoid also has a longer lifespan than some of the cheaper alternatives. This model of solenoid will do well in a system where you want to ensure actuation in a relatively quick manner, while also not spending too much money. The relation of output force to power consumption also shows how the system's efficiency is significantly better than other models.

While the Guardian Electric solenoid does reliably function, it has some potential drawbacks. When comparing the Guardian to other solenoids, we see that it may be more than what’s needed for the purpose of our project. The high force output is likely much more than what will actually be needed for a system like ours. With intermittent current draw reaching around 1.3A, it comes with potential for problems when using many of them.

3.1.4.6 JF-0530B Pull Solenoid

The JF-0530B is an extremely low-budget DC solenoid with very easy configurability. Looking at the sizing of the solenoid, we see a relatively compact design. The dimensions of the solenoid are 30 x 16 x 15mm with a 10mm stroke. The JF-0530B works by having a coil body, plunger, and spring working together to actuate. Upon being energized, the generated electromagnetic field pulls the plunger in towards the body. This is where the pull in pull solenoid comes from. The plunger, now retracted into the coil body, is pulled with around 5N of force. When deenergizing powered off, the built in spring will cause the plunger to be pushed outwards back to its original position. Since the stroke size for the JF-0530B is 10mm, the plunger will now extend 10mm from the body.

The main benefits of the JF-0530B can be seen when looking at its price and simplicity. Looking at the price per unit of one of these models, we see it is only around $3-8. In a system where you may need many of these, like ours, it is important to keep this low price point in mind. The availability of models like these also help when deciding if it’s the correct choice. Being that this

model is so popular, it is easy to find and order through commonly used sites like Amazon or AliExpress. The simplicity of its design also helps when choosing whether or not it is the right fit. The solenoid features a simple push-pull non-polarized design with no need for any type of external mechanisms required. It also allows for multiple operating voltages, with the option for 6, 12, or 24V DC. The JF-0530B also has relatively low power consumption, with around 3.6W. The rated current for the JF-0530B is 300mA, which is low enough for our specific application. The response time for this model is also very fast. We can expect a response time of around 20ms when using the solenoid.

Although the JF-0530B has some major benefits, it also has some limitations. Due to its cheaper price, we can expect the lifespan to be significantly lower than that of the more expensive models. We can expect a lifespan of around 50,000-100,000 cycles. Although this is more than enough for our application, it may cause problems later down the line. The low force of this solenoid could also be a problem for some applications, but for us it will likely be more than enough. The main disadvantage for this model is its inability to continuously hold position. Since it is on the cheaper end of solenoids, it has a max continuous run time of 2-5 seconds. What this means is that we are limited to the length of time we can power the solenoid. This is a very significant limitation when looking at solenoids. In the case where our system requires holding for longer periods of time, this may become a problem.

3.1.4.7 Adafruit #412 Push-Pull Solenoid

The Adafruitfruit #412 Push-Pull Solenoid is a highly affordable solenoid that can both push and pull. The Adafruit solenoid is a good middle ground between cheap and industrial solenoids, making it perfect for our application. This solenoid differs from the ones we have already looked at by implementing push or pull availability. Depending on orientation of the solenoid, it can be used as either a push or a pull. Like the other models, it has an armature and spring for returning to resting position. When the solenoid is energized, the plunger will be pulled toward the coil in the center of the device. This will act as a pull for the side with the spring and a push for the side without it. This is how the adafruit #412 solenoid is able to act as both a push and a pull solenoid.
Adafruit is a very reputable brand when it comes to electronics design. This means that we can expect relatively quick delivery times and good support in the event of some problem arising. This solenoid also benefits from its dual-functionality. Acting as both a push and a pull solenoid, it has a much wider range of applications than most the other ones on the market. The design having a captive armature will also be essential for our system. Not needing to worry about adding any protection against loose parts will make dealing with the Adafruit #41 solenoid significantly easier. All the necessary components are included with the solenoid, so no need to worry about acquiring springs or mounting brackets when purchasing it. The overall quality of the Adafruit #412 is one of the upsides to using it over other affordable solenoids.

The biggest problem with the Adafruit #412 is something we have already seen with the JF-0530B, intermittent duty. Common amongst cheaper solenoids, intermittent duty makes it so

that the solenoid can not be continuously powered for long. Typically, these solenoids can run for around 5 seconds before needing to de-energize. The push and pull functionality is good, but it comes at a cost. Short stroke length caused by needing to go through the entire solenoid will render it useless for some applications requiring a longer stroke. These cheaper solenoids are always going to have problems with quality when compared to more expensive models. While the Adafruit #412 is good, it may degrade significantly over the span of its life.

	Magnet-Schultz G TC A 070	Guardian Electric T8X16	JF-0530B	Adafruit #412
Force	High	Moderate	Low	Low
Type	Push & Pull	Pull	Pull	Push & Pull
Heat generation	Moderate	Low	Very Low	Moderate
Response Time	Moderate (100-150ms)	Fast (30-50ms)	Fast (20-40ms)	Fast (20-50ms)
Lifespan	Very High	Moderate-High	Low-Moderate	Moderate
Size	70mm x 167mm	25.4mm x 51.8mm	30mm x 16mm x 15mm	30mm x 17mm
Stroke Size	8mm-15mm	16mm	10mm	5.5mm
Cost	High ($100-150)	Moderate ($50-100)	Low ($3-8)	Low ($7-12)

Table 3.9 *Solenoid comparison table*

3.1.4.8 Addressable Selection Devices

With the chosen row based motor system, we need the ability to activate the columns being adjusted using an addressable solenoid system. To better explain, the motors will rotate 8 shafts along the x-axis, with this solenoid selector system being addressed via rows along the y-axis. This will allow for addressing all 64 columns to be able to raise/lower them as desired. There are different solutions for adding an ability to select a row on the y-axis, each adding a different amount of complexity and functionality.

3.1.4.9 74HC138

A 74HC138 is a demultiplexer that has a total of 16-pins, with 3 selector pins controlling which of 8 different outputs is chosen. This fits well into the use case of 8 different rows needing to be controlled, with a low power consumption of only 80µA to power the part. Another consideration is how much power can be output, as it will need to drive a row of small solenoids/locking mechanisms. In the datasheet, the 74HC138 can handle a maximum output of 20mA, and supports an output voltage the same as the supply voltage, with the recommended maximum of 6V [8]. A maximum output current of 20mA is not very high, so output current must be carefully considered to ensure that each row of locking mechanisms receive the power it needs.

Another desirable feature this component has is the ability to be used as a decoder. This would be helpful if the 3D project grid needs more than 1 output to be active. Instead of only one output being active the chip can be configured to include any or all of the 8 outputs active. This dual functionality would be useful and help make the project more flexible and resistant to late changes in the project.

3.1.4.10 74LS138

Having the same pins and functionality as the 74HC138, this demultiplexer has a different construction that differs in voltage/current conditions. A 74LS138 operates at lower voltages between 4.75 and 5.25V, with low-level output current at 8mA. Due to the requirement of having to drive a small selection device, 8mA constrains what parts can be used by a large margin. This is offset with a lower power consumption, however greater error tolerance is more desirable.

3.1.4.11 CD4051

Similar to the 74HC138, this is an 8-channel demultiplexer, however this allows for analog and digital signals. With 16 pins, the supply voltage can vary between -10V to 10V DC. However, after further research this type of demultiplexer is not designed to supply current and so would be limited without further circuitry. Also the functionality the CD4051 provides is analog options, which are not desired for the current project.

3.1.4.12 74LS154

This chip is a 16 output decoder that uses 4 selector lines to choose which outputs are active. While each row will only need to have 8 different selectors, by using a decoder with 16 outputs it would be possible to control two rows of 8 columns with only one chip. The added complexity of an additional selector line is offset by reducing the number of chips, so if two 3-8 demultiplexers were used, this increases the input lines to 6 rather than the one decoder with only 4 inputs. Supply voltage is less forgiving, with the chip being able to handle between 4.75V and 5.25V. The decoder can also only handle up to 0.1mA of input current when at max input voltage. As a more sensitive chip it will be less resilient to fluctuations in voltage and current.

3.1.4.13 CD4514B

A 24-pin decoder, this chip for TI adds additional features that could help achieve the project goals. One main feature is the latches inside the chip allowing it to store the data from the previous strobe transmission. Another improvement is in the supply voltage, which can handle large values up to 18V and can range as low as 3V. Output voltage ranges from 5V-15V depending on the supply voltage. With the amount of voltages inside the component, the input current has a maximum of around 1 μA [9].

3.1.4.14 Selection Circuit

Once the addressable solenoid can be selected using the digital control logic from the previous section, a system must be in place that uses the controlling logic to output a voltage and current that can drive the operation of the solenoid. It was decided that using one of the above selection devices, a circuit would be added to control the power being applied. This circuit will be used in close conjunction with the solenoids and will be a part of the solenoid system.

This circuit would be a good fit for the addressable solenoids, as the outputs of a demultiplexer or decoder could be wired to the input signal line, allowing for power to be applied to the solenoid or turned off. Because the above logic devices aren't designed to drive electronic devices, using this circuit will allow the use of cheaper, less specialised logic ICs and ultimately simplify project design.

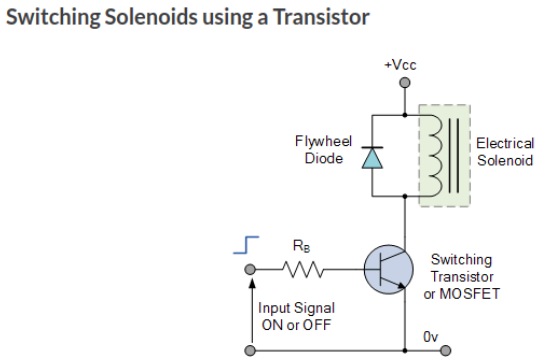


Figure 3.1 *An image of a switching circuit used to select a solenoid device using control inputs [10].*

	74HC138	74LS138	CD4051	74LS154	CD4514B
Output Channels	8-Channel	8-Channel	8-Channel	16-Channel	16-Channel
Total Pins	16	16	16	24	24

Supply Voltage	2-6V	4.75-5.25V	3-15V	5-5.25V	3-15V
Supply Current	2μA	8mA	1μA	16-32mA	1μA
Demux and Decoder	Yes	Yes	No	Yes	Yes
Output Type	Digital	Digital	Analog/ Digital	Digital	Digital

Table 3.9 *Addressable Solenoid table*

3.1.5 LEDs

LED selection is critical for the 3D projection grid, as an LED will need to be associated with each height varying column to achieve 3 dimensional representation of data. Some considerations when selecting the type of LED and LED circuits include being able to address each node individually. This is a hard requirement and must be met to consider the project successful. However, there are multiple ways to achieve this behavior, with different circuits and protocols that vary in strengths and weaknesses.

3.1.5.1 Type of LED

To achieve the varying colors needed to display data, single LEDs are not sufficient. This leaves RGB LEDs and RGBW. RGBs have three diodes for red, green, and blue, resulting in four pins (including a shared anode/cathode pin) necessary to control and operate the color of the LED. By altering the light intensity of each color, the desired color across the spectrum can be produced using three channels. RGBW LEDs include an additional fourth diode that displays pure white light. Useful in scenarios where a true white is needed, such as environmental lighting and high resolution displays, this additional functionality comes at the cost of a fourth channel needing to be controlled. RGBW LEDs also usually have higher power costs when compared to the simpler RGB ones. A detailed comparison between two common RGB and RGBW LEDs will be done to help in deciding which will be best for the 3D Projection Grid.

3.1.5.2 FD-5WSRGB-A/C

A commonly used component, this RGB LED is a 5mm Lamp LED and comes in two versions, one with a shared common anode and the other a shared common cathode. Each of the three

color diodes in this component has a forward current maximum of 20mA, with the color white at max brightness resulting in a maximum of 100mA per LED. Also, the forward voltage required to operate is 3.2V for all colors to function [3]. With only four pins and a small operating current per RGB LED, it would be feasible to include this component or similar into the final design. Given the full range of colors along with simple functioning any 5mm RGB would provide all needed functionality.

3.1.5.3 5050 PLCC-8 RGBW LED

This RGBW LED in the QT-Brighttek PLCC Series has an additional white diode allowing for better white LED behavior. Similarly to the previous RGB, each color diode has a maximum current draw of 20mA, resulting in only an additional 20mA of current per LED for a total of 120mA. Another feature is the ability to choose between warm-white, natural white, and cool white varieties [4]. Being able to choose a better looking white light is not in line with the project goals, and at the cost of increased complexity and wiring in most situations the RGB LED would be the clear choice.

3.1.5.4 LED Control

Deciding what kind of LED to use is not enough to meet the requirements of being addressable, so LED ICs must be considered to achieve this. There are many kinds of LED driver circuits, including single dataline circuits, two-wire datalines, and more advanced industry standard LED packages. For the scope of this project, the industry standard protocols and ICs such as LDP8806 and GS8208 will not be considered. Two types of LED control will be examined, single dataline protocols and clocked two-wire protocols.

3.1.5.5 Single Dataline Protocols

Single dataline protocols involve the use of one dataline to control and address all the LEDs in the system. A data stream is sent into one input, with the output of a single RGB/RGBW LED being fed into the next data-in pin of the following LED. The ICs that drive the LEDs use different protocols to determine the behavior and desired color/brightness of the light. It is common to call each of these RGB LEDs a pixel, and this term will be used in the following sections.

3.1.5.6 WS2811

This IC is used to drive a single pixel and send altered data to data-out. There are 8 pins, with 3 of them attached to the red, green, blue LEDs, along with the two data pins and other functional pins. Each IC will use the first 24 bits in the stream to set the different colors and then send the

remaining data to the next circuit. This data can be sent at two different speeds, 400kbps and 800kbps, corresponding to a low-speed and high-speed mode [5]. Using this type of LED driver circuit would meet the functional requirements and is a viable selection for the 3D grid. One downside, however, is that it uses three separate LEDs per pixel instead of a single LED with RGB pins.

3.1.5.7 WS2812

Building upon the WS2811 IC, the WS2812 introduces a single LED per pixel and also reduces the operating voltage from 12V to 5V. This allows for ease of operation and also increased amounts of pixels that are chained together. Given the amount of LEDs needed for the project (a minimum of 64), this is a significant improvement from the WS2811. Another improved feature is a reduced amount of pins needed to operate, needing only 6 as opposed to 8. Some downsides are a vulnerability to reverse polarity, and a small margin of error for operating voltage fluctuations. These issues are dealt with in the newer WS2812B that will be considered next.

3.1.5.8 WS2812B

An improved version of the WS2812, this type of LED driver introduces reverse polarity protections, higher variance in operating voltage, with a range in the power supply voltage between 3.5 - 5.3 volts. This is a significant improvement from the WS2812, which only had 1V of acceptable error for the power supply voltage. This type of IC also only requires 4 pins to control which will help reduce the amount of connections needed from the 3D grid. To use this protocol, an 800kbps communication speed is required and must be available in the selected MCU. Given the additional features and lack of complexity, WS2812B styled ICs fit the scope and requirements of the project.

3.1.5.9 SK6812

A different style of WS2812, LED driver adds the capability of white light and brighter LEDs. White LEDs are not required for the project, but a brighter light may be required depending on materials chosen for the height varying columns. It is worth being considered with a selection only being if higher brightness is needed, or if more colors are desired to extend the amount of data that can be displayed.

	WS2811	WS2812	WS2812B	SK6812
Addressable	Yes	Yes	Yes	Yes
Pins	8	6	4	4
Voltage	12	5	5	5
Current	80mA	80mA	80mA	100mA

White	No	No	No	Yes
Control Speed	400kbps/800kbps	800kbps	800kbps	800kbps

Table 3.10 *LED protocol comparison table*

3.1.5.10 Two-Wire Protocols

Two-Wire LED drivers include an additional clock line to precisely control the timing of when data is read into the color registers. This allows for faster communication using the external clock and smoother transitions between colors. It's for this reason two-wire protocols are known as SPI-based and their biggest strength is the ability to increase data rates from 800kbps by orders of magnitude. Following are some of the most used two-clock ICs that will be considered for use in this project.

3.1.5.11 APA102

Similar to WS2812-SK6812, APA102 runs at 5V and incorporates a 5050 LED chip system. The key difference is the clock line as mentioned before. This allows the LED to be ran by a separate clock, increasing the refresh rate from 800kbps to up to 20MHz. While this decreases flicker and allows for faster animations, the 3D Topology Grid does not require performance in these areas. Usually at a higher cost then normal WS2812B ICs, the APA102 increases performance but in the wrong areas for this project.

3.1.5.12 SK9822

Made by the same company who created the SK6812, this type of IC driver does the same function by providing a larger acceptable range of supply voltages along with improved timing tolerance. Another key feature is the ability to handle low brightness better than the APA102, which is not a desirable feature in this project. A more compelling case for two-wire protocols, it still doesn’t meet the design goals that one-wire protocols do.

3.1.5.13 GS8208

With an increased supply voltage back to 12V, this IC improves data reliability by including a data backup line that is resistant to dead pixels. In the previous two-wire ICs, a dead pixel would mean all pixels after would not be able to work. By slowing the clock time and adding the backup data line, dead pixels do not affect all pixels for the GS8208.

	APA102	SK9822	GS8208
Speed	Up to 20MHz	Up to 20MHz	Up to 20MHz
Failsafe	No	No	Yes
Voltage	5V	3.3-5V	12V

Table 3.11 *LED two-wire protocol table*

3.1.6 Languages and Repositories

3.1.6.1 C

C is a straightforward programming language known for its simplicity and ease of use. It is primarily used for low-level programming and lacks many object-oriented features. Because of its direct interaction with hardware, C is widely used in embedded systems and semiconductor development. The language also offers a large collection of libraries, allowing programmers to solve problems efficiently without needing to write every function from scratch. One of C's main advantages is its ability to provide direct access to hardware components such as registers, memory, and flags, enabling precise control over hardware connected to a C program. Since C must be compiled, its source code is translated into assembly and then into machine code so the computer can execute operations at the hardware level. Although C is an older language and lacks modern paradigms like object-oriented programming and inheritance, it remains an excellent choice for low-level programming and embedded software development—making it especially useful for projects that involve multiple microcontrollers.

3.1.6.2 Java

Java is a programming language that strongly emphasizes object-oriented and class-based design. It functions both as a language and as a software platform, making it highly portable and easy to run across different devices. Java code is first compiled into machine-independent bytecode, which is then executed on a system using the Java Development Kit (JDK). Unlike C, Java is built around object-oriented principles that simplify coding and organization. Core concepts such as classes, objects, encapsulation, inheritance, polymorphism, and abstraction are seamlessly integrated into the language. These features offer major advantages for team projects, including better code organization, reusability, scalability, and collaboration. Code organization helps separate and secure data while keeping it easy to locate. Reusability allows developers to define an object once and reference it multiple times throughout the program. Scalability enables easier debugging and independent testing of code components. Finally, Java's structure supports collaboration by allowing teams to divide work among different classes and modules. Overall, Java is an excellent programming choice due to its portability, integrated software platform, and strong object-oriented foundation.

3.1.6.3 Python

Python is a high-level programming language known for its versatility and powerful capabilities. Its simple syntax makes it beginner-friendly and easy to learn. With a vast collection of libraries, Python enables developers to build algorithms efficiently using built-in functions. It also excels in performing complex statistical analyses on large datasets, making it ideal for fields such as machine learning, data science, automation, and software development. In addition, Python can generate visualizations like graphs, charts, and 3D plots, which are essential tools for data analysis. Another major advantage is its ability to handle automation, scripting, and machine learning tasks—capabilities that set it apart from many other languages. Like Java, Python supports object-oriented programming, allowing developers to use classes and objects to keep code organized and modular. It incorporates key OOP concepts such as inheritance, encapsulation, and abstraction, along with features like method overloading and overriding for added flexibility. Overall, Python is an exceptionally powerful and adaptable language that offers unique advantages not easily matched by other programming languages.

	C	Java	Python
Language Level	Low-level	High-level	High-level
Ease of use	Moderate	Moderate	Easy
Memory	Manual	Automatic	Automatic
Speed	Very fast	Moderate	Slow
Libraries	Many but less than java/python	Extensive	Very extensive
Object-oriented support	Limited/None	Full	Full

Table 3.12 *Language comparison table*

3.1.6.4 Github

GitHub is a repository platform built on the Git version control system, designed to allow multiple users to share and collaborate on code simultaneously. It can host both private and public open-source repositories and is widely recognized for offering free access to public projects. GitHub operates using a principle called version control, where users can “clone” a repository to make their own modifications—a process known as branching. Once their changes are complete, they can upload or “merge” them back into the main repository without disrupting the existing codebase. This system makes teamwork efficient and organized, enabling developers to collaborate on shared projects while independently contributing their own updates without interfering with one another’s work.

3.1.6.5 Bitbucket

Bitbucket is a Git-based repository platform that enables teams to collaborate on software projects in real time. Similar to GitHub, it allows a group to maintain a shared repository for source code and use version control to manage edits and updates. However, what sets Bitbucket apart is its built-in integration with Jira and Trello. Jira is a project management tool that tracks bugs, helps streamline debugging, and organizes development tasks by identifying and prioritizing issues. It also supports efficient project management by monitoring progress and managing dependencies within the workflow. Trello, on the other hand, provides a highly visual approach to project organization, using boards, lists, and cards to assign and track team responsibilities. It also allows teams to manage members, due dates, attachments, and checklists in one central location. While Bitbucket offers these powerful integrations and collaboration features, one notable drawback is that it typically requires a paid subscription.

3.1.7 Rotating Connections

Options for allowing electrical connections that can be wired in a stationary-to-rotating fashion are limited. The first part that comes to mind is the widely used slip ring. Built for continuous rotation, it's a good option for wiring components on a spinning motor shaft that have to be connected to a non-rotating pcb or driver. Another option to be considered is using a hollow motor shaft and feeding wires through the center. The wires would remain stationary while the shaft rotates around them. At first it seems plausible, however when considering connecting the wires to the electric parts on the shaft it becomes clear that they will run into the same issue of twisting together and limiting rotation. Other options include rotary connectors with flexible ribbons, wireless power, and putting all electronics isolated on the motor shaft. Wireless power seemed promising, but the additional complexity and cost to add wireless signaling as well was weighted against the simplicity of a slip ring and was decided against. A similar decision was made with isolating the electronics on the shaft, as we would need to add wireless signals as well due to the requirement of communicating with the MCU. For these reasons the slip ring was decided on and in this section the different slip rings available will be considered for the 3D topology grid.

Taidacent High Speed Ball Conductive Slip Ring 12 Wire 2A

This slip ring is a through hole mount with 12 wires available to be used through the connector. It has an operating voltage of 240 Volts AC/DC, with the current being 2A. At this power level, it more than satisfies the need of the 3D topology grid. Also the 12 wires provided should be enough to control the eight solenoids needed, especially if including a shared ground connection. Another feature is the ability to meet up to 200 rpm. Given the high torque motors being considered, this gives well over the needed requirement to meet the matrix refresh rate. The overall size of the slip ring would fit well with the motor and projected size of the project.

QL-LINRUN 3 Wires 10A 22mm

Similar to the Taidacent slip ring, the 3 wire QL-LINRUN increases the amount of current and voltage able to be supplied, being able to handle up to 10 A and 440V AC/DC. With the increase

in power comes a cost in the amount of wires running through the slip ring. This would not be able to directly power each of the 8 solenoids, so additional selecting circuits would be needed to choose between the solenoids. The increase in power doesn't just affect the amount of wires, but also the cost with an additional 10 dollars per part as compared to the Tiadacent slip ring. Only very high power requirements would bring this slip ring to be used in the 3D grid.

Tiadacent 18-Wire Road Cap Slip Ring

A second Tiadacent slip ring that can provide 18 wires in comparison with the previous 12. Along with wire count the max rpm also increases, to a total of 250 rpm. However the price increases by 10 dollars, and the rpms are not a limiting factor in the project. It will be considered if the amount of wires needed for the solenoids increases or unexpected issues come up requiring more wires. A good slip ring with large wire count is rare, so this Tiadacent will be heavily considered for use in the final projection grid.

	Taidecent 12-Wire	Taidencent 18-wire	QL-LINRUN 3-Wire
Voltage	240 Volts	240 Volts	440 Volts
Current	2 Amps	2 Amps	10 Amps
RPM	200 RPM	250 RPM	0-300 RPM
Torque Required	0.03 N.M	0.04 N.M	0.03 N.M

Table 3.13 *Slip ring comparison table*

3.2 Part Selection

3.2.1 MCU

When deciding which microcontroller to go with, there were a couple options that could've worked. We ultimately decided on using the ESP32. The ESP32 has all the functionality that we will need at an affordable price. The ESP32 is widely used for a wide range of applications, making it the top contender for our system. WiFi and Bluetooth availability is a big reason we decided to go with this microcontroller.

When comparing the ESP32 against other similar microcontrollers, it can definitely hold its own. Its low price for such a powerful board makes it very popular among those interested in electronics design. For our project, we will need a number of important peripherals, all of which are on the ESP32. Having the option of multiple communication protocols like UART, SPI, and I2C is another reason we chose this board. Since we want to control the speed of our motors using PWM signals, it is essential for our board to have that featured.

The ESP32 does well at being an affordable solution to our problem, while still providing all the necessary functionality. With 26 GPIO pins, the ESP32 provides more than enough pins for us to work with. The ESP32 also consumes very little power, which is important to take into account for a project like ours.

3.2.2 Height Control

After going through each of the potential methods for height control, we decided on the Row-Based DC Motor. The choice came down to DC motors or stepper motors and we ultimately decided the DC motor would be best fit. We chose to disregard the pneumatic cylinder implementation due to its complexity when dealing with equal disbursement of air pressure throughout the system as well as its total cost. When analyzing the stepper motors, we saw the potential for problems with current draw throughout the system. Since the stepper motors would be required for each node, a lot of current would be drawn constantly. Since our system will have regular height adjustment, it is important to make sure we aren't consuming too much power. DC motors also provide a very simple method for controlling our height. The models of DC motors we are going to compare are the NFP-GM37-3429-EN Brushed Motor, Maxon EC 90 Flat Brushless Motor, and the Pololu 37D Metal Gearmotor.

Looking at the three different models of DC motor, it was easy to rule out one of the options. The Maxon EC90 Flat was something we initially thought could be a possibility. It was extremely high performing but also consumed more power than the others. The Maxons price point is where it lost all potential for use. The NFP and Pololu were relatively similar in their price and ability. Ultimately, it was decided that the NFP was the better of the two choices.

	NFP-GM37-3429-EN	Maxon EC90 Flat	Pololu 37D
Power Consumption	~2.4W	~5W	~2.4W
Operating Voltage	12V DC	24-48V DC	12V or 24V DC
Voltage Range	6-14V	18-30V	3-12V
Current (No-Load)	200mA	540mA	300mA
Current (Nominal)	~0.5-1A	6A	~1A
Efficiency	50-60%	84%	55%
Output Torque	19.5 Nm	444 mNm	0.45 Nm

(Nominal)			
Weight	300g	600g	170g
Speed (Max)	100 rpm	5000 rpm	630 rpm
Operating Temperature	-20 to 85 C	-40 to 100 C	-10 to 60 C
Cost	$15-20	$150-200	$20-50

Table 3.14 *DC Motor part selection table*

Looking at the comparison between the NFP and the Pololu, you can see they are very similar in certain areas. For our system we took into account things like current draw and that's where we decided on the NFP. While the Pololu is a good choice, it is likely more powerful than what we'd need. The Pololu had a speed of 600+ RPM. This was much more than we needed or even wanted for our system. The rest of the comparisons made between the two saw them pretty much level with each other. In the end, the NFP was the one that came out on top.

3.2.3 Motor Control

We have gone through the research and decided that using an H-bridge driver will be the best bet for our project. Dealing with H-bridges, we don't have to concern ourselves with the electromechanical or solid-state complexity associated with the other options for motor control.

Now we can look to compare the different types of H-bridges and decide on one for our use. The ones we will be looking at are the L298N Dual Motor H-bridge driver, the TB6612FNG, and the DRV8871.

	L298N	TB6612FNG	DRV8871
Motor Supply Voltage	5-46V	4.5-15V	6.5-45V
Logic Supply Voltage	4.5-7V	2.7-5.5V	3.3V or 5V
Operating Voltage	7-35V	6-12V	12-24V

Package Type	Multiwatt15 (through-hole)	SSOP24 (SMD)	HTSSOP-8 (SMD)
Peak Current	4A	3.2A	3.6A
Output Current (continuous)	2A	1.2A	2.1A
Efficiency	60-70%	90-95%	90-95%
Max PWM Frequency	40 kHz	100 kHz	250 kHz
Operating Temperature	-25 to 130 C	-20 to 85 C	-40 to 125C
Weight	2g	0.1g	0.05g
Cost	$2-4	$5-10	$5-12

Table 3.15 *H-bridge part selection table*

Being that the LN298N uses a BJT design, we were initially not sure whether it was the right choice. The slower speeds seen by using BJTs was not significant enough for us to be fully deterred. The affordability of the L298N made it very enticing to choose. Being that it is so popular among motor control hobbyists, there is a lot of documentation to look at when something is unclear. The design is relatively simple and easy to understand, especially when you pair it with the fact there's so much information for it online. It is built to last more than enough time for our purposes.

After looking through all the options, it was decided that we will choose the L298N. One of the main reasons we decided on this was its dual motor capabilities. The TB6612FNG also had dual motor capabilities, but the L298N was ultimately chosen due to its simplicity.

3.2.4 Locking Mechanism

After researching the potential methods for engaging/disengaging the height control, we have come to the conclusion that we will use the solenoid actuated pin. This solenoid will allow us the ability to easily choose whether the node will be able to be moved or not. The other options had some potential for use, but ultimately were not the right choice for us. Looking at price, functionality, and complexity, we were able to determine this method for locking being the best fit for our specific needs. The solenoid's simplicity is among the main benefits seen. When looking into specific models of solenoids, we found the best four to be the Magnet-Schultz G TC A 070 X43 A01, Guardian Electric T8X16-C-12D, JF-0530B, and the Adafruit #412 Push-Pull Solenoid.

	Magnet-Schultz G TC A 070	Guardian Electric T8X16	JF-0530B	Adafruit #412
Power Consumption	33W	5W	3.6W	3W
Operating Voltage	24V DC	12V DC	6, 12, 24V DC	12V DC
Current Draw	1.4A	424mA	300mA	300mA
Resistance	17.4 Ohms	27.3 Ohms	40 Ohms	40-48 Ohms
Force	~200N	~30N	5N	5N
Weight	1860g	113g	31g	39g
Cost	High ($100-150)	Moderate ($50-100)	Low ($3-8)	Low ($7-12)

Table 3.16 *Solenoid part selection table*

In the end, it came down to the JF-0530B and the Adafruit #412. These two were the most feasible price-wise so it wasn't hard to come to this conclusion. Comparing the two, we see they are very similar in many ways. Needing a push solenoid, it was decided that we were going to go with the Adafruit #412.

