TA-DA DATA

OEDK Data Display: Final Report

OEDK

Oshman Engineering Design Kitchen

Ibrahim Al-Akash, Daniel Cho, Camilo Soto Onate, Leticia Souto, Vivian Zheng

ENGI 120 SPRING 2022

Executive Summary

The Oshman Engineering Design Kitchen (OEDK) opened in 2009 and has been collecting statistics on its users since 2015, however, this data is sitting dormant in a database inaccessible to the public. Given a budget of $500, the OEDK has tasked our team, Ta-Da Data, with creating a visually appealing and engaging physical data display to showcase these statistics and highlight the growth and achievements of the OEDK over the years. Following the Engineering Design Process (EDP), our team clarified the team assignment, conducted preliminary research to understand the problem and context, defined design criteria through client interviews and additional research, brainstormed and evaluated potential ideas, developed low, medium, and high-fidelity prototypes, and conducted rigorous testing to arrive at our final product. Our final design utilizes electromechanical components to showcase yearly visitor data, hourly rush hour traffic data, and male to female ratio. We designed our display with four main components: 1) clock rush hour traffic heatmap, 2) rocket yearly visitor bar graph, 3) LED matrix, and 4) 3D printed models representative of all the clubs at the OEDK. In conclusion, over the course of the 2022 spring semester, our team has successfully produced, tested, and validated a finished product that the OEDK will proudly display in the lobby for all visitors to see.

Table of Contents

I. BACKGROUND

The Oshman Engineering Design Kitchen (OEDK) opened in 2009 to assist undergraduate students at Rice university in designing solutions to real-world engineering problems. A variety of data such as daily access, 3-D printer usage, and percentage of female users collected by OEDK over the past 12 years tells the story of how much progress it and the Rice Engineering program at large has made in student engagement and diversification. Yet, this raw data is not easily accessible, resulting in a lack of awareness regarding the achievements, availability of resources, and demographics of the space. A solution to this existing problem was attempted by previous ENGI 120 students **(Figure 1)**. However, their design lacked precision and the data was displayed inaccurately as the mechanical components of the display were not correctly synchronized. This existing solution is not currently displayed.

Our goal is to design a dynamic, engaging and visually pleasing display that accurately models growing data available to increase awareness regarding the achievements and the inclusivity of the OEDK. The display will be placed in the lobby for all OEDK visitors and users. This display will foster a sense of pride and achievement for the OEDK, strengthen the relationship between the OEDK and its users, and increases transparency to prospective students, companies, government agencies, and potential sponsors visiting the OEDK.

Figure 1: *Previous solution developed by previous ENGI 120 team in the 2021 fall semester.*

II. DESIGN CRITERIA

To objectively quantify success in this project, we created a list of objectives and constraints for our final design solution using our initial interviews to support the design criteria categories and research to back up the target values **(Table 1).**

Table 1: *Design criteria for OEDK data display. * User-Defined Scales (UDS) specified in Appendix.*

	Design Criteria	Target Value
CONSTRAINTS	Safety	**No** sharp edges, loose components, falling pieces, or electrical hazards. No physical touch required to view the display.
	Accuracy	Display is within **±5%** margin of error of the dataset.
	Intuitive to Understand	Scores average **≥ 4** on UDS
OBJECTIVES	Visual Appeal	Scores average **≥ 4** on UDS
	Engaging	Scores average **≥ 3** on UDS
	Durable	System maintains functionality for **one semester**
	Updateability	Must update display at most once **per semester**

Safety and Accuracy

Throughout our interview, our client highlighted the importance of safety and accuracy of data representation in our design. We categorized these features as constraints in our design criteria because our solution would be considered unsuccessful otherwise. The target value for safety is

specific to our client's desires and our target value for accuracy is based on common margin of errors.

Visual Appeal and Engagement

Our client heavily emphasized the visual appeal and engagement aspect of the data display, prompting us to use our research as a guide when defining the target values. For example, our research on visual processing and the psychology of engagement revealed that humans desire an enjoyable and memorable design and are dissuaded when presented with an unattractive or unpleasant visual experience. This allowed us to determine the scale of the target values for the "Visual Appeal" objective.

Durability and Ease of Maintenance

Due to the nature of the data, the client expressed that the solution may incorporate electronic and mechanical hardware. This guided our research into topics such as Arduino Uno, servo and stepper motors, and the applications of these components, allowing us to create an expectation of durability and maintenance of our visual display.

Intuitive

This objective design criteria is supported by our research of data visualization strategies and case studies. According to both our client and guiding research, a good display must be easily understood. Additionally, because our client mentioned that there is a wide audience that comes through the ODEK including prospective students, companies, and users, it is imperative that our display can be intuitively understood for all demographics.

Prioritizing Design Criteria

After creating our design criteria, we generated a Pairwise Comparison Chart to help us prioritize our objectives in preparation for the next step of the EDP process. We based our reasoning on our initial research and conversations with our client, resulting in "Visual Appeal" and "Intuitive to Understand" as the top objectives for this project.

Table 2: *Pairwise comparison chart (PCC) for ranking importance of design criteria.*

	VA	D	EM	IU	E	Total
Visual Appeal (VA)	—	1	1	0	1	3
Durability (D)	0	—	1	0	0	1
Easily Maintainable (EM)	0	0	—	0	0	0
Intuitive (IU)	1	1	1	—	1	4
Engaging (E)	0	1	1	0	—	2

Defining our design criteria and their target values provides us a framework as we move forward into the next stages of the EDP. Incorporating all these objectives will be what we strive for and are vital in making a successful solution for our client.

III. CONCEPTUAL DESIGN

Overview

Having been provided some datasets by the OEDK, we will display three of the most impactful datasets showcasing the growth and achievements of the OEDK. One dataset we will be displaying is the number of visitors that come into the OEDK each year. We noticed that there has been steady growth in the number of visitors year-over-year and our client expressed that the OEDK is proud to show off these statistics. The next dataset we will be showing is the rush hour traffic of the OEDK showing the busiest times in which the OEDK is full of students. This data will provide utility to students coming into the OEDK to better plan their schedules and take into account the high-traffic times. Another dataset we are showcasing is the male to female ratio of users at the OEDK. In 2022, there was a 60:40 split between male and female users, which greatly surpasses the national average for engineering schools and is one of the highest female percentages in engineering among all universities nationwide. The OEDK wants to share this statistic to show that the OEDK supports women in STEM and encourages female students to pursue engineering. Finally, we will include an exhibit on our display showing all the student clubs and organizations that are working on projects at the OEDK. By incorporating this exhibit, we increase transparency between the OEDK and its sponsors by showing the great work students are accomplishing at the OEDK and it shows prospective students the cool projects they can be working on and get involved with once they come to Rice. We designed our display to include electromechanical components with a cohesive engineering-based theme to make it both visually appealing and engaging **(Figure 2)**.

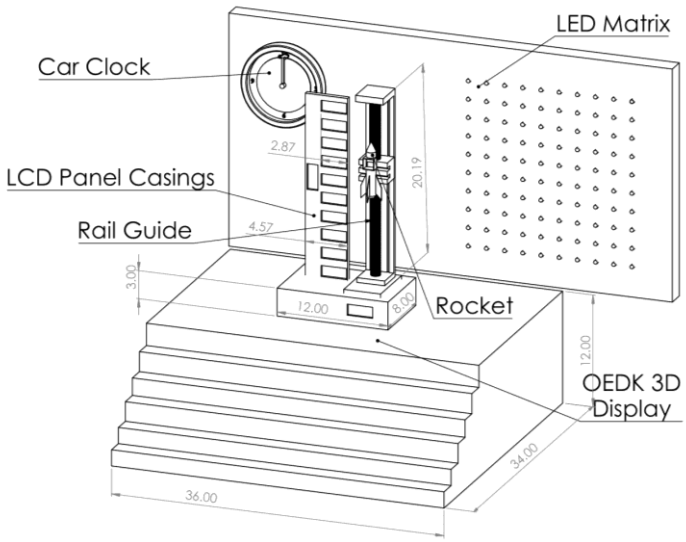


Figure 2: *Conceptual design CAD model of OEDK data display.*

A. Rocket

To showcase the number of yearly visitors, we will use a bar graph data visualization method **(Figure 3)**. We use a 3D printed rocket that moves up and down on a y-axis to show how many people visited the OEDK in a given year and it will continuously cycle through the years in our dataset **(Figure 4)**. The purpose of using a rocket is to represent Rice Engineering, the Rice ECLIPSE club, and the Houston area in general while simultaneously providing an engaging data visualization.

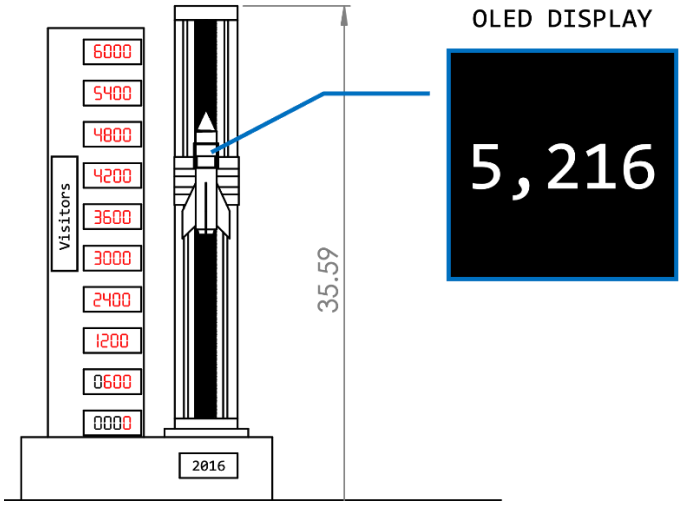


Figure 3: *Rocket bar graph data visualization.*

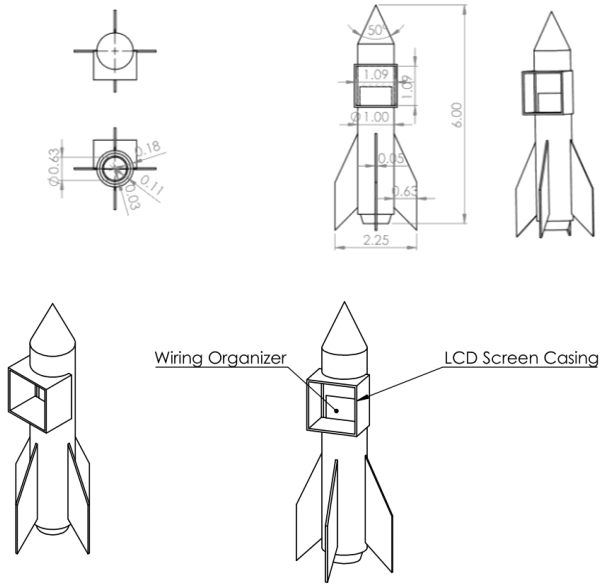


Figure 4: *Rocket concept design CAD model for OEDK data display.*

B. Car Clock

To showcase the rush hour traffic of the OEDK, we will use a clock with a heatmap data visualization method **(Figure 5)**. We use a racecar theme to represent the Rice Electric Vehicle (REV) club and add to the engineering aesthetic of the display **(Figure 6)**. It will showcase a "hotness" for each hour around the clock to indicate how busy the OEDK is at that given time.

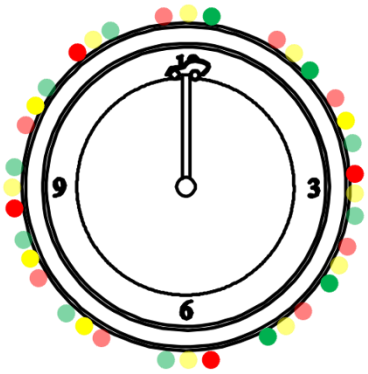


Figure 5: *Car clock heat map visualization.*

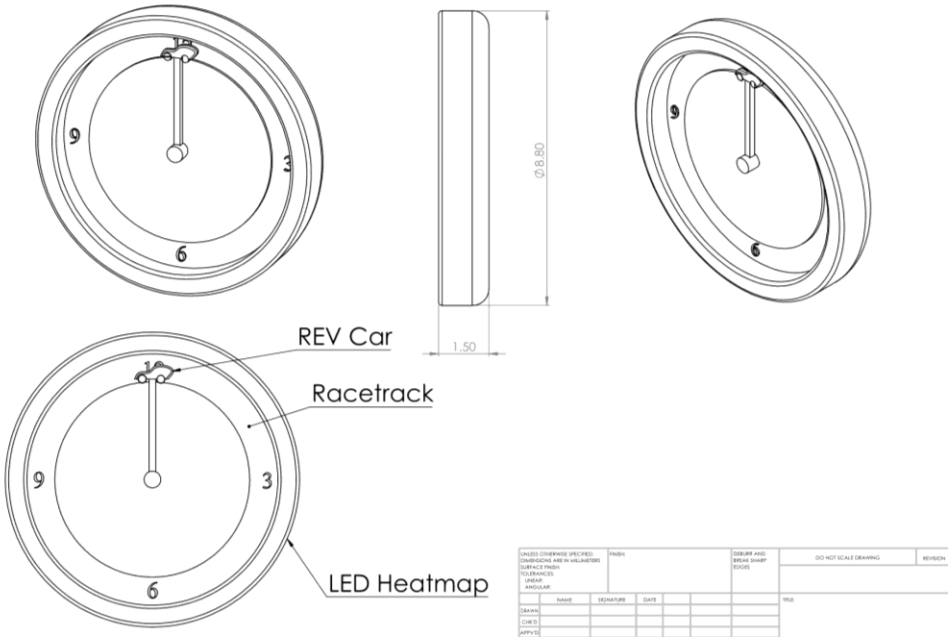


Figure 6: *Car clock conceptual design CAD model for OEDK data display.*

C. LED Matrix

To showcase the male to female ratio, we will use a 10x10 LED matrix with a pictogram data visualization method **(Figure 7)**. The LEDs representing the male OEDK users will be colored one

color and the LEDs representing the female OEDK users will be colored another color. The LEDs will follow a starry night theme to match the rocket.

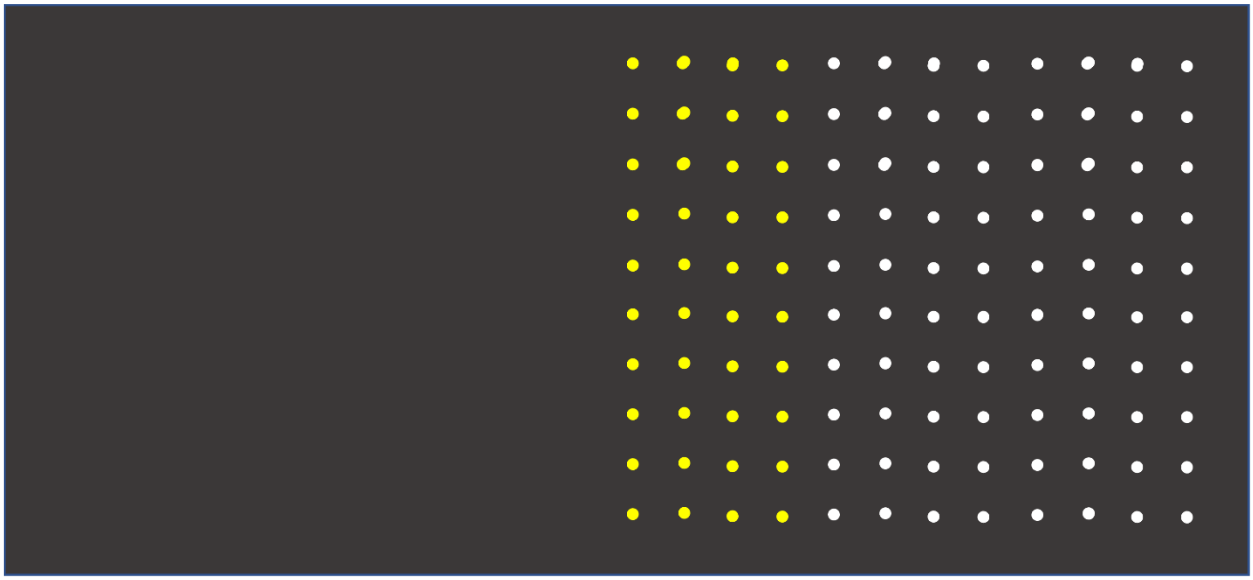


Figure 7: *LED matrix male to female ratio data visualization. Females are indicated by yellow LEDs and males are indicate by white LEDs.*

IV. PROTOTYPING

A. Low-Fidelity Prototype

To assess the conceptual design, we developed a low-fidelity representational prototype **(Figure 8)**. This prototype was built out of readily available materials at the OEDK such as cardboard, foam, plastic mesh, construction paper, and sticks. In this prototype, we included all of the parts from the conceptual design and from this we learned there were a couple of drawbacks in our conceptual design. Mainly, the staircase design in the concept did not translate well in practice. We concluded that the staircase was not apparent that it was representing the OEDK, defeating its purpose, so we decided to forego this component in our next prototypes. Also, we noticed that we need to reconsider how we arrange and organize the electronics for the display.

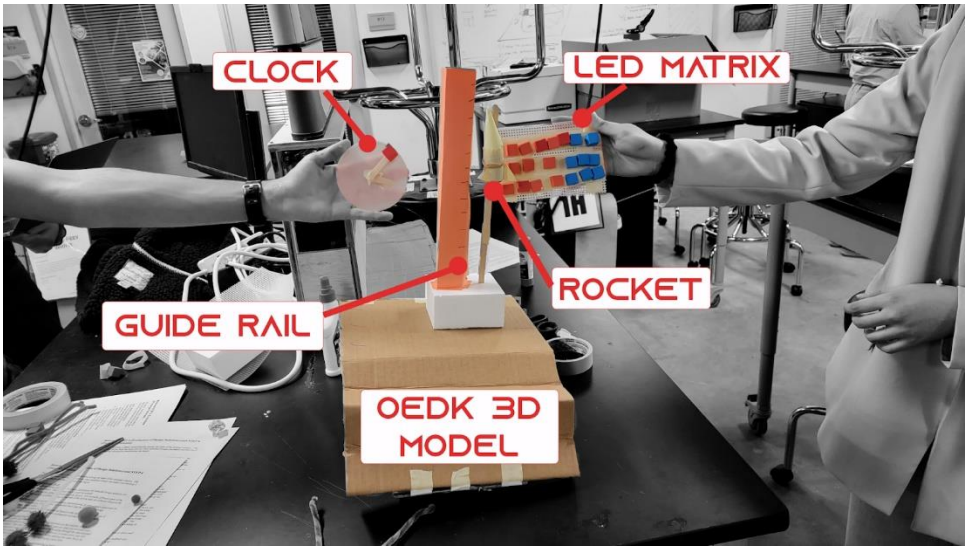


Figure 8: *Low-fidelity prototype for OEDK data display.*

B. Medium-Fidelity Prototype

Being more informed from our takeaways from the low-fidelity prototype, we commenced working on the medium-fidelity prototype. This prototype was a functional prototype, with the basic functionalities of the conceptual design working independent of each other **(Figure 9)**. We included more advanced materials in this prototype including a Raspberry Pi, 4 Arduinos (2 Unos and 2 Megas), LEDs, a clock, 3D models, and a cork base. The most notable changes we made from the conceptual are replacing the staircase design of the OEDK base to a simple box with foam lettering decorating the perimeter. We successfully added the rocket movement, LCD, OLED, and 7-

segment screen displays, clock heatmap, and we began working on an ultrasonic sensor system for collecting data for the clock heatmap.

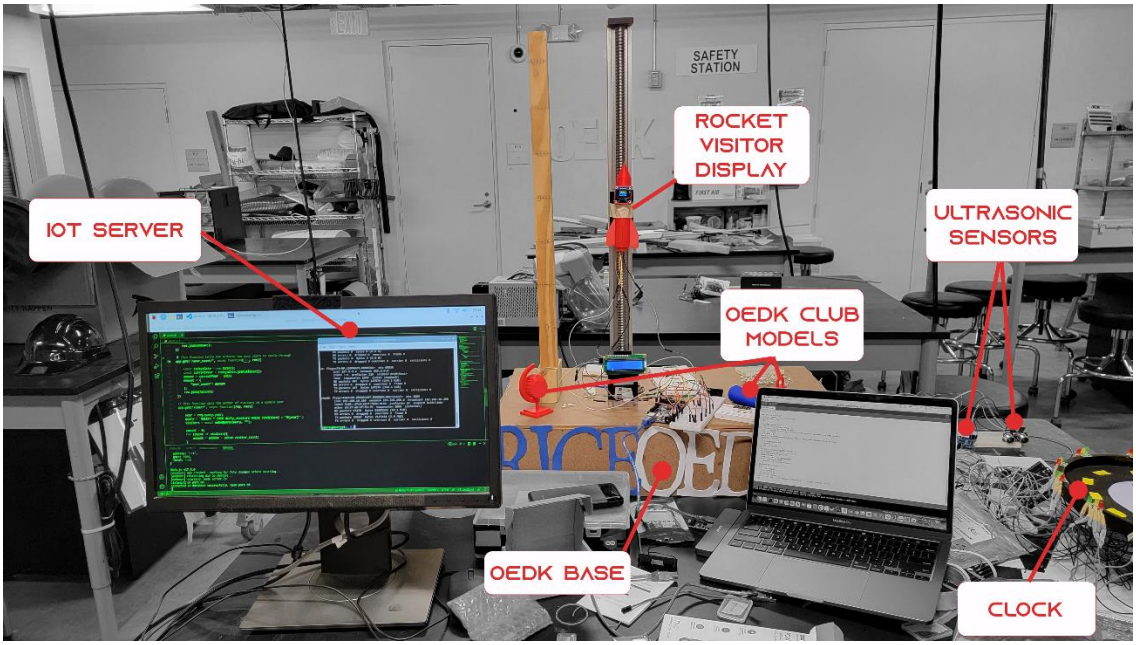


Figure 9: *Medium-fidelity prototype for OEDK data display.*

3D Printing

We 3D printed the models representing the various clubs and organizations that work at the OEDK. We included a model globe representing the Engineers Without Borders program, a model prosthetic hand representing the ENABLE prosthetics club, a model lock and key representing the Rice Escape club, a model car for the clock representing the REV club, and a model rocket for the screen representing the ECLIPSE rocketry club.

Figure 10: *Rocket - Rice ECLIPSE.*

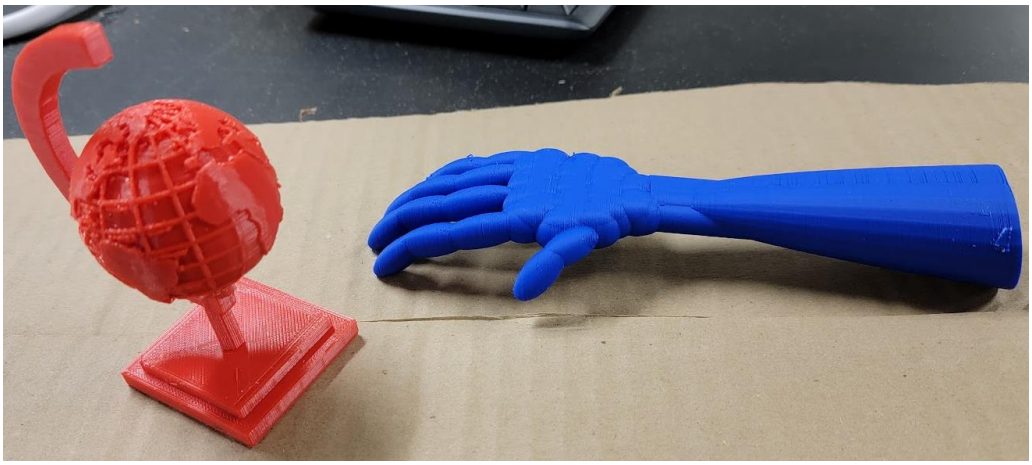


Figure 11: *Globe - Engineers Without Borders (left), Hand - ENABLE Prosthetics (right).*

Figure 12: *Car - Rice Electric Vehicle.*

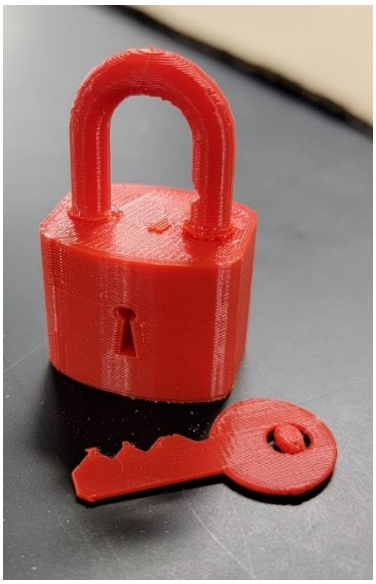


Figure 13: *Lock and key - Rice Escape.*

Rocket Functionality

The rocket functionality was built with 2 LCD screens, an OLED screen, an SD card module, an ESP-01 Wi-Fi module, a stepper motor driver, stepper motor, and an Arduino Mega **(Figure 14)**. The y-axis 7-segment displays had not arrived yet, so we made a makeshift y-axis using a wood plank and we made the tick marks using a pencil. The rocket circuit is the most complex circuit of all the circuits included in our design since it will drive 13 screens and a motor, and it requires two power supplies – one to power the Arduino controller circuit and another 24V to power the stepper motor **(Figure 15)**. To retrieve data, the Arduino Mega pulls the data from a localhost server database over the Internet of Things (IoT) backend via the ESP-01 Wi-Fi module. If the IoT connection is disrupted, the Arduino falls back to retrieving data from the SD card failsafe. This ensures the display is always able to showcase the latest data accurately. The Arduino continuously cycles the yearly data to keep the rocket moving the majority of the time and keep users engaged. To drive the electronics, the Arduino updates the LCD screen to display the year that is currently being shown and updates the OLED screen to display the exact number of visitors that came that year. The motor is controlled by the Arduino with a custom algorithm that dynamically converts the number of visitors to the respective number of steps the motor should move.

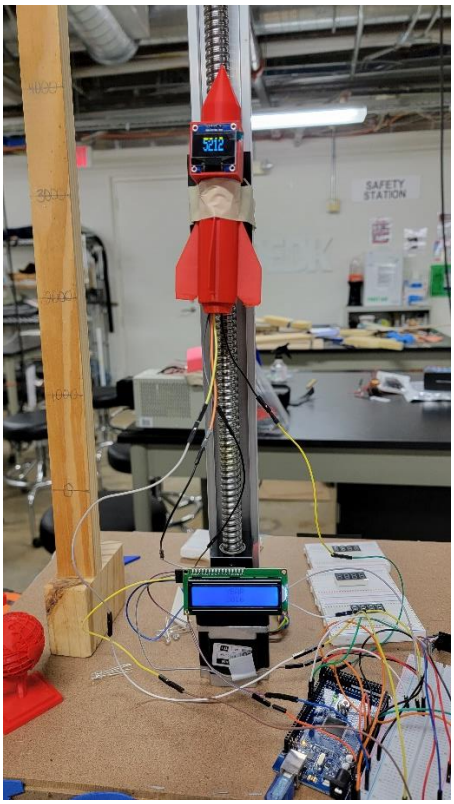


Figure 14: *Rocket mid-fidelity prototype.*

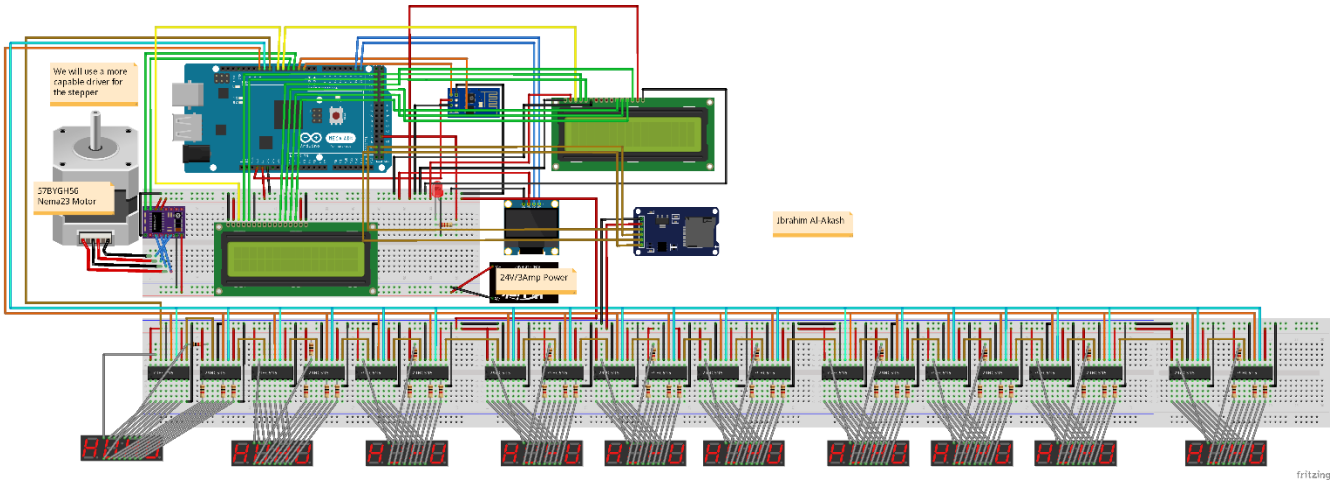


Figure 15: *Rocket circuit diagram.*

Car Clock Functionality

The car clock functionality is built with 36 total 5mm LEDs (12 LEDs each of green, red, and yellow color), an SD card module, an ESP-01 Wi-Fi module, and an Arduino Uno **(Figure 16)**. The circuit was designed to retrieve data from the IoT backend once we completed the ultrasonic sensor system. However, since we were taking longer to figure out the ultrasonic sensor system

than expected, we tested the circuit using a randomized data sample to output a sample traffic heatmap. To simplify the software development, we iterated the clock circuit 3 times. The first iteration of the circuit utilized 16-channel analog multiplexers to drive the LEDs, which made the software cumbersome to work with and used up many pins on the Arduino **(Figure 17, Figure 18)**. The next iteration of the circuit replaced the multiplexers with shift registers **(Figure 19)**. This greatly simplified the programming and used up less pins on the Arduino, making it possible to downgrade from requiring an Arduino Mega to a simpler Arduino Uno, cutting down on costs. Finally, the third iteration of the circuit incorporated both, the ESP-01 Wi-Fi module and the SD card module for data retrieval **(Figure 20, Figure 21)**. It is important to note that soldering all the LEDs to each other was very time consuming and tedious, with a lot of potential points to mess up the circuit. Therefore, we took this into consideration for the LED matrix, which includes more than three times the number of LEDs, adding up to six times more wiring than the clock circuit. To streamline the matrix development, we opted to transfer from using 5mm RGB LEDs to using neopixels, which would make it much easier and quicker to set up, prototype, and test the circuit for the high-fidelity prototype.

Figure 16: *Clock mid-fidelity prototype.*

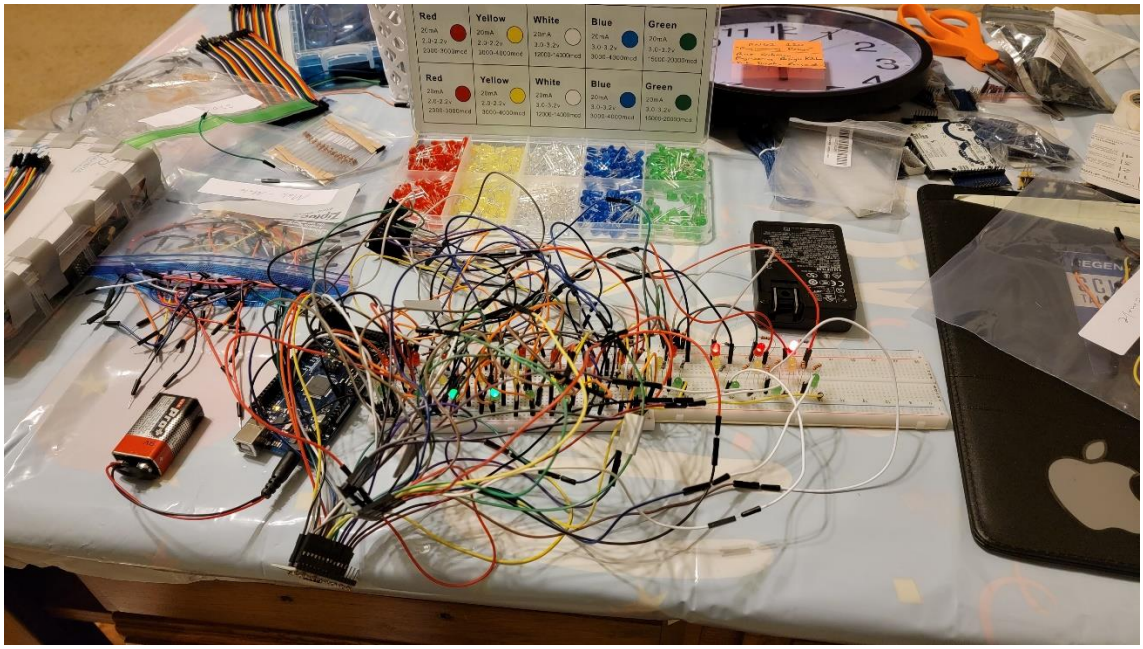


Figure 17: *Breadboard testing iteration #1 of clock circuit with 16-channel analog multiplexers.*

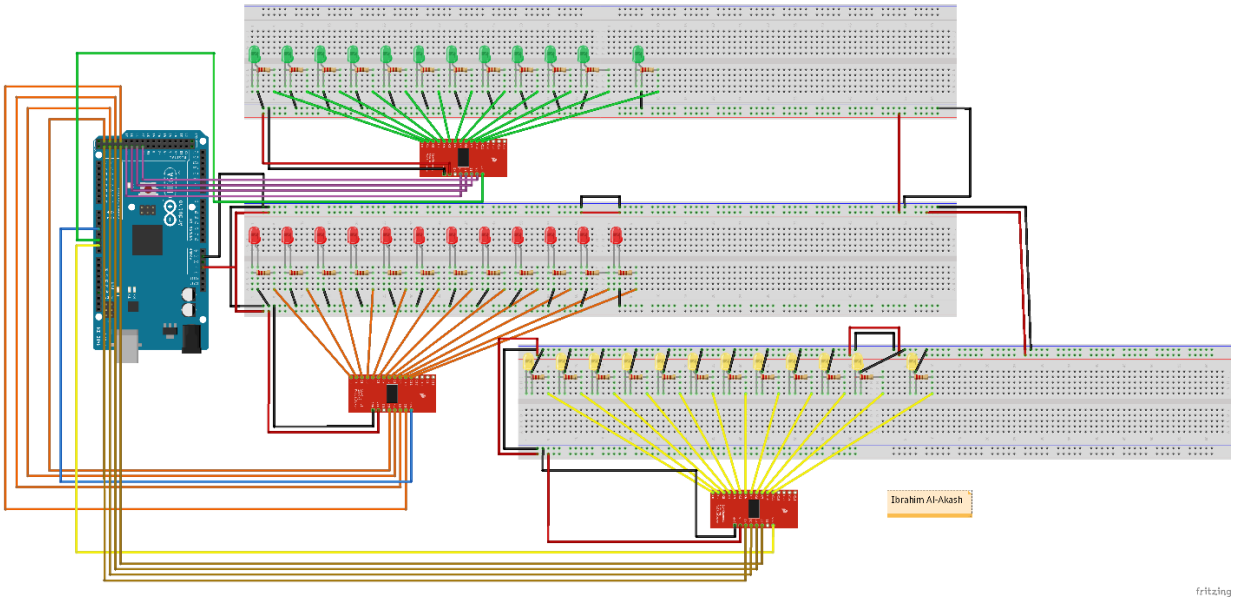


Figure 18: *Circuit diagram for iteration #1 of clock.*

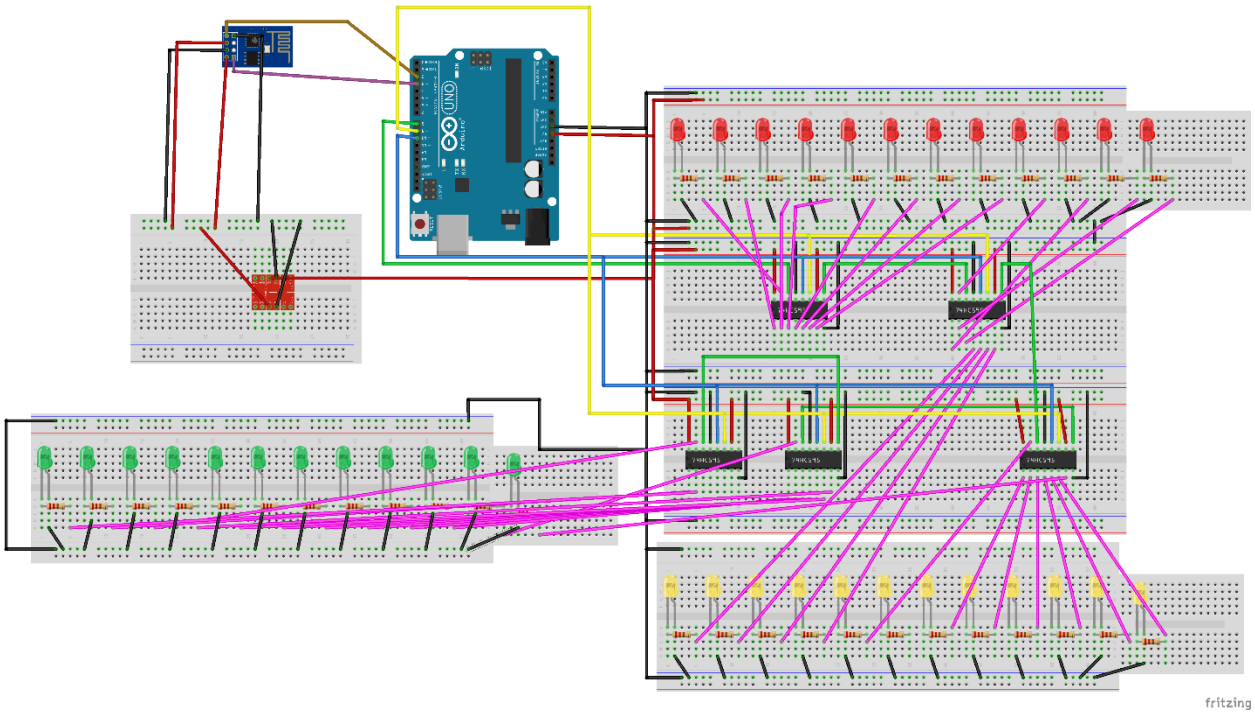


Figure 19: *Circuit diagram for iteration #2 of clock.*

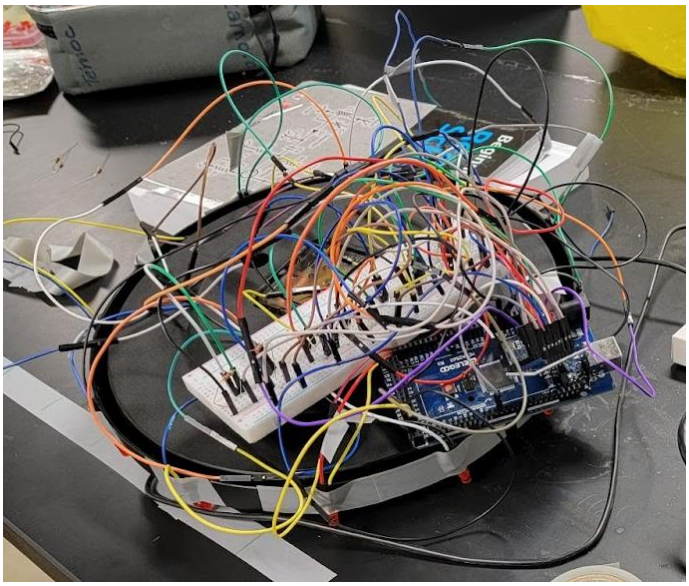


Figure 20: *Completed mid-fidelity clock prototype wiring.*

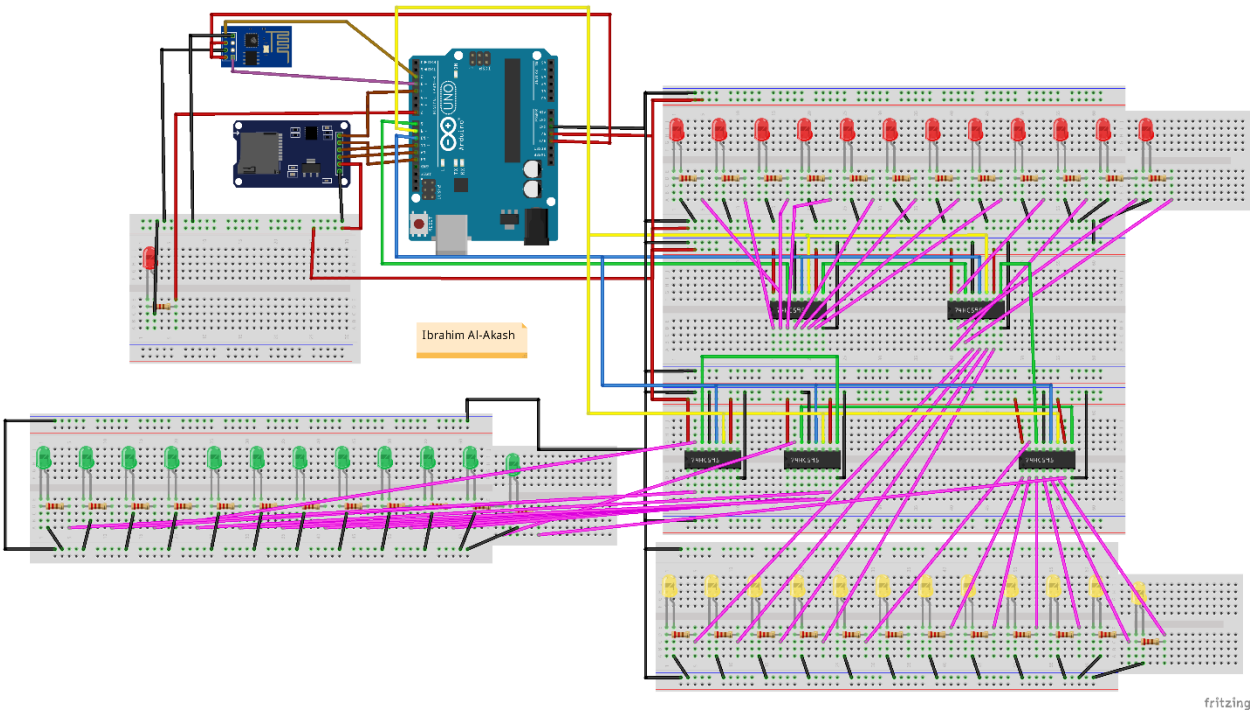


Figure 21: *Circuit diagram for iteration #3 of clock.*

Ultrasonic Sensor System Functionality

To enable the clock to show real-time data, we developed an ultrasonic sensor system which would count the number of visitors as they come in and subtract them as they leave the kitchen **(Figure 22, Figure 23)**. The sensor system consists of 2 sensors placed in front of each other. The sensors will have a set distance measurement threshold, which will be determined during testing, that will record a visitor when they walk in the OEDK. In scenario 1, a user enters the OEDK triggering the first sensor, S1, (highlighted in green) and the Arduino updates the database via the ESP8266 Wi-Fi module. In scenario 2, a user exits the OEDK triggering the second sensor, S2, (highlighted in green), and the Arduino will not update the database to prevent counting users twice **(Figure 24)**. We were planning on placing the sensor system near the OEDK entrance **(Figure 25)**. This would, in theory, track the occupancy of the OEDK throughout the day with timestamps. The data would be uploaded to the server via IoT, and the clock's Arduino will retrieve it, interpret the data, and output the correct rush hour traffic heatmap to the corresponding LEDs on the clock. However, during testing, we were encountering issues in which the system sometimes failed to count a visitor or failed to subtract a visitor if they were walking at a pace that was too fast or too slow. Due to time constraints, we discontinued work on this aspect of the project and pivoted to using historical data for the clock rush hour traffic heatmap data visualization.

