Capstone Design Report

Project Title: STEM Playground

Team 4

Team Name: Sadron

Team Members: Kyle Saul, Angelo Dallas, Andrew Grcic, Ellie Rannebarger, Matthew DiSanto

Electrical and Computer Engineering Department

College of Engineering

Ohio State University

ECE 3906 and ECE 4905

Date: 02/19/2024

Document Change Notice

Date	Change
02/19/2024	Initial Release for review
02/19/2024	Added Chapter 1: Problem Definition
03/01/2024	Added Section 1.2: Design Analysis and Concepts
03/01/2024	Added Section 1.3: Preliminary Design
03/04/2024	Added Chapter 7: Project Management
03/06/2024	Added Chapter 2: Detail Design
04/09/2024	Revisions to Chapter 2: Detail Design
04/17/2024	Revisions to Chapter 2: Detail Design
04/23/2024	Revisions to Chapter 2: Detail Design
10/23/2024	Added Chapter 3: Build Process
11/26/2024	Added Chapter 4: Test Plan and Results
11/26/2024	Added Chapter 5: Redesign, Lessons Learned, Compromises, and Future
11/26/2024	Added Chapter 6: User Manual

Executive Summary

Students in grades 6-12 should receive more proper learning experiences in the STEM field. Furthermore, countless students go through their entire pre-college schooling without having the opportunity to gain experience/see how integrated systems work. Electrical Engineering is more than ninety percent men, and people of color represent less than a third of the demographic. By exposing younger generations to STEM projects in their youth, they are more likely to display an interest and understanding in the field. This creates a foundation to bridge the gap of minorities in STEM careers and come in with basic knowledge of elementary concepts.

The team will be creating a STEM playground to instruct students on how integrated systems work. We will achieve this by utilizing a microcontroller to handle the I/O devices and programmed to function in specific ways. The device will output specific sounds, light, motion, etc. depending on the user input given. The inputs will be given by the students in ways such as a web UI, buttons, switches, etc. The UI will interact in a bidirectional way. This means not only will physically changes on the board be viewable on the UI, but interactions in the UI will also be displayed on the board. The students will learn how the device works "under the hood" with the provided documentation and learning tasks. Once the student understands the desired concepts, they can use the playground however they desire.

This problem will not be tackled alone. Along with the members of the Sadron team, the problem will receive additional assistance from the project sponsor Mark Morscher, and stakeholders Alan Gilbert and Clayton Greenbaum. Also, the group will receive help from the entire OSU K-12 STEM outreach program and an additional budget of $500 provided by The Ohio State University. In conclusion, Sadron will address the problem of the lack of STEM programs provided to students in grades K-12 with the STEM Playground device. This device will utilize a microcontroller with specific inputs and outputs to help students learn about integrated systems. Overall, the goal is to bring awareness of the STEM program so students can make a well-rounded decision on Career paths to pursue.

Table of Contents

List of Figures and Tables

Chapter 1: Problem Identification and Preliminary Design

1.1 Problem Definition

As Mark Morsher (STEM Outreach POC), I want the STEM Outreach program to have access to better resources to serve 6-12 grades so that students of this age group can interact with projects closer to their level of competency.

As Dr. Anderson (STEM Outreach POC),

I want…

Projects that illustrate more complex concepts of STEM

Projects that illustrate integration of subsystems

Projects that illustrate web integration with hardware

So that…

Students beyond a fundamental understanding of STEM can continue to have

their horizons broadened

As a student, I want to learn about STEM programs in high school, so that I can better choose my path when going to college.

As a student, I want to see how sensors, output devices, and hardware work, so that I can better understand how integrated systems function.

As a teacher, I want my students to partake in the OSU STEM outreach program, so that they can have an interactive experience with integrated systems to help them learn easier.

As the team, we want to deliver our knowledge of integrated systems to grades 6-12 students, so that we can help create more engineers out of the younger generation.

As a parent, I want my child to experience all the possible learning opportunities, so that they can have a better idea of what they want to do/become in the future.

As an educator (homeschooling parent w/o STEM experience), I want a kit I can take my students (my literal kids) through from beginning to end without prior experience, so that we can learn about complex systems integration.

As an enthusiast, I want a kit I can progress through mostly on my own, so that I can learn the concepts regardless of having an organization finding me.

Students in grades 6-12 should receive more proper learning experiences in the STEM field. Furthermore, countless students go through their entire pre-college schooling without having the opportunity to gain experience/see how integrated systems work. Electrical Engineering is more than ninety percent men, and people of color represent less than a third of the demographic. By exposing younger generations to STEM projects in their youth, they are more likely to show interest and understanding in the field. This creates a foundation to bridge the gap of minorities in STEM careers and come in with basic knowledge of elementary concepts.

There are many stakeholders related to this project from internal to external stakeholders. Internally, our stakeholders consist of Alan Gilbert and Clayton Greenbaum who are additional help to the team. As well, Mark Morscher who is the project sponsor. Externally, the stakeholders consist of our customers, i.e., the students and teachers that will use our product. Along with the students and teachers, the schools will be a large external stakeholder to allow us to provide their staff and faculty with STEM playground.

The project has an extensive list of needs. Some are necessary, while others would be nice to have but are not a priority. Starting with the necessities, a microcontroller with a viable power source with the ability to control a single input and output (I/O). This will be the bulk of the physical requirements for the kids to experiment with. To interact and see more details about the microcontroller, a bidirectional web user interface (UI) will be also needed. A 45-minute lesson plan needs to be created to walk students through the project process. One of the top nice to haves that we hope to implement is multiple I/O devices. An automated condition configurator, sensor driven outputs, custom PCB, and 3D printed cases are other features we hope to add. Along with these, battery power would be a plus.

Our current project faces several constraints which include power, environmental/sustainability, safety, size and weight, cost/budget, time, quality, security, and social factors. The microcontroller must maintain a constant power source throughout 45-minute lessons. It is necessary that the microcontroller minimizes energy consumption and waste to support environmental sustainability. Safety protocols must follow the National Electrical Safety Code (NESC) and the Institute of Electrical and Electronics Engineers (IEEE) guidelines to ensure playground safety for students. The playground's size and weight should facilitate easy transportation to schools and be manageable for student use. The research and design (R&D) budget is $500, with a stipulation at the final playground design cost no more than $50 per board. The STEM playground lessons must fit within a 45-minute class period. Quality standards dictate that the playground must withstand multiple uses and function properly over time. Security measures must be implemented to ensure the web UI is accessed safely, with code integrity maintained to prevent manipulation by users. Furthermore, the playground should offer an engaging and enjoyable experience for students while enhancing learning and comprehension of the material.

Safety is number 1 in the OSU STEM sponsor's and team's priorities. To minimize injury risk, dangerous electrical components must be covered to minimize injury risk. Visual inspection and multimeter probing will be used to verify the device is safe. The possibility of a custom PCB implementation and a 3D-printed case could also help minimize risk. An interactive UI webpage with bidirectional student input for function viewing and communicating with the microcontroller is required. Verification entails ensuring the webpage facilitates control and monitoring of I/O across various configurations. Sensors play a crucial role as inputs for the microcontroller, with feasible options including temperature, proximity, or camera sensors. Output devices enable students to visualize the process leading to desired outputs, with outputs potentially being shown as lights or sounds.

Table 1: Market Requirements Table

Description of Need / Design Parameter	Justification	Requirement / Constraint	Verification Methodology	Comments	Final Assessment
Safety	Safety is the number one priority for the OSU STEM sponsor and the team	Ensure dangerous electrical components are covered and injury risk is minimum	Visual inspection and multimeter probing	Possibly use of custom PCB and/or 3D printed case	3D case is designed but not printed. The structure of the project is safe for student use and all I/O has been tested.
Interactive UI Webpage	Use a bidirectional webpage for students to input certain functions to view outputs	Must work with microcontroller to provide a bidirectional input and output communication	Ensure webpage can be used to control and monitor I/O	Webpage must work for each different configuration	Webpage communicates effectively and refreshes at a fair level. Memory usage is a concern though.
Webpage Graphics	Use of graphics better helps students to see correlation between energy and devices.	Must be within memory requirements of webpage and stay for future use.	Image shows up and changes based on input/output.		Graphics have been added to a majority of the I/O individual pages, however currently some of the images used are on separate webpages on internet.
Sensors	Take advantage of sensors for the students to use as a form of input to the microcontroller	Microcontroller must react to the specifications of the sensors	Experimentation and research	Change the type of sensor such as temperature, proximity, camera, etc.	All sensors have been tested to communicate with the webpage and microcontroller.
Output devices	Make outputs so the students can see/visualize how the functions got	All pieces of the project must work together to achieve a desired output	Testing and analysis	Have the output be some form of a light or sound	The final subsystems we put together work separately and jointly.

	to that desired output				
Network Connectivity	The network and the devices should be robust and simple to setup/connect to	Student must be able to connect to the proper microcontroller ID to provide communication between devices	Verify connection between students' devices and microcontroller is reliable	Microcontroller hosts website, each browser connecting to the unique microcontroller	Router was used to connect devices together.
Security	Security for both the user and the network are important to ensure the system is secure	A form of safety measure to protect the network from tampering	Protocol and specification sheets research	Will have to pick a device with built in security	The device network is isolated from all others. The communication channels operate on trust but require sophisticated manipulation that yields unimpressive effects.
Power	Rechargeable power module that lasts at least 10 uses.	Needs to provide a constant 5V output and is rated for 500mA.	Test voltage and current under load with multimeter.	Need to figure out type of batteries and converter to be used.	Found a setup with converter and lithium batteries that meets the needs of the project. However, BMS needs a substitute.

The Market Requirements table is essential to what the team needs to build moving forward. It gives an overview of the most important parts of the project and what should be focused on.

Microcontrollers are versatile technological devices, making useful applications in diverse fields. However, their widespread use contributes to energy consumption and waste generation, raising concerns about environmental impact. To address this issue, several solutions emerge. Firstly, adopting renewable energy sources like hydro, solar, or wind power can shift microcontrollers to cleaner energy alternatives, leveraging the environment for sustainable operation. Additionally, selecting energy-efficient microcontrollers like the ESP32 can significantly reduce power consumption, extending battery life and lessening environmental impact. Writing high-quality code, utilizing software tools such as Visual Studio Code for code analysis, further optimizes microcontroller performance, minimizing power requirements.

Despite the environmental challenges posed by electronic devices, the project's primary goal is to cultivate STEM interest in young minds. By inspiring kids to explore STEM fields, the project aims to contribute to a growing community of problem solvers who can address the environmental costs associated with our increasing reliance on electronics. Focusing on outreach, the project plans to distribute kits to schools in need, particularly those in inner cities lacking STEM programs. Emphasizing accessibility, the project ensures that students unfamiliar with hardware or software can easily grasp its concepts. Moreover, the project aims to kindle interest in a unique and engaging manner, sparking curiosity and passion for STEM subjects among students. Through these efforts, the project aspires not only to mitigate environmental impacts but also to foster diversity and inclusivity within the STEM community, essential for tackling complex challenges.

Table 2: Comparison Matrix

Other Possible Solutions	**Key Features**	**Cost**	**Counter Argument**
Micro:bit	Self-contained board with possibility of modularity.	$20 “Go” kit.	*Coding* **Focused**.
Arduino Kit	Electronic, programming, and coding basics [4].	$70 Student Kit.	Must install Arduino IDE, cost, lesson time [3].
Inventr.io	Lesson plan with knock off Arduino MCU, sensors + actuators, wires.	$97 w/o discount.	Very self-guided when using their “30 days lost in space” lesson plan, [4].

The Comparison Matrix table describes other solutions and why they were not suitable for this project.

There are many risks associated with the STEM Playground project. Many of these risks are purely based on the unfamiliarities that come with a project in an unknown field. Some of these are learning a new coding language, ensuring reproducible cost with budget, and creating a lesson plan. Others are complex such as creating bidirectional web interface, writing bug free instruction for the microcontroller, and the implementation of various sensors. Through various mitigation strategies, the team aims to significantly reduce these risks. Included in Appendix A is a risk analysis table, an entire breakdown of the total risk each of these incur and their mitigation technique.

1.2 Research and Initial Design Ideas

1.2.1 Microcontroller Selection

Espressif is the designer of the very common ESP32 Microcontroller Unit (MCU)/System on Chip (SoC). At first glance this MCU seems to provide all functionality we are looking for, but there are other options to consider. Espressif has five categories of MCUs with eight total options of varying capabilities. The first version of this chip was the ESP8266 which has the necessary Wi-Fi module to make this project feasible, but it lacks a Bluetooth module. The Bluetooth module might be necessary to enable a meshed network approach but that has yet to be determined. A major issue with the ESP8266 is its lack of physical security. All data and memory, such as Wi-Fi passwords, can be read and written to with physical access to the board. For this reason, Espressif made the ESP32 which comes with hardware encryption modules and memory storage that cannot be read from with just physical access to the board. The ESP32 base model and its variants all come with this secure ability to support an Internet of Things (IoT) concept. A notable variant is the ESP32-S2. This chip has extra GPIO pins compared to the ESP32 and in some ways has more capability and in other ways it is the more economic choice with less memory space.

One of the most important factors to consider is how widespread each of these boards is with published information on how to work with these boards and use their features. The ESP32 as a secure platform seems to have the most information to aid in this project's development. Once learning how to program this style of board with the ESP32's breath of information it should be easier to upgrade to some of the other variants if the need arises.

Table 3: Microcontroller Requirements Table

Requirement	Description
Cost	The overall market price of the microcontroller must be within a reasonable range to meet the expected $50 per playground
Wi-Fi embedment	The device must have a form of Wi-Fi embedment to allow bidirectional communication with the UI webpage and the microcontroller
Programmability	Programming the board to function in a specific way is required. Therefore, the board must be easily programmed and consistently run the program
Compatibility	The microcontroller must be able to handle specific input and output devices. It must work with multiple different inputs and outputs (3 for each)
Low Power Mode	A low power mode for the microcontroller is a requirement to extend the battery/power life of the system

Market requirements for the microcontroller include a reproducible cost less than $50, WIFI communication ability, easily programmable, compatible with multiple inputs and outputs, and have a battery that promotes longevity of the microcontroller.

1.2.2 Battery Power

- Alkaline Batteries: (~$4/6)
    - Alkaline batteries typically provide 1.5V each, which isn't sufficient to directly power the ESP32.
    - Utilize a DC-DC Boost Converter (step-up) to raise the voltage to 3.3V from two alkaline batteries. This ensures high efficiency and utilizes full battery capacity.
    - Example: Utilize a boost converter like [MT3608 DC-DC Step Up Boost Converter Module] to step up the voltage.
    - Input Voltage Range: 2V to 24V, Output Voltage: Adjustable (typically set to 3.3V for ESP32), Output Current: Up to 2A, Efficiency: Up to 93%, Size: Compact module with screw terminals for easy connection.
- Lithium Iron Phosphate (LiFePO4) Battery: (~$80 (they're big))
    - LiFePO4 batteries provide a relatively stable voltage around 3.2V, suitable for powering the ESP32 directly.
    - Available in AA package for convenient use.
    - Ensure compatibility and proper connection to the ESP32.
- Lithium-ion (Li-ion) Battery: (~$6)
    - Li-ion batteries can power the ESP32 when connected properly.
    - Connect the positive terminal of the battery to the Vin pin and the negative terminal to a GND pin on the ESP32.
    - Use a voltage regulator (e.g., MCP1700-3302E) to stabilize the output voltage at 3.3V.

Battery Charging

- Battery Charger Module (TP4056): (~$6)
    - For rechargeable batteries, consider using a TP4056 battery charger module.
    - This module charges the battery and prevents overcharging, ensuring safe operation.
    - LED indicators (red for charging, blue for fully charged) provide visual feedback.
    - Properly connect the charger module to the battery and ESP32 for efficient charging and power delivery.

Solar Power

- Solar Panels: (~$7)
    - Solar panels can be utilized for energy-efficient and remote applications.
    - Connect solar panels to a TP4056 charger module to charge the battery.
    - The charged battery then powers the ESP32 through a voltage regulator.
    - Solar panels typically output around 5V to 6V with direct sunlight.
    - Ensure proper sizing of solar panels based on energy requirements and environmental conditions.

USB Power

- Micro USB Cable: (~$10 (unclear, may be included in ESP32))

- For simple and straightforward power during development and testing, connect the ESP32 to a USB power source using a micro-USB cable.
- Ensure the power source can provide sufficient current to the ESP32.

Table 4: Power Requirements

Requirement	Description
Cost	Batteries need to be relatively cheap to make sure each playground doesn't surpass $50. It is plausible to investigate rechargeable batteries to save cost and waste
Power	The microcontroller needs at least 5V to operate at full capacity. Therefore, the batteries must be selected or tied together to reach this voltage
Size	The batteries need to be small enough to not cause any size or weight issues with the playground. Must be smaller than a 4x4x2 inch area
Long lasting	Batteries must last for multiple uses of the playground before needed recharged or replaced

Market requirements for power source and charging option regarding the microcontroller.

Table 5: Power Comparison Matrix

Criteria	Weight	Alkaline Batteries	Weighted score	LiFePO4	Weighted score	Li-ion	Weighted score	Battery Charger	Weighted score	Solar Panels	Weighted score	Micro USB Cable	Weighted score
Voltage Compatibility	5	3	15	5	25	3	15	2.5	12.5	3	15	5	25
Cost effectiveness	4	5	20	0	0	5	20	5	20	4	16	3	12
Size and Convienience	3.5	3.5	12.25	4	14	3.5	12.25	1	0	3	10.5	0	0
Safety and Reliability	3	3	9	2.5	7.5	2	6	4	12	2	6	3	9
Longevity	3	1	3	2	6	1	3	4	12	3.5	10.5	4	12
Score			59.25		52.5		56.25		56.5		58		58

As shown, the Alkaline battery had the highest score, thus was chosen as the choice for power source in conjunction with the buck converter.

1.2.3 Printed Circuit Boards (PCB)

Utilizing a PCB in our project will be utilized in the later stages of development. PCB as a step in a products development comes after prototyping when a design is being finalized. A benefit PCBs provide is a cleaner and more orderly presentation of a circuit which in an educational environment is critical. When educating individuals without technical experience it's possible for little wires to loosen and a circuit to lose integrity. A PCB used as the foundation for a circuited product provides a more robust build that withstands inexperienced individuals.

The process of PCB development is standard with certain design criteria that do not deviate far from the norm. Trace width and thickness are very important considerations for ensuring safe operation with the electrical currents the board could see. To make the traces safe

for the operating currents means the conductors will remain at a safe temperature. Conductor temperature as a design constraint will make for longer lasting and more reliable boards. To determine trace dimensions the expected current ranges are required. Knowing the currents that will be carried on the traces means a well-developed circuit schematic is required. This will come as a final design is tested and analyzed.

Another important note on PCB design is the geometry of the traces, there are multiple reasons for preferring two 45degree bends when shaping out the path of a trace. A primary reason is for shortening the trace length. Shortening trace length means less voltage drop and less power loss on the board. There is also a warning for 90degree bends that when logic signals are involved there can be measured reflection off the wall seen by the signal that interferes with signal integrity, a smoother bend in the trace aids in reflection dampening.

Table 6: PCB Requirements

Requirement	Description
Temperature control	The designed PCB board needs to be able to always keep the microcontroller wires and connections cool and safe
Organization	The PCB needs to organize the connections in a way that is easy to follow, and students can understand what's connected where. As well, a clear PCB is required so it can be seen in encased
Lower voltage drop	The board must shorten the trace length to make sure the voltage drop will be lower over the wire

Engineering requirements for the PCB design include prevention from overheating, easy connect ability for inputs/outputs, legibility when encased, and adequate voltage drop between trace lengths.

1.2.4 Webpage Design and Hosting

Web hosting is an important portion of the STEM Playground project. There are two distinct options to choose from, a custom hosted webpage or an open-source Minnow server. Both options have their strengths and weaknesses that must be laid out.

A custom hosted webpage was the team's first idea. It's by far the most complex option, but that does come with added benefits. The main benefits of this option are versatility and customizability. Being able to code a 100% custom webpage will allow for a tailored design and layout with the ability to adapt to later revisions. The driving force behind a host like this is WebSocket. WebSocket is a bidirectional communication protocol that allows a client and server to talk at the same time until one of them terminates the connection. This is ideal over Hypertext Transfer Protocol (HTTP), because through HTTP the client can make a request and the server response. This is directional communication and would not allow for the web UI and microcontroller to communicate as needed.