The Adafruiit #412 Push-Pull Solenoid was the perfect mix of affordable and functional. Since we weren't going to need much force output, the Adafruit's 5N[17] was perfect. With the lowest power consumption, it was relatively easy to decide on this solenoid.

3.2.5 LEDs

Comparing between the single and double wire LED driver circuits, it was established that single protocol works best for the 3D grid. This is because we do not need fast refresh rates or clock control of the LEDs to meet the project requirements. Out of the single dataline ICs, the simplest and most functional selection is the WS2812B LED circuit. It is the easiest to control and provides minimum current requirements, which is a concern when 64 LEDs need to be active at the same time. Another reason for choosing WS2812B is the ability to be controlled with only 4 pins for the entire strip of LEDs. This reduces the amount of GPIO pins needed from the MCU which is a constraint for how many peripherals we can control.

A big decision was choosing between the RGB WS2812B and the RGBW that the SK6812 provides. Having a solo white LED that allows for more clear white light not only adds an additional color, and hence more variable datasets, but lowers the amount of current required to display white. Instead of setting all 3 LEDs in the RGB pixel to maximum, costing 80mA (20mA per LED), we would only need to set the white LED (20mA). Due to cost constraints, however, it was decided that the budget was better put into the solenoid and motor systems of the 3D topology grid.

3.2.6 Language and Repositories

After researching which programming language would be the best fit for our project, there were a few things to take into consideration. We wanted a relatively easy to use language, with a lot of documentation already online. C, Java, and Python are all very popular languages with their own benefits.

Python is easier to use than C or Java, but it suffers when looking at the speed. For us, the choice came down to either C or Java. As C is a low-level language, we thought it'd be best for our specific application. When looking at C documentation online,you can see that it has tons of predesigned systems that can work with our goals. The lack of object-oriented programming is not likely to be too much of a concern for our specific application.

Along with the language, the repository is something we have to decide on prior to really setting everything up. When given the choice between Github and Bitbucket, there are some differences to consider. While Bitbucket is Git-based, it differs from Github slightly. The option for integration with Jira or Trello makes it a great tool with extensive organizational features. Ultimately, we decided to go with Github. The reason for this choice came down to wanting a good tool for working together on software, without all the extra fluff. Using Bitbucket could be beneficial, but we decided that the integration of organizational tools was not fully necessary. This is why we decided to use Github in the end.

3.2.7 Slip Ring

After considering three different kinds of slip rings, it was decided to go with the Taidecent 18-wire slip ring. Although more expensive than a 12 wire, the 12 wire would only work if all solenoids shared a ground connection. However, due to the selection circuit used to control each solenoid, 16 wires would be needed with each having separate voltage supplies and voltage grounds. Another reason to choose the 18 wires over 12 is redundancy and error resilience. Parts ordered don't always come fully functional, and with the complexity of slip wires it would be useful to have 1 or 2 wires extra in case of failure in the component. The increase in risk of selecting the 12 wire over the 18 wire slip ring gives more credence to accepting the additional cost and ordering 18-wires. This is compounded with the fact that if one slip ring fails and

another is needed, the cost difference becomes closer than just choosing 18-wire slip rings from the start.

3.3 Power Supply Unit

3.3.1 Power Supply

The power supply of the 3D grid is very important to make sure all of the components work as intended. Not achieving adequate voltage to electrical components is a direct cause of inadequate performance. This section will go over the different considerations when it comes to powering the 3D grid. Since the grid is stationary, it is not restricted to only a battery to power it. This is very important due to the safety requirements for operation of the grid. Due to the power consumption concerns and the economic concerns of batteries, it is unlikely that a battery will be used but it will still remain a consideration.

3.3.2 Safety Requirements

One of the requirements for the 3D grid is not being concerned with safety when it comes to power and air compression. For this section, power consumptions will be considered. It is very important to first define what is considered as “safe”. For voltage, below 50 volts (V) DC and 35V AC (RMS) is typically considered safe for humans. For current, it is undesirable to reach 5-10 milliamperes (mA) into a human as this can already cause a painful shock to humans, and a greater amperage than this can already begin to cause muscle spasms and difficulty breathing, while reaching 100-200 mA is often fatal. Receptacles in the U.S deliver high voltage, 120V AC (60 Hz), which is far above the safety threshold for humans in terms of voltage. It is then very important for the grid to be “isolated” from mains. “Mains” are simply another term for the voltage that is found in the typical receptacle found in any home in the U.S. Isolation from the mains means that although a device might be powered by the receptacle (mains), the voltage found there is not the same voltage in the circuit of the device, meaning it is safe to touch and operate without worry of shock. The strategy we will use to isolate the circuit is to gradually step down the voltage in between each phase of the power delivery system until each component sees its desired voltage and current demand.

This is where SELV, FELV, and PELV circuits are useful. These three kinds of circuits are low-voltage systems designed to reduce the risk of shock. All three of them operate at what is considered to be “safe” voltages and ensure any current that enters a human if touched, will not cause any harm. The main differences of the three are how they are isolated, grounded, and protected.

3.3.3 SELV

SELV stands for “Safety Extra Low Voltage”. These circuits are fully isolated from higher voltage circuits and are not connected to earth. They are designed so that even in the event of a

fault, no dangerous voltage can appear on the low voltage side. SELV circuits are isolated via double or reinforced insulation or equivalent protective separation.

3.3.4 FELV

FELV (Functional Extra Low Voltage) circuits are very similar to SELV. Although FELV does operate at extra-low voltage levels like SELV, they do not meet the same isolation requirements that SELV and PELV do. FELV circuits are not isolated fully in the way that SELV or PELV are, meaning that if a fault occurs, there could be an instance where dangerous voltages can be seen on the FELV side.

3.3.5 PELV

PELV (Protective Extra Low Voltage) are most similar to SELV. They meet most of the same voltage and isolation requirements, but they *do* allow a connection to ground for functional or protective reasons. Although they might not be as isolated as a SELV circuit, they are still safe to touch. PELV circuits are separated from higher voltage through basic insulation along with protective measures such as more insulation or a safe, low resistance path to the earth.

3.3.6 Power for Grid

SELV is likely to be the best choice for our 3D grid in order to meet the safety requirements. Since we are considering using power from mains in our grid, it is important that the circuit is isolated or separated because very dangerous voltages can appear if not isolated. Moreover, most of our electronics are not going to directly need the kind of voltage seen from receptacles. Furthermore, most of the electronics in the grid *need* much lower voltage where devices such as regulators will be necessary. For example, most ESP chips need just 3.3V DC. If these devices, whether on a development board that already includes a regulator or alone, see the kinds of voltages from receptacles, they will be instantly destroyed.

Component	Voltage (V)	Current (A)	Power (W)	Total in Watts (Worst Case Scenarios)
ESP (1)	3.3	0.26	0.858	188.058
Motors (8)	6	3.5	168	
LEDs (64)	5	0.06	19.2	

Table 3.17 *Power table for system*

Total Current Draw (In general & not worst case)			
ESP (A)	LEDs (A)	Motors (A)	Total (A)
0.3	3.84	11.2	15.34

Table 3.18 *Current Draw table*

It is also very important that with either option, we are able to implement it within a PCB. The PCB will essentially be where all the power is distributed in the grid. For batteries, the typical idea would be to get a battery holder for AA or AAA batteries and assemble as needed. It is important to note that this project will be needing far more power than AA and AAA batteries can provide (which will be touched on a little later). For the kinds of batteries that could be used to successfully build this project, we would need to connect the PCB to an outside source of power. If we are to use mains receptacle power, then we also need to connect from an outside source of power. For the mains receptacle, it would be important that the power from the receptacles is converted to DC power because all components on the PCB will be DC.

3.3.7 Receptacle vs. Battery Considerations

The 3D grid does not have any movement requirements which brings up the debate as to whether to use a battery to power it or directly from mains. There are many considerations when this debate comes up and this section will sum those up. The points that will be considered are: power & current demand, runtime, safety, economics, and space & weight.

For battery considerations, we are only going to consider batteries that give 12V or 24V. We are not going to consider using smaller batteries that can be put in series to achieve 12V/24V, as it is far more inconvenient to do so. Batteries that outright produce 12V/24V can be considered in figure 3.3.

To begin with, power & current demand. The 3D grid's operation will rely *heavily* on the efficiency of the simultaneous delivery of power to its components. It is very important that all components see the proper voltages and currents necessary for operation to avoid system failure or damage. The grid will use one main power supply whether the battery or the mains and then split the power down to three separate sub-units. These units are the motors, the LEDs, and the ESP. The minimum voltage from the main power is 12V to be able to support the other three sub-units. The other three units will have regulators to step down voltages as necessary for the components. For example, the ESP takes 3.3V input, so connecting this directly to 12V would destroy the ESP.

Batteries				
	12V			
	Lead Acid		Lithium-Ion	
	Flooded-Cell	Absorbent Glass Mat	$LiFePO_4$	NMC (Nickel Manganese Cobalt)
Capacity (Ah)	105	4	7-200	5-30 Ah
Size	14in x 6.75in x 10in	3in x 3in x 4in	19in x 6.77in x 9.44in	6in x 2in x 2in
Weight	66 lbs	3 lbs	50 lbs	3 lbs
Cost	$150	$20	$30-$500	$60-$260
Considerations	Very big, very heavy	Smaller size, but smaller capacity	Very big, very heavy, price ranges vastly	Expensive for bigger capacity
Runtime hours (Worst case)	6.64556962	0.253164557	0.443037975	0.316455696
24 V				
	Lead Acid		Lithium-Ion	
	Absorbent Glass Mat		LiFePO4	NMC (Nickel Manganese Cobalt)
Capacity (Ah)	9-50 Ah		20-120	5-30 Ah
Size	5in x 5in x 8in		19in x 8in x 10in	9in x 6in x 4in
Weight	20 lbs		~50 lbs	~5-6 lbs
Cost	$60-$280		$70-$500+	$100-$400
Considerations	Expensive and big		Very expensive and very big	Quite Expensive
Runtime hours (Worst case)	0.569620253		1.265822785	0.316455696
AC/DC Converters				
	12V		24V	
Size	3in x 1.6in x 4.5in		6in x 2in x 2in	
Weight	0.2 lbs		2 lbs	
Cost	$22		$55	

Table 3.19 *Battery comparison table*

The three power sub-units will come from different layers in the PCB. It is most likely that we will use 24V, but many of the calculations are going to be with the bare minimum of 12V so ensure the project works even on the worst case scenarios. The first layer is for motors. We are choosing 6V motors to work with. The next layer is for the ESP and this layer will take 24V from the AC/DC converter to a linear voltage regulator, outputting 3.3V and 500mA. Lastly, the final layer is for the 64 LEDs. This layer will likely take the 24V from the AC/DC converter into a synchronous PWM buck controller. As seen in figure 3.2, the total current demand from the 64 LEDs is 3.84A, 11.2A from the motors, and 0.3A from the ESP for a total of 15.3A *when components are performing their functions*. Using the current demand, we can begin to consider the runtimes of batteries and how many we need, how much they are, and how big they are for the other categories of considerations. For mains power, we do not need to take this into consideration as we can assume a consistent and reliable power delivery. The worst case scenario total power for the grid is 188W (where the motors are stalling and take 3.5A), in a normal case, the power output is 87W. In the next considerations, this will be taken into account. In general, both batteries and receptacles can achieve this amount of power, although it is important to note that there is no runtime for mains, only batteries.

Runtime is the second consideration when deciding how to power the grid. For demonstration purposes, it is very important to ensure the grid will run for enough time to accurately demonstrate all of the features of the grid. One of the requirements of the grid is to be able to shuffle between at least 1 data matrix per minute with a stretch goal of 10 matrices per minute. With this information, the closer we get to our stretch goal of 10 matrices, the more power we are likely going to consume because the MCU will be executing instructions more frequently and the motors will be operating for a longer time overall for our stretch goal than for the base goal. The ESP32 also tends to take more current during high frequency bursts or when using Wi-Fi capabilities. From the tables seen in figure 3.3, the runtime using batteries ranges from 15 minutes to almost 7 hours. The 7 hours comes from the 12V flooded-cell battery. This is quite the outlier because the second longest runtime is around an hour and 20 minutes. Although there aren't any strict requirements on how long this system would need to be on for, it is better to have something that can run for a longer period of time. All in all, the runtimes for batteries work for this project for two reasons.

First of all, the minimum runtime is around 15 minutes. For the demonstration of this project, that is more than enough to show all of the functions of the project and ensure it was built properly and can execute all necessary functions. Second of all, since this project is being built as a sort of "theoretical" market product, the runtimes are all quite acceptable aside from the Lead-Acid AGM (absorbent glass mat) battery. Moreover, another one of the requirements is module expansion. This means that an identical grid must be able to be plugged into the first grid and operate as a larger composite grid. If the grid is to be powered via receptacles, then it would be in the best interest to ensure the second grid is also powered by the first one since power from receptacles is reliable, consistent, and will not run out in the same way a battery would. If the grid is powered via battery, it would be in the best interest to power the second grid via its own battery because increasing power draw from the first grid would mean both grids would run out of battery much faster. If another module were to be connected to the first grid, it would essentially cut the runtime in half if we use one battery for both. If we use a second battery for the second grid then this would add to the expenses of the project which is not ideal, but will be covered later. Lastly, it is important to mention that for mains power, runtime is completely not a worry. They provide a stable and reliable power source that can handle both the original grid and also the module expansion.

One of the requirements of the entire project is safety. Safe voltages have already been discussed previously but here are the considerations when it comes to choosing whether to power via battery or receptacles. Most batteries are safer than receptacles so this is very important. The batteries in consideration in figure 3.3 are SELV. This meets the safety requirement meaning no harmful voltages or currents will be present. As previously discussed, mains receptacle can provide *lethal* voltages which is why it is very important to isolate the circuit of the grid from mains and make it SELV. For flooded-cell lead-acid batteries, acid spills and hydrogen gas are the main worries. These batteries contain liquid sulfuric acid that can cause burns to humans and damage other equipment. The hydrogen gas is released during charging which can be explosive in enclosed spaces. It would be important and recommended for users to wear personal protective equipment (PPE), only charge the batteries in well-ventilated areas, and keep baking soda or a neutralizing agent nearby for acid spills. Next, for AGM lead-acid batteries, they are sealed and spill-resistant which makes them safer than the flooded cell type. There is a short

circuit risk due to high current capabilities, and overcharging can cause venting of hydrogen. Moving onto lithium-ion batteries, the $LiFePO_4$ batteries are some of the safest chemistries. This does not mean they are immune to short circuits and overcharging. Lastly, the NMC lithium-ion batteries have a higher energy density but are less thermally stable than the $LiFePO_4$. They can enter thermal runaway when overcharged, damaged, or exposed to high heat. Thermal runaway is when a temperature increase causes a feedback loop of chemical reactions that generate even more heat and flammable gasses.

All in all, these are the safety considerations for batteries, the safety considerations for mains receptacles are less. For mains receptacles, there are no chemicals involved making them the best option for chemical safety. Mains receptacles have better built-in safety. This is because when buildings are built, engineers take safety into heavy consideration and make sure there are multiple safety measures in place such as grounded outlets, breakers, and ground-fault circuit interrupters (GFCI). This does not mean that the grid is exempt from having its own safety measures because the grid's circuit should still be isolated from the harmful voltages from mains receptacles. If the project is done with mains receptacles, we would purchase an AC/DC converter that comes as those wall plug-ins with the big box connected after. This would isolate the circuit and complete all the requirements of SELV meaning there is nothing to worry about after. Due to the power calculations the output voltage of that converter is going to likely be 24V which is safe.

Economics are a huge consideration for this project as it should likely be as cheap as possible for the group. Starting the lithium-ion batteries, which are usually better and safer, are more expensive than the lead-acid. Starting prices go from $30 but can skyrocket to $500+ for high quality batteries. It is important to note that the cheaper lithium-ion batteries are not going to last as long. Figure 3.3 shows that when comparing 12V and 24V, the 24V batteries are going to last longer on average. Since we likely plan on using 24V, this means that our battery selection is going to be the cheapest possible. This means a sacrifice in runtime along with space & weight which will be touched on later. Moving onto lead-acid batteries, these are usually cheaper, especially 12V size which can be seen in figure 3.3 starting at $20. The 24V sizes start at $60 for a runtime of only half an hour. This is where the AC/DC converter that plugs into the wall comes in. As seen in figure 3.3 the adapters are much cheaper. Since all these do are convert power from AC to DC, it is not necessary to find a "high quality" converter because it will never run out of power, it just converts. When it comes to a battery, the consideration of an expensive battery is simply to make it last as long as possible for runtime purposes. A great way to see these economical comparisons is with $/sq inch or $/Ah. Putting the money spent in comparison with the lifetime can show easily and clearly why mains are a better choice. For example, the cheapest 24V $LiFePO_4$ battery usually comes in at around $70 for 20Ah. Doing the math, we get $3.5 per ampere-hour. Doing this comparison with mains is different as we spend $20 initially and get as much time as desired. Overall, the AC/DC converter is a much better choice as we get unlimited power for the $20-$50 spent. In terms of batteries, we would be paying for a limited amount of operation time for the grid.

The final consideration is space & weight. This determines how our grid is going to look. It should not be overly bulky or heavy. There are two routes this section goes. Whether the battery fits inside the grid, or if it does not. It is important to note the grid will be 8in x 8in x 4in. The

main purpose of a battery is to put it inside the grid to make everything look professional. If we have to connect wires from the grid to the battery, then there is no purpose for the battery because if we have to connect wires from the grid anyways, we might as well do it to mains and get unlimited power whereas if we connect outside the grid to a battery, we get limited power. Beginning with the 12V lead-acid batteries in figure 3.3, the smaller AGM batteries would be able to fit inside the grid. The downside of this is that these batteries are the ones that last the least, a little over 15 minutes. Economically, they are not too bad starting at $20. The size of the flooded-cell lead-acid batteries in figure 3.3 are the typical size for these kinds of batteries, meaning they won't fit inside the housing of the grid. This eliminates those from consideration. Moving onto 12V lithium-ion batteries, the $LiFePO_4$ batteries are quite big as well. The size shown in figure 3.3 is a typical size and they are not even close to being able to fit in the housing so they are also eliminated from consideration. The NMC lithium-ion batteries would fit in the grid. At the worst case, they provide 5 Ah (ampere-hour). This gives .325 of an hour or around 18 minutes of runtime. That is definitely enough runtime to demonstrate the project, but when taking into consideration the module expansion, this number drops to 9 minutes which is not very good for a quality project. The 24V lead-acid AGM batteries would possibly fit in the grid. Figure 3.3 says they are typically 5in x 5in x 8in. This is an approximation among multiple online examples of batteries. It is likely possible to find a 24V AGM battery that could fit in the grid so it can be considered. Coming in at $60 and a little over a half hour in runtime, the AGM battery is not too bad. Moving onto the 24V lithium-ion battery, the $LiFePO_4$ battery will not fit in the housing so it is removed from consideration. Lastly, the NMC 24V lithium-ion battery might fit in the grid due to approximations so it will be considered. Its worst-case scenario at 5 Ah also provides around 18 minutes of runtime. This battery runs into the same issue as the NMC lithium ion 12V battery. These batteries are more expensive than those coming in at a minimum of around $100. Lastly, the AC/DC Converters. The 12V converters come in at around $20 and provide no worries of runtime or capacity, which is why those considerations have been removed from the table. The 24V AC/DC converter comes in at $55. Overall, the AC/DC converters are the best choice here. Many of the batteries were eliminated due to size, and the ones that were not do not compare to the converters.

Finally, comparing all the batteries and AC/DC converters, the final choice is the AC/DC converters. It would not be a smart idea to purchase a battery that costs the same amount as the AC/DC converters but limit us to a runtime. Moreover, these converters are very lightweight, their maximum being around 5 lbs. In terms of A/$, W/$, weight/$, and size/$, they are easily the best choice.

3.3.8 Battery vs Mains Summary

All in all, we are going to use mains power for the grid. After calculating power & current demand, both of the options would work. Mains would provide more reliable power that would last for as long as necessary while batteries, in the worst case, would do just enough to get past the demonstration. Furthermore, as it comes to this product being a theoretical market product, it is better to allow for unlimited use without worrying about battery life and recharging. All in all, mains narrowly beats batteries in this consideration due to its reliable and consistent power. With safety, both of these options are equally viable. Although it might seem like mains can be far more dangerous, there is a lot more protection in place. Mains are protected from faults and

shorts via breakers in the panels supplying them, and the AC/DC converter is a plug-and-play option that has gone through rigorous testing and standards to ensure it is a product that can be put on the market for sale. For batteries, the concerns are much less about lethal voltages and currents, and more about chemical burns and gasses. Batteries are chemical reactions that convert chemical energy to electrical energy. If these batteries get chemicals on any users or group members, there would be a huge safety hazard. Furthermore, when it comes to promoting this project as a product, it would surely fail to pass many codes and standards in the U.S for what can be sold to the general population. Overall, batteries and mains land equal in this department. Both of them have safety issues that can be easily dealt with. Moving onto economics, mains wins easily. A 24V AC/DC converter is priced at around the same as the cheapest batteries. On top of that, there is no $ per watt or ampere comparison between the two because a converter wins easily. Although after paying the $50 for the 24V converter the user would still have to pay for an electrical bill due to the 3D grid, the user would have to pay a very similar amount every time they need to recharge a battery. Moreover, the battery would run out much faster than the time it took to charge it so the user would be recharging more frequently than using the 3D grid which means more billing. This was all taking into consideration the cheapest batteries. If the better batteries, which last longer, are taken into consideration, then the price can surge up from $100 all the way to $500+. Overall, due to economics, the receptacle mains is definitely the best choice here. Lastly, and the most overwhelming argument for the converters is space & weight. Most of those 12V and 24V batteries are huge and have no chance of fitting in the grid. It would make no sense for the group to choose to draw power from an external source from the 3D grid, meaning wires and having to mobilize the battery as well, for it to then run out of power at some point, when the 3D grid can be powered from mains and never need a recharge. Moreover, most of the batteries on the market are quite heavy when compared to the grid and the materials used for the grid. It would make the 3D grid very awkward to fit a battery in it that weighs 20 lbs when all the other components are 3D printed and the PCB is very light.

3.3.9 LED Part Selection

For our LED selection, we chose RGBW LEDs that are individually addressable. These LEDs have a built-in controller so that the brightness and color of each one can be changed independently of the others. These LEDs operate at 5V DC and have 4 pins. The pins are VSS for ground, DIN for data in, VDD for power supply and DOUT for data out. The LEDs must be arranged with DOUT of the first LED to feed into DIN for the second LED and so on for the entire grid. These LEDs will likely be soldered onto the PCB under each rising unit and not rise with the unit although this could be changed. Since these LEDs come with a controller, we do not need to purchase a controller for them to factor into the circuit. Having a controller/driver means that we can adjust the brightness and color of the LED.

3.3.10 Motor Selection

The motors in this project need to be small enough to fit in the housing, but be able to provide enough torque to move the cylinders up and back down. We selected a 37mm gear motor with an

encoder. With the encoder, we can measure and ensure the rotational speed we need, we can change the track rotation direction, we can detect stalls, and more. Without the encoder we would be limited to pulse-width modulation (PWM). With PWM the motor would still spin but if the load changes, the battery voltage dips, and from there more issues can occur. If we just wanted the motors to run, then a motor with no encoder would be sufficient but we need the motor to turn the opposite direction as well to lower the cylinders.

The exact motor we are looking at also has a great variety. This gives us a lot of options to tailor the motors for our project. Their voltage options start at 6V, then 12V and 24V. From there, gear ratios can be changed. They offer 6.3, 10, 90, and even 450. The 6V motors with a gear ratio of 6.3 provide greater than or equal to 1.2kg cm. This should be more than enough for us because the cylinders will be far less than 2.4lbs. We are only going to work with the 6V motors because we have no need for higher voltage motors. Their current demand is 1.4A under rated load. Rated load simply means that the motors are functioning continuously and safely, as designed. Their stall current is 3.5A. Stall current is how much current the motor will take when malfunctioning.

3.3.11 Power Distribution

The power delivery system originates from the AC/DC converter from wall receptacles. From there, it splits off into three separate systems. The ESP, the motors, and the LEDs all have their own regulators. The reason the power system is divided into three systems is because of the different voltages that they require. For example the ESP takes a maximum of 3.6V but the motors and LEDs take 5. Putting these two on the same power rail would mean the ESP sees too much voltage and gets damaged, or the motors see too little and they do not function properly. The reason that the motors and the LEDs are separated is because of the current draw. It is difficult to find regulators that are 5V but can also handle high currents. The motors take 1.4A *each* when under rated load. It is important to factor in inrush currents and stall currents. For the motors we are using it is 3.5A *each one*. Inrush currents are when the motor is being turned on and goes from being still to rotating. The current peaks here because since the motor is still, it needs to overcome its own inertia to move. This means the motor draws a lot of current for a split second. The inrush current does not last for very long so doing calculations with 3.5A as the worst case is sufficient. Next, the LEDs take 0.6A *each*. For 64 LEDs, that is 3.84A. When accounting for both LEDs and motors, the current demand is around 15A. Putting this all on one regulator would mean a lot of heat dissipation and strain on the regulator. It is important to mention that although the grid will only be consuming 15A while the motors are in motion, the worst-case where the current demand is highest is what must be considered so that the electronics do not fail if they are put under that load.

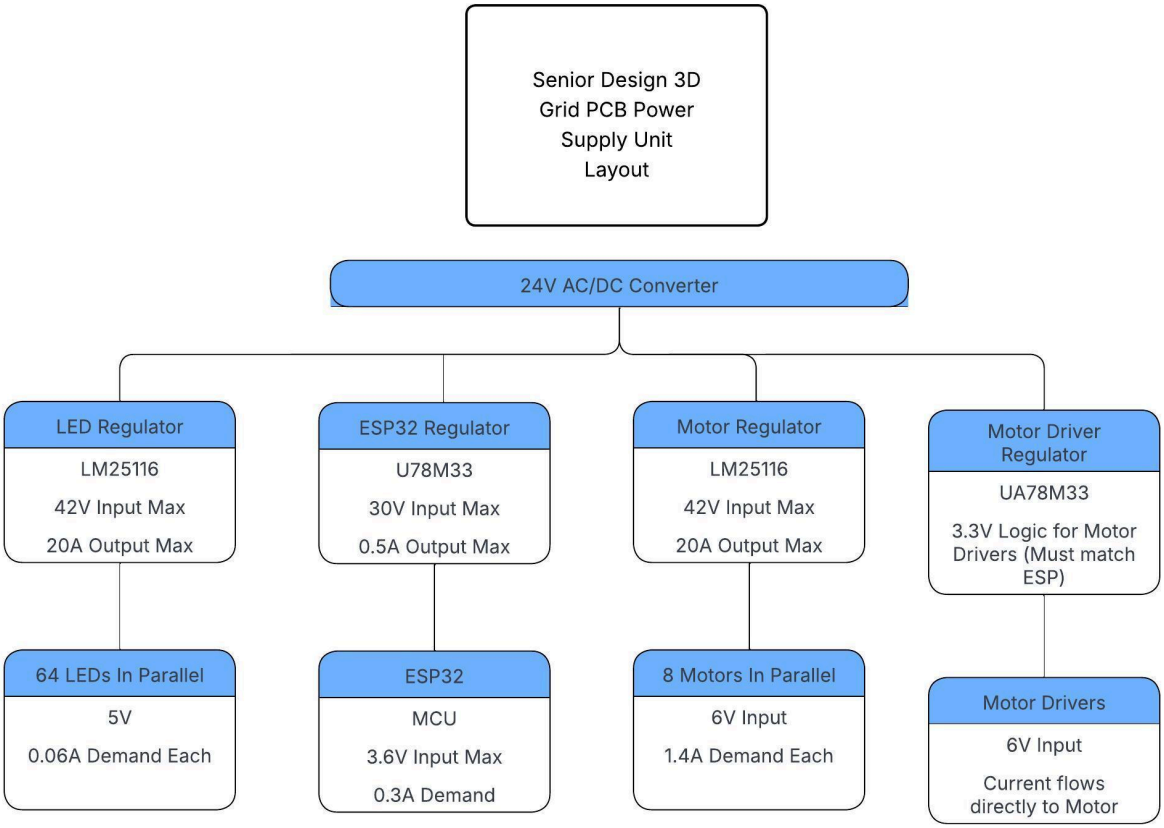


Figure 3.2 *PCB Layout*

3.3.12 Regulator Selection

Name	Downstream Device	Type	Vin, max	Vout, max	Iout, max
LM25116	Motors	PWM Buck Controller	42V	36V	20A
UA78M33	ESP32	Linear Votlage Regulator	24V	12V	0.5A
UA78M33	Motor Drivers	Linear Votlage Regulator	24V	12V	0.5A
LM25116	LEDs	PWM Buck Controller	42V	36V	20A

Table 3.20 *Regulator table*

Voltage regulators are hugely important components for power distribution and management. They ensure that components see their necessary voltages, not more or less. It's very important that the components in our project see proper voltages as some of them are sensitive and can malfunction. In the case of this project, the regulators will be seeing the 24V from the AC/DC converter. From there, the voltage needs to be stepped down for each component.

When going over regulators, the linear vs. switching regulators conversation is very important. Linear regulators maintain a constant output voltage by dissipating excess input voltage as heat. They are low noise, which is great for analog systems and systems that are sensitive to noise such as our ESP. This could be a great choice for the ESP because too much noise can begin to affect data transmission to and from the ESP. These regulators are definitely inefficient because any extra energy in the form of voltage is transformed into heat. Using too many of these or stepping down higher voltages to lower voltages mean that too much heat on the PCB could become a bigger issue.

Switching regulators on the other hand, use transistors to maintain a voltage on the output end. Some switching regulators use MOSFETs to connect an input source to an output. In between this connection are inductors, capacitors, and diodes. The inductors resist sharp changes in current, so when the MOSFET connection is closed, the current flowing through the inductor gradually and linearly increases until the voltage on the output is achieved. From there, the MOSFET opens when the output voltage goes higher than the desired value (within a set deviation above or below the desired voltage). Naturally, since the connection is open and there is no current flowing, the output voltage will begin to drop. The diodes in the circuit provide for an alternate route for current to flow while the MOSFET is open. The inductor in the circuit uses its stored energy from its magnetic field to provide the last bit of energy it has in the form of current to the output (meaning the voltage also drops linearly as it did rising). Once the output voltage drops within a certain value, the MOSFET closes again and allows current to flow through the circuit to the output to raise it again. Capacitors resist changes in voltage so they are placed within the circuit to help maintain the voltage at the output. Switching regulators will definitely be better with heat dissipation so it's important to note that these can be swapped in for linear regulators in areas that are not analog signal sensitive. Switching regulators can step voltages down, step voltages up, or invert voltages. In our case we will be using step down regulators.

