ECE4440/4991 – Capstone

Educational Multiplayer Outlet (EMO)

Table of Contents

Table of Figures

Statement of Work

Salvador Adrian

My contributions for EMO can be seen in the hardware, specifically: the PCB design, the microcontroller selection, the body of EMO, the embedded software, the power design, and the testing of all hardware components separately and together. The PCB design involved reading datasheets to design USB-C receptacles for powering the screen and the Raspberry Pi, as well as for selecting a barrel jack compatible with our wall plug-in transformer to power the circuit. Selecting the Raspberry Pi 4 Model B, relied on considering the technical needs of our project which were compatibility with the screen, computational power for rendering and storing the game, and affordability with the project's budget. The body of EMO was completed at the Architecture school using laser cutting and the FAB lab woodworking tools, which involved designing Rhino files for laser cutting, collecting and purchasing wood, and carefully measuring and processing the wood. The embedded software engineering involved flashing the OS for the Raspberry Pi on an SD card, configuring the GPIO pins to work with a momentary switch to implement an off and on feature, configuring the github repository on the Raspberry Pi, and having the game launch on startup.

The power design involved selecting a wall plug-in transformer capable of providing sufficient power for all devices; this meant designing with max current draw of the USB-C receptacles and Raspberry Pi. The testing of all of the components meant using an ohmmeter to measure the voltage from the USB-C receptacles, the screen, and microcontroller wires, checking the voltage output for the controllers, and verifying the software. My contributions ensured that the build and hardware could integrate the software and accomplish the project goal of an effective game based learning console.

Sara Inoue

For EMO, my main contributions were coding the educational game on the PICO-8 program and creating the narrative of the story. The coding aspect included the visuals, storyline, animation, questions, and textboxes of the game. First, in order to figure out what game we wanted to make, I brainstormed multiple storylines that would be entertaining and informative for young students. After discussing with our group, we decided on a spin-off of the “Three Little Pigs and the Big Bad Wolf”. The script was written for the narrative except for the climax which was written by Jennibelle. After the script, the concept designs for the prologue, level one through level three, and the climax was mapped out. I also drew the interactive obstacles, sprite designs, and their animations–walking animation, idle animation, jumping and knocking animation. As the prologue is a cutscene, I ensured that the user was not able to control the character but rather follow the storyline and the text explaining the story. The textboxes were coded to appear based on the coordinates of an x and y axis. Throughout the prologue, I created simple animations to give life to the story instead of simply having characters stand still and move left and right. I added jumping when a character exclaims

something, I also created a movement and animation combo that would create a knocking animation, and I also made an animation that will make it look like a door opened from excessive knocking. Finally, in order for this storyline to be an educational game, I created the logistics of instantiating a question. The question fills up the screen and allows the player to choose an answer from A to D. If the user get's the question incorrect, they are given a hint and are able to answer the question again. We created this game with the purpose of positive reinforcement so we do not have punishments for the players if they get the question incorrect. Rather, we simply remind them of the section of the story they got incorrect for the player to follow along with our narrative.

Jennibelle Khuu

I helped design and code the game by coding the collision functions, player movements, object movements, camera movements on PICO-8 for all three levels to ensure that the game can be played seamlessly by two players and is compatible with the two joy-con controllers we selected and bought. I also developed the game's climax, falling action, and resolution stages. For player interactions, I created unique mechanics for each character where Player 1 (Pip) can squeeze through small openings and flip switches to move platforms, while Player 2 (Pete) can push boxes to anchor seesaws or overcome obstacles like cliffs and ditches. For example, in Level 1, I implemented a seesaw mechanic where its tilt depends on player positions or box placement, requiring teamwork to stabilize it. In Levels 2 and 3, I added switch-triggered moving platforms and increased the teamwork dynamics by requiring players to push boxes simultaneously and activate switches together to progress. I also ensured smooth camera transitions between each level by panning the view once both players flipped their designated switches. For the climax stage, I introduced a boulder mechanic where a falling boulder breaks a bridge, pushing players into a ditch. I then implemented a flashlight effect, darkening the screen except for a transparent circle around Player 1, requiring both players to stick together, find a ladder, escape the ditch, and jump over the boulder. I incorporated dialogue throughout this stage, including pop-up dialogue when players meet the antagonists after overcoming the boulder.

Additionally, I created the start screen, which displays three images of a smiling face before transitioning to the main menu. On the main menu, I made sure players can select options like 'Start Game,' Instructions,' and 'Level Selection.' Selecting 'Start Game' loads the prologue stage, while the 'Level Selection' page features a plot diagram where players can choose between the prologue, Level 1, Level 2, Level 3, the boulder stage, or the post-game questions. Each stage is positioned on the diagram to show its place in the story's narrative. I also created the 'Instructions' page that shows the basic controls of the game using the joy-con controllers. I also refactored and organized the code into separate PICO-8 files, with each file representing a different level or stage. This setup allows the files to automatically load sequentially as players progress through the game. I also selected and ordered the Raspberry Pi screen, ensuring it was large enough for two players to play simultaneously and compatible with our Raspberry Pi microcontroller.

Joyce Park

Throughout our Capstone project, I specifically helped out with PCB design and development. I made the PCB with all of the required components (chosen by Sal) on KiCad, and shipped it out to JLC PCB to be delivered the following week. I also made the video game console housing with Sal by going to the Architecture School multiple days out of the week to cut/laser cut, sand, glue, and paint the wood. The 3D printing designs for the arms and legs were made through SolidWorks, and were later printed at the NI Lab. Throughout this process, I bought and delivered the Nintendo controllers, PCB, and art supplies for the video game console housing.

Abstract

For our Capstone project, we made a two-player video game console based on BMO, which is one of the main characters from the hit TV show, “Adventure Time”. The console has a retro feel to it, including a fun, educational 8-bit game that promotes collaboration between players. The game, and the console, is controlled by the two USB plugin joystick controllers attached to the console. The finished product of our project is called EMO, which stands for Educational Multiplayer Outlet.

Background

Due to the lack of in-person social interaction between toddlers during COVID-19, children now experience a decline in social skills. According to the Baylor College of Medicine, “Because in-person events and interactions were limited during the pandemic, people missed out on opportunities to develop social skills. While schools held classes remotely, children lost important time learning how to effectively relate with their peers and engage with authority figures like teachers” (Chiu, 2023). This informed us about how the pandemic affected not only adults but many students and young children. Additionally, “evidence from the COVID-19 pandemic demonstrates that education as a process is closely tied to well-being, with schools being a venue for social contact and participation in socialization and development of executive function” (Hayre, 2023). Through reading multiple articles, we were informed about the lack of social skills between young children that were not able to socialize as well with peers through the education system during COVID-19. Therefore, this project involves making a two-player video game console that would help mediate the lack of socialization due to COVID-19 and lack of engaging educational activities provided to the students. Similar projects that have been done in the past include mainstream video game consoles, such as the Xbox, PS5, etc. However, aside from them, our video game console can only be played with two players, forcing social interaction between users. Our video game console includes educational games targeted towards 3rd graders around the ages of 8-9, promoting opportunities for learning during free time in the classrooms.