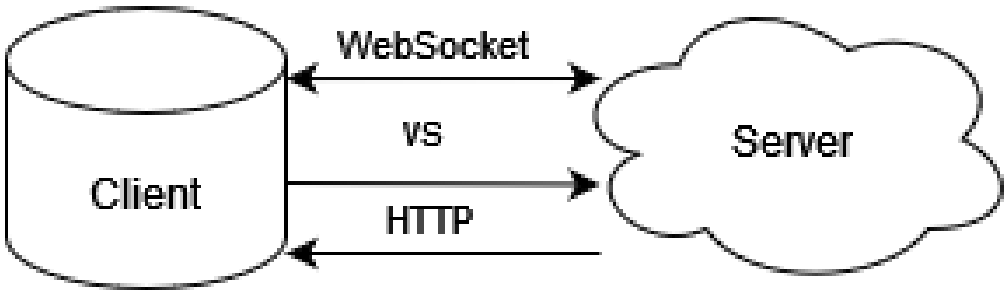


Figure 1: WebSocket vs. HTTP

The second option is to host an open-source Minnow server. It is a pre-compiled WebSocket webserver designed to be hosted on a microcontroller. This gives it a huge advantage in terms of ease of implementation. Only the installation and configuration of the server would be necessary. The configuration would entail defining the inputs, outputs, their pins, and their buttons. This would completely take away the need to design and program a web UI. The downside is the versatility is restricted and could prevent the use of certain inputs and outputs.

Webpage design involves a variety of approaches, each contributing to the overall user experience and visual appeal. One popular method is the use of responsive design, where webpages are crafted to adapt seamlessly to different screen sizes and devices. This ensures a consistent and user-friendly experience across desktops, tablets, and smartphones. Responsive design typically employs flexible grids and media queries to achieve this adaptability, enhancing accessibility and accommodating the diverse ways users access content.

Another approach is the utilization of a modular or component-based design, where the webpage is constructed from individual, reusable elements. This approach promotes consistency throughout the site, making it easier to maintain and update. Each module serves a specific purpose and can be arranged in various combinations, fostering flexibility and scalability. This method not only streamlines development but also enhances user navigation as they encounter familiar elements across different sections of the website.

In addition to these, there's the trend of emphasizing user experience (UX) and user interface (UI) design. UX design focuses on understanding and enhancing the overall user journey, ensuring that interactions are intuitive and enjoyable. UI design, on the other hand, centers around the visual elements, aiming for a visually appealing and aesthetically pleasing interface. A combination of these approaches ensures a well-rounded webpage design that caters to both functionality and aesthetics, ultimately providing a positive and engaging online experience for users.

A way to create these webpages is by using HTML which can be coded in languages such as JavaScript and Cascading Style Sheets (CSS). CSS and JavaScript are used in web page design yet have different roles [1].

• CSS is used to design webpages for better layouts for the user to make them feel more comfortable. This includes targeting various screen sizes to make the web page more responsive and applies the style to the web page elements.

• JavaScript is better at creating interactions between the webpage and the user. Specifically for a bidirectional connection between components. Also, this includes programmatic code to enhance the functionality.

How students connect to the webserver is the next problem. This comes down to 3 options:

1. Typing in the IP of the server manually
    a. Easy to implement
    b. Tedious and easy to mess up
    c. Could be confusing to students
2. QR code
    a. Simple use for students
    b. Easy to implement
    c. Requires a device with a camera
3. Custom DNS to give the server a URL
    a. Easy for all devices to use
    b. Complex to implement
    c. Might require additional hardware

Table 7: Webpage Requirements

Requirement	Description
Bidirectional Capability	The student must be able to interact with the webpage and see results on the microcontroller and vice versa.
Security	The webpage and server must be secure so that private information can't be taken when connected. As well, the schools must allow access to the server
Accessibility	The webpage must be able to be accessed from any device the student may be using such as a laptop, tablet, or phone
Interface	The interface must be created to be simple and informative and easy to traverse

Easy interaction, security, accessibility, and simple user interface are all attributes the webpage must have for this project to run smoothly for the client.

1.2.5 Microcontroller Connection

Easily connecting the microcontroller to Wi-Fi is a very important step to having a smooth learning experience for this project. One of the main concerns is not being able to connect to a network in general; whether it be caused by a network's configuration, firewall settings, or IP address conflicts. Therefore, one good option is to supply a router so that we do not have to deal with any unknown networks. This way, all the microcontrollers can be separately connected to the network.

A second option is to have one device as the host or so-called “mothership” and then connect each microcontroller to that access point. This method would also minimize some of the struggles of dealing with unfamiliar networks.

Obviously, one final way to connect the microcontrollers to Wi-Fi is to connect them to the current locations network directly. This method would require that the network’s privacy and restrictions allow this type of device, which might be hard to come by.

1.2.6 Inputs

- DS18B20 Temperature Sensor
    - It is a one-wire digital temperature sensor (requires just one data line and GND to communicate with ESP32).
    - You can get temperatures from multiple sensors using just one GPIO.
    - About $1.40
- PIR Motion Sensor (HC-SR501)
    - Detects motion based on changes in infrared light in the environment.
    - About $2.43
    - Alternative: RCWL-0516
        - Uses microwave radar to detect the presence of moving objects.
- HC-SR04 Ultrasonic Sensor
    - Uses sonar to determine the distance to an object.
    - Has a range of 0.8-157 inches, with an accuracy of 0.1 inches
    - It comes with ultrasonic transmitter and receiver modules.
    - About $1.80
- Load Cell with HX711 Amplifier
    - Electrical sensor that measures force or strain on an object.
    - About $7.50
- BH1750 Light Sensor
    - 16-bit ambient light sensor
    - 1-65535 lux
    - Communicates via I2C protocol
    - About $2.50
- Capacitive Touch Sensor
    - Need ESP32 development board
    - Don’t need to connect an external push button
    - Works by detecting electrical charge value variation on GPIO pins when someone touches the pins.
    - Very simple to use
- Buttons/switches
    - Buttons: around $0.35 each
    - Switches: around $0.80 each

1.2.7 Outputs

The outputs that the team wants to use can vary depending on how we can best deliver the lesson plan to the students. We need to keep in mind how simplistic we want to make the outputs so the students can understand the concepts but make it “fun”.

Table 8: Output Comparison

Type of output	Pros	Cons	Price	Complexity
LEDs	LEDs are an easy and reliable	Not very interesting for	Very cheap. You can purchase	Not complex. Easy to

	way for a student to start working with a microcontroller. Turning an LED on and off is where it all starts.	students to see as an output. Basically, the same as turning a light on at home for them.	over 100 LED 5 mm bulbs for $6.65 on Amazon.	configure. Can possibly make it more complex to have multiple LEDs run in a sequence.
Sound(speaker)	Something that is more interesting for a student to see as an output. Can have the inputs create multiple sounds through the speaker	Speakers are a very one directional output. They can output specific sounds and wavelengths and that's it.	Very cheap. You can purchase a speaker to wire to pins on a microcontroller for $5.45 on DigiKey.	Not very complex. Can feed the speaker different sounds based on the inputs given.
Motor	Can perform multiple "fun" outputs with a motor. The motor can be used to spin a wheel or power some interesting aspect.	The motor is more of a bridge between the inputs in the outputs, but it opens multiple possibilities.	More expensive. A DC 12V 50RPM gear motor costs roughly $2 a unit.	Very simple. Motor can be turned on manually or by receiving an analog input.
LED Indicators	Utilize a bigger LED that indicates voltages which is more based on our project goal to deliver the lesson that inputs and outputs are just voltages.	They are very similar to LEDs and perform basically the same function. Voltages might not get high enough to turn on LED	Very cheap. Around $0.95 for a single LED indicator	Very simplistic, LED turns on when a certain voltage is reached.
Counter/display	Opens multiple possibilities for specific outputs. Can show the voltage conversion from the input on the display.	Does not have a whole lot of the "fun" aspect since most displays look like calculators	More expensive. Roughly $10 for a quality display.	Most Complex of all the outputs. Team has experience with using a display but will require more research

Popular output options for the ESP32 and their pros and cons

1.2.8 Github

GitHub is a web-based platform that serves as a version control system for tracking changes in source code during software development. It facilitates collaboration among developers, allowing them to work on projects simultaneously, manage different versions of code, and merge their contributions seamlessly. GitHub is built around Git, an open-source distributed version control system. It provides a centralized hub for code repositories, issue tracking, and project management tools, making it an integral part of modern software development workflows. The team will be able to use software like this to improve productivity through collaboration on the software design portion of the project.

Key features of GitHub include:

- Version Control: Tracks changes in code and allows developers to collaborate on projects.
- Collaboration: Facilitates teamwork by enabling multiple contributors to work on the same project concurrently.
- Supports peer review processes, helping maintain code quality.
- Issue Tracking: Allows users to report and manage issues, enhancing project management.
- Branching: Permits the creation of branches for different features or bug fixes, keeping the main codebase clean.

Other web-based platforms for possible usage:

- GitLab: Like GitHub, GitLab is a web-based Git repository manager that provides version control, continuous integration, and collaboration features.
- Bitbucket: Owned by Atlassian, Bitbucket offers Git and Mercurial repository hosting along with features for code collaboration and continuous delivery.
- Azure DevOps: Microsoft's platform provides version control, build automation, release management, and other DevOps tools.
- SourceForge: An older platform that offers version control, bug tracking, and project management tools.
- GitKraken: Known for its Git client, GitKraken also provides collaboration and code review features in a visually intuitive interface.

1.2.9 Lesson Options to Guide Students

Classic Presentation

- Accompanying Google, Microsoft, or Prezi slides to illustrate design and concepts
    - Free, real-time collaboration, automatically saves, embedded videos and pictures, easy to share.
    - Students get a visual component to pair with auditory explanation.
    - Teacher/presenter can walk through each step with students.

Video Depicting how voltage/ sensors work

- PowerPoint Animated Video

- Effectively demonstrates the connection between “under the hood” concepts and communication between the input, process, and output.
- Captures students' attention and interest and gives a great introduction before getting too technical.
- Time intensive and requires a learning curve.

Physical Demonstration with Equipment

- Bring out MCU, sensors, etc. And go through each step with students.
  - Students know where to push buttons, what each component looks like, where to avoid touching, etc.
  - Someone would have to learn the process well enough to explain it to students and answer questions.

Worksheet

- Paper Print out
  - Students can answer questions to check for comprehension.
  - Physical reference sheet.
  - May seem boring or cause loss interest.

Flip Book/ Picture Book

- Fusion 360/ CAD
  - A Lego style flip book would give in-depth, detailed explanation for each step of the project.
  - Easier for teachers to help students as it resolves common questions.
  - No oral explanation may hinder interpretation or prevent questions.

Table 9: Lesson plan Requirements

Requirement	Description
Timing	The lesson plan must be delivered and completed in a period of 45-50 minutes. Which is roughly a class period.
Logic flow	The lesson must be delivered in a way that flows smoothly for teachers and OSU STEM members to teach
Understandability	After the lesson has been complete, the students should be able to understand the basics of integrated systems

Stakeholders articulated the main restrictions for the lesson plan being an under 45 lesson plan and easy to understand digestible content.

1.3 Preliminary Design

1.3.0 High Level Block Diagram

The block diagram below shows a high-level version of the team’s deliverable. An overview of how to traverse the block diagram is provided.

- The microcontroller will be powered by a form of battery that can be charged. This power will then be input into a buck converter to buck up or down the input voltage.
- The client will provide a function to the Input sensor which will then go into the microcontroller.
- Based on the input, the microcontroller will perform a specific output function to the output device.
- A PCB will be utilized to connect to the microcontroller.
- The microcontroller will be able to bidirectionally communicate with a server. The server will then bidirectionally communicate with a constructed webpage.
- Finally, the client will have access to control the webpage and is required to connect to the Wi-Fi feature of the microcontroller.

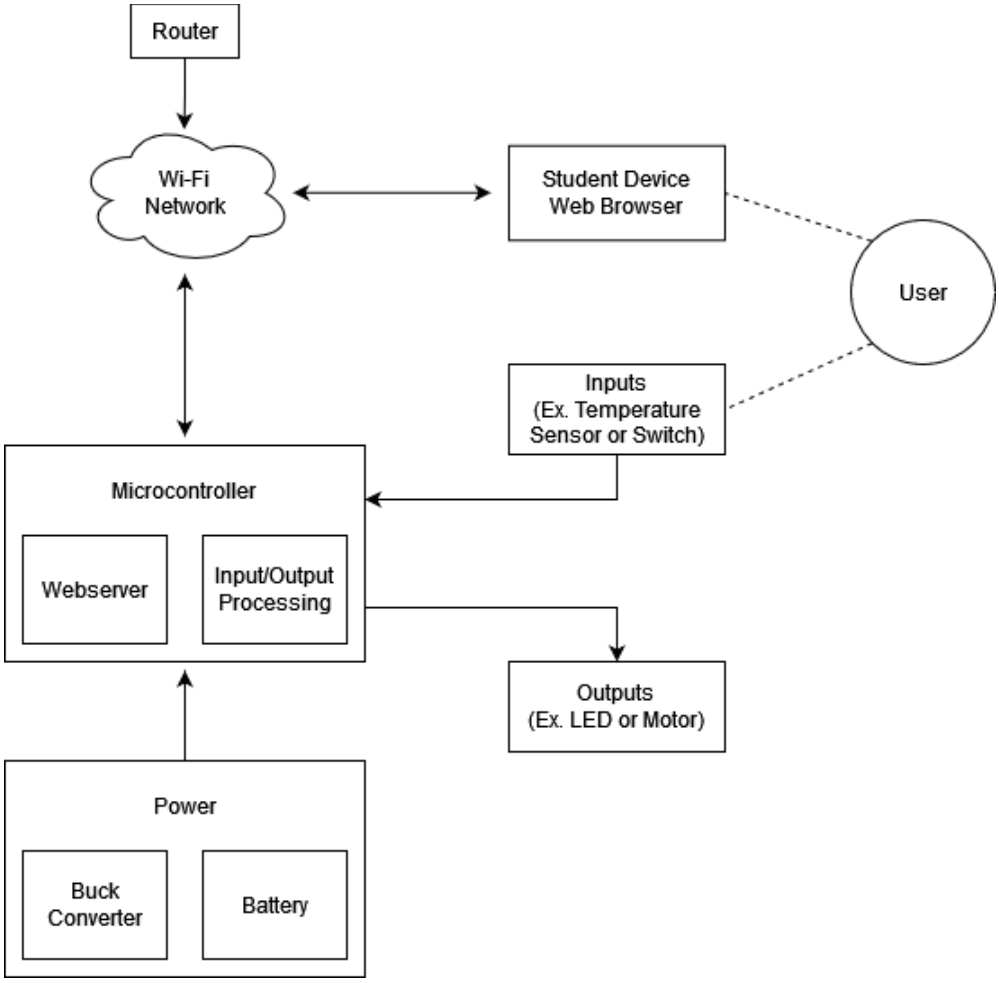


Figure 2: Block Diagram displaying how each subsystem will interact with each other

1.3.1 Microcontroller Selection

From the microcontrollers available on the market, the team has selected the base model ESP32 as the initial microcontroller moving forward with the project. The main reasons for this selection were the amount of open-source information available for the ESP32, the embedded

Wi-Fi system, multiple embedded sensors, Bluetooth connectivity, and utilizing a low power mode efficiently. Also, the prices for the ESP32 microcontroller variants are comparably affordable. Figure 3 shows some of the highlights of the board.

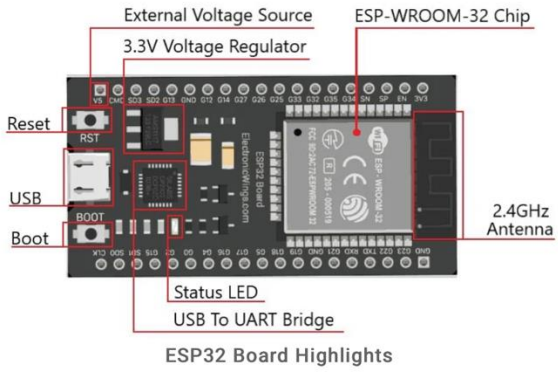


Figure 3: ESP32 Microcontroller basic specifications

Table 10: Microcontroller Decision Matrix

Criteria	Weight	ESP8266	Weighted score	ESP32-S2	Weighted score	ESP32-H2	Weighted score	ESP32-C6	Weighted score
Ease of Configuration	4	2	8	3	12	2	8	4	16
Cost	2	4	8	4	8	4	8	4	8
Security	1	2	2	3	3	3	3	4	4
Compatability	5	3	15	4	20	4	20	5	25
Programability	3	5	15	5	15	5	15	5	15
Score			48		58		54		68

Decision matrix to decide which ESP32 to use. From the results, the ESP32-C6 seems like the best option.

1.3.2 Battery Power

There are multiple ways to provide power for the ESP32 microcontroller. See section 1.2.2 for the entire battery options selection. The team has decided to move forward with the AA alkaline batteries. These batteries can be recharged to reduce waste and save money. Also, these batteries will feed into a voltage regulator to provide clean and consistent energy to the microcontroller board. On one charge, these batteries can provide the microcontroller with roughly 24 to 48 hours of constant power. Figure 4 shows a MIC5504 DC-DC voltage regulator integrated circuit (IC).

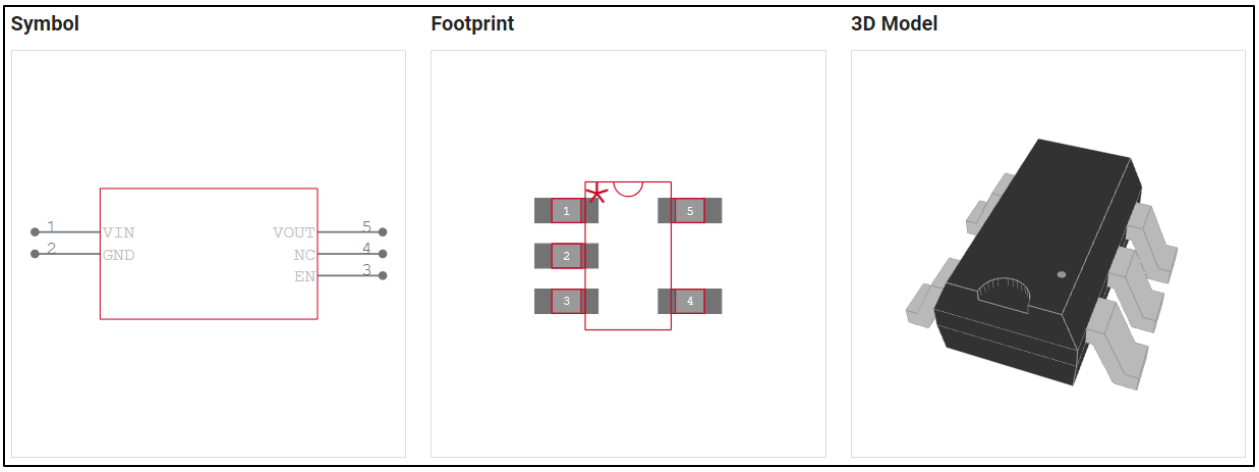


Figure 4: Linear Voltage Regulator MIC5504-3.3YM5

1.3.3 Printed Circuit Boards (PCB)

The team has decided that a PCB can be a beneficial addition to the ESP32 microcontroller. This PCB will be designed from scratch and the design will be sent to a company to build the board. The board will be used to mechanically support and electrically connect components from the ESP32 microcontroller.

1.3.4 Webpage Design and Hosting

At this time, the team is unable to decide on how the webpage should be hosted. Testing will have to be done before a decision is made. The usability, capability, and functionality of the Minnow server will need hands on testing to help better understand if it is a viable option. If the Minnow server completes these requirements, it will be used. However, if it does not, a custom hosted server will be implemented.