ESP Regulators						
Name	Type	Vin Max	Vout max	Vout min	Iout max	Considerations
LM25017	Buck Regulator	48V	40V	1.25V	0.6A	Huge voltage headroom
UA78M33	Linear Regulator	30V	12V	3.3V	0.5A	Noise is the least issue here
LM2596	Buck Regulator	40V	37V	3.3V	3A	Good current headroom
TPS54538	Buck Converter	28V	22V	0.6V	5A	Lots of current headroom

Table 3.21 *ESP regulator table*

All of the regulators in consideration for the ESP are seen in figure 3.6. I chose the U78M33 linear voltage regulator. This module can handle the voltage seen from the converter and step

that down. The minimum output voltage is 3.3V and the maximum is 30V. The maximum output current is 0.5A. The ESP usually takes 260mA during high frequency or Wi-Fi bursts. Using a regulator that can handle 500mA gives enough headroom for any current or thermal issues. Although there would be a lot of energy dissipated as heat from the 24V to 3.3V, linear regulators are much better on electrical noise. Since there is no MOSFET switching inside, there are no high frequency harmonics that can influence the data. Since the ESP is the only component working with data, and our project is a *data visualization* grid, we are going to take the thermal energy losses here to ensure safe data transmission.

The same regulator as above will be used for the motor drivers. They require 3.3V for logic and putting them on the same regulator as the ESP would cause issues with heat and current draw. Since we have enough headroom for the AC/DC converter, there is no issue with adding this regulator to the unit.

For the motors, we use one LM25116 regulator controller. This is one of the few regulators that was found that can handle such high current draw at 6V. We are likely to split the motor module into two smaller groups to be able to deal with the stall current issue. At 3.5A each, the total current draw when all eight motors stall is 28A. The LM25116 regulator can handle a maximum output current of 20A. We could split the module into 4 motors each with one LM25116 for each one. This way if all 4 motors stall, each LM25116 can handle the 14A. The other way to deal with the stall current is using fuses to open the circuit if the amperage reaches a certain amount. It is important to note here that the fuses cannot be set to trip at *too low* of a current because during inrush current situations, the fuses could trip and the circuit wouldn't work when the motors begin turning.

Lastly, for the LEDs, we are also going to use the LM25116. The LED current draw is around 3.84A for all 64. Although we could orient the LEDs all in series and only draw 60mA, this would mean that if one LED goes out or stops working, all of them would stop working. We would much rather orient them in parallel so that if one goes out, the rest still work. The LM25116 gives more than enough headroom in terms of current demands *and* voltage. Adjusting some of the values in the external power stage can help to further tailor this regulator to the LED's demands (and the motor's demands).

3.4 Fuses

A fuse is a device used as protection against short circuits and overcurrent. Fuses are commonly used in electronics where potential dangers could appear as a result of high current. Overcurrent is a term used to describe a current value higher than what is rated for the fuse. Every fuse has a value for current in which it will begin to act as protection for the circuit.

Fuses work by containing a wire or strip of metal that will heat up as the circuit approaches its overcurrent value. When the metal wire inside the fuse heats up to a certain temperature, it will melt away causing the connection between the points to be lost. This allows the circuit to have its

own limit of current, protecting the device from dangerous behavior as a result of shorts or high current.

3.4.1 Types of Fuses

There are many types of fuses, all with different functionality and benefits. In some systems, you may be worried about surges of current destroying your circuitry. There is a fuse to help with that. Any current related case you may encounter is met with a solution in the form of a fuse. Some of the fuses we will discuss may not be as commonly used, but serve their purpose for the user.

3.4.1.1 Fast-Acting Fuses

The first fuse we will look at are fast-acting fuses. Fast acting fuses work best in a system composed of sensitive electronics. This is due to the quick nature of the fast-acting fuse. These fuses are able to open very quickly. The moment their current rating is exceeded, the fuse will open. There is almost no delay seen by a fast acting fuse. These fuses are typically rated for current ranging from 0.1A to 30A. After this is when you will start to see the fuse heat to the point where internal wires are being burnt up as protection.

Fast-acting fuses have a response time of less than a millisecond. They are typically used in applications involving microcontrollers, circuits involving logic, or electronics containing sensitive ICs. Fast-acting fuses provide excellent protection against overcurrent and shorts and are very precise. One of the main disadvantages of fast-acting fuses is their inability to handle inrush current. Inrush current is a term used to describe the quick, high surge of current as a result of a device turning on.

3.4.1.2 Slow-Blow Fuses

Slow-Blow fuses are almost the opposite to fast-acting fuses. Before, we saw a fuse that wasn't good for quick, high surges of current on startup. These slow-blow fuses are made especially for that purpose. Opposite to how fast-acting excelled at sustained overcurrent and was poor with inrush current, the slow-blow fuse excels when dealing with inrush current and is poor with sustained overcurrent. Each of these fuses prove to be effective in their own situations, causing the user to go with one over the other.

Slow-Blow fuses will typically have a response ranging anywhere from 10 milliseconds to multiple seconds. At the very best, this is more than ten times worse than that of the fast-acting fuse. This slower response time plays a role in its importance and usage. With the fast-acting, the fuse will be blown extremely quick, this slow blow does not exhibit the same behavior. They are most often used in applications involving motors, solenoids, transformers, and circuits with

inductive loads. The biggest advantage of the slow-blow fuse is its ability to handle that quick, high surge of inrush current. You can expect a current range of half an amp to thirty amps. The biggest disadvantage of using a slow-blow fuse is the slow nature of them.

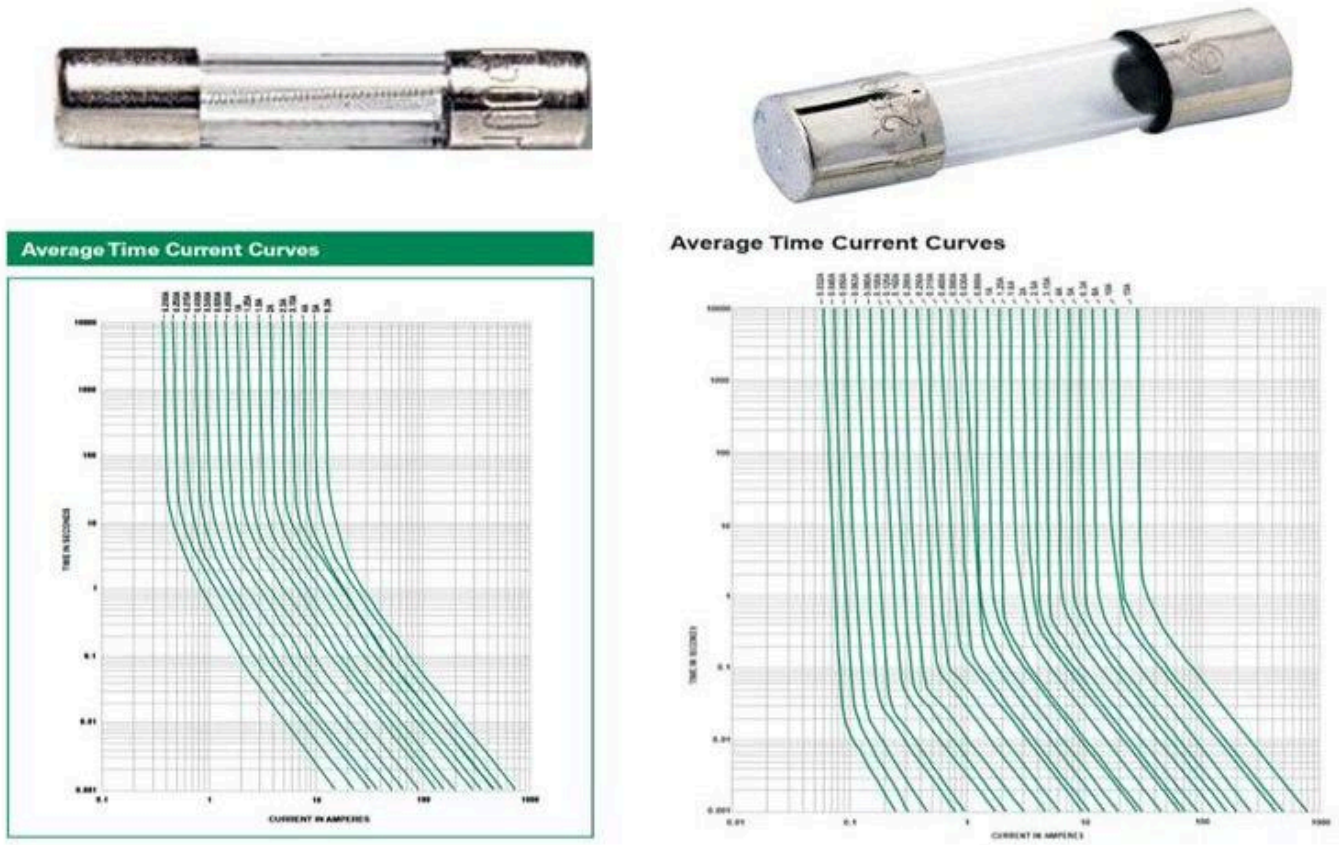


Figure 3.3 *Slow-Blow(left) and Fast-Acting Fuse(right) time comparison graph*

3.4.1.3 Blade Fuses

Blade fuses are fuses that are commonly found in automotive applications. They typically deal with 12V DC systems, which is closely related to our system. Different to the other fuses we've looked at, Blade fuses come with varying size and ratings. When looking to get a blade fuse, you may decide between a miniature, standard, or large model.

The response time for a blade fuse can range anywhere from 10 to 100 milliseconds. This range comes from the fact that blade fuses vary in performance metrics. Like previously mentioned, blade fuse's typical application is in automotive systems. Along with this, they are also commonly seen when dealing with power systems that use DC voltage. Blade fuses are especially good when frequent replacements are necessary. They are very easy to replace and widely available for purchase. One of the major downsides to using a blade fuse is its bulkier

frame. Another problem seen when using blade fuses in their limited precision. Comparing it to a fuse like the fast-acting fuse, we won't see anywhere near the precision. The range of current from a blade fuse can be anywhere from 1 amp to 40 amps.

3.4.1.4 Cartridge Fuses

Cartridge fuses are very small, cylindrical fuses that are most commonly found in small electronics and circuits used for control. You can expect a cartridge fuse to be in the range of 5-20mm and made from glass. The cartridge fuse, unlike the others we have looked at, is not a specific fuse, but a general term used for the small glass fuses. Since fuses are not as exact as the others, comparing will be a little different.

Cartridge fuses refer to a more broad description of fuses than the ones we have looked at already. Since this is the case, it's not as simple to determine response time ranges. With that being said, cartridge fuses do have a pretty fast response time for the most part. They are most commonly used in applications involving PCB mounting, control panels, or some other general purpose applications.

They are very compact, making them usable in many scenarios. There is also a visual cue to tell the user that the fuse has been blown. The cartridge fuse’s visual cue is also one of the problems with it. Since it can shatter, there's another thing you have to worry about when using them. There is also a very limited current when using them. The range of rated current is 0.1 to 10 amps.

3.4.1.5 Resettable Fuse

Resettable fuses refer to a fuse that is polymer-based. These resettable fuses are often referred to as PTCs or Polyfuses. The special thing about these resettable fuses is that when they heat up, they increase their resistance. This will limit the current of the circuit. When the fuse cools down, it will be able to perform a self reset. This reset functionality is where the name resettable fuse comes from. These resettable fuses, or PTCs, are good for this reason, but they are very slow.

As mentioned before, the resettable fuse is very slow. Before we were dealing with response times in the millisecond range, but now we are dealing with seconds taken to respond. They are typically used for USB ports, protection for batteries, or systems where frequent faults are apparent. The advantage to using a resettable fuse is, as the name suggests, it's reusable. This means that you won't have to worry about replacing one of these if you were to use it in your circuit. One of the big disadvantages to using one of these is its lack of precision. Another thing to worry about is degradation over time. Since it is constantly heating up and cooling down, it

will wear down over time. The rated current for one of these is also low compared to others we have looked at. The typical range for a resettable fuse, or PTC, is anywhere from 0.1 amps to 5 amps.

Type of Fuse	Fast-Acting	Slow-Blow	Blade	Cartridge	Resettable
Response Time	<1ms	10ms-Multiple Seconds	10-100ms	Varies	Seconds
Application	Micro-controllers, Sensitive circuits, logic circuits	Motors, Solenoids, Inductive loads, Transformers	Automotive, 12V Systems	PCB mounting, general purpose	USB ports, Battery protection
Pros	Precise	Inrush Current	Easy to Replace	Compact	Reusable
Cons	Cant handle Inrush current	Slower protection	Not very precise	Limited current, May shatter	Poor Precision
Current Range	0.1A-30A	0.5A-30A	1A-40A	0.1A-10A	0.1A-5A

Table 3.22 *Comparison of fuse types*

Chapter - 4 Standards and Design Constraints

4.1 Industrial Standards

4.1.1 PCB Design Standards

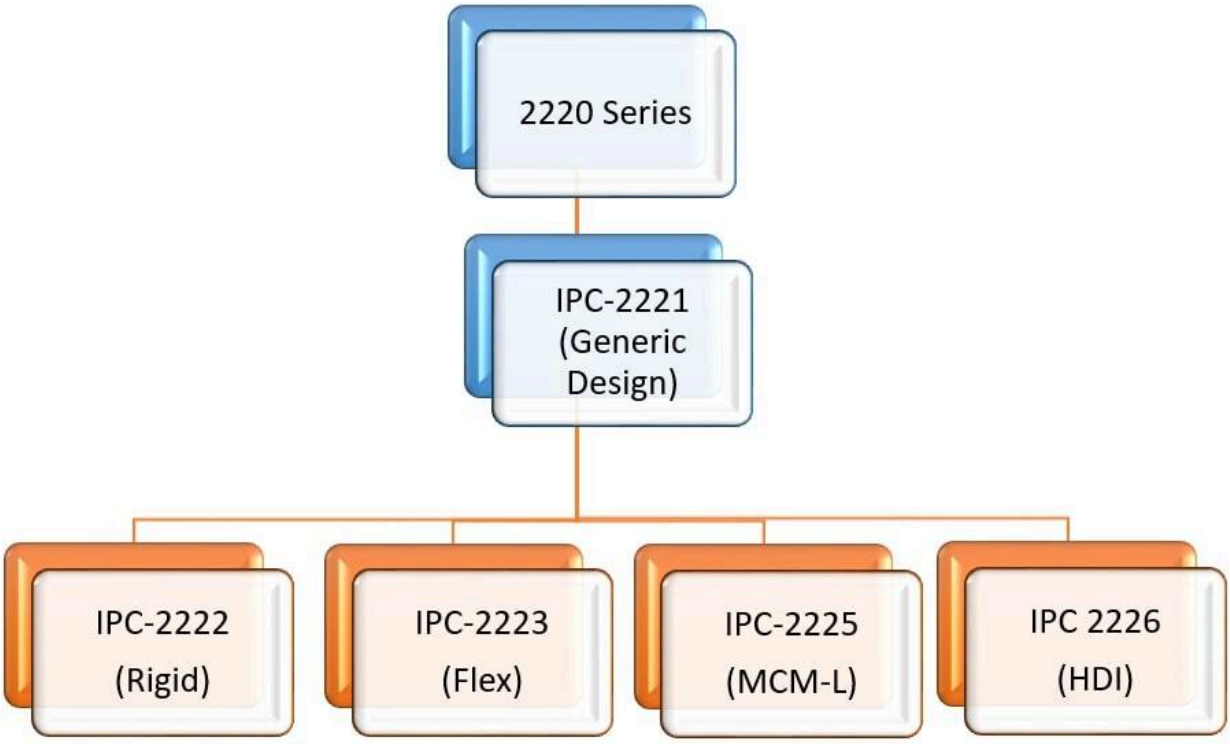


Figure 4.1 *IPC Standards in PCB Design*

PCB design standards define the manufacturing requirements set by each specific manufacturer. In this case, the PCBs will follow JLCPCB's manufacturing standards. Additional design rules are necessary to ensure proper functionality, such as good trace widths. These standards specify requirements like clearances, thicknesses, hole diameters, and acceptable file formats.

For JLCPCB, the selected 2-layer boards can use either 1 oz or 2 oz copper on the outer layers. Hole diameters must range between 0.3 mm and 6.3 mm, with vias having a minimum 0.3 mm hole and 0.5 mm pad. The minimum clearance between holes and pads of different nets is 0.5 mm. Vias must be at least 0.254 mm away from other vias or tracks, regardless of net connection. Pads should maintain a minimum distance of 0.127 mm from pads of other nets and 0.2 mm from tracks. The narrowest allowable trace width is 0.127 mm, and traces must be spaced at least 0.127 mm apart, with a 0.3 mm minimum distance between any trace and the board edge.

Additionally, the IPC-2221 standard provides broad guidelines for PCB design to ensure performance, manufacturability, and reliability. It covers key aspects such as:

1. **Design Guidelines:** Defines parameters for trace width, spacing, and layer setup to maintain signal integrity and proper power distribution.

2. **Material Selection:** Offers recommendations for PCB materials to meet performance and environmental needs.

3. **Component Placement:** Provides rules for optimal component layout to simplify assembly and testing.

4. **Trace Routing:** Advises on routing practices..

5. **Thermal Considerations:** Suggests methods for heat management, including the use of thermal vias and proper placement of heat-producing components.

6. **Design for Manufacturability (DFM):** Emphasizes designing PCBs that are easy to fabricate, assemble, and test, thereby reducing production errors and costs.

7. **Quality and Reliability:** Defines inspection and testing criteria to verify that finished boards meet industry standards and maintain high reliability.

Overall, IPC-2221 provides a comprehensive foundation for PCB design, ensuring that boards are both manufacturable and reliable while maintaining cost efficiency.

4.1.2 I2C Protocol Standards

The Inter-Integrated Circuit (I2C) is a two-wire, serial, bidirectional communication bus used to connect Integrated Circuits (ICs) or circuit boards. It is a standardized serial communication protocol commonly found in various electronic devices. To understand the importance of adhering to I2C standards, a general overview of how it operates is provided below.

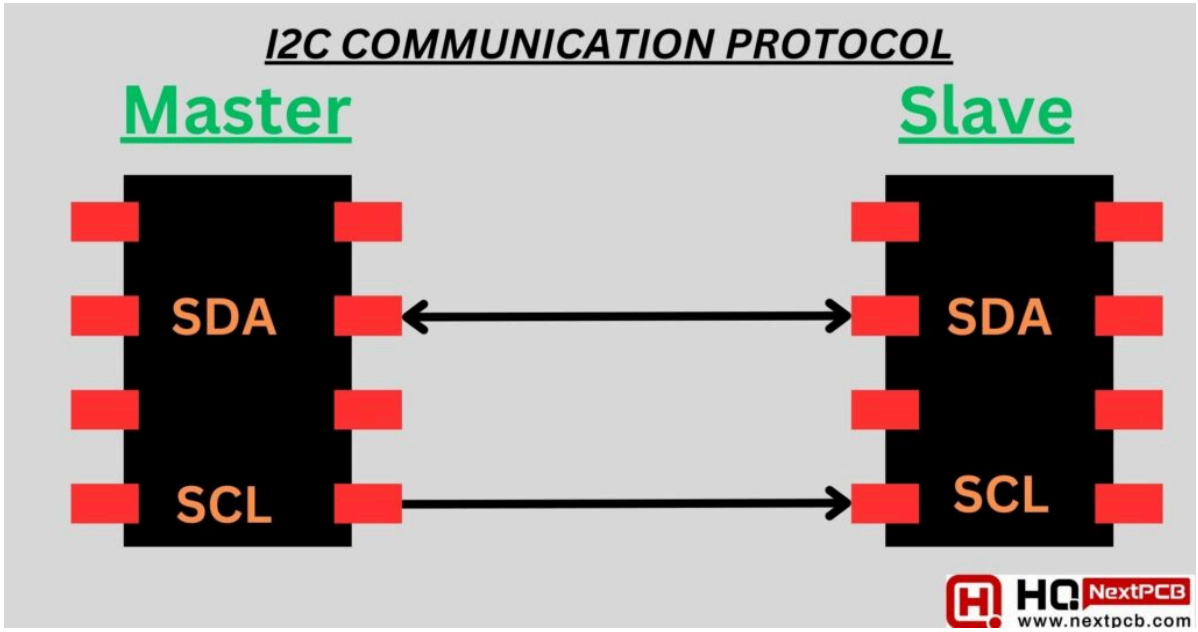


Figure 4.2 *I2C Communication Protocol*

The I2C system consists of two main lines: the Serial Clock (SCL) and the Serial Data (SDA). The master device generates the SCL signal to synchronize communication, while it can also

start or stop data transmission through the SDA line, allowing for asynchronous operation when necessary. The SDA line is bidirectional, meaning data can flow in both directions.

I2C uses an open-drain configuration, where slave devices cannot drive the line to a high voltage—they can only pull it low. Pull-up resistors are therefore required to return the lines to a high state when no device is actively pulling them low. The protocol also supports multiple slave devices connected to a single master, forming a multi-slave system. In this setup, the master manages communication to prevent data conflicts on the bus.

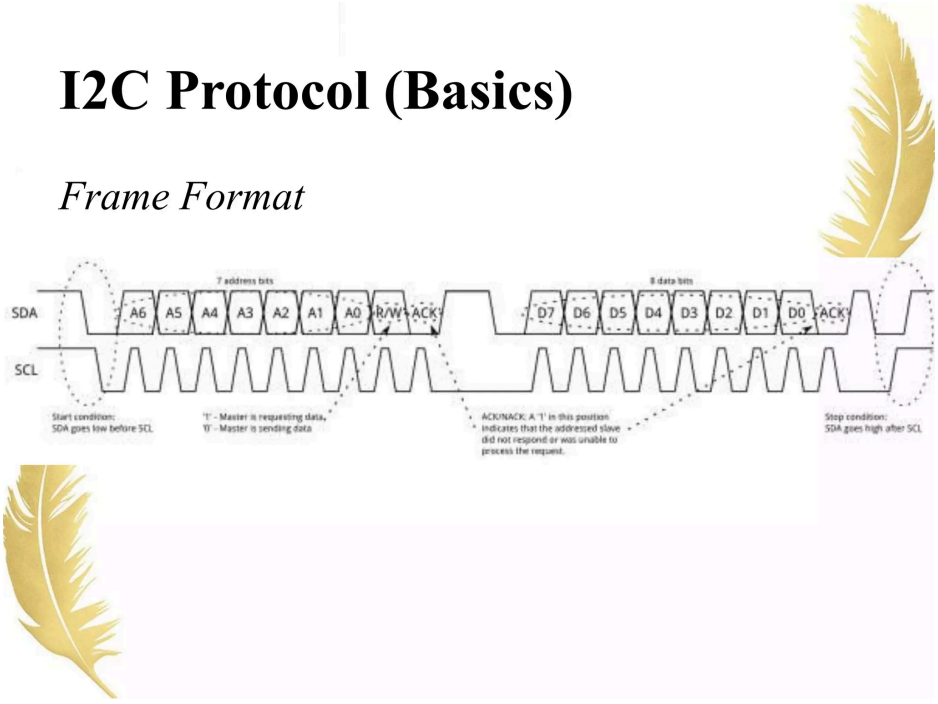


Figure 4.3 *I2C Protocol BIT Acknowledgement [19]*

In a standard I2C transaction, communication begins when both the SDA and SCL lines are high, indicating the bus is idle. A “start condition” occurs when the SDA line transitions from high to low while the SCL line remains high—signaling the beginning of data transfer. I²C typically uses 7-bit addressing, allowing up to 128 unique slave addresses. The master sends the desired address, followed by a register value to indicate whether it intends to read from or write to the slave.

Data transmission occurs one bit at a time, synchronized with the clock signal, beginning with the most significant bit (MSB). After each byte, the receiving device sends either an acknowledgment (ACK) by pulling SDA low or a non-acknowledgment (NACK) to indicate receipt status. The master ends communication with a “stop condition,” where SDA transitions from low to high while SCL remains high, notifying all devices that the transfer is complete.

Different I2C standards are defined by their clock speeds. The standard mode operates up to 100 kHz and is the most common. The fast mode increases speed to 400 kHz, while other less common versions—Fast Mode Plus and High-Speed Mode—can reach 1 MHz and 3.4 MHz,

respectively. However, the overall communication speed on the bus is limited by the slowest device connected.

Overall, I2C remains one of the most widely used serial communication protocols in microcontroller and integrated circuit systems due to its simplicity, flexibility, and standardization.

4.1.3 LED Standards

The WS2812B LED integrated circuit uses RGB LED along with a control circuit for addressable characteristics.

4.1.3.1 Electrical Specifications:

- Supply Voltage(VDD) - Typically a 5V operating voltage
- Operating Current - Each LED has a current draw of about 60 mA at full brightness
- Power Consumption - For 64 LED's in an 8x8 grid at max brightness, the current draw can be up to 3.84A at 5V

4.1.3.2 Communication Protocol Standards

Format:

- Each LED receives 24 bits of information (8 bits for each, red, blue, green)
- Data is transmitted most significant bit first
- After getting 24 bits, the LED sends the next data to the following LED in the daisy chain until the last LED gets the last 24 bits

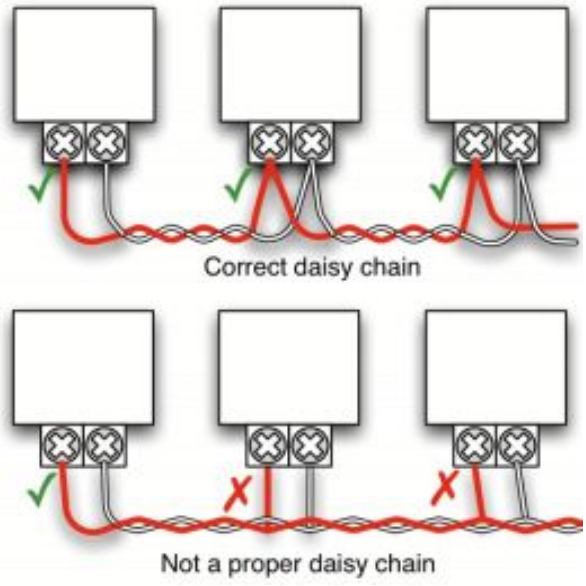


Figure 4.4 *Daisy Chain Concept [20]*

In figure 4.4 the concept of a daisy chain is shown as the output of one device is also seen as the input to the next device. We use this concept with our LEDs to 1. Minimize the amount of power consumption in our chain and 2. To be able to control all 64 LED's with one ESP.

4.1.3.3 PCB Layout Standards For LED

- Data Protection - Series resistor recommended on data input lines to prevent signal reflections
- Ground - A continuous ground plane under the array to provide a low-impeadance return path and to make the LEDs in parallel
- Trace Width - Minimum 0.5 mm trace width for power distribution
- Bypass Capacitor - 0.1µF capacitors placed at each LED's power pins

4.1.4 Motor Driver Standards

We are using an L298N Dual H-bridge driver to control our motors for this project.

Voltage and Current Ratings:

- Voltage Supply (VS) - 5V to 46V DC (Only need 12V)
- Logic Voltage Supply (VSS) - 4.5V to 7V (5V for MCU)
- Output Current - 2A continuously per channel
- Total Power Dissipation - 25W max

Onboard Regulator:

- Includes a 5V regulator onboard
- Supplies up to 500 mA
- Only functions when VS is between 7V and 35V
- With our VS of 12V, the regulator does function

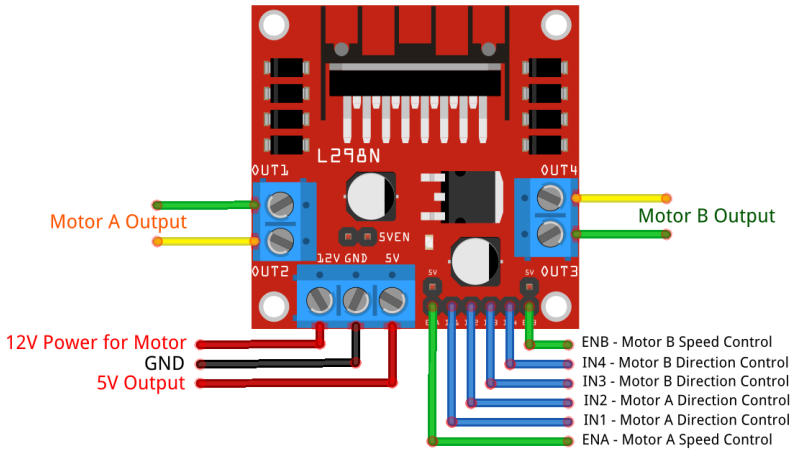


Figure 4.5 *Motor Driver Pinout*

Motor A Pins:

- IN1 - Control input 1
- IN2 - Control input 2
- ENA - Enable Pin (PWM speed Control)
- OUT1, OUT2 - Output Terminals

Motor B Pins:

- IN3 - Control Input 1
- IN4 - Control Input 2
- ENB - Enable Pin (PWM Speed Control)
- OUT3, OUT4 - Output Terminals

Power and Ground:

- VS - Motor Power Supply (12V)
- VSS - Logic Power Supply (5V)
- GND - Common ground for both motor and logic supply

ENA/ENB	IN1/IN3	IN2/IN4	Motor Action
HIGH	HIGH	LOW	Forward (Clockwise)
HIGH	LOW	HIGH	Reverse (Counter-Clockwise)
HIGH	LOW	LOW	Brake
HIGH	HIGH	HIGH	Brake
LOW	X	X	Coast

Table 4.1 *Motor Control Logic Truth Table*

4.1.5 Solenoid Standards

We are using 64 individual 12v micro push-pull solenoids as mechanical selectors to engage specific gears for tower raising. Each solenoid acts as a selective mechanism to engage towers with the rotating motor drive system.

Rated Parameters:

- Operating Voltage - 12V DC
- Power Consumption 5.5W Per Solenoid
- Operating Current - 5.5W/12V = 458mA Per Solenoid

- Resistance - 12V/458mA = 26.2 Ohms

Recommended duty time to be 30% - 50% max (ON time/total cycle time).

4.2 Constraints

4.2.1 Time Constraints

The primary limitation of this project is time. We have a fixed and relatively short period to develop, design, and implement our concept, which significantly restricts how much we can achieve. Because of this, we must sometimes prioritize meeting deadlines over creating a more advanced or optimized final product.

