3-D Data Visualization Grid

Dallas Moody, Sean Jacob, Trevor Chesley, Tomas Yepez

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This paper presents the design and building process of a physical, three-dimensional data display system. Using an 8×8 grid of LED lights mounted on motorized towers, the system brings data to life by moving each light across four distinct height levels. Rather than viewing data on a flat screen, users can observe it as a physical, glowing landscape that changes shape in real time. The project combines basic lighting and motor control to represent complex datasets in a way that is easy to understand at a glance. Testing shows that this approach makes interpreting data faster and more intuitive than traditional display methods.**

***Index Terms* — Arrays, Data Visualization, DC Motors, Gears, Microcontrollers, Solenoids, Switching Circuits,**

I. Introduction

In our world, data is everywhere. We view data in many different ways like drawing, website applications, etc. What we haven't seen are ways to visualize data in three dimensions that's not on a screen and more physical. There are physical blocks that are used but they can't change dynamically with new datasets. This is where our 3D Data Visualization Grid comes in.

This project will have uses in places like education, topography, and data analysis. We want this project to make it easier to view data in the three dimensions without having to click and drag around a screen to see. Some examples would be to see a topography chart of terrain as well as basic bar charts.

Our visualization grid takes 64 LED's in an 8x8 grid and attaches them to towers that will be moved vertically via gears and rack & pinions. Each LED will have its own tower and can move independently. We will be able to control the height and color of each specific LED as they are addressable. These LED's will be connected in a daisy chain configuration so we can address all of them on only one pin from the microcontroller. They also stay in parallel to reduce our power consumption. Our height control will have 4 distinct levels. Our goal is for our LED's to light up each tower through the inside.

We also have 8 DC motors (1 per row) that control these towers. We will have 8 gears on each motor rod to control each of the 8 towers for the row. There is also a bearing in each gear so the gear will only move with the motor rod if the solenoid for said gear is active.

II. System Components

The system can be divided into two main components, which can then be divided into smaller subcomponents. The two main components are the electrical systems and the mechanical systems.

The electrical systems include components such as the printed circuit board (PCB) and its peripherals. Peripherals are the components that communicate with the "brain" of the PCB and handle the specialized tasks. For example, the mico-controller unit (MCU) is the brain of the PCB and it tells the motor drivers (peripherals) when to turn the motors on, and what direction they must move in. Our other peripherals include the light emitting diodes (LEDs) and the solenoids. The LEDs light up inside the towers to distinguish a certain area of the grid from another. The MCU tells the LEDs when to turn on and what color they should be. For the solenoids, the MCU tells them when to activate which is how we decide the level for each tower. If one solenoid activates earlier with respect to the solenoid next to it after the motor has begun rotating, then the tower of the solenoid that activated first will be higher than the first one because it has had more rotating time.

A.) Microcontroller

The MCU chosen for this project is the ESP32-WROOM-S3. Some of the characteristics that solidified this decision were its 240MHz frequency, 45 GPIO pins, and Wi-Fi & bluetooth capabilities. This creates the possibility to program the grid to display data via bluetooth on a laptop from a web app. The ESP32 provides an unmatched balance of performance and price. Its advanced features include ultra low power modes, a vast amount of libraries, and more.

B.) Motors

For the motors that will turn the gears, the Greartisan 12V, 15 RPM stepper motors were chosen. Stepper motors take DC inputs and divide full rotation into precise and discrete steps, similar to how a clock works. Since it is a stepper motor, it allows for greater precision when deciding what position to end in. The 64:1 gear reduction means the motor provides 2,048 steps to stop for one full revolution. This becomes very important when it comes to Moreover, these motors are run via gears internally.

Although gears mean the motor will likely turn slower than a gearless motor, they allow for a greater torque output. Thankfully, this means that there are no issues when one of the motors has to lift all eight towers in its row (which is very likely to happen even if they are not all being lifted to the highest level).

C.) LEDs

The LED components for the project are the WS2812B RGB LEDs. Since they are not the typical two prong diode style LEDs, we are able to address them individually even if they are arranged in series like a string. These LEDs are able to display over 16 million different colors. They contain a metal plate on the back of the 10mm mini PCB board as a heatsink to pull heat back to prevent the LED from burning out.