Table 11: Webpage/Hosting Decision Matrix

Criteria	Weight	Minnow Server	Weighted score	Custom Design	Weighted score
Ease of Implementation	4	4	16	1	4
Ease of Configuration	2	4	8	1	2
Aesthetic	1	3	3	2	2
Device Compatibility	5	3	15	5	25
Features	3	2	6	5	15
Score			48		48

Decision matrix to choose how the webpage will be made. The results show that both options are viable.

To connect to the webpage, a combination of the methods mentioned will be used. A label with the microcontroller's IP address and a QR code will be attached to the device. Using a device with a camera, students can easily scan the QR code. Devices without a camera will be required to type in the IP address manually. Time permitting, a custom DNS server can be added to give each microcontroller its own unique URL.

1.3.5 Microcontroller Connection

The team decided for simplicity that a router will be used to link all the microcontrollers. The router will provide a new network that all the microcontrollers and student devices connect to. From there each system will be provided with an IP the students can use to connect to their playground. This will allow all the setup to be mobile from location to location without any configuration.

1.3.6 Inputs

A list of inputs can be found in section 1.2.6. As of now, the team has decided to move forward with three sensors: A temperature sensor, a button/switch, and a pressure sensor. A temperature sensor will be a great way to show how devices work under the hood with voltages while also providing a "fun" factor for the students. The student will be able to heat up or cool down the sensor in various ways and will be provided with the proper documentation of what's really happening.

Next, a button is the most straight forward sensor we could utilize but should be fun for younger students to interact with. To use a button with the ESP32, one of the button's pins should be connected to a digital input pin of the ESP32, while the other pin is connected GND. Also, a pull-up resistor must be used. When the button is pressed, the state of ESP32's pin is LOW; otherwise, the state of ESP32's pin is HIGH.

Finally, a pressure (or force) sensor will be used. A force sensor is a resistor in which its resistance is in inverse proportion to how much force is given. For the force sensor, the resistance is in proportion to voltage. By connecting a pin of the force sensor to an analog input pin, the analog value can be read.

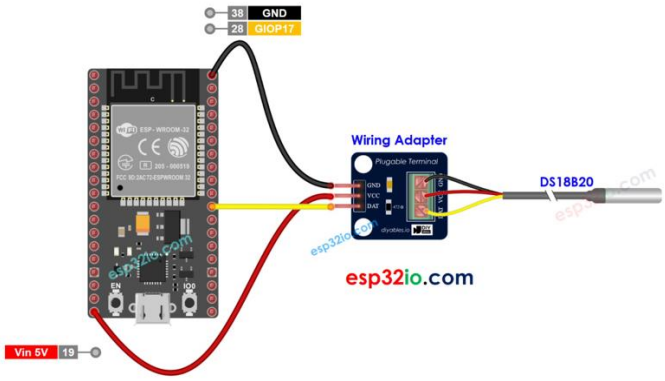


Figure 5: Temperature Sensor setup

Figure 5 shows a possible connection of a temperature sensor connected to the ESP32. As seen in the figure, it shows the pin connections, the voltage in, a wiring adapter device, and the temperature sensor probe.

Table 12: Input Decision Matrix

Criteria	Weight	Temperature Sensor	Weighted score	Motion Sensor	Weighted score	Ultrasonic Sensor	Weighted score	Force Sensor	Weighted score	Light Sensor	Weighted score	Buttons/ Switches	Weighted score
Ease of use	5	4	20	4	20	3	15	5	25	3	15	5	25
Ease of setup	4	4	16	3	12	4	16	3	12	4	16	5	20
Cost	2	4	8	2	4	3	6	2	4	3	6	5	10
Safety	3	3	9	3	9	3	9	4	12	3	9	3	9
"Fun" factor	5	3	15	4	20	3	15	4	20	2	10	3	15
Score			68		65		61		73		56		79

Decision matrix to determine the three best inputs for the project. The temperature sensor, force sensor, and buttons seem to be the best options.

1.3.7 Outputs

A list of possible outputs can be found in section 1.2.7. From this list, the team has decided to move forward with the LED indicator, motor, and counter/display options. While not shown in the table, the team has decided to add a servo motor as an output. This servo motor will allow for bidirectional connectivity between the ESP32 and the webpage. This motor will provide the “wow” factor for these students. These LED indicators will turn on when a certain voltage has been reached/tripped. This will work well with the temperature sensor input so when the student provides a specific temperature, the LED will either turn on or off. A motor will also be a beneficial output as it can ramp up and down when the voltage fluctuates depending on the specific input given. Such as the pressure sensor. Finally, the team will utilize a counter/display as an output sensor. This display can possibly show the voltage change as it updates in real time so students can see how voltage is the bones of how integrated systems work, especially for analog sensors.

Table 13: Output Decision Matrix

Criteria	Weight	LED Indicator	Weighted score	Speaker	Weighted score	Motor	Weighted score	Display	Weighted score
Ease of Implementation	4	3	12	1	4	3	12	2	8
Cost	2	5	10	2	4	3	6	4	8
Ease of configuration	1	4	4	2	2	2	2	3	3
Device Compatability	5	5	25	4	20	3	15	5	25
"Fun" factor	3	2	6	5	15	4	12	3	9
Score			57		45		47		53

Decision matrix for determining the three best outputs for the project. The LED, speaker, and display are the best options.

1.3.8 GitHub

For the coding portion of the project, the team will utilize GitHub to collaborate and work on the code. GitHub does a great job of creating repositories that can connect to coding platforms such as Visual Studio Code (VS Code). Along with this, there are multiple open-source codes that can be used by other users in GitHub. This will help the team learn new languages such as JavaScript while also creating our own open-source code so other capstone teams in the upcoming years and use our work.

1.3.9 Lesson Options to Guide Students

As far as delivering the lesson plan, the team will create a presentation that will be used in the classroom to help the students through the activity. It is still early in the project creation so a flipbook or in-hand documentation may also be provided alongside the presentation. The goal is to make the process as smooth as possible with little confusion and still delivering the desired learning experiences.

Chapter 2: Detail Design

2.1 System Diagram

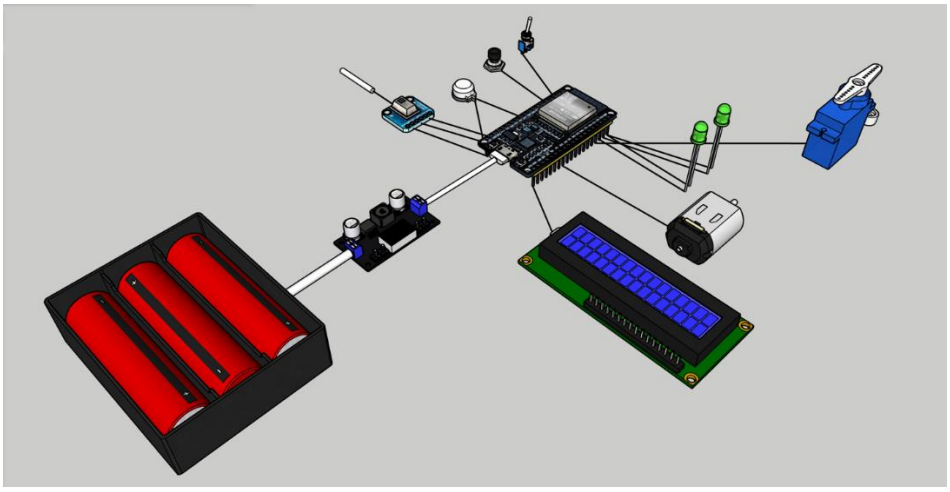


Figure 6: Simple CAD design showing the various inputs, outputs, and power supply

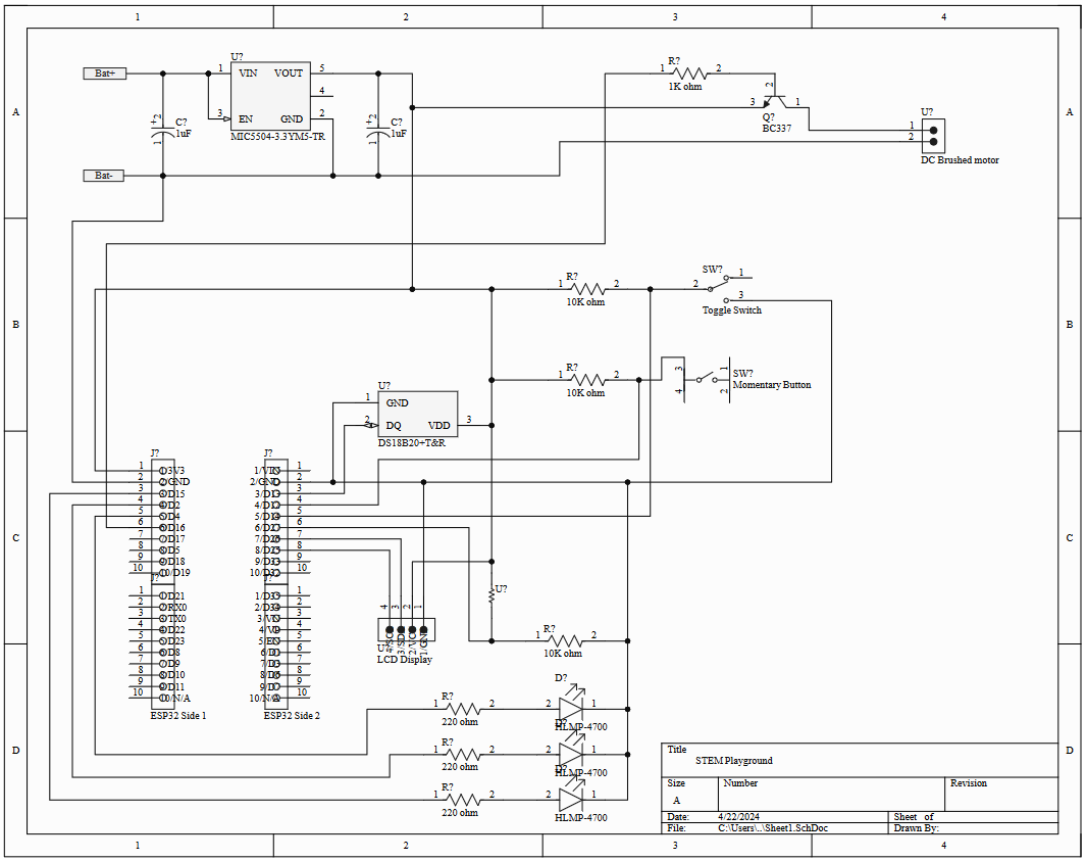


Figure 7: Electrical schematics showing the connections connected to the IO devices, voltage regulator, and microcontroller.

2.2 Hardware Design

2.2.1 Microcontroller Selection

After multiple discussions and meetings, the team has decided to advance with the ESP32-WROOM-32 microcontroller for its robust features that are well-suited for the demands of this project. The decision was influenced by its dual-core processor and integrated Wi-Fi and Bluetooth capabilities, which are necessary for a smooth connected classroom experience.

The dual-core processor of the ESP32-WROOM-32 means our system can handle multiple sensor inputs and outputs simultaneously. This enables the device to support a variety of student projects concurrently, making it possible to have more complex and varied lessons for different educational objectives and learning paces. With the built-in Wi-Fi and Bluetooth functionalities each module can easily connect to the internet and other devices. This connectivity ensures reliable communication so that students' projects are interactive and interconnected. Additionally, the extensive set of peripherals, including GPIOs, SPI, I2C, and UART interfaces, offers flexibility, making it easy to integrate a wide range of sensors and actuators. This adaptability allows us to design kits that can be customized for different subjects, ranging from physics and biology to engineering. Another advantage of the ESP32-WROOM-32 is its energy efficiency. Given that schools often operate under tight budget constraints, the power management features of the ESP32-WROOM-32 help minimize power consumption, which is essential for prolonging battery life and reducing operational costs. This consideration means that our project is sustainable over long-term use.

2.2.2 Power and Recharging

While searching for a DC-DC voltage regulator, the team encountered some hurdles with various options. As the team designs and tests the power portion of this project the board will be powered by USB connection to further development until the power module can be built.

After careful consideration of various power options, see Table 5 for comparison, the team has opted to proceed with AA alkaline batteries for several reasons. By utilizing three AA batteries connected in parallel with a DC-to-DC voltage regulator, the team can ensure a stable output voltage of 3.3V, meeting the requirements of the ESP32 microcontroller. This configuration not only provides the necessary power but also allows for easy integration with existing battery holders and compartments commonly designed for AA batteries.

Furthermore, the decision to employ a voltage regulator ensures the delivery of clean and consistent energy to the microcontroller board. The MIC5504-3.3YM5 Linear DC-DC regulator IC, depicted in Figure 4, offers the versatility and reliability needed for the project's power management requirements. An adjustable output voltage capability ensures compatibility with the ESP32 while maximizing efficiency and minimizing power loss. The estimated runtime of 24-48 hours on a single charge provides sufficient operational time for the microcontroller, offering practicality and convenience for extended use scenarios without frequent battery replacement.

In summary, the selection of AA alkaline batteries, coupled with a DC-to-DC regulator, offers a balanced combination of sustainability, accessibility, compatibility, and reliability, making it the preferred power solution for the STEM Playground project. Once the team has gotten to a suitable point of testing with products in hand, it can be determined if a PCB board-based voltage regulator would be best.

2.2.3 Inputs

The inputs the team has decided to begin implementing are a temperature sensor (KIT0021 made by DFRobot), force sensor (Tekscan A301-25), and buttons/switches. The buttons being used have the product number TS02-66-60-BK-160-LCR-D by CUI Devices, and the switches being used have the product number RA1113112R made by E-Switch. The team wants to use multiple input sensors so that students can compare and confirm that voltage is what really drives these integrated systems. Each of these sensors are great choices to show how the conversion of voltages is what makes the microcontroller perform a specific function or program. This program will cause an output device to perform a function based on what the input was.

The temperature sensor was the team's first choice for a sensor to utilize. The team has experience with using temperature sensors on microcontrollers and comprehends the underlying factors that come into play. Also, it provides a “fun” factor for the students as they can cool down the sensor or heat up the sensor in various ways. The temperature sensor has 3 pins. To connect the temperature sensor, the black GND pin will be connected to GND on the ESP32, the red V_{CC} pin will be connected to V_{CC} on the ESP32, and the yellow data pin will be connected to a digital pin on the ESP32. The wiring adapter has a built-in pullup resistor that is necessary for the temperature to work properly.

The force sensor was the team’s second choice of input sensor. This is another sensor that shows the conversion of voltage well. Creativity is required to identify a “fun” way to showcase this sensor’s attributes. Through setting up a configuration like the carnival game “high striker,” the force sensor should be able to provide a fun experience for the kids. To do this, multiple LEDs will be lined up, and depending on how hard the sensor is pressed, a certain amount of the LEDs will light up. The force sensor has 2 pins, and we will need one 10kOhm resistor. Connect pin 1 of the force sensor to the 3V pin of the ESP32 and connect the other pin to the terminal of the resistor and any ADC pin on the ESP32.

The buttons/switches were the team's last choice for the input sensor to use. This is mainly because the buttons don’t provide much amusement and interest for the students to use. However, we think it is vital for these to be included to add to the complexity of the project and for students to enjoy seeing the outputs given from these inputs. A major teaching point to these inputs is their relation to logic circuits that operate with voltage HIGH and LOW states. Only 2 of the 4 pins on the button will be needed, and we will choose 2 pins that are diagonal from each other. One pin is connected to a digital input pin of the ESP32, and the other pin is connected to GND OR V_{CC} on the ESP32. If GND is chosen, a pull-up resistor must be used, but if V_{CC} is chosen, a pull-down resistor must be used. Since the switches being used have just 2 pins as well, the setup for them is identical to the buttons, having the same choice between using GND or V_{CC}.

2.2.4 Outputs

The utilization of LED indicators, DC motor, servo motor, and displays as outputs for a microcontroller offers a multifaceted approach to visually represent the voltage aspects of the system. LEDs serve as excellent indicators due to their responsiveness to changes in voltage levels. By varying the intensity or blinking pattern of the LEDs, the team can visually convey the voltage status, whether it's within a normal range or approaching critical levels. LEDs are not only energy-efficient but also provide an intuitive and immediate visual feedback mechanism, aiding in quick diagnostics and troubleshooting.

Motors, on the other hand, are dynamic outputs that respond to changes in voltage by altering their speed or rotation. This adds a kinetic dimension to the voltage representation, allowing the team to create a tactile and audible response to voltage fluctuations. For instance, a motor running at different speeds could symbolize varying voltage levels, providing a real-time, hands-on experience for users and developers. The motor's responsiveness adds an extra layer of engagement and comprehensibility to the voltage dynamics, making it an effective tool for conveying the microcontroller's performance.

Displays, such as LCDs, are useful for presenting comprehensive voltage information in a more detailed and user-friendly format. These visual interfaces can showcase real-time voltage values, graphs, or warning messages, allowing users to grasp the voltage characteristics briefly. Integrating a display into the microcontroller system enhances its user interface capabilities, providing a more sophisticated and informative representation of voltage aspects. This visual feedback through displays facilitates a deeper understanding of the microcontroller's behavior, contributing to effective monitoring and control in voltage-sensitive applications.

In our system, LEDs are utilized as primary indicators for voltage status. The chosen LEDs will need to cover a range of voltages and be clearly distinguishable under varying light conditions. We will use high-brightness LEDs in three colors: green, yellow, and red. Green LED: Indicates that the system voltage is within the normal operating range. Yellow LED: Signals that the voltage is nearing the upper or lower limits of the acceptable range. Red LED: Alerts that the voltage has exceeded safe operating conditions and requires immediate attention.

Each LED will be connected to a corresponding GPIO (General-Purpose Input/Output) pin on the microcontroller through a current-limiting resistor, typically around 330 ohms, to prevent excessive current flow. The blinking patterns and intensity modulation will be controlled via PWM (Pulse Width Modulation) signals, which will be programmed into the microcontroller. This allows for adjustments in brightness and blinking rate without additional hardware components, also conservatively managing power consumption.

DC motors and servo motors will serve as active outputs to represent the dynamic changes in voltage through kinetic feedback. The DC motor's speed will vary with voltage fluctuations, whereas the servo motor will change its position angle proportionally to voltage variations.

DC Motor: A standard 6V DC motor will be used. It will be interfaced with the microcontroller, which allows for both speed control and directional change. Voltage variations will directly affect the duty cycle of the PWM signal used to control the motor speed.

A micro servo motor (e.g., SG90) will be employed to indicate voltage levels through discrete angular positions. It will be controlled using a PWM signal generated by the microcontroller, capable of achieving precise positional control.

Both motors will include necessary protective circuitry, such as capacitors to smooth out any voltage spikes during operation.

For comprehensive voltage monitoring and display, an LCD panel will be incorporated into the design. A 16x2 LCD display will be used, which provides sufficient space to display voltage values and warning messages. The display will be interfaced with the microcontroller.

The LCD will display real-time voltage values updated at a frequency of 1 Hz, which allows users to monitor the system voltage effectively. Additionally, graphical representations like textual warnings ("Voltage High!" or "Voltage Low!") can be programmed to appear based on predefined voltage thresholds.

2.3 System and Software Design

2.3.1 Webpage Design and Hosting

The decision regarding how the webpage would be designed and hosted was uncertain for quite some time. Originally, Minnow server was going to be implemented, barring it achieved the functionally required. Minnow server is a barebones webserver hosting a configurable interactive webpage designed to interact with microcontrollers just as the team intends. Documentation on the exact abilities of the Minnow server is lacking, leading to its true capabilities [to be unknown. After multiple issues with errors and university technology services, the plan to use the Minnow server was redacted. Instead, a custom designed webpage will be used.

To host our custom webpage, we still need a webserver. Luckly there are many opensource libraries out there to do this. The team opted to use the ESPAsyncWebServer library. This specific library was chosen because of the multitude of tutorials online. Also because of its easy-to-use API, multiconnection support, and speed. With these features, it will be easy to integrate and have speedy response times.

The designing of a custom webpage will be a difficult undertaking. This is mostly because the team does not have experience with HTML, JavaScript, or CSS, the program languages used in web design. HTML is the main language used to house the webpage. CSS is used to make complex and elegant looking graphical and textual based features. The brains of the webpage that manage the data is JavaScript. An example of how all three languages work together would be a thermometer on a webpage. HTML creates the basic blank white page, CSS adds the image and colors to display the thermometer, and JavaScript gets the value to display the temperature. Using this setup, inputs/outputs will be set up through the webpage as a graphical interface.