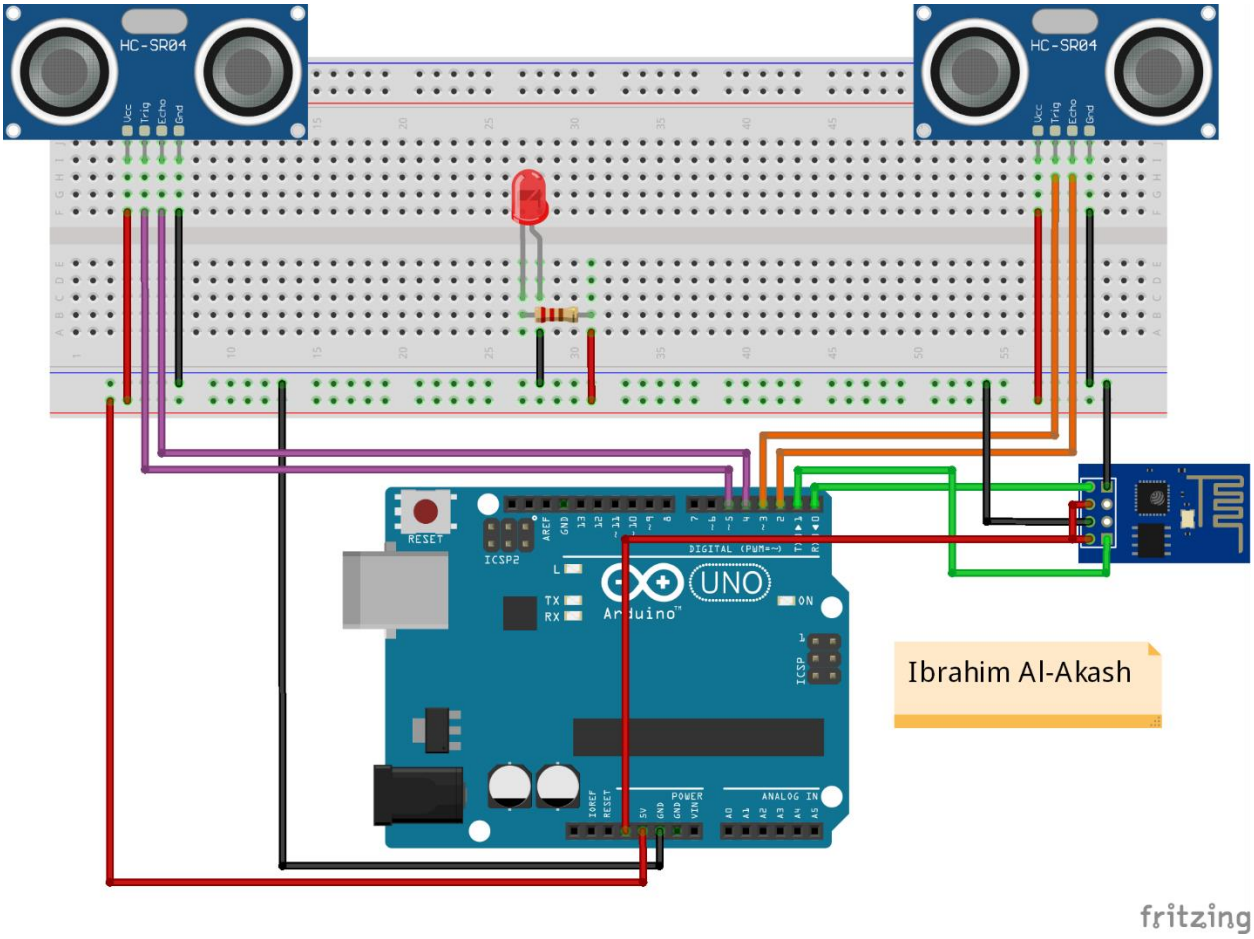


Figure 22: *Ultrasonic sensor system circuit diagram.*

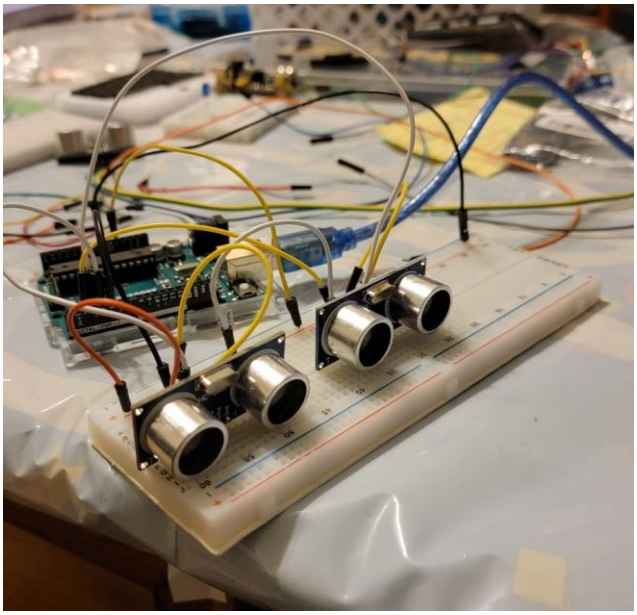


Figure 23: *Ultrasonic sensor system prototype.*

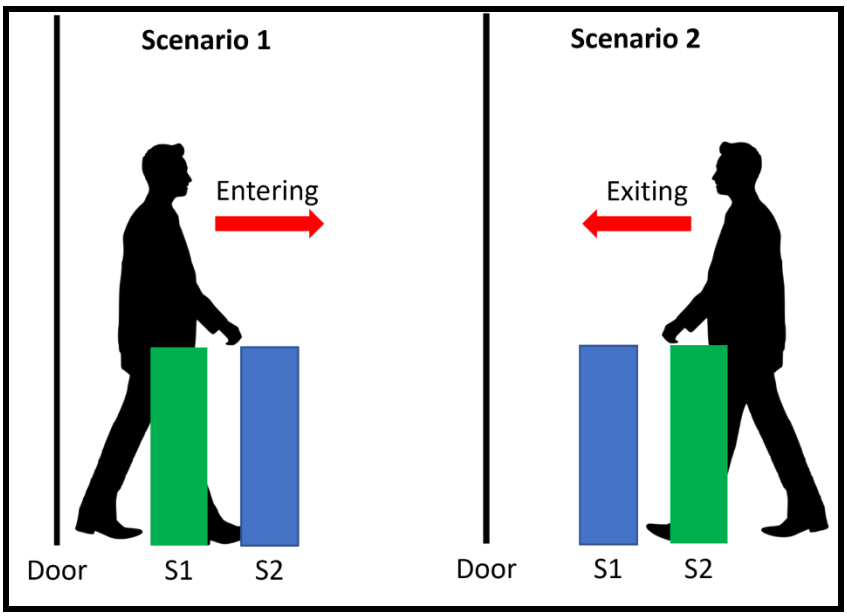


Figure 24: *Ultrasonic sensor system counting methodology.*

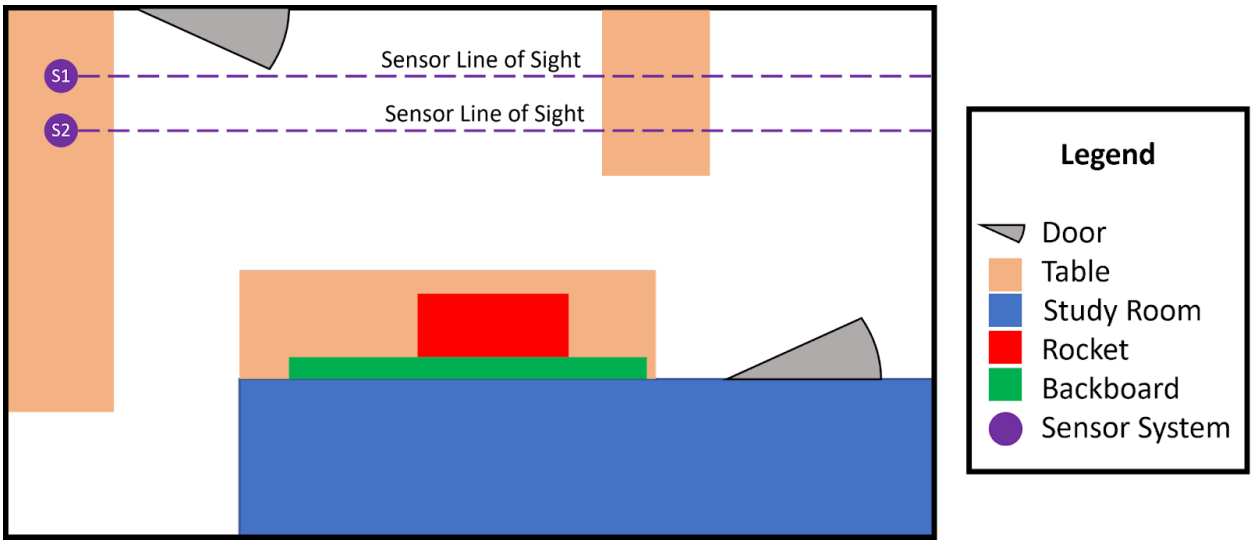


Figure 25: *Sensor system location in OEDK lobby.*

Internet of Things (IoT) Functionality

To facilitate updateability of data in our display, we planned on utilizing IoT capabilities to update, store, and retrieve OEDK statistics and usage data **(Figure 26)**. In order to add this functionality, we programmed an application programmable interface (API) to make updating and retrieving data to the Arduino a seamless process. Then, we converted the raw comma separated value (CSV) file that we received from the OEDK into a MySQL database table. This makes it easier to compute logic for the Arduino and display various aspects of data, such as calculating the number

of visitors per year, per hour, or per day and organizing the data by sorting using different parameters such as timestamp or quantity of visitors. The next step was to flash a Raspberry Pi, download the necessary dependencies including NodeJS, MySQL, Visual Studio Code, and PM2 and download the API and server code. When prototyping, we realized that the default Wi-Fi at the OEDK is not always operating on 2.4GHz frequency, which is the frequency of the ESP-01 Wi-Fi module of the Arduino. This was causing issues with connectivity, resulting in a 50/50 success rate. To solve this issue, we set up our own Wi-Fi router to operate only on 2.4GHz. The Raspberry Pi then hosts the backend on this Wi-Fi router **(Figure 27)**. The PM2 dependency on the Raspberry Pi resets the server in the unlikely case that it crashes, adding some extra cushion for errors that may arise, mitigating the need to fall back to the SD card offline failsafe. We tested the IoT pipeline for the rocket circuit and successfully transmitted data from the server backend to display on the OLED screen **(Figure 28)**.

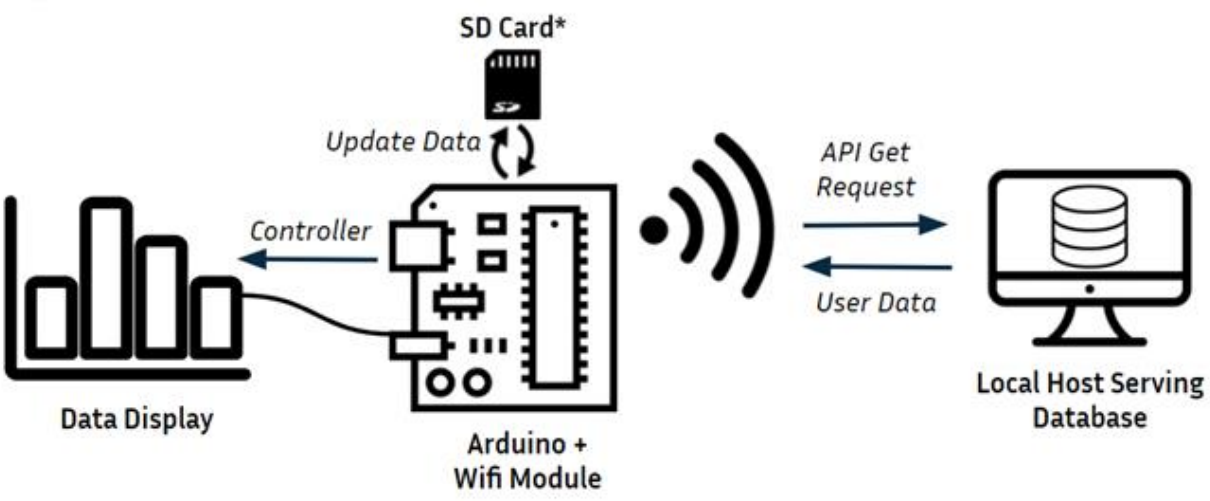


Figure 26: *Internet of Things software pipeline.*

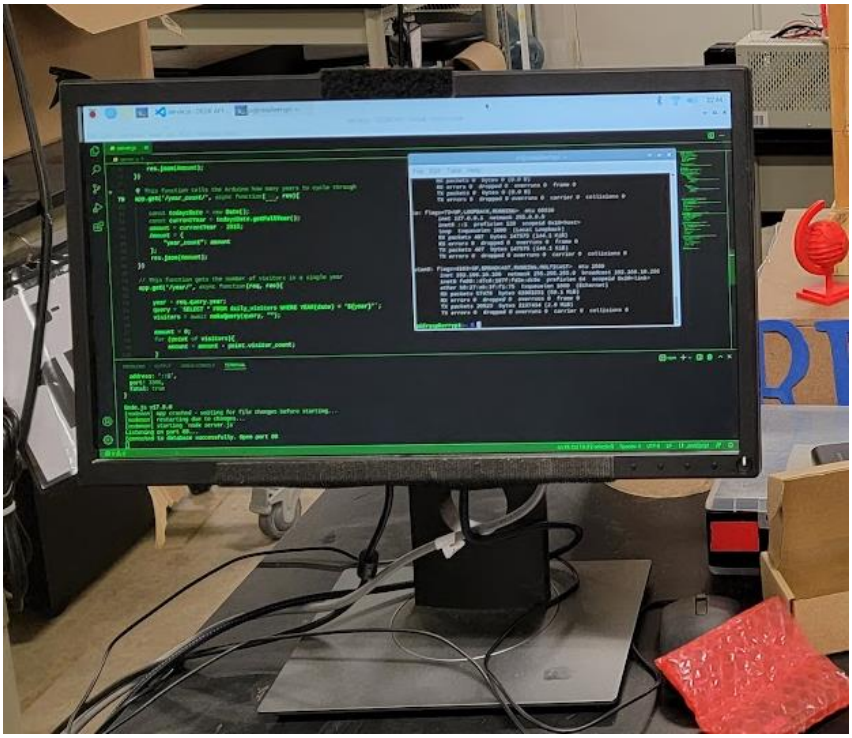


Figure 27: *Raspberry Pi server running with database hosting.*

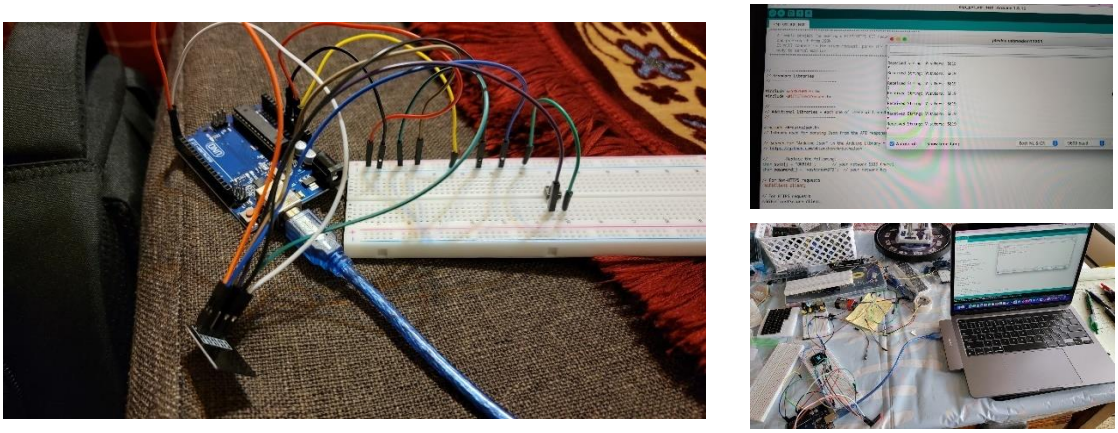


Figure 28: *IoT prototype testing.*

For the LED matrix, we were still waiting for supplies to arrive, so we did not have a physical prototype yet. However, we had planned out the circuit design and started working on the software **(Figure 29)**. The mid-fidelity prototype helped us better understand the challenges of some of the components and better plan out the next steps to build the high-fidelity prototype. Due to time and budget constraints, we simplified the design for our display for the high-fidelity prototype. Some of the key takeaways from this stage in the EDP were 1) circuit designs for the various

components of our system had evolved into simpler circuits that are easier to program, 2) will use neopixels instead of 5mm RGB LEDs for the matrix, and 3) we will simplify the display to work only in offline mode and remove the sensor system.

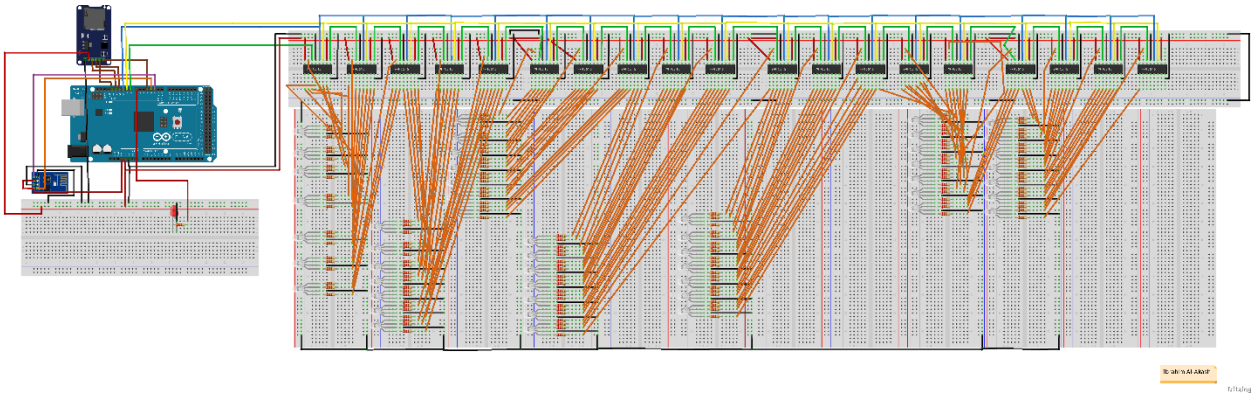


Figure 29: *LED matrix circuit diagram.*

C. High Fidelity Prototype

Having refined our design from the mid-fidelity prototype, we started working on our high-fidelity prototype. In this prototype, we simplified some of the complicated parts of our mid-fidelity prototype and focused more on the aesthetics of the display, polishing the prototype and adding more visually appealing elements **(Figure 30)**.

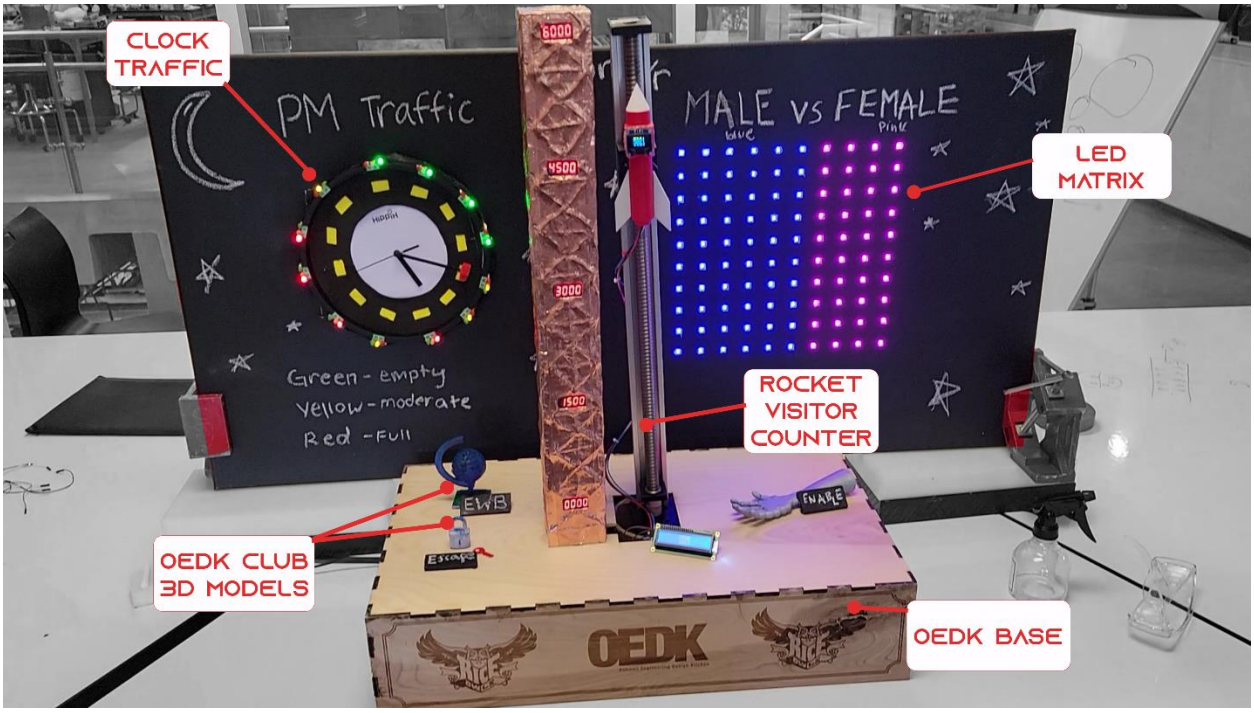


Figure 30: *High-fidelity prototype for OEDK data display.*

Rocket

To improve the aesthetics of the rocket and make it look more finished and polished, we painted white accents on the nosecone and fins. Additionally, our order of 7-segment displays arrived, and we built the y-axis using wood and copper tape. We laser cut the wood to leave openings for the 7-segment displays to show through. To follow the theme of a rocket display, we were striving to make the y-axis look like the support structure that NASA uses to hold up their rockets. We achieved this look by first placing hot glue on the chassis to look like a frame structure and then covering it with copper tape. This added to the visual appeal of our display and improved the cohesion of the different components of the display, effectively improving the intuitive aspect as well.

Clock

To make the clock look like a racetrack, we pasted foam boards on the center of the clock to have yellow lane markings with black asphalt **(Figure 31)**. Then, we designed the hour markings to look like street signs and super glued them on the perimeter of the clock **(Figure 32)**. We used electrical tape, which matched the color of the clock, to hide the visible wiring on the perimeter of the clock. To calculate the rush hour traffic heatmap, the clock uses an SD card with a list of the

aggregate historical data for how many visitors walk in at each hour of the day that has been collected over the past 7 years. Then, using a custom-written program, the Arduino dynamically labels the data into low, medium, and high traffic for each hour and outputs the results to the clock. If the LED is red for a given hour, it means that the OEDK is full. If it is green, it means it is empty, and if it is yellow, it means the OEDK is moderately occupied.

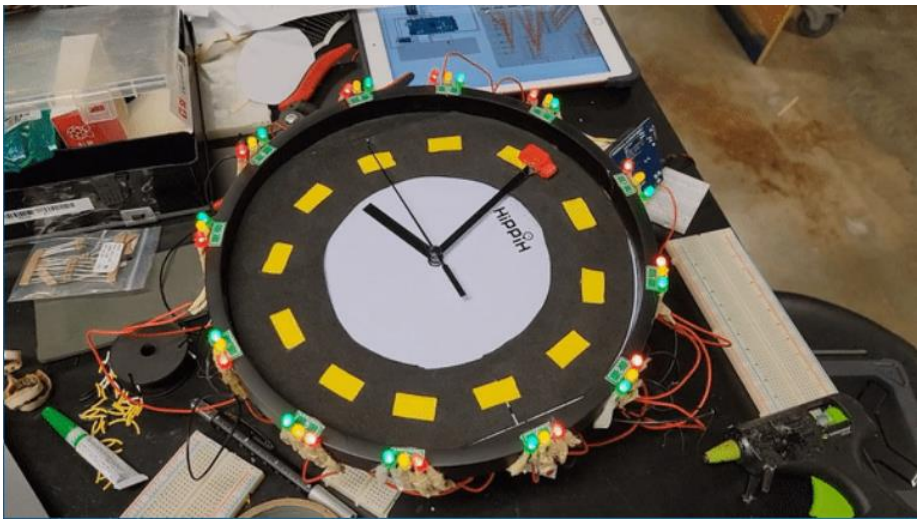


Figure 31: *High-fidelity clock prototype.*

Figure 32: *Sample street sign-themed hour marker for 12 O'clock.*

LED Matrix

To build the LED matrix, we used neopixels which were very simple to set up and program. During testing, we realized that if we the matrix was static and only showing the male to female ratio, we would not be using the matrix to its fullest potential. By adding some supplementary animations such as displaying the Rice logo, Texas flag, and rainbow, and cycling between these animations and the male to female ratio, we make the display more eye-catching and engaging. Furthermore, now that we had both the matrix and the clock completed, we started working on integrating the two onto the canvas backboard. To make the backboard more appealing and match

the “starry night” theme, we painted it black using a chalk-compatible paint and drew stars and the moon onto the board with chalk. Then, we added titles to all of our data visualizations and drew legends to make the display easier to understand and more intuitive for users **(Figure 33)**.

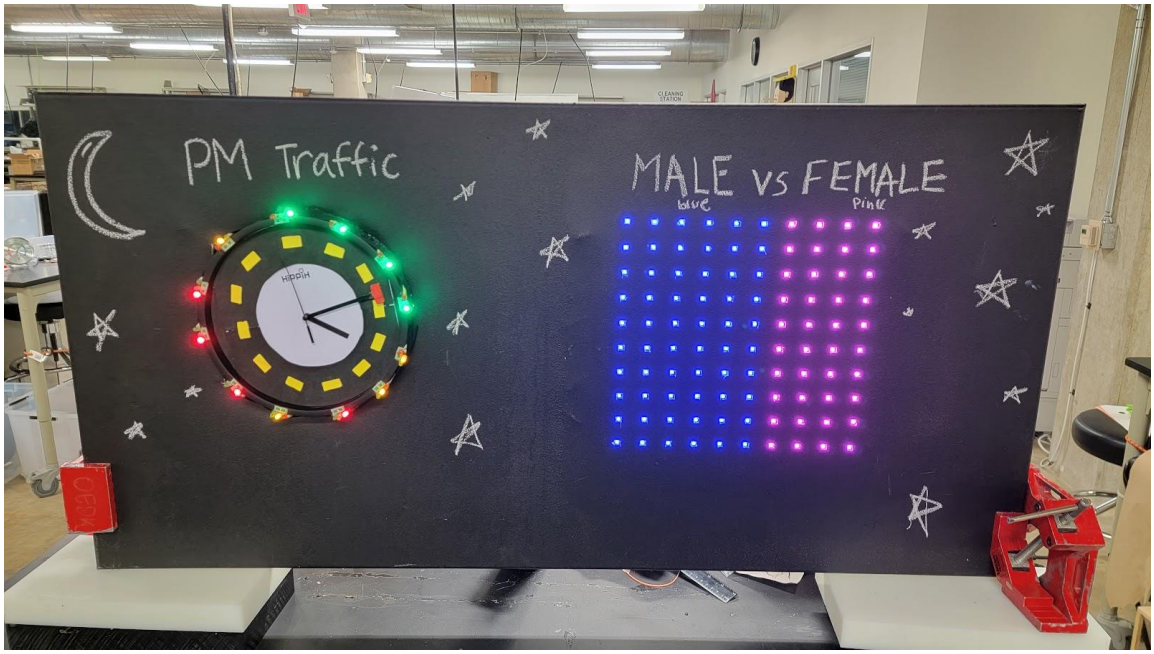


Figure 33: *Canvas backboard with high-fidelity clock and LED matrix prototypes integrated.*

The high-fidelity prototype achieves all of the main functions of the conceptual design. We made a few slight changes to how the system works under the hood, however, from our user’s perspective, there are no differences between the high-fidelity prototype and the conceptual except for us forgoing the staircase design. We were able to accomplish our goals and commence with testing the success of our design against our design criteria.

V. TESTING

Table 3: *Testing methodologies and summary of results.*

Design Criteria	How will it be tested?	How many times will it be repeated?	Who is involved in testing?	Results*
Safety	Get 3 OEDK experts' opinion on the safeness of our display	Until safe	**Participants:** OEDK experts, Tada-Data Team **Testers:** Vivian	Passed
Accuracy	1) Measure the accuracy of the rocket on the y-axis compared to the data value on the OLED 2) Calculate the clock heatmap manually and compare to the display output 3) Ensure sensor system doesn't double count or miscount visitors	Until accuracy of within ±5% is achieved	**Participants:** N/A **Testers:** 1. Ibrahim, Daniel 2. Ibrahim, Leticia 3. 88%Camilo, Vivian	1) Passed 2) Passed 3) Passed
Intuitive to Understand	Additionally, our team will grade each user's understanding of the data with the actual meaning of the	3	**Participants:** 35 OEDK users and visitors **Testers:** Daniel, Leticia, Vivian	Passed

	data such that a higher grade means that the user correctly interpreted the data.			
Visual Appeal	Users will view the display for 2 minutes. Once 2 minutes are up, they will have to rate their experience.	3	**Participants:** 35 OEDK users and visitors **Testers:** Daniel, Leticia, Vivian	Passed
Engaging	Place the display in the front desk. Observe and record (1) if people stop and look at it and (2) if they tell others about it.	2	**Participants:** 35 OEDK users and visitors **Testers:** Daniel, Leticia	Passed
Durability	1) Software: intentionally disconnect the Wi-Fi connection and observe if the offline SD card fail/safe works. 2) Hardware: run the motor overnight for 10 hours and record the error of the motor. If the error is outside the ±5%, it is a	3	**Participants:** N/A **Testers:** 1) Ibrahim, Camil0 2) Leticia, Daniel	1) Passed 2) Passed 3) Passed

	failure. 3] Drop display from a height of one meter which is around the height of a table			
Updateability	Feed display randomly generated data. Test if the display will continue to accurately represent new incoming data.	3	**Participants:** N/A **Testers:** Ibrahim	Passed

* See Appendix A for more information

VI. STATUS OF FINAL DESIGN

The status of our final design has many different fine details that portray cleanliness to our project. The wiring seen in the back of the canvas is now covered by a white Velcro back plate which allows easy access to wiring if necessary but doesn't disturb the aesthetics of the design nor the versatility of the piece. For instance, it can be placed near a clear wall and posters can be pinned on the back side leaving room for more static data display.

Additionally, all the components of our final design have been soldered and the wiring is secured to prevent any breakage when moving.

Moreover, the original design was a hanging backboard like the other posters in the lobby, however as an additional method of accessibility we added laser cut stands to hold the blackboard on top of the box. In this method, the labeling for 'Visitors per year' is not visible on the blackboard so a laser cut sign with the engraved label is placed on top of the y-axis of the model.

VII. CONCLUSION

Our team was tasked with designing a dynamic, engaging and visually pleasing display that accurately models growing data available to increase awareness regarding the achievements and the inclusivity of the OEDK, and we accomplished the goal. By creating this display and showing data that was not public before, we hope to develop a sense of pride in the users of the accomplishment of the OEDK, foster the relationship between the OEDK as an institution and its users, and increase transparency for visitors like prospective students, companies, and sponsors.

Through the whole process we encountered a series of challenges that changed the course of our solution. When we moved on from the mid fidelity to the high-fidelity prototypes, we had a lot of electrical components still unfinished, and they were just adding unnecessary complications to the project. Because of time constraints we refrained from further using Internet of Things to update the data as it was mainly used to improve updatability, our least important objective. Furthermore, we decided not to use ultrasonic sensors because we were not able to figure out a set up that gave us less than 5% accuracy error. Even after these changes the final solution was almost identical to the conceptual design and the project was a success.

Even if we accomplished all our goals, the project still has some improvements that we are working on. By implementing the ultrasonic sensors to count people inside the OEDK we would be able to have real time live data and have a more accurate heat map representing the business of the OEDK. In addition to this, we are trying to figure out a way to display more sets of data regarding the demographic of the OEDK users, like residential college or major, using the matrix in an intuitive way. We also want to show more sets of data using the rocket; however, this requires more LCD screens. We want to improve the aesthetics of it by adding more neopixels around the display to make it more eye-catching. Finally, we want to implement Bluetooth capabilities in the Arduinos to be able to update the data without having to disassemble the display.

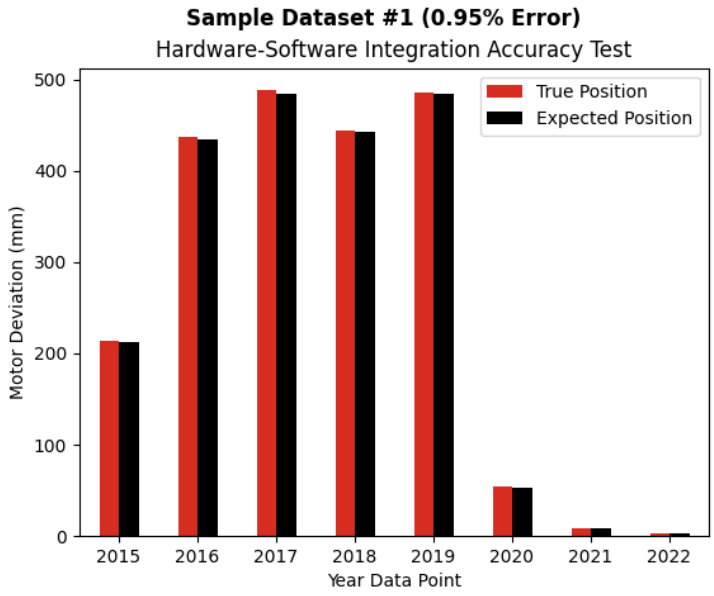
Overall, the whole project was a success as we were able to achieve all our design constraints and objectives. By using high quality components, we ensured that our display was durable and able to accurately show the growing data of the OEDK in an engaging and appealing way.