D.) Solenoids

For the solenoids, we chose 12V non-latching solenoids. What this means is that they need current to maintain their position. Some solenoids actuate and only require current in the moment it is changing states, as soon as it has changed state, it can hold that position and not require any more power. Ours need current to hold its state which means they can't be actuated for very long because heat will become an issue. Thanks to the pawl mechanism, we do not have to worry about overheating because we can de-actuate the solenoid as soon as the tower reaches the level it needs to be at.

E.) Optocouplers

In order to maintain isolation between sensitive components like an MCU and noisy components like motors or solenoids, optocouplers were used. The LTV-847 4 channel optocoupler was used due to it allowing for more inputs than the 1 or 2 channel models.

F.) Drivers

When dealing with certain components, like motors for example, a driver is necessary. In the case of the 3D data grid, the need for DC motor drivers was apparent. Upon careful consideration, the L298 DC motor driver was chosen to help with control of the system. The L298 motor driver is a dual h-bridge driver with the ability to control 2 motors at once.

G.) Mechanical

The mechanical systems include the gear & motor rod, rack & pinion, and the mechanism we have elected to maintain the towers up.

As mentioned before, 8 gears are attached via a bearing to a motor rod. The long rod is attached to the motor and rotates as the motor rotates. Each gear is 3D printed and is designed with multiple holes on the sides of it. This is where the solenoid activates to select the towers that are meant to go up and the ones that aren't. The gears are not supposed to rotate at the same time as the motor rod rotates because then every single tower would go up at the same time.

The rack and pinion are the mechanical portion that bring the towers up and down. The racks are placed into a U-bracket that stabilizes them in between the bracket and the teeth from the gear as they go up and down. The top part of the rack is attached to the tower that will be visible to represent the different levels of data. Along the height of the rack is where the wiring for the LED will pass and at the base of the tower is where the LED will actually be placed.

Lastly for the mechanical portion of the components is the mechanism we have to keep the gears up. As we mentioned before, the gears are not supposed to rotate with the motor rods as that would cause every tower to go up together and would eliminate the distinct levels that create the representation of the data we want to show. Instead, the inner part of the bearing in the gear rotates while the outer part doesn't when we don't want the gear to rotate. The issue this creates is that when the rack is raised, the tangential weight of the rack moves the gear back in the opposite direction and lowers the tower. This could be solved by keeping the solenoid engaged at all times but the solenoids require constant current while they are engaged to maintain that state. The constant current means that they heat up & if all 64 solenoids are activated, then the current demand can cause massive heat issues not only with each solenoid, but with the entire PCB itself.

III. System Concept

The concept of our system comes in steps. First we upload an array onto our microcontroller with the color code and the height level. Then the microcontroller reads the array and sends the data to the motor drivers, the LED's and the switching circuit. These three systems then work together to get each node to the correct height and color that's stated in the array. To go more in-depth, we

can split this into three stages: receive, process, and display.

In the *receive* stage of this system, the data is sent to the microcontroller via an external source, a computer. This will most likely just be the array of LEDs with the color code and the height level. The data is received by the ESP and the system then goes to the *process* stage.

The *process* stage is when the ESP interprets the data it has just received, the array. The ESP will send all the color codes in a single file to the daisy chain of LED's who will read their color code then send the rest to the next LED and on. The ESP will also send the signal to turn on motors through the motor driver as well as to turn on the solenoids through the selection circuit for a specific amount of time depending on the height level.

We have now reached the *display* stage. This is where each process works. The LED's turn on to their designated color code. The motors turn on if they need to move towers up. The solenoids turn on for the amount of time relative to the height level. For example, if the height level is 2, the solenoid will turn on for 2 seconds and then turn off because it overheats fast.