Getting the data from the microcontroller to the webpage and vice versa is also needed. This can be completed through the use of HTTP and WebSocket protocols. HTTP can be used to get static information from the microcontroller, whereas WebSocket provides real time updates bidirectionally. So, while WebSockets can provide a continues data stream back and forth between the microcontroller and webpage, HTTP only provides a single response that only updates after the page is refreshed. For this reason, a majority of the communication will be done using WebSocket.

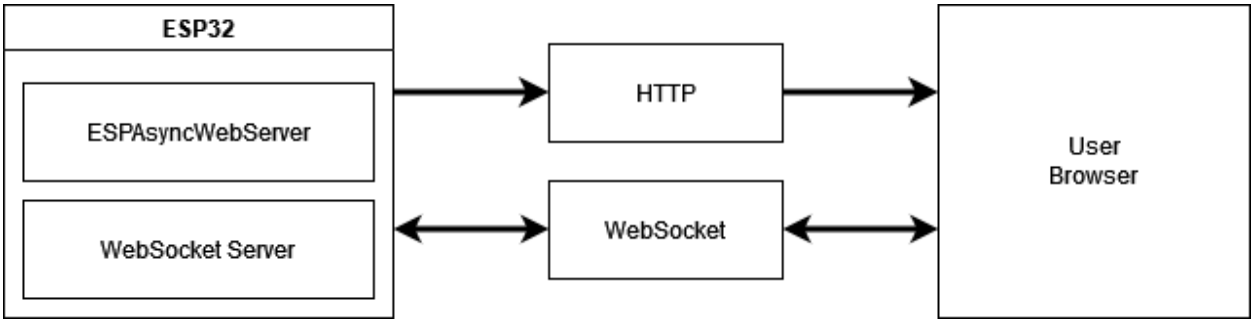


Figure 8: Web Stack Diagram. The diagram shows how the web interface is hosted and the communication links between them.

As for the visual webpage design, the main drop page is a homepage. The primary use of the page will be to route the user to subpages relating to the different inputs/outputs. The homepage will also have graphics and basic information about the playground. See Figure 9.1 for a prototype. The homepage is also where users will be rerouted if they run into any kind of network errors such as a 404 or 405 error.

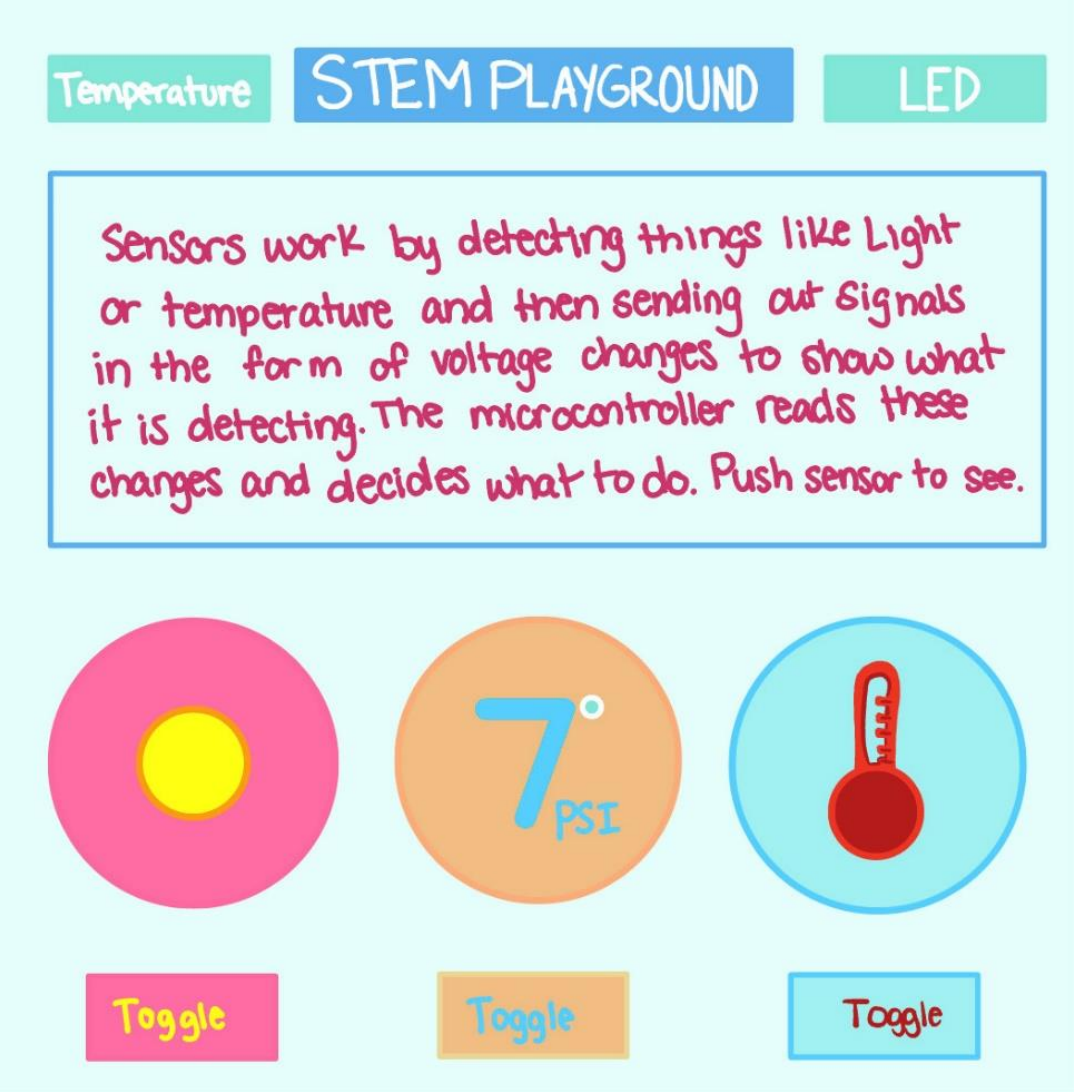


Figure 9.1: Web interface prototype. This depicts the homepage of the website showing each sensor with a brief description of how they work in conjunction with the microcontroller.

Each subpage will be specific to the input/output that it pertains to. Figures 9.2, 9.3, and 9.4 show mockups of the LED, pressure sensor, and temperature sensor subpages. The subpages will display the current value of the input/output, the data being sent or received by the microcontroller, and any other relating information. If applicable, a graphical representation of the device will be displayed. This means, for things such as the temperature, a thermometer will be the visual aid displaying the temperature. On all the specific sensor pages there will be a toggle button indicating that the student can press this to turn on the sensor. The data the microcontroller is receiving is an important part of the playground. This will be used to help the user see the information the microcontroller sees and can be used to explain how it is used to calculate the real-world value being displayed. Each subpage will include a button returning the user to the homepage as well.

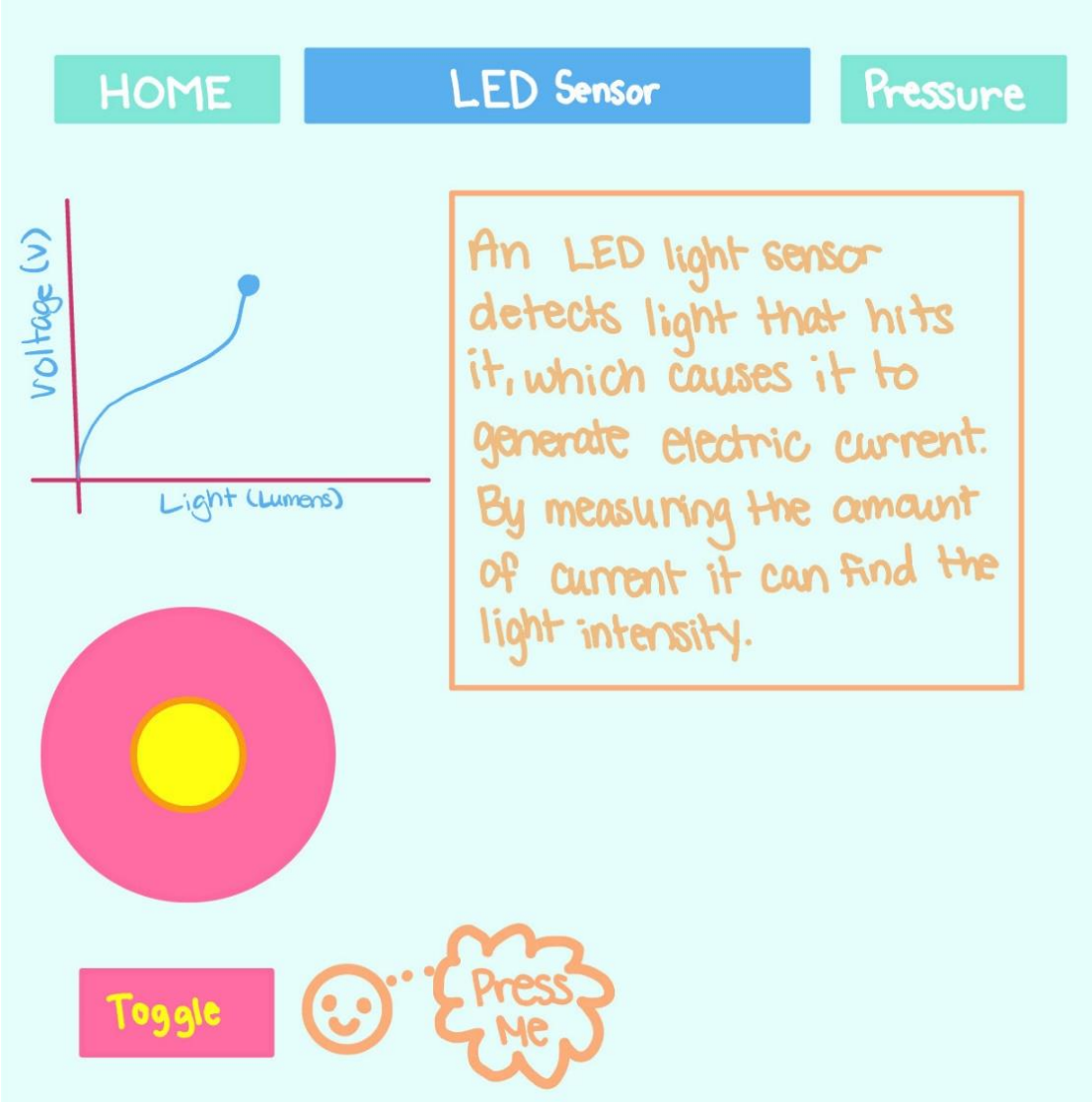


Figure 9.2: The next page on the interface mockup shows the LED light sensor page with links going back to the home page and to the next sensor, as well as a description of how the sensor works and a real time graph displaying the voltage to light ratio.

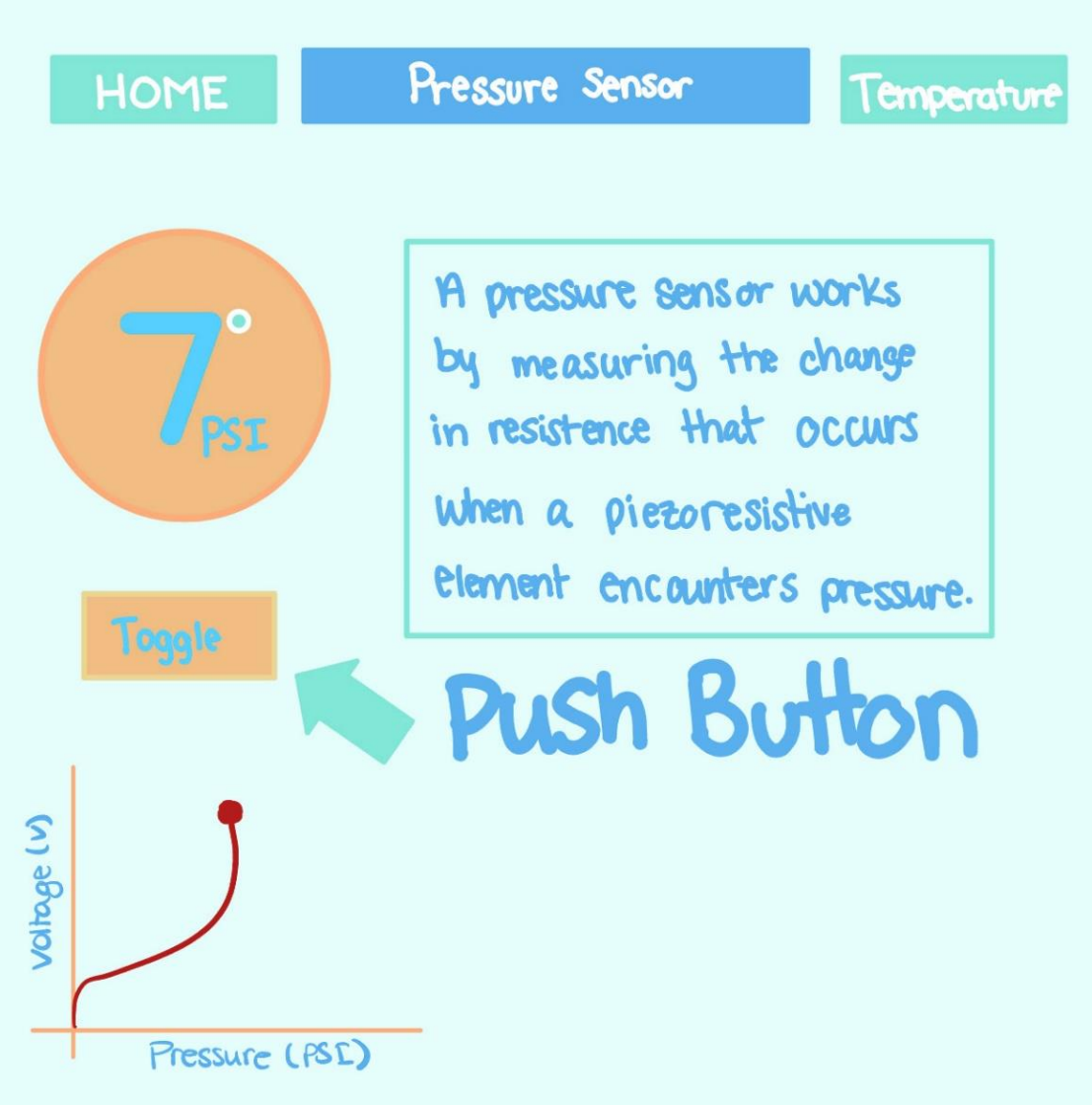


Figure 9.3: The next page on the interface mockup shows the force pressure sensor page with links going back to the home page and to the next sensor, as well as a description of how the sensor works and graph showing the voltage to pressure ratio.

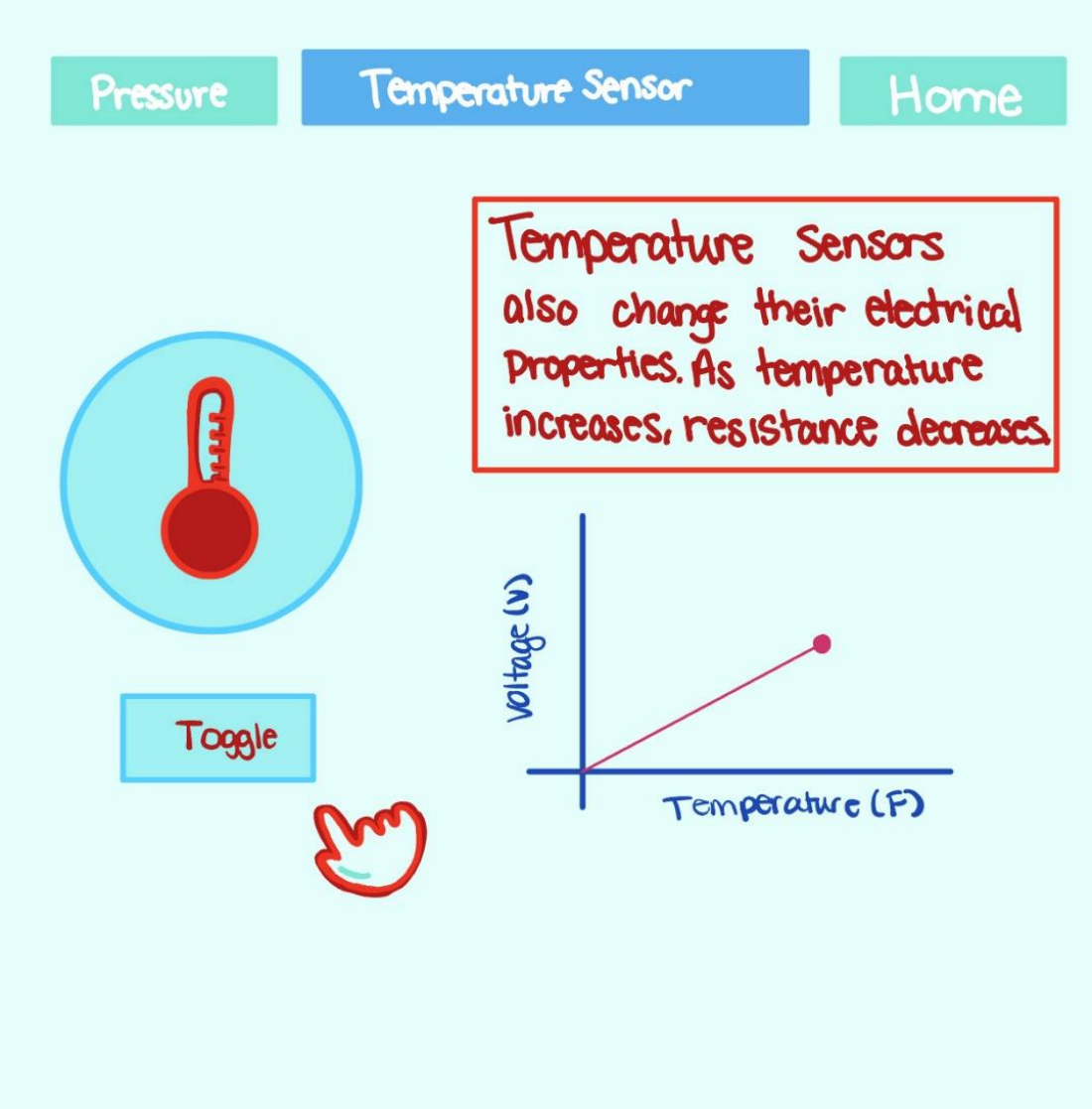


Figure 9.4: The next page on the interface mockup shows the temperature sensor page with links going back to the home page and to the next sensor, as well as a description of how the sensor works and graph showing the voltage to temperature ratio.

Additionally, a subpage for configuration will be included. This subpage will be designed to give more advanced configuration to users who are interested. It is here where they will be able to map inputs to different outputs than they come by default. The vison for this will be in some form of “If this, then that.” The user can choose an input from a drop-down menu, then parameters relating to it can be changed. From there an output can also be chosen from a similar drop-down menu, allowing its parameters to be altered. An example would be choosing the temperature sensor and DC motor. Using the parameters given, the DC motor could be assigned to spin at a certain speed based on the current temperature (“If temperature is hot, then spin DC motor”). Just like that, a basic climate control system has been built. Options like this give the

user more functionality and freedom to play, learn, and get excited with and about the playground.

To connect to the microcontroller, a provided router broadcasting a network will be used. All playgrounds will be preconfigured to connect to this network, making setup for an activity extremely easy. Users will just connect the device they wish to use to communicate with the playground to the same network. To access the homepage of the user's playground, the IP address will be entered into their browser. To identify the playground's address, a label will be attached. This label will include a QR code for camera equipped devices to easily connect and the IP address for other devices. Time permitting, a local DNS could be created to give each playground its own custom URL (e.g., playground1.stem). With the combination of these features, instructors and users should be able to easily set up and connect to playgrounds.