APPENDIX A: TESTING AND RESULTS

Safety

	Safe	Unsafe
OEDK Expert #1	✓	
OEDK Expert #2	✓	
OEDK Expert #3	✓	

Accuracy and Updateability

	Collected Data
Test 1	

Test 2

Hour (PM)	Visitor Count	Calculated Color	True Color
12	207	GREEN	GREEN
1	1546	GREEN	GREEN
2	4359	RED	RED
3	5393	RED	RED
4	3519	YELLOW	YELLOW
5	1616	GREEN	GREEN
RESULT	PASS ✔		

Test 3

Trial	Male:Female	Calculated Percentage	True Pink LED Percentage
1	2545:2455	49%	49%
2	2512:2488	50%	50%
3	2465:2535	51%	51%
4	2501:2499	50%	50%
5	2520:2480	50%	50%
RESULT	PASS ✔		

Intuitive to Understand (IU)

- Graded and recorded users' interpretation of data on a scale of 1-100%.

Visual Appeal (VA)

- Recorded users' visual experience on a scale of 1-5

Engaging (E)

- Recorded Yes or No (Y/N) whether or not a user stopped and looked at display

Participant	IU	VA	E
1	85	4	Y

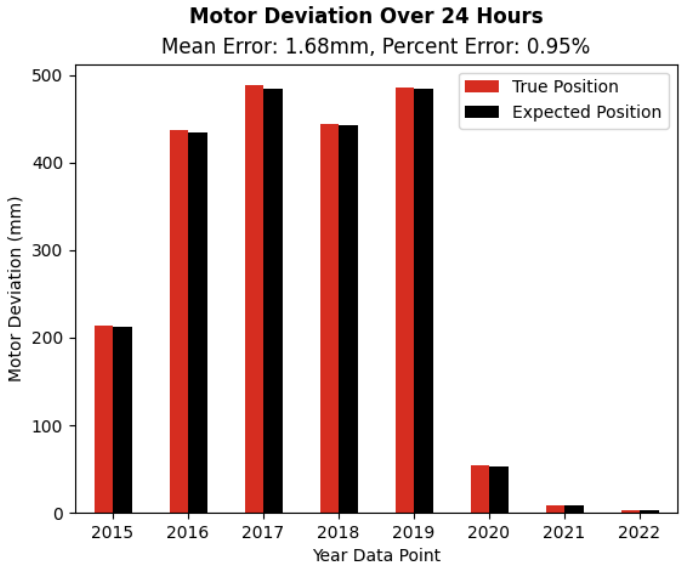
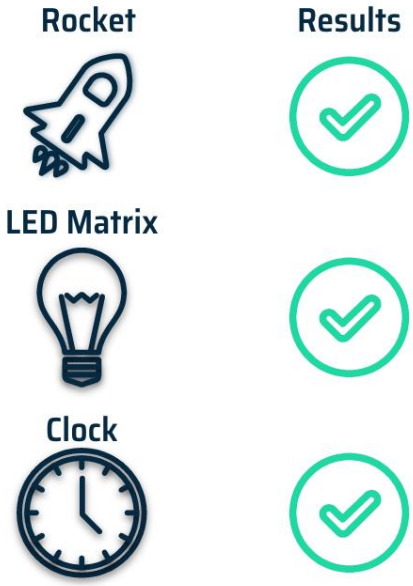
Participant	IU	VA	E
22	90	5	Y
23	90	5	Y

2	95	5	Y
3	80	5	Y
4	70	5	Y
5	90	4.5	Y
6	–	–	N
7	90	5	Y
8	95	5	Y
9	95	5	Y
10	85	5	Y
11	–	–	N
12	–	–	N
13	90	5	Y
14	85	5	Y
15	90	5	Y
16	90	5	Y
17	80	4	Y
18	80	4	Y
19	95	5	Y
20	90	5	Y
21	90	5	Y

24	85	5	Y
25	100	4.5	Y
26	–	–	N
27	85	5	Y
28	90	5	Y
29	–	–	N
30	95	5	Y
31	90	5	Y
32	80	4	Y
33	–	–	N
34	90	5	Y
35	90	5	Y
36	85	4	Y
37	90	4	Y
38	90	5	Y
39	75	5	Y
40	90	5	Y
41	90	5	Y
Mean	88%	4.8	82.9%

Durable

Collected Data

Test 1	
Test 2	
Test 3	

APPENDIX B: SUPPORTING RESEARCH

Problem Context Area:

Exploration of existing solutions Governs the problem Background Business perspective

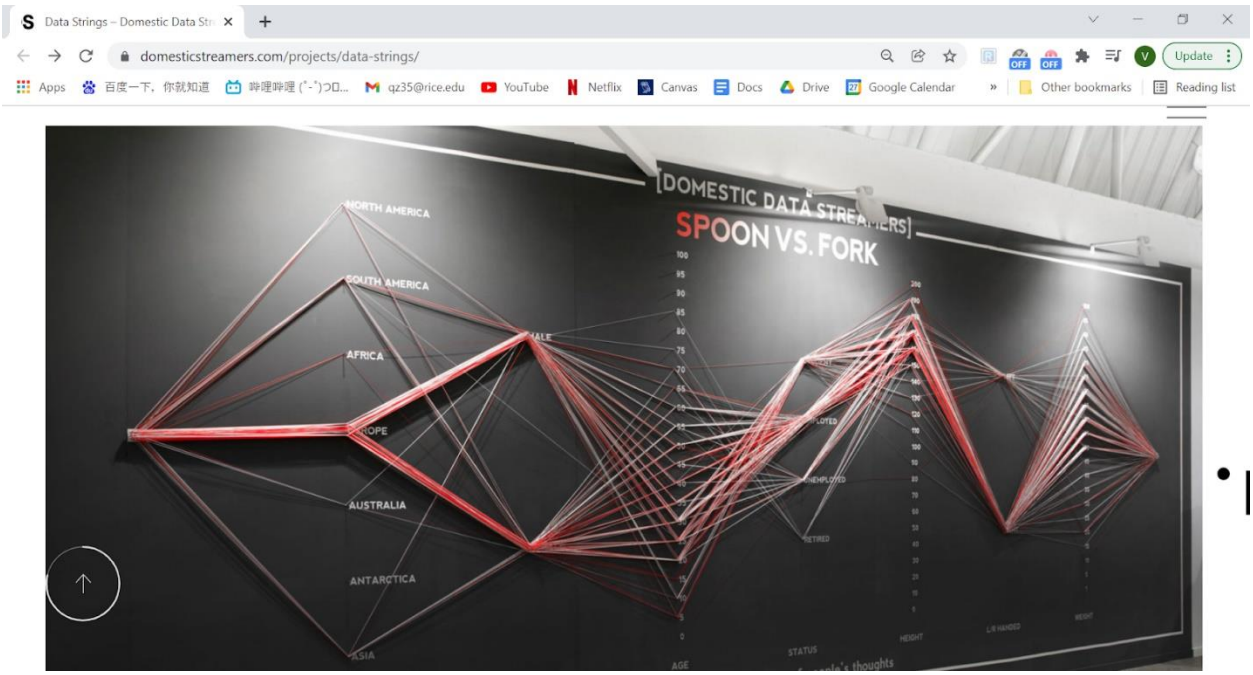
Complete Citation/Source:

Data strings. Domestic Data Streamers. Accessed January 28, 2022. https://domesticstreamers.com/projects/data-strings/

Is this source peer-reviewed: Yes No

Summary of Key Points:

- Data strings: interactive poll display. "A physical visualization of people's opinion to use anywhere, anyhow."
- Participants are asked to knit their answers to a series of questions, generating a physical flow chart of the public opinion at large.
- More than 65 live displays around the world.



Why is this source important to the project?

- A past solution to a similar problem.
- We could draw inspiration from this system for our design
- It is engaging, dynamic, and visually appealing - key features that we wish to include to our solution.

Problem Context Area:

Exploration of existing solutions	Governs the problem	Background	Business perspective

Complete Citation/Source:

Actuators explained: Types of actuators, Application Choice, maintenance. Control Engineering. (2021, August 6). Retrieved January 28, 2022, from https://www.controleng.com/articles/actuators-explained-types-of-actuators-application-choice-maintenance/

Is this source peer-reviewed: Yes No

Summary of Key Points:

The author describes different types of actuators, their uses, and how to select the appropriate actuator for an engineering project.

Why is this source important to the project?

This is important because it allows us to develop decision-making guidelines for when we are deciding on the materials and components of our solution.

Problem Context Area:

Exploration of existing solutions | Governs the problem | Background | Business perspective

Complete Citation/Source:

Yvonne Jansen. Physical and tangible information visualization. Other [cs.OH]. Universit´e Paris Sud - Paris XI, 2014. English. <NNT : 2014PA112039>. <tel-00981521>

Is this source peer-reviewed): Yes | No

Summary of Key Points:

- A thesis on physical and tangible information visualization
- provides an exploration of possible benefits and current limitations of physical and tangible information visualizations
- Provides formal methods to describe, compare, and criticize such systems, and tools to support users in creating such systems.
- Includes several case studies about the advantages and limitations of physical visualization.

Why is this source important to the project?

- Covers the most essential challenge of our problem - physical representation of data - in great detail.

Problem Context Area:

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

Saiganesh Swaminathan, Conglei Shi, Yvonne Jansen, Pierre Dragicevic, Lora Oehlberg, et al.. Supporting the Design and Fabrication of Physical Visualizations. Proceedings of the 2014 Annual Conference on Human Factors in Computing Systems (CHI 2014), ACM, Apr 2014, Toronto, ON, Canada. pp.3845–3854, ff10.1145/2556288.2557310ff. ffhal-00935978f

Is this source peer-reviewed: Yes No

Summary of Key Points:

- Three case studies that illustrate limitations of current visualization fabrication workflows
- MakerVis: a prototype tool that integrates the entire process of creating physical visualizations

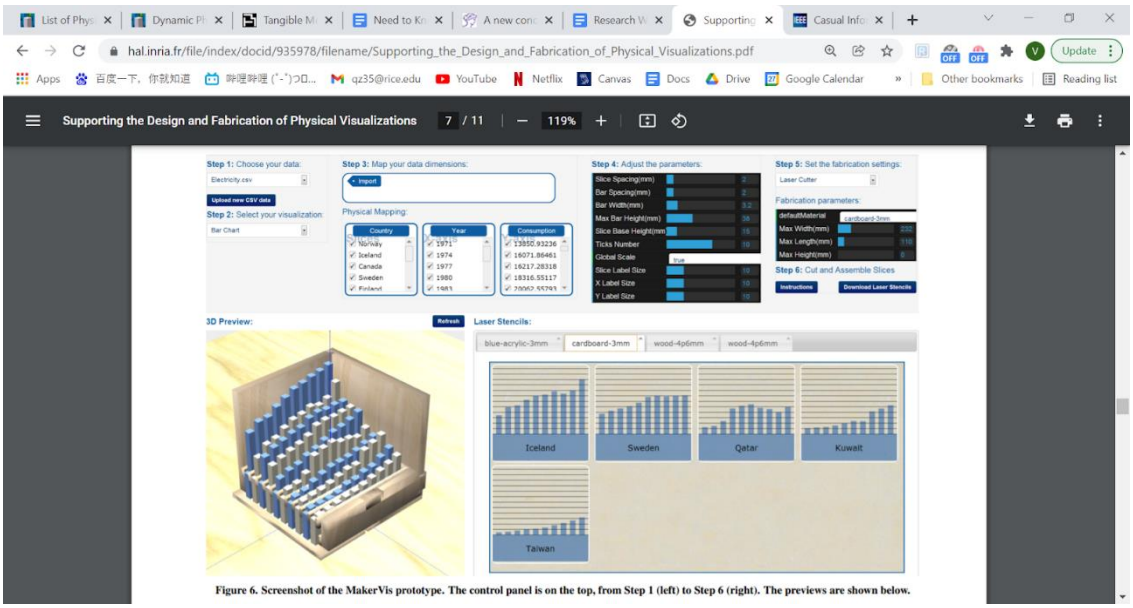


Figure 6. Screenshot of the MakerVis prototype. The control panel is on the top, from Step 1 (left) to Step 6 (right). The previews are shown below.

Problem Context Area (circle one that best fits):

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

Vande Moere, A., & Patel, S. (2009). The physical visualization of information: Designing data sculptures in an educational context. *Visual Information Communication*, 1–23. https://doi.org/10.1007/978-1-4419-0312-9_1

Is this source peer-reviewed: Yes No

Summary of Key Points:

The authors detail that useful data sculpture requires a strong design rationale underlying its perceived physical qualities, including: embodiment, metaphorical distance, multi-modality, interaction, affordance and physicality.

Why is this source important to the project?

This is important because it highlights the important features of a physical data display, so we can keep these factors in mind when designing and engineering our solution.

Problem Context Area:

Exploration of existing solutions | Governs the problem | Background | Business perspective

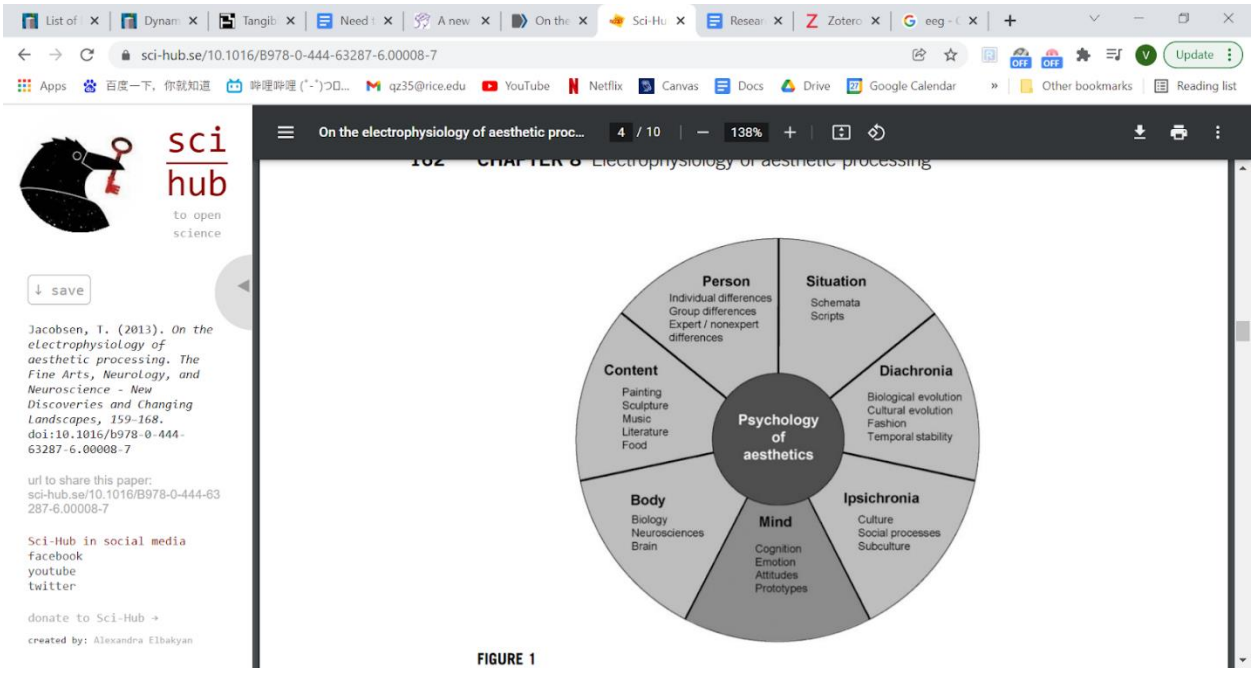
Complete Citation/Source:

Jacobsen T. On the electrophysiology of aesthetic processing. *Prog Brain Res.* 2013;204:159-168. doi:10.1016/B978-0-444-63287-6.00008-7

Is this source peer-reviewed: Yes | No

Summary of Key Points:

- Electrophysiology can be applied to gain an understanding of the underpinning of aesthetics in the brain.
- Electroencephalogram (EEG) is used to measure electrical activity of the brain. Efforts have been made to distinguish event-related brain potentials (ERPs) from the spontaneous EEG signal to assess only ERPs
- Aesthetic processing is determined by a host of factors: biological and human brain evolution, fashions, culture, subcultures, socially constituted conventions, and many other factors.
- mental architecture of aesthetics.



Why is this source important to the project?

- In order to proceed with our task, we need to understand the science of aesthetic

Problem Context Area:

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source: Arduino Uno Data Sheet farnell.com

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

Arduino Uno is a microcontroller board that can control PWM output

It has a low price and it is easy to customize to fit the project requirements

Arduino can run a web server allowing it to receive information in real time and change the display based on incoming data

Arduino can receive direct input from sensors

It uses a simple programming language that is easy to learn

Arduino is better that Raspberry pi for this specific project

Why is this source important to the project?

This information is relevant to the project as Arduino is the simplest and cheapest way to control servo motors. If we decide that we want to make a moving display, we are likely to use an Arduino board to control the motors moving the parts. The board has 6 PWM output pins and 6 analogue input pins. This means that we can connect up to 6 servos and sensors to one board. The analogue input pins allow us to use sensors to record data and change the display using sensors. In addition to this, if we decide to change the display as it receives data in real time, Arduino allows us to run a web server and based on the data we can change the display. Overall the Arduino is a must if we decide to create a moving display, unless it moves solely on physical components and it doesn't require any motors. In

comparison to some other boards like Raspberry pi, Arduino can handle less inputs and outputs and it has less storage, however Raspberry pi is more expensive and complicated to use. For our project, Arduino will be able to handle everything that we need. Arduino Uno cost around 20$ alone but we would need to purchase additional electrical components like servo motors and sensors.

Problem Context Area:

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source: Overview of Difference Between Servo and Stepper Motor Technology

Download_PDF

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

Stepper motors tend to be cheaper but allow lower torque options

Servo motors generate high torque with a limited range of motion

Stepper motors can rotate 360° but can also have precise rotation

Both motors have their pros and cons, however one will be better than the other to do a specific task

Why is this source important to the project?

Servo and stepper motors are the most popular rotary actuators and can be used in many different ways to create a dynamic display. It is hard to decide which motor is better without knowing the specific tasks that we want to accomplish with the motor. If we would like a part to have a specific small motion, a servo motor might be ideal. If we want a part to rotate or have a continuous movement, a stepper motor would be needed to accomplish that task.

Problem Context Area:

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

https://www.designcouncil.org.uk/sites/default/files/asset/document/the-principles-of-inclusive-design.pdf

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

- Inclusive design must be considered at the outset of the design process, and remain integral throughout.
- This will help deliver an environment in which everyone can access and benefit.
- The principles relate as much to the design process as to the final product and equally to management, operation and information.
- Users and other potential consumers should be involved during all parts of the process from the planning phase, through detailed design.
- Convenient so everyone can use them without too much effort or separation.
- Accommodating for all people, regardless of their age, gender, mobility, ethnicity or circumstances.
- Welcoming with no disabling barriers that might exclude some people.
- Realistic offering more than one solution to help balance everyone's needs and recognising that one solution may not work for all.

Why is this source important to the project?

The users of the OEDK and it's visitors are a diverse group. We want everyone regardless of race, color, religion, gender, gender identity, sexual orientation, genteic disability, age, or veteran status to be able to understand and learn more about the OEDK with our design.

Problem Context Area (circle one that best fits):

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

https://stephanieevergreen.com/workshops/ => https://youtu.be/9okGu9Kw6yI

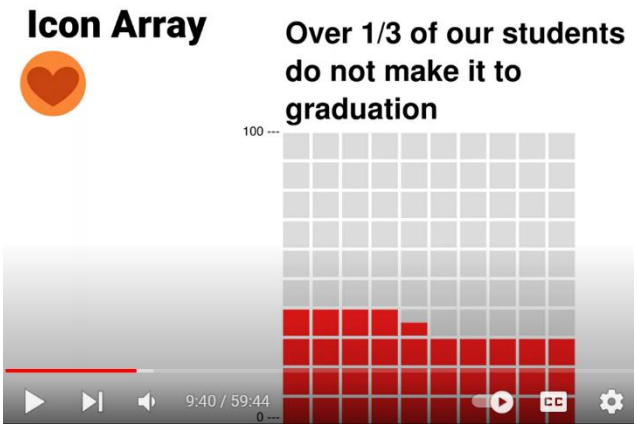
Sage Publishing posted this webinar of Dr. Stephanie Evergreen walks participants through charts she doesn't love and why they don't work and presents alternative charts that she does love and explains why they are more effective.

Is this source peer-reviewed (circle one): Yes No

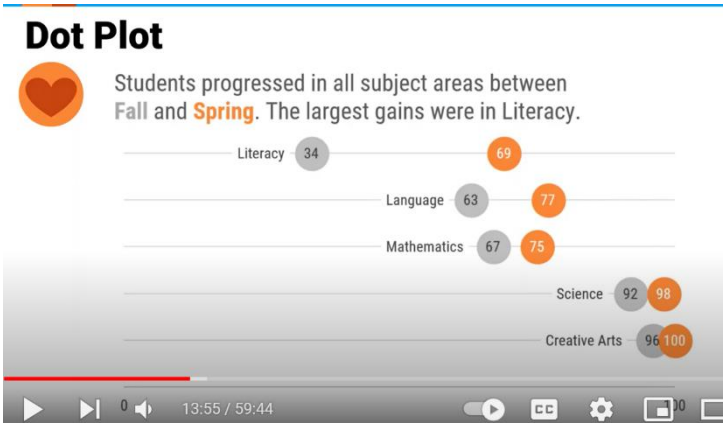
This was a worship talking about her book which was a published, peer reviewed book. Sage Publishing posted this webinar of Dr. Stephanie Evergreen walks participants through charts she doesn't love and why they don't work and presents alternative charts that she does love and explains why they are more effective.

Summary of Key Points:

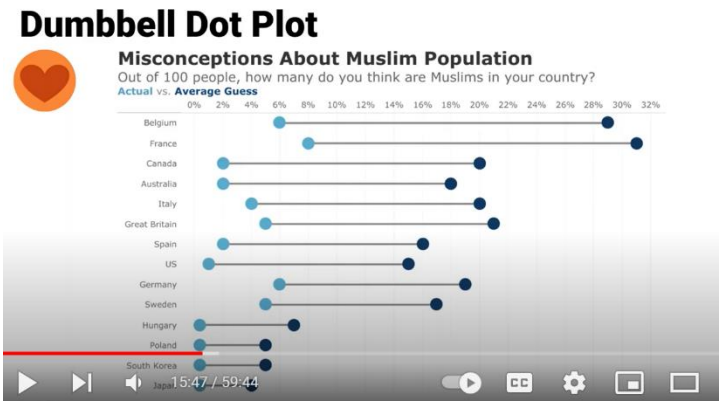
- Because our brains are limited to how much information we can process at once the research that we have on working memory tells us that people can really only juggle 3 to 5 chunks of information at one time and when we just show a bunch of random numbers it's a lot of chunks and that makes it really hard for anybody to retain it anything
- People who saw only one big number remembered it better than like a pie chart or bar graph of the percentage, donut around number would give a quicker mental process

- 

Icon Array helped patents whose literacy skills were lower find and answer. This was a medical case study published in a medical journal.
- Stay away from cheesy icons
- Stick figure dudes donate more money, create a connection, and conduct an international development fundraising context. But they are male and they are able-bodied, so make stick to abstract shapes, people read a lot into things and try to find meaning
- Research shows people are good at judging length but only good at comparing things that are right next to each other
- Position is the easiest way for people to interpret charts



- But excel nerds do not normal like this because it is not a usually type of chart
- If you are trying to tell a story about gap or distance or disparity:

- 

- 

Lollipop Chart is good to get engagement and make the pop of something of interest.
- People have hard time reading area and area differences
- But the area that is the most in the upper left. We live in a culture that reads left to right. Up to down.
- Tree maps are not easy to read with a lot of categories.
- Journey Map- used in Smithosian visitor studies for a museum

- 

Spectrometer chart is good because eliminates that left to right means progress and allows to show the middle of importance
- Need words to tie it into one big picture
- Read left to right something in top left to catch your eye and something on the bottom left to leave you with.

Why is this source important to the project?

This gives great advice on engagement and audience retention features that purposely will include and remember to not to include.

Some specific advantages:

-Large focal number that is static people remember

-Specific types of graphs that are eye-catching and case studied proven to be remembered

-Not to use any person stick figure because though it is a connection point not everyone is represented only able-bodied "male" stereotype picture is shown

-People mostly our users will read left to right have something significant on top left and leave something for the user to leave with like a QR code bottom right

Problem Context Area (circle one that best fits):

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

https://www.researchgate.net/publication/282525843_Light_Art_Perception_and_Sensation

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

- Skyspace, created by James Turrell isolates the qualities of daylight and focuses attention on the impact of the sky's light on the landscape.
- Carlos Cruz-Diez's Chromosaturation highlights the ocular perception and emotional experience of colour, while Olafur Eliasson's Model for a Timeless Garden highlights the temporality of visual perception as well as the persistence of notions about the sublime to appreciation of landscape.
- Tino Seghal's This Variation investigates the impact of darkness on the perception of space and its potential for fostering conviviality and sociality

Why is this source important to the project?

Shows the importance of light and how its incorporation can be eye-catching and important connection to the human experience

Problem Context Area (circle one that best fits):

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

https://www.bostoncyberarts.org/data-flow-an-exhibition-of-algorithmic-art

https://www.lozano-hemmer.com/projects.php

https://www.youtube.com/watch?v=I-EIVIHvHRM

http://dataphys.org/list/*

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

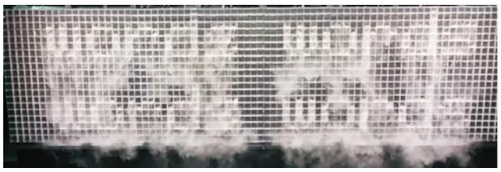
Nathalie Miebach

Mark J. Stock

Nervous System

Cloud Display

"Cloud Display" is a vertical water fountain consisting of 1,600 ultrasonic atomizers, controlled by a machine-learning voice recognition system. When a participant speaks into an intercom, the piece writes any words or sentences spoken using wisps of pure water vapour. The words appear and disappear slowly, forming an evocative and temporary display of language. When no one is participating, from time to time, the piece becomes a waterfall of vapour. The piece was premiered at the "Atmospheric Memory" exhibition performance at the Manchester International Festival in 2019 and is part of a series of water writing installations started with "Call on Water" in 2016. The project can work in most languages, and recognizes different accents.



5500 BC – Mesopotamian Clay Tokens

The earliest data visualizations were likely physical: built by arranging stones or pebbles, and later, clay tokens. According to an eminent archaeologist (Schmandt-Besserat, 1999): "Whereas words consist of immaterial sounds, the tokens were concrete, solid, tangible artifacts, which could be handled, arranged and rearranged at will. For instance, the tokens could be ordered in special columns according to types of merchandise, entries and expenditures; donors or recipients. The token [...]

Added by: **Pierre Dragicevic**. Category: Passive physical visualization Tags: anthropology, archaeology, clay, mesopotamians, rearrangeable

---

2600 BC – Quipus

Quipus were complex assemblies of knotted ropes that were used in South America as a data storage device and played an important role in the Inca administration. Only a handful of specialists could use and decipher them. Their meaning mostly remains a mystery but it seems that color, relative position of knots, knot types and rope length were used to encode categorical and quantitative variables. The oldest known Quipu is 4600 years old. In the late 16th century quipus were still being used [...]

Added by: **Pierre Dragicevic**. Category: Passive physical visualization Tags: anthropology, incas, knots, peru, quipus

---

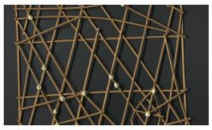
500 BC – Pebble Voting

The earliest participatory visualizations were probably voting systems. Voting in Greece was introduced in the 5th century BC. Adult male citizens were invited to express their opinion by dropping a pebble in an urn: a white pebble meant "yes" and a black pebble meant "no". Sometimes two urns were used. The left image is a detail of a Greek wine cup from the 5th century BC, and is one of the earliest known depictions of the act of voting. The middle image is a modern reconstruction from a [...]

Added by: **Pierre Dragicevic**. Category: Passive physical visualization Tags: archaeology, democracy, greeks, participatory, voting

---

1862 – Marshall Islands Stick Charts

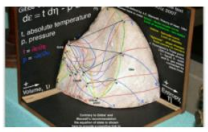


These physical visualizations show ocean swell patterns, and were built by native Micronesians from the Marshall Islands to facilitate canoe navigation. They were memorized before trips. The Western world remained unaware of the existence of these artifacts until 1862. The photo above is a stick chart from 1974. Straight sticks represent regular currents and waves, curved sticks represent ocean swells, and seashells represent atolls and islands. Sources: Wikipedia. Marshall Islands Stick [...]

Added by: **Pierre Dragicevic**. Category: Passive physical visualization Tags: anthropology, cartographic, marshall islands, stick chart

---

1871 – Thermodynamic Surfaces



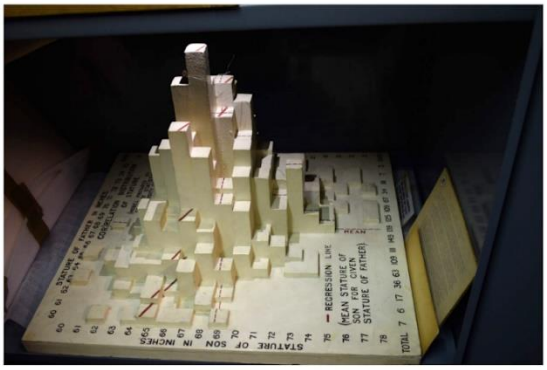
A physical visualization by Scottish physicist James Maxwell (left), constructed over the course of about seven months, from November 1874 to July 1875, based on the descriptions of thermodynamics surfaces described in two 1873 papers by American engineer Willard Gibbs. The molded shape depicts the geometry of the three-dimensional thermodynamic surface of the various states of existence of water: solid, liquid, orgas, shown on Cartesian coordinates of the entropy (x), volume (y), and energy [...]

Added by: **Yvonne Jansen**, sent by: **Fanny Chevalier**. Category: Passive physical visualization Tags: clay, Maxwell, science, thermodynamics, Thompson

---

and Related Artifacts

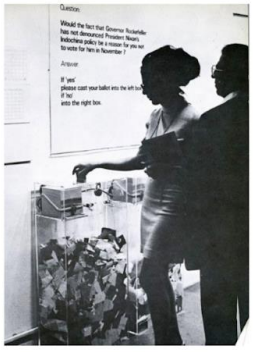
1900 – Pearson and Lee's Height Correlation Chart



The physical model on the left is a bivariate histogram showing the correlation between the heights of fathers (horizontal axis) and sons ("vertical" axis). This data was famously collected by Karl Pearson and Alice Lee between 1893 and 1898. The physical visualization is thought to have been constructed around this time period or soon after, possibly under the supervision of Pearson. It is kept at the Department of Statistical Science, University College London, founded by Pearson in 1911.

and Related Artifacts

1970 – MoMA Poll: Participatory Bar Chart



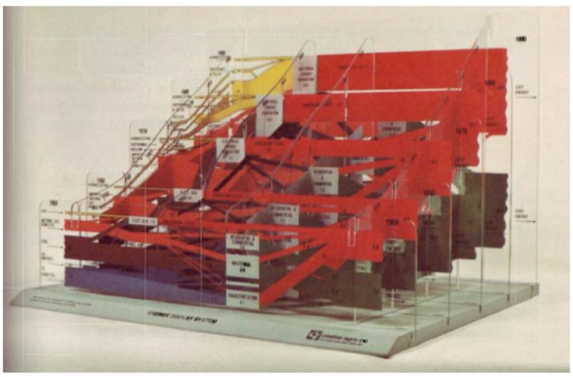
German-American artist Hans Haacke created a participatory physical bar chart as part of a 1970 exhibition at the Museum of Modern Art (MoMA). The audience expressed his opinion on the question *"Would the fact that Governor Rockefeller has not denounced President Nixon's Indochina Policy be a reason for your not voting for him in November?"*. The left plexiglass box collected *"Yes"* answers, while the right box collected *"No"* answers. Rockefeller was running for re-election and was a major donor and board member at the museum. He tried to have the piece removed the next day but without success.

For more recent examples of participatory bar charts, see our entry on Lucy Kimbell's inverted participatory bar charts and our entry on Jennifer Payne's physical visual sedimentation.

Sources:

1. Found on Loren Madsen's lecture slides Art as Information - Information as Art.
2. Graham Budgett MoMA Poll - Hans Haacke.
3. Wikipedia article on Hans Haacke.
4. Hans Haacke (2009) Lessons learned.
5. Left photo from Irmgard Emmelhainz, right photo from language.cont3xt.net.

1970 – 3D Sankey Diagram



This physical 3D Sankey diagram shows complex energy flows and was created in the 70s by the Center for Strategic & International Studies (CSIS). Little information is available about it. It seems to be composed of five layers of transparent sheet, with four additional layers running perpendicularly. Physical size unknown.

Source: Energy Education References Wiki. Energy Flow Diagrams 1949-2009.

Why is this source important to the project?

- Inspiration for our data display
- Shows how fluid art can display data
- Shows historical data visualizations that have been the foundation for many data displays we can incorporate into our design

Problem Area Context: Examples of Rube Goldberg Machines

Name of Source (eg. The New York Times, or, Handbook of Materials):

What is a Rube Goldberg Machine?

Location of Source (include enough information that someone else could find it):

https://wonderopolis.org/wonder/what-is-a-rube-goldberg-machine

Is this source peer-reviewed (circle one): **Yes** No

Problem Context Area (circle one that fits the best):

- Exploration of existing solutions
- Governs the problem
- Background
- Business perspective

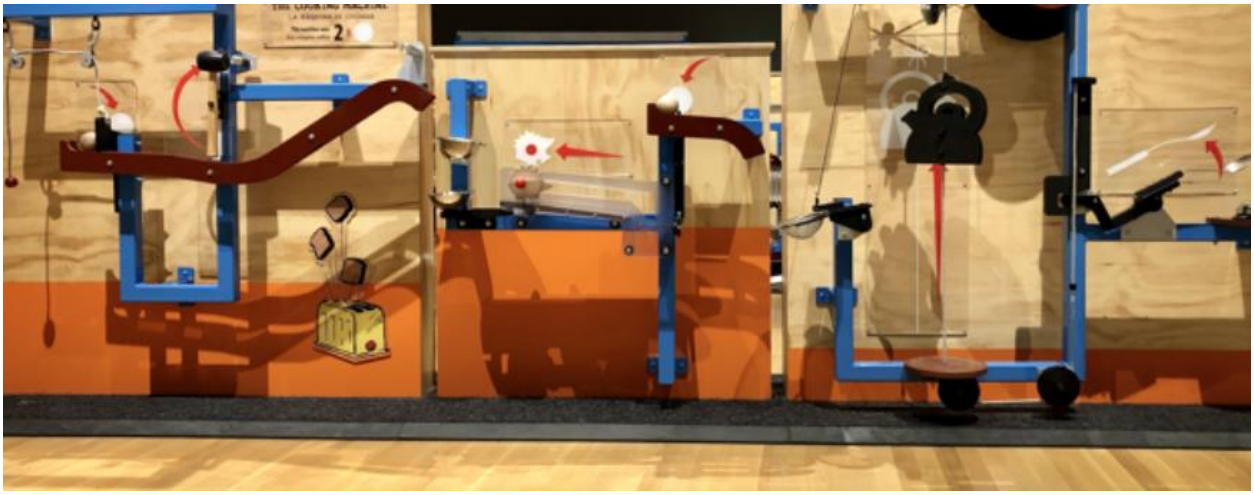
Summary of key points from source (this summary should be thorough enough that you do not need to return to the source when writing your DCR or continuing the project):

A RGM is a contraption that uses a chain reaction to carry out a simple task.

For example, in order to turn the light bulb on…

1. the punching glove must first knocks over the bowling ball
2. the bowling ball will push the bowling pin
3. the bowling pin will open the bird cage
4. the bird will and cage will push the pool ball down the “slide”
5. etc…
6. and the light bulb turns on.

RGM can be as simple as these…



Or as complicated as this:

World's Largest RGM lights up Christmas Tree:

https://www.youtube.com/watch?v=RBOqfLVCDv8

RGM with electrical stuff:

https://www.youtube.com/watch?v=LQvG8bK-ASk

https://www.youtube.com/watch?v=G_8EIAGERzE

https://www.youtube.com/watch?v=YFSojjrj3og

Common techniques:

- Pushing
    - ex. a ball rolls into a standing object and pushes the object down
- Pulling down
    - ex. a heavy object pulls down lever
- Dominoes
- Slide
    - ex. ball reaches from point A to a lower point B by sliding down

Common Materials:

- PVC pipe
- aluminum foil, cardboard box, plastic containers, typical household stuff
- water bottles
- dominoes
- wood
- electronics
- string
- balls

Problem Area Context: What ensures engagement?

Name of Source (eg. The New York Times, or, Handbook of Materials):

Designing for Engagement (for digital)

10 Tips for Museum Exhibit Design Success

Location of Source (include enough information that someone else could find it):

https://www.canva.com/learn/design-for-engagement/

https://colorcraft3d.com/blog-post/10-tips-for-museum-exhibit-design-success/

Is this source peer-reviewed (circle one): Yes No

Problem Context Area (circle one that fits the best):

- Exploration of existing solutions
- Governs the problem
- Background
- Business perspective

Summary of key points from source (this summary should be thorough enough that you do not need to return to the source when writing your DCR or continuing the project):
Canva:

- Affect user's mood with color
    - diff colors warrant different moods (see https://www.verywellmind.com/color-psychology-2795824 for more)
- Sufficient space and good composition can create a positive impression for the viewers

- Display must be easy to navigate. A good composition will allow the users to feel comfortable and encourage them to spend more time with the display
- On the other hand, a messy or confusing display will make the users decide not to interact or engage with the display at all
    - Maximize user threshold
- Some ways to do this are:
    - space things out
    - be consistent with fonts and alignment
    - intuitive hierarchy

Museum Exhibit

- Tell a story
    - Immerse the visitor with the information being provided through storytelling
        - Note: so maybe we can incorporate historical data??
    - Include the stories within the story
        - tell specific stories about the pieces that fit within the larger framework of the display
- Create a sense of time and place in the display
    - don't just throw data and info out there, make sure that you use a linear or chronological flow such as a timeline to make the display completing
    - this can be achieved through graphics, labels, signage, sounds, and interactive tech such as visual cues
- Graphic Design is vital

    - Graphic design is essential in bringing the display to life. This can be anything from signs, labels, banners, and set pieces
    - on a basic level, graphic design can help organize the flow of information traffic
    - it also immerses the user with the display
- gamification
    - try including the gaming industry's techniques to create a fun and interactive experience
    - some examples that can are low-tech are:
        - treasure hunt such as finding hidden items in the visuals and graphics
        - easter eggs
- embrace technology
    - anything such as video monitors or audio and mobile apps
- divide into sections
    - if there is a lot of info, make sure that the info is categorized in order to make the info and data more digestible and get a sense of completion as they move from section to section
- use consumer-centric marketing
    - ex: marketing, press
        - once done, we can talk to thresher or put it up flyers stating that the OEDK has a display in the front desk or smthg
- Have a roadmap of your display in terms of info
    - What is the purpose of our display? How do we go about doing it?