One major aspect of the time constraint is that the entire project must be completed within a single academic year. This fixed schedule leaves no room for extensions, meaning the project must be finished by the designated deadline in order for us to pass the course.

Another challenge posed by limited time is the need for extensive testing, debugging, and design refinement within a short period. Unforeseen errors or setbacks are inevitable, and the limited time frame provides little flexibility to recover from these issues. This makes the time constraint one of the biggest risks to the project's overall success, as problems during design, construction, or testing could easily cause delays.

Additionally, ordering parts—especially the PCB—can become a critical time concern. Supply chain delays or shipping issues could postpone component deliveries, preventing progress on certain parts of the design and wasting valuable time in an already tight schedule.

If not managed carefully, these time limitations could lead to a rushed final product that fails to meet performance expectations, forcing last-minute design changes or compromises. In the worst case, this could result in project failure and the need to retake the senior design course.

To mitigate these challenges, our team has committed to several key values: strong time management, accountability, productivity, clear communication, and consistent attendance. Most importantly, we follow the guiding principle of senior design—"Do What You Say You Will Do" (DWYSYWD). By adhering to this mindset and maintaining disciplined planning, we aim to stay on schedule and deliver a successful project despite the time constraints.

4.2.2 Economic Constraints

One of the main challenges in this project is the economic constraint. We were provided a budget of $250, which serves as the primary funding source for materials and components. While we are allowed to use personal funds to supplement this amount, the provided budget still establishes a strict financial guideline for planning and decision-making. This limitation creates a balance between maintaining cost efficiency and achieving strong performance within available resources.

Due to these constraints, the team must often choose more affordable parts that may not offer the same performance or durability as higher-end alternatives. Components such as motors, sensors,

or communication modules may operate at lower efficiency or speed but can still fulfill the project's basic requirements when integrated effectively. The challenge lies in designing the system to make the most of these components without compromising functionality.

Similarly, in choosing microcontrollers or other electronic modules, premium options that include advanced features are often avoided to stay within budget. Instead, cost-effective alternatives that still meet the necessary specifications are prioritized. Personal contributions may be used selectively to enhance specific aspects of the project or to address unexpected needs without jeopardizing the main budget.

Overall, the financial constraint requires careful planning, research, and resource management. By prioritizing essential components, setting aside emergency funds, and strategically using personal contributions when necessary, the team can ensure that the project stays within financial limits while still meeting its performance goals.

4.2.3 Environmental Constraints

Another design constraint for this project is the environmental limitation. Since the system is intended for indoor use only, it cannot be tested or operated under outdoor environmental conditions, which restricts how it can be designed and evaluated. This constraint affects factors such as lighting, temperature, humidity, and spatial setup, all of which must be controlled within an indoor environment.

One environmental factor is space. Because the project is tested indoors, it is limited by available room dimensions. Indoor testing also prevents the inclusion of real-world interference factors such as wind, sunlight variation, and uneven terrain.

Additionally, since the project is not exposed to outdoor elements, considerations such as dust, moisture, or network range are handled differently. The system does not need to be sealed against dust or rain but must function efficiently in a clean, controlled space with limited signal distance. Power and connectivity are typically more stable indoors, but wireless performance and layout can still be influenced by walls, reflective surfaces, or other electronic devices.

Overall, these environmental constraints define the project's operational boundaries. While indoor testing simplifies certain aspects of the design, it also limits how the system could perform in broader or uncontrolled environments. The goal is to demonstrate a reliable proof of concept within indoor conditions while providing a framework that could be adapted for larger-scale or outdoor applications in the future.

4.2.4 Sustainability

This isn't a very big constraint when it comes to our project as we are only going to use it indoors so we don't have to worry about any environmental factors. Only real constraint we would have with this is the power sustainability and to make sure the internal temperature doesn't get too high.

4.2.5 Political

We don't have many if not any at all when it comes to political constraints. Our project isn't sponsored by a specific company so where we order our parts from isn't an issue and since it's just a prototype for a lab, we would be good to make our own changes.

4.2.6 Power

The 3D data projection grid faces significant power distribution challenges due to the combined power requirements of multiple subsystems operating simultaneously.

4.2.6.1 LED Array Power Requirements

The 8×8 WS2812B LED array represents the largest power consumer in the system.

- **Maximum Current Draw**: 64 LEDs × 60mA = 3.84A at full white brightness
- **Typical Operating Current**: Assuming 50% average brightness across mixed colors, typical draw is about 1.9A
- **Power Supply Requirement**: 5V rail capable of delivering at least 4A

4.2.6.2 Motor and Solenoid Power Budget

The mechanical actuation system adds substantial power requirements:

- **Motor Specifications**: 8 motors operating simultaneously during height adjustments
- **Solenoid Power**: 64 solenoids (1 per tower position) require individual activation
- **Peak Current Scenario**: When multiple motors and solenoids activate during position changes, instantaneous current demand can exceed 5A
- **Duty Cycle Considerations**: Motors and solenoids operate intermittently, reducing average power consumption

To address these power demands:

- **Voltage Regulation**: Dedicated regulators for 5V (LEDs) and motor voltages
- **Power Sequencing**: Staggered activation of motors and solenoids to prevent simultaneous peak current draw

4.2.6.3 Constraint Impact

The power constraint directly limits:

- Maximum simultaneous LED brightness
- Number of towers that can be moved concurrently
- Refresh rate and animation capabilities

- Portability (requires wall power rather than battery operation)

4.2.7 Mechanical

The physical design of the 3D projection grid introduces several mechanical limitations that impact performance and scalability.

4.2.7.1 Tower Height Resolution

The system's ability to represent data is constrained by mechanical precision:

- **Discrete Height Levels**: Only 4 height positions available per tower, limiting vertical resolution (Can be scaled to more levels but staying at 4 for now)
- **Positional Accuracy**: Gear-driven lifting mechanism may have variance
- **Repeatability**: Mechanical wear over repeated cycles may reduce long-term accuracy
- **Data Representation Limitation**: Four levels may inadequately represent continuous data

4.2.7.2 Structural Stability

The physical arrangement of components creates stability challenges:

- **Gear Alignment**: Precise alignment of 64 gears (8 motors × 8 gears each) required for reliable operation
- **Bearing Placement**: Each of 64 positions requires properly seated bearings to prevent binding
- **Epoxy Tower Integrity**: Towers must maintain clarity and structural integrity under repeated movement
- **Base Rigidity**: 8×8 grid structure must prevent flexing that could misalign mechanical components

4.2.7.3 Size and Footprint Constraints

Physical dimensions restrict deployment options:

- **8×8 Grid Limitation**: Expanding beyond 64 positions would require additional motors and control complexity
- **Minimum Tower Spacing**: LED pitch and gear size dictate minimum spacing between towers
- **Vertical Clearance**: Four height levels with adequate separation require vertical space
- **Overall Footprint**: Complete assembly (including base, motors, and control electronics) occupies significant desk/table space

4.2.8 Manufacturing and Fabrication

The complexity of the system introduces significant manufacturing challenges that affect reliability and cost.

4.2.8.1 Custom Fabrication Requirements

Many components require custom manufacturing:

- **PCB Fabrication**: Custom PCB design must be outsourced to JLCPCB or similar manufacturer with 2-4 week lead time
- **Epoxy Tower Production**: 64 identical towers must be cast, cured, and finished with consistent optical properties
- **3D Printed Components**: Motor mounts, gear housings, and structural elements require precise 3D printing
- **Assembly Complexity**: Integration of electrical, mechanical, and optical systems demands careful hand assembly

4.2.8.2 Component Availability and Lead Time

Dependence on external suppliers creates scheduling risks:

- **PCB Turnaround**: 1-3 week fabrication plus 1-2 week shipping from overseas manufacturers
- **Component Sourcing**: Specialized components (solenoids, specific motors, WS2812B LEDs) may have limited availability
- **Supply Chain Vulnerability**: Chip shortages or shipping delays could halt project progress
- **Vendor Reliability**: Quality inconsistencies from low-cost suppliers may require component reordering

4.2.8.3 Testing and Iteration Constraints

Limited budget and time restrict prototype iteration:

- **Single PCB Revision**: Budget may only allow one PCB fabrication run, leaving no room for design errors
- **Component Testing**: Insufficient quantities for destructive testing or failure analysis
- **No Redundancy**: Limited spare parts means component failures during development could cause significant delays
- **Proof-of-Concept Focus**: Constraints force prioritization of functional demonstration over optimized final product

Chapter 5: AI Integration

5.1 Overview of AI Integrations/Applications

The concept of Artificial Intelligence is still rather new in 2025, but people are becoming rather familiar with these softwares/services. AI allows users to input anything they desire and receive a response from a software that thinks, compares, considers, reviews, and more in the same way that a human could to produce inhumanely fast outputs at a stunning accuracy. Integrating these kinds of software could provide huge benefits and ease-of-access for users. In our 3D grid project, it could be hugely beneficial to use AI so that users could input the data they want displayed and AI could convert this information and tell the grid what to do. As of now, the data displayed would need to be input in 2 different ways. First, pre-set data could be loaded onto a USB and from there, the grid can shuffle through. Secondly, a group member can have the code displayed on their computer and manually input the numbers for the data set to display. This would mean individually addressing each block and inputting their height values as this is the only thing that would change. These two methods are reliable and stable, but what they fail to bring to the table is flexibility and real-time adaptation. It would be very time consuming and boring to have to wait for a human to input data. In our implementation, AI could take a description of a dataset from a user such as the topology of a certain area and implement that dataset into the grid. AI could take the current trendline of a stock and display it on one column, with 8 columns, users could show the trendlines for 8 different stocks as they change in real time.

In short, we would implement a Large Language Model (LLM) for this project. An LLM is an advanced type of artificial intelligence. It is called a Large Language Model because it is trained on massive amounts of data. By massive amounts of data, we mean billions or trillions of “tokens”. Tokens are basically just words. When a user might input “The cat sat on the mat” the LLM just received 7 tokens (1 for each word). For the most part, one token is equivalent to about 4 characters in English. That's why every word in the sentence above counts as its own token. The AI takes the tokens input from a user, and one by one, tries to predict the next token. A huge misconception is that AIs are going looking into databases for information after they are asked a question to answer it. This is not true. LLMs have been trained on billions of text data such as books, websites, articles, and more. When a user inputs data, the LLM has already seen multiple (millions) examples of responses that come after “What is a GUI?”. It then predicts, token by token, the text that most often follows a similar sequence of words than what was input by a user.

5.2 Review of Viable AI Options

In 2025, there are various options when it comes to AI. Many different groups of people have released their versions of AI. Existing options include, but are not limited to: ChatGPT, Grok, Claude, Gemini, and Meta AI. Of these options, it's important to review which platform would work best with our 3D grid.

The AI that will be chosen for the grid will need to be tapered to our specific needs. Since our needs are an AI with good language & thinking skills to understand what the user needs, and

exceptional coding skills to make it happen, we must take the best AI platform for *both* of those. First, language & thinking skills. For language, we need the AI to be able to understand the user's request. It is imperative that the AI can visualize *or* fetch the information we need. For example, if we need one column to track the S&P 500 for the day, it must be able to access the web and obtain values to input to the grid. For values that are constant day to day, such as the topography of a certain area of land, the AI must be able to remember (visualize) the values it needs to input for this set of data, or be able to access the internet to find it. In other words, the AI must have flexibility in how it solves and goes about realizing the data needed. In terms of thinking skills, this means it must be able to recognize that it doesn't know the values for certain things and recognize that it needs to access these values. All in all, this means that the AI must be flexible, and be *overall* good in many areas. Out of the box, the GPT models from OpenAI provide a fantastic "jack of all trades" approach that is sufficient for most use cases today, says *TechRadarPro.com*. They state that ChatGPT has also been integrated into GitHub to help programmers with their coding, which we will touch on later, but is very important. GPT-5 stands out for advanced logical reasoning, innovative features like Deep Research, and comprehensive problem-solving capabilities, says *Fello AI,* which in turn demonstrates that Chat GPT from OpenAI is likely to be the best down the road of overall comprehension, language skills, understanding user prompts, and logical reasoning.

The other important feature that is essential from the AI model, is coding. Even if the model is able to determine user requests and find data values for these requests, it must be able to properly implement code to the MCU to communicate with the grid on what to do. Whether the group will input a "baseline" code to the MCU and have the AI manipulate it to change the data, or have the AI completely handle the code has not been decided yet, but it still must be reliable with code to function properly. In terms of coding, Claude Opus 4.1 continues to be the developer favorite for real-world coding scenarios, excelling at solving complex programming issues with well-explained solutions that help developers understand the underlying logic, says *Fello AI*. It is very important to take into consideration that Claude is an industry *favorite* in terms of coding. This weighs heavily on its reliability and accuracy when solving real world problems. Moreover, *zdnet.com* performed their own tests to see the accuracy of AI's in the coding field. In the end, Claude 4 Sonnet and OpenAI's ChatGPT-5 were tied amongst 2 others with passing all 4 tests they created. This speaks heavily on the decision for AI in our 3D grid because while Claude is regarded by many as the best AI in coding, ChatGPT-5 is regarded as the best overall. If ChatGPT-5 tied Claude's Sonnet, then this speaks heavily onto choosing ChatGPT for this project.

5.3 Current Implementations of AI

AI has been a huge helping factor in performing calculations, recommending parts for PCB, and getting valuable information. It has helped by taking the exact products/parts we are using and running power calculations with them. It has helped with choosing the regulators, especially the type of regulators to be used in the PCB. Moreover, it was able to provide quick and concise information on battery safety, mains safety, SELV, PELV, and FELV circuits, and more.

To begin with, AI has run some very important calculations that have helped to decide the method that will power the circuit. I have chosen to use ChatGPT for a majority of the beginning of the research. Supplying ChatGPT with links has been sufficient for it to analyze voltage and current demands and understand whether some parts will work or not. ChatGPT has given good recommendations and has done the very basic jobs quite well. For example, when checking whether some regulators would work with the LEDs, it made good points when it said that the regulator wouldn't be able to handle the current demands. I definitely overlooked this and saw the voltage output it could provide and deemed it good for use, but when checking with GPT, it correctly told me that it wouldn't work. Next, ChatGPT has shown a very complex ability to read through datasheets quickly to make assessments. Datasheets can be long and complex so GPT's ability to read through it at a speed that isn't humanly possible and extract necessary data is impressive.

ChatGPT and all of the AI's aren't perfect though. While providing useful information and important considerations when selecting power management components, it can certainly take users in one big loop. ChatGPT seems to be quite conservative when it comes to safety measures and very small considerations. For example, when mentioning certain linear regulators that can handle the voltage and current demand for one of the components, ChatGPT will mention caveats like heat dissipation and issues with efficiency in a way that is far more concerning than necessary. When speaking with experienced PCB designers, they have said to not worry about efficiency and heat in the way that GPT has said. This drove me to constantly try and find another regulator that would work and have enough headroom not only for the current demand, but also for heat issues when in reality heat won't be an issue here. Moreover, ChatGPT and AI tools always seem to need constant reminders of context for the project you are working on. For example when speaking with GPT on finding a female header for the 5.5mm x 2.5mm AC/DC converter jack, GPT said that I need to make sure that the mounting style fits my board. It said that since the female header is a surface mounted piece, that I need to make sure my board takes surface mounted pieces as if the board was already printed. Moreover, when asking GPT if I can simply connect the LM25116 regulator directly to the female headers where the motors will be plugged into, it says no. It then goes on to explain how the LM25116 is a regulator driver with an external power stage. This is obvious and already factored in when selecting the LM25116 as the regulator for the motors. Although in the end, GPT gives a good answer, it's not without feeding you obvious and useless information in regards to the initial question that was asked.

To move further into what AI can't do is circuit simulation. AI is not at the capability yet to be able to connect circuit components properly, let alone perform simulations on how they would work. Even if any of the generative AI's could do these simulations, there would surely be a delay until it could actually be trusted. For example, when asking GPT to create an image for a first order, non-inverting, high pass filter, it gave me this:

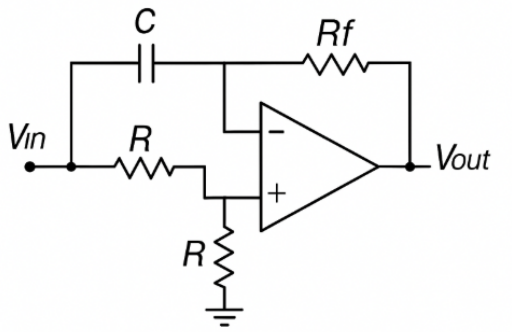


Figure 5.1 *AI generated high-pass filter*

This circuit would certainly not work as a first order filter because the op amp can no longer compare two signals, since the inverting and non-inverting inputs are both tied to V_{IN}. The input differential would now be 0. These are kinds of wiring and schematic mistakes that can happen in real life. Causes of following this schematic are output drifts, unpredictable noise amplification, or the circuit becoming unstable.

5.4 Methods to Implement AI

The first method to implement AI with the grid is the website. Since we are already required to make a website, it would be prudent to simply add a tab to the menu that takes the user to the AI we use to input a request (we will go over these methods with ChatGPT). This is the front end. The backend uses OpenAI's API to interpret user inputs and send commands to the ESP32. The flow of the website idea goes: user → website → backend AI → command JSON → ESP32 → grid display. This method provides a quick, user friendly and reliable method to get the job done. It provides a very clean look and doesn't require any previous knowledge of coding to work (which is a great marketability trait for a theoretical product on the market). Moreover, we can use a database to log commands and AI previous responses. This method does come with downsides such as requiring internet and a slight delay. For the internet issue, since our grid is going to be powered with receptacles, meaning everything will be indoors, this is not a dealbreaker, it's just a moderate issue. The delay in the cloud services is from the user's request needing to reach out to OpenAI's servers to process. The request from the user isn't going to be processed locally on the computer/laptop, it needs to be sent to OpenAI's servers to be solved. Moreover, network latency can cause some severe to mild latency depending on the strength of the network.

The second method is to use a local desktop app. A local desktop app would mean that the internet is not required. The app would call an offline LLM such as LLaMA 3 or Mistral or if connected to the internet, could call OpenAI's API. LLaMA 3 and Mistral are two options for an offline LLM. Essentially, unlike OpenAI, Claude, Grok, and more of the "big-name" LLM models, these models can run offline. The way they do this is by running locally, on the user's device, instead of reaching out online to the respective AI's servers. There are different options

for the size of the offline LLM to allow users to use these LLMs on weaker/less powerful GPUs or CPUs as the better, bigger models require more powerful processors. The command would then be sent to the ESP32 via USB serial or local Wi-Fi. USB serial just means that a direct connection from the user's device (laptop, desktop) can be connected via usb and send the data to the ESP via a direct wired connection. Sending commands through Wi-Fi means that the ESP32 hosts its own server through the Wi-Fi. The offline AI can then connect to this server and transmit any data needed to the ESP32 to update the grid.

Another method is to use the serial terminal. The ESP32 needs to be connected to the inputting device via USB. The user then opens a terminal such as PuTTY, Tera Term, minicom, or more. The command is typed into the terminal and the AI processes the input, locally or via cloud if online. If offline, the python script will run the local AI model to generate an output. If online, the python script will take the typed command and send it to OpenAI's API (as an example) and then wait for a structured command to be returned. Lastly, python script sends the data over to the ESP via direct connection. It is important to note that for this to work, there must be a bridge program that connects the communication from the python script to the AI. Otherwise, there is no way for the local terminal on a device to know to send anything to AI or to know that anything is even being sent to AI.

The last method to implement AI is to develop a mobile app. This mobile app would have its own user interface where the user can input their request. From there, the mobile app would send the request to the AI/Cloud backend to be worked on. Once the AI backend works on the request and develops the code needed, it sends the code to the ESP. The ESP processes the code and tells the hardware components what to do. The application would serve at the primary user interface for interacting with the AI-driven system. It would need a clean, intuitive, and beginner-friendly interface to allow users to easily input commands/requests. The app could be developed with *Flutter* to ensure compatibility with iOS and Android devices. Communication between the mobile app and the AI backend would occur through secure *RESTful APIs* or *WebSocket* connections. The *Restful APIs* connections are standardized ways for the mobile app to communicate with the AI backend server. When secure, it means the connection and data exchange follow specific protection measures such as: HTTP secure, authentication tokens, rate limiting & access control, and more. HTTP secure means that communication happens over encrypted channels, preventing any interception of the communication. Authentication tokens verify the user's identity and permission before accessing backend AI resources. This one may need a little bit of adjusting so that the access and communication is free to anyone (especially as this is treated as a theoretical market product). Rate limiting & access control is the least important of the three because it should be able to process data requests as often as possible. The *WebSocket* connections are a two-way communication channel between the mobile app and backend that work in real time. Their security is with WebSocket Secure (WSS) and works in a similar way to how HTTPs secures HTTP. They can also optionally use end-to-end encryption to secure. This method would be more important to send data that updates and interacts in real time such as stock market data. Overall, this approach provides a modular AI implementation, where the mobile app acts as a bridge between the user and the intelligent backend.

5.5 Summary

All in all, the AI implementation would be a massive part of this project in terms of taking a big step from just a project to a theoretically marketable product. It would be very important for the AI to be part of this, or any system that is similar enough to provide a cleaner bridge from the ideas a user wants to input and visually displaying that data in the grid.

In terms of which AI is most likely to be used, it is Claude. As mentioned before the AI needs to be able to complete two different functions at a high level: interpretation of real language and coding. Most of the AI LLM models on the market have very similar interpretation skills of what a user is saying. This is important because it needs to be able to understand what the user needs in the first place. From there, it must be able to take that and convert it to machine language to communicate with the ESP. If the AI cannot get a grasp of what the assignment even is, it is impossible for it to deliver good and accurate results. But, as mentioned before most of the AIs on the market have an equivalent proficiency in this department. The most difficult skill that is needed from AI is coding. This skill is more difficult because taking a user-defined input and *converting* it to machine language is where the inconsistencies may appear and inaccurate results may come. This is because human language and programming logic does not have a direct one-to-one relationship. The AI needs to interpret what the user needs and determine how this would apply to the hardware system. Applying the request to the hardware system requires the AI to have knowledge of the system, how it works with the ESP, and how to code that system. In the department of coding, Claude AI by *Anthropic* is definitely the choice. Claude is the #1 pick from programmers in the world of coding.

Chapter 6: Hardware Design

6.1 Electrical Hardware Design

6.1.1 LED & Motor Regulator LM25116

The 3D grid's system will be composed of one PCB with all systems and components on it. The power delivery system will originate directly from the AC/DC converter to a female header to plug into. The power will then be narrowed to each component as mentioned before. It is important to build the schematics of each regulator as its "typical use" shown on the data sheets. This allows for realization of all the voltage ratings and current demands

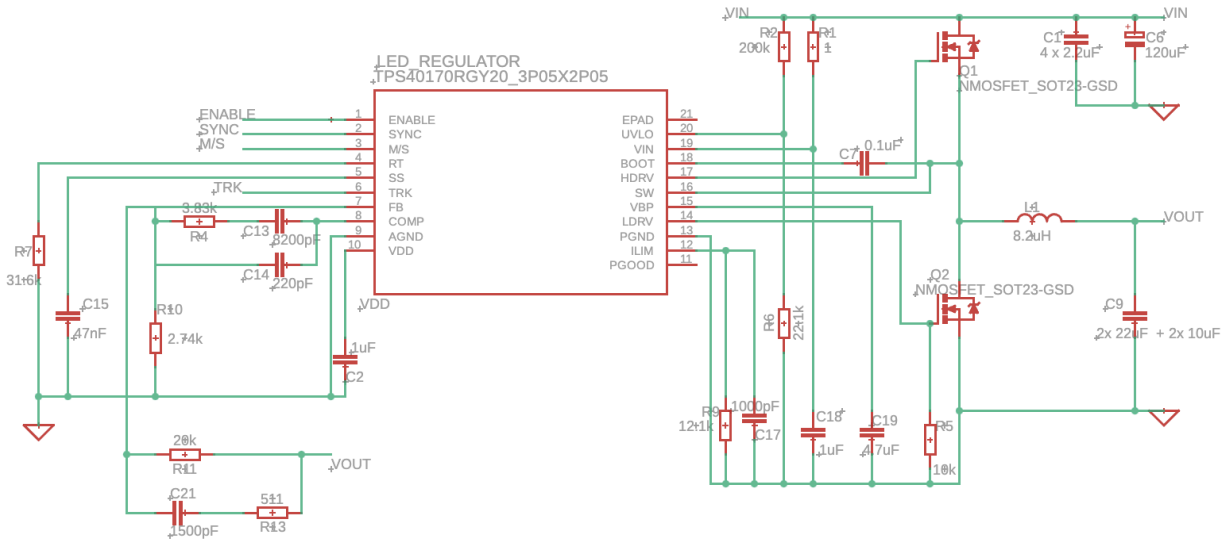


Figure 6.1*LED regulator schematic*

In figure 6.1 above we can see the regulator used for the LEDs. This is the standard application for the regulator controller. The datasheet shows all of the equations along with the values for every capacitor and resistor. As can be seen above, two MOSFETs are used in the design. These two MOSFETs are used for higher efficiency, less heat, and higher current capability. The MOSFETs are what we have to thank for the regulator's ability to handle the current for all 8 motors and 64 LEDs without dissipating huge amounts of heat. UVLO protections in this module allow for the regulator to be programmed to only turn on when the voltage is high enough. This helps avoid undervoltage situations where components could get damaged or malfunction.

For the LEDs, we are going to want to choose our switching frequency more carefully. At a higher frequency, we are going to reduce any flickering that might occur. This module supports a large bandwidth of 100-600kHz.

For the motors, the LM25116 will directly feed the motor drivers. Fuses will be later implemented either right before or after the motor drivers to open the circuit if there are any issues.

6.1.2 LM25116 to L298N Connection

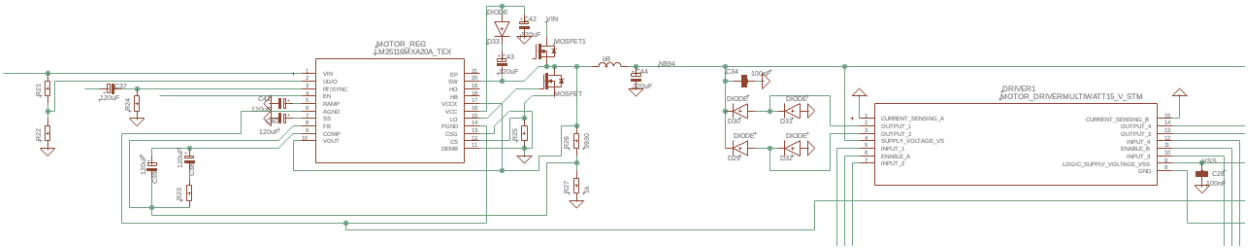


Figure 6.2 *LM25116 to L298N*

Here is the power connection to one of the motor drivers. The other three drivers are to the right of the image. The pin supply_voltage_vs goes up and connects with V_{OUT} from the regulator. This connection extends all the way down so the other three regulators can connect. At the bottom it can be seen that the motor driver ground and AGND/PGND are connected along with the other three driver's grounds. It is important to note that this connection is what powers the motors themselves. The connection for logic supply is shown in figure 6.4.

6.1.3 ESP Regulator UA78M33

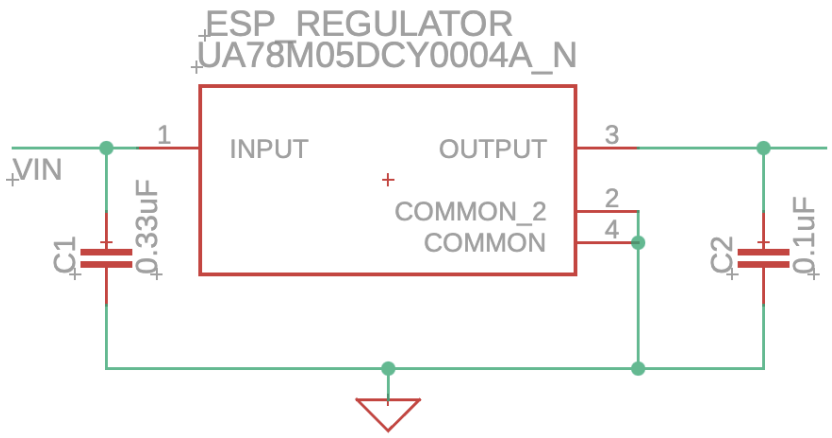


Figure 6.3 *ESP32 regulator schematic*

The above schematic shows the UA78M33 regulator for the ESP32. It only requires input and output capacitors for stability and accepts a wide input range, up to 35V. Going from 24V to 3.3V is going to dissipate quite a bit of heat, but as mentioned before, using a linear regulator will ensure we don't have to worry as much about noise. We may need to consider heatsinking here on the PCB, but if we find it to not be compromising to the operation of the project entirely, we may leave it as is.

6.1.4 UA78M33 to L298N

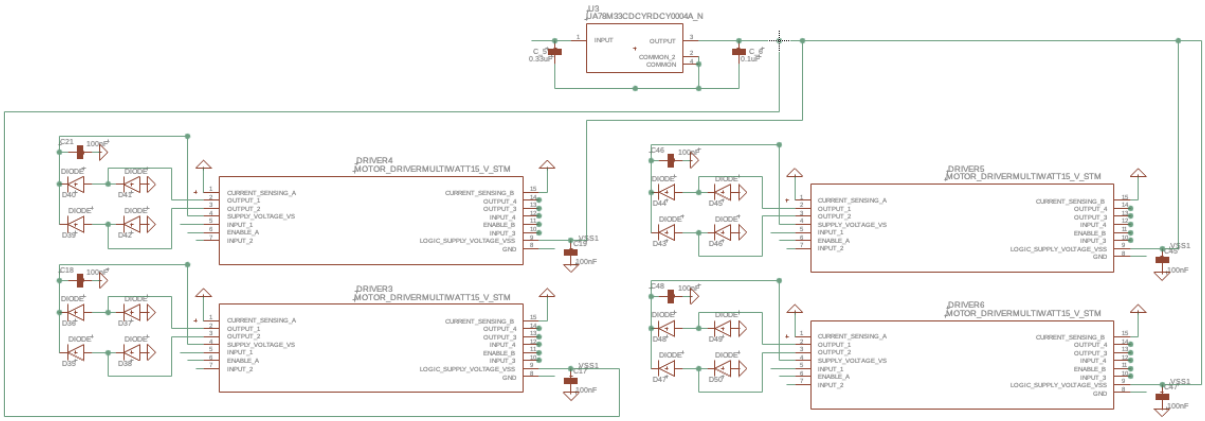


Figure 6.4 *UA78M33 to L298N for Logic Supply*

We decided to implement another UA78M33 for the motor drivers. Each driver takes around 40mA of current. The UA78M33 can only handle around 500mA. Given that the ESP takes ~300mA at peak, we are left with 200mA to work with. Knowing that the four motor drivers take a total of 160mA, we are left with 40mA of headroom. We felt that this was not enough because this is a linear regulator. Knowing that a lot of the energy will be dispersed as heat means that having 40mA of headroom could risk the regulator entering thermal shutdown in some situations. This would shut off the ESP and the drivers. To combat this, we are using another UA78M33 since we have enough headroom from the AC/DC converter in terms of voltage.