In terms of our coursework background, all four of us have taken numerous electrical engineering and computer science courses, including embedded courses that specifically goes into exactly how the two concepts intersect. We have also had numerous internships working

with software and hardware.

Project Description

Hardware

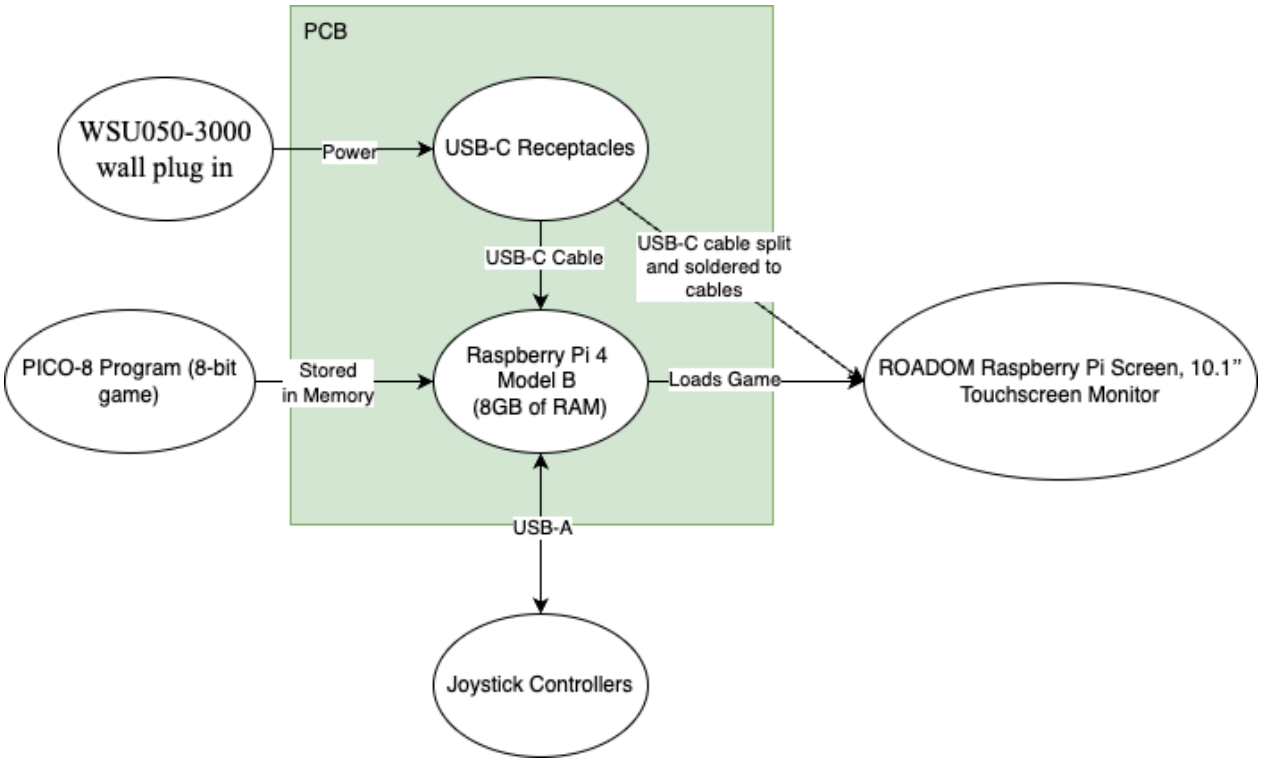


Figure 1: Block Diagram of the Hardware Connections and PICO-8 game load and storage