PROJECT WORK: Multi-perspective Art

Name of Source (eg. The New York Times, or, Handbook of Materials):

Multi-Perspective Art by Christopher Derek Bruno

Display Blocks

Location of Source (include enough information that someone else could find it):

https://theinspirationgrid.com/multi-perspective-art-by-christopher-derek-bruno/

https://dspace.mit.edu/bitstream/handle/1721.1/79883/Maes_Display%20Blocks%20Cubic.pdf?sequence=1&isAllowed=y

Is this source peer-reviewed (circle one): Yes No

Problem Context Area (circle one that fits the best):

- Exploration of existing solutions
- Governs the problem
- Background
- Business perspective

Summary of key points from source (this summary should be thorough enough that you do not need to return to the source when writing your DCR or continuing the project):

Multi-perspective art plays with shifts in perspectives such that different angles will result is different visual forms of the art

Display Blocks: The design consists of six light emitting displays arranged in a cube in order to allow coordinated visuals across the cube. As a result, the cube is able to display different forms of information by simple manipulation of the cube (aka different perspectives). Essentially, this device has multiple interpretations of a piece of data by mapping them to different faces of a volumetric display

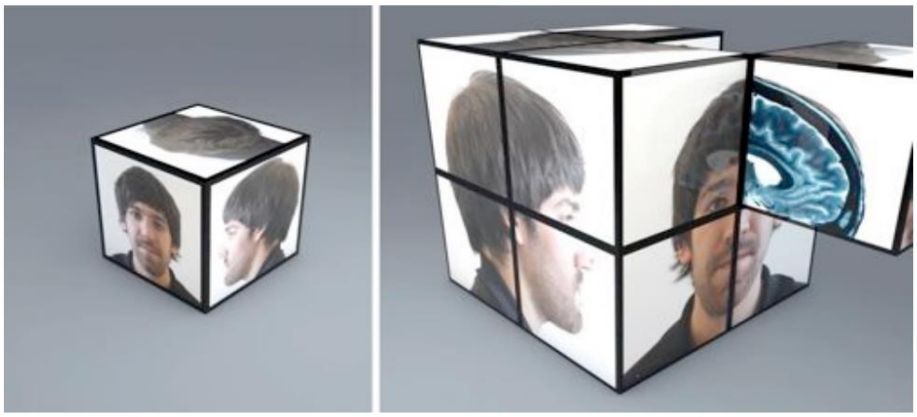


Figure 4. Example of orthographic visualization application. In the first picture we can see how one Display Block would visualize a head, in the second we can see how multiple devices will visualize the same head. In the case of having multiple blocks, they could be explored to reveal hidden content.

Problem Context Area (circle one that best fits):

Exploration of existing solutions | Governs the problem | Background | Business perspective

Complete Citation/Source:

Choi, G.-H., Nakamura, H., & Kobayashi, H. (1996). Calibration of Servo System with redundant actuators. *IFAC Proceedings Volumes*, *29*(1), 678–683. https://doi.org/10.1016/s1474-6670(17)57739-3

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

The paper proposes an experimental system that demonstrates high precision and high robustness against disturbances in spite of heavy gear backlash, coulomb friction, and sensor dead-zone.

Why is this source important to the project?

This is important because we keep in mind that there are solutions to servos being inaccurate, so when we are deciding on what materials or components to use in our design, we will be more well informed about our options.

Problem Context Area (circle one that best fits):

Exploration of existing solutions Governs the problem Background Business perspective

Complete Citation/Source:

Lee, B., Choe, E. K., Isenberg, P., Marriott, K., Stasko, J., & Rhyne, T.-M. (2020). Reaching broader audiences with data visualization. *IEEE Computer Graphics and Applications*, *40*(2), 82–90. https://doi.org/10.1109/mcg.2020.2968244

Is this source peer-reviewed (circle one): Yes No

Summary of Key Points:

The authors outlined four research topics—personal data visualization, data visualization on mobile devices, inclusive data visualization, and multimodal interaction for data visualization: which would help us reach broader audiences and increase the practical impact of data visualization.

Why is this source important to the project?

This is important because it provides insight to how an audience might receive a data display and which types of displays are more effective at both being visually pleasing, accurate, and easy to interpret by a broad audience.

APPENDIX C: USER-DEFINED SCALES

[1] Intuitive to Understand

5	**Can be fully understood at a glance**
4	**Can be understood without too much interpretation**
3	Requires a significant time to process and understand the presented data
2	If another person explains the data, the user *can* understand the display
1	If another person explains the data, the user still *cannot* understand the display

[2] Visual Appeal

5	**Display is memorable and eye-catching to users and improves the aesthetics of the OEDK**
4	**Display is enjoyable to look at but not eye-catching or memorable**
3	Display does not necessarily improve the overall aesthetic of the OEDK and is not noticeable
2	Display causes headaches and strain on eyes and is unpleasant and ugly
1	Display deters visitors

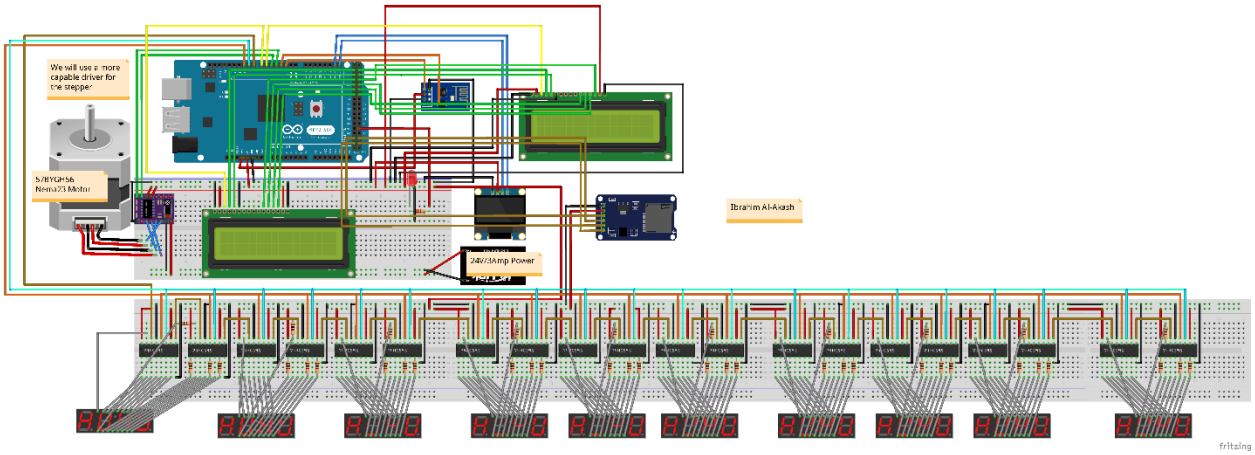
[3] Engaging

5	**User enjoys time interacting with display and prompts users to tell others about the display**
4	**User enjoys time interacting with display but does not feel compelled to tell others about it**
3	**User feels compelled to interact with it, but display is forgettable**
2	User does not feel compelled to interact with it, and display is forgettable
1	User is frustrated when interacting with display

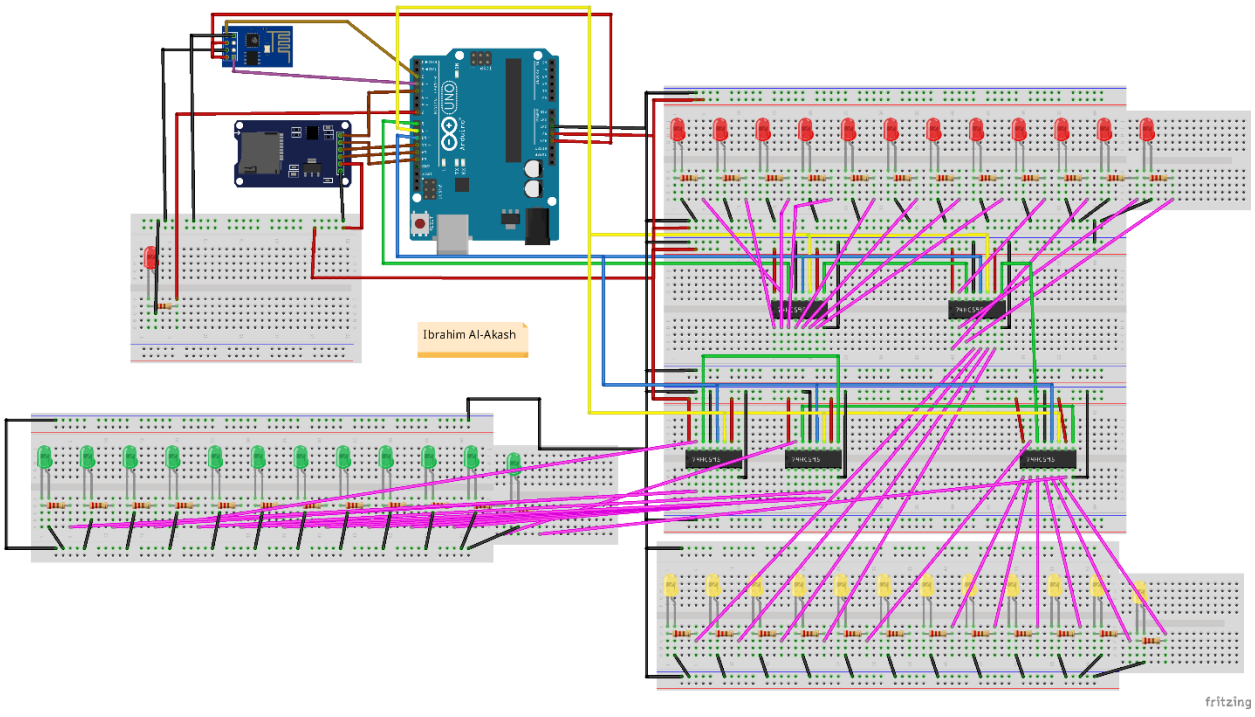
*Highlighted sections indicate target values for UDS.

APPENDIX D: CIRCUIT DIAGRAMS

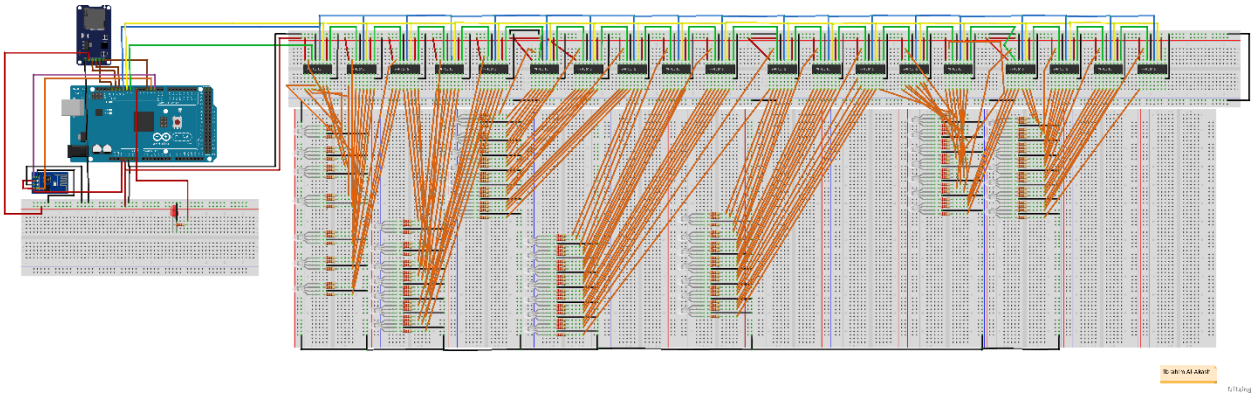
Rocket



Clock



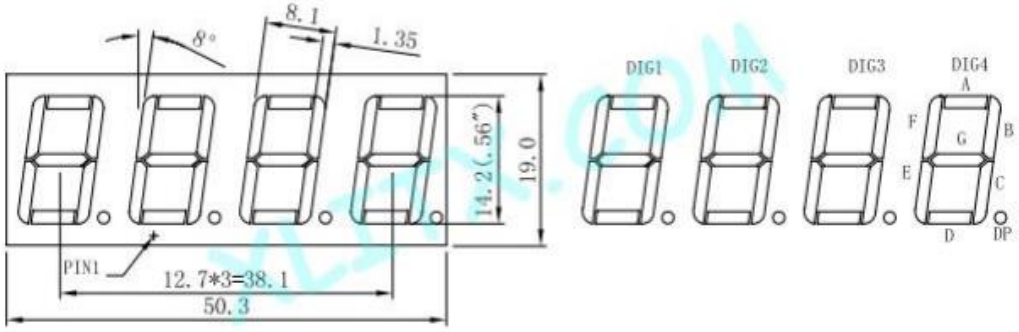
LED Matrix

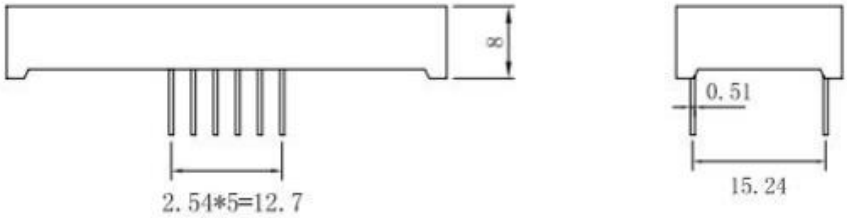


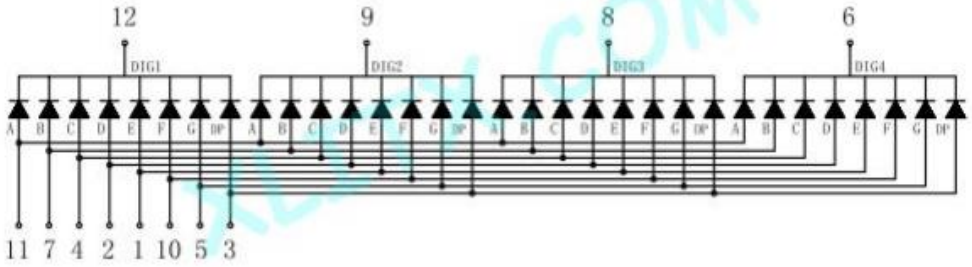
APPENDIX E: HARDWARE DATASHEETS

4-Digit 7-Segment Display

Model :	5641AS
Size :	0.56-inch
Emitting color :	Red (Ultra-Bright)
Mode :	Common-Cathode (CC)
Digit :	4-Digit
Category :	LED 7-Segment Display
Maker :	XLITX Technology


8.1
1.35
8°
14.2(.56")
19.0
PIN1
12.7*3=38.1
50.3
DIG1
DIG2
DIG3
DIG4
A
B
C
D
E
F
G
DP


8
2.54*5=12.7
0.51
15.24


12
9
8
6
DIG1
DIG2
DIG3
DIG4
A B C D E F G DP
11 7 4 2 1 10 5 3

1. Electro-Optical Characteristics(Ta=25℃)

PARAMETER	SYMBOL	DEVICES (ULTRA-BRIGHT RED)		UNIT	TEST CONDIONS
		TYP	MAX		
Peak Emission Wavelengrth	λp	640		nm	IF=10mA
Forward Voltage	VF	1.8		V	IF=10mA
Reverse Current	IR		50	μA	VR=5V
Segment To Segment (Dot To Dot) Luminonous Intensity Ratio	IV-M	1.5:1			IF=20

2. Absolute Maximun Ratings(Ta=25℃)

PARAMETER	SYMBOL	DEVICES (ULTRA-BRIGHT RED)	UNIT
Power Dissipation Per Dice	pad	100	mw
Derating Linear From 25℃ Per Dice		0.5	mA/℃
Continuous Forward Current Per Dice	laf	30	mA
Peak Forward Ourrent Per Dice (Duty Cycle 1/10,10KHz)	lpf	200	mA
Reverse Voltage Per Dice	Vr	5	V
Operating Temperature	Topr	-20℃ (to) +75℃	
Storage Temperature	Tstg	-20℃ (to) +85℃	
Solder Temperature		1.6 Inch Below Seating Place for 5 seconds at 230℃	

5mm Monochrome LED

Model No.: YSL-R531Y3D-D2

Applications:

- Decorations
- Inceticidal Lights
- Bill Insperctor
- Medical Appliance

Absolute Maximum Ratings: (Ta=25℃).

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering Temperature	T_{SOL}	Max. 260℃ for 3 Sec. Max. (3mm from the base of the expoxy bulb)	

Absolute Maximum Ratings: (Ta=25℃)

ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	I_F=20mA	1.8	---	2.2	V
Wavelenength (nm) or TC(k)	$\Delta\lambda$	I_F=20mA	587	---	591	nm
*Luminous Intensity	I_V	I_F=20mA	150	---	200	mcd
50% Viewing Angle	$2\theta 1/2$	I_F=20mA	40	---	60	deg

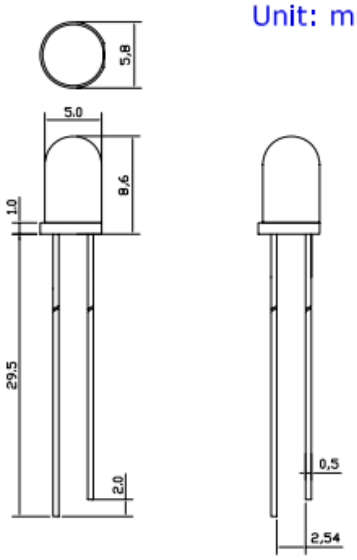
Light Degradation in mcd: (I_F=20mA)

Hours / Colors	Light Degradation in mcd after Different Hours					
	216 Hrs	360 Hrs	792 Hrs	1104 Hrs	1992 Hrs	2328 Hrs
Red	1.52%	-1.22%	-3.10%	-4.68%	-5.72%	-8.27%
Yellow	-1.71%	-2.97%	-5.93%	-8.13%	-8.90%	-11.10%
Blue	3.13%	-0.33%	-3.84%	-8.23%	-21.32%	-24.92%
Green	-8.02%	-9.78%	-14.25%	-17.37%	-20.79%	-22.30%
Hours	48 Hrs	168 Hrs	336 Hrs	360Hrs	720 Hrs	1008 Hrs
Cool White	10.56%	6.72%	-2.29%	-7.68%	-17.32%	-22.48%
Pure White	13.66%	8.22%	-1.45%	-8.50%	-19.52%	-25.26%
Warm White	3.02%	-4.38%	-15.18%	-21.15%	-27.19%	-29.97%

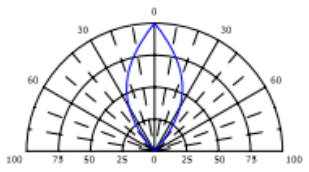
Mechanical Dimensions:

- All dimension are in mm, tolerance is ±0.2mm unless otherwise noted
- An epoxy meniscus may extend about 1.5mm down the leads.
- Burr around bottom of epoxy may be 0.5mm Maximum

Unit: mm



Viewing Angle Drawing



5mm RGB LED

Model No.: YSL-R596AR3G4B5C-C10
RED/GREEN/BLUE Triple Color LED

Applications:

- Moving Message Display
- Full Color Display
- Banking Board
- Score Boards
- Digital Display

LED Chip Absolute Maximum Ratings: (Ta=25℃)

Pararmeter	Symbol	Red	Green	Blue	Unit
Forward current	I_F	20	20	20	mA
Peak forward current(Duty Cycle=1/10,1OKHz)	I_{PF}	30	30	30	mA
Reverse current (V_R=5V)	I_R	10	10	10	μ A
Operating temp	T_{OPR}	-25~ 85	-25~ 85	-25~ 85	℃
Storage temp	T_{STG}	-30~85	-30~85	-30~85	℃
Peak Emission Wavelength	λ PH	625	520	467.5	nm

* Soldering Bath: not more than 5 seconds @260 ℃.The bottom ends of the plastic reflector should be at least 2mm above the solder surface

Soldering Iron: not more than 3 seconds @300 ℃ under 30W

LED Chip Typical Electircal & Optical Characteristics: (Ta=25℃)

ITEMS	Color	Symbol	Condition	Min.	Typ.	Max.	Unit
Forward Voltage	Red	V_F	I_F=20mA	1.8	2.0	2.2	V
	Green			3.0	3.2	3.4	
	Blue			3.0	3.2	3.4	
Luminous Intensity	Red	I_V	I_F=20mA	– – –	– – –	800	mcd
	Green			– – –	– – –	4000	
	Blue			– – –	– – –	900	
Wavelenength	Red	Δ λ	I_F=20mA	620	623	625	nm
	Green			515	517.5	520	
	Blue			465	466	467.5	
Light Degradation after 1000 hours	Red	-4.68% ~ -8.27%					
	Green	-11.37% ~ -15.30%					
	Blue	-8.23% ~ -16.81%					

57BYGH56-401A NEMA 23 Stepper Motor

Description

Parameters:

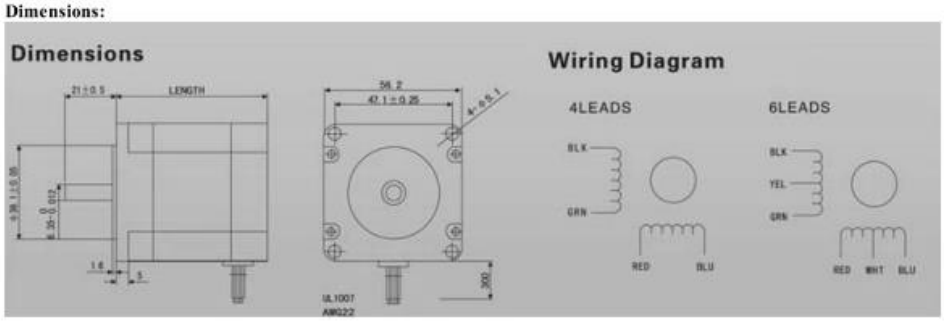
- Step Angle Accuracy: +-5%(full step, no load)
- Resistance Accuracy: +-10%- Inductance Accuracy: +-20%
- Temperature Rise: 80 Degrees Celsius CMax.(ratod current,2 phase on)- Ambient Temperature: -10°C-+50°C
- Insulation Class: B
- Dielectric Strength: 500V AC for one minute
- Shaft Radial Play: 0.06Max.(450g-load)
- Shaft Axial Play: 0.08Max.(450 9-load)

Specifications:

Model Single Shaft	Step angle °	Motor length L(mm)	Rated current A	Phase resistance Ω	Phase inductance mH	Holding torque Oz-in	Kgf.cm	Lead wire No.	Rotor inertia g.cm2	Weight kg
57BYGH56-401A	1.8	56	2.8	0.9	2.5	175	12.6	4	300	0.7

We also manufacture products according to customer's requirements. If you have any other need, please contact us.

Dimensions:



Package included:

- 1 x 57BYGH56-401A stepper motor

CD74HC595 8-Bit Shift Registers

CD74HC595 8-Bit Shift Registers With 3-State Output Registers

1 Features

- 8-Bit serial-in, parallel-out shift
- Wide operating voltage range of 2 V to 6 V
- High-current 3-state outputs can drive up to 15 LSTTL loads
- Low power consumption, 80-µA max I_{CC}
- Typical t_{PD} = 14 ns
- ±6-mA output drive at 5 V
- Low input current of 1 µA max
- Shift register has direct clear

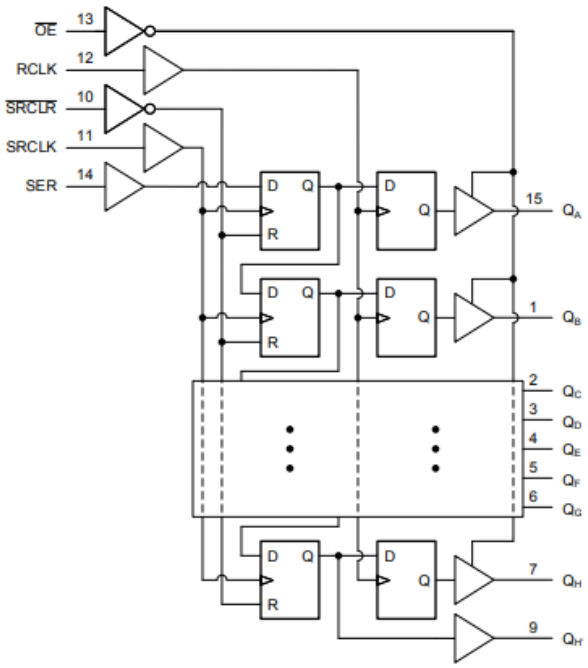
2 Description

The CD74HC595 is an 8-bit serial-input parallel-output shift register with output registers and 3-state outputs.

Device Information

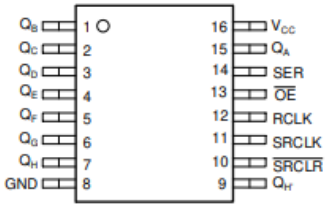
PART NUMBER	PACKAGE(1)	BODY SIZE (NOM)
CD74HC595E	PDIP (16)	19.31 mm × 6.35 mm
CD74HC595DW	SOIC-DW (16)	10.30 mm × 7.50 mm
CD74HC595M	SOIC-D (16)	9.90 mm × 3.90 mm
CD74HC595NS	SO (16)	10.20 mm × 5.30 mm
CD74HC595SM	SSOP (16)	6.20 mm × 5.30 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.



Functional Block Diagram

4 Pin Configuration and Functions



D, DW, N, NS, or DB Package
16-Pin SOIC, PDIP, SO, or SSOP
Top View

SER - Data Pin - DS
RCLK - Latch Pin - STCP
SRCLK - Clock - SHCP
SRCLR - Master Reset - MR
QH' - Data Out - used in daisy-chaining,
goes to DS of next shift register

5 Specifications

5.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

			MIN	MAX	UNIT
V_{CC}	Supply voltage		-0.5	7	V
I_{IK}	Input clamp current(2)	For $V_I < 0$ or $V_I > V_{CC}$		±20	mA
I_{OK}	Output clamp current(2)	For $V_O < 0$ or $V_O > V_{CC}$		±20	mA
I_O	Continous output current	For -0.5V < V_O = 0 to V_{CC}		±35	mA
	Continuous current through V_{CC} or GND			±70	mA
T_{stg}	Storage temperature		-65	150	°C

(1) Stresses beyond those listed under *absolute maximum ratings* may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under *recommended operating conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The input and output voltage ratings may be exceeded if the input and output current ratings are observed.

5.2 Recommended Operating Conditions(1)

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2V	1.5			V
		V_{CC} = 4.5V	3.15			
		V_{CC} = 6V	4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2V			0.5	V
		V_{CC} = 4.5V			1.35	
		V_{CC} = 6V			1.8	
V_I	Input voltage		0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	V
t_t (2)	Input transition rise and fall time	V_{CC} = 2V			1000	ns
		V_{CC} = 4.5V			500	
		V_{CC} = 6V			400	
T_A	Operating free-air temperature		-55		125	°C

(1) All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.
(2) If this device is used in the threshold region (from V_{IL}max = 0.5 V to V_{IH}min = 1.5 V), there is a potential to go into the wrong state from induced grounding, causing double clocking. Operating with the inputs at t_t = 1000 ns and V_{CC} = 2 V does not damage the device; however, functionally, the CLK inputs are not ensured while in the shift, count, or toggle operating modes.

5.3 Thermal Information

THERMAL METRIC		N (PDIP) 16 PINS	DW (SOIC) 16 PINS	D (SOIC) 16 PINS	NS (SO) 16 PINS	DB (SSOP) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance(1)	67	57	73	64	82	°C/W

(1) For more information about traditional and new thermal metrics, see the Semiconductor and IC Package Thermal Metrics application report.

5.4 Electrical Characteristics

PARAMETER	TEST CONDITIONS[1]	V_{CC} (V)	25°C			-40°C to 85°C		-55°C to 125°C		UNIT
			MIN	TYP	MAX	MIN	MAX	MIN	MAX	
HC TYPES										
V_{OH}	I_{OH} = – 20 µA	2	1.9	1.998		1.9		1.9		V
		4.5	4.4	4.499		4.4		4.4		
		6	5.9	5.999		5.9		5.9		
	Q_H, I_{OH} = – 4 mA	4.5	3.98	4.3		3.84		3.7		
	Q_A-Q_H, I_{OH} = – 6 mA		3.98	4.3		3.84		3.7		
	Q_H, I_{OH}= – 5.2 mA	6	5.48	5.8		5.34		5.2		
	Q_A-Q_H, I_{OH} = – 57.8 mA		5.48	5.8		5.34		5.2		
V_{OL}	I_{OL} = 20 µA	2		0.002	0.1		0.1		0.1	V
		4.5		0.001	0.1		0.1		0.1	
		6		0.001	0.1		0.1		0.1	
	Q_H, I_{OL} = 4 mA	4.5		0.17	0.26		0.33		0.4	V
	Q_A-Q_H, I_{OL} = 6 mA			0.17	0.26		0.33		0.4	V
	Q_H, I_{OL}= 5.2 mA	6		0.15	0.26		0.33		0.4	V
	Q_A-Q_H, I_{OL} = 7.8 mA			0.15	0.26		0.33		0.4	V
I_I	V_I = V_{CC} or 0	6			±0.1	±100		±1000		±1000	nA
I_{OZ}	V_O = V_{CC} or 0, QA-Q_H	6			±0.01	±0.5		±5		±10	µA
I_{CC}	V_I = V_{CC} or 0, I_O = 0	6				8		80		160	µA
C_i		2 to 6		3	10		10		10	pF

(1) V_I = V_{IH} or V_{IL}

DM556T Stepper Motor Driver

2.1 Electrical Specifications

Parameters	DM556T			
	Min	Typical	Max	Unit
Output Current	1.8	-	5.6(4.0 RMS)	A
Supply Voltage	20	24 - 48	50	VDC
Logic signal current	7	10	16	mA
Pulse input frequency	0	-	200	kHz
Minimal pulse width	2.5	-	-	µS
Minimal direction setup	5.0	-	-	µS
Isolation resistance	500			MΩ

2.2 Environment

Cooling	Natural Cooling or Forced cooling	
Operating Environment	Environment	Avoid dust, oil fog and corrosive gases
	Ambient Temperature	0°C — 65°C (32°F - 149°F)
	Humidity	40%RH—90%RH
	Operating Temperature	0°C — 50°C (32°F - 122°F)
	Vibration	10-50Hz / 0.15mm
Storage Temperature	-20°C — 65°C (-4°F - 149°F)	
Weight	Approx. 300g (10.6oz)	

2.3 Mechanical Specifications

(unit: mm [1inch=25.4mm])

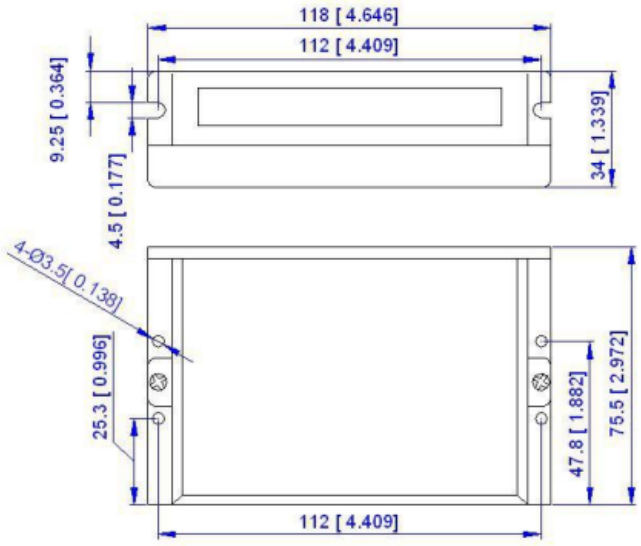


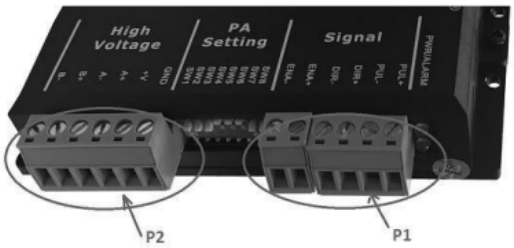
Figure 1: Mechanical specifications

*** Side mounting recommended for better heat dissipation**

2.4 Elimination of Heat

- DM556T reliable working temperature should be < 60℃ (140°F)
- It is recommended to use automatic idle-current mode to reduce motor heating. That means set the SW4 pin of DIP switch at "OFF" position.
- It is recommended to mount the drive vertically to maximize heat sink area. Use forced cooling method to cool if necessary.

3. Connection Pin Assignments and LED Indication



The DM556T has two connector blocks P1&P2 (see above picture). P1 is for control signals connections, and P2 is for power and motor connections. The following tables are brief descriptions of the two connectors. More detailed descriptions of the pins and related issues are presented in section 4, 5, 9.

3.1 Connector P1 Configurations

Pin Function	Details
PUL+	Pulse signal: Pulse active at rising edge; 4-5V when PUL-HIGH, 0-0.5V when PUL-LOW. Minimal pulse width of 2.5μs. Add a resistor for current-limiting at +12V or +24V input logic voltage (1K for +12V, 2k for +24V). The same as DIR and ENA signals.
PUL-	
DIR+	DIR signal: This signal has low/high voltage levels to represent two directions of motor rotation.. Minimal direction setup time of 5μs. Also swapping the connection of two wires of a coil (e.g. A+ and A-) to the drive will reverse motor direction.
DIR-	
ENA+	Enable signal: This signal is used for enabling/disabling the drive. High level +5V (NPN control signal) for enabling the drive and low level for disabling the drive. PNP and Differential control signals are on the contrary, namely Low level for enabling. By default it is left **UNCONNECTED (ENABLED)**.
ENA-	

Notes: (1) shielding control signal wires is suggested; (2) To avoid interference, don't tie PUL/DIR control signal and motor wires together

3.2 Connector P2 Configurations

Pin Function	Details
GND	Power supply ground connection.
+V	Power supply positive connection. Suggest 24-48VDC power supply voltage
A+,A-	Motor Phase A connections. Connect motor A+ wire to A+ Pin; motor A- wire to A-
B+,B-	Motor Phase B connections. Connect motor B+ wire to B+ Pin; motor B- wire to B-

ER-0.96in OLED Display

1.1 Order Number

Series Number	Order Number	Description
ER-OLEDM0.96-1-I2C	ER-OLEDM0.96-1W-I2C	White 0.96" OLED Display Panel with I2C Interface Board
	ER-OLEDM0.96-1B-I2C	Blue 0.96" OLED Display Panel with I2C Interface Board
	ER-OLEDM0.96-1YB-I2C	Yellow/Blue 0.96" OLED Display Panel with I2C Interface Board
ER-OLEDM0.96-1-SPI-I2C	ER-OLEDM0.96-1W-SPI-I2C	White 0.96" OLED Display Panel with I2C,3/4-wire Interface Board
	ER-OLEDM0.96-1B-SPI-I2C	Blue 0.96" OLED Display Panel with I2C,3/4-wire Interface Board
	ER-OLEDM0.96-1YB-SPI-I2C	Yellow/Blue 0.96" OLED Display Panel with I2C,3/4-wire Interface Board

1.2 Image

ER-OLEDM0.96-1W-I2C ↓

ER-OLEDM0.96-1B-I2C ↓

ER-OLEDM0.96-1Y-I2C ↓

ER-OLEDM0.96-1W-SPI-I2C ↓

ER-OLEDM0.96-1B-SPI-I2C ↓

ER-OLEDM0.96-1YB-SPI-I2C ↓

2. SPECIFICATION

2.1 Display Specification

Item	Standard Value	Unit
Display Format	128 x 64 Dots	--
Display Connector	Pin Header	--
Operating Temperature	-40 ~ +85	℃
Storage Temperature	-40 ~ +85	℃
Sunlight Readable	No	--

2.2 Mechanical Specification

Item	Standard Value	Unit
Outline Dimension	27.30(W) x 27.80(H)	mm
Visual Area	23.744(W) x 12.864(H)	mm
Active Area	21.744(W) x 10.864(H)	mm
Dot Size	0.15x0.15	mm
Dot Pitch	0.17x0.17	mm

2.3 Electrical Specification

Item	Standard Value	Unit
IC Package	COG	--
Controller	SSD1306	--
Interface	I2C Interface for ER-OLEDM0.96-1-I2C	--
	I2C,3/4-Wire SPI Interface for ER-OLEDM0.96-1-SPI-I2C	--

2.4 Optical Specification

Item	Standard Value	Unit
Display Type	OLED (Passive Matrix)	--
Viewing Angle Range	Free	degree
OLED Duty	1/64	--

4.1 Pin Configuration for ER-OLEDM0.96-1-I2C

Pin No	Pin Name	Description
1	GND	Ground
2	VCC	Positive voltage supply
3	SCL	Serial clock input with I2 interface.
4	SDA	Serial data input/output with I2C interface.

4.4 Absolute Maximum Ratings

Item	Symbol	Min.	Max.	Unit	Note
Supply Voltage for Logic	VDD	-0.3	+5.5	V	1,2
Operating Temperature	Top	-40	+85	℃	
Storage Temperature	Tstg	-40	+85	℃	3
Life Time(120cd/㎡)	--	10,000	--	hour	4
Life Time(80cd/㎡)	--	30,000	--	hour	4
Life Time(60cd/㎡)	--	50,000	--	hour	4

Note 1: All the above voltages are on the basis of Vss=0V.

Note 2: When this module is used beyond the above absolute maximum ratings, permanent breakage of the module may occur. Also, for normal operations, it is desirable to use this module under the conditions. If this module is used beyond these conditions, malfunctioning of the module can occur and the reliability of the module may deteriorate.

Note 3: The defined temperature ranges do not include the polarizer. The maximum withstood temperature of the polarizer should be 80℃.

Note 4: Ta=25℃., 50% Checkerboard.

End of lifetime is specified as 50% of initial brightness reached. The average operating lifetime at room temperature is estimated by the accelerated operation at high temperature conditions.