2.3.2 Multi-Page Website Code

Due to frequent unresolved issues with the Minnow server, the decision to move forward with hosting a custom web server was made. Through the help of ESP32IO tutorials, a foundation for running the web server and setting up the functionality for an LED and temperature sensor was completed. At the current moment, the code can route to five different webpages including the following: home page(index), temperature sensor UI, LED UI, error 404, and error 405. The languages used to create this code are HTML and JavaScript. For further updates to the code, the team will simply add to the code in the Arduino IDE and then recompile.

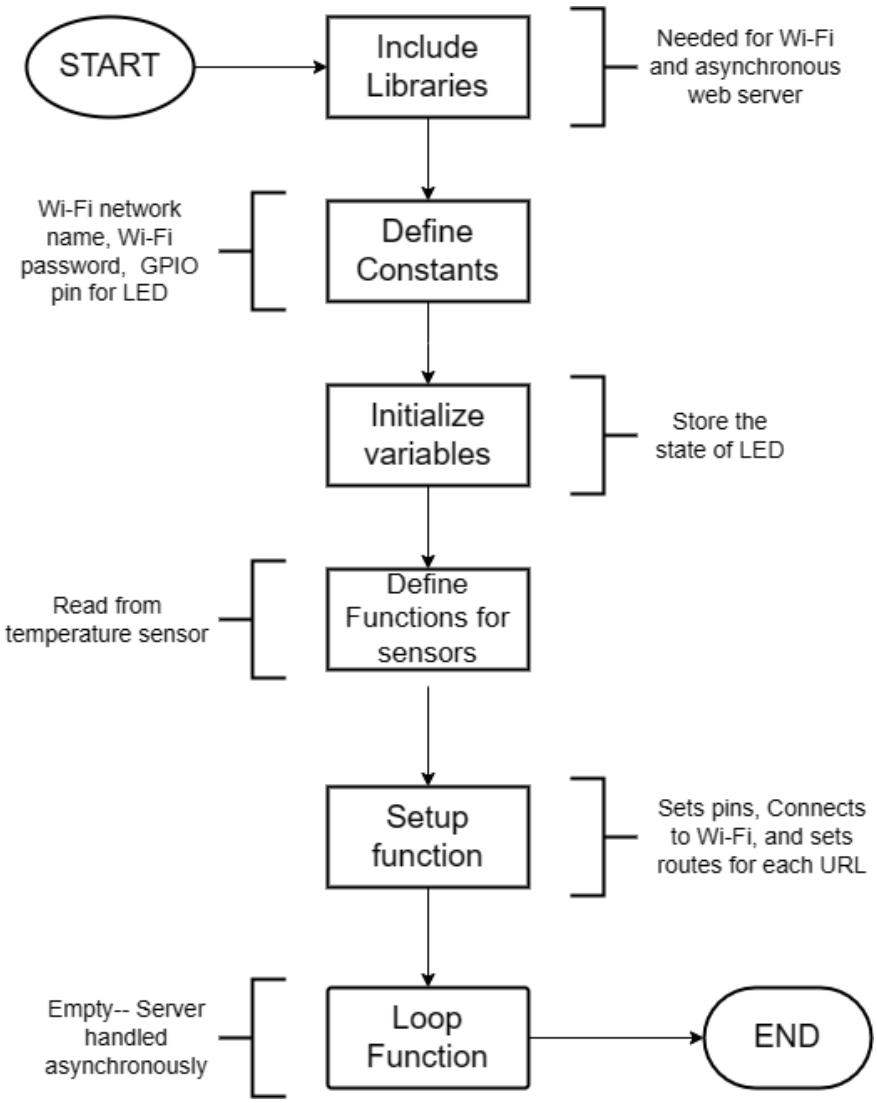


Figure 10: Outline of the code to setup the webserver, webpage, and implement the functionality of the sensors

As seen in Figure 10, there are 6 main steps to our current design. Including the libraries for Wi-Fi connectivity and asynchronous web server functionality is crucial to ensure the code works as intended. There are constants defined for the Wi-Fi network name, the Wi-Fi password, and the GPIO pin connected to the LED, as the LED is the only component the team has acquired thus far. Naturally, the next step initializes the variable that stores the state of the LED. After this, the functions to implement our sensors are created. Currently, the function for implementing the temperature sensor is a basic random number generator just so the team could test the webpage UI. Once the temperature sensor is acquired, the function will be updated with a proper implementation. Finally, the setup function is where most things are happening. First, the setup function initializes the serial communication for debugging purposes. Next, the LED pin is set as an output. The ESP32 is then connected to Wi-Fi using the specified SSID and password,

and its IP address for accessing the web server is printed. Finally, the routes for different URL's are set up such as '/led.html' or /temperature.html', and the webserver is started.

2.3.3 GitHub

GitHub is crucial in coding projects, offering a powerful platform for version control with Git integration. Its branching and merging capabilities facilitate smooth collaboration, allowing concurrent work on different aspects of a project. GitHub's user-friendly interface and centralized repository enhance coordination, monitoring, and code management. It outshines competitors through its extensive ecosystem, fostering collaboration with open-source projects, and offering features like GitHub Actions for automated testing and deployment. The platform's versatility and broad marketplace make GitHub the preferred choice for developers, ensuring an efficient and streamlined coding experience.

2.4 Cost Estimation

Bill of Materials

Item Name	Quantity	Cost			
ESP32-DEVKIT-V1	1	$ 10.00	Resistor 220 ohm	6	$ 0.12
Temperature Probe	1	$ 8.00	Lithium 18650	4	$10.00
Analog Pressure Pad	1	$ 15.20	Lithium BMS	1	$ 1.80
Servo	1	$ 3.62	Lithium Charger	1	$ 3.86
Transistor for DC motor control	1	$ 0.37	Case for Playground	1	$ 5.00
DC motor	1	$ 3.34	Voltage Regulator Pololu	1	$ 6.95
Switch	1	$ 0.67	Custom PCB	1	$ 0.13
LEDs	6	$ 0.41	Header Pins (M/F)	2	$ 2.20
				Cost Per Device	$ 71.66

(7) AA Batteries and Battery holder replacing Lithium batteries lowers cost to **$ 60.63**

Table 14: Overall Cost Estimation for a Single Playground

2.5 Mentor Recommendations

Jonah Mikesell, the hardware mentor for our team has aided preliminary design options. The main concern for our group's expertise was in designing the power module. Ensuring a safe and reliable method to power this device as a handheld design was something no team member has tackled from start to finish and Jonah provided some well-informed guidance. The method we are using for testing and getting the project propped up initially is with the on-board USB power option. The board we are designing can be powered and fed our development software both through a USB connection. Our final design for the power module will consist of a battery

pack of easy to source AA batteries fed to a voltage regulator. This voltage regulator takes the battery voltage down to the working voltage the microcontroller uses for operation. This design was informed by Jonah's recommendations for our team to move forward with a design concept. It should be noted that the voltage regulator selected recommends capacitors on the input and output sides to smooth voltage transients that will be considered. Further work is required to draw up schematic diagrams to include all components and these will be reviewed with Jonah when available.

Chapter 3: Build

3.1 Hardware Testing

3.1.1 Inputs Testing

- Button Test: HT_1.1: September 10th, 2024, Tested by Andrew
    - Testing the button was a simple process, the idea was to connect the button to the breadboard and use a sample code supplied through the ESP32 to turn on an LED. The board was set up using the following layout from newbie.com.

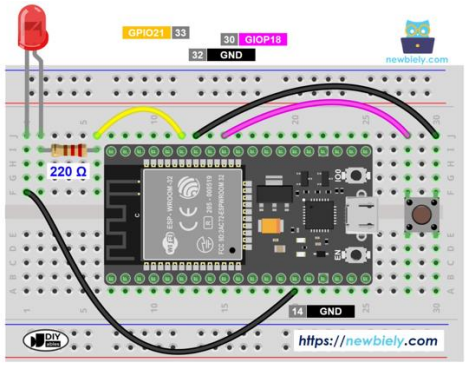


Figure 11.1: Button Test Configuration

Figure 11.1 shows that a 220-ohm resistor was used to connect the LED to the ESP32. However, the group decided to use a 10K ohm resistor in this situation. Once this diagram was wired in. A sample code for the ESP32 was provided through the Arduino coding platform. This code essentially debounced the button and set the flags when the button was pressed to toggle the LED on and off as the button was pressed.

In conclusion, the button test PASSED

- Switch Test: HT_2.1: September 10th, 2024, Tested by Andrew
    - Testing the switch was also a simple process. Similar to the button, the switch would be wired to the ESP32 and provided a sample test code to see if the switch is on or off. On the Arduino coding platform, the sample code would have a constant output (every second) that would state whether the switch was turned on, or off. The switch was connected to the ESP32 in figure 11.2 below.

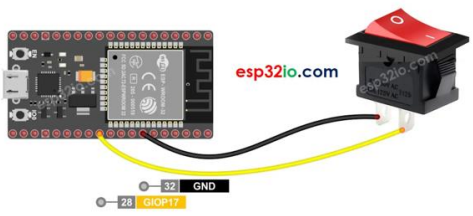


Figure 11.2: Switch Test Configuration

In conclusion, the switch test PASSED

- Pressure Sensor: HT_3.1: September 3rd, 2024, Tested by Kyle, Angelo, Ellie, Matt, Andrew
    - Testing the pressure sensor proved more complex. Initial trials encountered inaccurate voltage readings, especially when pressure was applied inconsistently. Calibration issues were identified, and the sensor wiring had to be adjusted for more reliable performance. Once resolved, the sensor successfully displayed pressure data in real time on the web interface. This was shown by converting the force received (analog values) to a voltage value via a test code.

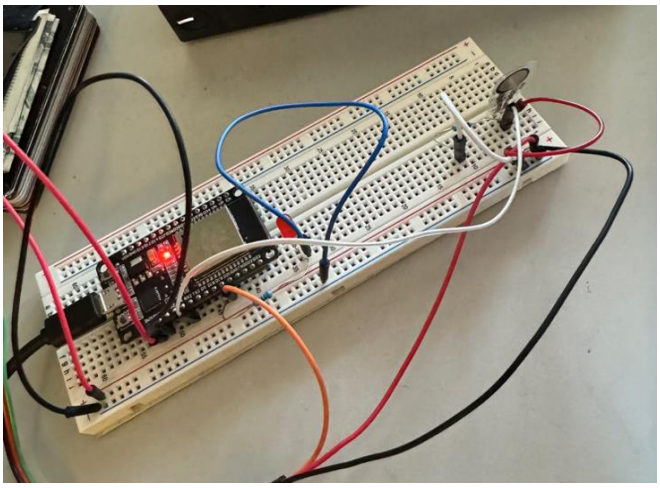


Figure 11.3: Pressure Sensor Testbench

Above is an image taken from the test bench for the pressure sensor test. This was supplied a code via the Arduino coding platform. All test code has been saved for future instances if the team needs to retest.

In conclusion, the pressure sensor test PASSED

- Temperature Probe: HT_4.1: September 3rd, 2024, Tested by Kyle
    - Testing the temperature probe involved checking the response to varying temperatures. The setup successfully measured the temperature via the ESP32 and displayed the data through the web interface. However, early tests indicated fluctuating readings when the probe connections weren't properly secured. The temperature sensor was configured the following way in figure 11.4.

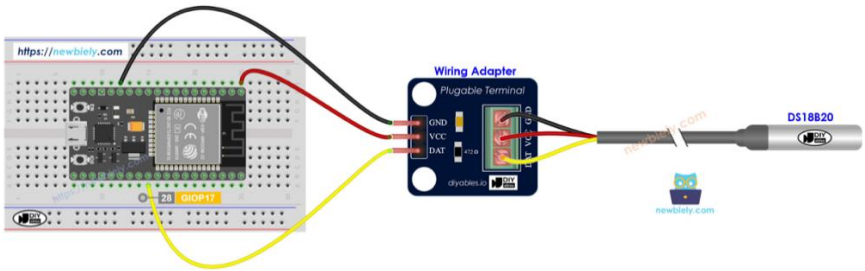


Figure 11.4: Temperature Sensor Configuration

The probe was placed in the hand of our teammates, and we saw the temperature spike on the webpage in real time. The probe was then placed in a cup of cold water, and we saw the temperature plumet on the webpage diagram.

In conclusion, the temperature sensor test PASSED

3.1.2 Outputs Testing

- DC Motor: HT_5.1: September 12rd, 2024, Tested by Andrew
    - A DC motor was tested by linking its speed to sensor inputs. The team successfully controlled the motor using the pressure sensor, with varying speeds based on applied force. Some early issues with motor stalling were resolved by improving the wiring connections and ensuring the motor received consistent voltage.

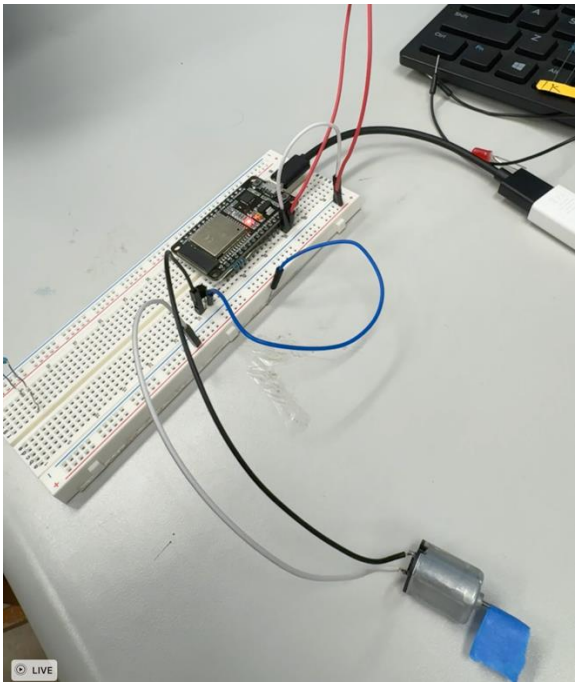


Figure 11.5: DC Motor Configuration

This motor was fed more power than the typical 3.3V we have used for the other smaller sensors. As well a transistor was used to be a relay so we will feed it battery voltage, a leg of the motor to the transistor, and the final leg is connected to the output pin. An increased voltage passed through the transistor will then increase how fast the motor spins.

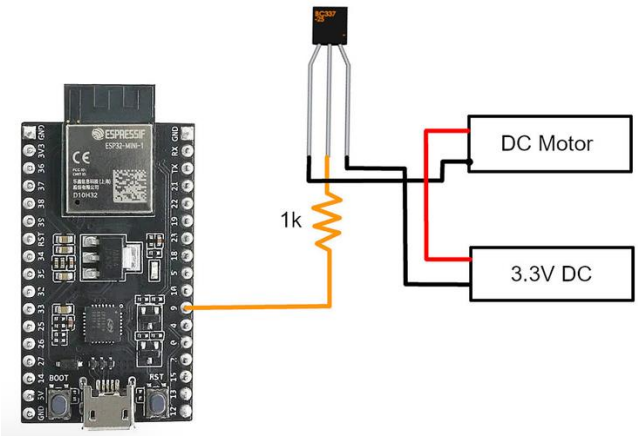


Figure 11.6: Transistor Connection to DC Motor

Figure 11.6 shows how the transistor was connected to the ESP32, voltage, and DC motor.

In Conclusion, the DC motor test PASSED.

- LED Sensor: HT_6.1: September 12th, 2024, Tested by Andrew, Matthew
    - While testing the LED sensor, the team connected the sensor to the microcontroller. Then powered on the microcontroller and wirelessly connected to the webpage. Once the ESP32 was supplied with a piece of code the LEDs lit up. They were able to be toggled on and off via the website, so this test was a success.

      In conclusion, the LED sensor test is PASSED.

- LCD Display Test: HT_7.1: September 12th, 2024, Tested by Andrew, Matthew
    - This was a difficult test to complete and is currently still in progress. The LCD display has an I2C which takes all the pins of the LCD and outputs only 4 pins. This configuration makes it easier for us to use since our microcontroller has limited pins. The first step of this test was to confirm the address of the LCD display by feeding it code to check the IP address. The LCD was connected to the ESP32 in the following figure 11.7. This connection required the LCD to be fed 5V instead of our usual 3.3V.

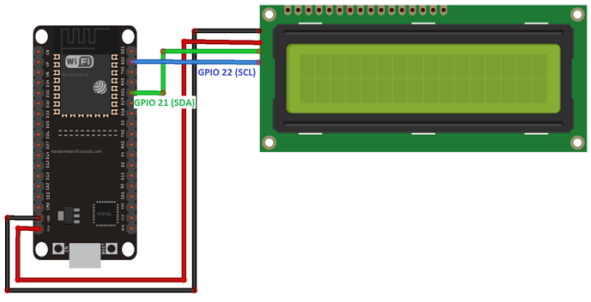


Figure 11.7: LCD Configuration

With the LCD now connected to the microcontroller, the sample code to find the IP address can be run. After running this code, we confirmed the LCD is at address 0x27.

In conclusion, this part of the test PASSED

However, when trying to display a sample code of "hello world" to the LCD, it did not display. Therefore, more testing needs to occur for the LCD to work.

In conclusion, this part of the test is IN PROGRESS

- HT_7.1: October 29th, 2024, Tested by Andrew, Matthew
    - Upon obtaining a new LCD display the display lit up but it still did not put out hello world. It was concluded that the display would require an excessive amount of troubleshooting, and the team determined it would be best to put this in the parking lot.

      In conclusion, this test FAILED.

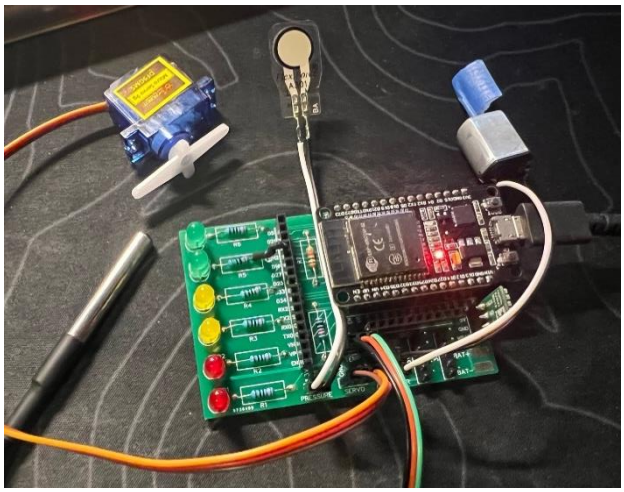
- Temperature Sensor with DC motor: HT_12.1: September 26th, 2024, by Andrew, Matthew
    - The goal of this test was to ensure that the temperature sensor inputs can affect the speed of the DC motor. The DC motor should speed up when the sensor reads a higher temperature and should slow down when it reads a lower temperature. The team initially had a minor issue with the code which caused the temperature sensor to repeatedly read a constant incorrect value. This caused the DC motor to not work properly as well. However, the code was debugged, and the components were verified to work together as intended.

      Thus, the Temperature Sensor with DC motor test PASSED

- Pressure Sensor with LEDs: HT_13.1: November 18, 2024, by Andrew
    - The goal of this test was to ensure that the pressure sensor works properly with our 6 LEDs separate from the webpage. The requirement was that when the sensor receives a force, LEDs will continue to toggle depending on how high the force applied is. A low force will only toggle up to a few LEDs while a large force could toggle all 6.
      The Pressure Sensor with LEDs test PASSED
- Additional LEDs test (6 LEDs): HT_15.1: November 18th, 2024, by Andrew
    - The goal of this test was to ensure that up to 6 LEDs could be connected to the microcontroller and that they could toggle off and on properly.
      The Additional LEDs test PASSED

- PCB Functionality test: HT_16.1: November 19th, 2024, by the whole team
    - The goal of this test was to ensure our components work properly on the PCB, verifying its design. This test required that all components can be connected at the

same time, and each behaves as expected while also working in conjunction with its respective webpage.
The PCB Functionality test PASSED



3.1.3 Power Testing

- Voltage Regulation: HT_9.1: September 12th, 2024, Tested by Kyle, Angelo, Ellie
    - Before testing the voltage regulator, the group had to solder on wires to connect it to the microcontroller. This proved to be a challenging task as the regulators ended up being much smaller than anticipated. Nonetheless, the wires were soldered on, and we were able to connect it and reach voltage levels within an acceptable range.

      Thus, the voltage regulator test is PASSED.