6.1.5 ESP Regulator UA78M33 to ESP Connection

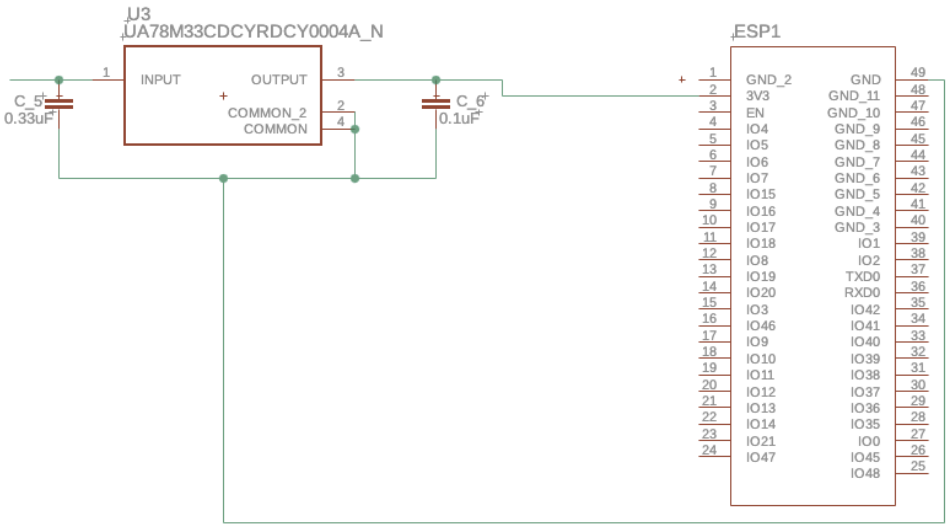


Figure 6.5 *ESP Regulator to ESP*

Here shows the regulator's connection with the ESP32. The output goes directly to the 3V3 pin where the input is and the grounds are tied together. In the full schematic, the grounds here are tied together with the entire system ground.

Figure 6.6 shows the layout of the motor driver as we are going to use it in the grid. It is important we have a motor driver so that we can control the motor more meticulously. We want to make sure we can control the duration of rotation, the speed, and the direction of rotation. The driver's setup gives us multiple different options to manipulate features and work with the motor.

6.1.6 Motor Driver L298N

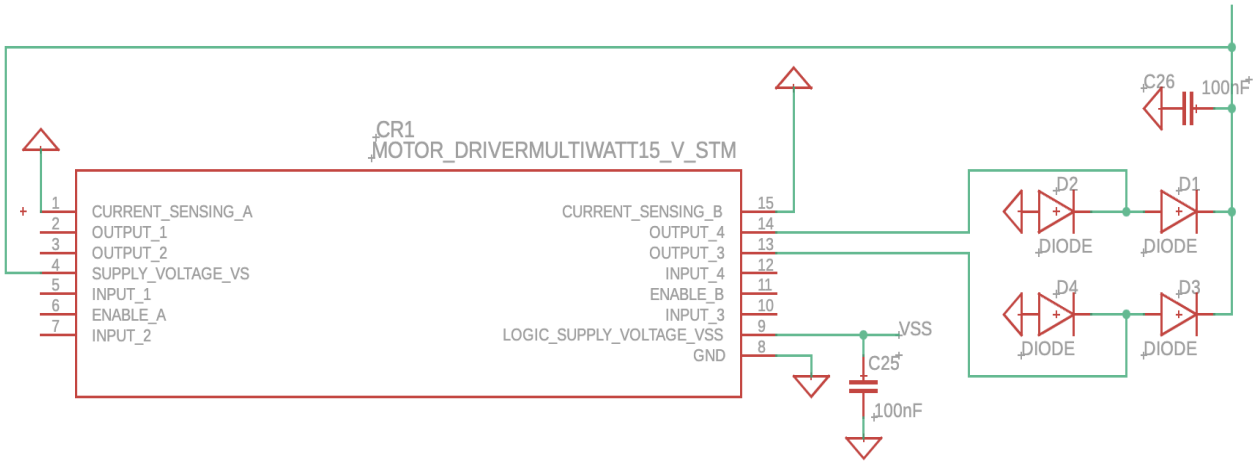


Figure 6.6 *Motor driver schematic*

For example, the 4 diodes on the right side of the schematic are the H-bridge. The H bridge uses diodes to control the direction of the motor. When the top left and bottom right diodes are switched on and the other two turned off, the current will flow through the motor in one direction, causing rotation of the motor in one direction. When the opposite layout is applied, the top right and bottom left switches are on, with the other two off, current will flow through the motor in the opposite direction. This causes the motor to rotate in the opposite direction. The motor drivers we have chosen are able to control two motors at a time with two H bridges.

In1 and In2 with EnableA are for one motor while In3 and In4 with EnableB are for a secondary motor. The left side is for the connections of one motor and the right side mainly contains the connections for a second H bridge. As of right now, we have the motor drivers set up to work directly with one motor. We may make adjustments to this later, but as of right now we have one motor driver per motor. Another interesting built-in tool of the motor drivers is the current sensing pins. These pins allow us to detect the current flowing through the driver to the controller. Pins 1 and 15 control this with a resistor added to ground.

As of right now we are not employing the current sensing tool because we plan on simply using fuses to open the circuit if the current reaches a certain amperage. The reason we are doing this design as of right now is because it becomes quite useless to measure that there is too much current going through the motors and not be able to directly do anything on the PCB. We prefer for a fuse to simply open the circuit and have us restart everything so that protection is instant. With a fuse, we will be able to safely prevent dangers associated with overcurrent surges in our circuit.

6.1.7 Motor Driver to ESP Connection

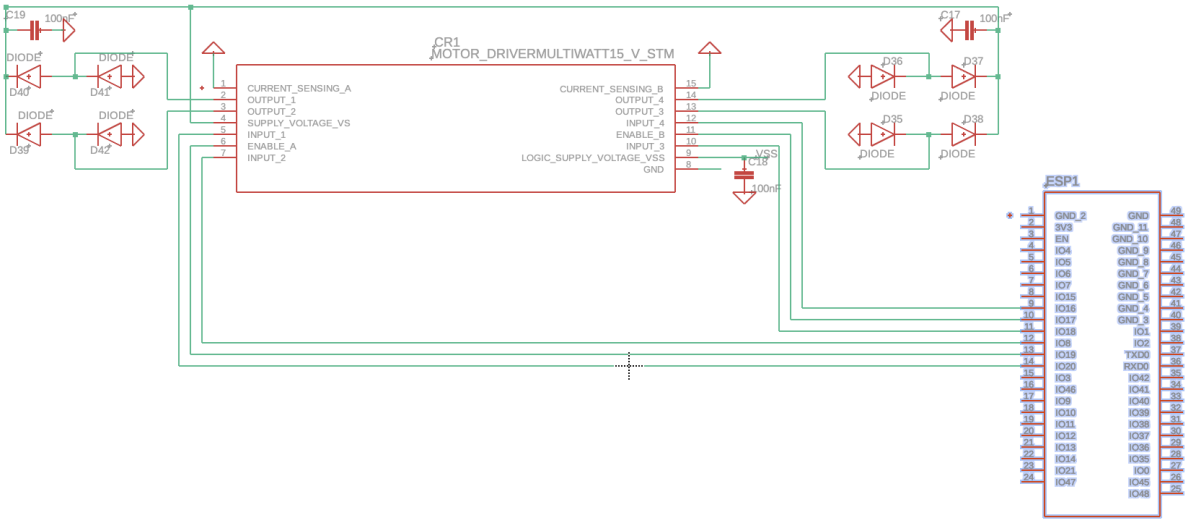


Figure 6.7 *Motor driver to ESP schematic*

Here is the motor driver's connection to the ESP32. The motor driver can control two motors. The left side shows the control connections for one motor, and the right side shows the connections for the other motor with its respective H bridge. All of the motor drivers and their connections fit on the ESP with enough connections left to control the LEDs.

6.1.8 LED Wiring Array WS2812B

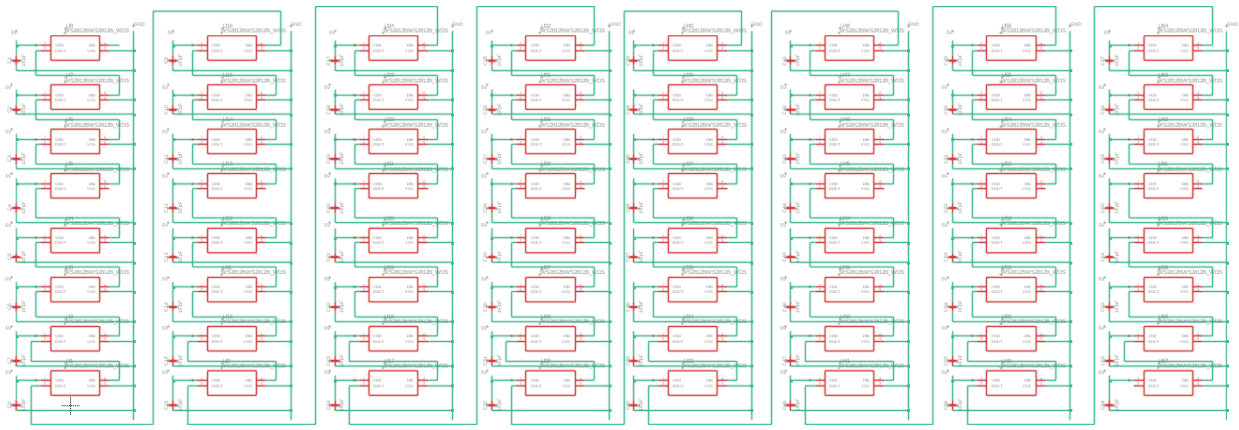


Figure 6.8 *LED Wiring Array schematic*

Figure 6.1.8 shows the daisy chain of the LED's in parallel with each output going into the next input and the last output floating. The first input will be connected to the MCU. All of the grounds are connected to the same wire to keep it parallel instead of series so we have a lower

value voltage needed. All of the 5 V wires will be connected to one wire which will, in turn, be connected to the LED regulator. We have capacitors connected to each LED which help filter out the noise and keep the voltage stable for each LED. These LED's are addressable that receive data through the DIN pin. The LED then processes that data for the specific LED and passes the remaining data to the next LED with DOUT. Each LED has an integrated driver IC that controls the colors individually which allows for independent control of each LED in the array.

6.1.9 USB to UART Conversion

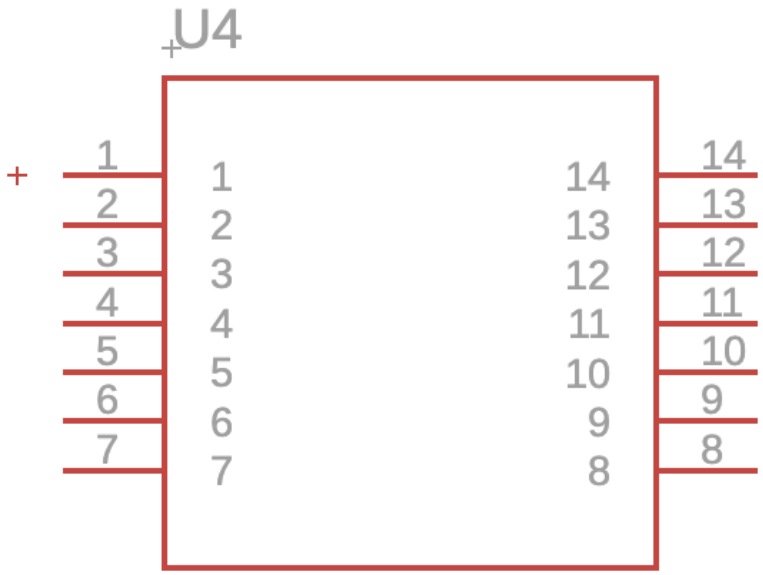


Figure 6.9 *MCP2221A*

Above is the image of the USB to UART conversion chip we are going to use. It will receive power from the first UA78M33 that powers the ESP since it only takes 24mA maximum. This brings our total current draw for the ESP's UA78M33 to ~324mA which still gives a good amount of headroom when compared to the 500mA the regulator can handle.

6.2 Mechanical System Design

6.2.1 Rack and Pinion

With a heavy reliance on mechanical components, it seemed necessary to include their schematics and designs to fully portray the 3D topology grid. Following are schematics of the rack and pinion along with a description of how they will interact with the electrical solenoids and motor. Along with the rack and pinion a standard 608 2RS ball bearing will be used in the center of the spur gear. Following are the drawings and dimensions of the rack and pinion.

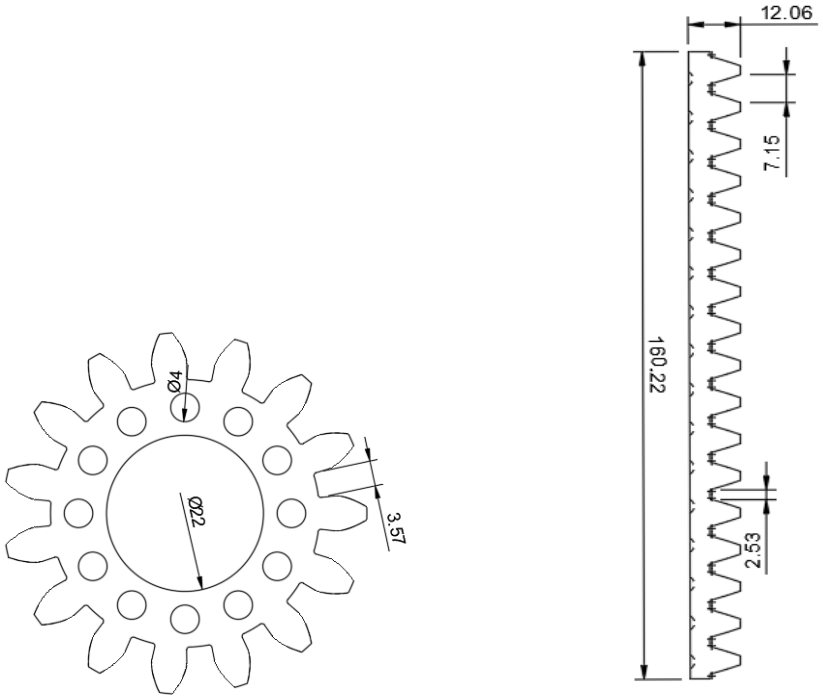


Figure 6.10 *Drawing schematics of the spur gear and rack measured in mm.*

In figure 6.2.1 it is shown that the center diameter of the gear is 22 mm. This fits the chosen 608 2RS bearing exactly with no clearance to allow for a tight fight. Using the tight fit allows the bearing to become resilient to loosening or becoming off center of the gear. Another feature that can be seen is the holes that correspond to each of the gear teeth. Each of the holes is used for catching the solenoid when it is activated, pushing the gear which drives the rack. When the solenoid is not active, the previously mentioned bearing allows for the gear to remain still. This works because the inner diameter of the bearing will rotate separately from the gear teeth. Solenoids will rest on the motor shaft and always be spinning, so when the solenoid is extended it catches a hole in the gear and drives the gear to spin. The amount of holes was designed to minimize error in height accuracy while still maintaining structural integrity.

Another aspect to note is the design of the rack and gear use the same pressure angle and module. This ensured that the two components would function together and prevent track jumping or failure. Clearances were designed into both components, with the root fillet radius given a clearance of 0.25 mm. Backlash, which is a small intentional gap placed between the teeth of meshing gears, was given a value of 0.25 mm. The use of these clearances prevent the system from seizing or generating too much friction. Given the torque requirements on the electric motor, increased friction or pressure on the teeth would increase torque required to drive the system. With eight of these rack and pinion designs per motor, the clearance provided helps reduce this friction and keep the motor from stalling under load.

6.2.2 Rack Holder and Adjuster

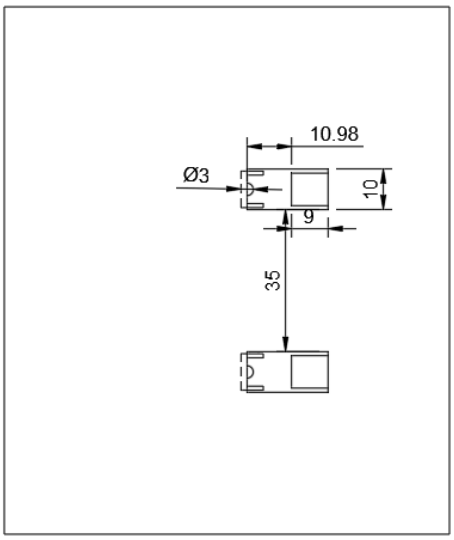


Figure 6.11 *Top view of the upper layer of the grid. Racks protrude though the holds and catch on the flap.*

A mechanism was needed to keep the racks from falling once the gear was disengaged and loose on the bearing. It was decided to incorporate it as a flexible flap that has the strength to hold the rack up, but bends when the gear is driving it up or down. The hole dimensions allow for the rack a lot of clearance, but is paired with guide rails with a 7.5 mm clearance, giving 0.5mm of free space. This will reduce friction and load on the motor, as well as keeping the rack meshed with the gear teeth. A half sphere is protruding near the rack guides, so the divots in the racks will rest on the sphere. Adding these created a more discrete level of height and eases with keeping track of how high each tower is. The flap took consideration as well, because it needed to be long enough to have enough friction with the teeth but also not increase jumping up multiple steps. After multiple iterations a flap of 9mm was decided, reaching enough into the tooth to operate smoothly and keep the towers up.

Chapter 7 - Software Design

Using the ESP32 ESP-IDF, all software will be written in C code and uploaded via a usb connection due to the increased memory, performance, and control given to developers using this sdk. FreeRTOS tools will also be utilized in design to provide higher control over tasks and scheduling. Because of long runtime requirements, using dynamic memory allocation (DMA) will be avoided as to not introduce complex bugs into the system. Adding DMA will prevent the system from experiencing nondeterministic timing issues or null returns. All runtime memory

will be handled through static arrays with defined size limits that cannot be overflowed via design. This will be achieved through the use of predefined buffers and global size constants, ensuring safe operation.

To accurately display a 3D dataset, two systems will need to be controlled via software in the MCU. The first is the LED system, which is the simplest to implement using the WS2812B protocol and libraries designed to implement it. After taking the input data from the user, all that is needed to store it in a 2D array which will get sent to the LED grid for displaying. Each LED circuit has three values used to choose the color that is displayed. To store accurate data in the LED pixel, the 2D array must store three values per pixel, and structs are the natural way to accomplish this on an MCU. Updating each pixel with the new values is simple and only requires a for loop and a small amount of code.

The Use Case diagram shows us how the user is able to interact with the system. The diagram provides a high-level view of how the system will function. The Use Case diagram above illustrates how the user will interact with the system. Using this Use Case diagram, we are able to clearly define the scope of our project as well as visualize the requirements from the point of view of the user.

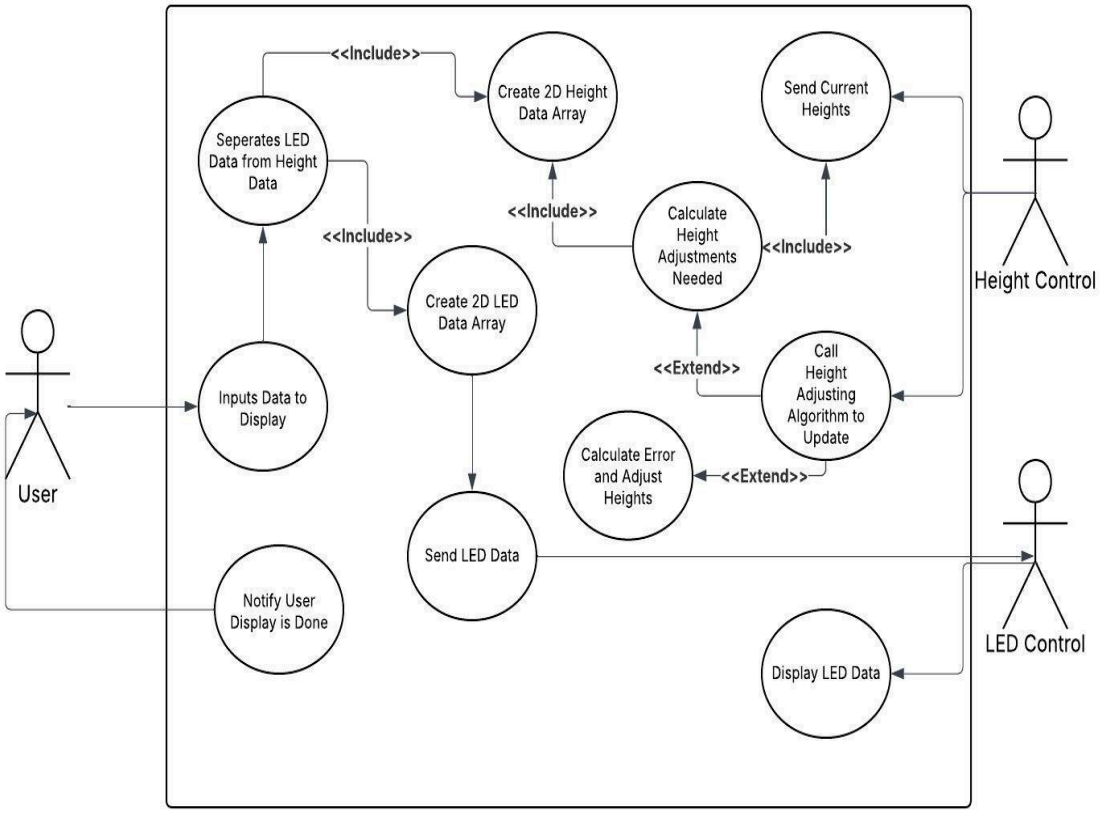


Figure 7.1 *Use Case diagram*

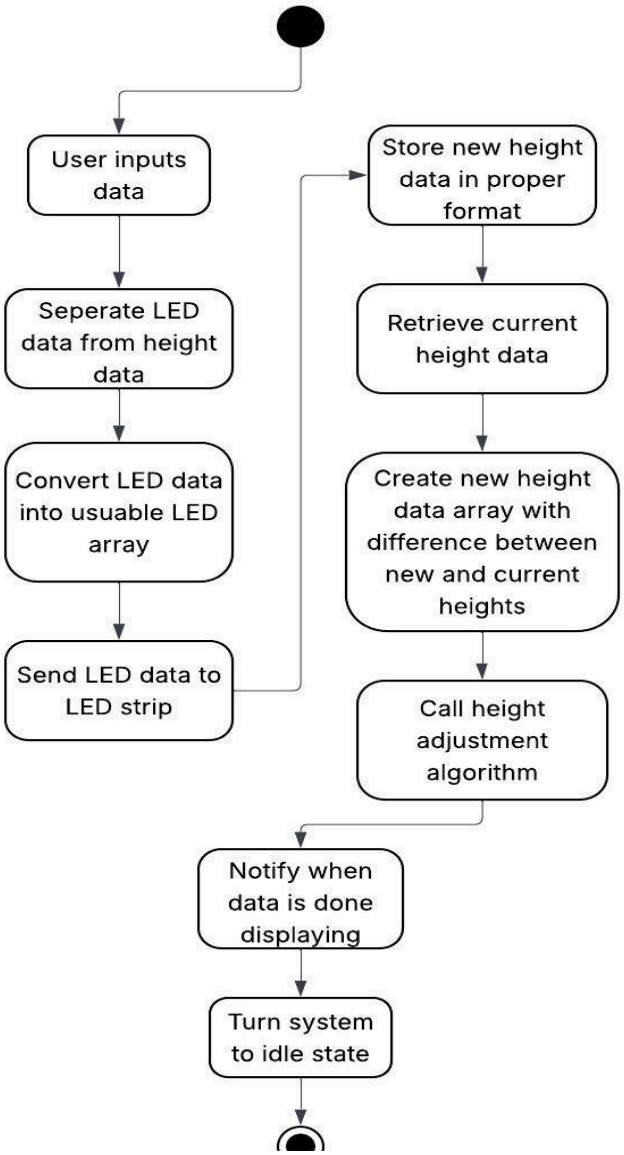


Figure 7.2 *Software flow chart diagram*

As mentioned above, one of the main components of the software system is the height control algorithm. Given the amount of steps a diagram is provided showing the flow of how the heights are adjusted until it fits the data given by the user. The following algorithm diagram will take the form similar to the activity diagram given above. At the end of the algorithm all 64 towers should be in the proper position to accurately display the dataset provided by the user.

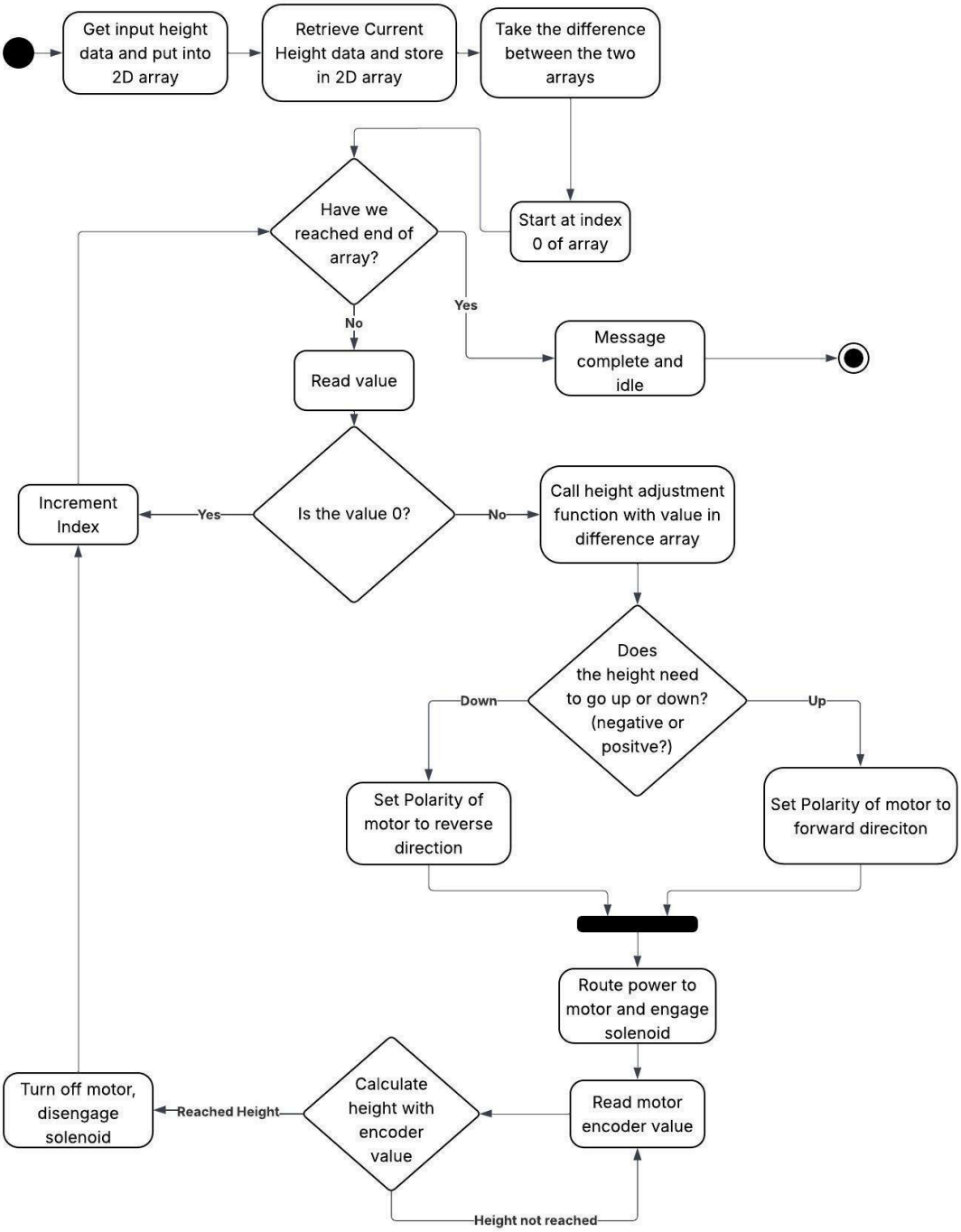


Figure 7.3 *An activity chart showing the flow of the height control software*

Figure 7.3 shows the software flow for adjusting each of the towers to the new input height. By taking the current height data stored in software and subtracting each tower's height with the desired height taken from the input, we get the difference in discrete tower levels. This difference shows the amount of levels a tower needs to change, if negative it needs to be lowered and if positive it needs to be raised. For now, the software will iterate through the array representing all

64 towers, and if the array value is nonzero it will either lower or raise that specific tower, updating the array. This way all towers will be adjusted to the correct height.

The above diagrams and designs represent the high level overview of the software design. In the following sections, a deeper explanation of each system will be outlined. More attention to the LED and height systems will be given, along with the software to control the solenoids and motor drivers.

7.1 LED Software System

Control of the WS2812B led strip will be achieved using the FastLED library that is available for the esp32 microcontroller. An input array is taken from the user, and it is then translated to the corresponding color for each LED. An array of color structs, with each struct containing the three RGB values, will be passed into a loop. After setting each LED/pixel to the correct color, all it takes is to call a display with FastLEDs and the colors will be displayed.

One issue that needs to be addressed is how the user inputs colors in the dataset. To achieve the highest resolution of displaying data, the more colors available to users means more data that can be displayed. Converting each color into tuples must be done through pre-coded values and offering a selection to the users. The amount of precoded colors that can be shown depends on the memory requirements of the MCU and how much code is needed to control other subsystems. At minimum all 7 colors will be given to the user to display on each LED. This will provide enough functionality to display many different situations. Display black unfortunately is not possible, so a tuple value of (0,0,0) will represent black which turns the LED off. Another feature will be for the user to be able to input a new color by providing three values representing the intensity of each red, green, blue LED. This makes the colors flexible and provides full use of the WS2812B LEDs. Limits will still need to be placed on the maximum amount of colors to be added, as dynamic memory allocation will be avoided and instead an open array of color structs mentioned above will represent the limit of the amount of colors. Each time the system is booted up the colors will reset to the standard 7.