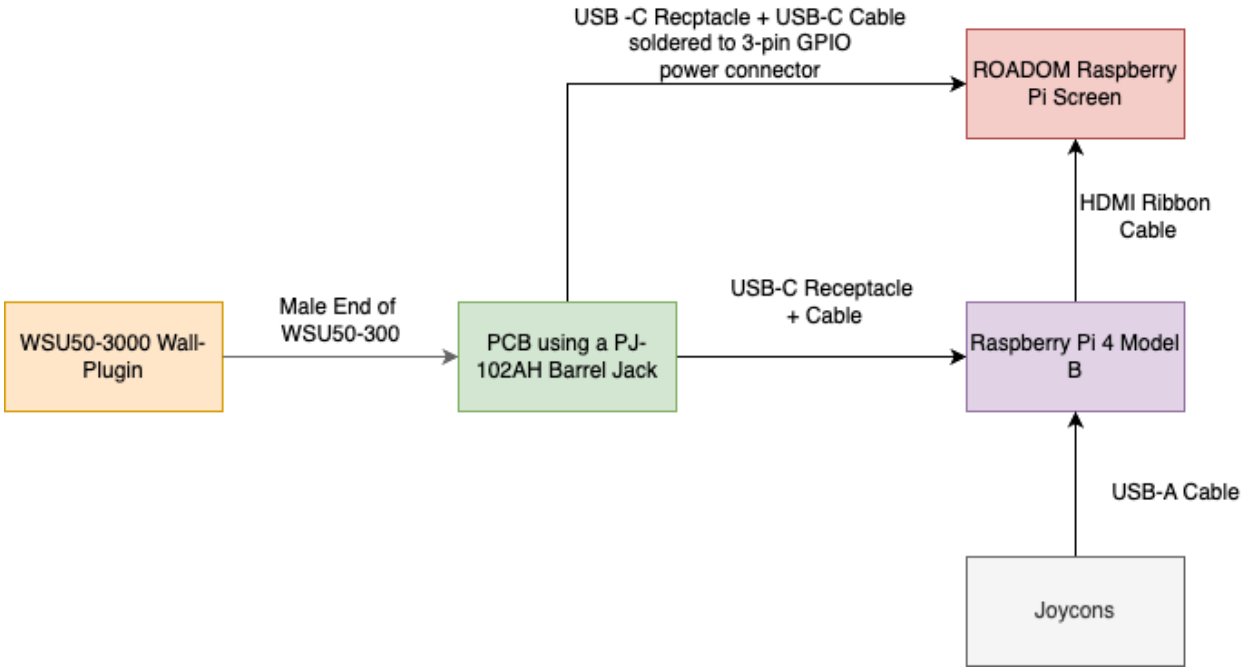


Figure 2: Block Diagram of the Hardware Connections

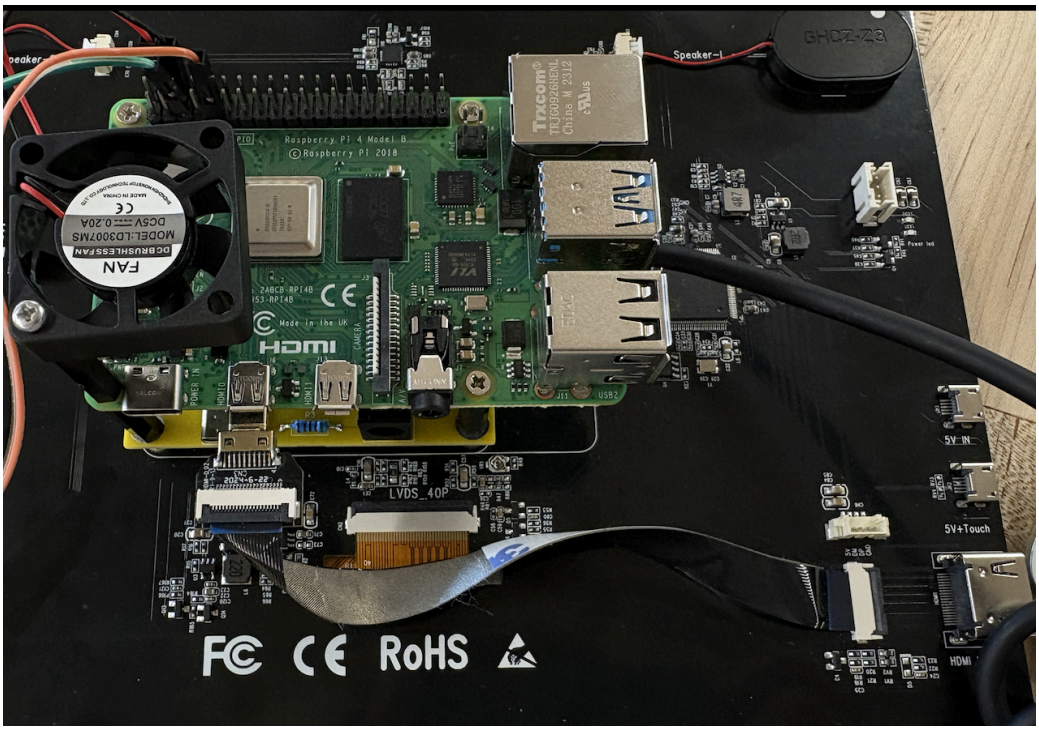


Figure 3: HDMI Ribbon connection to the screen.

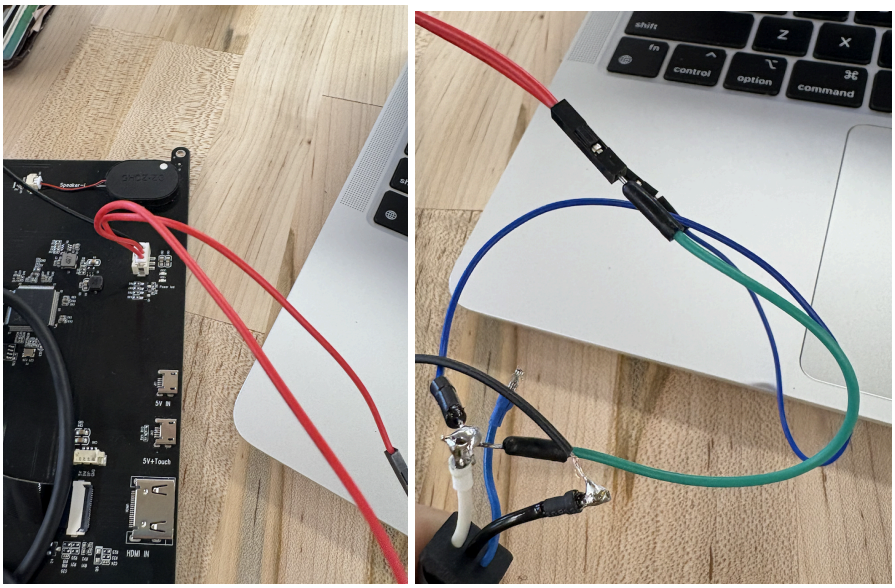


Figure 4: Special USB-C End Soldered to 3-Pin GPIO Cable

Figure 1 shows the overall block diagram of the hardware connections we begin by using a WSU050-3000 wall plug in transformer to take in the wall voltage rating of 90–254 VAC and current of <0.5A (RMS) at 115VAC. The wall plug in transformer brings the input voltage and input current down to 5V DC and 3A which will be fed into a PJ-102AH barrel jack which is compatible with the male end of the wall plug-in. The barrel jack is attached into the PCB which powers two USB-C receptacles. One USB-C receptacle is used to power the Raspberry Pi Model

B. The second USB-C receptacle is used with a USB-C cable except the other end was stripped to expose the positive wire and ground, this stripped cable was ordered online but can also be done by stripping an existing USB-C cable. The exposed endings were soldered to a jumper wire so that it can be connected with the 3-pin GPIO cable and connected to the power port as shown in figure 4, as well a HDMI ribbon cable is provided to transmit data between the microcontroller and the screen as shown in figure 3.

Now that the power circuit has been properly set up the Raspberry Pi was set up using an SD card. The SD card was first written on before being inserted into the microcontroller. The SD card was flashed with the Raspberry Pi OS which can be found on the Raspberry Pi website under “Software”. The process of flashing presents various prompts and ends with the naming of the microcontroller. Once that was complete, the SD card was ready to be inserted in the microcontroller in the SD port located under the microcontroller. We required a keyboard and mouse to navigate the desktop of the OS. Otherwise, we used the USB-A cable provided by the ROADOM screen to power the touch screen and use the on-screen keyboard. This desktop is similar to most OS and can be used to store the PICO-8 game’s github repository in the files as we would on a laptop.

In order for the game to be launched the PICO-8 engine was downloaded from the PICO-8 website. This was achieved by launching the OS internet explorer and navigating to the PICO-8 website and downloading the Raspberry Pi version. Once PICO-8 was set up, through the terminal, the Raspberry Pi was configured to boot on CLI by launching the configuration tool by bash: ‘sudo raspi-config’ from there we navigated to System Options -> Boot/Auto Login -> Console Autologin and set to CLI. For the startup script, we ran in bash ‘nano ~/start_pico8.sh’ then we added the lines and saved ‘#!/bin/bash/home/pi/pico-8/pico8 -run /home/pi/pico-8/carts/yourgame.p8’. Next, the file was made executable by running ‘chmod +x ~/start_pico8.sh’.