4.5 Electrical Characteristics

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Supply Voltage for Logic	VDD	External supply	3.0	3.3	5.0	V
Supply Voltage for Logic IO	VDDIO	Internal supply	3.0	--	3.3	V
High Level Input	VIH	-	0.8xVDDIO	--	VDDIO	V
Low Level Input	VIL	-	0	--	0.2xVDDIO	V
High Level Output	VOH	IOUT=100uA,3.3MHZ	0.9x VDDIO	--	VDDIO	V
Low Level Output	VOL	IOUT=100uA,3.3MHZ	0	--	0.1xVDDIO	V
Operating Current for VDD	IDD	Note 5	--	28.5	33	mA
Sleep Mode Current for VDD	IDD,Sleep	--	--	-	1	mA

Note5: VDD=3.3V,100% Display Area Turn on.

ESP8266 ESP-01 Wi-Fi Module

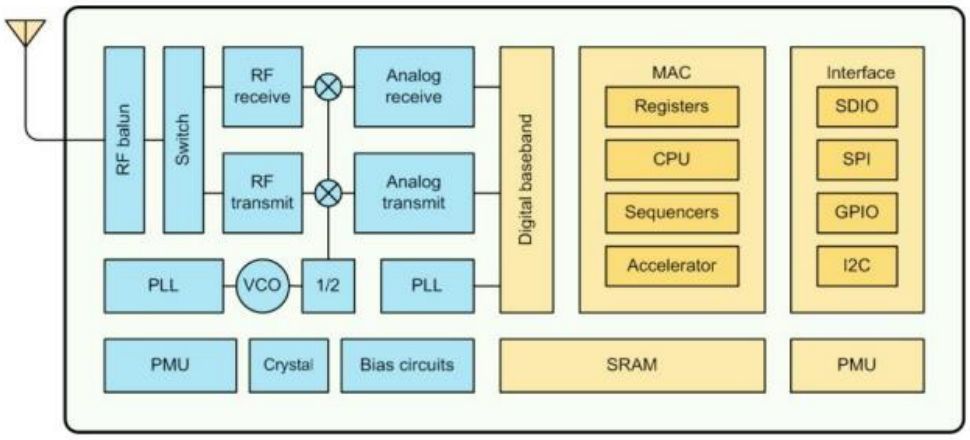


Figure 1 ESP8266EX Block Diagram

Table 1 Parameters

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	14.3mm*24.8mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

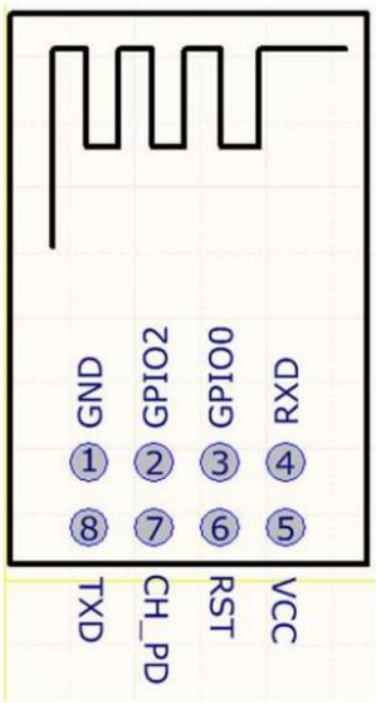


Table 2 Pin Descriptions

NO.	Pin Name	Function
1	GND	GND
2	GPIO2	GPIO,Internal Pull-up
3	GPIO0	GPIO,Internal Pull-up
4	RXD	UART0,data received pin RXD
5	VCC	3.3V power supply (VDD)
6	RST	1) External reset pin, active low 2) Can loft or external MCU ,
7	CH_PD	Chip enable pin. Active high
8	TXD	UART0,data send pin RXD

4.5. Absolute Maximum Ratings

Table 7 Absolute Maximum Ratings

Rating	Condition	Value	Unit
Storage Temperature		-40 to 125	°C
Maximum Soldering Temperature		260	°C
Supply Voltage	IPC/JEDEC J-STD-020	+3.0 to +3.6	V

4.6. Recommended Operating Conditions

Table 8 Recommended Operating Conditions

Operating Condition	Symbol	Min	Typ	Max	Unit
Operating Temperature		-40	20	125	°C
Supply voltage	VDD	3.0	3.3	3.6	V

4.7. Digital Terminal Characteristics

Table 9 Digital Terminal Characteristics

Terminals	Symbol	Min	Typ	Max	Unit
Input logic level low	V_{IL}	-0.3		0.25VDD	V
Input logic level high	V_{IH}	0.75VDD		VDD+0.3	V
Output logic level low	V_{OL}	N		0.1VDD	V
Output logic level high	V_{OH}	0.8VDD		N	V

Note: Test conditions: VDD = 3.3V, Temperature = 20 ℃, if nothing special is stated.

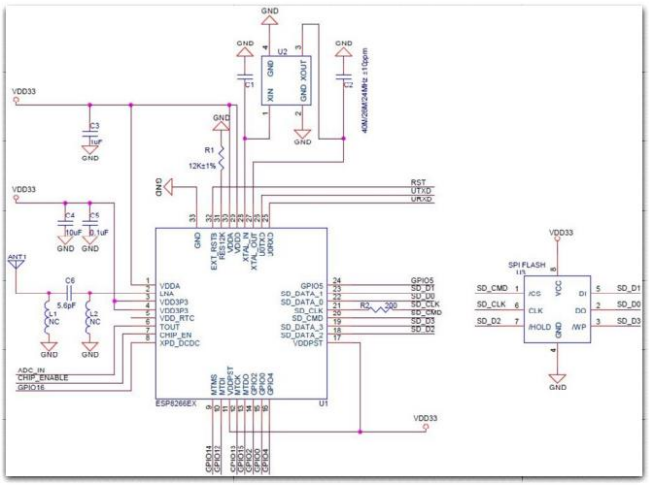


Figure 4 Schematics of Esp-01 WiFi Module

HC-SR04 Ultrasonic Sensor Module

Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

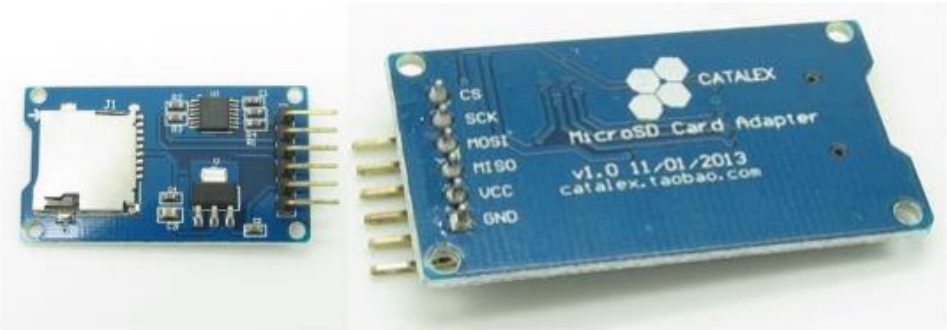
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

SD Card Module

Micro SD Card Micro SDHC Mini TF Card Adapter Reader Module for Arduino



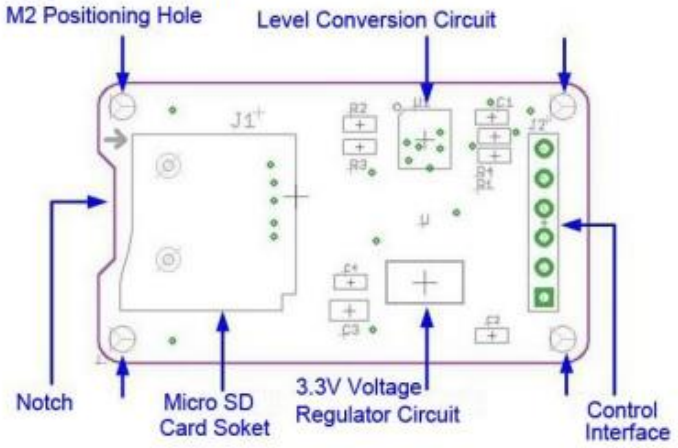
Description

- The module (MicroSD Card Adapter) is a Micro SD card reader module for reading and writing through the file system and the SPI interface driver, SCM system can be completed within a file MicroSD card
- Support Micro SD Card, Micro SDHC card (high speed card)
- Level conversion circuit board that can interface level is 5V or 3.3V
- Power supply is 4.5V ~ 5.5V, 3.3V voltage regulator circuit board
- Communications interface is a standard SPI interface
- 4 M2 screws positioning holes for easy installation
- Control Interface: A total of six pins (GND, VCC, MISO, MOSI, SCK, CS), GND to ground, VCC is the power supply, MISO, MOSI, SCK for SPI bus, CS is the chip select signal pin;
- 3.3V regulator circuit: LDO regulator output 3.3V for level conversion chip, Micro SD card supply;
- Level conversion circuit: Micro SD card to signal the direction of converts 3.3V, MicroSD card interface to control the direction of the MISO signal is also converted to 3.3V, general AVR microcontroller systems can read the signal;
- Micro SD card connector: self bomb deck, easy card insertion.
- Positioning holes: 4 M2 screws positioning holes with a diameter of 2.2mm, so the module is easy to install positioning, to achieve inter-module combination.

Interface Parameters:

Items	Min	Typical	Max	Unit
Power Voltage VCC	4.5	**5**	5.5	V
Current	0.2	**80**	200	mA
Interface Electrical Potential	3.3 or 5			V
Support Card Type	Micro SD Card(<=2G), Mirco SDHC Card(<=32G)			—
Size	42X24X12			mm
Weight	5			g

Mirco SD Card Interface Module:



APPENDIX F: SOFTWARE USED

Adobe Illustrator

Adobe Illustrator is a vector graphics editor and design program developed and marketed by Adobe Inc.

This software was used to generate designs for the laser cutter.

Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Microsoft Windows, macOS, and Linux) that is written in the Java programming language. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. The source code for the IDE is released under the GNU General Public License, version 2.

This software was used to upload code to the Arduinos and ESP modules.

Google Workspace

Google Workspace is a collection of cloud computing, productivity and collaboration tools, software and products developed and marketed by Google. Google Workspace consists of Gmail, Contacts, Calendar, Meet and Chat for communication; Currents for employee engagement; Drive for storage; and the Google Docs suite for content creation.

This software was used to organize team meetings, draft technical memos, share code, plan out tasks, assign jobs to team members, and document progress.

MySQL Workbench 8.0 CE

MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system.

This software was used to generate the databases for the server and API backend.

NodeJS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

This programming framework was used to develop our backend API.

NPM

NPM (originally short for Node Package Manager) is a package manager for the JavaScript programming language maintained by NPM, Inc. NPM is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called NPM, and an online database of public and paid-for private packages, called the NPM registry. The registry is accessed via the client, and the available packages can be browsed and searched via the NPM website. The package manager and the registry are managed by NPM, Inc.

This software package was used to host our server backend.

PM2

PM2 is a process manager for the JavaScript runtime Node.js.

This software was used to revive our server backend in the case of it crashing.

Python 3.8

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

This software was used to generate randomized datasets for testing and analyzing the test results.

Raspberry Pi OS

Raspberry Pi OS (formerly Raspbian) is a Debian-based operating system for Raspberry Pi. Since 2013, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers.

This was used to host the server and API backend on the Raspberry Pi.

SolidWorks

SolidWorks is a solid modeling computer-aided design (CAD) and computer-aided engineering (CAE) application published by Dassault Systèmes.

This software was used to generate all CAD models for our conceptual design.

Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

This software was used as our development environment.

APPENDIX G: CUSTOM-WRITTEN SOFTWARE

Arduino Code

button_test.ino

```
//This program tests the Arduino's ability to read button input
// This was developed for testing whether the ESP-01 can send data to the
database backend
#define READER 13

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(READER, INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  int val = digitalRead(READER);
  if(val == 0){
    Serial.print(1);
    };
    delay(2000);
}
```

calibrate_stepper.ino

```
// This program calibrates the stepper motor
#include <TM1637.h>

#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>


SoftwareSerial myserial(2, 3); //TX, RX
// CHANGED: Connect ESP TX and RX to RX and TX, respectively, of the MEGA
```

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C
for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LEDPIN  11

#define STEPPER_DIR 2
#define STEPPER_PULL 3

#define CLK 22//pins definitions for TM1637 and can be changed to other ports
#define DIO 23
TM1637 tm1637(CLK,DIO);

LiquidCrystal_I2C lcd1(0x27, 16, 2);
LiquidCrystal_I2C lcd2(0x26, 16, 2);

boolean setdir = LOW;
int stepper_delay = 100;

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
```

```
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(250000);
//  Serial.begin(9600);
  myserial.begin(9600);
  pinMode(STEPPER_PULL, OUTPUT);
  pinMode(STEPPER_DIR, OUTPUT);
  lcd1.init();
  lcd2.init();
  lcd1.backlight();
  lcd2.backlight();
  tm1637.init();
  tm1637.set(BRIGHT_TYPICAL);//BRIGHT_TYPICAL = 2,BRIGHT_DARKEST = 0,BRIGHTEST =
7;
  tm1637.display(0,1);
  tm1637.display(1,2);
  tm1637.point(1);
  tm1637.display(2,10);
  tm1637.display(3,11);
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(5000);
  lcd1.print("YEAR");
  lcd2.setCursor(4,0);
  lcd2.print("VISITORS");

  // Clear the buffer
  display.clearDisplay();
//  stepBackward(160000);
  reverseMotor();


}
```

```
void displayTextOLED(String subject) {
  display.clearDisplay();

  display.setTextSize(3); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(30, 5);
  display.println(subject);
  display.display();      // Show initial text
  delay(100);
}

long step_count = 0;

void loop() {
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;

  digitalWrite(STEPPER_DIR, setdir);
  digitalWrite(STEPPER_PULL, HIGH);
  delayMicroseconds(stepper_delay);
  digitalWrite(STEPPER_PULL, LOW);
//  step_count = step_count + 1;

  while(Serial.available()){
    incoming_string = Serial.readString();
    string_ready = true;
    };

  if(string_ready == true){
    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
    }
    int year = String(strings[0]).toInt();
```

```
    int visitors = String(strings[1]).toInt();
    String oled_vis = String(visitors);
    if(visitors<100){
      oled_vis = "00" + String(visitors);
      }else if(visitors<1000){
        oled_vis = "0" + String(visitors);
        }
    displayTextOLED(oled_vis);
    lcd1.clear();
    lcd1.setCursor(6,0);
    lcd1.print("YEAR");
    lcd1.setCursor(6,1);
    lcd1.print(String(year));
    Serial.print("Year: " + String(year) + "\tVisitors: " + String(visitors) +
"\n\n");
    };

//    Serial.println(step_count);

}

void reverseMotor(){
  setdir = !setdir;
}

void stepForward(long steps){
  for(long i = 0; i<steps; i++){
  digitalWrite(STEPPER_DIR, setdir);
  digitalWrite(STEPPER_PULL, HIGH);
  delayMicroseconds(stepper_delay);
  digitalWrite(STEPPER_PULL, LOW);
  delayMicroseconds(stepper_delay);
  }
  }

void stepBackward(long steps){
  reverseMotor();
  for(long i = 0; i<steps; i++){
  digitalWrite(STEPPER_DIR, setdir);
  digitalWrite(STEPPER_PULL, HIGH);
  delayMicroseconds(stepper_delay);
  digitalWrite(STEPPER_PULL, LOW);
  delayMicroseconds(stepper_delay);
  }
  }
```

clock.ino

```
// This program is used to control the car clock LED circuit

#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

File myFile;
LiquidCrystal_I2C lcd1(0x27, 16, 2);

#define DS_PIN 8
#define SHCP_PIN 9
#define STCP_PIN 10

int RED = 0;
int YELLOW = 1;
int GREEN = 2;

const int register_count = 5;

int HOURS[12][3] = {{12,24,37},
                    {0,13,25},
                    {1,14,26},
                    {2,15,27},
                    {3,16,28},
                    {4,5,17},
                    {6,18,29},
                    {7,19,30},
                    {8,23,31},
                    {9,20,32},
                    {10,21,33},
                    {11,22,34}};

float sample_data[12] = {
    207,
    1546,
    4359,
    5393,
    3519,
    1616,
```

```
    3101,
    2094,
    1553,
    614,
    504,
    0
};

boolean registers[8*register_count];

void setup() {
  Serial.begin(250000);
  pinMode(DS_PIN, OUTPUT);
  pinMode(SHCP_PIN, OUTPUT);
  pinMode(STCP_PIN, OUTPUT);
  pinMode(4,OUTPUT);
  lcd1.init();
  lcd1.backlight();
  lcd1.setCursor(2,0);
  lcd1.print("OEDK TRAFFIC");
  for(int hour = 0; hour<8*register_count; hour++){
    registers[hour] = LOW;
  };
  writereg();
}

void writereg(void){
  digitalWrite(STCP_PIN, LOW);
  for (int i = 8*register_count; i>=0; i--){
    digitalWrite(SHCP_PIN, LOW);
    digitalWrite(DS_PIN, registers[i]);
    digitalWrite(SHCP_PIN, HIGH);
    };
    digitalWrite(STCP_PIN, HIGH);
  }


void loop() {
  int clock_time = clockTime();
  if(clock_time == 0){
    Serial.println("AM");
    lcd1.setCursor(7,1);
    lcd1.print("AM");
    morningSD();
    }else{
```

```
      Serial.println("PM");
      lcd1.setCursor(7,1);
      lcd1.print("PM");
      afternoonSD();
      }
  for(int i = 0; i < 12; i++){
    registers[HOURS[i][trafficLevel(i)]] = HIGH;
    writereg();
  };
}

int clockTime(void){
  int time = 1;
  return time;
}


 float largest(float arr[], int n)
{
    int i;

    // Initialize maximum element
    int max = arr[0];

    // Traverse array elements from second and
    // compare every element with current max
    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}

//This function normalizes the traffic data for each hour and converts to analog
output (0 to 255)
int trafficLevel(int hour){
  float current_visits = sample_data[hour];
  if(current_visits <= 1000){
    return GREEN;
    }else if(current_visits <= 3000){
      return YELLOW;
      }else{
        return RED;
        }
//  float largest_num = float(largest(sample_data, 12));
```

```
//    float weight = current_visits/largest_num;
//    weight = 255*weight;
//    if(weight>255*2/3){
//      return RED;
//      }else if(weight<=255*2/3 && weight>=255*1/3){
//        return YELLOW;
//        }else{
//          return GREEN;
//          }
}

void afternoonSD(void){
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;

  // re-open the file for reading:
  myFile = SD.open("pm_data.txt");
  if (myFile) {
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      incoming_string = myFile.readStringUntil('\n');
      string_ready = true;
      if(string_ready == true){
//    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
    }
    int hour = String(strings[0]).toInt();
    int visitors = String(strings[1]).toInt();
    sample_data[hour-1] = visitors;
    Serial.println(String(hour) + "-" + String(visitors));
    delay(1000);
      }
    }
```

```
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening pm.txt");
  }
  }

void morningSD(void){
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;

  // re-open the file for reading:
  myFile = SD.open("am_data.txt");
  if (myFile) {
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      incoming_string = myFile.readStringUntil('\n');
      string_ready = true;
      if(string_ready == true){
//    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
    }
    int hour = String(strings[0]).toInt();
    int visitors = String(strings[1]).toInt();
    sample_data[hour-1] = visitors;
    Serial.println(String(hour) + "-" + String(visitors));
    delay(1000);
      }
    }
    // close the file:
    myFile.close();
  } else {
```

```
    // if the file didn't open, print an error:
    Serial.println("error opening am.txt");
  }
  }
```

esp_get_api_test_pi.ino

```
// This program is for testing the ESP-01 connection to the Raspberry Pi server
// This code is to be uploaded to the ESP, NOT the Arduino
/**************************************************************
    A sample project for making a HTTP/HTTPS GET request on an ESP8266
    and parsing it from JSON
    It will connect to the given request, parse the JSON and print the
    body to serial monitor
 **************************************************************/

 //Find ip address on mac: "ipconfig getifaddr en0"


// ----------------------------
// Standard Libraries
// ----------------------------

#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

// ----------------------------
// Additional Libraries - each one of these will need to be installed.
// ----------------------------

#include <ArduinoJson.h>
// Library used for parsing Json from the API responses

// Search for "Arduino Json" in the Arduino Library manager
// https://github.com/bblanchon/ArduinoJson

//------- Replace the following! ------
char ssid[] = "WAVLINKN";       // your network SSID (name)
char password[] = "EngI120sp22";  // your network key

// For Non-HTTPS requests
 WiFiClient client;

// For HTTPS requests
```

```
//WiFiClientSecure client;


// Just the base of the URL you want to connect to
#define TEST_HOST "192.168.10.186"

// OPTIONAL - The finferprint of the site you want to connect to.
//#define TEST_HOST_FINGERPRINT "89 25 60 5D 50 44 FC C0 85 2B 98 D7 D3 66 52 28
68 4D E6 E2"
// The finger print will change every few months.


void setup() {

  Serial.begin(250000);

  // Connect to the WiFI
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  // Attempt to connect to Wifi network:
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  IPAddress ip = WiFi.localIP();
  Serial.println(ip);
  delay(5000);

  //--------

  // If you don't need to check the fingerprint
//   client.setInsecure();

}

void getVisitors(String year) {
```

```
  // Opening connection to server (Use 80 as port if HTTP)
  if (!client.connect(TEST_HOST, 80))
  {
    Serial.println(F("Connection failed"));
    Serial.print("SD");
    return;
  }

  // give the esp a breather
  yield();

  // Send HTTP request
  client.print(F("GET "));
  // This is the second half of a request (everything that comes after the base
URL)
  client.print("/year?year="+year); // %2C == ,
  client.println(F(" HTTP/1.1"));

  //Headers
  client.print(F("Host: "));
  client.println(TEST_HOST);

  client.println(F("Cache-Control: no-cache"));

  if (client.println() == 0)
  {
    Serial.println(F("Failed to send request"));
    return;
  }
  //delay(100);
  // Check HTTP status
  char status[32] = {0};
  client.readBytesUntil('\r', status, sizeof(status));
  if (strcmp(status, "HTTP/1.1 200 OK") != 0)
  {
    Serial.print(F("Unexpected response: "));
    Serial.println(status);
    Serial.print("SD");
    return;
  }

  // Skip HTTP headers
  char endOfHeaders[] = "\r\n\r\n";
  if (!client.find(endOfHeaders))
  {
```

```
    Serial.println(F("Invalid response"));
    Serial.print("SD");
    return;
  }

  // This is probably not needed for most, but I had issues
  // with the Tindie api where sometimes there were random
  // characters coming back before the body of the response.
  // This will cause no hard to leave it in
  // peek() will look at the character, but not take it off the queue
  while (client.available() && client.peek() != '{')
  {
    char c = 0;
    client.readBytes(&c, 1);
    Serial.print(c);
    Serial.println("BAD");
  }

  //  // While the client is still availble read each
  //  // byte and print to the serial monitor
  //  while (client.available()) {
  //    char c = 0;
  //    client.readBytes(&c, 1);
  //    Serial.print(c);
  //  }

  //Use the ArduinoJson Assistant to calculate this:

  //StaticJsonDocument<192> doc;
//  DynamicJsonDocument doc(192); //For ESP32/ESP8266 you'll mainly use dynamic.

//  DeserializationError error = deserializeJson(doc, client);

  // Stream& input;

  StaticJsonDocument<48> doc;

  DeserializationError error = deserializeJson(doc, client);

  if (error) {
    Serial.println(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    Serial.print("SD");
    return;
  } else {
```

```
    int visitors = doc["visitors"]; // 5819
    Serial.print(year+","+visitors);

  }


}

int getYears(void) {

  // Opening connection to server (Use 80 as port if HTTP)
  if (!client.connect(TEST_HOST, 80))
  {
    Serial.println(F("Connection failed"));
    Serial.print("SD");
    return 0;
  }

  // give the esp a breather
  yield();

  // Send HTTP request
  client.print(F("GET "));
  // This is the second half of a request (everything that comes after the base
URL)
  client.print("/year_count"); // %2C == ,
  client.println(F(" HTTP/1.1"));

  //Headers
  client.print(F("Host: "));
  client.println(TEST_HOST);

  client.println(F("Cache-Control: no-cache"));

  if (client.println() == 0)
  {
    Serial.println(F("Failed to send request"));
    Serial.print("SD");
    return 0;
  }
  //delay(100);
  // Check HTTP status
  char status[32] = {0};
  client.readBytesUntil('\r', status, sizeof(status));
  if (strcmp(status, "HTTP/1.1 200 OK") != 0)
```

```
  {
    Serial.println(F("Unexpected response: "));
    Serial.println(status);
    Serial.print("SD");
    return 0;
  }

  // Skip HTTP headers
  char endOfHeaders[] = "\r\n\r\n";
  if (!client.find(endOfHeaders))
  {
    Serial.println(F("Invalid response"));
    Serial.print("SD");
    return 0;
  }

  // This is probably not needed for most, but I had issues
  // with the Tindie api where sometimes there were random
  // characters coming back before the body of the response.
  // This will cause no hard to leave it in
  // peek() will look at the character, but not take it off the queue
  while (client.available() && client.peek() != '{')
  {
    char c = 0;
    client.readBytes(&c, 1);
    Serial.print(c);
    Serial.println("BAD");
  }

  //  // While the client is still availble read each
  //  // byte and print to the serial monitor
  //  while (client.available()) {
  //    char c = 0;
  //    client.readBytes(&c, 1);
  //    Serial.print(c);
  //  }

  //Use the ArduinoJson Assistant to calculate this:

  //StaticJsonDocument<192> doc;
//  DynamicJsonDocument doc(192); //For ESP32/ESP8266 you'll mainly use dynamic.

//  DeserializationError error = deserializeJson(doc, client);

  // Stream& input;
```

```
  StaticJsonDocument<48> doc;

  DeserializationError error = deserializeJson(doc, client);

  if (error) {
    Serial.println(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    Serial.print("SD");
    return -1;
  } else {
    int year_count = doc["year_count"]; // 7
    return year_count;

  }


}

void loop() {
  // put your main code here, to run repeatedly:
  int year_count = getYears();
  for (int i = 0; i < year_count+1; i++){ // dynamically cycle through each
year's visitor count
  getVisitors(String(i+2015));
  int minutes = 1;
  delay(1000*100); //update every 100 seconds
  }

}
```

esp_get_api_test_home.ino

```
// This program is for testing the ESP-01 connection to the server hosted on my
// computer
// This code is to be uploaded to the ESP, NOT the Arduino
//API GET request test
#include <WiFi.h>
#include <HttpClient.h>
#include <EthernetClient.h>
#include <ArduinoJson.h>

char* ssid = "ORBI43"; //Wifi Access Point name
char* password = "vastcream373"; //Wifi password
```

```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid,password);
  Serial.println("Connecting to WiFi...");

  while(WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
    };

   Serial.println("\nConnected to WiFi!");
   Serial.println("IP Address: ");
   Serial.println(WiFi.localIP());
}

void loop() {
  if((WiFi.status() == WL_CONNECTED)){

    EthernetClient client;
    HttpClient http(client);

    client.begin("http://localhost/year?year=2019");
    int httpCode = client.GET();

    if(httpCode > 0){

    String payload = client.getString();
    Serial.println("\nStatus Code: " + String(httpCode));
    Serial.println(payload);

    char json[500];
    payload.replace(" ", "");
    payload.replace("\n", "");
    payload.trim();
    payload.remove(0,1);
    payload.toCharArray(json,500);

    StaticJsonDocument<200> doc;
    deserializeJson(doc,json);

    int visitors = doc["visitors"];
    Serial.write(visitors);

    client.end();
```

```
      }else{
        Serial.println("Error on HTTP request!");
        }

    }else{
      Serial.println("Connection lost!");
      }
  delay(10000);
}
```

esp_test.ino

```
// This program is testing the Serial communication between the ESP and the
// Arduino by having the ESP write and the Arduino read
// This code is to be uploaded to the ESP, NOT the Arduino

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.write("Hello from ESP-01");
  delay(2000);
}
```

esp_trigger_us_test.ino

```
// This program is testing whether the ESP module can read Arduino ultrasonic
// sensor data and record a visitor with the timestamp to the database
// This code is to be uploaded to the ESP, NOT the Arduino
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

char ssid[] = "ORBI43";       // your network SSID (name)
char password[] = "vastcream373";  // your network key

WiFiClient client;

#define TEST_HOST "192.168.1.6"
```

```
void setup() {
Serial.begin(9600);
// Connect to the WiFI
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  // Attempt to connect to Wifi network:
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  IPAddress ip = WiFi.localIP();
  Serial.println(ip);
  delay(5000);
}

void recordVisitor(void) {

  // Opening connection to server (Use 80 as port if HTTP)
  if (!client.connect(TEST_HOST, 80))
  {
    Serial.println(F("Connection failed"));
    Serial.print("SD");
    return;
  }

  // give the esp a breather
  yield();

  // Send HTTP request
  client.print(F("GET "));
  // This is the second half of a request (everything that comes after the base
URL)
  client.print("/sensor_triggered"); // %2C == ,
  client.println(F(" HTTP/1.1"));

  //Headers
```

```
  client.print(F("Host: "));
  client.println(TEST_HOST);

  client.println(F("Cache-Control: no-cache"));

  if (client.println() == 0)
  {
    Serial.println(F("Failed to send request"));
    return;
  }
  //delay(100);
  // Check HTTP status
  char status[32] = {0};
  client.readBytesUntil('\r', status, sizeof(status));
  if (strcmp(status, "HTTP/1.1 200 OK") != 0)
  {
    Serial.print(F("Unexpected response: "));
    Serial.println(status);
    Serial.print("SD");
    return;
  }

  // Skip HTTP headers
  char endOfHeaders[] = "\r\n\r\n";
  if (!client.find(endOfHeaders))
  {
    Serial.println(F("Invalid response"));
    Serial.print("SD");
    return;
  }

  // This is probably not needed for most, but I had issues
  // with the Tindie api where sometimes there were random
  // characters coming back before the body of the response.
  // This will cause no hard to leave it in
  // peek() will look at the character, but not take it off the queue
  while (client.available() && client.peek() != '{')
  {
    char c = 0;
    client.readBytes(&c, 1);
    Serial.print(c);
    Serial.println("BAD");
  }

  //  // While the client is still availble read each
```

```
  //  // byte and print to the serial monitor
  //  while (client.available()) {
  //    char c = 0;
  //    client.readBytes(&c, 1);
  //    Serial.print(c);
  //  }

  //Use the ArduinoJson Assistant to calculate this:

  //StaticJsonDocument<192> doc;
//  DynamicJsonDocument doc(192); //For ESP32/ESP8266 you'll mainly use dynamic.

//  DeserializationError error = deserializeJson(doc, client);

  // Stream& input;

  StaticJsonDocument<48> doc;

  DeserializationError error = deserializeJson(doc, client);

  if (error) {
    Serial.println(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    Serial.print("SD");
    return;
  } else {
    String post_status = doc["Status"]; // 5819
    Serial.print(post_status);

  }


}

void loop() {
if(Serial.available()){
  int reading = Serial.readString().toInt();
  if(reading>0){
    recordVisitor();
    }
  }
}
```

i2c_scanner.ino

```
// This program is to determine if the solder bridge on the 2nd LCD screen
// was successful and the I2C address had changed

// --------------------------------------
// i2c_scanner
//
// Version 1
//    This program (or code that looks like it)
//    can be found in many places.
//    For example on the Arduino.cc forum.
//    The original author is not know.
// Version 2, Juni 2012, Using Arduino 1.0.1
//     Adapted to be as simple as possible by Arduino.cc user Krodal
// Version 3, Feb 26  2013
//    V3 by louarnold
// Version 4, March 3, 2013, Using Arduino 1.0.3
//    by Arduino.cc user Krodal.
//    Changes by louarnold removed.
//    Scanning addresses changed from 0...127 to 1...119,
//    according to the i2c scanner by Nick Gammon
//    https://www.gammon.com.au/forum/?id=10896
// Version 5, March 28, 2013
//    As version 4, but address scans now to 127.
//    A sensor seems to use address 120.
// Version 6, November 27, 2015.
//    Added waiting for the Leonardo serial communication.
//
//
// This sketch tests the standard 7-bit addresses
// Devices with higher bit address might not be seen properly.
//

#include <Wire.h>


void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);             // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}
```

```
void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

LCD_code.ino

```
// This code tests the LCD screen to make sure it is working properly
#include <LiquidCrystal.h>
int Contrast=75;
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

 void setup()
 {
    analogWrite(6,Contrast);
     lcd.begin(16, 2);
  }
     void loop()
 {
     lcd.setCursor(0, 0);
     lcd.print("Only Innovative");

    lcd.setCursor(0, 1);
     lcd.print("Subscribe");
 }
```

LED_matrix_V1.ino

```
// This software tests the LED matrix to see how
// shift register indexing works

// Definition for that latch clock to be connected to pin 8
// See https://www.arduino.cc/en/Reference/PortManipulation
#define ShiftRegisterPWM_LATCH_PORT PORTB
#define ShiftRegisterPWM_LATCH_MASK 1
#include "ShiftRegisterPWM.h"

int numOfRegisters = 3;
int resolution = 60;

void setup() {
  pinMode(2, OUTPUT); // data
  pinMode(3, OUTPUT); // clock
  pinMode(8, OUTPUT); // latch pin is now 8 (opposed to the default value 4)
}

void loop()
{
```

```
  ShiftRegisterPWM sr(numOfRegisters, resolution); // Create a shift register
object with 2 chained registers and a resolution of 255

  sr.interrupt(ShiftRegisterPWM::UpdateFrequency::SuperFast);
  while(true){
//    sr.set(0,resolution); //led 1 green
//    sr.set(1,resolution); //led 1 red
//    sr.set(2,resolution); //led 2 blue
//    sr.set(3,resolution); //led 2 red
//    sr.set(4,resolution); //led 2 green
//    sr.set(5,resolution); //led 3 blue
//    sr.set(6,resolution); //led 3 green
//    sr.set(7,resolution); //led 3 red
//    sr.set(8,resolution);
//    sr.set(9,resolution);
//    sr.set(10,resolution);
//    sr.set(11,resolution);
//    sr.set(12,resolution);
//    sr.set(13,resolution);
//    sr.set(14,resolution);
//    sr.set(15,resolution); //led 1 blue
//    sr.set(16,resolution);
    for(int i = 0; i<numOfRegisters*8; i++){
      sr.set(i,resolution);
      delay(500);
      sr.set(i,0);
      delay(500);
      };
    };

}
```

LED_matrix_V2.ino

```
// This program tests a new method of controlling the shift registers

#define DS_PIN 8 //data pin
#define SHCP_PIN 9 // clock pin
#define STCP_PIN 10 // latch pin

const int register_count = 5;

void setup() {
```

```
  pinMode(DS_PIN, OUTPUT);
  pinMode(SHCP_PIN, OUTPUT);
  pinMode(STCP_PIN, OUTPUT);
  writereg();
}

boolean registers[8*register_count];

//void tester(void){
//  digitalWrite(latchPin, LOW);
//
//    // Shift out the bits
//    shiftOut(dataPin, clockPin, MSBFIRST, numberToDisplay);
//
//    // ST_CP HIGH change LEDs
//    digitalWrite(latchPin, HIGH);
//  }

void writereg(void){
  digitalWrite(STCP_PIN, LOW);
  for (int i = 8*register_count; i>=0; i--){
    digitalWrite(SHCP_PIN, LOW);
    digitalWrite(DS_PIN, registers[i]);
    digitalWrite(SHCP_PIN, HIGH);
    };
    digitalWrite(STCP_PIN, HIGH);
  }

void loop() {
  for(int i = 0; i<register_count*8; i++){
    registers[i] = HIGH;
    writereg();
    delay(1000);
    };

}
```