A.) Hardware Concept

The hardware consists of multiple different systems working together. First, we have the brains of the operation, the ESP. This manages all the data sent in and tells each system what to do. Then we have the motor driver and motors. These take the signal from the ESP and run it through the motor driver to know which motor to turn on and for how long. Then we have the selection circuit that controls the solenoids. This also takes the data from the ESP to know how long to turn on each solenoid for and when one turns off we can turn another one on since we can only have 4 active solenoids at once. Finally we have the LED's. These aren't necessarily a circuit but just 64 LED's soldered together in a daisy chain configuration. We chose to do this because it's easier to use only one GPIO pin to send all of the color codes at once and the LED's know to only take the maximum amount of bits for their color code and then send the rest of the bits on.

IV. Hardware

This section will focus on going into more detail about each of the system components described in the second section of the paper, titled System Components. Specifics regarding each of the components mentioned will be broken down to provide the reader with a deeper understanding of how the system works and how the components interact with each other.

A.) MCP23017 Pin Expander

When it comes to hardware, one of the most essential parts is the pin expanders. With 64 solenoids that need to be individually addressed, 64 LEDs that also need to be individually addressed, and eight motors, the ESP32 has nowhere near the amount of pins necessary to drive these components. The pin expanders allow us to increase the capabilities of the ESP32 and run the grid without having to purchase a second one. With all of the pins necessary for the components, it's unlikely even two ESP32s would get the job done. The pin expanders used in this project are the MCP23017 expanders. They add 16 pins which are split into two 8-bit ports. One good thing about them is that they only use two pins on the ESP32, a serial data line (SDA) and a serial clock line (SCL). They communicate with the ESP32 via I^2C. The expanders also feature two interrupt pins. The interrupt pins can signal to the MCU when a pin's state changes. The important part about this feature is that it prevents the need for constant polling. Polling is very energy demanding so removing this allows the code to be more efficient.

B.) DC Motor for Vertical Height Adjustment

In order for the data visualization grid to work in the z-axis, we needed a way to successfully control the height of desired nodes. To achieve this vertical height control, a motorized system was put in place. The specific motor used in the case of the 3D data visualization grid was the Greartisan 12V DC Motor. The motor features a gearbox with a 37m diameter built into a sturdy metal frame perfect for high torque applications. The Greartisan comes in many models, all with varying torque ratings. The specific model chosen for the visualization grid had 15 RPM and allowed for significant torque ratings.

The motorized system works in conjunction with mechanical components like a rack and pinion and pawl to allow for vertical movement of a node. The motor on the rod is able to rotate the gear allowing for movement of the rack.

C.) Addressable LEDs

Our LED's are soldered to each other in a daisy chain format. Since each LED is addressable and can change color, we needed a way to send the color to each LED without having to use 64 pins. The daisy chain solves this problem with a super simple solution. With the configuration and the LED's being smart enough, we can send all of the color codes in one pin to the LED's and the

first LED will know to only take the amount of bits needed for its color code and then send the rest of the bits to the next LED. This happens with each LED which will end with the last LED getting the last color code.

With the WS2812B LEDs, RGB colors are available, allowing for a wide range of colors. Using the addressability of each LED along with the hex code for a specific color, any node can be simply adjusted and set based on the desired dataset. While a perfect white color is not possible using this method, something close enough can be achieved, simulating a color close enough to white for the eye to recognize it as such.

D.) Solenoid with Selection Circuit

Working alongside our DC motor and Rack and pinion, a solenoid is required to activate the specific node, allowing it to be adjusted. The main idea is that the gear will have openings around aligned with the height of the solenoid. When a node requires change in height, the solenoid will actuate, filling in the opening and allowing the gear to rotate, and thus moving the rack. Rotation of the rod along with solenoid actuation will allow the pin to catch onto the gear opening and successfully move the node.

One of the large concerns with doing it this way is the need for a way to select which solenoid needs to be actuated based on which node requires height change. Simply powering or unpowering a solenoid is not achievable when dealing with 64 separate nodes that need to be changed. To perform this, a selection circuit was needed. The MOSFET based selection circuit that was chosen can be seen below.