7.2 Height Control Software

Similar to the LEDs, the height control data will be stored in an array. However because there are only 4 discrete heights, the software will store an array of integers from 0-3. There will be three arrays used to successfully update and store height data. The first is the current position array. Each height will start at 0 and this current position array will be updated to store the most recent height data processed. The second array will be the new position array and be used to store user/algorithm generated data of what the next position will be. After successfully adjusting the heights, the new position array will be swapped to become the current position array, saving memory usage during runtime. A third array will be used to store the amount of adjustment needed for each height rack. By taking the difference of the new array against the current position, the old array will be generated in one loop and contain the integer number of levels the rack should go up or down.

To adjust the height of each rack, the software will go one motor at a time. First, it will find which towers need to go up, activate the corresponding solenoids, and then turn the motor clockwise. By using the amount of time and rotation, the software will run it with an algorithm that calculates the distance traveled using experimental data. This works because the motor is slow and only runs at one speed, so should be accurate with acceptable error. Once the desired height is reached. When any one solenoid reaches its height, an interrupt will be called that deactivates the solenoid. Finally after all the solenoids are deactivated the heights should be in their correct positions. It will then do the same for racks needing to be lowered, and follow the same protocol until all racks are in the correct position. Once the motor is no longer needed, the GPIO pins will be set to deactivate the current motor and activate the new one, until all motors set the racks to the new height.

7.3 Motor Drivers

Using the chosen L298N H-bridge motor drivers, each of these drivers will be able to control up to 2 motors. The software will control this by using the 4 IN pins on the driver. Because we will only be running the motors at full power due to low rpms, we can set all enable pins to 5 volts and control each motor using 2 pins only. To control which motor is active, there will be a variable holding the information of which motor should be on, and using this variable the software can swap out the information using the variable. For example, a variable name motor_active will hold the integer number of which motor is active, 1-8, and by passing this into a run motor function it will turn that specific motor either clockwise or counterclockwise. Another parameter will be passed into the function indicating the direction the motor should spin in. So, in conjunction with the height control, the software will activate all the solenoids for the racks moving in the same direction, and then call the motor function on the proper motor and direction. The motor will stay on for the duration with the solenoids deactivating at their proper heights. For the opposite direction, the motor driver software will apply reverse polarity on the IN pin to rotate the racks downwards with the same solenoid action for each rack. Thus the software achieves full control of the motors via the L298N motor driver.

7.4 Solenoid Controls

Solenoids are controlled through the correct selection of GPIO pins, and the software must take into account the decoder/demux being used. A function to translate the output into which pins to activate will help increase readability of the code and create an easy way to communicate with the solenoids. Using the height control data received for the new data the correct output to the solenoids can be determined. An 8 integer array will be used to store data for a column of gears attached to a single motor. With each motor getting 8 solenoids, the adjustment array mentioned in section 7.2 will be read 8 values at a time. Using the 8 integer array, the position of the solenoids will correspond to the position in the array, so index 0 will be the furthest solenoid from the motor. When reading each of the 8 values, if the value is a positive non zero number it will get passed into the correct position in the array. This will create an array where a positive value means the solenoid should be activated. Using this array, we calculate how the selector

pins on the decoder should be activated to achieve the proper output that will send the signal to the solenoids. When the motor starts spinning, there will be a task to read the height value and adjust the table as necessary. As a value gets incremented to zero, an interrupt will be triggered to adjust the selector pins to deactivate the solenoid that's finished its height adjustment. Once all values in the array are zero the software will return to read each negative number from the adjustment array into the 8 integer array. The exact process will be repeated, however instead of the task decrementing values, it will increment until they're zero. When zero the same process of calculating the new selector pins on the decoder will be called to deactivate the solenoid. When all values are zero, the entire motor row will have been adjusted and the software will move to the next motor.

7.5 Runtime User Interface

After initializing the software using the usb cable on the PCB, we need a way to interact and upload datasets to be displayed. Given the wifi capabilities on the ESP32, a web app will be developed to upload data via a local wifi connection. Through the web app, we will be able to send all color data to the software, along with the different heights. A conversion will be done on the web app to convert all color data into the correct array of tuples to indicate the correct colors. For height data, all that is needed is an array of integers varying from 0-4, so a validation check will be done on the UI along with the initialization checks on the ESP32 to prevent failure in the system. After inputting the data to be displayed control will be passed onto the above software subsystems.

Chapter 8 - System Fabrication

8.1 PCB Layout

We designed our PCB using Fusion 360. Some adjustments were done with changing the trace width and adding more features that don't come with the MCU chip. All PCBs include ground panes and labels for efficient connections. We added jumper connections at different places for easy testing. We placed regulators in a more isolated area away from sensitive components to reduce noise. We also placed capacitors close to relevant devices to reduce the size of the current loops. We also added resistors at power pins to make sure the current draw wasn't too large in regard to power consumption. Originally, our first PCB design was thought to be good until we found that some of our components didn't have footprints for our PCB board. We also added header pins for different connections like USB to UART so we can send data directly to our MCU via USB cable.

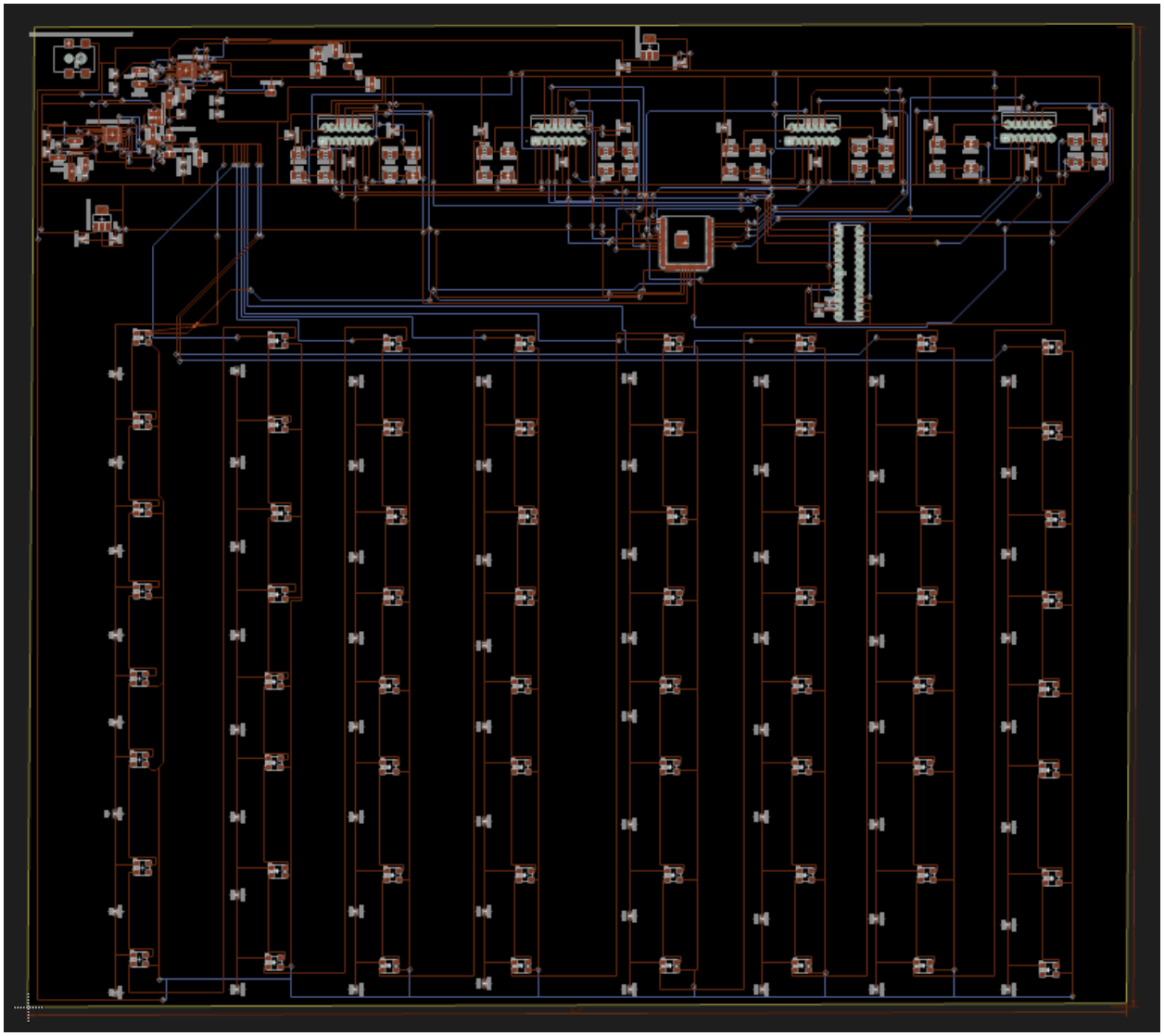


Figure 8.1 *Overall PCB Layout*

In figure 8.1 we can see the PCB as it will be laid out. This picture is very zoomed out to accommodate for all parts but later images will show the components more zoomed in. The bottom majority of the PCB is the LED array which also forms how the motors will be laid out with the rods. Each LED represents one of the units in the grid where the gear and the columns will be placed that change in vertical height. Above that is the ESP32 with the pin expander being the larger slender component next to it. Above the ESP32 are the four motor drivers. These include the diodes and everything necessary with this circuit to allow the motors to move in both directions. Lastly, the top is where the two UA78M33 regulators are to power the motor drivers and the ESP32, with the top left corner being the LM25116 regulators for the LEDs and the motors. The component in the top left corner is the adapter for the AC/DC converter that will connect from the wall. The layout of the PCB was to allow for that connection to be made and then flow the current and power downstream from there to the hidden components that make the grid function.

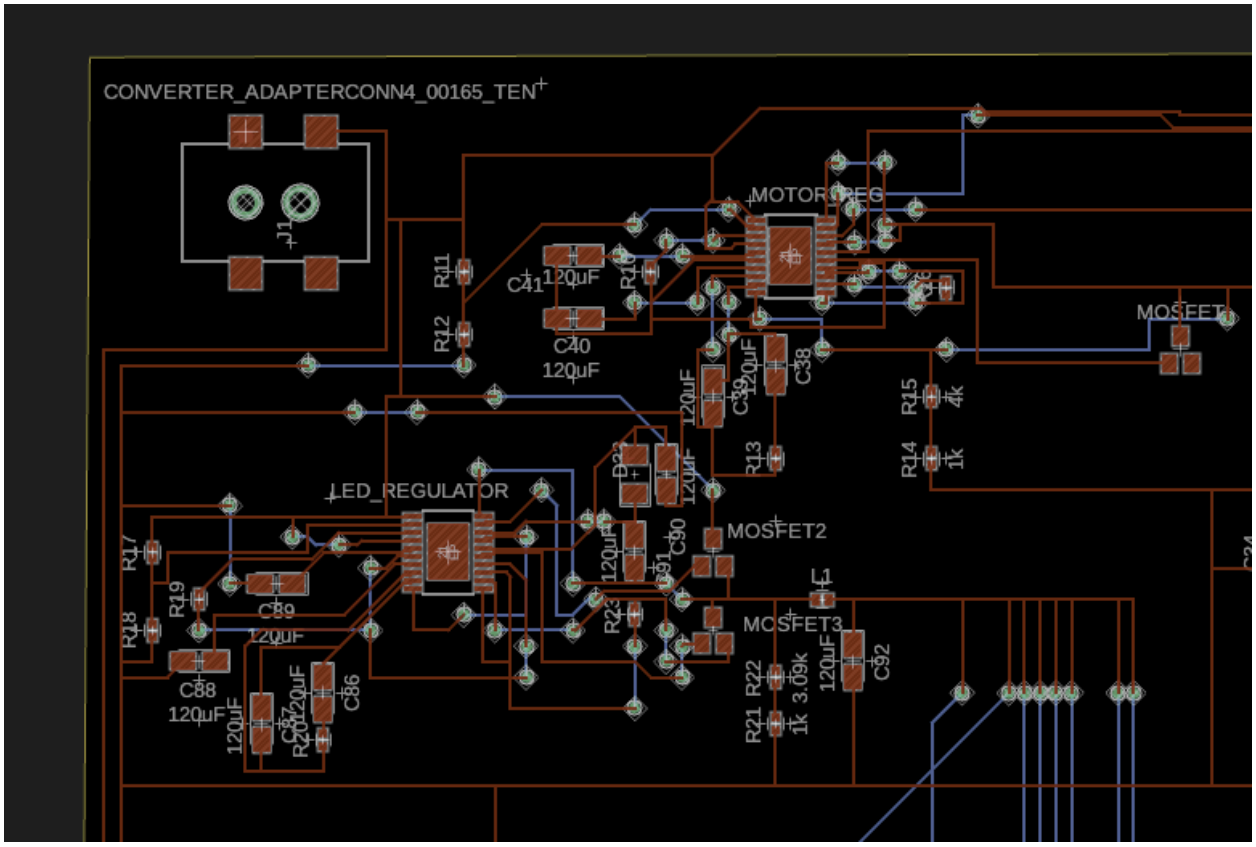


Figure 8.2 *Motor and LED regulators*

Figure 8.2 shows the top left of the PCB where the adapter and the two LM25116 regulators are placed. Since they are switching regulators, there isn't as much of a worry of heat dissipation so they can be placed rather close. We also made sure to place the adapter so that it is facing outward and a connection can be made through the CAD designed housing.

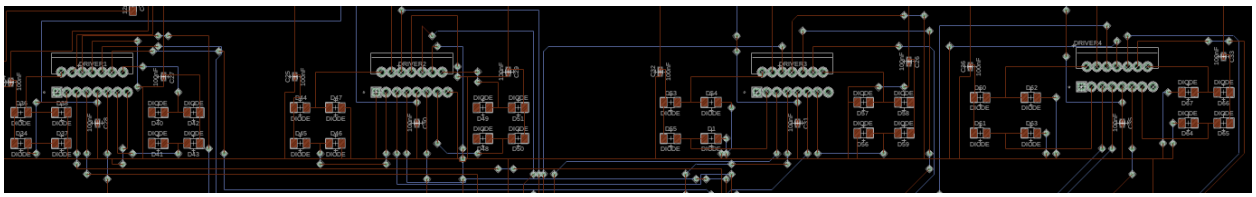


Figure 8.3 *Motor Driver Configurations*

Figure 8.3 shows the motor driver connections as they will be placed. They are oriented very similarly to the schematic such that they are placed next to each other. Their power comes from above from the UA78M33 which is not pictured here.

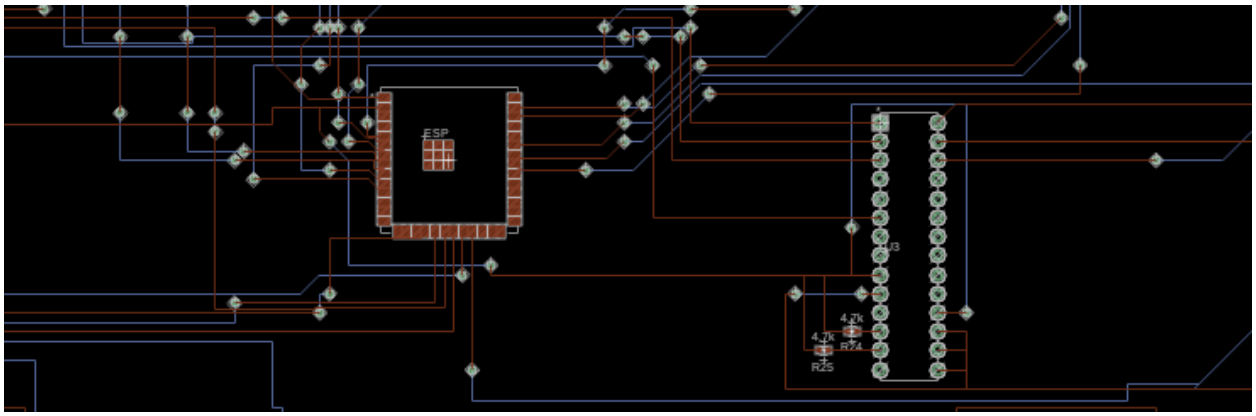


Figure 8.4 *ESP32 and pin expander configuration*

Figure 8.4 shows the ESP32 and its pin expander. All connections from the motor drivers are properly shown here. The pin expander is properly powered by the ESP32's 3.3V trace. The expander also has the pull-up 4.7k ohm resistors needed on SDA and SCL pins.

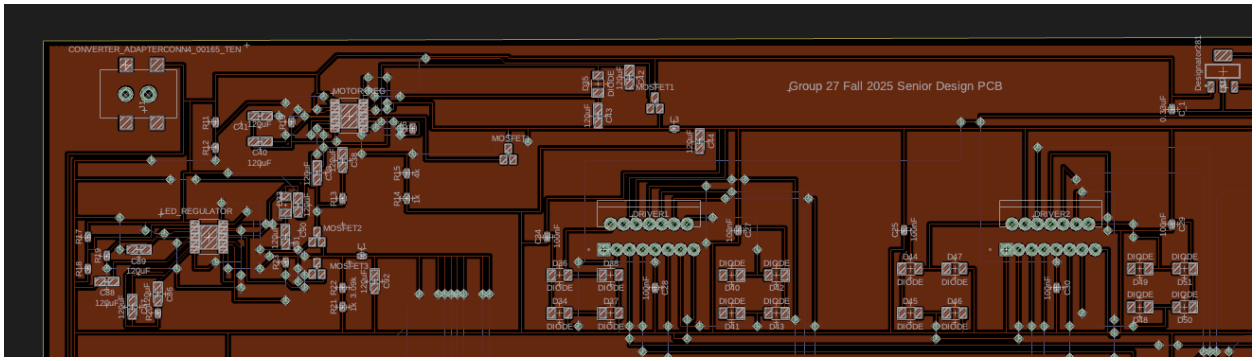


Figure 8.5 *Ground plane placed with group name and wording*

Figure 8.5 shows the top left of the PCB after the polygon pour. The polygon pour was named GND. The isolation was set to 20 mills, similar to the junior design project.

8.2 CAD Diagrams

For the mechanical aspects of the topology grid, CAD models were created to meet the 3D printed project requirements. Multiple components are required to successfully control the height and rotation of eight different towers for one motor. It was decided a rack and pinion design would be utilized, along with a bearing to allow for individual addressable towers at different heights.

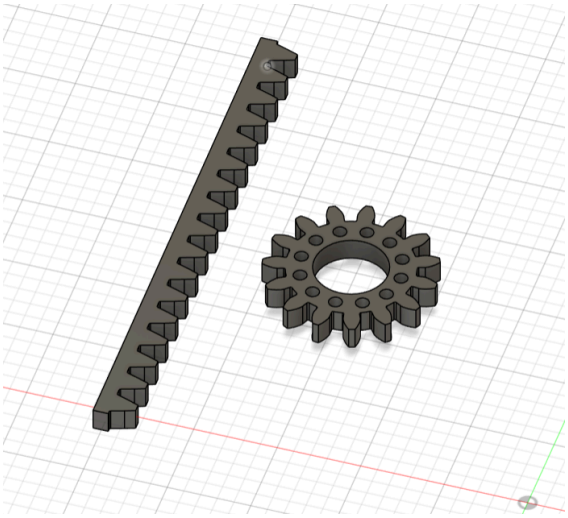


Figure 8.6 *CAD model used for rack and pinion design. Bearing was inserted inside the gear for smooth rotation.*

Another vital component was the shaft for the motor rod. The motor rod needed to meet the measurements of the standard 608 2RS bearing, to allow for free movement when the solenoid was not engaged. Also a fitting was added to attach to the D-Shaft motor to drive the multiple rack and pinions. Included in the motor shaft is the model for the solenoid stand-up used to align the electric actuator with the gear holes.

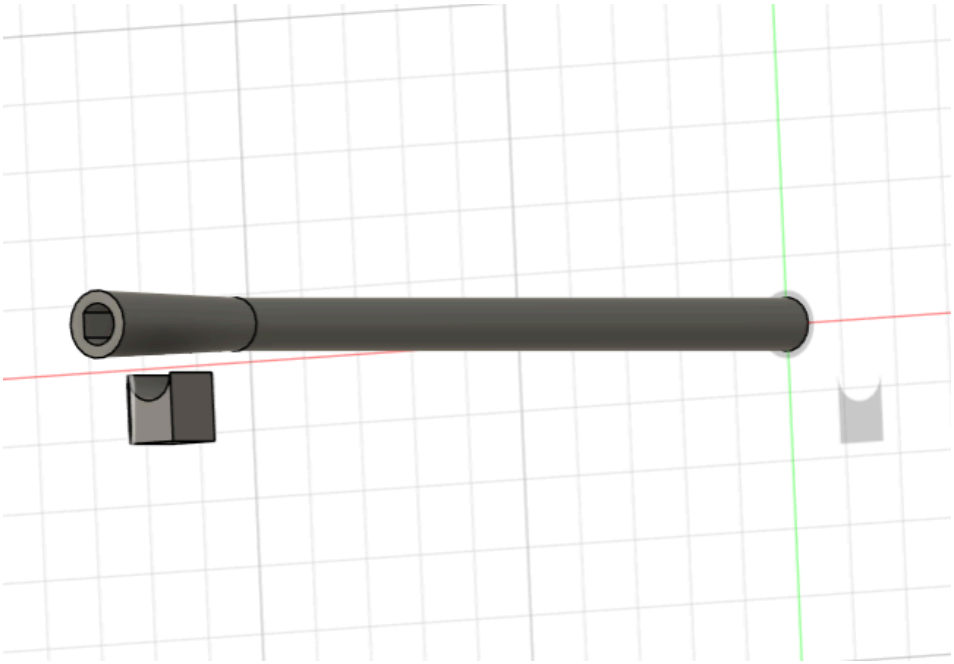


Figure 8.7 *Motor shaft and solenoid stand-up. Solenoids are placed on top of the stand-up.*

To keep the racks for the rack and pinion held up, a special mechanism was used in the CAD design. Divots were added to the back of the racks and the roof was added with a flap to allow for motion up or down, but remained still when not moving. A sphere was added to the back of the flap to increase the accuracy of where the rack would rest after movement. Guide rails helped steady the racks to prevent wobbling or jumping of the gear teeth.

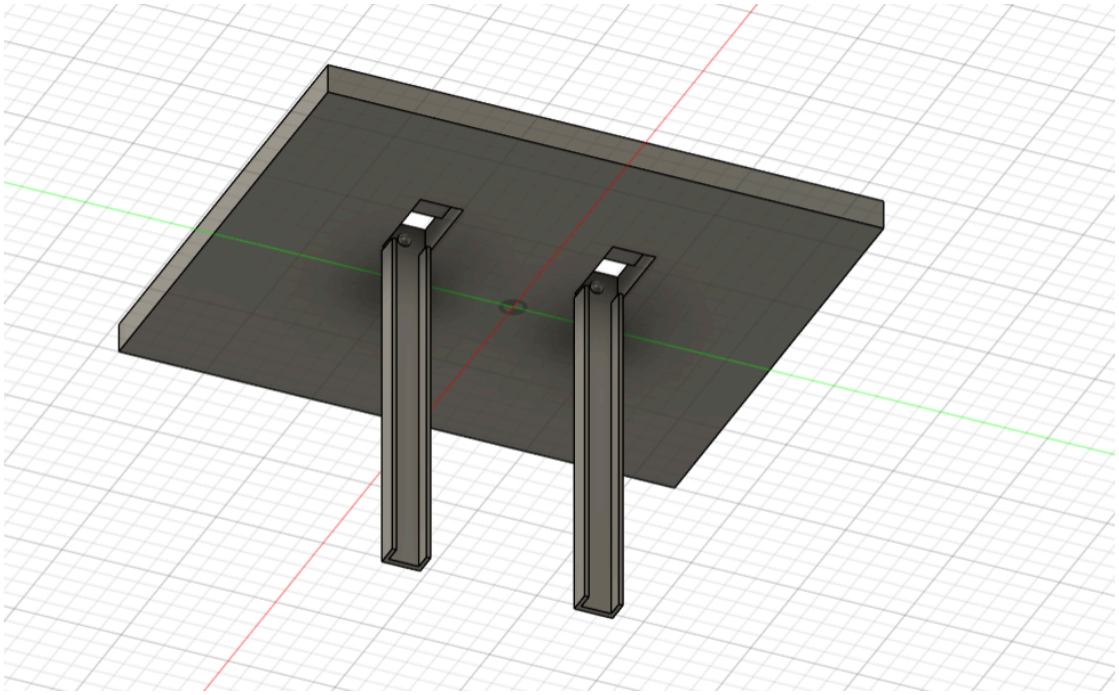


Figure 8.8 *The roof for the 3D grid. The flap and sphere can be seen in the CAD model at the entrance holes.*

When the motor shaft rotates and pushes the rack through the opening, the increased force allows movement up. However, when the solenoid disengages the flap is sturdy enough to prevent the rack from falling back down through the hole. In this way we can control the individual heights and let the solenoids be powered for less time, saving power consumption and component lifetime.

Finally the last CAD model used was the supports for the motor itself and the slip ring. Because the motor shaft was offset from the center of the motor the support must be higher than the slip ring fitting. The blue axis line was used to make sure of proper alignment. The motor rod attaches to the slip ring to stop it from hanging. Attaching the rod to the slip ring keeps the centerline through the rod, which prevents wobble to achieve smooth operation. A bearing may be added in future iterations to increase this stability as the rod length grows to incorporate more rack and pinions.

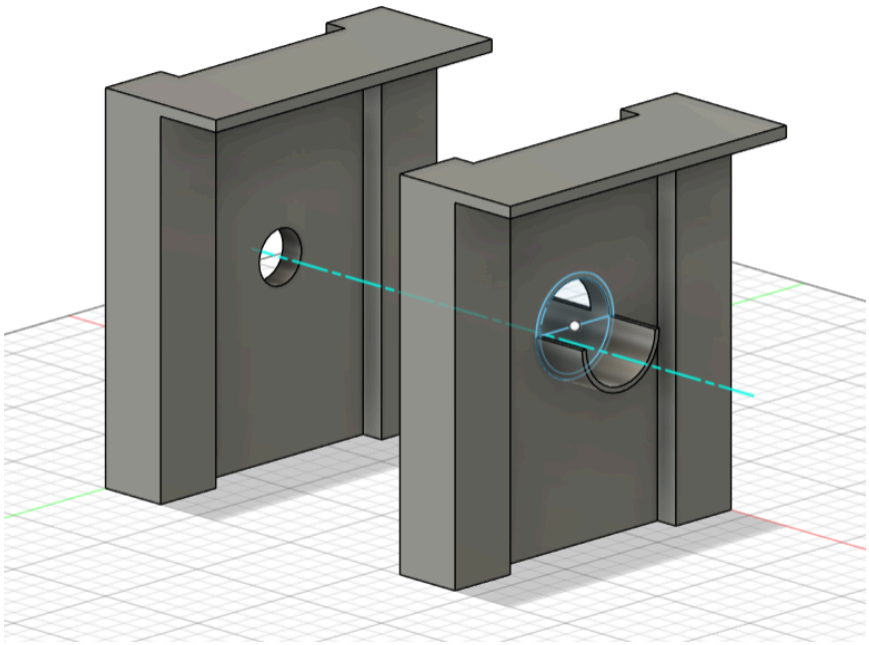


Figure 8.9 *CAD of the supports to hold the roof, motor, and split ring. The blue axis is where the motor shaft rests at its center.*

Chapter 9 - Prototype and System Testing

9.1 Hardware Testing

9.1.1 ESP32

Upon beginning testing with the ESP32 DEVKIT V1, a simple powering of the microcontroller was used to ensure it worked properly. For this specific ESP32 model, we were able to use a micro USB-B cable plugged into the onboard port. After powering the board, an LED indication tells us our microcontroller is properly powered. Another test was performed to check that the voltage of the output pins was correctly showing our expected logic voltage levels of 3.3V. We used a multimeter on a few GPIO pins to check that this was the case. All was confirmed to be working as expected, giving the right values for all of our testing points. The boot and enable buttons built into the ESP32 were also used to check that all functionality was working correctly. After performing all these tests, we could safely say that the ESP32 was working and ready for use.

9.1.2 Motor

To begin testing of our motor, we applied 12V to it using a power supply. This was the first check that told us whether the motor worked or not. When applying voltage to the motor, we made sure not to have too much current flowing through it, just to be safe. Having too much current flow could result in dangerous behavior from the motor. Our next test involved swapping polarity of the power connections. This served as our check to make sure that the motor was able to rotate in both directions. After it was confirmed to work properly, a multimeter was used to verify that the current draw with no load was within the expected range. A general check of the motor was performed to see if there were any visible or audible problems with its functionality. No weird noises or jittery movements were observed, meaning the check of our motor had been completed.

9.1.3 Motor Driver

The first test performed on the L298N motor driver was a simple voltage and ground connection to a power supply to make sure the onboard LED turned on. After seeing it successfully powered on, we could move on to our next test. This test involved providing voltage to the input pins and checking how it affected the output voltage on the two output pins. We were able to use a multimeter to confirm that voltage flipped on the output pins as a result of applying voltage to the different input pins. More specifically, when we apply voltage to IN1 we see an output voltage coming from OUT1 and not OUT2. The reverse of this can be seen when applying the voltage to IN2. This changing of polarity will be the main application of the motor driver. Another quick visual inspection was performed on the driver to make sure no part of it looked damaged or faulty. The final test was to ensure the PWM enable pin worked properly for speed control. Changing frequencies and monitoring changes in voltage helped us conclude that the speed control was somewhat operational. This was the first of our tests that wasn't fully successful though. We determined that the 12V motor wasn't able to work with anything less than 9V. Attempting to do so resulted in some noise from the motor, but no actual movement. Since the motor we went with was pretty low speed and high torque, it wasn't the end of the world to have to run it at near max power.

This control over speed and direction is important to our design, but does not fully encapsulate what the driver is there for. The driver's second important use is controlling two motors at once. Testing for this was similar to what was done previously, except this time we also tested voltage of OUT3 and OUT4. The function is exactly the same as it was for controlling one motor, so testing looked very similar as well. Reading voltage levels on a multimeter while applying voltage to the different input pins helped us confirm that the dual-motor functionality was working as it should.

9.1.4 LEDs

For the WS2812B addressable LEDs, we performed some pretty simple tests. First, a simple connection of one LED to our ESP32's Vin, GND, and GPIO pin was made. From here we were able to test voltage levels using a multimeter to make sure 5V was being output. Due to the LEDs requiring soldering, it was also important to do a visual check to make sure that no damage occurred during the process. After this check, another LED was soldered on to ensure that each subsequent LED was receiving the 5V. This was again checked using a multimeter. Proper connections of DOUT to the next LEDs DIN were checked for and confirmed to be good. Checks for proper flow of data from one LED to the next will be tested using software methods.