MatrixFinal.ino

```
// This program is the final code for the LED matrix that uses neopixels

#include <FastLED.h>
```

```
#define LED_PIN     13
#define NUM_LEDS    100
#define BRIGHTNESS  64
#define LED_TYPE    WS2811
#define COLOR_ORDER GRB

int male_ratio[3] = {0, 0, 255};
int female_ratio[3] = {105, 255, 180};

int rice[100][3] = {};

CRGB leds[NUM_LEDS];

#define UPDATES_PER_SECOND 100

// This example shows several ways to set up and use 'palettes' of colors
// with FastLED.
//
// These compact palettes provide an easy way to re-colorize your
// animation on the fly, quickly, easily, and with low overhead.
//
// USING palettes is MUCH simpler in practice than in theory, so first just
// run this sketch, and watch the pretty lights as you then read through
// the code.  Although this sketch has eight (or more) different color schemes,
// the entire sketch compiles down to about 6.5K on AVR.
//
// FastLED provides a few pre-configured color palettes, and makes it
// extremely easy to make up your own color schemes with palettes.
//
// Some notes on the more abstract 'theory and practice' of
// FastLED compact palettes are at the bottom of this file.



CRGBPalette16 currentPalette;
TBlendType    currentBlending;

extern CRGBPalette16 myRedWhiteBluePalette;
extern const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM;


void setup() {
    delay( 3000 ); // power-up safety delay
    FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection( TypicalLEDStrip );
```

```
    FastLED.addLeds<LED_TYPE, 12, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 11, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 10, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 9, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 8, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 7, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 6, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 5, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.addLeds<LED_TYPE, 4, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
    FastLED.setBrightness(  BRIGHTNESS );


    currentPalette = RainbowColors_p;
    currentBlending = LINEARBLEND;
}

void Rice(void){
  leds[0]=CRGB::White;
  leds[1]=CRGB::White;
  leds[2]=CRGB::Black;
  leds[3]=CRGB::Blue;
  leds[4]=CRGB::Blue;
  leds[5]=CRGB::Black;
  leds[6]=CRGB::Black;
  leds[7]=CRGB::Blue;
  leds[8]=CRGB::Blue;
  leds[9]=CRGB::Blue;
  leds[10]=CRGB::White;
  leds[11]=CRGB::White;
  leds[12]=CRGB::Black;
  leds[13]=CRGB::Blue;
  leds[14]=CRGB::Blue;
  leds[15]=CRGB::Black;
  leds[16]=CRGB::White;
  leds[17]=CRGB::Blue;
  leds[18]=CRGB::Blue;
```

```
 leds[19]=CRGB::Blue;
 leds[20]=CRGB::Black;
 leds[21]=CRGB::White;
 leds[22]=CRGB::Black;
 leds[23]=CRGB::Black;
 leds[24]=CRGB::Blue;
 leds[25]=CRGB::Black;
 leds[26]=CRGB::White;
 leds[27]=CRGB::Blue;
 leds[28]=CRGB::Black;
 leds[29]=CRGB::Blue;
 leds[30]=CRGB::Black;
 leds[31]=CRGB::White;
 leds[32]=CRGB::Black;
 leds[33]=CRGB::Black;
 leds[34]=CRGB::Blue;
 leds[35]=CRGB::Black;
 leds[36]=CRGB::White;
 leds[37]=CRGB::Blue;
 leds[38]=CRGB::Black;
 leds[39]=CRGB::Blue;
 leds[40]=CRGB::White;
 leds[41]=CRGB::White;
 leds[42]=CRGB::Black;
 leds[43]=CRGB::Black;
 leds[44]=CRGB::Blue;
 leds[45]=CRGB::Black;
 leds[46]=CRGB::Black;
 leds[47]=CRGB::Blue;
 leds[48]=CRGB::Blue;
 leds[49]=CRGB::Blue;
 leds[50]=CRGB::White;
 leds[51]=CRGB::White;
 leds[52]=CRGB::Black;
 leds[53]=CRGB::Black;
 leds[54]=CRGB::Blue;
 leds[55]=CRGB::Black;
 leds[56]=CRGB::White;
 leds[57]=CRGB::Black;
 leds[58]=CRGB::Blue;
 leds[59]=CRGB::Blue;
 leds[60]=CRGB::Black;
 leds[61]=CRGB::White;
 leds[62]=CRGB::Black;
 leds[63]=CRGB::Black;
```

```
  leds[64]=CRGB::Blue;
  leds[65]=CRGB::Black;
  leds[66]=CRGB::White;
  leds[67]=CRGB::Blue;
  leds[68]=CRGB::Blue;
  leds[69]=CRGB::Blue;
  leds[70]=CRGB::Black;
  leds[71]=CRGB::White;
  leds[72]=CRGB::Black;
  leds[73]=CRGB::Black;
  leds[74]=CRGB::Blue;
  leds[75]=CRGB::Black;
  leds[76]=CRGB::White;
  leds[77]=CRGB::Blue;
  leds[78]=CRGB::Black;
  leds[79]=CRGB::Blue;
  leds[80]=CRGB::White;
  leds[81]=CRGB::White;
  leds[82]=CRGB::Black;
  leds[83]=CRGB::Blue;
  leds[84]=CRGB::Blue;
  leds[85]=CRGB::Black;
  leds[86]=CRGB::White;
  leds[87]=CRGB::Blue;
  leds[88]=CRGB::Black;
  leds[89]=CRGB::Blue;
  leds[90]=CRGB::White;
  leds[91]=CRGB::White;
  leds[92]=CRGB::Black;
  leds[93]=CRGB::Blue;
  leds[94]=CRGB::Blue;
  leds[95]=CRGB::Black;
  leds[96]=CRGB::White;
  leds[97]=CRGB::Blue;
  leds[98]=CRGB::Black;
  leds[99]=CRGB::Blue;
  FastLED.show();

  }

void texas(void){
  leds[0] = CRGB::White;
  leds[1] = CRGB::White;
  leds[2] = CRGB::White;
  leds[3] = CRGB::White;
```

```
  leds[4] = CRGB::White;
  leds[5] = CRGB::White;
  leds[6] = CRGB::Blue;
  leds[7] = CRGB::Blue;
  leds[8] = CRGB::Blue;
  leds[9] = CRGB::Blue;
  leds[10] = CRGB::White;
  leds[11] = CRGB::White;
  leds[12] = CRGB::White;
  leds[13] = CRGB::White;
  leds[14] = CRGB::White;
  leds[15] = CRGB::White;
  leds[16] = CRGB::Blue;
  leds[17] = CRGB::Blue;
  leds[18] = CRGB::Blue;
  leds[19] = CRGB::Blue;
  leds[20] = CRGB::White;
  leds[21] = CRGB::White;
  leds[22] = CRGB::White;
  leds[23] = CRGB::White;
  leds[24] = CRGB::White;
  leds[25] = CRGB::White;
  leds[26] = CRGB::Blue;
  leds[27] = CRGB::Blue;
  leds[28] = CRGB::Blue;
  leds[29] = CRGB::Blue;
  leds[30] = CRGB::White;
  leds[31] = CRGB::White;
  leds[32] = CRGB::White;
  leds[33] = CRGB::White;
  leds[34] = CRGB::White;
  leds[35] = CRGB::White;
  leds[36] = CRGB::Blue;
  leds[37] = CRGB::Blue;
  leds[38] = CRGB::Blue;
  leds[39] = CRGB::Blue;
  leds[40] = CRGB::White;
  leds[41] = CRGB::White;
  leds[42] = CRGB::White;
  leds[43] = CRGB::White;
  leds[44] = CRGB::White;
  leds[45] = CRGB::White;
  leds[46] = CRGB::Blue;
  leds[47] = CRGB::White;
  leds[48] = CRGB::White;
```

```
  leds[49] = CRGB::Blue;
  leds[50] = CRGB::Red;
  leds[51] = CRGB::Red;
  leds[52] = CRGB::Red;
  leds[53] = CRGB::Red;
  leds[54] = CRGB::Red;
  leds[55] = CRGB::Red;
  leds[56] = CRGB::Blue;
  leds[57] = CRGB::White;
  leds[58] = CRGB::White;
  leds[59] = CRGB::Blue;
  leds[60] = CRGB::Red;
  leds[61] = CRGB::Red;
  leds[62] = CRGB::Red;
  leds[63] = CRGB::Red;
  leds[64] = CRGB::Red;
  leds[65] = CRGB::Red;
  leds[66] = CRGB::Blue;
  leds[67] = CRGB::Blue;
  leds[68] = CRGB::Blue;
  leds[69] = CRGB::Blue;
  leds[70] = CRGB::Red;
  leds[71] = CRGB::Red;
  leds[72] = CRGB::Red;
  leds[73] = CRGB::Red;
  leds[74] = CRGB::Red;
  leds[75] = CRGB::Red;
  leds[76] = CRGB::Blue;
  leds[77] = CRGB::Blue;
  leds[78] = CRGB::Blue;
  leds[79] = CRGB::Blue;
  leds[80] = CRGB::Red;
  leds[81] = CRGB::Red;
  leds[82] = CRGB::Red;
  leds[83] = CRGB::Red;
  leds[84] = CRGB::Red;
  leds[85] = CRGB::Red;
  leds[86] = CRGB::Blue;
  leds[87] = CRGB::Blue;
  leds[88] = CRGB::Blue;
  leds[89] = CRGB::Blue;
  leds[90] = CRGB::Red;
  leds[91] = CRGB::Red;
  leds[92] = CRGB::Red;
  leds[93] = CRGB::Red;
```

```
  leds[94] = CRGB::Red;
  leds[95] = CRGB::Red;
  leds[96] = CRGB::Blue;
  leds[97] = CRGB::Blue;
  leds[98] = CRGB::Blue;
  leds[99] = CRGB::Blue;
  FastLED.show();
  }


void loop()
{
    Rice();
    delay(1000*5);
    ChangePalettePeriodically();

    static uint8_t startIndex = 0;
    startIndex = startIndex + 1; /* motion speed */

    FillLEDsFromPaletteColors( startIndex);

    FastLED.show();
    FastLED.delay(1000 / UPDATES_PER_SECOND);
    delay(5000);
    maleToFem(40);
    delay(1000 * 20);
    texas();
    delay(1000*5);

}

void maleToFem(int percentage){
  for(int i = 0; i < NUM_LEDS; i++){
    if(i%10 < int(percentage/10)){
      leds[i] = CRGB(female_ratio[1],female_ratio[0], female_ratio[2]);
      }else if(i%10 < 10){
        leds[i] = CRGB(male_ratio[1],male_ratio[0], male_ratio[2]);
        }
    }
    FastLED.show();
  }

//void maleToFem(void){
//  for(int i = 0; i < NUM_LEDS; i++){
//    if(i < 4){
```

```
//      leds[i] = CRGB(female_ratio[1],female_ratio[0], female_ratio[2]);
//      }else if(i < 10){
//        leds[i] = CRGB(male_ratio[1],male_ratio[0], male_ratio[2]);
//        }else if(i >= 10){
//          if(i - 10 < 4){
//            leds[i] = CRGB(female_ratio[1],female_ratio[0], female_ratio[2]);
//            }else if(i - 10 < 10){
//              leds[i] = CRGB(male_ratio[1],male_ratio[0], male_ratio[2]);
//              }else if(i >= 20){
//                if(i - 20 < 4){
//                  leds[i] = CRGB(female_ratio[1],female_ratio[0],
female_ratio[2]);
//                  }else if(i - 20 < 10){
//                    leds[i] = CRGB(male_ratio[1],male_ratio[0], male_ratio[2]);
//                    }else if(i >= 30){
//                      leds[i] = CRGB(male_ratio[1],male_ratio[0],
male_ratio[2]);
//                      }
//                }
//          }
//    }
//    FastLED.show();
//  }

void FillLEDsFromPaletteColors( uint8_t colorIndex)
{
    uint8_t brightness = 255;

    for( int i = 0; i < NUM_LEDS; ++i) {
        leds[i] = ColorFromPalette( currentPalette, colorIndex, brightness,
currentBlending);
        colorIndex += 3;
    }
}


// There are several different palettes of colors demonstrated here.
//
// FastLED provides several 'preset' palettes: RainbowColors_p,
RainbowStripeColors_p,
// OceanColors_p, CloudColors_p, LavaColors_p, ForestColors_p, and PartyColors_p.
//
// Additionally, you can manually define your own color palettes, or you can
write
// code that creates color palettes on the fly.  All are shown here.
```

```
void ChangePalettePeriodically()
{
    uint8_t secondHand = (millis() / 1000) % 60;
    static uint8_t lastSecond = 99;

    if( lastSecond != secondHand) {
        lastSecond = secondHand;
        if( secondHand ==  0)  { currentPalette =
RainbowColors_p;         currentBlending = LINEARBLEND; }
        if( secondHand == 10)  { currentPalette =
RainbowStripeColors_p;   currentBlending = NOBLEND;  }
        if( secondHand == 15)  { currentPalette =
RainbowStripeColors_p;   currentBlending = LINEARBLEND; }
        if( secondHand == 20)  {
SetupPurpleAndGreenPalette();             currentBlending = LINEARBLEND; }
        if( secondHand == 25)  {
SetupTotallyRandomPalette();              currentBlending = LINEARBLEND; }
        if( secondHand == 30)  {
SetupBlackAndWhiteStripedPalette();       currentBlending = NOBLEND; }
        if( secondHand == 35)  {
SetupBlackAndWhiteStripedPalette();       currentBlending = LINEARBLEND; }
        if( secondHand == 40)  { currentPalette =
CloudColors_p;           currentBlending = LINEARBLEND; }
        if( secondHand == 45)  { currentPalette =
PartyColors_p;           currentBlending = LINEARBLEND; }
        if( secondHand == 50)  { currentPalette = myRedWhiteBluePalette_p;
currentBlending = NOBLEND;  }
        if( secondHand == 55)  { currentPalette = myRedWhiteBluePalette_p;
currentBlending = LINEARBLEND; }
    }
}

// This function fills the palette with totally random colors.
void SetupTotallyRandomPalette()
{
    for( int i = 0; i < 16; ++i) {
        currentPalette[i] = CHSV( random8(), 255, random8());
    }
}

// This function sets up a palette of black and white stripes,
// using code.  Since the palette is effectively an array of
// sixteen CRGB colors, the various fill_* functions can be used
// to set them up.
```

```
void SetupBlackAndWhiteStripedPalette()
{
    // 'black out' all 16 palette entries...
    fill_solid( currentPalette, 16, CRGB::Black);
    // and set every fourth one to white.
    currentPalette[0] = CRGB::White;
    currentPalette[4] = CRGB::White;
    currentPalette[8] = CRGB::White;
    currentPalette[12] = CRGB::White;

}

// This function sets up a palette of purple and green stripes.
void SetupPurpleAndGreenPalette()
{
    CRGB purple = CHSV( HUE_PURPLE, 255, 255);
    CRGB green  = CHSV( HUE_GREEN, 255, 255);
    CRGB black  = CRGB::Black;

    currentPalette = CRGBPalette16(
                                   green,  green,  black,  black,
                                   purple, purple, black,  black,
                                   green,  green,  black,  black,
                                   purple, purple, black,  black );
}


// This example shows how to set up a static color palette
// which is stored in PROGMEM (flash), which is almost always more
// plentiful than RAM.  A static PROGMEM palette like this
// takes up 64 bytes of flash.
const TProgmemPalette16 myRedWhiteBluePalette_p PROGMEM =
{
    CRGB::Red,
    CRGB::Gray, // 'white' is too bright compared to red and blue
    CRGB::Blue,
    CRGB::Black,

    CRGB::Red,
    CRGB::Gray,
    CRGB::Blue,
    CRGB::Black,

    CRGB::Red,
    CRGB::Red,
```

```
    CRGB::Gray,
    CRGB::Gray,
    CRGB::Blue,
    CRGB::Blue,
    CRGB::Black,
    CRGB::Black
};



// Additional notes on FastLED compact palettes:
//
// Normally, in computer graphics, the palette (or "color lookup table")
// has 256 entries, each containing a specific 24-bit RGB color.  You can then
// index into the color palette using a simple 8-bit (one byte) value.
// A 256-entry color palette takes up 768 bytes of RAM, which on Arduino
// is quite possibly "too many" bytes.
//
// FastLED does offer traditional 256-element palettes, for setups that
// can afford the 768-byte cost in RAM.
//
// However, FastLED also offers a compact alternative.  FastLED offers
// palettes that store 16 distinct entries, but can be accessed AS IF
// they actually have 256 entries; this is accomplished by interpolating
// between the 16 explicit entries to create fifteen intermediate palette
// entries between each pair.
//
// So for example, if you set the first two explicit entries of a compact
// palette to Green (0,255,0) and Blue (0,0,255), and then retrieved
// the first sixteen entries from the virtual palette (of 256), you'd get
// Green, followed by a smooth gradient from green-to-blue, and then Blue.
```

oled_backup.ino

```
// This program is the offline failsafe for the rocket OLED screen

void setup() {
  Serial.begin(9600);
}

int visitors[8][2] = {
      {2015, 2552}, {2016, 5212}, {2017, 5821}, {2018, 5319}, {2019, 5819},
{2020, 644}, {2021, 107}, {2022, 45}
  };
```

```
void loop() {
  // put your main code here, to run repeatedly:
  for(int i = 0; i<8; i++){
    Serial.print(String(visitors[i][0]) + "," + String(visitors[i][1]));
    delay(10000);
    }

}
```

oled_display_iot.ino

```
// This program tests the IoT pipeline to see if the Arduino can read
// data from the ESP module and output it to the OLED screen

#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

SoftwareSerial myserial(2, 3); //TX, RX

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C
for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LEDPIN  11

//// defines pins numbers
//const int trigPin1 = 7;
//const int echoPin1 = 6;
```

```
//const int trigPin2 = 10;
//const int echoPin2 = 11;
//// defines variables
//long duration;
//int distance;

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(9600);
  myserial.begin(9600);
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(5000);

  // Clear the buffer
  display.clearDisplay();

}

void displayTextOLED(String subject) {
```

```
  display.clearDisplay();

  display.setTextSize(3); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(30, 5);
  display.println(subject);
  display.display();      // Show initial text
  delay(100);
}

void loop() {
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;

  while(myserial.available()){
    incoming_string = myserial.readString();
    string_ready = true;
    };

  if(string_ready == true){
    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
    }
    int year = String(strings[0]).toInt();
    int visitors = String(strings[1]).toInt();
    displayTextOLED(String(visitors));
    Serial.print("Year: " + String(year) + "\tVisitors: " + String(visitors) +
"\n\n");
    };

}
```

rocket_final.ino

```
// This is the finished program for the rocket circuit

#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <TM1637.h>
#include <SD.h>

File myFile;

// CHANGED: Connect ESP TX and RX to RX and TX, respectively, of the MEGA

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C
for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LEDPIN  11

#define STEPPER_DIR 2
#define STEPPER_PULL 3


LiquidCrystal_I2C lcd1(0x27, 16, 2);
LiquidCrystal_I2C lcd2(0x26, 16, 2);

int seg1_clk = 22;
int seg1_dio = 23;
TM1637 tm(seg1_clk, seg1_dio);
```

```
boolean setdir = LOW;
int stepper_delay = 100;

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(250000);
//  Serial.begin(9600);
  pinMode(STEPPER_PULL, OUTPUT);
  pinMode(STEPPER_DIR, OUTPUT);
  pinMode(53, OUTPUT);
  lcd1.init();
  lcd2.init();
  lcd1.backlight();
  lcd2.backlight();
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  if (!SD.begin(53)) {
Serial.println("initialization failed!");
return;
}
Serial.println("initialization done.");

  tm.init();
  tm.set(2);
```

```
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(5000);
  lcd1.print("YEAR");
  lcd2.setCursor(4,0);
  lcd2.print("VISITORS");

  // Clear the buffer
  display.clearDisplay();
  stepForward(160000);
  stepBackward(160000);
  reverseMotor();


}

void displayTextOLED(String subject) {
  display.clearDisplay();

  display.setTextSize(3); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(30, 5);
  display.println(subject);
  display.display();      // Show initial text
  delay(100);
}

long step_count = 0;
long moved = 0;

void loop() {
//  boolean iotstatus = iotMethod();
//  if(iotstatus == false){
//    sdBackup();
//    }
sdBackup();

//    Serial.println(step_count);

}

boolean iotMethod(void){
```

```
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;
  tm.display(0,0);
  tm.display(1,0);
  tm.display(2,0);
  tm.display(3,0);

//  digitalWrite(STEPPER_DIR, setdir);
//  digitalWrite(STEPPER_PULL, HIGH);
//  delayMicroseconds(stepper_delay);
//  digitalWrite(STEPPER_PULL, LOW);
//  step_count = step_count + 1;

  while(Serial.available()){
    incoming_string = Serial.readString();
    string_ready = true;
    };

  if(string_ready == true){
//    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
    }
    int year = String(strings[0]).toInt();
    int visitors = String(strings[1]).toInt();
    if(year == 0 || visitors == 0){
      return false;
      }
    String oled_vis = String(visitors);
    if(visitors<100){
      oled_vis = "00" + String(visitors);
      }else if(visitors<1000){
        oled_vis = "0" + String(visitors);
        }
```

```
    displayTextOLED(oled_vis);
    lcd1.clear();
    lcd1.setCursor(6,0);
    lcd1.print("YEAR");
    lcd1.setCursor(6,1);
    lcd1.print(String(year));
    long stepsy = map(visitors, 0, 6000, 0, 160000);
    stepForward(stepsy);
    stepBackward(stepsy);
    reverseMotor();
//    Serial.print("Year: " + String(year) + "\tVisitors: " + String(visitors) +
"\n\n");
//    delay(1000);
//    digitalWrite(STEPPER_DIR, setdir);
//    digitalWrite(STEPPER_PULL, HIGH);
//    delayMicroseconds(stepper_delay);
//    digitalWrite(STEPPER_PULL, LOW);
//    delayMicroseconds(stepper_delay);
    };
    return true;
  }

void sdBackup(void){
  String incoming_string = "";
  boolean string_ready = false;
  int visitors = 0;

  // re-open the file for reading:
  myFile = SD.open("data.txt");
  if (myFile) {
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      incoming_string = myFile.readStringUntil('\n');
      string_ready = true;
      if(string_ready == true){
//    Serial.println("Received String: ");
//    visitors = incoming_string.toInt();
    char *strings[2]; // an array of pointers to the pieces of the above array
after strtok()
    char *ptr = NULL;
    byte index = 0;
    ptr = strtok(incoming_string.c_str(), ",");  // parse comma-delimited string
as array
    while (ptr != NULL)
    {
```

```
      strings[index] = ptr;
      index++;
      ptr = strtok(NULL, ",");
    }
    int year = String(strings[0]).toInt();
    int visitors = String(strings[1]).toInt();
    String oled_vis = String(visitors);
    if(visitors<100){
      oled_vis = "00" + String(visitors);
      }else if(visitors<1000){
        oled_vis = "0" + String(visitors);
        }
    displayTextOLED(oled_vis);
    lcd1.clear();
    lcd1.setCursor(6,0);
    lcd1.print("YEAR");
    lcd1.setCursor(6,1);
    lcd1.print(String(year));
    long stepsy = map(visitors, 0, 6000, 0, 160000);
    stepForward(stepsy);
    delay(1000*10); //wait 10 seconds
    stepBackward(stepsy);
    delay(3000);
    reverseMotor();
      }
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
  }

void reverseMotor(){
  setdir = !setdir;
}

void stepForward(long steps){
  for(long i = 0; i<steps; i++){
  digitalWrite(STEPPER_DIR, setdir);
  digitalWrite(STEPPER_PULL, HIGH);
  delayMicroseconds(stepper_delay);
  digitalWrite(STEPPER_PULL, LOW);
  delayMicroseconds(stepper_delay);
```

```
  }
  }

void stepBackward(long steps){
  reverseMotor();
  for(long i = 0; i<steps; i++){
  digitalWrite(STEPPER_DIR, setdir);
  digitalWrite(STEPPER_PULL, HIGH);
  delayMicroseconds(stepper_delay);
  digitalWrite(STEPPER_PULL, LOW);
  delayMicroseconds(stepper_delay);
  }
  }
```

sample_shift_reg.ino

```
// This program is used to test the shift registers

#define DS_PIN 8
#define SHCP_PIN 9
#define STCP_PIN 10

const int register_count = 1;

void setup() {
  pinMode(DS_PIN, OUTPUT);
  pinMode(SHCP_PIN, OUTPUT);
  pinMode(STCP_PIN, OUTPUT);
  writereg();
}

boolean registers[8*register_count];

void writereg(void){
  digitalWrite(SHCP_PIN, LOW);
  for (int i = 8*register_count; i>=0; i--){
    digitalWrite(STCP_PIN, LOW);
    digitalWrite(DS_PIN, registers[i]);
    digitalWrite(STCP_PIN, HIGH);
    };
    digitalWrite(SHCP_PIN, HIGH);
  }
```

```
void loop() {
  for(int i = 0; i<8; i++){
  registers[0] = HIGH;
  writereg();
  }

}
```

sd_card_reader.ino

```
// This software tests the SD card failsafe feature

/*  Reading csv file from SD card example for:
https://github.com/michalmonday/CSV-Parser-for-Arduino
    It's based on the following "DumpFile" example:
https://www.arduino.cc/en/Tutorial/DumpFile

    This example is reading "file.csv" from sd card and prints the parsed values
from it.
    Example contents of the "file.csv" (used to test this example) were:

      column_1,column_2\n
      1,2\n
      11,22\n
      111,333\n
    This example requires SPI connection to connect with SD card as described at
the "DumpFile" link.
    */

#include <CSV_Parser.h>

#include <SPI.h>
#include <SD.h>

const int chipSelect = 4;
#define csv_file_name     "/TEST.CSV"

void setup() {
  Serial.begin(9600);
  delay(5000);

  Serial.print("Initializing SD card...");
```

```
  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");

    // don't do anything more:
    while (1);
  }
  Serial.println("card initialized.");
  SD.begin(chipSelect);


  CSV_Parser cp(/*format*/ "dd", /*has_header*/ false, /*delimiter*/ ',');

  // The line below (readSDfile) wouldn't work if SD.begin wasn't called before.
  // readSDfile can be used as conditional, it returns 'false' if the file does
not exist.
  if (cp.readSDfile(csv_file_name)) {
    int16_t *column_2 = (int16_t*)cp[0];

    if (column_2) {
      for(int row = 0; row < cp.getRowsCount(); row++) {
        Serial.print("row = ");
        Serial.print(row, DEC);
        Serial.print(", column_2 = ");
        Serial.println(column_2[row], DEC);
      }
    } else {
      Serial.println("ERROR: At least 1 of the columns was not found, something
went wrong.");
    }

    // output parsed values (allows to check that the file was parsed correctly)
    cp.print(); // assumes that "Serial.begin()" was called before (otherwise it
won't work)

  } else {
    Serial.println("ERROR: File called " + String(csv_file_name) + " does not
exist...");
  }
}

void loop() {

}
```

traffic_test.ino

```
// This program is used to test the traffic-determination algorithm
// Use this to check if the correct color LEDs are turning on

#define DS_PIN 8
#define SHCP_PIN 9
#define STCP_PIN 10

int RED = 0;
int YELLOW = 1;
int GREEN = 2;

const int register_count = 5;

int HOURS[12][3] = {{12,24,37},
                    {0,13,25},
                    {1,14,26},
                    {2,15,27},
                    {3,16,28},
                    {4,5,17},
                    {6,18,29},
                    {7,19,30},
                    {8,23,31},
                    {9,20,32},
                    {10,21,33},
                    {11,22,34}};

float sample_data[12] = {
    10,
    10,
    3,
    2,
    1,
    20,
    20,
    35,
    35,
    50,
    55,
    55
};
```

```
boolean registers[8*register_count];

void setup() {
  Serial.begin(250000);
  pinMode(DS_PIN, OUTPUT);
  pinMode(SHCP_PIN, OUTPUT);
  pinMode(STCP_PIN, OUTPUT);
  for(int hour = 0; hour<8*register_count; hour++){
    registers[hour] = LOW;
  };
  writereg();
}

void writereg(void){
  digitalWrite(STCP_PIN, LOW);
  for (int i = 8*register_count; i>=0; i--){
    digitalWrite(SHCP_PIN, LOW);
    digitalWrite(DS_PIN, registers[i]);
    digitalWrite(SHCP_PIN, HIGH);
    };
    digitalWrite(STCP_PIN, HIGH);
  }


void loop() {
  for(int i = 0; i < 12; i++){
    registers[HOURS[i][trafficLevel(i)]] = HIGH;
    writereg();
  };


}


 float largest(float arr[], int n)
{
    int i;

    // Initialize maximum element
    int max = arr[0];

    // Traverse array elements from second and
    // compare every element with current max
    for (i = 1; i < n; i++)
        if (arr[i] > max)
```

```
            max = arr[i];

    return max;
}

//This function normalizes the traffic data for each hour and converts to analog
output (0 to 255)
int trafficLevel(int hour){
  float current_visits = sample_data[hour];
  float largest_num = float(largest(sample_data, 12));
    float weight = current_visits/largest_num;
    weight = 255*weight;
    if(weight>255*2/3){
      return RED;
      }else if(weight<=255*2/3 && weight>=255*1/3){
        return YELLOW;
        }else{
          return GREEN;
          }
}
```

ultrasonic_system_comms.ino

```
// This program tests the ultrasonic sensor IoT system

#include <SoftwareSerial.h>

//CONNECT RX of ESP to TX of Uno, TX of ESP to RX of Uno
#define STATUS_LED  7

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(STATUS_LED, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:


}
```

```
void recordVisitor(void){
  delay(1000);
  Serial.print(1); // print 1 to Serial to trigger ESP to send POST request
  delay(2000);
  Serial.print("\n");
  while(!Serial.available()){
  delay(1000);
  }
  String receiver = Serial.readString(); // read response from Serial
  Serial.println(receiver);
  if(receiver != "SUCCESS!"){
    digitalWrite(STATUS_LED, HIGH); // if the ESP does not successfully send the
data, then turn on error status LED
    }
  else{
    digitalWrite(STATUS_LED, LOW);
    }
  delay(2000);
  }
```

us_system_arduino.ino

```
// This software is used in the ultrasonic sensor system

// ---------------------------------------------------------------- //
// OEDK Visitor Counting Ultrasonic Sensor System
// Ibrahim Al-Akash
// Using HC-SR04 Module
// Tested on 17 March 2022
// ---------------------------------------------------------------- //

// First Sensor
int echoPin = 8; // Attach pin D2 Arduino to pin Echo of HC-SR04
int trigPin = 9; // Attach pin D3 Arduino to pin Trig of HC-SR04

//Second Sensor
int echoPin2 = 4; // Attach pin D4 Arduino to pin Echo of HC-SR04
int trigPin2 = 5; // Attach pin D5 Arduino to pin Trig of HC-SR04

//Status LED
int ledPin = 7; // Attach pin D7 Arduino to cathode of LED

// defines variables
```

```
long duration; // Variable for the duration of sound wave travel
int distance1; // Variable for the distance measurement
int distance2; // Variable for the distance measurement
int baseline = 500; // Variable for the baseline distance measurement

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin2, INPUT); // Sets the echoPin as an INPUT
  pinMode(ledPin, OUTPUT); // Sets the ledPin as an OUTPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of
baudrate speed
}
void loop() {

  distance1 = ultrasonicSensor(trigPin, echoPin);
  if(distance1 < baseline){ // If sensor 1 is triggered, check sensor 2
    delay(100); // Allow "cool down" for sound waves of sensor 1 to dissipate
    distance2 = ultrasonicSensor(trigPin2, echoPin2);
    if(distance2 < baseline){ // Sensor 2 is triggered
      while(distance2 < baseline){
        distance2 = ultrasonicSensor(trigPin2, echoPin2);
        delay(1); // Wait until visitor finishes walking past sensor
        }
        alertESP(); // Signal to the ESP to send HTTP POST request
        digitalWrite(ledPin, HIGH);
        delay(500);
        digitalWrite(ledPin, LOW);
      }
    delay(100);
    }

}

// This function runs the code for the sensor to measure distance
float ultrasonicSensor(int trigpin, int echopin){
  // Clears the trigPin condition
  digitalWrite(trigpin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
```

```
  duration = pulseIn(echopin, HIGH);
  digitalWrite(trigpin, LOW);
  // Calculating the distance
  float dist = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and
back)
  return dist;
  }

// This function alerts the ESP that a visitor has passed by
// so it can initiate the HTTP POST request to database
void alertESP(){
  Serial.print(1);
  }

// This function is the failsafe for the IoT system
void checkPost(void){
  if(Serial.available()){
    String post_status = Serial.readString(); // Read the message from the ESP
after sending the HTTP POST request
    if(post_status != "SUCCESS"){
      digitalWrite(ledPin, HIGH); // If the message does not read success, turn
on the error status LED
      }else{
        digitalWrite(ledPin, LOW); // If the message reads success, turn off the
error status LED
        }
    }
  }
```

us_system_esp.ino

```
// This is the software for the ultrasonic sensor system
// Upload this to the ESP, NOT the Arduino

#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

char ssid[] = "WAVLINKN";       // your network SSID (name)
char password[] = "EngI120sp22";  // your network key

WiFiClient client;
```

```
#define TEST_HOST "192.168.10.186"

void setup() {
Serial.begin(9600);
// Connect to the WiFI
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  // Attempt to connect to Wifi network:
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password); // if wifi has password, then use
WiFi.begin(ssid,password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  IPAddress ip = WiFi.localIP();
  Serial.println(ip);
  delay(5000);
}

void recordVisitor(void) {

  // Opening connection to server (Use 80 as port if HTTP)
  if (!client.connect(TEST_HOST, 80))
  {
    Serial.println(F("Connection failed"));
    Serial.print("SD");
    return;
  }

  // give the esp a breather
  yield();

  // Send HTTP request
  client.print(F("GET "));
  // This is the second half of a request (everything that comes after the base
URL)
  client.print("/sensor_triggered"); // %2C == ,
```

```
  client.println(F(" HTTP/1.1"));

  //Headers
  client.print(F("Host: "));
  client.println(TEST_HOST);

  client.println(F("Cache-Control: no-cache"));

  if (client.println() == 0)
  {
    Serial.println(F("Failed to send request"));
    return;
  }
  //delay(100);
  // Check HTTP status
  char status[32] = {0};
  client.readBytesUntil('\r', status, sizeof(status));
  if (strcmp(status, "HTTP/1.1 200 OK") != 0)
  {
    Serial.print(F("Unexpected response: "));
    Serial.println(status);
    Serial.print("SD");
    return;
  }

  // Skip HTTP headers
  char endOfHeaders[] = "\r\n\r\n";
  if (!client.find(endOfHeaders))
  {
    Serial.println(F("Invalid response"));
    Serial.print("SD");
    return;
  }

  // This is probably not needed for most, but I had issues
  // with the Tindie api where sometimes there were random
  // characters coming back before the body of the response.
  // This will cause no hard to leave it in
  // peek() will look at the character, but not take it off the queue
  while (client.available() && client.peek() != '{')
  {
    char c = 0;
    client.readBytes(&c, 1);
    Serial.print(c);
    Serial.println("BAD");
```

```
  }

  //  // While the client is still availble read each
  //  // byte and print to the serial monitor
  //  while (client.available()) {
  //    char c = 0;
  //    client.readBytes(&c, 1);
  //    Serial.print(c);
  //  }

  //Use the ArduinoJson Assistant to calculate this:

  //StaticJsonDocument<192> doc;
//  DynamicJsonDocument doc(192); //For ESP32/ESP8266 you'll mainly use dynamic.

//  DeserializationError error = deserializeJson(doc, client);\

  // Stream& input;

  StaticJsonDocument<48> doc;

  DeserializationError error = deserializeJson(doc, client);

  if (error) {
    Serial.println(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    Serial.print("SD");
    return;
  } else {
    String post_status = doc["Status"]; // 5819
    Serial.print(post_status);

  }


}

void deleteVisitor(void) {

  // Opening connection to server (Use 80 as port if HTTP)
  if (!client.connect(TEST_HOST, 80))
  {
    Serial.println(F("Connection failed"));
    Serial.print("SD");
    return;
```

```
  }

  // give the esp a breather
  yield();

  // Send HTTP request
  client.print(F("GET "));
  // This is the second half of a request (everything that comes after the base
URL)
  client.print("/sensor_delete"); // %2C == ,
  client.println(F(" HTTP/1.1"));

  //Headers
  client.print(F("Host: "));
  client.println(TEST_HOST);

  client.println(F("Cache-Control: no-cache"));

  if (client.println() == 0)
  {
    Serial.println(F("Failed to send request"));
    return;
  }
  //delay(100);
  // Check HTTP status
  char status[32] = {0};
  client.readBytesUntil('\r', status, sizeof(status));
  if (strcmp(status, "HTTP/1.1 200 OK") != 0)
  {
    Serial.print(F("Unexpected response: "));
    Serial.println(status);
    Serial.print("SD");
    return;
  }

  // Skip HTTP headers
  char endOfHeaders[] = "\r\n\r\n";
  if (!client.find(endOfHeaders))
  {
    Serial.println(F("Invalid response"));
    Serial.print("SD");
    return;
  }

  // This is probably not needed for most, but I had issues
```

```
  // with the Tindie api where sometimes there were random
  // characters coming back before the body of the response.
  // This will cause no hard to leave it in
  // peek() will look at the character, but not take it off the queue
  while (client.available() && client.peek() != '{')
  {
    char c = 0;
    client.readBytes(&c, 1);
    Serial.print(c);
    Serial.println("BAD");
  }

  //  // While the client is still availble read each
  //  // byte and print to the serial monitor
  //  while (client.available()) {
  //    char c = 0;
  //    client.readBytes(&c, 1);
  //    Serial.print(c);
  //  }

  //Use the ArduinoJson Assistant to calculate this:

  //StaticJsonDocument<192> doc;
//  DynamicJsonDocument doc(192); //For ESP32/ESP8266 you'll mainly use dynamic.

//  DeserializationError error = deserializeJson(doc, client);\

  // Stream& input;

  StaticJsonDocument<48> doc;

  DeserializationError error = deserializeJson(doc, client);

  if (error) {
    Serial.println(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    Serial.print("SD");
    return;
  } else {
    String post_status = doc["Status"]; // 5819
    Serial.print(post_status);

  }
```

```
}

void loop() {
if(Serial.available()){
  int reading = Serial.readString().toInt();
  if(reading>0){
    recordVisitor();
    }else if(reading < 0){
      deleteVisitor();
      }
  }
}
```

vanilla_us_system.ino

```
// This software is used to test the ultrasonic sensor system's IoT
// capability and accuracy in the OEDK

// ---------------------------------------------------------------- //
// OEDK Visitor Counting Ultrasonic Sensor System
// Ibrahim Al-Akash
// Using HC-SR04 Module
// Tested on 17 March 2022
// ---------------------------------------------------------------- //

// First Sensor
int echoPin = 2; // Attach pin D2 Arduino to pin Echo of HC-SR04
int trigPin = 3; // Attach pin D3 Arduino to pin Trig of HC-SR04

//Second Sensor
int echoPin2 = 4; // Attach pin D4 Arduino to pin Echo of HC-SR04
int trigPin2 = 5; // Attach pin D5 Arduino to pin Trig of HC-SR04

//Status LED
int ledPin = 7; // Attach pin D7 Arduino to cathode of LED

// defines variables
long duration; // Variable for the duration of sound wave travel
int distance1; // Variable for the distance measurement
int distance2; // Variable for the distance measurement
int baseline; // Variable for the baseline distance measurement

void setup() {
```

```
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin2, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of
baudrate speed
}
void loop() {

  distance1 = ultrasonicSensor(trigPin, echoPin);
  if(distance1 < baseline){ // If sensor 1 is triggered, check sensor 2
    delay(100); // Allow "cool down" for sound waves of sensor 1 to dissipate
    distance2 = ultrasonicSensor(trigPin2, echoPin2);
    if(distance2 < baseline){ // Sensor 2 is triggered
      while(distance2 < baseline){
        distance2 = ultrasonicSensor(trigPin2, echoPin2);
        delay(1); // Wait until visitor finishes walking past sensor
        }
        alertESP(); // Signal to the ESP to send HTTP POST request
      }
    delay(100);
    checkPost();
    }

}

// This function runs the code for the sensor to measure distance
float ultrasonicSensor(int trigpin, int echopin){
  // Clears the trigPin condition
  digitalWrite(trigpin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echopin, HIGH);
  digitalWrite(trigpin, LOW);
  // Calculating the distance
  float dist = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and
back)
  return dist;
  }

// This function alerts the ESP that a visitor has passed by
```

```
// so it can initiate the HTTP POST request to database
void alertESP(){
  Serial.print(1);
  }

void checkPost(void){
  if(Serial.available()){
    String post_status = Serial.readString();
    if(post_status != "SUCCESS!"){
      digitalWrite(STATUS_LED, HIGH);
      }else{
        digitalWrite(STATUS_LED, LOW);
        }
    }
  }
```

y-axis.ino

```
// This software runs the code for the 7-segment displays on the y-axis

#include <TM1637.h>

int CLK = 3;
int DIO = 2;
int DIO2 = 4;
int CLK2 = 5;
int DIO3 = 7;
int CLK3 = 8;
int DIO4 = 9;
int CLK4 = 10;
int DIO5 = 12;
int CLK5 = 13;

TM1637 tm(CLK,DIO);
TM1637 tm2(CLK2,DIO2);
TM1637 tm3(CLK3,DIO3);
TM1637 tm4(CLK4,DIO4);
TM1637 tm5(CLK5,DIO5);

void setup() {
  // put your setup code here, to run once:
  tm.init();
  tm2.init();
```

```
  tm3.init();
  tm4.init();
  tm5.init();
  tm.set(2);
  tm2.set(2);
  tm3.set(2);
  tm4.set(2);
  tm5.set(2);
}

void loop() {
  // put your main code here, to run repeatedly:
  tm.display(0,4); //4500
  tm.display(1,5);
  tm.display(2,0);
  tm.display(3,0);

  tm2.display(0,6); //6000
  tm2.display(1,0);
  tm2.display(2,0);
  tm2.display(3,0);

  tm3.display(0,0); //0000
  tm3.display(1,0);
  tm3.display(2,0);
  tm3.display(3,0);

  tm4.display(0,1); //1500
  tm4.display(1,5);
  tm4.display(2,0);
  tm4.display(3,0);

  tm5.display(0,3); //3000
  tm5.display(1,0);
  tm5.display(2,0);
  tm5.display(3,0);
}
```