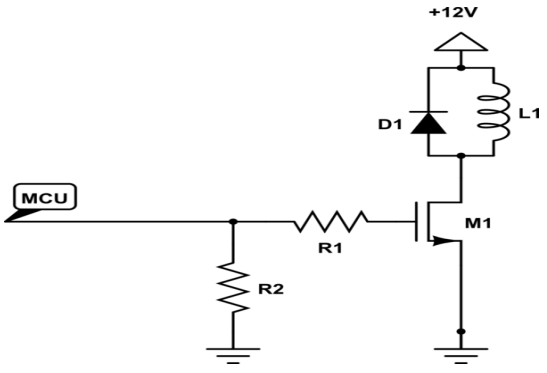


Fig 1. Mosfet selection circuit for solenoid actuation

In the figure above, the inductor(L1) is the solenoid used for vertical movement. It is labeled this way due to the solenoid being fundamentally an inductor in functionality. The selection circuit works by receiving a signal from the MCU which is fed into the mosfet. Due to the large amount of solenoids needed for the visualization grid, pin expanders were used as inputs instead of MCUs, but it works in the same way. Based on the state of the MCU pin, the solenoid will either engage or disengage with the gear openings. Another component of the circuit is the flyback diode in parallel with the solenoid. This flyback diode is fully protective and acts as a guard from large current spikes that are generated as a result of the solenoid being powered off.

E.) Optical Isolation

Another very useful hardware component is the optocouplers. Optocouplers are a piece of technology that helps to facilitate the flow of data signals and remove noise. Data that is flowing through a trace flows into one end of the optocoupler and gets converted into light via an LED. The other end of the optocoupler contains a phototransistor that conducts current when exposed to the LED's light. This way, there is a galvanic isolation between two circuits, while also allowing for the safe transfer of information. The optocouplers are used to bridge the gap between the two ground pours. They provide an extra layer of protection to the signals being sent and received.

F.) DC Motor Driver

The use of DC motors in the system requires the need for a method of controlling them. A motor driver must be used to control and direct the motor in the desired direction on command. The L298 motor driver was chosen as the motor driver for the project.

The L298 motor driver works by providing it with a 12V supply for the motor, 5V logic supply, 4 input pins, and a ground. The 12V supply is used to power each of the DC motors depending on what the software asks of it. Each of the 4 input pins corresponds to a certain motor. Since it is capable of controlling two motors, input 1and 2 control the first motor, and inputs 3 and 4 control the second. Each of these inputs will be tied to a GPIO pin on the MCU, and changing the state of one will cause rotation. In the case of input 1 being set high and input 2 being set low, the motor will spin forward. Reversing the polarity will result in a reversal of rotation by the motor. The same works for the second pair of inputs.

Using 4 of these dual h-bridge drivers, 8 motors can be fully controlled to spin whichever direction is desired. These 8 motors will serve as the 8 rows in the 8x8 grid of the project.

V. Power Distribution

The 3D grid is going to be powered via two AC/DC wall adapters. The power for each component trickles down from these two main sources of power. The reason for choosing the wall adapter is because there are no movement requirements for the grid. Since the grid itself is not moving, it is much more convenient to plug it into the wall and not worry about batteries losing power and having to recharge. Moreover, for the power that is necessary, it is much more economically convenient to go with the wall adapters. The two wall adapters are separated for one sole reason: signal integrity & noise reduction. Since the entire premise of the project is data representation, it would be very imprudent to neglect the integrity of the signals that carry the data. The first wall adapter is solely feeding our MCU (ESP32), while the second wall adapter is feeding everything else such as the mechanical components.

The wall adapters are the initial power source for the grid. Everything from there trickles down to the respective components. Voltage regulators are introduced in between the main power and the actual component to ensure all components see the proper voltage, not too high or too low. The voltage regulators are going to be mini PCBs that contain the regulator's IC per their respective datasheets' typical application circuit. Each mini regulator PCB will have male header pins on them to plug-and-play into the main PCB via female headers.

The first wall adapter is a set wall adapter that outputs 12 volts (V) with 2 ampere (A) capabilities. The ESP32 that it feeds only takes about 500mA at peak use so 2A is more than enough headroom to handle this.

After the wall adapter comes the first regulator. It is the Texas Instruments (TI) LM2576 5V fixed regulator. This surface mount integrated circuit (IC) can handle up to 65V on input and can output up to 37V. This regulator is a switching regulator, meaning there is a MOSFET inside the IC that switches on and off thousands of times per second to maintain the voltage desired. In this case, it will take 12V from the wall and output 5V.