9.1.5 Push Solenoid

To test our push solenoid, we began by connecting it to a 12V power supply to ensure it was actuated upon powering. The solenoid was monitored to verify that it was being pushed when receiving the 12V. After this test, a switching circuit was added to make sure we could deal with more than one solenoid at a time. The switching circuit was made up of a 2N2222 transistor, 1N4007 flyback diode, and 1k Ohm resistor. The transistor served as our switching mechanism while the flyback diode was used as protection from the high voltage kickback that occurs when the circuit is broken. The resistor is used here to limit current to the transistors base. The GPIO pin connected to the solenoid circuit acted as our switch to either turn on the solenoid or turn it off. With this switching circuit, we were able to now test solenoids in a chain instead of on their own. Multiple trial runs of energizing and deenergizing were run to make sure that we could consistently get it to work. Stress testing was also done on the solenoid to see how far we could push them. They are typically rated for no more than a minute of constant actuation so we wanted to see how accurate that was. With a single solenoid, we were able to go for a little over a minute before we decided to turn it off. Obviously testing a single one of these is not the same as a system with many of them since conditions may be different, but this gives us a good gauge as to what we can expect.

9.1.6 Gear rack, Pinion, Rod, and Bearings

Testing for all the parts of the gear mechanism used for height control required mostly visual inspection and alignment checks. Due to most of the parts being 3D printed, it was important that we made sure no misprints occurred. For the purpose of testing, we mounted our rod to the 12V DC motor to make sure the fit was right. Since the bearings were the only item bought from a store, we had to verify that they fit tightly into the gear. Once we confirmed the fit to be tight enough, we were able to slide them onto the rod to make sure they would stay in place. With this done, we were then able to test how the gear moved when aligned with the gear rack. After seeing it smoothly move up and down the rack, we were confident that the fit was perfect for our

requirements. Now with all the components lined up and attached, we could see the mechanism in its finished form. Verifying with visual inspections, we were confident to move forward with the next portion of our system.

9.1.7 Slip Ring

The slip ring is an essential part of our system. Being able to feed wires through to the solenoids while the whole thing is rotating can come with some problems if not done correctly. Tangled messes of wires or connections coming loose are among these problems. To mitigate our concerns for this, we used the slip ring. The slip ring works by feeding wires from one side of it through a small hole where inside, small conductive wires maintain connections. To test this piece of hardware, we started by connecting a simple circuit with a power supply and an LED. We tested to see that even with extensive rotating of the slip ring, the LED continued to be powered. We also wanted to make sure that the slip ring wires weren't being damaged so we did a decent amount of stress testing on them. They were on the thinner side, but seemed to reliably hold form and not get damaged from simple rotation. We knew power was able to continuously flow through the circuit, but we wanted to make sure it was fluctuating in values. For this we removed the LED and used a multimeter to make sure the voltage remained constant during rotation. After all these checks were done successfully, a quick visual inspection of everything gave us the confidence that the device would be the right fit for our needs.

9.2 Software Testing

For testing of our software, we decided to go through each component and run one or two simple programs to ensure individual functionality of them was successful. For some of the components, using our microcontroller as well as the component was necessary to test it worked. After testing the components on their own with basic programs, we moved on to testing them together. The integration of more than one component will play a large role in our final design, so testing it now will prove to be very useful. Many of the programs we will run in this section will be simple, that is on purpose. Getting the most basic functionality of the components will serve as not only a good benchmark for usability, but will also help us test to see if the components are performing as they should be.

9.2.1 ESP32

The first component we wanted to test was the ESP32 microcontroller. This will be the most important part of our system, as it will be responsible for integrating every component together. To set up testing of the ESP32 DEVKIT V1, we first cloned an ESP-IDF Github framework to

our project. The actual coding and testing was done through the Mac terminal Z Shell(Zsh). Copying the framework allowed us to set up templates for our testing programs as well as use some preexisting example code. For our first software test, we first decided to go with a simple hello world program. This hello world c file seen below in figure 9.1 was one of the examples of preexisting code from copying the Github repo that we could use immediately without the need for making any changes to it.

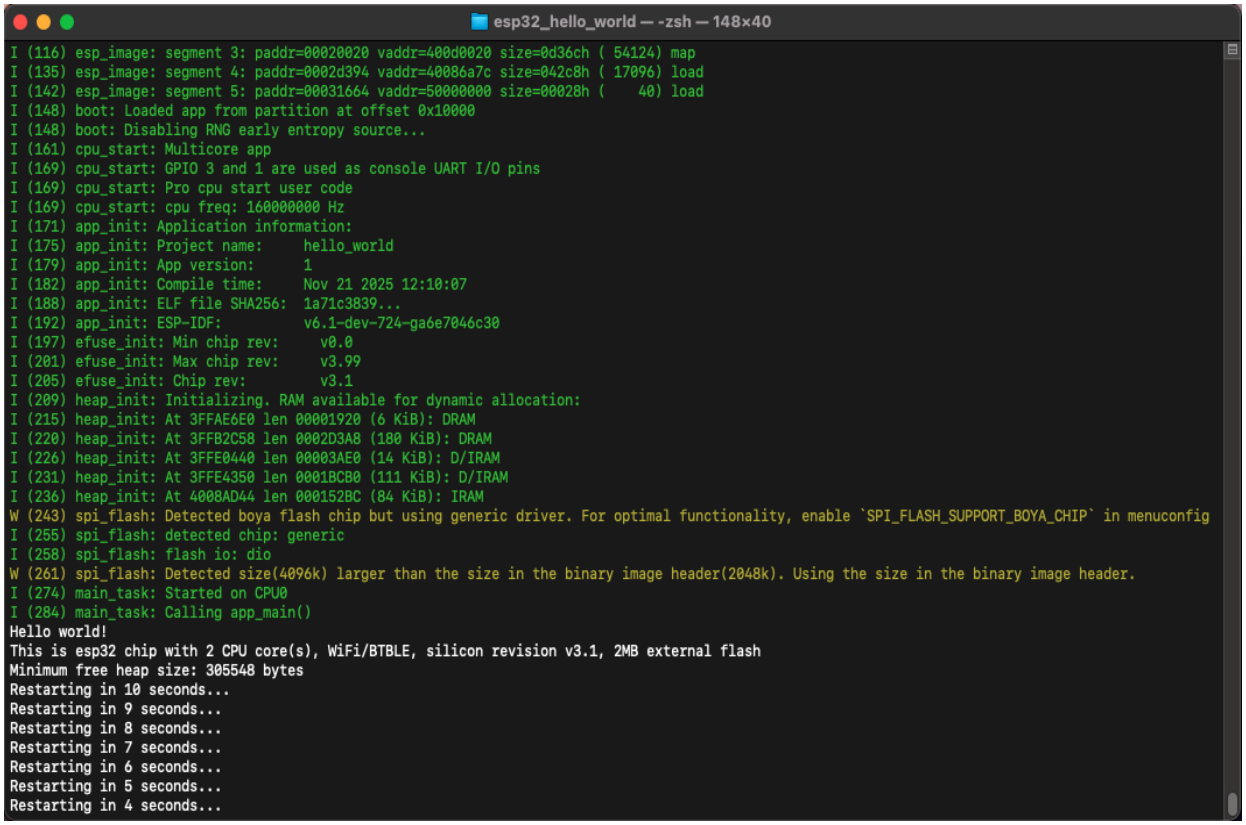


Figure 9.1 *hello_world_main.c*

As we can see, the program successfully runs and displays the hello world message to the terminal. Along with this hello world program, there are many others prebuilt into the Espressif ESP-IDF framework software. After getting this hello world program working, we moved on to getting the ESP's onboard LED to blink using another preexisting program. For this, the new idf directory was created and the blink_example_main.c program seen in figure 9.2 was run. The functionality of the code is very simple, but will help us test to make sure the ESP is working properly. The built in LED for the ESP32 is assigned to GPIO pin number 2. The code will call the blink_led function and toggle the state of the LED. When flashing to the port, we were able to see it alternate from on to off. As well as toggling the state of the LED, we also see a message logged with the state the LED is currently in.

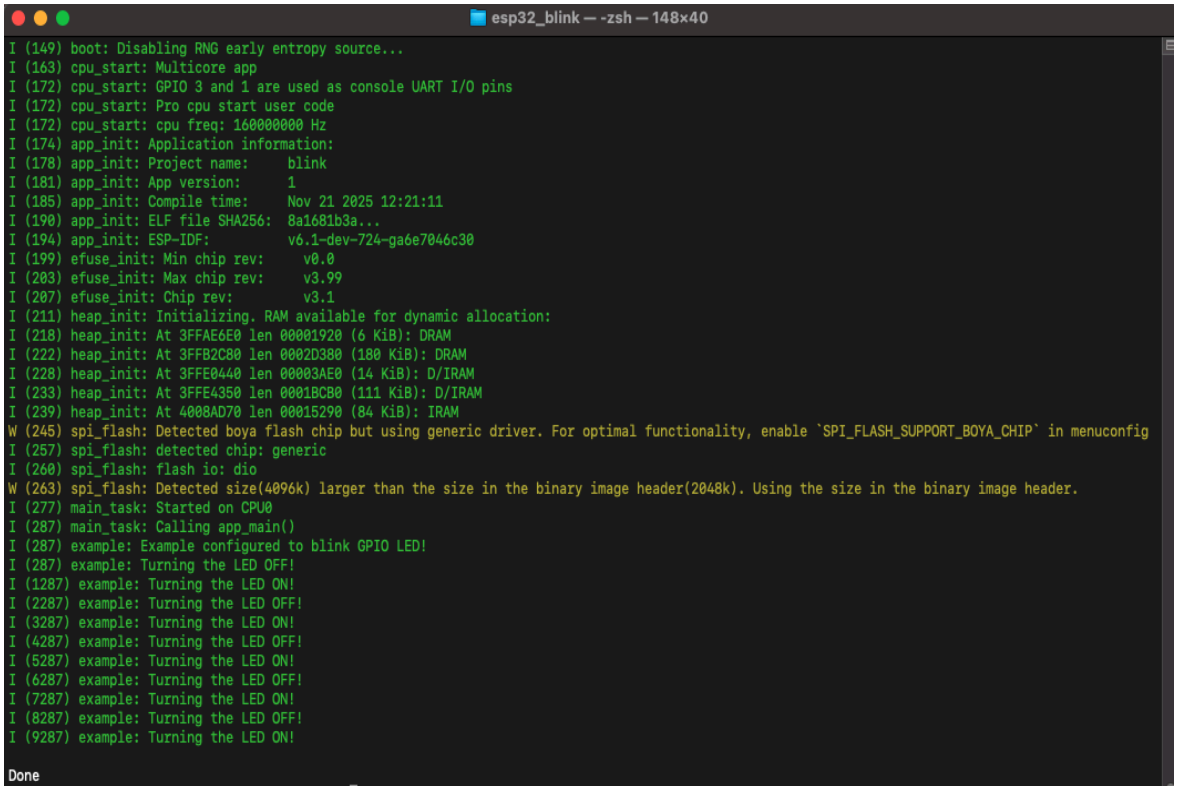


Figure 9.2 *blink_example_main.c*

For the next bit of testing, we will use the ESP32 as well as each peripheral to check that software runs properly. Making sure each component can be integrated with the ESP is important to further enhancing our design.

9.2.2 Motor

To test the motor by itself we decided to implement it using the ESP's LEDC, or LED Control output. This was used because of its PWM functionality, which is an important part of our speed control. Our motor is a 12V DC motor that can perform lower than 100% by using lower input voltage to it. Testing began with a PWM frequency of 5 kHz, and worked its way up as the program progressed. We wanted to test what percentage of max speed worked best and which worked in general. The program motor_test.c can be seen below in figure 9.3 going through each of the PWM frequencies that equate to percentages of the max power. The problem seen during this testing was related to power under 75%. When doing 25% or 50% of full power, the motor would not work as it was supposed to. This meant that for all future motor use, if we were to use PWM speed control, it would have to be 75% or higher.

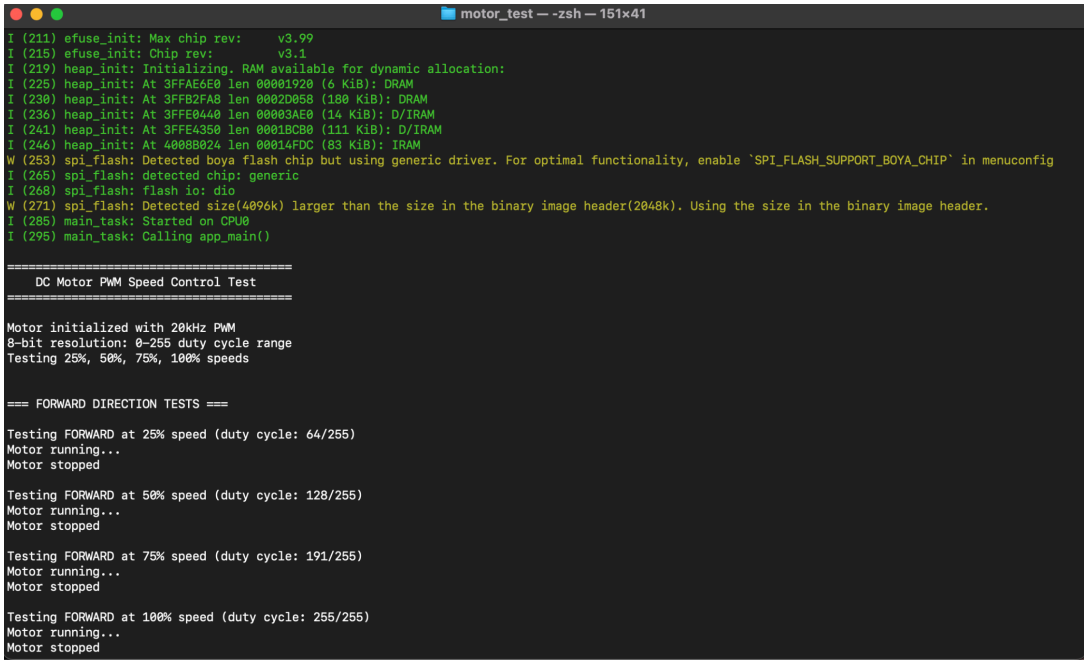


Figure 9.3 *motor_test.c going through PWM frequencies*

When going through each of the frequencies during testing, we see a problem with the lower ones. When having the motor powered with less than 75%, we see some unwanted behavior. In the case of the 50%, we see a movement in one direction but not the other. A noise comes from the motor, but no rotation occurs. When the motor is being powered with 25% of the rated voltage, we do not see any movement. We weren't likely to run our motor at 25% so this is not terribly important, but it is good to know for reference.

Along with the motor is the motor's driver. Using the L298N motor driver is key to allowing not only speed control, but direction control as well. The motor driver will be looked at in the following section in more detail.

9.2.3 Motor Driver

The L298N motor driver has a significant amount of functionality that will be useful for the purposes of our project. One of the biggest things we are looking for from the motor driver is control over our DC motor. The driver allows us to control the speed and direction of two different motors simultaneously. The driver works by sending signals to one of two input pins which in turn send voltage to one of two output pins. These output pins are fed to the pins of the motor, where reversing polarity will cause them to rotate in the opposite direction. Looking at figure 9.4 below, we can see the motor alternating from forward to backward, stopping in between each.

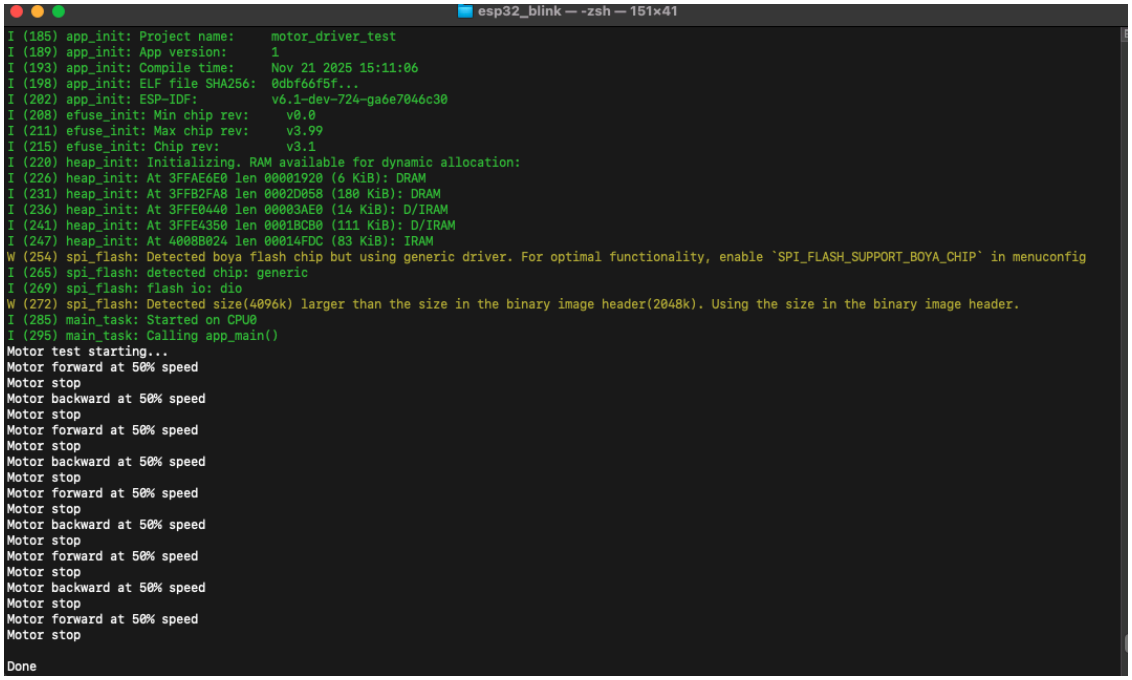


Figure 9.4 *motor_driver_test.c for forward and backward*

Switching the motors rotation direction is an important function, but isn't the only thing we are looking at for testing. We also want to test to make sure the driver is able to control both the connected motors at the same time. For the program to test this, we ran the motors individually and together. This helped us confirm that they worked for all cases we might encounter. GPIO pins were connected to the 4 input pins and setting them high or low would result in different motor behavior.

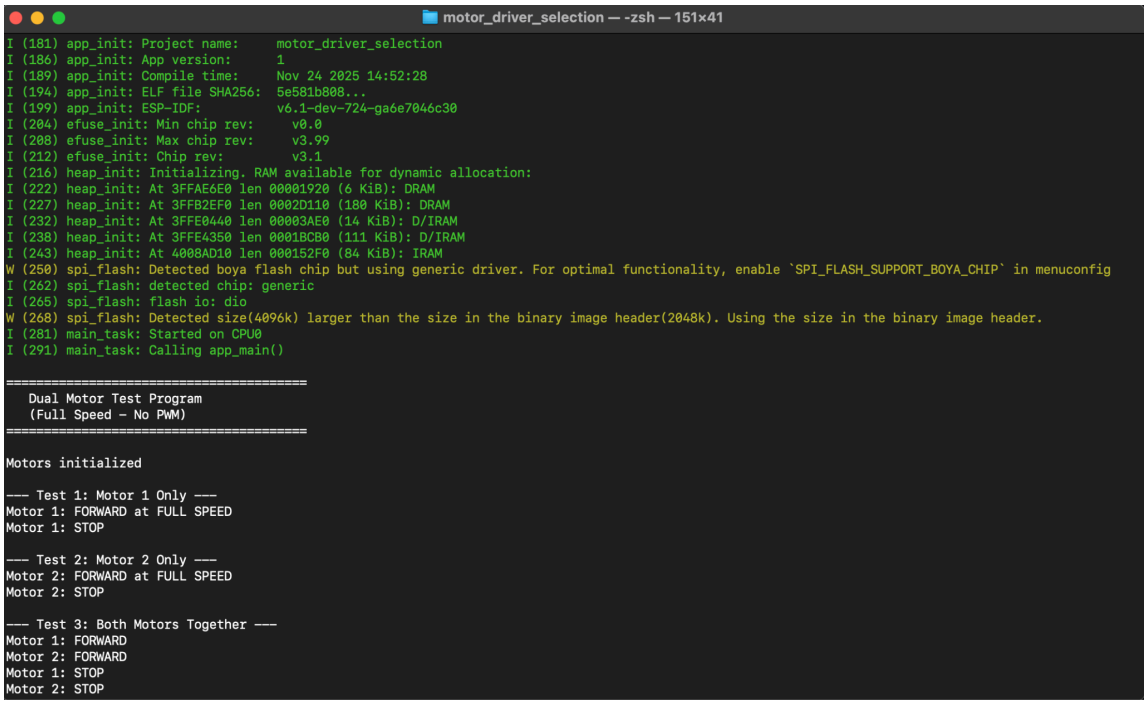


Figure 9.5 *motor_driver_selection.c to test different motors*

As we can see from figure 9.5 above, the motors first rotate individually then together. This dual-motor capability is a big reason we decided to go with this specific driver compared to the other options we had. This testing of each of the motors gave us the confidence to move on to the next portion of our system testing.

9.2.4 LEDs

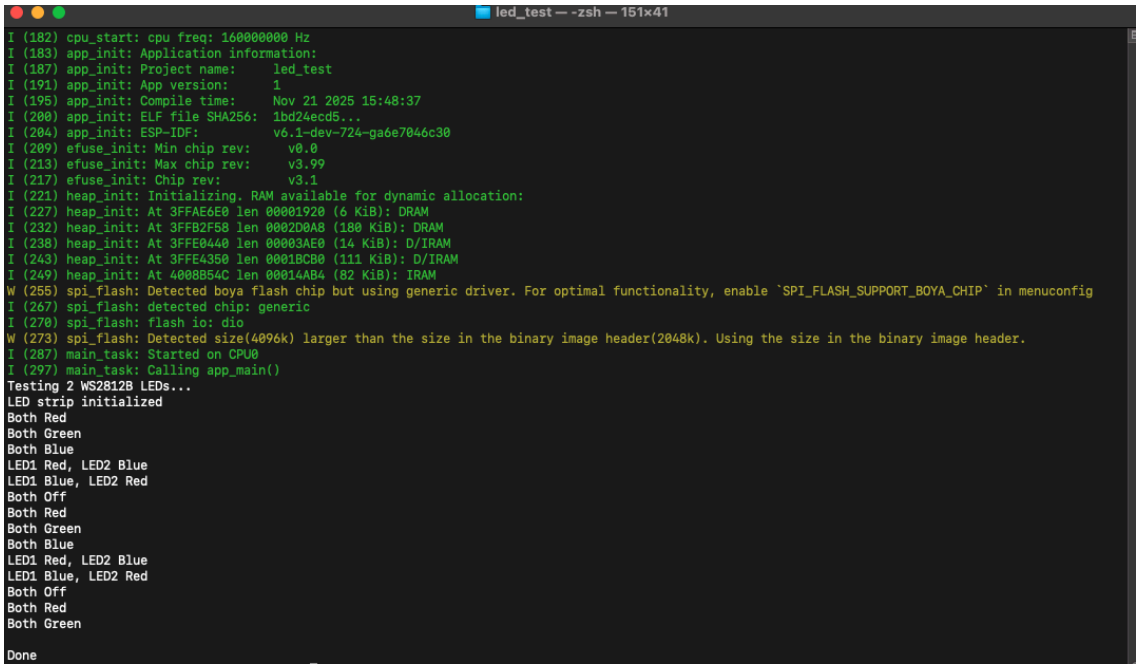


Figure 9.6 *led_test.c cycling through colors for two LEDs*

For testing of the WS2812B addressable LEDs, we decided to connect two of them to the ESP32s GPIO, 5V, and GND. With these connections, we were able to send data to tell the LEDs which color we want to display. Looking at figure 9.6, we see the program cycling through different colors shown on the LED. Since we are using addressable LEDs, we have the ability to pick whatever color we want for each specific LED. This means we aren't limited to having them all be the same color.

9.2.5 Solenoid

For solenoid testing, we decided on a simple program that powers one, then powers the other, then finally powers both. We are able to achieve this by sending a signal through the GPIO pin attached to the switching circuit. When the transistor's base is signaled high, we allow power to flow, actuating the solenoid. This can be done for however many solenoids we wish to extend and retract.

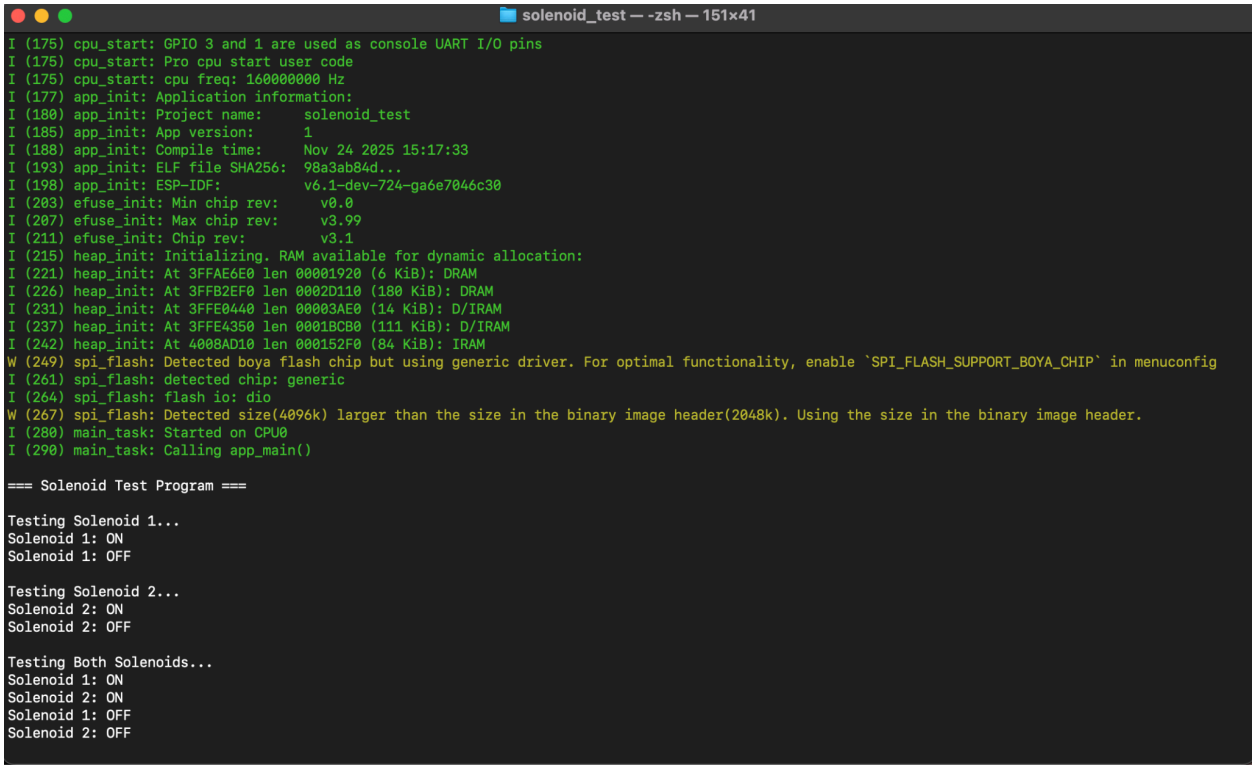


Figure 9.7 *solenoid_test.c showing different cases for solenoid usage*

9.2.6 Slip Ring

Software testing for the slip ring was pretty insignificant. What we tested here is exactly the same as what we tested for the motor. We checked that when the motor is rotating, the slip ring is keeping wires untangled. The code used to test the slip ring is just the motor test code from figure 9.3. Quick visual checks to ensure all lines stayed connected were performed. Also, testing of the solenoid during rotation ensured that connections were solid.

9.3 System Integration

Now we need to come up with a plan for how we are going to integrate the hardware components with each other for the final design. Getting parts to work on their own is one thing, but combining into one cohesive system is a whole other problem. In this section we will cover our plan for integration of our system.

9.3.1 Integration of DC Motor with Gear Rack System

To begin integrating the DC Motor to work with the gear rack system, we need to determine a way to control position accurately. Since our goal is to have four discrete height levels, we need to be able to have the motor move the gear rack by a precise amount every time. Our method for doing this involved calculating how long it would take for a full rotation and using that in combination with the size of our rack. We were then able to develop an algorithm that could move the gear rack precisely enough for our purposes. This algorithm will be precise, but not perfect. To make up for the minor error we will see, we also made flaps in the top part of the housing to lock the rack into place at the specified height level. For a simple calculation, it took around 3 seconds to go the full way around. Using this, we can assume it will take around 30ms to rotate 1%. This will be used in our calculations for motor movement time. Diving deeper into the algorithm we developed, we can see its steps:

1. Create a variable for our current position that will be used to tell the program where we are relative to the minimum and maximum heights. This variable will be used alongside another variable for our target position. This target position will be given to the program from whatever input form is decided upon.
2. The next step of the process is calculating how much change we want the system to go through. We can calculate a position delta to tell the system how much we need to change by. This will come from the difference of the target position and the start position. These values will be relative to the program.
3. The delta value we receive will be used to not only tell the system how much movement, but also in what direction we move. If the change is positive then we know our target is higher up and therefore we must extend the rack. If we have a negative delta, we know that our target is lower, and we must lower the rack.
4. Now that we have a relative value for how much movement we want, we can move onto converting that into motor rotation. Using our developed algorithm, we can take the absolute value(in case the delta is negative) and multiply it by 30. This 30 comes from the calculated value earlier. Determining a 0-100 value for delta and multiplying it by 30 should tell our motor how much rotation is needed to get to the desired height.
5. Precautionary measures will be taken to ensure that the system isn't being put under too much stress. For rotations lasting longer than half a second, limits will be placed, and rotation will be broken down into steps. In the case where the calculated delta exceeds 100, whatever the value is will floor to 100.
6. After movement is completed, the current position variable will be updated to match the new position. This will set up the node to be ready for future height change.

This system ensures that our system is not trying to extend/retract the gear rack further than it should. With the safety measures in place, we should be able to safely have the motors working with the gear rack and not worry about mechanical shock or destruction of the system.