Backend Code

Overview.md

```
 # Backend
 There are 3 functions:
 ### 1) "get_visitors_between"
 This function takes inputs from the GET request of an interval between one year
and another to return the number of visitors.

 SAMPLE GET REQUEST:
http://localhost:4800/get_visitors_between/?time_start=2015&time_end=2021
 RESPONSE: 25519

 ### 2) "year"
 This function takes input from the GET request as a year and returns the number
of visitors that year

 SAMPLE GET REQUEST: http://localhost:4800/year?year=2021
 RESPONSE: 107

 ### 3) "sensor_triggered"
 This function records a user entering the OEDK once the ultrasonic sensor is
triggered. No inputs for this function.
 SAMPLE POST REQUEST: http://localhost:4800/sensor_triggered/
```

server.js

```
// To host the server, cd into this file's root folder
// Then, run "npm run devStart"
// Remember to check the computer's IP address and use that for API calls

const express = require('express');
const path = require('path');
const mysql = require("mysql");
var bodyParser = require('body-parser');
const { time } = require('console');

const connection = mysql.createConnection({
    host: "localhost",
    user: "root",
    //port: "3306",
    // password: "mytOUrnEysql2003$",
    password: "password",
    database: "oedk",
    timezone: 'utc'
});
```

```
connection.connect(function(error){
    if (error) throw error
    else console.log(`Connected to database successfully. Open port ${port}`);
});

const app = express();
const port = 80;
app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');

app.use(express.static(path.join(__dirname, '')));

makeQuery = function(query,vals){
    return new Promise(function(resolve,reject){
        if(vals){
            connection.query(
                query,
                vals,
                function(err,rows){
                    if(rows === undefined){
                        console.log(new Error(`Error, the query was not
successful - ${err}`));
                        reject(0);
                    }else{
                        resolve(rows);
                    }
                }
            )
        }else{
            connection.query(
                query,
                function(err,rows){
                    if(rows === undefined){
                        console.log(new Error(`Error, the query was not
successful - ${err}`));
                        reject(0);
                    }else{
                        resolve(rows);
                    }
                }
            )
        }

    })
}
```

```
// This function gets the number of visitors between two years
app.get('/get_visitors_between/', async function(req, res){

    time_start = req.query.time_start;
    time_end = req.query.time_end;
    query = `SELECT * FROM daily_visitors WHERE YEAR(date) BETWEEN ${time_start}
AND ${time_end}`;
    visitors = await makeQuery(query, "");

    amount = 0;
    for (point of visitors){
        amount = amount + point.visitor_count;
    }
    Amount = {
        "visitors": amount
    };
    res.json(Amount);
})

// This function tells the Arduino how many years to cycle through
app.get('/year_count/', async function(req, res){

    const todaysDate = new Date();
    const currentYear = todaysDate.getFullYear();
    amount = currentYear - 2015;
    Amount = {
        "year_count": amount
    };
    res.json(Amount);
})

// This function gets the number of visitors in a single year
app.get('/year/', async function(req, res){

    year = req.query.year;
    query = `SELECT * FROM daily_visitors WHERE YEAR(date) = '${year}'`;
    visitors = await makeQuery(query, "");

    amount = 0;
    for (point of visitors){
        amount = amount + point.visitor_count;
    }
    Amount = {
        "visitors": amount
```

```
    };
    res.json(Amount);
})

app.get('/time/',async function(req,res){

    const todaysDate = new Date();
    const currentYear = todaysDate.getHours;
    let timer = "AM";
    if(currentYear >= 11){
        timer = "PM";
    };
    Time = {
        "time":  timer
    };
    res.json(Time);

});

app.get('/hourly/', async function(req, res){
    let hours = [];
    for(let i = 1; i < 25; i++){
        query = `SELECT * FROM daily_visitors WHERE HOUR(date) = '${i}'`;
        visitors = await makeQuery(query, "");

        amount = 0;
        for (point of visitors){
            amount = amount + point.visitor_count;
        }
        hours[i-1] = amount;
    }
    Amount = {
        "12am": hours[0],
        "1am": hours[1],
        "2am": hours[2],
        "3am":  hours[3],
        "4am":  hours[4],
        "5am":  hours[5],
        "6am":  hours[6],
        "7am":  hours[7],
        "8am":  hours[8],
        "9am":  hours[9],
        "10am":  hours[10],
        "11am": hours[11],
        "12pm": hours[12],
```

```
        "1pm":  hours[13],
        "2pm":  hours[14],
        "3pm":  hours[15],
        "4pm":  hours[16],
        "5pm":  hours[17],
        "6pm":  hours[18],
        "7pm":  hours[19],
        "8pm":  hours[20],
        "9pm":  hours[21],
        "10pm": hours[22],
        "11pm": hours[23]
    };
    res.json(Amount);
})

// This function allows the user to update the active times table with the US
sensor
app.get('/sensor_triggered', async function(req, res){

    query = `INSERT INTO active_times SET ?`;
    vals = {
        visitor_tracker: 1
    };

    visitors = await makeQuery(query, vals);

    res.json({
        "Status": "SUCCESS"
    });
})

app.listen(port);
console.log(`Listening on port ${port}...`);
```

oedk_visitors.sql

```
CREATE DATABASE  IF NOT EXISTS `oedk` /*!40100 DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `oedk`;
-- MySQL dump 10.13  Distrib 8.0.27, for Win64 (x86_64)
--
-- Host: 127.0.0.1    Database: oedk
-- ------------------------------------------------------
-- Server version 8.0.27
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `active times`
--

DROP TABLE IF EXISTS `active times`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `active times` (
  `recording_id` int NOT NULL AUTO_INCREMENT,
  `timestamp` datetime NOT NULL,
  `hour` int NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`recording_id`),
  UNIQUE KEY `recording_id_UNIQUE` (`recording_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `active times`
--

LOCK TABLES `active times` WRITE;
/*!40000 ALTER TABLE `active times` DISABLE KEYS */;
/*!40000 ALTER TABLE `active times` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `daily_visitors`
--

DROP TABLE IF EXISTS `daily_visitors`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `daily_visitors` (
  `daily_id` int NOT NULL AUTO_INCREMENT,
  `date` datetime NOT NULL,
  `visitor_count` int NOT NULL,
  PRIMARY KEY (`daily_id`),
  UNIQUE KEY `iddaily_visitors_UNIQUE` (`daily_id`)
) ENGINE=InnoDB AUTO_INCREMENT=841 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `daily_visitors`
--

LOCK TABLES `daily_visitors` WRITE;
/*!40000 ALTER TABLE `daily_visitors` DISABLE KEYS */;
INSERT INTO `daily_visitors` VALUES (1,'2015-07-13 13:54:23',91),(2,'2015-07-13
19:56:43',66),(3,'2015-07-21 20:27:04',109),(4,'2015-07-22 15:13:50',3),(5,'2015-
07-22 20:21:38',1),(6,'2015-07-27 16:15:03',121),(7,'2015-08-03
18:57:28',126),(8,'2015-08-04 13:39:22',5),(9,'2015-08-05 17:29:25',5),(10,'2015-
08-05 17:30:41',2),(11,'2015-08-06 21:22:59',18),(12,'2015-08-06
21:37:07',30),(13,'2015-08-10 16:11:19',128),(14,'2015-08-11
19:00:12',8),(15,'2015-08-12 17:49:39',3),(16,'2015-08-17
18:16:23',100),(17,'2015-08-17 20:36:59',20),(18,'2015-08-21
21:07:12',2),(19,'2015-08-24 16:12:34',112),(20,'2015-08-24
16:17:09',31),(21,'2015-08-31 20:22:52',25),(22,'2015-09-09
17:54:06',2),(23,'2015-09-14 17:26:03',45),(24,'2015-09-14
19:01:40',29),(25,'2015-09-14 23:29:21',1),(26,'2015-09-15
21:07:00',15),(27,'2015-09-20 17:40:30',2),(28,'2015-09-21
20:13:23',26),(29,'2015-09-22 16:07:35',8),(30,'2015-09-22
16:09:09',7),(31,'2015-09-22 16:11:20',7),(32,'2015-09-23 19:14:37',5),(33,'2015-
09-23 20:29:54',3),(34,'2015-09-28 19:35:00',63),(35,'2015-09-28
19:36:36',20),(36,'2015-09-28 23:43:30',12),(37,'2015-09-30
19:13:18',1),(38,'2015-10-01 21:44:53',45),(39,'2015-10-06
17:35:40',45),(40,'2015-10-07 18:48:48',2),(41,'2015-10-07
18:49:43',4),(42,'2015-10-09 15:26:58',14),(43,'2015-10-09
15:28:45',125),(44,'2015-10-12 16:14:44',15),(45,'2015-10-12
16:19:25',1),(46,'2015-10-19 20:04:39',5),(47,'2015-10-20 17:10:51',8),(48,'2015-
10-20 22:11:33',5),(49,'2015-10-21 17:04:46',26),(50,'2015-10-21
17:11:14',1),(51,'2015-10-21 22:25:11',50),(52,'2015-10-22
16:55:10',30),(53,'2015-10-26 15:28:07',2),(54,'2015-10-26
18:06:31',30),(55,'2015-10-26 18:55:39',41),(56,'2015-10-26
19:07:58',3),(57,'2015-10-26 19:09:42',39),(58,'2015-10-30
17:25:32',7),(59,'2015-11-04 14:35:51',10),(60,'2015-11-04
```

```
15:15:17',22),(61,'2015-11-04 20:44:21',15),(62,'2015-11-05
20:38:08',3),(63,'2015-11-06 14:35:03',45),(64,'2015-11-09
16:23:26',54),(65,'2015-11-11 19:11:19',25),(66,'2015-11-12
16:41:11',38),(67,'2015-11-12 16:42:14',45),(68,'2015-11-16
15:11:50',39),(69,'2015-11-16 15:39:04',5),(70,'2015-11-16
16:07:39',1),(71,'2015-11-16 16:35:12',4),(72,'2015-11-16 16:45:56',3),(73,'2015-
11-16 17:23:56',25),(74,'2015-11-17 16:06:16',3),(75,'2015-11-20
14:49:36',2),(76,'2015-11-23 17:29:22',191),(77,'2015-11-23
18:04:51',45),(78,'2015-11-30 15:38:37',30),(79,'2015-11-30
20:06:14',13),(80,'2015-12-01 15:01:49',2),(81,'2015-12-02
21:08:27',98),(82,'2015-12-07 12:32:40',3),(83,'2015-12-11
16:35:12',20),(84,'2015-12-11 16:37:40',25),(85,'2015-12-11
17:23:39',30),(86,'2015-12-17 22:43:24',6),(87,'2016-01-04
20:26:12',18),(88,'2016-01-04 22:07:18',1),(89,'2016-01-06
15:59:52',44),(90,'2016-01-06 20:35:29',1),(91,'2016-01-06
23:10:22',60),(92,'2016-01-07 15:57:10',20),(93,'2016-01-08
19:50:26',1),(94,'2016-01-14 14:53:40',12),(95,'2016-01-14
15:23:22',3),(96,'2016-01-21 17:03:46',20),(97,'2016-01-25
21:46:07',1),(98,'2016-01-26 16:11:36',1),(99,'2016-01-26
17:12:24',9),(100,'2016-01-29 19:04:47',2),(101,'2016-02-01
18:43:46',3),(102,'2016-02-02 15:01:51',2),(103,'2016-02-02
15:22:53',13),(104,'2016-02-04 15:05:14',3),(105,'2016-02-04
15:31:18',4),(106,'2016-02-04 19:13:28',2),(107,'2016-02-05
20:08:47',3),(108,'2016-02-08 17:15:51',25),(109,'2016-02-08
17:18:43',1),(110,'2016-02-10 19:43:39',1),(111,'2016-02-11
16:47:17',20),(112,'2016-02-11 20:45:38',2),(113,'2016-02-12
21:47:55',2),(114,'2016-02-12 22:02:14',2),(115,'2016-02-15
15:49:26',149),(116,'2016-02-16 19:24:36',1),(117,'2016-02-17
20:03:45',5),(118,'2016-02-19 14:25:16',2),(119,'2016-02-19
18:39:13',1),(120,'2016-02-19 20:18:32',1),(121,'2016-02-22
20:00:59',2),(122,'2016-02-24 16:28:49',12),(123,'2016-02-24
20:07:52',15),(124,'2016-02-25 19:42:04',7),(125,'2016-02-27
03:04:10',10),(126,'2016-02-29 15:09:37',4),(127,'2016-03-02
16:49:54',12),(128,'2016-03-03 13:29:03',8),(129,'2016-03-03
21:40:24',250),(130,'2016-03-04 13:54:35',35),(131,'2016-03-08
13:54:32',15),(132,'2016-03-08 16:15:35',129),(133,'2016-03-08
16:23:21',1),(134,'2016-03-10 02:14:58',2),(135,'2016-03-11
15:37:30',15),(136,'2016-03-14 19:51:46',290),(137,'2016-03-14
22:50:30',3),(138,'2016-03-15 21:01:55',1),(139,'2016-03-18
15:39:45',35),(140,'2016-03-18 20:18:46',244),(141,'2016-03-21
15:52:52',53),(142,'2016-03-22 15:08:30',1),(143,'2016-03-22
15:13:08',4),(144,'2016-03-22 17:22:31',4),(145,'2016-03-22
19:40:35',6),(146,'2016-03-22 19:44:01',4),(147,'2016-03-22
19:46:02',2),(148,'2016-03-23 18:50:00',3),(149,'2016-03-24
15:08:52',10),(150,'2016-03-24 16:36:01',30),(151,'2016-03-24
```

```
20:57:15',2),(152,'2016-03-25 14:57:47',148),(153,'2016-03-25
21:58:09',1),(154,'2016-03-28 00:57:25',3),(155,'2016-03-31
19:32:37',114),(156,'2016-04-01 17:14:44',2),(157,'2016-04-04
06:20:15',20),(158,'2016-04-04 17:37:40',43),(159,'2016-04-07
18:33:17',30),(160,'2016-04-07 18:42:47',15),(161,'2016-04-08
17:17:39',1),(162,'2016-04-08 18:16:15',3),(163,'2016-04-08
20:46:45',4),(164,'2016-04-08 21:32:44',1),(165,'2016-04-08
21:38:46',1),(166,'2016-04-09 23:57:04',30),(167,'2016-04-11
20:23:05',25),(168,'2016-04-12 18:31:24',20),(169,'2016-04-13
17:56:20',2),(170,'2016-04-13 20:04:39',21),(171,'2016-04-13
21:19:37',5),(172,'2016-04-14 00:47:14',60),(173,'2016-04-15
17:33:23',10),(174,'2016-04-26 16:02:54',32),(175,'2016-04-26
16:39:46',7),(176,'2016-05-02 16:04:37',3),(177,'2016-05-04
02:08:54',2),(178,'2016-05-05 15:39:50',20),(179,'2016-05-06
14:34:12',1),(180,'2016-05-06 16:09:19',2),(181,'2016-05-11
15:26:30',3),(182,'2016-05-16 16:41:24',9),(183,'2016-05-16
19:11:46',14),(184,'2016-05-18 21:34:57',30),(185,'2016-05-20
18:02:47',3),(186,'2016-05-20 20:01:48',50),(187,'2016-05-22
23:19:08',5),(188,'2016-05-23 16:35:48',13),(189,'2016-05-24
18:16:13',6),(190,'2016-05-31 16:07:55',3),(191,'2016-05-31
17:30:03',8),(192,'2016-06-02 18:07:39',40),(193,'2016-06-03
19:54:40',9),(194,'2016-06-06 00:12:34',2),(195,'2016-06-06
15:44:25',21),(196,'2016-06-07 17:39:02',71),(197,'2016-06-07
20:16:27',1),(198,'2016-06-13 16:46:03',54),(199,'2016-06-20
16:21:36',20),(200,'2016-06-20 16:24:03',14),(201,'2016-06-20
17:24:31',60),(202,'2016-06-21 22:03:28',21),(203,'2016-06-28
16:53:12',74),(204,'2016-06-28 20:20:52',25),(205,'2016-06-29
17:11:13',5),(206,'2016-06-30 17:08:50',40),(207,'2016-07-11
16:42:49',91),(208,'2016-07-14 02:09:02',7),(209,'2016-07-18
16:25:46',10),(210,'2016-07-21 17:59:00',8),(211,'2016-07-25
17:56:22',81),(212,'2016-07-25 17:58:08',128),(213,'2016-07-26
19:47:27',10),(214,'2016-07-26 21:18:57',3),(215,'2016-07-27
04:17:44',5),(216,'2016-07-27 16:21:56',20),(217,'2016-07-28
15:38:01',12),(218,'2016-08-01 15:28:15',73),(219,'2016-08-02
17:46:00',15),(220,'2016-08-05 15:19:52',165),(221,'2016-08-05
15:21:06',59),(222,'2016-08-06 22:36:47',3),(223,'2016-08-10
17:01:31',3),(224,'2016-08-12 15:27:06',109),(225,'2016-08-12
18:46:04',30),(226,'2016-08-15 17:31:55',67),(227,'2016-08-17
16:26:16',4),(228,'2016-08-17 18:55:53',6),(229,'2016-08-19
17:53:13',30),(230,'2016-08-19 18:42:57',3),(231,'2016-08-23
19:16:45',3),(232,'2016-08-24 20:26:36',26),(233,'2016-08-29
18:43:46',25),(234,'2016-08-30 21:12:29',3),(235,'2016-08-31
03:25:39',6),(236,'2016-08-31 05:04:08',1),(237,'2016-09-01
04:45:37',3),(238,'2016-09-02 21:58:42',2),(239,'2016-09-06
18:36:14',2),(240,'2016-09-06 22:00:39',5),(241,'2016-09-07
```

```
21:22:15',4),(242,'2016-09-08 21:20:42',4),(243,'2016-09-09
16:31:07',25),(244,'2016-09-12 16:04:18',20),(245,'2016-09-12
20:50:02',3),(246,'2016-09-13 21:37:21',30),(247,'2016-09-14
15:14:08',4),(248,'2016-09-14 15:49:03',4),(249,'2016-09-15
22:00:56',2),(250,'2016-09-16 15:17:42',9),(251,'2016-09-16
15:19:52',12),(252,'2016-09-27 20:01:57',15),(253,'2016-09-28
16:07:12',35),(254,'2016-09-29 15:38:14',1),(255,'2016-09-29
16:36:24',2),(256,'2016-09-30 15:12:40',61),(257,'2016-10-03
18:25:54',2),(258,'2016-10-04 18:23:39',10),(259,'2016-10-04
18:52:01',5),(260,'2016-10-07 15:38:24',83),(261,'2016-10-07
19:49:47',50),(262,'2016-10-10 17:19:39',12),(263,'2016-10-12
20:57:08',80),(264,'2016-10-13 16:45:21',38),(265,'2016-10-13
17:09:25',3),(266,'2016-10-13 21:53:04',4),(267,'2016-10-14
17:59:32',44),(268,'2016-10-14 20:18:45',130),(269,'2016-10-17
18:56:00',2),(270,'2016-10-18 15:49:01',2),(271,'2016-10-19
22:21:33',1),(272,'2016-10-24 18:22:24',9),(273,'2016-10-26
20:23:48',3),(274,'2016-10-30 22:06:14',18),(275,'2016-10-31
19:08:24',4),(276,'2016-11-01 21:49:40',18),(277,'2016-11-02
21:34:44',1),(278,'2016-11-03 19:26:22',80),(279,'2016-11-03
20:33:02',15),(280,'2016-11-04 17:28:10',5),(281,'2016-11-07
18:36:43',25),(282,'2016-11-11 15:25:31',6),(283,'2016-11-14
15:44:36',30),(284,'2016-11-14 21:21:10',6),(285,'2016-11-15
15:21:12',13),(286,'2016-11-17 18:14:32',18),(287,'2016-11-18
19:05:17',25),(288,'2016-11-21 17:24:35',289),(289,'2016-11-22
15:31:05',2),(290,'2016-11-22 17:00:58',13),(291,'2016-11-28
16:51:51',11),(292,'2016-11-30 13:40:03',10),(293,'2016-12-01
20:25:07',2),(294,'2016-12-05 18:44:35',2),(295,'2016-12-10
13:46:12',5),(296,'2016-12-21 21:52:50',12),(297,'2016-12-22
22:45:39',2),(298,'2016-12-23 17:56:53',2),(299,'2017-01-02
22:31:12',20),(300,'2017-01-06 18:38:25',1),(301,'2017-01-08
19:51:59',3),(302,'2017-01-09 13:10:10',4),(303,'2017-01-09
18:06:44',17),(304,'2017-01-09 18:08:02',7),(305,'2017-01-09
18:10:06',5),(306,'2017-01-10 21:59:22',35),(307,'2017-01-12
14:20:17',3),(308,'2017-01-23 15:33:32',14),(309,'2017-01-23
23:01:54',1),(310,'2017-01-24 21:58:44',4),(311,'2017-01-24
22:24:43',3),(312,'2017-01-25 02:31:47',90),(313,'2017-01-26
13:54:47',2),(314,'2017-01-28 04:48:15',2),(315,'2017-01-30
14:21:09',12),(316,'2017-02-01 16:20:18',20),(317,'2017-02-02
20:35:53',3),(318,'2017-02-03 14:54:23',6),(319,'2017-02-06
21:10:37',30),(320,'2017-02-07 16:54:41',1),(321,'2017-02-07
18:00:09',2),(322,'2017-02-07 18:55:21',17),(323,'2017-02-08
15:01:46',7),(324,'2017-02-10 15:54:06',17),(325,'2017-02-13
17:29:42',40),(326,'2017-02-13 19:51:30',4),(327,'2017-02-14
22:26:21',3),(328,'2017-02-22 19:44:23',10),(329,'2017-02-23
18:27:35',74),(330,'2017-02-27 15:48:55',24),(331,'2017-02-27
```

```
16:30:07',3),(332,'2017-02-27 20:32:16',250),(333,'2017-03-03
14:29:51',17),(334,'2017-03-03 18:38:18',8),(335,'2017-03-03
19:06:49',8),(336,'2017-03-05 17:25:41',3),(337,'2017-03-07
16:22:07',47),(338,'2017-03-07 19:29:49',5),(339,'2017-03-10
17:09:24',55),(340,'2017-03-15 15:51:06',161),(341,'2017-03-15
16:46:34',2),(342,'2017-03-15 18:36:11',8),(343,'2017-03-17
15:33:43',251),(344,'2017-03-20 18:10:41',44),(345,'2017-03-21
15:20:17',100),(346,'2017-03-21 17:19:10',2),(347,'2017-03-23
16:55:49',2),(348,'2017-03-24 17:11:43',20),(349,'2017-03-24
17:13:50',50),(350,'2017-03-24 18:19:26',48),(351,'2017-03-24
18:51:06',20),(352,'2017-03-27 19:24:19',20),(353,'2017-03-27
19:26:26',74),(354,'2017-03-29 14:26:14',2),(355,'2017-03-30
17:17:53',4),(356,'2017-03-31 16:58:08',90),(357,'2017-04-03
19:53:29',29),(358,'2017-04-04 19:38:59',3),(359,'2017-04-06
15:27:30',126),(360,'2017-04-07 16:42:00',35),(361,'2017-04-12
16:52:52',30),(362,'2017-04-17 14:48:50',20),(363,'2017-04-17
16:59:25',106),(364,'2017-04-18 14:50:57',20),(365,'2017-04-19
02:07:14',4),(366,'2017-04-19 17:14:08',5),(367,'2017-04-20
16:47:09',5),(368,'2017-04-22 16:48:36',22),(369,'2017-04-24
18:01:40',8),(370,'2017-04-28 16:38:06',6),(371,'2017-05-01
15:28:21',22),(372,'2017-05-15 14:30:11',13),(373,'2017-05-15
22:52:44',45),(374,'2017-05-16 17:54:52',7),(375,'2017-05-17
17:35:45',25),(376,'2017-05-18 13:10:57',4),(377,'2017-05-18
17:15:08',8),(378,'2017-05-24 20:56:59',60),(379,'2017-05-25
18:22:16',33),(380,'2017-06-01 00:56:52',20),(381,'2017-06-01
16:04:44',2),(382,'2017-06-02 01:52:39',25),(383,'2017-06-02
20:36:39',22),(384,'2017-06-05 15:41:36',5),(385,'2017-06-05
17:46:20',20),(386,'2017-06-05 21:54:51',12),(387,'2017-06-06
00:58:40',17),(388,'2017-06-07 19:39:49',27),(389,'2017-06-08
15:58:04',67),(390,'2017-06-08 16:02:23',103),(391,'2017-06-12
17:32:28',20),(392,'2017-06-16 13:43:04',4),(393,'2017-06-19
15:41:07',46),(394,'2017-06-20 19:22:58',4),(395,'2017-06-23
19:43:09',30),(396,'2017-06-27 15:47:02',23),(397,'2017-06-27
16:22:00',68),(398,'2017-06-28 17:25:43',60),(399,'2017-06-29
15:19:33',25),(400,'2017-06-30 19:35:55',2),(401,'2017-07-05
20:40:19',25),(402,'2017-07-10 19:09:12',45),(403,'2017-07-10
19:10:12',128),(404,'2017-07-12 15:12:21',4),(405,'2017-07-14
21:44:17',30),(406,'2017-07-17 16:52:34',95),(407,'2017-07-17
17:27:06',100),(408,'2017-07-18 19:39:52',2),(409,'2017-07-24
16:48:00',110),(410,'2017-07-28 23:20:18',2),(411,'2017-07-31
18:42:30',92),(412,'2017-08-07 17:43:19',62),(413,'2017-08-08
16:55:13',25),(414,'2017-08-10 14:22:27',280),(415,'2017-08-11
14:41:39',128),(416,'2017-08-14 19:09:59',111),(417,'2017-08-17
16:44:48',67),(418,'2017-08-18 04:56:01',2),(419,'2017-08-21
20:26:01',43),(420,'2017-08-23 19:30:22',3),(421,'2017-08-24
```

```
22:19:50',2),(422,'2017-08-28 18:16:27',5),(423,'2017-09-08
19:37:55',5),(424,'2017-09-11 15:44:01',2),(425,'2017-09-11
17:57:19',2),(426,'2017-09-11 18:25:53',18),(427,'2017-09-14
17:11:08',90),(428,'2017-09-18 19:15:18',14),(429,'2017-09-19
16:52:43',2),(430,'2017-09-25 18:06:23',13),(431,'2017-09-28
18:35:26',18),(432,'2017-09-29 14:59:24',60),(433,'2017-10-02
13:07:02',20),(434,'2017-10-02 19:16:34',24),(435,'2017-10-02
21:54:46',2),(436,'2017-10-09 17:33:35',115),(437,'2017-10-10
21:15:52',61),(438,'2017-10-12 00:39:56',1),(439,'2017-10-16
15:38:13',35),(440,'2017-10-17 14:37:05',1),(441,'2017-10-23
17:35:24',22),(442,'2017-10-25 17:57:52',26),(443,'2017-10-25
19:04:39',1),(444,'2017-10-27 19:36:00',45),(445,'2017-10-27
23:03:33',3),(446,'2017-10-30 16:23:49',34),(447,'2017-10-31
16:50:00',20),(448,'2017-11-03 23:50:45',2),(449,'2017-11-06
18:13:31',10),(450,'2017-11-08 16:39:35',150),(451,'2017-11-08
16:42:21',100),(452,'2017-11-08 17:03:23',40),(453,'2017-11-09
03:27:32',5),(454,'2017-11-09 17:56:27',24),(455,'2017-11-10
21:41:46',2),(456,'2017-11-13 17:15:15',16),(457,'2017-11-17
19:40:26',297),(458,'2017-11-20 15:52:30',2),(459,'2017-11-21
22:15:33',4),(460,'2017-11-27 20:17:03',12),(461,'2017-11-29
00:00:10',4),(462,'2017-11-30 22:57:01',45),(463,'2017-12-04
17:35:40',50),(464,'2017-12-06 17:15:51',2),(465,'2017-12-11
15:53:22',4),(466,'2017-12-11 18:04:44',4),(467,'2017-12-13
21:07:25',2),(468,'2017-12-14 21:19:09',30),(469,'2017-12-22
12:59:57',8),(470,'2018-01-09 17:09:20',20),(471,'2018-01-11
01:49:09',35),(472,'2018-01-12 15:04:50',2),(473,'2018-01-12
16:41:58',120),(474,'2018-01-15 17:39:29',4),(475,'2018-01-18
15:35:06',8),(476,'2018-01-24 14:56:33',11),(477,'2018-01-24
15:43:27',1),(478,'2018-01-24 18:24:44',7),(479,'2018-01-29
14:47:57',33),(480,'2018-01-29 15:12:04',14),(481,'2018-02-02
19:26:29',3),(482,'2018-02-06 18:35:41',10),(483,'2018-02-06
19:19:56',10),(484,'2018-02-07 09:39:16',1),(485,'2018-02-08
21:12:02',30),(486,'2018-02-09 15:37:31',1),(487,'2018-02-12
16:03:02',45),(488,'2018-02-15 19:20:53',3),(489,'2018-02-16
19:31:36',2),(490,'2018-02-19 17:48:48',1),(491,'2018-02-19
19:46:40',221),(492,'2018-02-21 20:23:47',1),(493,'2018-02-21
20:56:02',2),(494,'2018-02-26 19:46:25',15),(495,'2018-02-28
15:00:00',50),(496,'2018-02-28 15:02:37',50),(497,'2018-02-28
22:10:32',70),(498,'2018-03-05 14:56:04',12),(499,'2018-03-08
17:54:29',18),(500,'2018-03-12 16:49:33',232),(501,'2018-03-12
17:44:51',2),(502,'2018-03-15 23:06:38',15),(503,'2018-03-15
23:11:33',30),(504,'2018-03-16 15:12:56',190),(505,'2018-03-16
18:29:11',2),(506,'2018-03-19 15:44:08',56),(507,'2018-03-19
17:59:29',20),(508,'2018-03-19 18:11:46',30),(509,'2018-03-19
23:01:03',2),(510,'2018-03-21 20:56:21',2),(511,'2018-03-26
```

```
19:38:10',86),(512,'2018-03-26 21:16:00',30),(513,'2018-03-27
19:04:08',30),(514,'2018-03-29 07:02:41',20),(515,'2018-04-02
18:41:38',114),(516,'2018-04-04 13:23:44',1),(517,'2018-04-05
15:35:19',13),(518,'2018-04-05 21:55:47',22),(519,'2018-04-12
14:20:06',15),(520,'2018-04-12 14:25:56',15),(521,'2018-04-13
21:54:37',56),(522,'2018-04-14 03:29:33',6),(523,'2018-04-16
05:32:23',14),(524,'2018-04-16 16:44:07',14),(525,'2018-04-16
19:37:00',50),(526,'2018-04-16 19:59:24',25),(527,'2018-04-17
03:04:33',20),(528,'2018-04-17 22:20:00',15),(529,'2018-04-18
16:37:28',15),(530,'2018-04-20 19:22:39',10),(531,'2018-04-23
16:36:13',23),(532,'2018-04-24 21:56:48',25),(533,'2018-04-26
15:13:26',20),(534,'2018-04-26 15:22:15',12),(535,'2018-04-30
18:29:48',19),(536,'2018-05-01 19:13:06',22),(537,'2018-05-04
19:31:31',30),(538,'2018-05-04 19:38:04',30),(539,'2018-05-04
19:55:20',30),(540,'2018-05-04 19:58:16',30),(541,'2018-05-07
17:50:04',16),(542,'2018-05-09 20:08:28',25),(543,'2018-05-11
18:20:10',20),(544,'2018-05-14 20:15:17',21),(545,'2018-05-15
17:40:06',38),(546,'2018-05-17 04:25:23',25),(547,'2018-05-21
20:33:12',40),(548,'2018-05-23 14:48:31',2),(549,'2018-05-23
15:15:07',13),(550,'2018-05-24 19:24:37',20),(551,'2018-05-25
20:39:46',3),(552,'2018-05-30 20:53:29',5),(553,'2018-06-05
15:03:57',72),(554,'2018-06-08 17:58:49',24),(555,'2018-06-11
17:25:41',50),(556,'2018-06-14 16:50:04',4),(557,'2018-06-18
15:09:58',67),(558,'2018-06-21 20:48:51',35),(559,'2018-06-22
00:06:08',5),(560,'2018-06-22 14:38:30',30),(561,'2018-06-22
19:07:53',2),(562,'2018-06-25 15:40:00',70),(563,'2018-06-26
17:09:42',2),(564,'2018-07-02 14:43:35',113),(565,'2018-07-03
18:48:27',6),(566,'2018-07-06 17:29:37',7),(567,'2018-07-09
15:15:24',60),(568,'2018-07-16 16:03:51',113),(569,'2018-07-17
16:26:19',7),(570,'2018-07-17 17:52:23',2),(571,'2018-07-24
17:18:18',45),(572,'2018-07-24 18:12:57',88),(573,'2018-07-25
20:02:30',30),(574,'2018-07-25 20:07:57',4),(575,'2018-07-26
23:03:53',30),(576,'2018-07-27 18:54:55',4),(577,'2018-07-30
16:37:44',10),(578,'2018-07-30 16:40:38',119),(579,'2018-08-06
16:07:14',15),(580,'2018-08-06 19:42:59',89),(581,'2018-08-07
21:16:00',4),(582,'2018-08-08 16:03:17',5),(583,'2018-08-10
14:16:20',126),(584,'2018-08-10 17:48:50',2),(585,'2018-08-20
17:16:36',50),(586,'2018-08-20 17:19:22',53),(587,'2018-08-20
17:21:11',72),(588,'2018-08-22 17:51:58',4),(589,'2018-08-27
17:32:44',52),(590,'2018-08-29 19:42:41',23),(591,'2018-09-04
19:10:17',30),(592,'2018-09-05 15:40:41',10),(593,'2018-09-07
21:36:22',17),(594,'2018-09-10 15:38:59',59),(595,'2018-09-12
16:28:36',3),(596,'2018-09-17 17:39:43',24),(597,'2018-09-18
21:59:24',20),(598,'2018-09-18 22:24:19',15),(599,'2018-09-21
11:14:16',30),(600,'2018-09-21 22:22:39',5),(601,'2018-09-26
```

```
17:24:10',4),(602,'2018-10-01 15:13:02',38),(603,'2018-10-01
20:05:15',4),(604,'2018-10-01 20:56:49',2),(605,'2018-10-04
22:17:28',3),(606,'2018-10-07 23:58:35',2),(607,'2018-10-08
14:32:54',198),(608,'2018-10-09 19:32:43',30),(609,'2018-10-10
20:36:24',50),(610,'2018-10-11 17:46:59',33),(611,'2018-10-11
19:52:47',40),(612,'2018-10-15 16:19:06',40),(613,'2018-10-15
16:25:09',112),(614,'2018-10-22 17:01:49',30),(615,'2018-10-22
17:47:45',50),(616,'2018-10-22 18:08:01',57),(617,'2018-10-23
15:30:02',19),(618,'2018-10-24 20:11:10',3),(619,'2018-10-25
04:05:29',25),(620,'2018-10-26 18:46:48',30),(621,'2018-10-29
15:52:40',47),(622,'2018-11-05 15:58:51',35),(623,'2018-11-05
16:42:46',3),(624,'2018-11-05 19:20:20',6),(625,'2018-11-07
20:26:11',80),(626,'2018-11-09 20:06:09',15),(627,'2018-11-15
19:07:11',100),(628,'2018-11-26 15:45:32',83),(629,'2018-11-26
20:59:59',12),(630,'2018-12-04 15:43:28',40),(631,'2019-01-02
15:05:02',2),(632,'2019-01-07 16:17:43',111),(633,'2019-01-08
01:03:16',3),(634,'2019-01-08 17:09:41',4),(635,'2019-01-09
03:11:57',23),(636,'2019-01-14 16:48:56',16),(637,'2019-01-17
19:04:16',55),(638,'2019-01-23 17:29:27',9),(639,'2019-01-28
15:30:30',30),(640,'2019-01-28 17:15:53',3),(641,'2019-01-30
15:48:30',6),(642,'2019-02-08 17:13:41',10),(643,'2019-02-11
15:48:25',35),(644,'2019-02-11 16:12:17',10),(645,'2019-02-11
16:54:19',1),(646,'2019-02-12 17:30:14',8),(647,'2019-02-14
17:58:20',30),(648,'2019-02-15 22:19:10',15),(649,'2019-02-18
20:14:37',2),(650,'2019-02-18 20:18:41',30),(651,'2019-02-18
21:49:37',203),(652,'2019-02-18 22:39:03',2),(653,'2019-02-19
16:45:47',10),(654,'2019-02-19 19:18:36',2),(655,'2019-02-20
03:44:13',5),(656,'2019-02-22 21:44:07',35),(657,'2019-03-01
21:46:00',4),(658,'2019-03-04 21:15:07',2),(659,'2019-03-07
15:40:43',17),(660,'2019-03-11 15:05:01',220),(661,'2019-03-14
15:29:47',1),(662,'2019-03-15 17:11:21',100),(663,'2019-03-18
15:38:18',105),(664,'2019-03-18 22:05:42',30),(665,'2019-03-19
20:20:07',25),(666,'2019-03-19 20:41:39',6),(667,'2019-03-20
16:40:37',30),(668,'2019-03-20 16:57:40',50),(669,'2019-03-20
17:00:02',40),(670,'2019-03-20 17:02:41',3),(671,'2019-03-24
21:18:12',30),(672,'2019-03-25 19:19:44',137),(673,'2019-03-27
02:10:26',20),(674,'2019-03-27 18:03:13',3),(675,'2019-04-01
15:35:51',2),(676,'2019-04-01 16:26:21',102),(677,'2019-04-02
01:10:27',1),(678,'2019-04-03 17:50:19',3),(679,'2019-04-03
23:22:45',125),(680,'2019-04-04 17:25:24',2),(681,'2019-04-06
03:46:32',15),(682,'2019-04-07 03:48:17',3),(683,'2019-04-09
03:56:08',20),(684,'2019-04-09 04:03:00',2),(685,'2019-04-10
15:31:35',20),(686,'2019-04-11 17:59:25',36),(687,'2019-04-15
02:39:55',15),(688,'2019-04-15 17:05:40',10),(689,'2019-04-19
00:17:53',14),(690,'2019-04-19 17:34:41',40),(691,'2019-04-22
```

```
16:12:19',117),(692,'2019-04-23 16:29:35',3),(693,'2019-04-25
21:52:43',29),(694,'2019-05-06 17:00:29',15),(695,'2019-05-06
17:01:05',4),(696,'2019-05-09 17:05:50',3),(697,'2019-05-12
00:57:32',3),(698,'2019-05-13 19:29:09',47),(699,'2019-05-14
16:35:02',24),(700,'2019-05-15 14:54:15',35),(701,'2019-05-16
22:06:32',70),(702,'2019-05-21 23:18:24',5),(703,'2019-05-23
17:01:43',3),(704,'2019-05-23 17:14:41',19),(705,'2019-06-02
22:18:30',40),(706,'2019-06-03 04:23:12',2),(707,'2019-06-03
18:49:35',2),(708,'2019-06-03 20:41:01',148),(709,'2019-06-04
15:25:24',25),(710,'2019-06-04 21:03:18',4),(711,'2019-06-05
17:01:11',20),(712,'2019-06-07 17:11:59',3),(713,'2019-06-10
16:37:03',153),(714,'2019-06-10 18:34:35',20),(715,'2019-06-13
16:29:24',45),(716,'2019-06-14 19:58:19',25),(717,'2019-06-17
16:52:18',171),(718,'2019-06-18 20:00:13',35),(719,'2019-06-18
20:00:18',35),(720,'2019-06-21 15:05:13',28),(721,'2019-06-21
15:17:28',6),(722,'2019-06-23 01:51:17',2),(723,'2019-06-24
16:12:14',169),(724,'2019-06-26 17:43:09',6),(725,'2019-06-27
14:48:27',60),(726,'2019-06-27 16:23:33',83),(727,'2019-07-09
05:54:20',5),(728,'2019-07-12 16:21:51',2),(729,'2019-07-12
19:53:43',12),(730,'2019-07-15 18:31:18',1),(731,'2019-07-15
23:17:19',25),(732,'2019-07-15 23:24:52',25),(733,'2019-07-15
23:29:26',25),(734,'2019-07-17 17:55:41',35),(735,'2019-07-17
22:50:35',10),(736,'2019-07-19 15:31:01',93),(737,'2019-07-19
15:33:56',100),(738,'2019-07-23 16:40:22',193),(739,'2019-07-26
17:01:57',30),(740,'2019-07-27 01:50:17',30),(741,'2019-07-29
15:18:18',196),(742,'2019-07-29 21:00:24',3),(743,'2019-07-31
22:40:28',2),(744,'2019-08-05 17:44:39',3),(745,'2019-08-05
22:56:23',18),(746,'2019-08-06 15:44:47',3),(747,'2019-08-09
13:16:33',3),(748,'2019-08-09 15:53:05',4),(749,'2019-08-09
22:57:14',2),(750,'2019-08-12 16:37:34',96),(751,'2019-08-13
18:41:04',1),(752,'2019-08-15 22:58:55',1),(753,'2019-08-21
21:43:04',55),(754,'2019-08-21 22:35:46',2),(755,'2019-08-23
18:00:57',14),(756,'2019-08-26 14:22:28',98),(757,'2019-08-27
20:06:15',55),(758,'2019-08-28 22:09:36',5),(759,'2019-09-05
23:46:31',60),(760,'2019-09-09 15:37:50',45),(761,'2019-09-11
16:06:15',40),(762,'2019-09-11 19:11:07',2),(763,'2019-09-12
15:33:27',30),(764,'2019-09-12 22:29:21',6),(765,'2019-09-18
03:54:44',24),(766,'2019-09-23 16:04:15',60),(767,'2019-09-23
18:52:50',50),(768,'2019-09-25 16:12:51',2),(769,'2019-09-27
16:41:17',1),(770,'2019-09-30 16:58:59',62),(771,'2019-10-03
15:35:47',15),(772,'2019-10-07 22:52:20',50),(773,'2019-10-10
19:43:04',47),(774,'2019-10-14 15:45:49',215),(775,'2019-10-14
19:59:39',3),(776,'2019-10-22 17:03:25',79),(777,'2019-10-22
17:05:40',8),(778,'2019-10-24 17:26:01',8),(779,'2019-10-25
19:48:10',5),(780,'2019-11-01 15:24:19',6),(781,'2019-11-04
```

```
16:07:02',49),(782,'2019-11-07 20:25:22',12),(783,'2019-11-08
07:46:35',15),(784,'2019-11-11 22:13:01',2),(785,'2019-11-14
01:51:47',35),(786,'2019-11-18 16:04:27',48),(787,'2019-11-25
16:21:59',195),(788,'2019-12-03 15:22:01',6),(789,'2019-12-09
21:55:46',120),(790,'2019-12-13 15:14:41',4),(791,'2019-12-18
18:35:33',10),(792,'2019-12-20 23:46:22',6),(793,'2020-01-06
16:29:29',149),(794,'2020-01-07 18:32:49',65),(795,'2020-01-07
19:52:46',25),(796,'2020-01-09 20:44:38',1),(797,'2020-01-13
15:50:57',19),(798,'2020-01-16 20:33:48',40),(799,'2020-01-21
21:28:09',5),(800,'2020-01-27 17:01:56',23),(801,'2020-01-31
17:14:50',4),(802,'2020-02-04 16:26:20',22),(803,'2020-02-10
14:55:35',3),(804,'2020-02-10 15:48:36',36),(805,'2020-02-10
20:24:47',21),(806,'2020-02-10 20:27:09',13),(807,'2020-02-11
15:45:37',8),(808,'2020-02-28 19:14:26',5),(809,'2020-03-03
23:25:23',30),(810,'2020-03-05 01:09:22',60),(811,'2020-03-05
01:11:35',50),(812,'2020-03-05 19:15:36',20),(813,'2020-03-10
21:16:14',45),(814,'2021-03-18 14:07:58',8),(815,'2021-04-16
16:35:56',3),(816,'2021-05-10 22:59:41',6),(817,'2021-05-21
15:07:55',6),(818,'2021-07-07 21:16:15',5),(819,'2021-07-15
16:20:14',2),(820,'2021-07-15 17:28:47',4),(821,'2021-07-27
17:46:34',5),(822,'2021-07-28 20:45:07',3),(823,'2021-07-28
23:47:46',3),(824,'2021-07-29 17:39:06',2),(825,'2021-07-29
17:53:57',3),(826,'2021-08-02 16:26:48',2),(827,'2021-08-24
22:36:07',4),(828,'2021-09-12 23:43:42',3),(829,'2021-09-22
17:36:34',20),(830,'2021-10-07 20:30:54',3),(831,'2021-10-13
21:11:47',6),(832,'2021-10-13 21:29:48',3),(833,'2021-10-19
19:39:01',1),(834,'2021-10-20 14:24:31',5),(835,'2021-11-01
23:58:22',2),(836,'2021-11-09 21:11:04',5),(837,'2021-12-18
20:14:41',3),(838,'2022-01-10 16:07:49',2),(839,'2022-01-28
17:22:31',40),(840,'2022-01-30 00:00:37',3);
/*!40000 ALTER TABLE `daily_visitors` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2022-03-06 16:03:50
```