Lastly, the Joystick controllers are “Newegg for the Retro Gamepad Joystick Raspberry Pi Gamepad Controller” and were connected to the USB-A ports on the microcontroller. Figure 2 shows a simple diagram of how all of the components are connected in a high level for reference.

According to the product description on Newegg for the “Retro Gamepad Joystick Raspberry Pi Gamepad Controller”, it can connect to a Raspberry Pi by a standard 2.0 USB port. It comes with a 5.6ft long cord, which is long enough for playing video games at a comfortable distance, and it comes with two controllers which are needed for our teamwork-based two player video game. The Raspberry Pi 4 Model B is known to have more RAM for better performance. It is also compatible with PICO-8, making it easier for us to code for the Raspberry Pi.

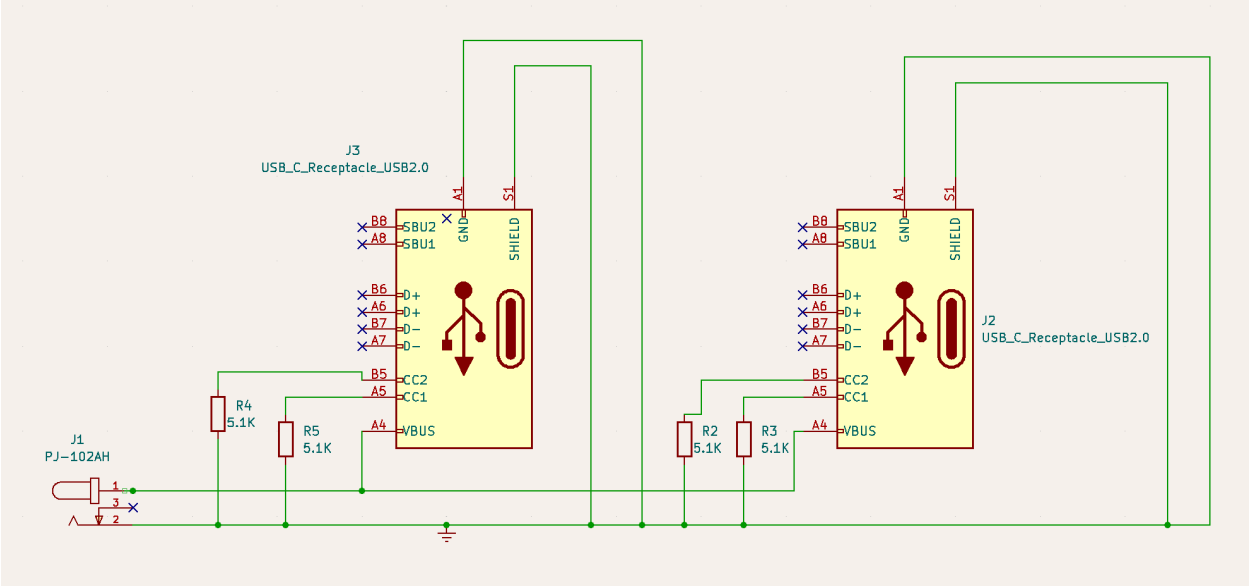


Figure 5: PCB Circuit Schematic

Figure 5 shows the schematic for the PCB layout. J1 in the leftmost part of the figure is the DC power connector for our wall plug in (transformer). The three pins on J1 are the power (pin 1), switch (pin 3), and ground (pin 2). Pin 3 has no connection because this pin serves as a switch for projects where a device may have multiple power sources. If our device were to have a battery, when the wall plug jack is connected to the DC power connector that switch is disconnected from pin 2 to properly direct the power source. Since our design does not include another power source pin 3 will not be connected. Lastly, the wall plug in, WSU050-3000, is rated for 5V/3A and gives us a total power of 15W ($5V * 3A = 15W$). We power three total devices: a screen and two controllers. The screen uses a micro usb cable and the two controllers use a USB-A cable. Assuming the maximum power is drawn from all cables the total power consumption sums up to

$$P_{usb_{micro}} = 10W,\ P_{usb_a} = 2.5W \rightarrow P_{Total} = P_{usb_{micro}} + 2(P_{usb_a}) = 15W.$$

Therefore, the total power consumed is met with our current power supply assuming all cables draw the max power.

Table 4-21 Sink CC Termination (Rd) Requirements

Rd Implementation	Nominal value	Can detect power capability?	Max voltage on pin
± 20% voltage clamp[1]	1.1 V	No	1.32 V
± 20% resistor to GND	5.1 kΩ	No	2.18 V
± 10% resistor to GND	5.1 kΩ	Yes	2.04 V

Note:
1. The clamp implementation inhibits *USB PD* communication although the system can start with the clamp and transition to the resistor once it is able to do *USB PD*.

Figure 6: USB-C Sink CC Requirements

Next, there are two USB-C receptacles that are configured to be sinks to provide power. The only pins configured are the CC and Vbus pins. The CC pins are for configuring the USB-C receptacle. By using 5.1kΩ, we set the receptacle to be a sink as instructed by the datasheet and shown in figure 6. As for the D+ and D- pins this is for data transfer, USB-C receptacles can be used to receive data over the cable connection; however, given the scope of our project no data transmission is needed for either receptacle as they are just to power the components. Therefore, the D+ and D- pins will not be configured. The same reasoning applies to the SBUS pins; they will not be connected as they are not needed to power the components.