The next downstream component that feeds the ESP32 directly is the TI UA78M33 regulator. This is a linear regulator meaning it doesn't switch on/off inside and simply takes the difference in voltage (from input to output) and dissipates it as heat.

The decision behind the step-down design here is two reasons. First off, the linear regulator would dissipate a lot of heat if it took a 12V input from the wall adapter and fed the ESP32 directly. Since the ESP32 takes 3.3V to function, the difference from input to output (12V - 3.3V = 8.7V) would be dissipated as heat. Too much heat can cause issues with the IC itself and with the rest of the PCB. This is why the LM2576 regulator takes the 12V input from the wall. Although the design could simply be the 12V input directly to the LM2576 and then to the ESP32, the LM2576 is a *switching* regulator. The internal switching of that IC introduces sinusoidal waves/ripples on the output. These waves/ripples create noise in the copper traces of the PCB and can negatively affect the ESP32 and the data we transmit. To mitigate this, the UA78MC33 is placed in between the LM2576 and the ESP32. Since the UA78M33 has no switching inside and is a linear regulator, it will remove a majority of the noise coming from the LM2576. Moreover, this regulator is designed with one input and one output capacitor that further smoothens the voltage ripples.

The second wall adapter feeds all the other components of the project. This wall adapter variable AC/DC, meaning a knob can be turned to increase/decrease output voltage. Being that it is a 144W adapter, it can output up to 9A when fixed to 16V. This wall adapter feeds the solenoids, motors, motor drivers and LEDs. The setup for this power distribution is very similar to the first wall adapter in that it is trickled down.

First, the solenoids take 12V and about 460mA each. The LM2576 was chosen for this. The solenoids are arranged in parallel so they can all share the same voltage. Since the project requires 64 solenoids, it is preferable to arrange them this way because if they were in series, 768V would be needed to power them all. The LM2576 12V fixed output surface mount device (SMD) was chosen for this regulator IC.

Secondly, the motors also take 12V. Their current demand is less at around 45mA. All 8 motors are sharing the same 12V line from another LM2576 regulator IC. That regulator can handle up to 3A at a time and given that there are only 8 motors, their demand is only .368A if all are running at the exact same time.

The motor drivers we chose to control the motors are the L298N drivers. These run at 5V for their logic portion (the portion that communicates with the ESP32), so we have another LM2576 5V fixed regulator IC to power these. Each driver takes about 36mA so with only four of these (each driver can control 2 motors) the current demand is .144A which is well within the limits of the regulator.

Lastly, are the LEDs which are also arranged in parallel. They take 5V and only 60mA of current depending on the color being output (60mA is the maximum current

demand). They have their own LM2576 5V fixed regulator as well.

VI. Mechanical Components

A.) Rack and Pinion

Our rack is 3D printed like most of our mechanical components. We have it about 235 mm long and teeth all along it to connect with the gears. The gear pushes the rack up through the teeth, which, in turn, pushes our tower that will be connected at the top of the rack. We will have 64 racks so we have one per LED/tower that can be changed individually for different heights of the tower.

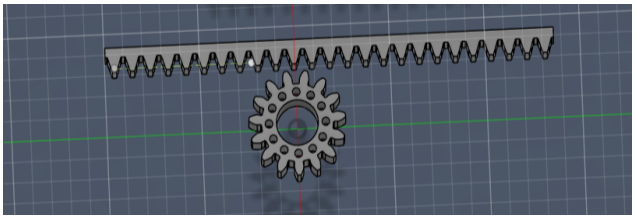


Fig 2. Screenshot of the 3D schematic of our rack and pinion

B.) Pawl

The pawl is our choice to hold the rack up once we get it to the designated height. Since we can only keep the solenoid on for a short amount of time, we had to figure out a different way to keep the racks from falling back down when we disengage the solenoids. The pawl uses a little lip and a spring behind it so when the rack moves up it pushes the pawl back and the spring locks it back after each tooth. Then, when we want to disengage the pawl, we have a mechanism to pull all the pawls back so the racks can lower back down to the starting position.