C Code

clock_heatmap.c

```
/*

    This program serves to help drive the development of the clock heatmap
software.
        1 - Given a sample of the number of users per hour for the last day,
first normalize the data.
        2 - To normalize, divide all values by the maximum value.
        3 - Convert normalized data to analog values.
        4 - Write the analog values to the corresponding channels on the mux.

*/

#include <stdio.h>

/*

    To compile the code to an executable file, type this into the command prompt:
        gcc -o "Name of Exec File" filename.c
    Then, to run the compiled program:
        ./name_of_exec_file

*/

float sample_data[12] = {
    10,
    10,
    3,
    2,
    1,
    20,
    20,
    35,
    35,
    50,
    55,
    55
};

float largest(float arr[], int n)
{
    int i;
```

```
    // Initialize maximum element
    int max = arr[0];

    // Traverse array elements from second and
    // compare every element with current max
    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}

//This function normalizes the traffic data for each hour and converts to analog
output (0 to 255)
float weights(int hour){
    printf("%0.0f\t",largest(sample_data, 12));
    float weight = (sample_data[hour]/largest(sample_data, 12));
    weight = 255*weight;
    return weight;
}

int main(){

    float analogVals[12];

    for(int i = 0; i < 12; i++){
        analogVals[i] = weights(i);
        printf("%0.0f\t",sample_data[i]);
        printf("Channel %d is set at %0.0f\n",i,analogVals[i]);
    };


}
```

C++ Code

failsafe_sd.cpp

```
// g++ -o <name-you-want-to-give> source.cpp

#include <iostream>
#include <fstream>
#include <string>
```

```
#include <stdio.h>
#include <cstring>

using namespace std;

void writeFile(void){
  ofstream myfile;
  myfile.open ("example.txt");
  myfile <<
"2015,2552;2016,5212;2017,5821,2018,5319;2019,5819;2020,644;2021,107;2022,45";
  myfile.close();
}

string readFile(void){
  // Create a text string, which is used to output the text file
  string myText;

  // Read from the text file
  ifstream MyReadFile("example.txt");
  string raw;
  // Use a while loop together with the getline() function to read the file line
by line
  while (getline (MyReadFile, myText)) {
    // Output the text from the file
    raw = myText;
  }

  cout << raw << endl;

  // Close the file
  MyReadFile.close();
  return raw;
}


char *result[] splitPoints(string raw){
  int n = raw.length();

  // declaring character array
  char myString[n + 1];

  // copying the contents of the
  // string to char array
  strcpy(myString, raw.c_str());
  char *answers[8];
```

```
  char *p = strtok(myString, ";");
  int i = 0;
  while (p) {
      printf ("Token: %s\n", p);
      answers[i] = p;
      i++;
      p = strtok(NULL, ";");
  }
  cout << answers;
  return answers;
}

int main () {
  string raw_data = readFile();
  char *points[] = splitPoints(raw_data);
  return 0;
}
```

led_matrix.cpp

```
// This program was used to test the matrix software while supplies were yet
// to arrive to allow for hardware testing

#include <iostream>
#include <stdio.h>
#include <string>

const int register_count = 19;
const int led_count = 7*7; // 7x7 matrix

// Matrix for RGB LED pins, ordered as leds[led#][r,g,b]
int leds[led_count][3] = {0};

bool registers[8*register_count];

void makeRed(int led){
    leds[led][0] = 255;
    leds[led][1] = 0;
    leds[led][2] = 0;
}

void makeGreen(int led){
    leds[led][0] = 0;
    leds[led][1] = 255;
```

```
    leds[led][2] = 0;
}

void makeBlue(int led){
    leds[led][0] = 0;
    leds[led][1] = 0;
    leds[led][2] = 255;
}

void makeYellow(int led){
    leds[led][0] = 0;
    leds[led][1] = 255;
    leds[led][2] = 255;
}

void turnOffLed(int led){
    leds[led][0] = 0;
    leds[led][1] = 0;
    leds[led][2] = 0;
}

int main(){
    // Set LED #4 to red
    leds[4][0] = 255;

    printf("LED #\tRed\tGreen\tBlue\n\n");
    for(int i=0; i<led_count; i++){
        if(i<=15 && i>=0){
            makeRed(i);
        }else if(i<=30 && i>15){
            makeGreen(i);
        }else if(i<=45 && i>30){
            makeBlue(i);
        }else{
            makeYellow(i);
        }
        printf("LED %d\t%d\t%d\t%d\n",i+1,leds[i][0],leds[i][1],leds[i][2]);
    }

    std::string x;

    std::cin >> x;

    return 0;
}
```

Python Code

clock_determination.py

```
# This program is used to calculate the expected output for the Arduino to be
# used in testing

from random import randrange
import numpy as np
import pandas as pd

#  Prominent Arduino map function :)
def _map(x, in_min, in_max, out_min, out_max):
    return int((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)

visitors = [randrange(1000,6000) for i in range(12)]
colors = [_map(i,0,6000,0,3) for i in visitors]

print(visitors)
for i in range(len(colors)):
    if colors[i] == 0:
        print(f'Hour {i}: Green')
    elif colors[i] == 1:
        print(f'Hour {i}: Yellow')
    else:
        print(f'Hour {i}: Red')
```

fastled.py

```
# This python program writes the code for the neopixels, since there is a lot
# of copying and pasting, this program removes the tedious nature of this job

from random import randrange
import pyperclip

res = ""

for i in range(8):
    num = randrange(1000,6000)
    year = i+2015
    res += f"{year},{num}\n"

print(res)
pyperclip.copy(res)
```

```
# Dataset #2
# 2015,5562 //53.7cm
# 2016,2378 //27.5cm
# 2017,2083 //24.4cm
# 2018,2686 //29.2cm
# 2019,1322 //18.4cm
# 2020,5689 //54.4cm
# 2021,5962 //54.7cm
# 2022,1128 //17.0cm

# Dataset #3
# 2015,1268 //17.8cm
# 2016,3146 //33.0cm
# 2017,1527 //19.7cm
# 2018,3228 //34.2cm
# 2019,5494 //53.2cm
# 2020,2049 //24.9cm
# 2021,4746 //47.0cm
# 2022,1327 //18.0cm

res = ""
for i in range(100):
    res += f"\tleds[{i}] = CRGB::White;\n"

pyperclip.copy(res)
```

matrix_determination.py

```
# This program is used to calculate the expected output for the Arduino to be
# used in testing

from random import randrange
import numpy as np
import pandas as pd

#  Prominent Arduino map function :)
def _map(x, in_min, in_max, out_min, out_max):
    return int((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)

visitors = [randrange(0,2) for i in range(5000)]

print(sum(visitors))
print(len(visitors) - sum(visitors))
perc = 1 - (sum(visitors)/len(visitors))
```

```
perc = perc*100
print(f"Percent Female: {perc:.0f}%")
```

oedk_data_exploration.py

```
# This program is used to explore the data the OEDK has given us to analyze
# the most useful datasets

import pandas as pd
import matplotlib.pyplot as plt

# Give the location of the file
loc = ("E:/Rice/Rice S22/Intro to Engineering Design and
Communication/Software/Backend Code/TeamTa-daData_VISITOR_DEMOGRAPHICS.xlsx")

# To open Workbook
wb = pd.read_excel(loc)
sheet = wb
loc = ("E:/Rice/Rice S22/Intro to Engineering Design and
Communication/Software/Backend Code/cleaned_demographics.xlsx")
cleanwb = pd.read_excel(loc)

def isfloat(num):
    try:
        float(num)
        return True
    except ValueError:
        return False

# For row 0 and column 0
my_index = []
for i in range(0,len(sheet["Check below if your objective is to gain  ideas for
building a similar facility or running a similar design program."])):
    if isfloat(sheet["Check below if your objective is to gain  ideas for
building a similar facility or running a similar design program."][i]) == False:
        my_index.append(i)

for i in my_index:
    cleanwb["similar_facility"][i] = 1

visitors = sheet["Estimated Attendance"]
date_created = sheet["Date Created"]

data = {"Date": date_created,
```

```
        "Visitors": visitors}

df = pd.DataFrame(data)

print(df)

monthly_2015 = []
monthly_2016 = []
monthly_2017 = []
monthly_2018 = []
monthly_2019 = []
monthly_2020 = []
monthly_2021 = []
monthly_2022 = []
months1 = []
months2 = []
months3 = []
months4 = []
months5 = []
months6 = []
months7 = []
months8 = []

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2022:
        monthly_2022.append(df["Visitors"][i])
        months1.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2021:
        monthly_2021.append(df["Visitors"][i])
        months2.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2020:
        monthly_2020.append(df["Visitors"][i])
        months3.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2019:
        monthly_2019.append(df["Visitors"][i])
        months4.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2018:
```

```
        monthly_2018.append(df["Visitors"][i])
        months5.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2017:
        monthly_2017.append(df["Visitors"][i])
        months6.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2016:
        monthly_2016.append(df["Visitors"][i])
        months7.append(df["Date"][i].month)

for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2015:
        monthly_2015.append(df["Visitors"][i])
        months8.append(df["Date"][i].month)

fig, axes = plt.subplots(4, 2)
axes[0,0].bar(months8,monthly_2015)
axes[0,0].set_title("OEDK Visitors 2015")
axes[0,0].set_xlabel("Month")
axes[0,0].set_ylabel("Number of Visitors")

axes[0,1].bar(months7,monthly_2016)
axes[0,1].set_title("OEDK Visitors 2016")
axes[0,1].set_xlabel("Month")
axes[0,1].set_ylabel("Number of Visitors")

axes[1,0].bar(months6,monthly_2017)
axes[1,0].set_title("OEDK Visitors 2017")
axes[1,0].set_xlabel("Month")
axes[1,0].set_ylabel("Number of Visitors")

axes[1,1].bar(months5,monthly_2018)
axes[1,1].set_title("OEDK Visitors 2018")
axes[1,1].set_xlabel("Month")
axes[1,1].set_ylabel("Number of Visitors")

axes[2,0].bar(months4,monthly_2019)
axes[2,0].set_title("OEDK Visitors 2019")
axes[2,0].set_xlabel("Month")
axes[2,0].set_ylabel("Number of Visitors")

axes[2,1].bar(months3,monthly_2020)
```

```
axes[2,1].set_title("OEDK Visitors 2020")
axes[2,1].set_xlabel("Month")
axes[2,1].set_ylabel("Number of Visitors")

axes[3,0].bar(months2,monthly_2021)
axes[3,0].set_title("OEDK Visitors 2021")
axes[3,0].set_xlabel("Month")
axes[3,0].set_ylabel("Number of Visitors")

plt.tight_layout()
plt.show()

years = [2015, 2016, 2017, 2018, 2019,
        2020, 2021]
yearly_2015 = 0
yearly_2016 = 0
yearly_2017 = 0
yearly_2018 = 0
yearly_2019 = 0
yearly_2020 = 0
yearly_2021 = 0
for i in range(0,len(df["Date"])):
    if df["Date"][i].year == 2015:
        yearly_2015 += df["Visitors"][i]
    elif df["Date"][i].year == 2016:
        yearly_2016 += df["Visitors"][i]
    elif df["Date"][i].year == 2017:
        yearly_2017 += df["Visitors"][i]
    elif df["Date"][i].year == 2018:
        yearly_2018 += df["Visitors"][i]
    elif df["Date"][i].year == 2019:
        yearly_2019 += df["Visitors"][i]
    elif df["Date"][i].year == 2020:
        yearly_2020 += df["Visitors"][i]
    elif df["Date"][i].year == 2021:
        yearly_2021 += df["Visitors"][i]

yearly = [yearly_2015 , yearly_2016, yearly_2017, yearly_2018, yearly_2019,
yearly_2020, yearly_2021]
print(yearly_2019.sum())

plt.bar(years, yearly)
plt.title("OEDK Visitors per Year")
plt.xlabel("Year")
plt.ylabel("Number of Visitors")
```

```
plt.show()

plt.plot_date(df["Date"], df["Visitors"])
plt.title("OEDK Visitors per Day")
plt.xlabel("Date")
plt.ylabel("Number of Visitors")
plt.show()
```

rocket_determination.py

```
# This program is used to calculate the expected output for the Arduino to be
# used in testing

from random import randrange
import numpy as np
import pandas as pd

#  Prominent Arduino map function :)
def _map(x, in_min, in_max, out_min, out_max):
    return int((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)

visitors = [randrange(0,6000) for i in range(8)]

print(visitors)
for i in range(len(visitors)):
    year = 2015 + i
    steps = _map(visitors[i],0,6000,0,160000)
    distance = _map(visitors[i],0,6000,0,500)
    print(f"{year}: {steps} steps ({distance} mm)")
```

test_results.py

```
# This program is used to analyze the results of our tests and generate plots

from operator import index
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

#  Prominent Arduino map function :)
def _map(x, in_min, in_max, out_min, out_max):
    return int((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)

def percentage_error(experimental, theoretical):
```

```
    errors = []
    for i in range(len(experimental)):
        difference = np.abs(experimental[i] - theoretical[i])/theoretical[i] *
100
        errors.append(difference)
    return np.mean(errors)

z = ""

rocket_data = [2.45, 3.50, 3.20, 0.98, 2.34, 0.74, 0.23, 0]
raw_data = [2552, 5212, 5821, 5319, 5819, 644, 107, 45]
true_dist = [_map(i,0,6000,0,500) for i in raw_data]
perc_dev = [rocket_data[i]/true_dist[i]*100 for i in range(len(rocket_data))]
rocket_data1 = np.array([rocket_data]) + np.array([true_dist])
years = [i+2015 for i in range(8)]

true_dist = np.array([true_dist])
rocket_data = np.array([rocket_data])
nominal_dist = true_dist + rocket_data


error1 = percentage_error(nominal_dist, true_dist)

df = pd.DataFrame({'True Position': rocket_data1.flatten(),
                   'Expected Position': true_dist.flatten()}
                   ,index=years)

colors = ['#d62d20', 'k', 'y', 'pink','orange','cyan','darkgrey']
ax = df.plot(kind='bar', color=colors, legend=False, rot=0)

ax.legend()
plt.title("Hardware-Software Integration Accuracy Test")
plt.suptitle(f"Sample Dataset #1 ({error1:.2f}% Error)",fontweight="bold")
plt.xlabel("Year Data Point")
plt.ylabel("Motor Deviation (mm)")
plt.show()

# fig = plt.figure()
# plt.bar(years, rocket_data)
# fig.text(.6,.7,f"Mean Error: {np.mean(rocket_data):.2f}mm",color="r")
# plt.title("Nominal Motor Deviation Over 21,000 Uses")
# plt.suptitle("Accelerated Usage Test")
# plt.xlabel("Year Data Point")
# plt.ylabel("Motor Deviation (mm)")
# plt.show()
```

```
# fig = plt.figure()
# plt.bar(years, perc_dev)
# fig.text(.3,.5,f"Mean Error: {np.mean(perc_dev):.2f}%",color="r")
# plt.title("Percent Motor Deviation Over 21,000 Uses")
# plt.suptitle("Accelerated Usage Test")
# plt.xlabel("Year Data Point")
# plt.ylabel("Motor Deviation (%)")
# plt.show()



# fig = plt.figure()
# plt.plot(years, nominal_dist.T,'.-',label="True Distance")
# plt.plot(years, true_dist.T,'.-',label="Calculated Distance")
# fig.text(.3,.5,f"Mean Error: {np.mean(perc_dev):.2f}%",color="r")
# plt.title("Hardware-Software Integration Accuracy Test")
# plt.xlabel("Year Data Point")
# plt.ylabel("Motor Movement (mm)")
# plt.legend()
# plt.show()

X = [f'{i+2015}' for i in range(8)]
Ytrue = nominal_dist.T
Zexpec = true_dist.T

X_axis = np.arange(len(X))

Ytrue = nominal_dist.T
Zexpec = true_dist.T

# plt.figure()

# plt.title("Hardware-Software Integration Accuracy Test")
# error = percentage_error(nominal_dist, true_dist)
# plt.suptitle(f"Sample Dataset #1 ({error:.2f}% Error)",fontweight="bold")
# plt.legend()
# plt.show()

rocket_data2 = [467.1, 203.2, 169.6, 224.4, 112.2, 470.3, 500.2, 96.5]
raw_data2 = [5562, 2378, 2083, 2686, 1322, 5689, 5962, 1128]
true_dist2 = [_map(i,0,6000,0,500) for i in raw_data2]

print(true_dist)
```

```
true_dist2 = np.array([true_dist2])
rocket_data2 = np.array([rocket_data2])
nominal_dist2 = rocket_data2

error2 = percentage_error(rocket_data2, true_dist2)

Ytrue = nominal_dist2.T
Zexpec = true_dist2.T

df = pd.DataFrame({"True Values": [467.1, 203.2, 169.6, 224.4, 112.2, 470.3,
500.2, 96.5],
                   "Expected Values": [_map(i,0,6000,0,500) for i in
raw_data2]},index=years)
ax = df.plot(kind='bar', color=colors, legend=False, rot=0)

ax.legend()

plt.title("Hardware-Software Integration Accuracy Test")
error = percentage_error(rocket_data2, true_dist2)
plt.suptitle(f"Sample Dataset #2 ({error:.2f}% Error)",fontweight="bold")
plt.legend()
plt.xlabel("Year Data Point")
plt.ylabel("Motor Deviation (mm)")
plt.show()

# Dataset #3
# 2015,1268 //17.8cm
# 2016,3146 //33.0cm
# 2017,1527 //19.7cm
# 2018,3228 //34.2cm
# 2019,5494 //53.2cm
# 2020,2049 //24.9cm
# 2021,4746 //47.0cm
# 2022,1327 //18.0cm

rocket_data3 = [107.8, 263.0, 128.7, 271.2, 457.0, 175.0, 397.3, 111.3]
raw_data3 = [1268, 3146, 1527, 3228, 5494, 2049, 4746, 1327]
true_dist3 = [_map(i,0,6000,0,500) for i in raw_data3]



true_dist3 = np.array([true_dist3])
rocket_data3 = np.array([rocket_data3])
nominal_dist3 = rocket_data3
error3 = percentage_error(rocket_data3, true_dist3)
```

```
true1 = nominal_dist.T
expec1 = true_dist.T


true2 = nominal_dist2.T
expec2 = true_dist2.T

true3 = nominal_dist3.T
expec3 = true_dist3.T

print("TRUE 1:")
print(true1.flatten())
print("EXPECT 1:")
print(expec1.flatten())

print("\nTRUE 2:")
print(true2.flatten())
print("EXPECT 2:")
print(expec2.flatten())

print("\nTRUE 3:")
print(true3.flatten())
print("EXPECT 3:")
print(expec3.flatten())

df = pd.DataFrame({"True Values": [107.8, 263.0, 128.7, 271.2, 457.0, 175.0,
397.3, 111.3],
                   "Expected Values": [_map(i,0,6000,0,500) for i in
raw_data3]},index=years)
ax = df.plot(kind='bar', color=colors, legend=False, rot=0)

ax.legend()


plt.title("Hardware-Software Integration Accuracy Test")
plt.suptitle(f"Sample Dataset #3 ({error3:.2f}% Error)",fontweight="bold")
plt.legend()
plt.xlabel("Year Data Point")
plt.ylabel("Motor Deviation (mm)")
plt.show()
```

APPENDIX H: BILL OF MATERIALS

Item	Quantity	Unit Cost	Price
Ultrasonic Sensors (x3)	1	$7.99	$7.99
Resistors (x2000)	2	$10.99	$21.98
LEDs (x900)	2	$12.99	$25.98
RGB LEDs (x100)	1	$7.96	$7.96
LCD Screens (x4)	2	$10.99	$21.98
4-Digit 7-Seg Displays (x20)	2	$11.49	$22.98
OLED Screens (x2)	2	$6.99	$13.98
Clock	1	$11.99	$11.99
ESP-01 Wifi (x6)	2	$10.99	$21.98
SD Card Module (x5)	1	$6.99	$6.99
16-Channel Mux (x10)	1	$9.99	$9.99
Linear Actuator + Rail Guide	1	$121.00	$121.00
24" x 48" Blank Canvas	1	$34.99	$34.99
Wood (x8)	8	$5.00	$40.00
16GB SD Cards (x5)	1	$19.00	$19.00
Arduino Mega (x2)	2	$22.99	$45.98
Shift Registers (x50)	2	$6.99	$13.98
22 Guage Wires (x6)	1	$14.94	$14.94
Motor Driver	1	$10.99	$10.99
Stepper Driver	1	$22.99	$22.99
24V Power Supply	1	$18.95	$18.95
Portable Power	1	$14.99	$14.99
2.4 Ghz Wifi	1	$22.98	$22.98
3D Prints	----	----	$2.59
Neopixel	1	$18.99	$18.99
7 Seg Disp	2	$10.99	$21.98
Total	----	----	$598.15

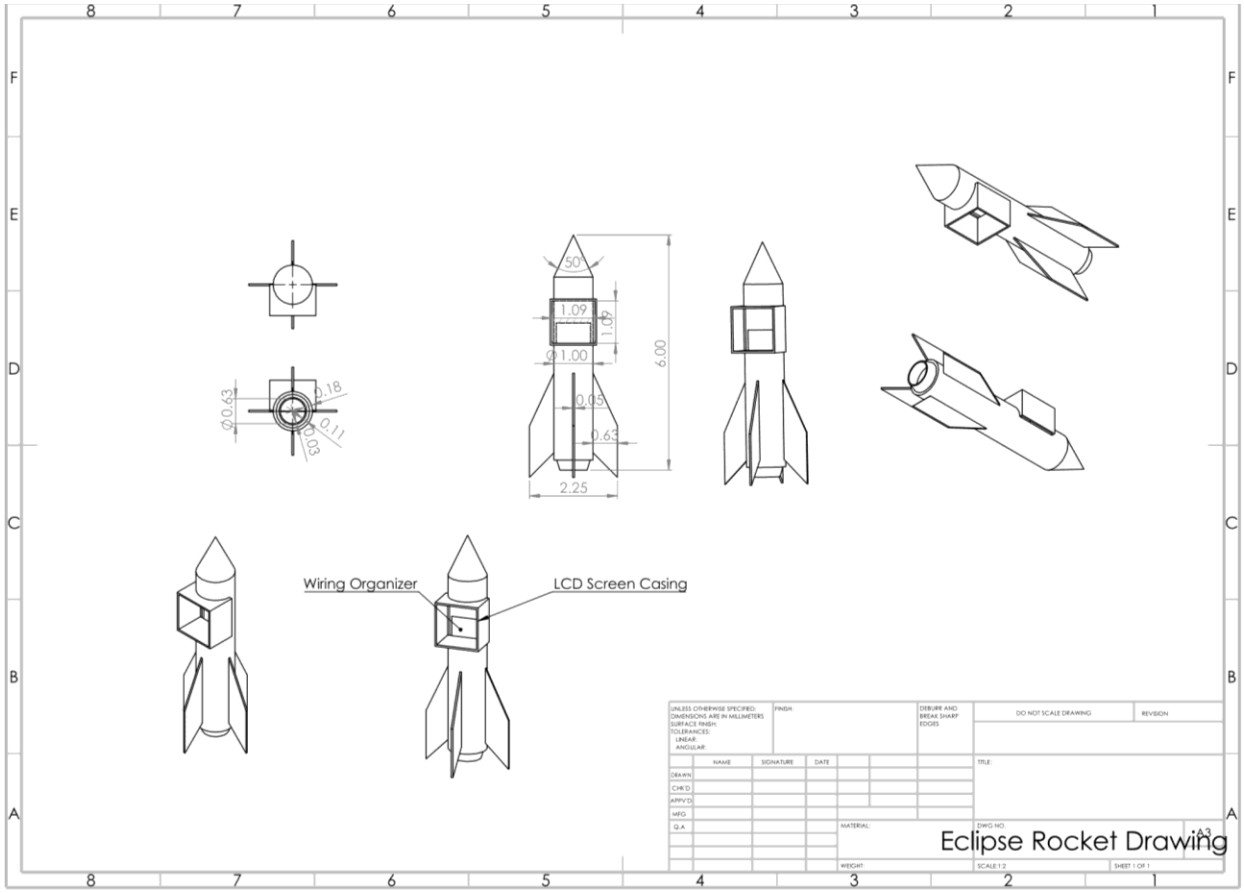
3D Prints	
Rocket	$0.78
Arm	$0.83
Globe	$0.84
Lock and Key	$0.12
Car	$0.02

Summary	
Total Budget	$500.00
Remaining	**$0.72**

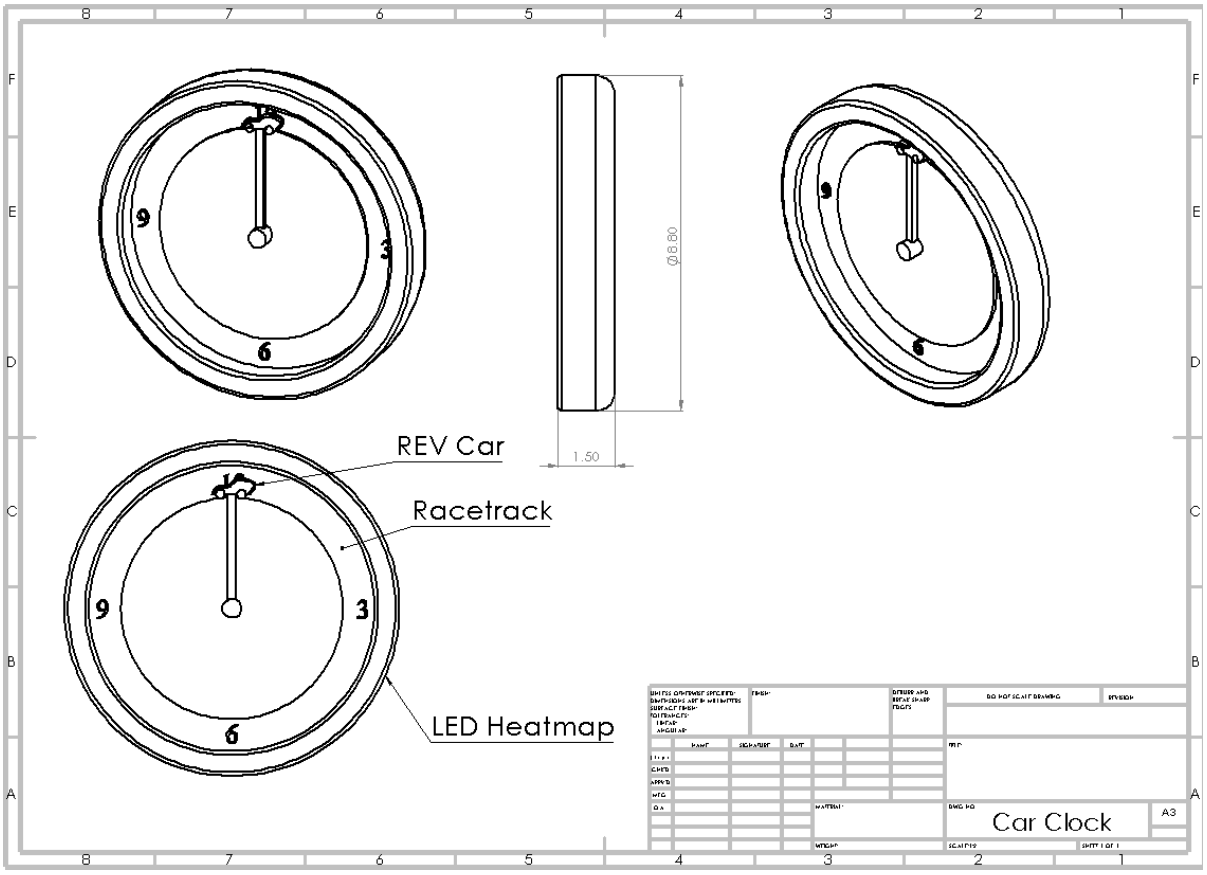
Returning Extraneous Items			
	Item	*Quantity*	*Unit Cost*
Return	LEDs (x450)	2	$21.98
	7 Seg Display (x	2	$22.98
	OLED Screen	1	$6.99
	16-Channel Mux	1	$9.99
	Motor Driver	1	$10.99
	Shift Register	1	$6.99
	RGB LEDs	1	$7.96
	Resistor Pack	1	$10.99
Refunds	----	----	$98.87
Remaining	----	----	$0.72

APPENDIX I: CAD Models

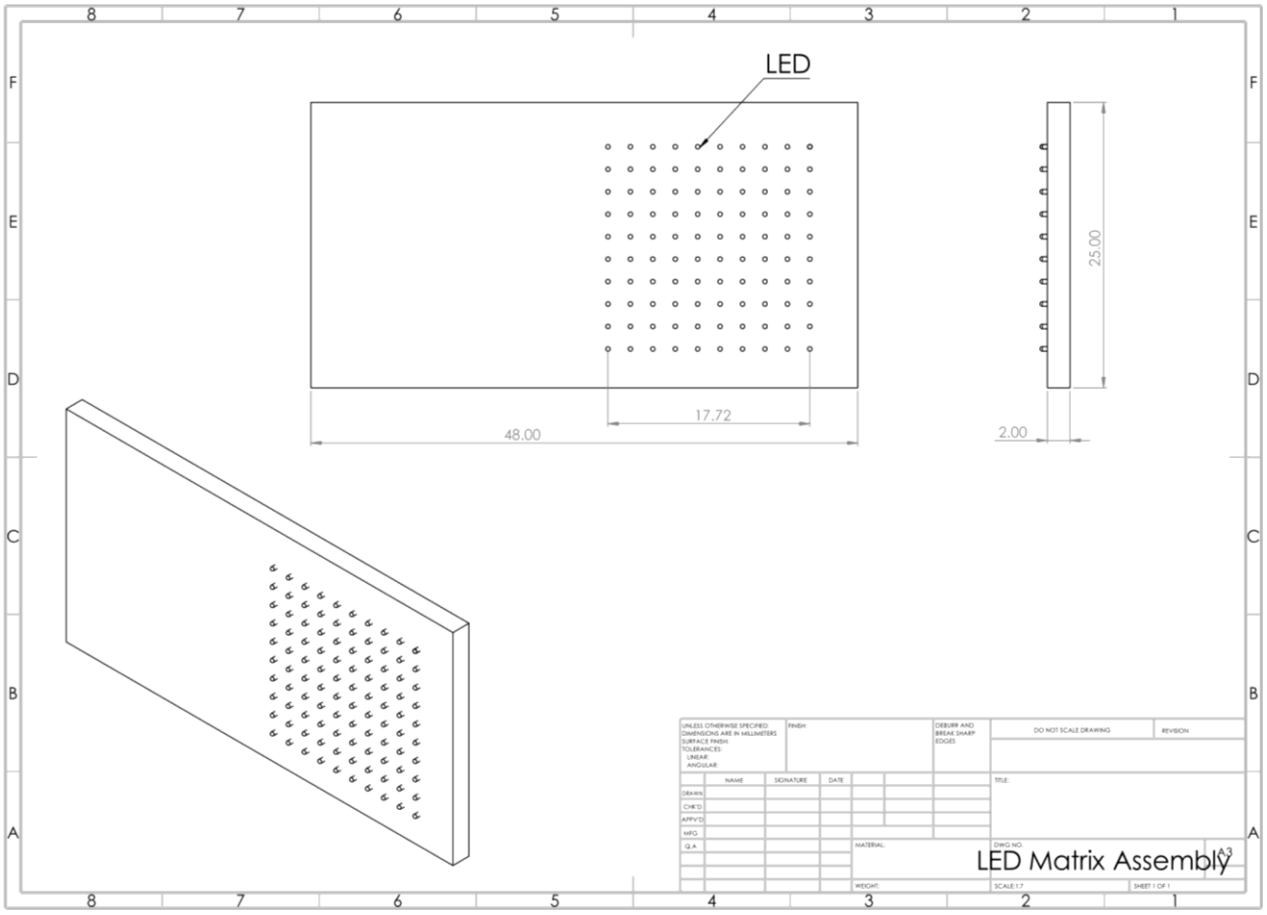
Rocket



Clock



LED Matrix



Data Display