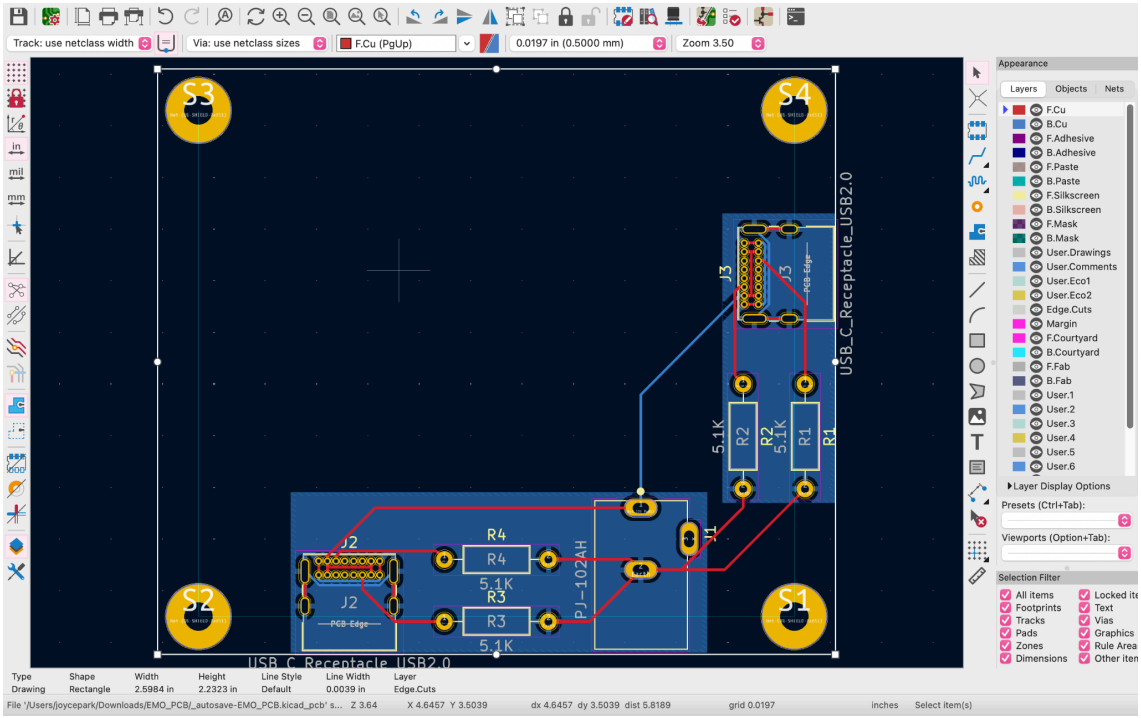


Figure 7: PCB Layout

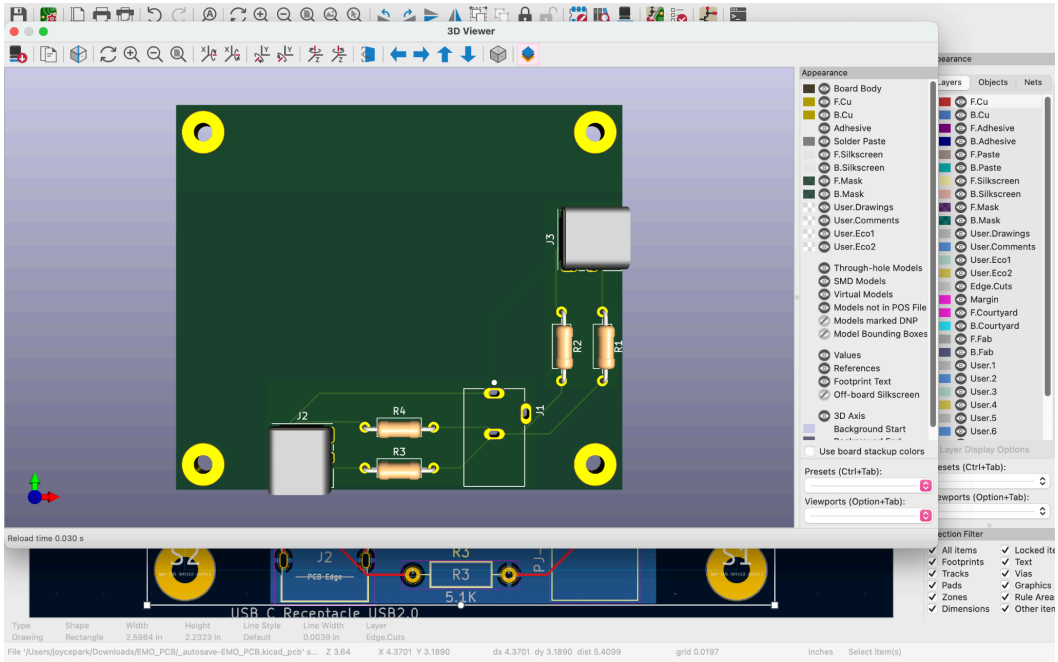


Figure 8: 3D Model of PCB

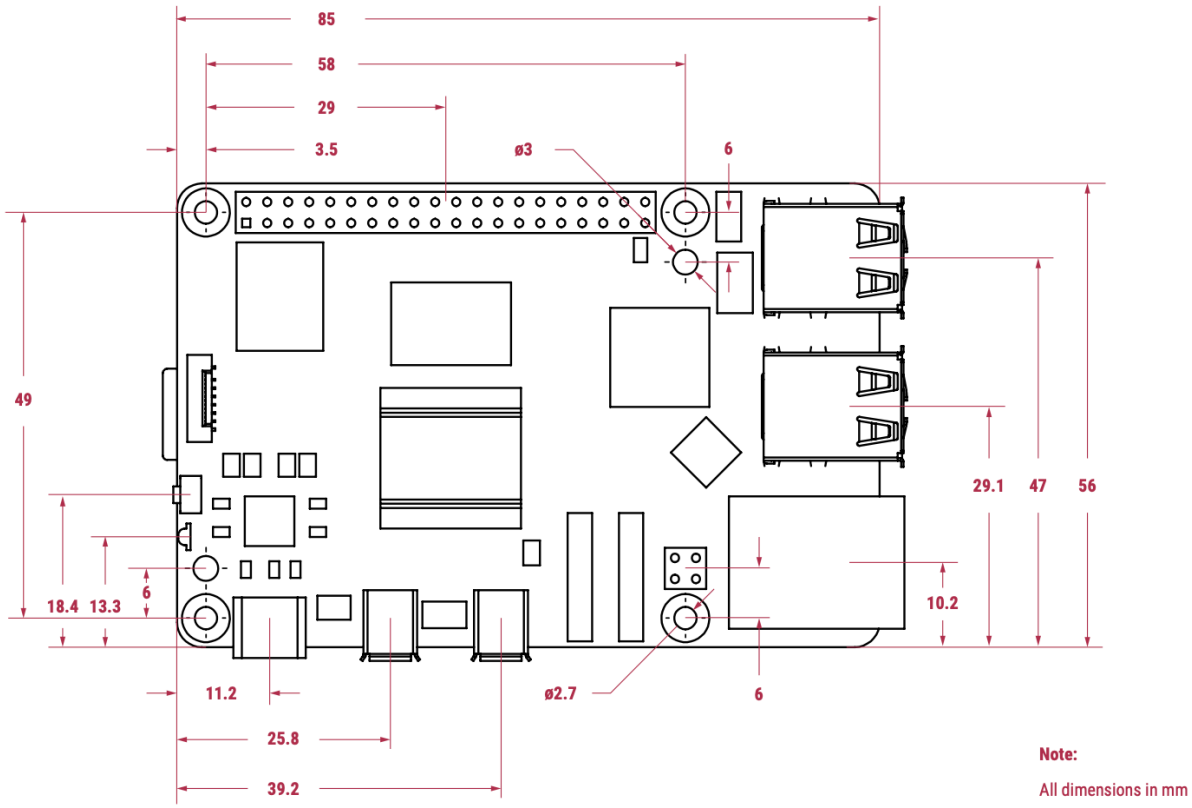


Figure 9: Mechanical Drawing of the Raspberry Pi 4 Model B

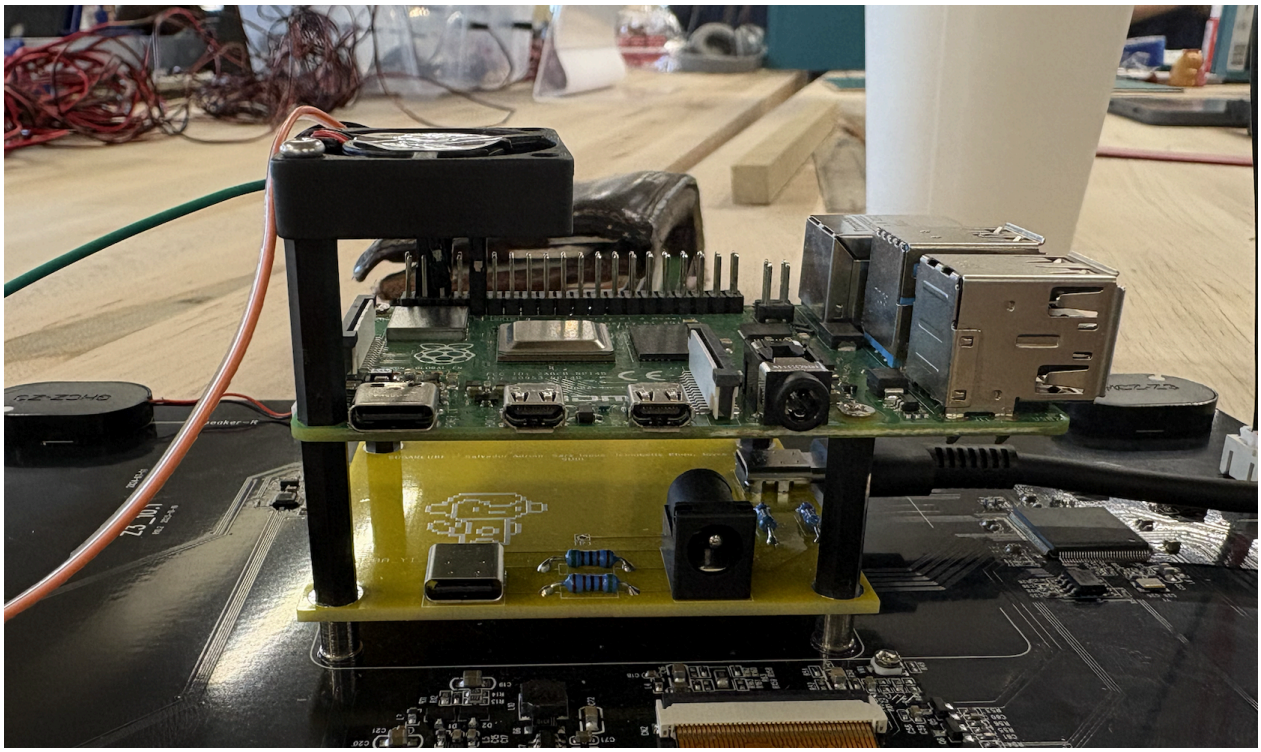


Figure 10: Stacked PCB and Raspberry Pi

Figure 7 and 8 show the layout of the PCB and its tracings. The PCB is designed so that the USB-C receptacles and barrel jack are weighted on the edge of the PCB and not in the center. The

four corners and the size of the PCB is matched to that of the mechanical drawings shown in figure 9. The purpose for this is so that the PCB and the microcontroller can be stacked on one another using NACX M2.5 X 20mm + 6mm male female hex thread nylon space standoffs. When combining that we can attach it to the screws design on the back of the ROADOM screen and leave it stacked as shown in figure 10.

Video Game Console Housing

The video game console housing was made to look like BMO, who is characterized by his boxy figure and short and thin arms and legs. This was done by cutting wood for all sides of the box, and later laser cutting out holes for the screen, USB ports, etc. A band saw was used to cut down the wood, and planners were used to process the wood so that it was level. The sides of the box were made from free wood panels cut to be (15 x 4.75 x 0.875) inches, whereas the top and bottom of the box were cut to be (10 x 5 x 0.875) inches. The front and back of the box were made from store-bought plywood cut to be (15 x 11.5 x 0.125) inches. In the meantime, buttons were laser cut for the front of the box, including a large D-pad, large and small circle, and small triangle by uploading laser cut files on Rhino. Once all sides were cut to perfection, the box was assembled together using wood glue. The box sat to dry for at least 24 hours with the help of heavy duty clamps. The box and buttons were then painted their respective colors, and we later mounted all of the hardware components within the box. The arms and legs were 3D printed using SolidWorks, and were later painted to match the color of the box as well.