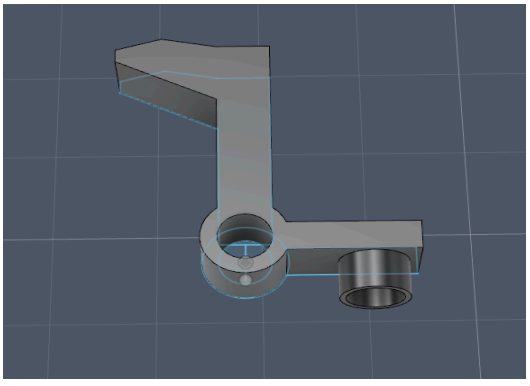


Fig 3. Screenshot of our pawl in our CAD software

C.) Solenoid Housing

We needed a way for the solenoid to always be able to engage in the holes of our gear, as well as a way to keep it spinning with the motor rod. To do this, we decided to make a housing for the solenoid. This housing is just a square to fit the solenoid in with a hole at the bottom to fit the motor rod so the solenoid can stay spinning with the rod and it will reach the gear holes.

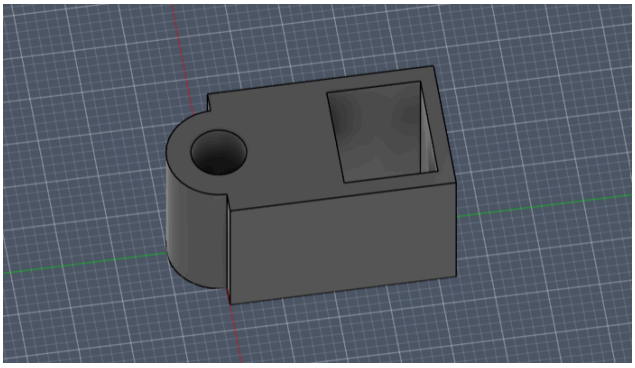


Fig 4. Screenshot of Solenoid housing in CAD software

D.) Slip Ring

With all of the wires coming from the solenoids and from our PCB we needed a way to keep them from getting twisted and tangled within our motor rod. To solve this, we found slip rings which allow the wires to spin with the motor rod so they don't get caught and cause problems. We will have the slip ring be on the opposite side of the motor rod from the motor.

E.) Motor Mechanics

For our motor mechanics, we have 4 different pieces that go with it. We have the motor rod that is just a thin metal rod that we connect to the motor via a 3D printed piece that fits both the motor and rod into it. We also have a 3D printed holder for the slip rings to fit into directly across from the motor. Finally, we have the motor housing. The motor housing is also 3D printed to wrap around the motor and it screws into the wood where we are placing the motors.

VII. Software

Software is needed to control all subsystems of the 3D grid, including motor control, solenoid switching, and the LEDs. Using this control the software system needs to be able to adjust the height of each of the towers using a combination of motor and solenoid control. Care must be taken to also make sure electrical components don't exceed their rated values, such as the solenoids, which can

only be active for a maximum of two minutes to prevent overheating and damage to the part. This leads to software controlled failsafes along with care in the algorithms used to adjust the 3D grid. Another consideration is the requirement to track the state of the system as there is no feedback from the motors or solenoids. This will be done using state arrays, with additional data structures used to store what adjustments are needed and the final state required to successfully display the dataset. With these requirements, the software system was decided to be separated into different subsystems; the control subsystem, the state management subsystem, the file input/output subsystem, the failsafe subsystem, and finally the LED subsystem. Further details about these subsystems will be provided in the following sections.

A.) Control Software

To accurately control the height of individual towers, the control software has a straightforward algorithm to adjust height. First, the control software gets data from the state management system to find out which solenoid needs to be activated for the new dataset. It then turns the solenoid on, activating a timer used by the failsafe software. Once the solenoid is active, the corresponding motor is powered using GPIO pins on the pin expander, either being powered clockwise/counterclockwise depending if the tower needs to be lowered or raised. Using an experimentally calculated timing variable, the motor will be active for the correct amount of time as needed to adjust the height. When the time is achieved, the motor disconnects, followed by the solenoid. Control software then repeats this process being in close communication with the state management system to adjust current values of the array and what future adjustments are needed.