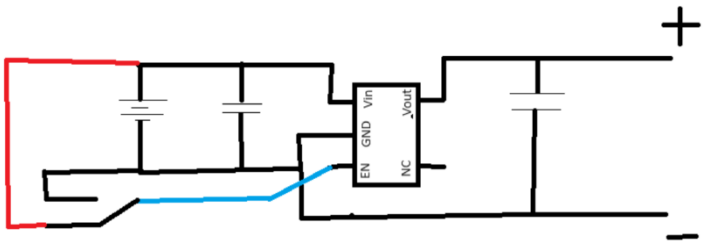


Figure 11.7: Voltage Regulation Circuit Schematic.

- Voltage Regulation: HT_9.2: October 22nd, 2024, Tested by Kyle, and Ellie
    - Although HT_9.1 passed, HT_10.1 failed. This is because the regulator regulated the voltage to acceptable levels but did not provide enough current to support the device. Due to this the team had to order parts to design a new style of regulator, a

buck-boost converter. After assembling and testing the new regulator, we were unable to get an output voltage higher than 0.062.

Thus, the voltage regulator test FAILED.

- Voltage Regulation (Converter): HT_9.3: October 23rd, 2024, Tested by Ellie
    - A new regulator from Pololu came, upon initial testing without load, regulator was able to hold a steady output of 5V even with changes to supply input.

Thus, the new regulator test PASSED.

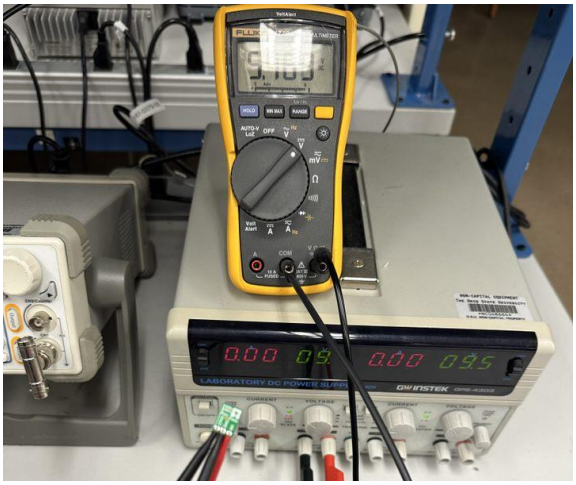


Figure 11.8: Voltage Regulation 5V output from 9.5V input.

- HT_9.4: October 31st, 2024, Tested by Kyle
    - Load tested the converter with a full working subsystem and lithium batteries, and they steadily output the 5v required without burning up.

Thus, the converter/ regulator test PASSED.

- Transistor Testing: HT_8.1: September 5th, 2024, Tested by Kyle, Ellie
    - The transistor is used as a relay to provide the DC motor with battery voltage and controller via a signal pin. The team connected one leg to a battery, one leg to the dc motor, and one leg to the output pin, as you increase the voltage on pin it increases the voltage on motor. In conclusion, the transistor effectively increased the voltage and allowed the DC motor to run efficiently.

Thus, the transistor test is PASSED.

- Power Testing: HT_10.1: September 26th, 2024, Tested by Kyle, Andrew
    - This test was to verify that the voltage regulator and batteries can reliably power the device. Unfortunately, the device's current draw is too high. This causes the voltage regulator to overheat and go into over-current mode. Multiple times, this caused the soldered leads to desolder. This means that the team will have to test

the current requirements of the device (HT_14.1) and order a compatible voltage regulator.

Thus, the power test FAILED.

- Current Requirements: HT_14.1: September 26th, 2024, Tested by Kyle, Andrew
    - Failed test HT_10.1 taught us that the device draws more current than originally calculated. Before ordering a new voltage regulator, the team needed to determine the current requirements. This was done by connecting to the lab bench power supply. Under normal use, the device drew 250mA. Under max load, the device drew 350mA. To test a malicious use case, the servo was physically stalled. This led to a surge of current up to 500+mA. This data can now be used to find a suitable voltage regulator.

      Thus, the current test is PASSED.
- Battery Life: HT_11.1: November 4th, 2024, Tested by Kyle, Angelo, Ellie
    - The charging circuit although worked drew a little too much current than what it should be for the inductor in the circuit despite saying it was rated as such. Nonetheless the team was able to add a motor in series to dissipate some of the heat.

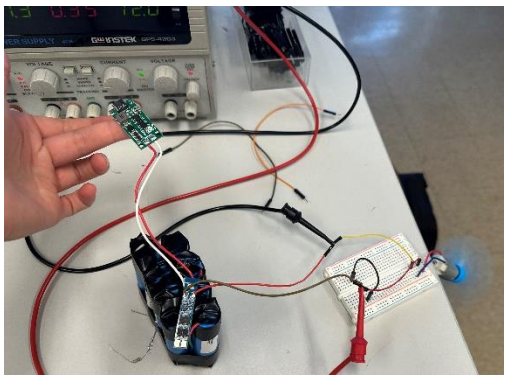


Figure 11.9: Lithium battery charging setup.

Thus, this test PASSED.

3.2 Software Testing

3.2.1 Subsystems

- Button Testing: WP_1.1: September 3rd, 2024, Tested by Kyle
    - For this test, a physical button was connected to the ESP32 and was linked using WebSocket to a button on the webpage. When the button was pressed physically, it was shown as pressed on the webpage. When the button on the webpage was pressed, it activated the same LED that the physical button does.
      In conclusion, the button test PASSED.

- Temperature Probe Testing: WP_3.1: September 3rd, 2024, Tested by Kyle
    - For this test, the temperature sensor was connected to the ESP32 and was linked using WebSocket to a thermometer on the webpage. The thermometer displayed an accurate temperature representation.
      In conclusion, the temperature probe test PASSED.
- Pressure Sensor Testing: WP_4.1: October 8th, 2024, Tested by Matt, Andrew
    - This test was done to verify if the pressure sensor can communicate with the webpage to display the analog data and voltage in real time while the sensor undergoes a force. While the sensor and related hardware works perfectly (HT_3.1), the team has had many issues getting the values to display on the webpage. The team is continuing to debug and potentially getting new eyes on the problem.
      Thus, the Pressure Sensor test has FAILED
- DC Motor Testing: WP_5.1: October 17th, 2024, Tested by Matt, Andrew, Kyle
    - This test was done to verify basic functionality and communication between the DC Motor and the webpage. The DC Motor can be toggled off and on by clicking a button on the webpage.
      The DC Motor test PASSED
- Webpage LED Test: WP_6.1: October 17th, 2024, Tested by Kyle
    - This test was done to verify the webpage would have the ability to toggle the LED(s) and see the LED(s) turn on real time. This was a simple test that incorporates 6 LEDs with 3 different color types.
      The Webpage LED Test PASSED
- LCD Webpage Test: WP_7.1: October 31st, 2024, Tested by Andrew, Matt, Kyle, Angelo, Ellie
    - This test was done to verify that a user could send a string or phrase to the LCD and the LCD will display the string from the user. However, when the webpage was created and run, the LCD unfortunately did not display the text from the user.
      The LCD Webpage test FAILED
- Open Webpage: WP_8.1: September 3rd, 2024, Tested by Kyle, Angelo, Ellie, Matt, and Andrew
    - This was a basic test to ensure that the ESP32 could broadcast an accessible webpage. The entire team could access the website's homepage.
      In conclusion, the open webpage test PASSED.
- Host Server: WP_9.1: September 3rd, 2024, Tested by Kyle, Angelo, Ellie, Matt, and Andrew
    - This test was to ensure that the ESP32 could host a WebSocket server to allow bi-directional communication. The entire team used the webpage as an input and output.
      In conclusion, the host server test PASSED.
- Servo Motor Testing: WP_11.1: October 18th, 2024, Tested by Andrew
    - This test was done to verify that the servo motor can communicate with our webpage. On the servo motor webpage, there is a slider that the user can manipulate to rotate the servo motor a desired amount between 0-180 degrees.

The Server Motor Test PASSED

- Webpage Pressure Sensor: WP_4.2: October 15th, 2024, Tested by Kyle, Andrew, and Matthew
    - This test was to ensure that the ESP32 main code could run on the new page created. After hooking up the pressure sensor, we were able to successfully see the data come to the webpage in real time. The issue before was that the pin we were utilizing had been rendered mute by the Wi-Fi.
      In conclusion this test PASSED

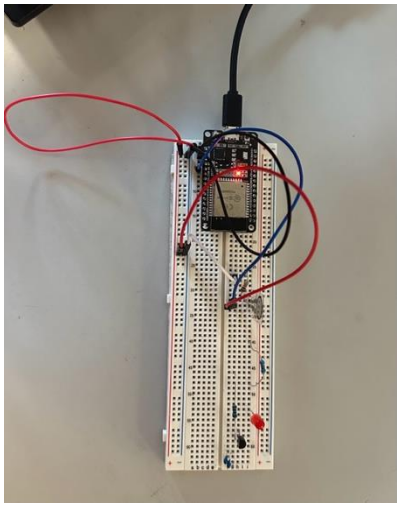


Figure 11.10: Breadboard setup for pressure sensor testing.

- Webpage Temperature Sensor and DC Motor Test: WP_12.1: November 12th, 2024, tested by Andrew and Matt
    - This test was conducted to make sure the esp32 could host a webpage that utilizes two different sensors/components in the Temperature Sensor and the DC Motor. The goal was to give the user the ability to set a "threshold" for what they want the temperature to be for the DC motor to turn on. We want this to simulate and be like an AC unit when it would kick on once a certain temperature has been met.
      The Temp DC motor Test PASSED
- Webpage Pressure Sensor and LED(s) Test: WP_13.1: November 18th, 2024, tested by Andrew
    - This test was conducted to ensure that a new webpage could be created to turn on and off the LED lights by how hard the user is pressing on the pressure sensor. The webpage will also show when the lights turn on. Currently we have the pressure force (N) "thresholds" hidden from the user so they will have to use the webpage to see what force causes each LED to turn on. As well the voltage value will be available for the user to see.
      The Pressure LED Test PASSED
- Additional LED(s) Test: WP_14.1: November 18th, 2024, Tested by Andrew

    - This test was conducted to add 5 more LEDs to toggle to the LED webpage. Previously, the LED webpage only toggled 1 LED. With the new webpage it should toggle 6 different LEDs of multicolor. The webpage shows which LED is toggled by turning on a virtual LED on the webpage.
      The Additional LED Test PASSED
- Webpage Separation Test: WP_15.1: November 19th, 2024, Tested by Andrew, Matt, Kyle
    - This test was conducted to ensure that all webpages will function separately from each other to ensure isolation. This was needed because we have 2 webpages for subsystems that utilize multiple components, but we also want those components to function normally in their singularity webpage. This means that if the user is on the singularity pressure sensor webpage, the LEDs will not turn on unless they are on the pressure LED webpage.
      The Webpage Separation Test PASSED

3.3 Component Inspections

All Components used in the hardware/software tests were verified by comparing the received components with what was ordered. Some specific verifications include the temperature sensor probe has at least 2 ft of cable, the motor spins under hand power, and the batteries each have around 1.5V.

3.4 The Build Process

The testing mentioned in previous sections paved the road for the team to successfully assemble the final build. The implementation of a PCB leads to easy assembly of the build. The team solders components onto two PCBs. These components include female headers for the ESP32, s female header for the ground bus, a female header for the voltage regulator, male headers for the I/O, male headers for 3.3V and 5V buses, 8 resistors, 2 red LEDs, 2 yellow LEDs, 2 green LEDs, and the battery pack. Unfortunately, only one PCB resulted in a working product without problems.

The battery pack was assembled from eight 18650 Lithium-Ion batteries soldered together in a 2S4P configuration. That means two banks for 4 cells are soldered in parallel, followed by the two banks being soldered in series with each other. The result being an 8-amp hour battery with a voltage of 8.4V.

With a functional PCB and battery pack, it was time to assemble everything. All I/O devices and the ESP32 were designed to connect using header pins, making assembly easy. Once connected the device was powered on and tested to determine flawless execution.

Due to time constraints, the team was left unable to test the 3D printed case designed. A final fitment using the case will be tested soon.

Chapter 4: Test Plan and Results

4.1 High Level Test Overview

Testing was planned initially into subgroups comprised of inspection, hardware, software, user acceptance, and automated tests. The group initially was diligent in documenting our testing in the tracking spreadsheet, written results, build process, and JIRA updates, however, as iterations became more widespread and complex, the documentation was put in the parking lot.

The inspection tests were done to ensure the parts that came were the correct parts for building. Hardware tests were in most cases making sure the ESP32 was compatible with the hardware or working on the power module. Software testing in large was putting together the webpage and debugging issues with communication between the webpage and I/O. Lastly, the user acceptance tests we were not able to complete but were designed to allow a test run of the entire product and its instructions. Automated testing is difficult to procure due to numerous subsystems and integration. However, we were able to have the sensors each test run upon start up to let the user know if anything was amiss.

In summary, testing was marked by several key events; finding a reliable converter, redesigning the power module to be rechargeable, discovering pins lost their digital to analog abilities when connected to Wi-Fi, and finally getting the pressure sensor to work. There were also cuts made due to time constraints for the button/switch webpages and abandoning the LCD display after sinking time into it. As a whole, testing illustrated for the team the importance of testing early and planning for failure. Half of our tests failed upon first try and later passed after multiple iterations. Additionally, the team was reminded of the importance of documentation.

4.2 Test Plan and Results Summary

The teams Test+Planning+and+Tracking+Template.xlsx file was used as an overview to track the status and summary of all tests.

4.3 Detailed Test Plan and Results

The team's Test Plan Draft #1 - Copy.docx file includes a detailed outline of the testing plan and results.

Chapter 5: Redesign, Lessons Learned, Compromises, and Future

5.1 Requirements / Design Changes

Regulated Supply Voltage

- Originally intended on supplying the ESP32 with 3.3V
- Determined the DC motor and LCD required 5V to operate
- Switched to supplying 5V

- Voltage Regulator
  - Originally used a 3.3V linear regulator
    - Supply voltage changed to 5V
    - Did not supply enough current

    - Attempted to design 5V buck/boost converter
        - Testing failed to output a consistent 5V
    - Finalized the design using a predesigned buck/boost converter from Pololu
        - Testing resulted in clean and consistent power
- Battery solution
    - Started with disposable AA
    - Discussed using rechargeable AA
    - Switched to rechargeable lithium-ion
- Web server
    - Started with Minnow Server
        - Ran into issues getting it to work
        - This option was less modular than making custom webpages
    - Switched to custom webpages
        - Allowed the team to be more familiar with the design since they were built from scratch
- Hosting on ESP32
    - Had the idea to use a mothership to host webpages
    - Found easier to host individually on each esp32

5.2 Compromises Made

- Lack of subsystems
    - Due to time and limited devices the team was only able to complete 2 subsystems consisting of the dc motor with the temperature sensor, as well as the pressure sensor with the LEDs an additional subsystem had been brainstormed utilizing the Pressure Sensor with the Servo. This subsystem would turn the servo as the pressure increases, simulating a pressure gauge.
    - Coding these multi-component subsystems takes hours of debugging and writing, as mentioned before, with the time constraint adding more was not an option for our team.
- Pricing
    - In priority of time and convenience the team decided on more expensive components
    - Utmost priority was functionality, and then cost, so the final product cost is not as low as the team had hoped for, but it was necessary for a working product
    - The Pressure Sensor costed about $15, but the team believes there are alternative options that would lower overall cost
- LCD
    - Time constraints and failed testing led to the LCD being put on the back burner.
    - Extensive testing was conducted on the LCD such as finding the effective address of the I2C board, which is what the user writes to. As well as trying multiple LCD boards and being unsuccessful.

    - The code for the LED webpage and our attempt is still located on GitHub, it has just been commented out. When this device is picked up, that should be the first place to look when trying to solve this dilemma.
- BMS
    - To save time we used an unverified 3rd party BMS for battery protection without any datasheet
    - It would be advised to replace this with a custom or verified solution
- Charging circuit
    - Just like the BMS, an unverified 3rd party battery charger without any datasheet was used
    - It would be advised to replace this with a custom or verified solution
- 3D printed case
    - The device case was overlooked until last minute which led to the inability to test it
    - This case will need to be tested and may need updates
- Webpage images
    - Device storage was sacrificed to include images on some of the webpages
    - Each image uses approx. 4-5% storage
    - Images must be converted to base 64 to import into the website. This is due to the router not having a connection to the internet. The website to do this conversation is as follows: https://www.base64-image.de

5.3 Future Improvements and Features

Although the team is proud of the accomplishments made, there are some future improvements and features that should be added. This includes design considerations of the group and feedback given by stakeholders. Storage capacity, IP address management, configuration changes, relating voltage to I/O, and changing the DC motor speed are some of the main improvements that could be made.

- ESP32 is low on storage
    - The storage capacity of the ESP32 is running low (77% usage)
    - Could have a host computer store them
    - Memory is currently not an issue though
- ESP32 IP address
    - Currently must be changed in the code
    - Could use dip switches to determine last digits of IP
    - A host computer using MAC addresses to map IPs
- A central config file
    - Used to change config such as pressure sensors force to turn on LEDs
    - Could use a host server to run a webpage that can be used to change the config
        - Then the playgrounds request that config on startup
- Every I/O has a voltage/current/resistance associated with value being tracked

    - Integrate Ohms law into activity and provide charts/graphs
- DC motor speed
    - Ability to change the speed of DC motor
        - Also show the voltage depending on the speed
- Servo motor with a relationship on the angle
    - The Servo used for this project was a digital servo motor
- Additional webpages need corresponding voltages, current, and resistance readings associated with the sensors.
    - Integrate Ohms law and how it is changing.

Chapter 6: User Manual

6.1 Product Explanation

The STEM Playground is an interactive device created specifically for students, teachers, and parents to learn about integrated systems. The device consists of a main ESP32 microcontroller that connects to Wi-Fi and hosts the server. Connected to the microcontroller are various inputs/outputs consisting of LEDs, Pressure Sensor, Servo Motor, DC Motor, and a Temperature Probe. The server hosted on the webpage will broadcast a webpage that allows students to interact with these inputs/outputs in various fun and meaningful ways. The main goal of this Playground is to show the correlation between these sensors and the voltages that are associated with them. Along with the interactive device, instruction slides will be provided to drive home this main topic during a 55-minute class period.

Here is a thorough introduction and walk-through of the steps to follow within a class period: Instruction Slides.pptx

6.2 Safety Considerations

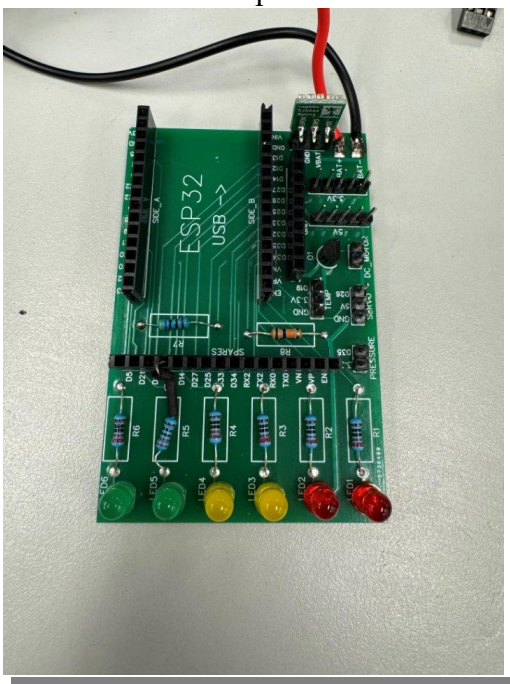
Overall, the team developed a final product that is safe. One issue that is possible though unlikely could is electrical components becoming exposed out of the case. Exposed electrical components can cause fires, shocks, burns, or electrocution if mistreated. If there is some exposure of an electrical component, be cautious when dealing with it. Another thing that could be dangerous includes charging the lithium batteries. The PCB used in the charging circuit can heat up quite a bit, so be careful where you touch when the batteries are being charged. Just be sure to have done due diligence regarding the batteries and charging system. Finally, be sure to avoid bridging the positive and negative leads of battery.