Software

One of the main goals of this project is to make our game educational, so we chose to focus on improving children's reading comprehension skills by creating a story-based game and aligning the story with reading comprehension questions that are similar to the questions seen in the Virginia Standards of Learning (SOL) for 3rd graders. Reading SOLs are standardized tests in Virginia designed to assess students' reading comprehension skills. In Virginia's public schools, instructions are guided by the Standards of Learning (SOL), which outline the "commonwealth's expectations for student learning and achievement in grades K-12" (Virginia Department of Education, 2022). The types of questions we will ask in the game will be recalling the events that happened in certain levels and indicating what the conflicts and theme of the story were. We analyzed previously released 3rd grade reading SOL questions from 2011-2015 to understand the stories and questions' structure and format. Based on those previous SOL exams, we created our own questions using a similar structure and difficulty level.

We chose to use the PICO-8 game engine for our game because it can run completely offline, allowing both kids and teachers to easily run the game without needing an internet connection. Additionally, its compatibility with Raspberry Pi 4 made it an ideal choice since it's simple to save into the Raspberry Pi and works very well with Raspberry Pi joy con controllers, with very minimal code configurations to work with the Raspberry Pi joy con controllers. This ensures that the game can be easily set up and played in any classroom

environment. Also, its retro pixel style helps make the experience feel more like a video game than a reading comprehension test.