B.) State Management Software

This subsystem uses three critical arrays, the current state array, adjustment array, and dataset array. The current state array stores all the current status of the system, and is actively updated using information received from the control software. Initially, the file IO system will read the values into the C code, and store it into the dataset array. This represents what the final state of the system should look like. Using this and the current status array, a function is called that takes the difference of the current state from the dataset final state. This tells the software how many discrete levels need to be adjusted for each of the 64 towers. Once the adjustment array is calculated, control software takes over to make the adjustments necessary.

C.) File Input Output System

A smaller subsystem, this software is used to read datasets from an external source. This data is input to the ESP32 as a text file, and the software iterates over the text using a special format to read in the data values for each tower. This data is stored in the dataset array used by the state management software, and once the text file is fully read into the system, the File IO concludes and control is passed to the state management system.

D.) Failsafe

Due to current limits in the PCB and solenoid components, failsafes are required to ensure damage is not done to the system. Each solenoid has a high current requirement of 500 mA, which is a major limiting factor for the speed with which the 3D grid can be adjusted. Due to trace size and regulator current limits, it was decided that four solenoids be active at once as the safety limit, with a max of 30 seconds each. As said in the previous control software section, once the solenoid is on a software timer is started with information of the solenoid address. Once 30 seconds have passed, the software timer interrupts and makes sure the solenoid has been turned off. This situation should not be encountered, however this failsafe is introduced in case of unforeseen bugs and errors.

E.) LED Software System

It was decided to use the FastLED software library that works with the ESP32 to control the LED array. Working with the file IO system, the color for each of the 64 LEDs is stored and calling the LED software system converts this data into an LED array that can be transmitted to the hardware. For each new dataset, the file IO will store the data and call the same function to shift datasets.

VIII. PCB Design

The PCB design can essentially be broken down into two separate sections. Two sections because there are two separate ground pours, one for the signal based components and one for the mechanical components. The reason that there are two separate pours is to further protect the integrity of the signals being sent and received. If the ground between the mechanical components and the data processing components was shared, then noise can leak in from the common ground point and affect the ESP32. The fixed 12V output AC/DC wall adapter feeds the signal side while the adjustable adapter feeds all of the other components on the power heavy side of the PCB.

The signal side of the PCB contains the ESP32 and the USB connector, which is where the code to run all of the components is uploaded.

IX. Conclusion

The 3D LED Data Projection Grid set out to find a better way to display data, and the results show that this goal was successfully achieved. By using a grid of lights mounted on towers, the system is able to turn numbers and information into a physical, three-dimensional shape that anyone can look at and understand right away.

Building this project was a great learning experience. The team worked through several challenges along the way, from getting the PCB to work correctly to making sure all the lights responded accurately to incoming data. The team overcame these challenges, resulting in a system that works reliably and performs its intended functions.

The most important takeaway from this project is that seeing data in three dimensions makes it much easier to understand than looking at it on a flat screen. Users can see patterns and differences in the data right away by looking at the physical shape of the grid instead of numbers or charts.

In the future, this project could be expanded in many ways, such as making the grid larger, adding more height levels, or making the system smaller and more portable. These improvements would make the system even more useful in real-world settings like classrooms, weather stations, or live event displays.

Overall, the 3D LED Data Projection Grid is a creative and practical project that shows how engineering can be used to make information more engaging and easier to understand for everyone.

References

[1] “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

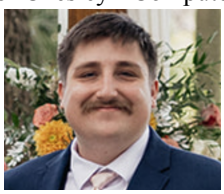
[2] “WS2811 Signal line 256 Gray level 3 channal Constant current LED drive IC.” Available: https://cdn-shop.adafruit.com/datasheets/WS2811.pdf

[3] 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

Meet the team

Sean Jacob - Computer Engineering

Trevor Chesley - Computer Engineering



Dallas Moody - Computer Engineering

Tomas Yepez - Electrical Engineering

Acknowledgement

The authors would like to thank Dr. Borowczak and Draco labs for support on the inspiration and progression of the work mentioned.