6.3 Product Guideline

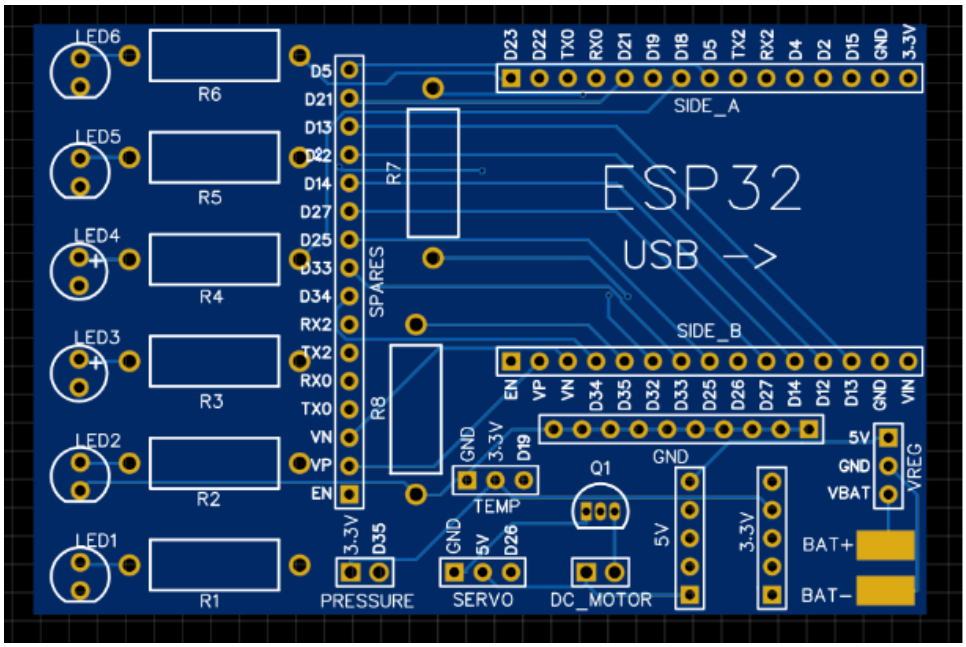
6.3.1 How recreate this project from scratch

1. Assemble the PCB and attach the ESP32 Microcontroller, all input/output sensors, and battery pack.

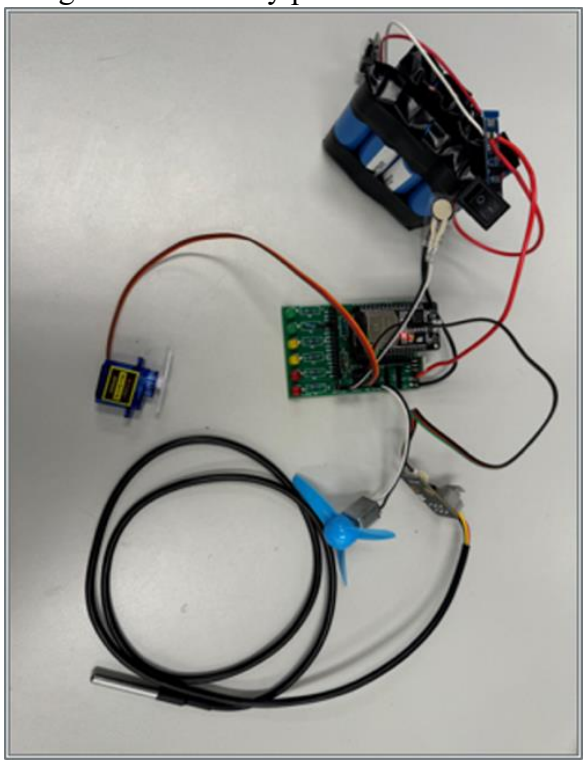
a. The PCB for this project has been designed to the specifications of the current sensors and all devices. There are 5 total PCBs already created, however here is the link to the schematic to change/update as needed: https://buckeyemailosu-my.sharepoint.com/my?id=%2Fpersonal%2Fgrcic%5F4%5Fbuckeyemail%5Fosu%5Fedu%2FDocuments%2FECE%203906%20STEM%20Playground%2F3%2E0%20Design%2FPCB
    i. The .zip file linked is a backup of the current PCB project. The PCB design was created using the free EasyEDA PCB designer. The backup file can be opened and modified using EasyEDA.



b. Resistors: R1 – R6 220 Ohm (6 ea) for LEDs
R7 470 Ohm (1 ea) for DC Motor
R8 10K Ohm (1 ea) for Analog Pressure Sensor

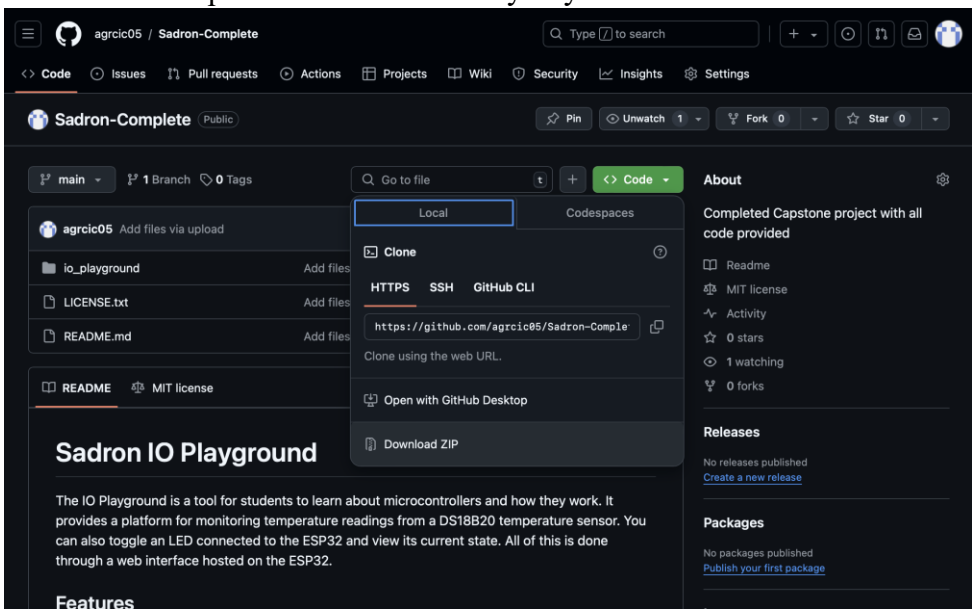


c. The current inputs and outputs can be connected to the PCB in this configuration along with the battery pack.

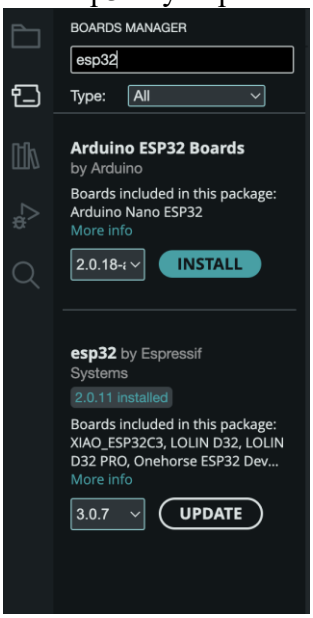


2. Once the device has been set up, the microcontroller will need to be programmed to host and run the web server along with all the functionality of the system. To start on a laptop device, download and install the Arduino IDE coding platform. The link to download can be found here: https://www.arduino.cc/en/software
3. Once installed, the user will need to download the zip file for the STEM Playground from GitHub.

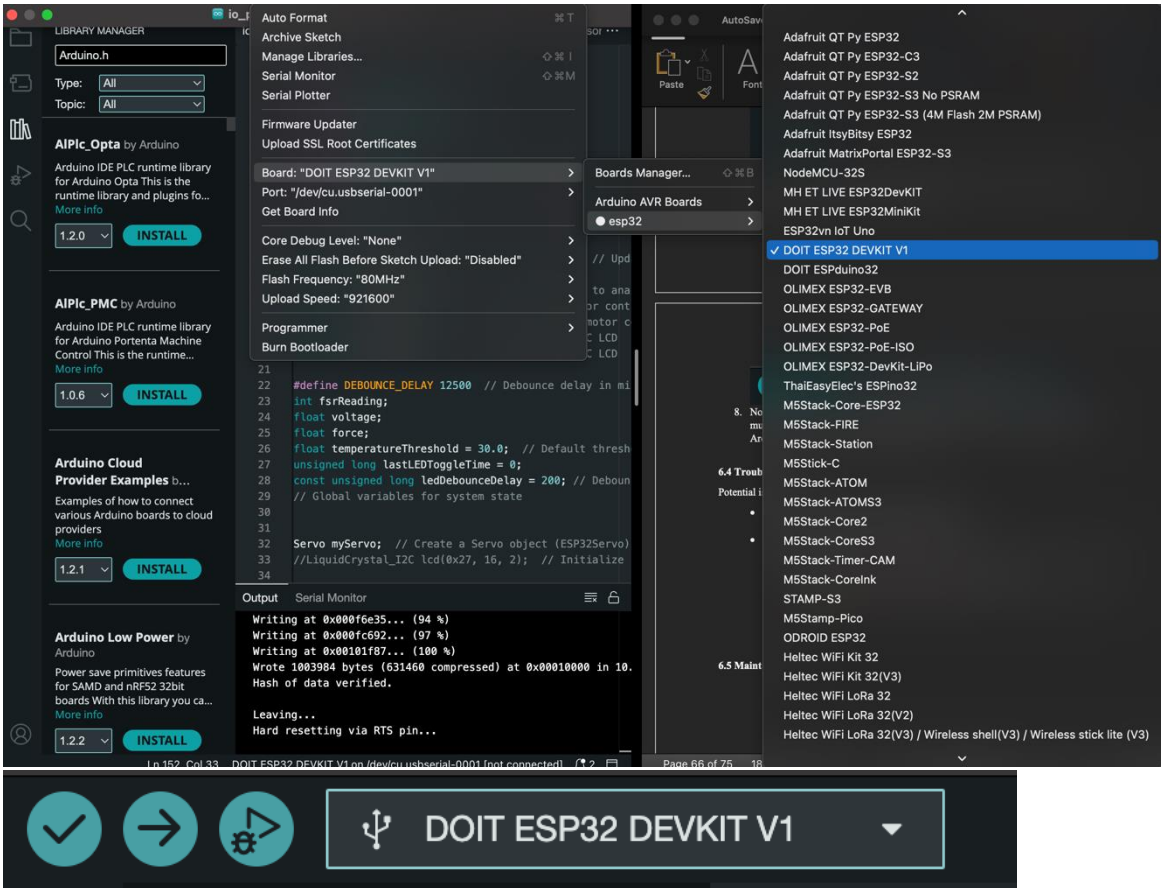
a. Go to: https://github.com
b. Search for the repository “Agrcic05/Sadron-Complete”
c. Within the repository, select the dropdown button for “code” in the green box and download the zip file somewhere locally to your disk.



4. Unzip this file and open the “io_playground.ino” file. This should open in the Arduino IDE platform along will all the html (.h) webpage files.
5. Connect the board to your device using a micro-USB cable/adapter
6. On the left side of the Arduino IDE platform, open the Boards Manager and search/install the “esp32 by Espressif Systems” to your device.



7. Once installed, open the dropdown for the select board and port and update it so the DOIT ESP32 DEVKIT V1 has been selected.



8. Now that the code and board have been imported on your Arduino IDE platform, multiple libraries need downloaded and installed. Open the Library Manager on the Arduino IDE and install the following libraries:
    a. ArduinioJson by Benoit Blanchon
    b. AsyncTCP by dvarrel
    c. DallasTemperature by Miles Burton
    d. ESP32Servo by Kevin Harrington
    e. ESPAsyncTCP by dvarrel
    f. ESPAsyncWebServer by lacamera
    g. OneWire by Jim Studt, Tom Pollard…
    h. WebSockets by Markus Sattler
    i. ezButton by ArduinoGetStarted.com
    j. hd44780 by Bill Perry
9. With all the libraries now installed, navigate to the main io_playground.ino code and go down to line 62. Here is where the Static IP Address is located. If you are programming a new device, this IP address will need to be changed. Increment this IP address so that it is unique from other boards already programmed.

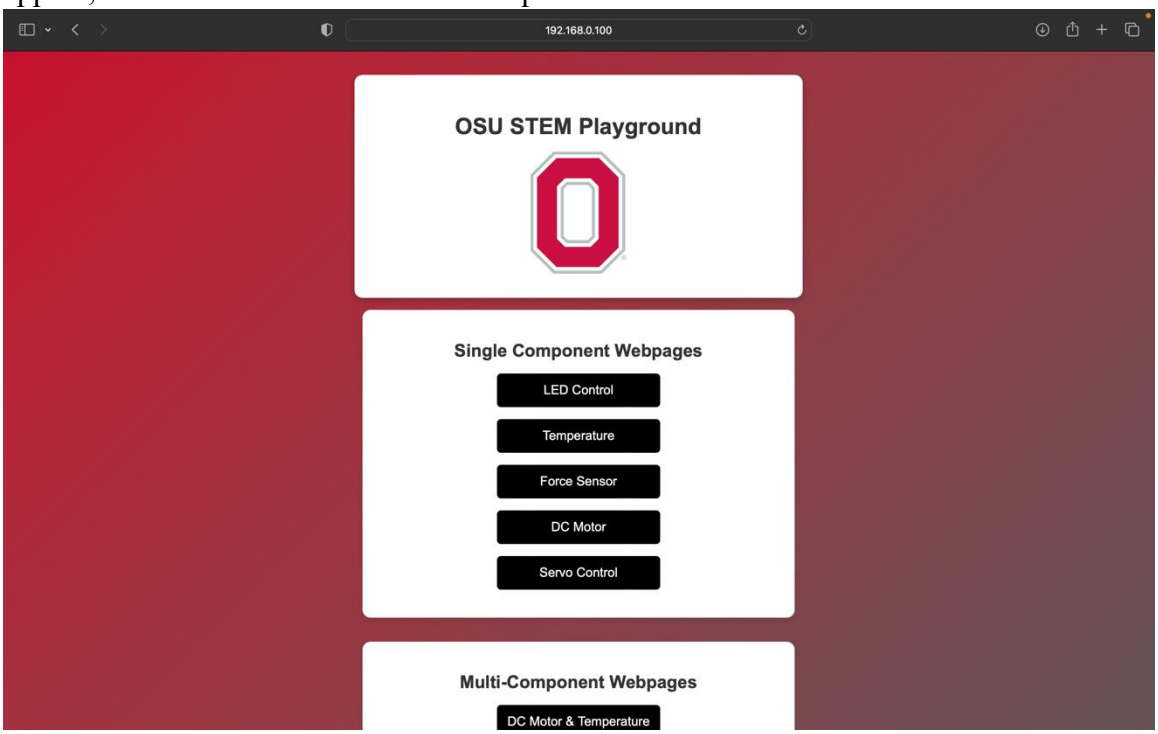
```
61  // Static IP, Gateway, and Subnet
62  IPAddress local_IP(192, 168, 0, 100);  // Your desired static IP address
63  IPAddress gateway(192, 168, 0, 1);    // Your network's gateway
64  IPAddress subnet(255, 255, 255, 0);   // Your network's subnet mask
65
```

10. After steps 1-9 have been completed, the board is ready to be programmed. Make sure the router for the Playground is plugged in and on. On your device, open your WIFI settings and connect to the router. Once done, select the arrow in the upper left-hand corner to upload the code to the ESP32.

11. If done correctly the Serial Monitor will inform the user, the device has been successfully connected to the WIFI and print out the IP Address.

```
entry 0x400805e4
Connecting to WiFi..
Connection successful!
IP Address: 192.168.0.100
```

12. Open a web browser and paste this IP Address in the search bar, the webpage should appear, the user can use the device as required.

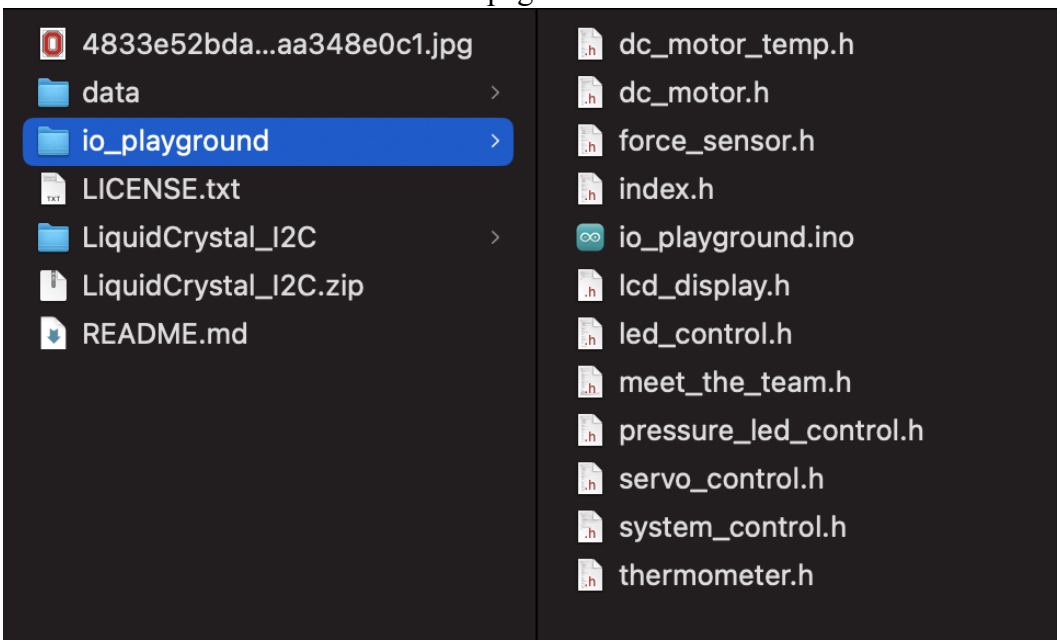


6.3.2 How to connect additional I/O to the PCB

Connecting additional I/O to the PCB can easily be done by utilizing the spare ESP32 pins. How exactly that is done is heavily dependent on the device. To help with this, the ground, 3.3V, and 5V buses can be used to supply the device with its preferred voltage. From there, the user can reference the ESP32 and new devices datasheets to determine the best pin to use and if additional hardware is required.

6.3.3 How to add additional webpages to the website

1. To start with creating a new webpage, a new html file (.h) file needs to be created and added to the io_playground folder. Make sure to make the name of the html file to resemble what the user wants the webpage to be.



2. There are multiple locations where this new file needs to be referenced/included.
    a. In the main .ino file and the index.h file, it needs to be included/referenced in these locations ->
        1. Include webpage section (io_playground.ino)

```
// Include required HTML pages
#include "index.h"
#include "led_control.h"
#include "thermometer.h"
#include "force_sensor.h"   // Force sensor HTML page
#include "dc_motor.h"       // DC motor HTML page
#include "servo_control.h"  // Servo motor control HTML page
//#include "lcd_display.h"    // LCD display HTML page
#include "dc_motor_temp.h"
#include "pressure_led_control.h"
#include "meet_the_team.h"
```

        2. The Server On webpage section (io_playground.ino)

```
// Configure server routes for different pages
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", index_html);
});
server.on("/led_control", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", led_control_html);
});
server.on("/thermometer", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", thermometer_html);
});
server.on("/force_sensor", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", force_sensor_html);
});
server.on("/dc_motor", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", dc_motor_html);
});
server.on("/servo_control", HTTP_GET, [](AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", servo_control_html);
});
```

3. A link to the webpage from the homepage (index.h)

```
<div class="section-container">
  <h2>Single Component Webpages</h2>
  <a href="/led_control" class="button">LED Control</a>
  <a href="/thermometer" class="button">Temperature</a>
  <a href="/force_sensor" class="button">Force Sensor</a>
  <a href="/dc_motor" class="button">DC Motor</a>
  <a href="/servo_control" class="button">Servo Control</a>
</div>

<div class="section-container">
  <h2>Multi-Component Webpages</h2>
  <a href="/dc_motor_temp" class="button">DC Motor & Temperature</a>
  <a href="/pressure_led_control" class="button">Force Sensor and LEDs</a>
</div>

<div class="section-container">
  <h2>Extra</h2>
  <a href="/meet_the_team" class="button">Meet the Team</a>
</div>
```

3. Open the new file that has been created, it needs to be coded to send broadcast messages back to the main .ino file. This message can look different depending on the functionality the user wants to see but an example of this can be seen here ->

```
function refreshForceData() {
  if (socket.readyState === WebSocket.OPEN) {
    socket.send('refreshForce');
  } else {
    console.error("WebSocket connection is not open.");
  }
}
```

4. Now that a message is being sent back to the main .ino file, the code needs to be able to handle that message accordingly. Depending on the functionality of this new webpage, it

can look different with different functions. Here is an example that handles the 'refreshForce' message ->

```
if (message == "refreshForce") {
  String forceData = readForceSensor();  // Read the sensor values
  webSocket.broadcastTXT(forceData);     // Send the voltage and force values to client
```

What this does is goes through the "readForceSensor" function to retrieve the desired data and assign it to a string. This data is then broadcasted back to the webpage so the user can see it be displayed.