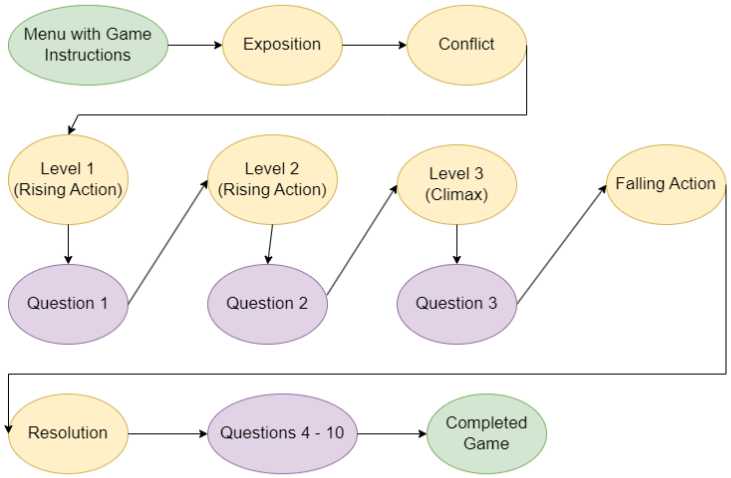


Figure 11: Flow Chart for the Stages of the Game

The game starts off with the menu page with instructions on how to play the game and then it will move onto the story part of the game, introducing the exposition and the conflict of the story. Then, it progresses into the three main levels of the game and each level will contain an obstacle course and also continue the narrative of the story. After each level, the player will answer a reading comprehension question that will ask them to recall what happened in the previous levels. After those levels and questions are completed, the game will progress into the falling action and resolution part of the story and then finally, the players will answer the remaining 7 reading comprehension questions. Once those remaining 7 questions are answered and completed, then the game is completed.

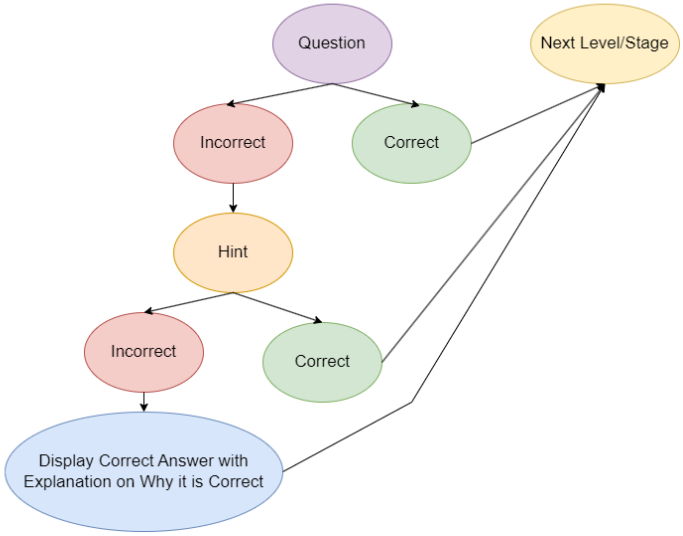


Figure 12: Flow Chart for Each Question

For each question, if the players answer correctly, they will move on to the next question, level, or stage. If they answer incorrectly, a hint will be provided, such as a

passage summarizing what has happened so far to help refresh their memory. If they still get the question wrong, the correct answer will be revealed, and they can proceed to the next question or level. This hint system is meant to help players learn from their mistakes and focus on improving their reading comprehension skills, rather than just worrying about getting answers right or wrong. It creates a positive and supportive environment where players aren't afraid to make mistakes. Instead, they're encouraged to communicate with their teammates and keep learning, even when they don't have all the answers right away.

Another goal of our game was to improve students' teamwork, communication, and social skills. This was achieved by giving each player's character a unique strength, requiring them to rely on each other and communicate effectively to overcome each of the obstacles in the levels. For example, Player 1 can move boxes to balance seesaws or place them under cliffs to help reach higher areas, while Player 2 can use their shorter height to access enclosed spaces and trigger switches to move platforms. There are also levels where both players must push boxes simultaneously to clear their paths and flip switches together to activate moving platforms. Additionally, after each level, both players must flip their designated switches to progress onto the next level or stage, ensuring neither player is left behind. These teamwork based challenges can be seen in Levels 1, 2, and 3 as shown in Figures 30, 31, and 32. Teamwork will also be crucial in the climax stage, where the players are trapped in a ditch and can only see within a circle around Player 1. Player 2 must work with Player 1 to figure out how to escape the ditch, which can be seen in Figure 33. These multiplayer-based challenges foster communication and highlight the unique strengths of each player, enhancing their duo dynamic.

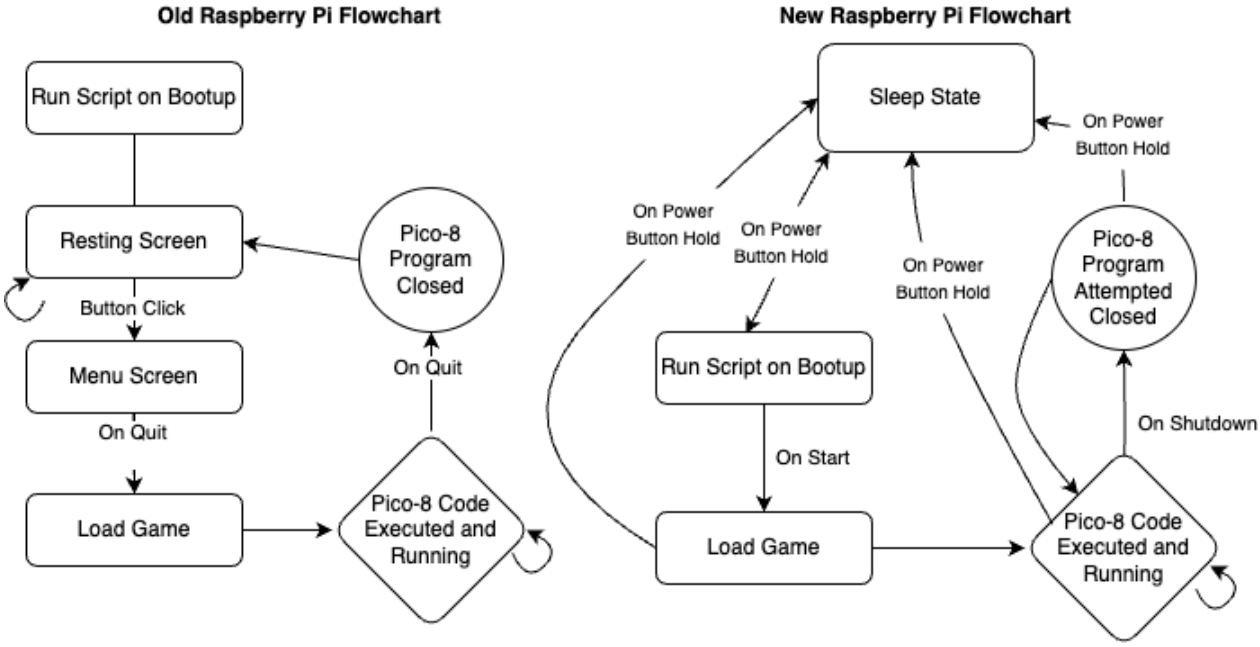


Figure 13: Old vs. New Raspberry Pi Flowchart

The flowchart above shows the old and new logic of the Raspberry Pi on bootup. Before, we did not take the on and off power button into consideration. Additionally, with the old flowchart, we assumed that the resting screen and menu screen would be an external program and would not be within the Pico-8 program. However, after considerations, the resting screen and the menu screen was written on the Pico-8. Now, with the new flowchart we have a sleep state. The sleepstate makes sure no power will flow into the Raspberry Pi. After holding the power button during its sleepstate, the power will flow into the Raspberry Pi and turn the Raspberry Pi on. As soon as the Raspberry Pi is turned on, it automatically runs a script that will launch the Pico-8 menu page cartridge. During this state of running the Pico-8 program, the video game is constantly running. In the situation that the program is complete or the shutdown has been executed, the Raspberry Pi will detect that the Pico-8 program has been closed. To combat the players that may try to access the Raspberry Pi interface instead of the Pico-8 Program, the script is constantly detecting if the Pico-8 program is closed. If this occurs, Raspberry Pi will automatically execute the starting Pico-8 cartridge again. Throughout all these states, if the external power button is held down, the Raspberry Pi will return to its original sleepstate.