9.3.2 Integration of Solenoids with Height Control System

Implementing the height control system to work with the solenoids will take some coordination between components. The things we are concerned with when dealing with the solenoids are:

making sure it's actuating when we want, ensuring it’s not getting caught and raising unwanted nodes, making sure the solenoid isn't actuated for too long, and getting alignment correct with gears. Since we are using an external switching circuit to determine which solenoids are activated, we also need to take that logic into account. For our system, we have a unique solenoid at each node of the 8x8 grid. When the solenoid is actuated, movement of the gear rack is enabled. For proper setup of this system, we took the following steps:

1. The first thing we did was define each of the solenoid based on the nodes location in the grid. For example, the first solenoid would be defined as position 1(written differently in code).
2. Before any movement of the system, we determine if the node in question is one we want to move or keep in place. If we want to move it, then we extend the solenoid to fit into the gear hole.
3. Pulsing the solenoid at this stage can ensure that we aren’t getting caught on a part that we don't want to.
4. From this point, the motor may rotate in whichever direction we are targeting. Once arriving at the desired height, the motor may stop and the solenoid may retract.
5. The built-in flaps of the housing will catch the rack teeth and keep it in place without the need for a solenoid holding it.
6. For going back to the original position, the solenoid will be signaled to extend and the motor will move based on calculations.

Having our system function in this way ensures we have reliable and repeatable movement of the gear racks. This also helps us to make sure the solenoid isn't being overworked as a result of needing to be held in position for too long. Since we already limit motor movement to half a second bursts, we don’t ever have to worry about getting near that 1 minute maximum its rated for.

9.3.3 Integration of LEDs and Height Control

Integrating the LEDs with the height control will involve interacting with the stored arrays in the runtime software. As the topology grid's goal is to display data in three dimensions, we need both the height control and LEDs so be able to display independently of each other. So a discrete height is not indicative of a certain color. The same heights may have different colors, allowing for unique data to be displayed. As mentioned two data arrays will be used to create this effect. LED colors are stored in tuples which each store three values ranging from 0-255. Each unique tuple is a different color that will be displayed, so an array of tuples will represent the colors being transmitted. When the new LED data is inputted, a direct swap of the data makes an easy transition to the new colors. Due to the speed of refresh, more care must be given to the height control. For the height control, there will be 4 discrete heights, so an array of integers ranging from values 1-4 can represent what height each rack is displaying. When new data is input for a new display of heights, the new array will be subtracted from the current state array. This will create the amount each tower needs to adjust in a neat array. For example, if the current state of a tower is 4 and the new data wants it at 2, 2-4 would result in -2. So in the array a value of -2 means lower by 2 states. This also prevents lower than zero calculations, as there can only be positive values using this subtraction method.

9.4 SD2 Plan

For our Senior Design 2 plan, it all starts with the PCB. In the first couple of weeks we need to get our PCB checked off and approved for purchase which will take another 2-3 weeks to be delivered to us. While waiting for that, we are planning on ordering all of our other components and making our web application for sending in data to our MCU. Once we get our PCB delivered we will try and get all the soldering done within a week and testing done in the subsequent week. If all of the testing goes well and our PCB is working as expected, we will begin final testing and minor improvements or fixes. If our PCB doesn't work as expected or components aren't up to par, we will have to reorder and revise our PCB/components and unfortunately have to wait for the delivery again. Once it comes back in we retest it and hopefully everything works as expected and we can begin final preparations for our final demo. Below is a table for an expected timeline of our next semester.

Task	Start Date	End Date
PCB Approval	01/12	01/25
PCB Ordering	01/26	02/15
Web Application Production	01/26	02/15
PCB Soldering	02/15	02/22
PCB Testing	02/22	03/01
PCB Revision (If needed)	03/01	03/08
PCB Reordering (If needed)	03/08	03/29
PCB Retesting (If needed)	03/29	04/05
Final Improvements/Fixes	04/05	04/19
Final Demo	04/19	05/05

Figure 9.8 *SD2 Schedule*

Chapter 10 - Administrative Content

10.1 Budgeting and Costs

An initial budget of $250 dollars is provided by DRACO labs. This will cover the LEDs, MCUs and power systems. Anything over this budget is covered by group members, giving additional incentive to keep the costs low. Given the cost constraints, 3D printing will be heavily used to provide cheap and customizable parts for the 3D grid. Another way this project keeps costs low is by focusing on low voltage/low pressure operation.

The two most expensive modules in the project will be the motor system and the LEDs. One reason the height control system will have such a high cost is due to the amount of different components needed when designing it. These parts have lots of expensive metal and seals that drive the costs upward. For the LEDs, one of the requirements for the project is to have each LED be addressable. This introduces complicated chips for each LED that also increase the costs.

For these reasons the project was given a budget of $500. A focus on low cost parts and solutions will help keep the project within this allotted amount.

Modules	Cost
LEDs and Connectors	$125
MCU/Controllers	$20
Power System	$30
3D Printed Parts	$40
PCBs and Shipping	$100
Height Control System	$180
Total	$495

Table 10.1 *Project Budget*

10.2 Bill of Materials

Component	Name	Price	Quantity	Cost
ESP32-D0WD-V3	ESP32	$1.84	1	$1.84
L298N	H-bridge Motor Driver	$1.01	4	$4.04
NFP-GM37-3429-EN	DC Gear Motor	$20.00	8	$160
Adafruit #412	Push-Pull Solenoid	$7.50	64	$480
UA78M33	Linear Regulator	$0.62	10	$6.20
ALITOVE 24V 6A Power Supply Adapter Converter	Barrel-Jack Power Supply	$22.79	1	$22.79
LM25116	PWM Buck-Controller	$7.04	2	$14.08
WS2812B	Addressable Smart RGB LED	$14.99	1 (100)	$14.99
Slip Ring 2A	8-channel Slip	$6.80	8	$54.4

240V	Ring			
Total Cost				$758.34

Table 10.2 *Bill of Materials*

10.3 Project Milestones

Task	Start Date	End Date
Group Formation & Brainstorming	08/18/2025	08/26/2025
Divide and Conquer & Committee Formation	08/26/2025	09/05/2025
Meeting with Instructor	09/08/2025	09/08/2025
Selection of Materials	09/12/2025	09/19/2025
Schematic Design	10/03/2025	10/20/2025
Midterm Milestone	09/25/2025	10/31/2025
PCB Design	10/05/2025	11/01/2025
Prototyping	11/05/2025	11/12/2025
Final Report	11/18/2025	11/25/2025
Ordering Components	SD2 - TBD	SD2 - TBD
PCB Assembly	SD2 - TBD	SD2 - TBD
Testing	SD2 - TBD	SD2 - TBD

Revision of PCB Assembly	SD2 - TBD	SD2 - TBD
Retesting	SD2 - TBD	SD2 - TBD
Final Demo	SD2 - TBD	SD2 - TBD

Table 10.3 *Table of Milestones*

Chapter 11 - Conclusion

The three-dimensional data projection grid shows a huge step forward in physical data visualization, making three-dimensional datasets into tangible and interactive displays. Through an implementation of an 8x8 LED grid with mechanically adjusted epoxy towers, we can demonstrate that complex spatial data can be represented in a way that removes the need for digital and screen based interaction.

Our implementation addresses the challenge proposed by Draco Lab and Dr. Borowczak. This challenge is to create a prototype that makes three-dimensional data more accessible and easier to understand. By combining addressable LEDs, precision motor control, solenoid-activated gear systems, and illuminated epoxy towers capable of four distinct height levels, we have created a platform with applications spanning education, topography, and multi-variable data analysis.

We successfully took on the challenge of electrical design, mechanical design, and software integration for this project. We were able to discover and overcome limitations regarding power consumption, precision and system synchronization through research and component selection, breadboard prototype testing and final implementation. Before choosing our final PCB and purchasing, we were able to revise it based on our results from early testing.

Our current 8x8 grid shows the idea of this concept, however, this project serves as a prototype and template for future expansion. The modular design of our project allows for scalability to larger grids, higher levels, and even improved visual features. We can see our project being used in a classroom setting for students to visualize data, in geological settings where terrain and topology can be displayed and in business environments for multi-variable data visualization.

Our three main concepts for this project are the LEDs, the motor and our epoxy towers. With the LEDs being at the base of every tower and the motor pushing up each tower to specific heights. With this, we are able to take a basic grid of 64 lights and expand them upward to display whatever our minds can come up with, whether it be a physical layout of terrain, a graph or chart of data with multiple variables or even just a visualization of a three-dimensional figure used for children in classrooms. This project has many uses for multiple day-to-day problems or decisions.

Our 3D data visualization grid proves that in the new age of technology, there still remains value in physical data representation. By making data more tangible and visible, we are making it more understandable and, in turn, more useful.

Appendix A - References

[1] "The Difference Between AC and DC Motors | eMotors Direct," *www.emotorsdirect.ca*, Sep. 08, 2023. https://www.emotorsdirect.ca/knowledge-center/article/the-difference-between-ac-and-dc-motors

[2] Electrical4U, "DC Shunt Motor: Speed Control & Characteristics | Electrical4U," *Electrical4U*, Feb. 24, 2012. https://www.electrical4u.com/shunt-wound-dc-motor-dc-shunt-motor/

[3] "Shenzhen Fedy Technology Co., LTD FD-5WSRGB-A." Available: https://cdn-shop.adafruit.com/datasheets/FD-5WSRGB-A.PDF

[4] "QT-Brightek PLCC Series 5050 PLCC-8 RGBW LED Part No.: QBLP679-RGBXW RGB=Color Code X: White Color Code X=W (Warm White) X=N (Natural White) X=C (Cool White)," 2025. Accessed: Oct. 24, 2025. [Online]. Available: https://www.qt-brightek.com/datasheet/QBLP679-RGBXW.pdf

[5] "WS2811 Signal line 256 Gray level 3 channal Constant current LED drive IC." Available: https://cdn-shop.adafruit.com/datasheets/WS2811.pdf

[6] Right Light Inc, “Difference between WS2811, WS2818, WS2812b, WS2813 - Right Light Inc.,” *Right Light Inc.*, Sep. 12, 2023. https://holiday-lights.ca/w-knowledge-base/difference-between-ws2811-ws2818-ws2812b-ws2813-and-ws2815-led (accessed Oct. 24, 2025).

[7] “WS2812B Intelligent control LED integrated light source Features and Benefits.” Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

[8] “SNx4HC138 3-Line To 8-Line Decoders/Demultiplexers.” Accessed: Mar. 10, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74hc138.pdf

[9] Texas Instruments Inc., *CD4514B/ CD4515B 4-to-16 Line Decoders/Drivers Data Sheet*, Rev. A, Dallas, TX, June 2003. [Online]. Available: https://www.ti.com/lit/ds/symlink/cd4514b.pdf

[10] Electronics Tutorials, “Linear Solenoid Actuator Theory and Tutorial,” *Basic Electronics Tutorials*, Feb. 15, 2018. https://www.electronics-tutorials.ws/io/io_6.html

[11] NFP Motor, "37mm Metal Gear Motor Model NFP-GM37-3429-EN Datasheet," NFP Motor. [Online]. Available: https://nfpshop.com/product/37mm-metal-gear-motor-model-nfp-gm37-3429-en

[12] Maxon Motor, "EC 90 flat ⌀90 mm, brushless, 90 Watt Motor Datasheet," Maxon Group. [Online]. Available: https://www.maxongroup.com/medias/sys_master/8825435389982.pdf

[13] Pololu Corporation, "37D Metal Gearmotors Datasheet," Pololu, Jan. 2020. [Online]. Available: https://www.pololu.com/file/0J1736/pololu-37d-metal-gearmotors-rev-1-2.pdf

[14] STMicroelectronics, "L298 Dual Full-Bridge Driver Datasheet," STMicroelectronics, Jan. 2000. [Online]. Available: https://cdn.sparkfun.com/assets/7/1/d/6/c/Full-Bridge_Motor_Driver_Dual_-_L298N.pdf

[15] Toshiba Semiconductor, "TB6612FNG Driver IC for Dual DC Motor Datasheet," Toshiba Electronic Devices & Storage Corporation. [Online]. Available: https://www.pololu.com/file/0J86/TB6612FNG.pdf

[16] Texas Instruments, "DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense (PWM Control) Datasheet," Texas Instruments, Jul. 2016. [Online]. Available: https://www.ti.com/lit/gpn/DRV8871

[17] Adafruit Industries, "Small Push-Pull Solenoid - 12VDC Product ID: 412 Datasheet," Adafruit Industries. [Online]. Available: https://media.digikey.com/pdf/data%20sheets/adafruit%20pdfs/412_web.pdf

[18]A. Yard, “L298N Motor Driver With Arduino - A Complete Guide - ArduinoYard,” *ArduinoYard*, Feb. 17, 2025. https://arduinoyard.com/l298n-motor-driver-with-arduino/ (accessed Nov. 25, 2025).

[19]T. S. Team, “IPC-2221 Standards in PCB Design,” *Sierra Circuits*, Jul. 20, 2021. https://www.protoexpress.com/blog/ipc-2221-circuit-board-design/

[20] “Daisy Chain [],” *Supremainc.com*, 2015. https://kb.supremainc.com/knowledge/doku.php?id=en:daisy_chain (accessed Nov. 25, 2025).

[21] “Projects,” *Thedracolab.com*, 2015. https://thedracolab.com/projects/ (accessed Nov. 25, 2025).

[22] “Compare Autodesk Fusion vs Autodesk Fusion for Personal Use | Autodesk,” *Autodesk.com*, Jan. 19, 2021. https://www.autodesk.com/products/fusion-360/personal?cjdata=MXxOfDB8WXww&AID=10282382&PID=100357191&SID=oc5A05L8E_cFrofF6SHThESpPJQwtFNTR3gRlX-mLGRJWCCSEKCMAkRqPDeULOuy&mktvar002=afc_us_deeplink&cjevent=667b779a8a6911f08338006c0a82b824&affname=100357191_10282382. (accessed Nov. 25, 2025).

[23] “6 Layer PCBs,” Jlcpcb.com, 2025. https://jlcpcb.com/resources/6-layer-pcbs?from=VBS_Free6layerPCBs&utm_source=bing&utm_medium=cpc&utm_campaign=422890460&utm_content&utm_term=e_pcb%20jlcpcb&adgroupid=1345803553234414&msclkid=d82c96449f7719d27048fefe6121d4fc. (accessed Nov. 25, 2025).

Appendix B - Copyright Requests

Figure 4.2 Request

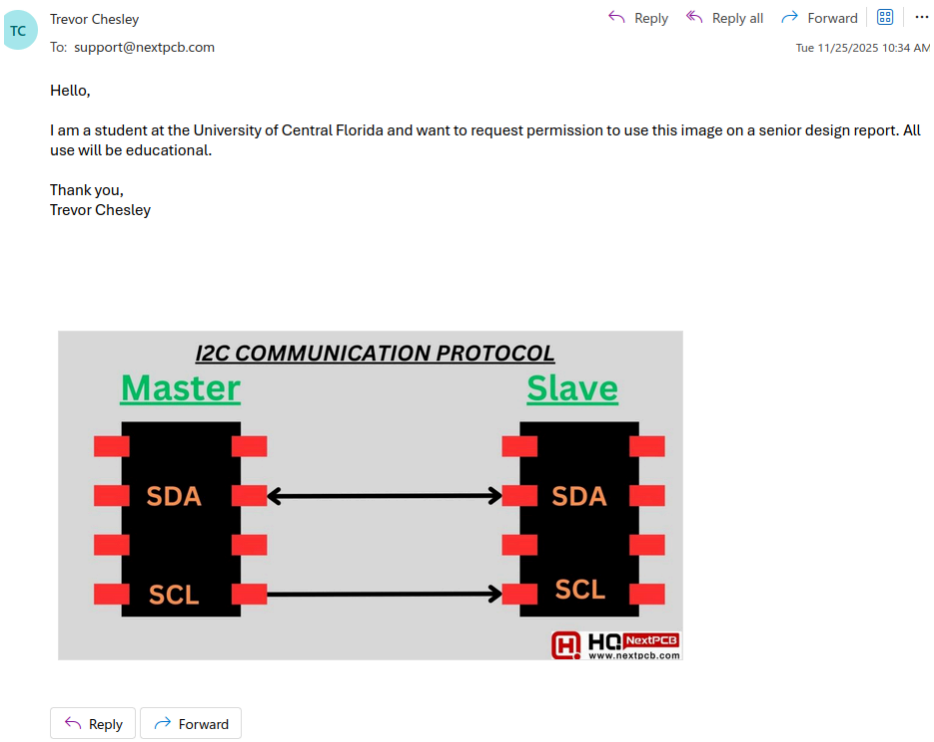


Figure 3.3 Request

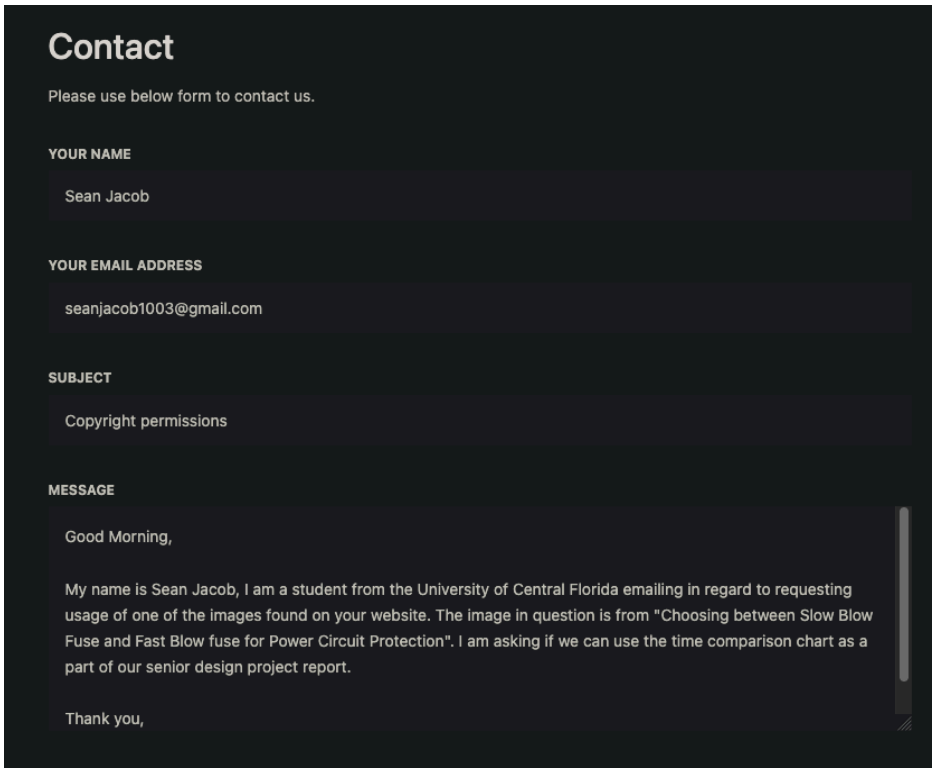
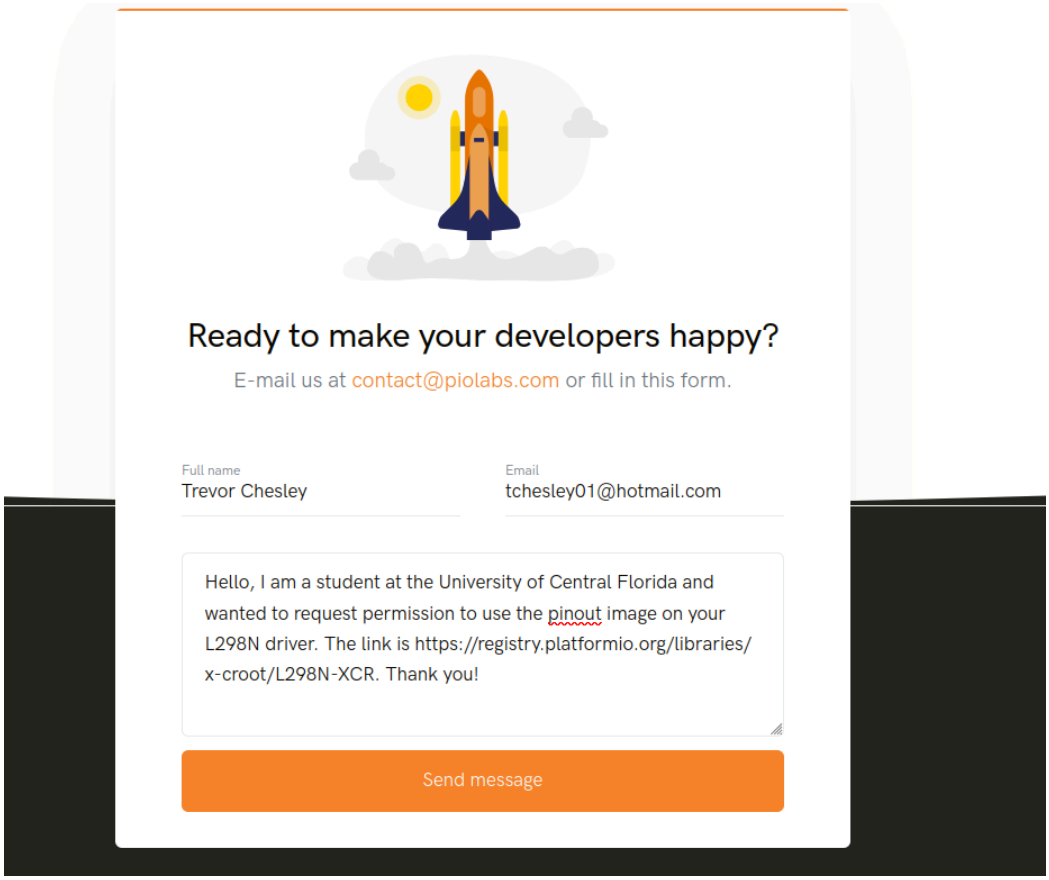


Figure 4.5 Request



Appendix D - Software Code

```cpp
#include <Arduino.h>
#include <FastLED.h>

// Solenoid pins
#define SOLENOID1_GPIO 23
#define SOLENOID2_GPIO 22

// Motor driver pins
#define MOTOR_IN1 25
#define MOTOR_IN2 26
#define MOTOR_ENA 27

// LED strip pins
#define LED_STRIP_GPIO 13
#define NUM_LEDS 2

CRGB leds[NUM_LEDS];

class Solenoid {
private:
    int pin;
    int id;

public:
    Solenoid(int gpio_pin, int solenoid_id) : pin(gpio_pin), id(solenoid_id) {
        pinMode(pin, OUTPUT);
        digitalWrite(pin, LOW);
    }

    void activate() {
        digitalWrite(pin, HIGH);
        Serial.printf("✓ Solenoid %d ON\n", id);
    }

    void deactivate() {
        digitalWrite(pin, LOW);
        Serial.printf("✓ Solenoid %d OFF\n", id);
    }

    void pulse(int duration_ms) {
        Serial.printf("Pulsing Solenoid %d for %dms...\n", id, duration_ms);
        activate();
        delay(duration_ms);
        deactivate();
    }
};
```

```
    void pulse(int duration_ms) {
        Serial.printf("Pulsing Solenoid %d for %dms...\n", id, duration_ms);
        activate();
        delay(duration_ms);
        deactivate();
    }
};

class Motor {
private:
    int in1, in2, ena;

public:
    Motor(int pin_in1, int pin_in2, int pin_ena)
        : in1(pin_in1), in2(pin_in2), ena(pin_ena) {

        pinMode(in1, OUTPUT);
        pinMode(in2, OUTPUT);
        pinMode(ena, OUTPUT);

        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);

        ledcSetup(0, 20000, 8);
        ledcAttachPin(ena, 0);
        ledcWrite(0, 0);
    }

    void forward(uint8_t speed) {
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        ledcWrite(0, speed);
        Serial.printf("Motor forward at speed %d\n", speed);
    }

    void backward(uint8_t speed) {
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        ledcWrite(0, speed);
        Serial.printf("Motor backward at speed %d\n", speed);
    }

    void stop() {
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
```

```
    void stop() {
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        ledcWrite(0, 0);
        Serial.println("Motor stopped");
    }
};

class LEDStrip {
public:
    LEDStrip() {}

    void begin() {
        FastLED.addLeds<WS2812B, LED_STRIP_GPIO, GRB>(leds, NUM_LEDS);
        FastLED.setBrightness(128);
    }

    void setBothLEDs(uint8_t r, uint8_t g, uint8_t b) {
        for (int i = 0; i < NUM_LEDS; i++) {
            leds[i] = CRGB(r, g, b);
        }
        FastLED.show();
    }

    void setIndividualLEDs(uint8_t r1, uint8_t g1, uint8_t b1, uint8_t r2, uint8_t g2, uint8_t
        leds[0] = CRGB(r1, g1, b1);
        leds[1] = CRGB(r2, g2, b2);
        FastLED.show();
    }

    void clear() {
        FastLED.clear();
        FastLED.show();
    }
};

Solenoid solenoid1(SOLENOID1_GPIO, 1);
Solenoid solenoid2(SOLENOID2_GPIO, 2);
Motor motor(MOTOR_IN1, MOTOR_IN2, MOTOR_ENA);
LEDStrip ledStrip;

void printMenu() {
    Serial.println("ESP32 Hardware Control Menu");
    Serial.println("\n--- SOLENOIDS ---");
    Serial.println("1 - Solenoid 1 ON");
```

```
void printMenu() {
    Serial.println("ESP32 Hardware Control Menu");
    Serial.println("\n--- SOLENOIDS ---");
    Serial.println("1 - Solenoid 1 ON");
    Serial.println("2 - Solenoid 1 OFF");
    Serial.println("3 - Solenoid 1 PULSE (1 sec)");
    Serial.println("4 - Solenoid 2 ON");
    Serial.println("5 - Solenoid 2 OFF");
    Serial.println("6 - Solenoid 2 PULSE (1 sec)");

    Serial.println("\n--- MOTOR ---");
    Serial.println("f - Motor Forward (speed 100)");
    Serial.println("b - Motor Backward (speed 100)");
    Serial.println("s - Motor STOP");

    Serial.println("\n--- LEDs ---");
    Serial.println("r - LEDs Red");
    Serial.println("g - LEDs Green");
    Serial.println("u - LEDs Blue");
    Serial.println("y - LEDs Yellow");
    Serial.println("p - LEDs Purple");
    Serial.println("c - LEDs Cyan");
    Serial.println("w - LEDs White");
    Serial.println("o - LEDs OFF");
    Serial.println("a - LEDs Alternate (Red/Blue)");

    Serial.println("\n--- SPECIAL ---");
    Serial.println("d - Run Full Demo");
    Serial.println("x - ALL OFF (Emergency Stop)");
    Serial.println("m - Show Menu");

    Serial.println("Enter command: ");
}

void runDemo() {
    Serial.println("\nFull Demo");

    // Solenoid 1 + Motor Forward + Red LEDs
    Serial.println("Part 1: Solenoid 1 + Motor Forward + Red LEDs");
    ledStrip.setBothLEDs(255, 0, 0);
    solenoid1.activate();
    motor.forward(100);
    delay(1500);
    solenoid1.deactivate();
    motor.stop();
```

```
void runDemo() {
    Serial.println("\nFull Demo");

    // Solenoid 1 + Motor Forward + Red LEDs
    Serial.println("Part 1: Solenoid 1 + Motor Forward + Red LEDs");
    ledStrip.setBothLEDs(255, 0, 0);
    solenoid1.activate();
    motor.forward(100);
    delay(1500);
    solenoid1.deactivate();
    motor.stop();
    ledStrip.clear();
    delay(500);

    // Solenoid 2 + Motor Backward + Blue LEDs
    Serial.println("Part 2: Solenoid 2 + Motor Backward + Blue LEDs");
    ledStrip.setBothLEDs(0, 0, 255);
    solenoid2.activate();
    motor.backward(100);
    delay(1500);
    solenoid2.deactivate();
    motor.stop();
    ledStrip.clear();

    Serial.println("Demo Complete\n");
}

void allOff() {
    Serial.println("\nEmergency Stop");
    solenoid1.deactivate();
    solenoid2.deactivate();
    motor.stop();
    ledStrip.clear();
}

void processCommand(char cmd) {
    switch(cmd) {
        // Solenoid 1
        case '1':
            solenoid1.activate();
            break;
        case '2':
            solenoid1.deactivate();
            break;
        case '3':
```

```
        // Solenoid 2
        case '4':
            solenoid2.activate();
            break;
        case '5':
            solenoid2.deactivate();
            break;
        case '6':
            solenoid2.pulse(1000);
            break;

        // Motor
        case 'f':
        case 'F':
            motor.forward(100);
            break;
        case 'b':
        case 'B':
            motor.backward(100);
            break;
        case 's':
        case 'S':
            motor.stop();
            break;

        // LEDs
        case 'r':
        case 'R':
            Serial.println("✓ LEDs Red");
            ledStrip.setBothLEDs(255, 0, 0);
            break;
        case 'g':
        case 'G':
            Serial.println("✓ LEDs Green");
            ledStrip.setBothLEDs(0, 255, 0);
            break;
        case 'u':
        case 'U':
            Serial.println("✓ LEDs Blue");
            ledStrip.setBothLEDs(0, 0, 255);
            break;
        case 'y':
        case 'Y':
            Serial.println("✓ LEDs Yellow");
            ledStrip.setBothLEDs(255, 255, 0);
```

```
        case 'Y':
            Serial.println("✓ LEDs Yellow");
            ledStrip.setBothLEDs(255, 255, 0);
            break;
        case 'p':
        case 'P':
            Serial.println("✓ LEDs Purple");
            ledStrip.setBothLEDs(255, 0, 255);
            break;
        case 'c':
        case 'C':
            Serial.println("✓ LEDs Cyan");
            ledStrip.setBothLEDs(0, 255, 255);
            break;
        case 'w':
        case 'W':
            Serial.println("✓ LEDs White");
            ledStrip.setBothLEDs(255, 255, 255);
            break;
        case 'o':
        case 'O':
            Serial.println("✓ LEDs OFF");
            ledStrip.clear();
            break;
        case 'a':
        case 'A':
            Serial.println("✓ LEDs Alternating");
            ledStrip.setIndividualLEDs(255, 0, 0, 0, 0, 255);
            break;

        case 'd':
        case 'D':
            runDemo();
            break;
        case 'x':
        case 'X':
            allOff();
            break;
        case 'm':
        case 'M':
            printMenu();
            break;

        default:
            Serial.println("Invalid command");
            break;
```

```
            break;
        case 'm':
        case 'M':
            printMenu();
            break;

        default:
            Serial.println("Invalid command");
            break;
    }
}

void setup() {
    Serial.begin(115200);
    delay(1000);

    Serial.println("\n\n");
    Serial.println("ESP TEST: ");
    Serial.println();

    ledStrip.begin();

    ledStrip.setBothLEDs(0, 255, 0);
    delay(500);
    ledStrip.clear();

    printMenu();
}

void loop() {
    if (Serial.available() > 0) {
        char cmd = Serial.read();

        // Ignore newline and carriage return
        if (cmd == '\n' || cmd == '\r') {
            return;
        }

        processCommand(cmd);
    }
}
```