5. Please reference previously made html files to structure the new webpages to have a similar format, background color, button colors, etc… As well, please reference these html files to understand how the webpage functions, this will help in creating new webpages with different sensors.

6.3.4 How to change the WiFi paraphrase

1. Connect to the WiFi router provided by our team. The current network name is OSU_STEM and the password is 123456789.
2. To change/update the network name and password, search these IP addresses ->
    a. 192.168.1.1
    b. 192.168.0.1
3. One of these two IP addresses will open up a webpage that lets you configure the WiFi settings. Here you can change the password or router name as needed.

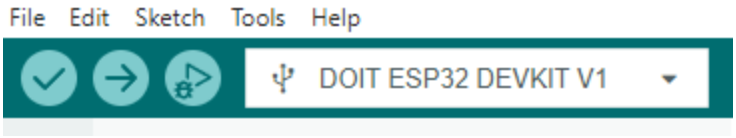
6.4 Troubleshooting

Potential issues:

- ESP32 does not power on upon toggling the switch
    - Likely that batteries need charged

- Cannot get the code to open in the Arduino IDE
    - Make sure the code is downloaded from GitHub and extracted to the preferred location
    - Make sure the "io_playground_ino" file is being opened.
    - In the Arduino IDE take the following steps: File -> Open -> Browse files to find the "io_playground_ino" in the location you saved it. This should open all related files as well

- Getting errors when trying to compile
    - Ensure that all necessary libraries were installed (See above section 6.3)
    - Ensure that the ESP32 is recognized by the Arduino IDE
        - Top left should display "DOIT ESP32 DEVKIT V1



- Webpage does not open when entering the IP in a browser
    - Ensure that the IP was changed in the code if using a new board (See 6.3 above)
    - Recheck connectivity to proper Wi-Fi network
- A sensor is not working / providing data to webpage
    - Check the wires connecting the sensor to the PCB, make sure no wires are broken or unplugged.
    - Validate the sensor is connected to the correct pins. This can be checked by looking at the io_playground.ino code to see what sensors are defined to what pins.
    - Make sure the sensor configuration is correct. i.e. make sure the ground connection is connected to the ground pin and so on.
        - It is possible to plug in the sensors/actuators backwards. If the DC motor is plugged in backwards it will spin in reverse. If the Servo or Temperature Probe is plugged in backwards it might not work. Try reversing plug if errors are present.

6.5 Maintenance

- Changing the Wi-Fi password
    - Type the IP of the router into a browser
    - Should be able to navigate the UI to change Wi-Fi password or network name

- Charge batteries
    - 8 Lithium battery pack lifespan is about 27 hours

- Check for visible damage after use
    - Take note of anything that seems off or out of place within the case

- Check functionality of components periodically
    - Make sure nothing unexpected has occurred which affects components

Chapter 7: Project Management

7.1 Management Overview

Developing and referencing our Team Charter aids us in managing our project and team effort. The team charter laid out assigned roles and how work is divided up. Some roles are more demanding than others, therefore we opted to rotate the responsibilities of Team Lead and

Scribe. This system has worked to allow members to learn and exercise the various skills required. Rotating team lead gives our team flexibility in leadership styles and expertise leveraging. Distributing the work also benefits group cohesion given that there is clear expectation and understanding of workload. Briefly considering each member's skills from prior experience we laid out a basic table to showcase each member's field of interest or expertise that makes assigning topic specific work a smoother process.

Conflict resolution was an important section that provides a clear game plan to follow when issues are found. This has made for minor problems to be addressed and resolved before developing into larger more complex issues. Clear conflict resolutions provide confidence elements to the concerned parties for we have all started this journey with an understanding that problems need resolved quickly for the good of our overall goal as a team.

Communication methods were agreed upon in the Team Charter and have proved highly effective. Our weekly team meetings stay productive, and all concerns are communicated and addressed. Our team has stayed on the same due to our open communication avenues also. The team found our group messages a valuable way to discuss things when working outside of meetings as well.

7.2 Tools Used to Manage the Project

For communication and project management, our team used various tools to facilitate smooth collaboration and organization. Our primary communication channels included Slack and GroupMe, allowing us to engage in real-time discussions, share updates, and coordinate tasks efficiently. Additionally, we utilized Jira as a comprehensive project management platform, where we could track tasks, allocate responsibilities, and monitor progress towards our goals. Alongside these tools, we also found the class schedule and assignment breakdown served as a resource for managing specific tasks or assignments delegated by Alan Gilbert, our stakeholder.

7.3 Work Breakdown, Schedules, Budgets, Assignments

In structuring our project workflow, we implemented a rotating leadership structure to ensure shared responsibility and leadership experience within the team. Initially led by Matthew DiSanto, the leadership role rotated monthly, cultivating a collaborative environment, and empowering each team member to take on leadership responsibilities. Our meetings, held biweekly with sponsors, provided a forum for discussing project progress, addressing concerns, and refining our approach. During weekly meetings, keeping minutes and progress reports ensure continuous accountability for assignments and workflow. Work distribution was tailored to each member's area of focus, centered around equitable allocation of tasks. Furthermore, rotating leaders and scribes for meetings and assignments ensured diverse perspectives and accountability throughout the project lifecycle. Overall, our approach emphasized collaboration, equitable workload distribution, and effective communication to drive project success.

7.4 Tools and processes used to plan the work

High-level goals (epics):

Utilizing the Jira team account, we were able to create progress reports that showed a timeline of work that needed to be done. Each assignment started out as an epic with a specific deadline. Within that epic, child issues (tasks) were created and assigned to each team member. Having Childs within each epic specifies tasks within the project's overall goal, which can be assigned to team members.

Tasks:

Tasks were assigned to team members in an even/fair fashion. The tasks were divided up precisely to not give one team member more work than another. Also, team members can collaborate on their tasks and receive help from other team members if needed.

Assignments / ownership:

Each assignment was owned by the current team lead. This means the team leader was responsible for ensuring the assignment was completed on time and that the work was up to the quality of standards. Also, the leader was responsible for submitting the assignment when it was due unless other arrangements were made.

Tracking process:

Once an assignment or task has been completed. The Gannt chart (see appendix) would be updated to show which assignments are complete or still in progress. This also helps the team know who still needs to complete their tasks or if it is already finished.

Learning and improving team process:

As time has gone on with this project, the team has fallen into a rhythm of what works and what does not work. Splitting these assignments into tasks assigned to each member allows the workload to not be too heavy on any team member. Additionally, having CATME reviews and Alan as a form of checks and balances requires an equal contribution to the team.

Asynchronous communication:

The team has multiple forms of communication for this project. These include Slack, GroupMe, Jira and Outlook email. Depending on what the topic of discussion is, it decides which platform the conversation will take place. Email is more formal and meant for professors and stakeholders while GroupMe and Slack are used for quicker and less important conversations.

Meetings and meeting minutes:

Meetings will be conducted by the team leader who makes sure all members stay on topic and all key issues of discussion are held. Meeting minutes fall into the hands of the team scribe who describes key issues and deadlines along with filling out progress reports. Whenever a meeting occurs, the scribe will make notes about the conversation and share them with the rest of the team. The progress reports inform Alan of how the team allocates time, allowing Alan to guide our work and priorities.

Documenting research results:

Each member was assigned to research various parts of this project and document the research in section 1.2. Later, the team made decisions for the Preliminary design in section 1.3. These decisions were made based on the research found and members who had the most expertise in that area. These were then translated to Jira through assigned Childs and weekly Sprints.

7.5 Resources required outside of the capstone lab

At this point in the project, we contacted Alan and Jonah to incur what resources the lab would have. It was determined for our testing phase we only needed to order a temperature sensor and pressure sensor. The lab has enough resistors, capacitors, microcontrollers, and other starting materials to begin analyzing inputs. The only parts we have not ordered or found in the first testing phase are the choice of power source/ charging and the PCB board. This is due to further research on the best power source needing to be conducted, and difficulty finding appropriate parts through the allowed order forums. More elementary, we have used our computers or iPads to work on these documents to collect and compile information. Also, our sponsor Mark Morscher, along with another member of the STEM outreach program, Clayton Greenbaum, have been pivotal resources in the initial development stages for this project. With their previous experience, they were able to guide our team in the right direction to make quick decisions that propelled the team's progress.

7.6 Risk identification, mitigation, and management

Risks for this project were identified when we identified the market requirements and rudimentary subsystems included in the project. These risks are identified in the table via Appendix a.2. Most risks were mitigated through research delegated by each teammate's skillset and specialty. Examples include having the microcontroller utilize an embedded Wi-Fi system that students can connect to on ANY device or creating correct and bug free code that runs the same every time. Other risks were identified through research on current solutions. This means we will have to look at things like cost, features, and ease of use and determine how our project will stand out. Our risks will continue to be mitigated through research, testing, and debugging, and hands on experience with the project's subsystems.

7.7 Problem resolution

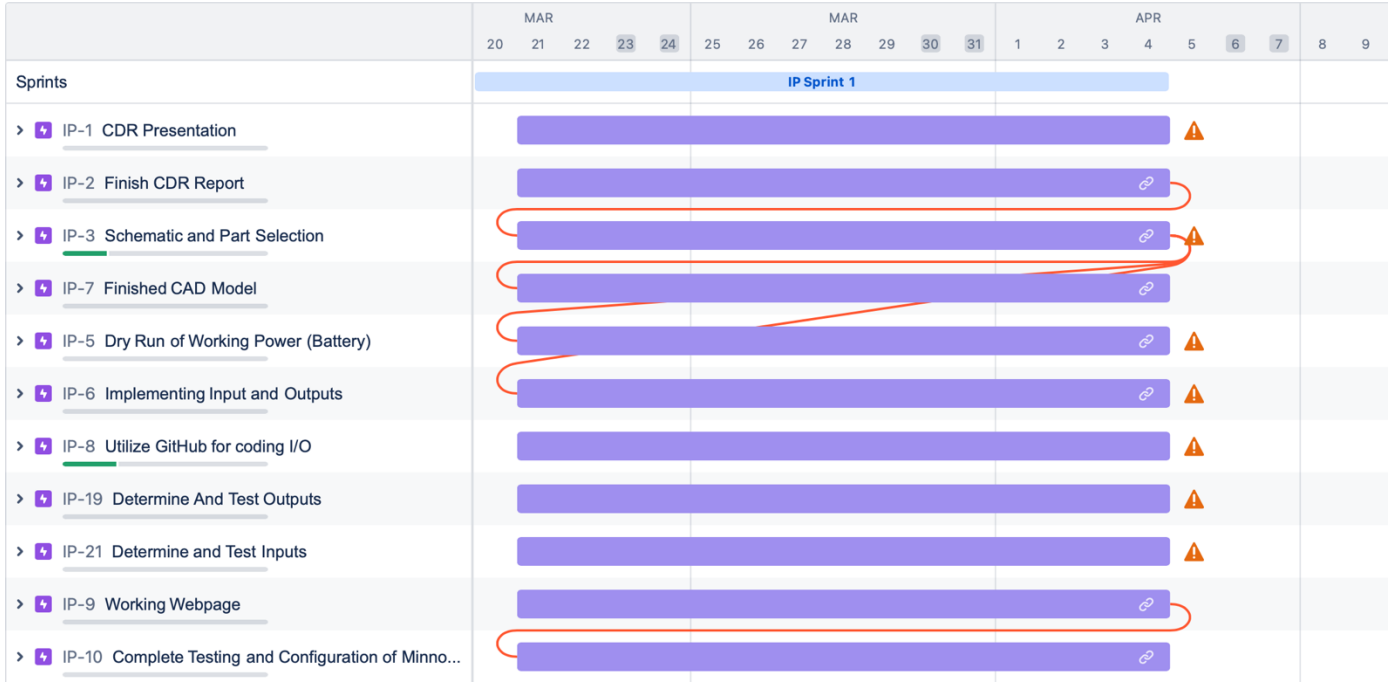
There have not been many setbacks to delay the team thus far. The only notable issue encountered was the pivot from the original project. In the beginning, the team was tasked with improving the traffic light. After two weeks of research, it was decided that there was not a viable solution to this 'super problem.' After consideration from Alan's pitch, it was decided that we would switch to the STEM Playground. It was not until then that the group could make progress towards planning a final design.

7.8 Things to change based on learned experience

The number one thing the team would do differently if we could go back to the beginning of the semester would be work distribution. For many of the early assignments, the workload was not evenly split, leaving some members with little to contribute. This was solved after it was brought up during a team meeting. However, knowing the correct way to divide work from the start would have been ideal.

Appendix A

Figure A1.1-A1.2: Team Gantt Chart



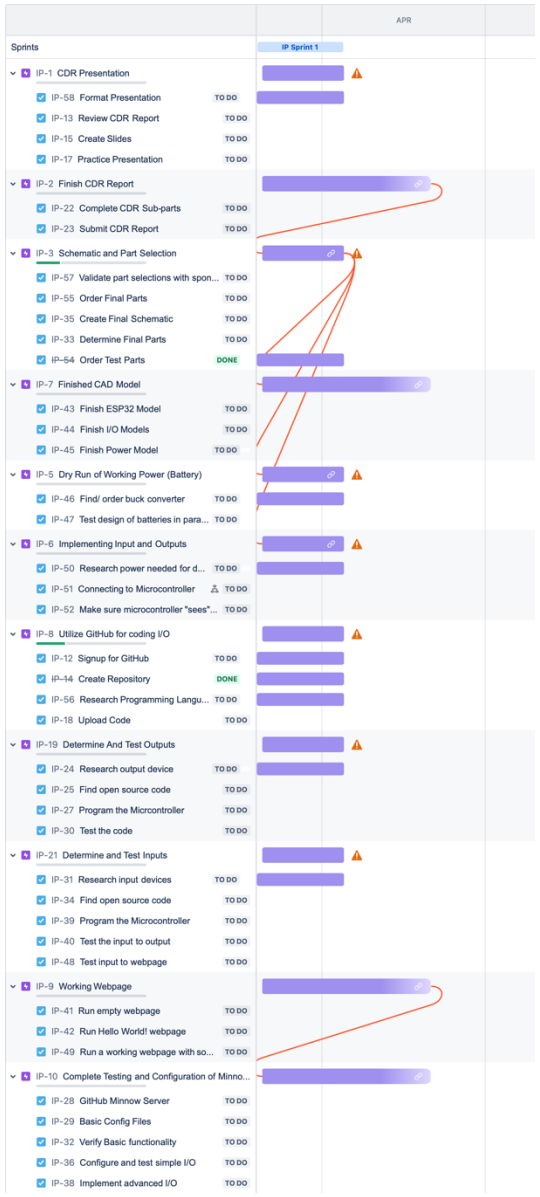

APR
Sprints
IP Sprint 1
IP-1 CDR Presentation
IP-58 Format Presentation TO DO
IP-13 Review CDR Report TO DO
IP-15 Create Slides TO DO
IP-17 Practice Presentation TO DO
IP-2 Finish CDR Report
IP-22 Complete CDR Sub-parts TO DO
IP-23 Submit CDR Report TO DO
IP-3 Schematic and Part Selection
IP-57 Validate part selections with spon... TO DO
IP-55 Order Final Parts TO DO
IP-35 Create Final Schematic TO DO
IP-33 Determine Final Parts TO DO
IP-54 Order Test Parts DONE
IP-7 Finished CAD Model
IP-43 Finish ESP32 Model TO DO
IP-44 Finish I/O Models TO DO
IP-45 Finish Power Model TO DO
IP-5 Dry Run of Working Power (Battery)
IP-46 Find/ order buck converter TO DO
IP-47 Test design of batteries in para... TO DO
IP-6 Implementing Input and Outputs
IP-50 Research power needed for d... TO DO
IP-51 Connecting to Microcontroller TO DO
IP-52 Make sure microcontroller "sees"... TO DO
IP-8 Utilize GitHub for coding I/O
IP-12 Signup for GitHub TO DO
IP-14 Create Repository DONE
IP-56 Research Programming Langu... TO DO
IP-18 Upload Code TO DO
IP-19 Determine And Test Outputs
IP-24 Research output device TO DO
IP-25 Find open source code TO DO
IP-27 Program the Mircontroller TO DO
IP-30 Test the code TO DO
IP-21 Determine and Test Inputs
IP-31 Research input devices TO DO
IP-34 Find open source code TO DO
IP-39 Program the Microcontroller TO DO
IP-40 Test the input to output TO DO
IP-48 Test input to webpage TO DO
IP-9 Working Webpage
IP-41 Run empty webpage TO DO
IP-42 Run Hello World! webpage TO DO
IP-49 Run a working webpage with so... TO DO
IP-10 Complete Testing and Configuration of Minno...
IP-28 GitHub Minnow Server TO DO
IP-29 Basic Config Files TO DO
IP-32 Verify Basic functionality TO DO
IP-36 Configure and test simple I/O TO DO
IP-38 Implement advanced I/O TO DO

Table A1.3: Risk Analysis Table

	Pre-Mitigation (Present)				Post Mitigation		
Risk	**Complexity**	**Unfamiliarity**	**Risk Score**	**Mitigation**	**Complexity**	**Unfamiliarity**	**Risk Score**
Microcontroller that does not perform as the team expects.	2	6	12	Perform investigation/research on different microcontroller capabilities to make an informed decision	1	3	3
Coding in a language (such as JavaScript) that the team has little to no knowledge on.	5	9	45	Finding an open-source code to learn and understand the language and how to implement it in our own program	3	6	18
Creating correct and bug free code that runs the same every time (depending on inputs).	8	6	48	Extensive testing and debugging to identify issues and fix them	4	3	12
Create a web interface that students can access to play with the microcontroller.	6	10	60	Conduct research, test, reach out to other students/peers for guidance to create a well-designed web UI	3	7	21
Device is not easily connected to and interface with.	3	5	15	Purchase a microcontroller that already has a Wi-Fi system embedded. Host local Wi-Fi from students and the microcontroller to connect to	2	2	4
Lesson is not clear and concise so students can learn how the playground functions.	3	6	18	Speaking with educators and STEM programs that develop lesson plans, lower unfamiliarity	1	4	4
Total reproducible cost is less than $50	5	2	10	Get a working playground first then worry about how to decrease the cost/budget of each playground	2	1	2

Lesson run time is less than 45 minutes to new students/ teachers	6	7	42	Practice and test the playground to make sure it meets the time requirements	4	4	16
Understanding and implementing various sensors	7	5	35	Decide best input sensors and how intensive associated code is	5	3	15
Power consumption with outputs	3	4	12	Use low power components and have a strong power source	2	3	6

Appendix B

To Do for Capstone II

1. Design PCB
    a. Power Module
        i. Voltage regulator and capacitors
        ii. Battery pack
    b. Microcontroller Socket Mount
    c. Sensor Sockets
    d. Output Sockets
2. Have webpage hosting simplified with router and not requiring existing network
3. Learn how force sensor analog voltage can be displayed through ESP32.
4. Learn how temperature sensor voltage can be displayed through ESP32.
5. Learn how to use LCD to display various messages/graph.
6. Learn CSS to make webpages more aesthetic.
7. Develop Lesson Plans for specific sensor/actuator usage
    a. Develop lesson plan for student to design a control circuit
        i. If this temperature happens, then this light comes on
            1. Webpage has an input box for a temperature value as a limit
8. Have a certain webpage that shows students “ohm’s law” with device interaction
    a. Animation of our circuit with simple resistor, light and the current resulting