Test Plan for Verifying the Project Functionality

The following test plan is mostly the same as our proposal we made nearly three months ago. However, with the changes in our test plan, we added a few comments if anything differed or added something.

1. Software
    a. Video Game
        i. Confirm system compatibility with the laptop first. Then confirm the overall system's compatibility.
        ii. Perform local performance testing and get the framerate, resource usage, and load time to verify smooth performance.
        iii. Gameplay flow testing: Play through each level to confirm no glitches or bugs exist.
            1. Tested multiple ways to try to break the level. Did not run into any issues that allowed the user to run and break the map.
        iv. Game data: confirm saving and database functionality
            1. Returning back to this, we did not end up creating a database to save the information of each student.
        v. Test multiple connections: as it is a two-player game, confirm that the game works with two device connections
        vi. Error handling: confirm that in the situation something unexpected happens, the system will throw an error without crashing.
            1. When an error occurs, Raspberry Pi will rerun the game and reset the state to the beginning.

b. Menu / Nongame Software
    i. UI/UX testing: check if the menu's buttons, navigation, and functionality are working properly.
    ii. Control input testing: confirm if controls are working: First with the keyboard buttons, then with the controller.
    iii. Verify aspect ratio: Confirm that aspect ratio of the menu is properly fitted to the screen.
    iv. Confirm errors are handled properly.

2. Hardware
    a. PCB
        i. Confirm layout design: Make sure that PCB follows the schematic originally created.
        ii. Test Power Supply: Plug into the wall and confirm that power is being drawn.
            1. This was properly tested with a multimeter.
        iii. Test each component incrementally: Confirm that each component in the PCB is drawing power and functioning as expected.
        iv. Test the overall system: Check overall input and output of the system.
        v. Thermal testing: confirm that after long usage, the temperature of PCB does not exceed the burning temperature.
            1. A fan was added to ensure that PCB does not overheat.
        vi. Stress test: make sure that PCB can handle long times of the system.
            1. Stress test was run and the PCB did not cause any discrepancies after running for multiple hours.
    b. Power Supply
        i. Voltage: test and confirm that the amount of power being drawn is proper and within expectation
            1. Multimeter used. Confirmed all results were correct.
        ii. Oversupply situation: Make sure that in the situation of drawing too much current, there is a capability of safety.
            1. Calculated the current that was being drawn. However, according to datasheets, the current level being drawn is not enough to ruin any of the components.
    c. Microcontroller
        i. Confirm power connection
        ii. Firmware: Confirm that the functions and code uploaded to the firmware are properly working
            1. We confirmed that on bootup, the Raspberry Pi automatically boots up our game.
        iii. Peripheral Testing: Confirm that the surrounding systems function

properly.
        1. Sound systems are working properly.
    iv. Test the interrupts.
d. Display
    i. Confirm Power Connection
    ii. Brightness and contrast: Confirm that the brightness and contrast of the display follow what is required.
        1. Display settings were set before inserting the screen into the console box.
    iii. Pixel integrity: confirm that the pixels follow what the system requires from the display.
    iv. Resolution verification: Make sure that the resolution is what is expected.
e. Joystick
    i. Confirm USB is connected to the microcontroller
    ii. Calibration testing: ensure that the calibration of the buttons and system are represented correctly in the UI.
        1. Controller testing done on both laptop and Raspberry Pi.
    iii. Response test: Make sure that the system reacts in a timely manner.
        1. Controller button presses are nearly instantly correlating to the movement.

Physical Constraints

Hardware

When designing our project the hardware team had determined that the max power limit is 15W, which is sufficient given the project's goal. However, if more devices are needed or any other hardware accessories were added the barrel jack and the wall transformer would need to be changed. Furthermore, if too many plugins are plugged into the Raspberry Pi an overcurrent warning icon is prompted on the screen in the upper right corner. This will not affect the performance of the device but the error prompt will occur periodically while the system is on. Therefore there is a constraint on how much hardware the microcontroller can power and operate.

The ROADOM Raspberry Pi screen is sold on Amazon and is a large screen that is affordable at $99.99; however, there is no datasheet available for the screen and the back is designed with the intent of using only the gift cables. Therefore many of the connections made throughout this project had to be custom. The off and on button for the microcontroller was custom made and raised with a piece of wood. If the button breaks there is no way to easily replace it since the wire is custom. Furthermore the OS on the Raspberry Pi is limited to Python and any embedded software engineering must be done in Python. Other languages such as C can be used; however, much of the UI and available coding IDE’s are designed to encourage Python.

Lastly, for cost constraints the two largest purchases were the Raspberry Pi Model B 8gb Ram for $75.00 and the screen for $99.99. The microcontroller can possibly be degraded to a smaller RAM, reducing the cost constraint.

The manufacturing of all parts were received in a timely manner and the hardware team experienced no back logging. In order for this project to go into a production version the code demands would need to be investigated to see how much the microcontroller could be degraded to and screen alternatives that meet compatibility and cost reductions. Wood is a durable and stable material but another material would need to be considered for cost, replication, and still be able to retain the structural integrity.

Video Game Console Housing

When it came to building the housing for all of the hardware components, we obviously wanted it to be based off of BMO's character. However, we did run into a few constraints that could not be ignored. First, we were limited to the free wood at the Architecture School on top of a $30 budget for plywood. This made it very time consuming for us to find the perfect sides for our box, as we had to keep looking for wood panels that met our specifications, such as size, weight, etc. Second, when it came to glueing all of our wood panels together, the glue left our wood with a very uneven surface that could not get fixed even with hours of sanding. Because of this, we needed to even out the surface as best as we could with Modge Podge.

Software

Our game engine, PICO-8, had a few memory, pixel, and color limitations that affected the design of our game. Firstly, each PICO-8 file can only hold up to 65535 characters and 8192 tokens, which meant we had to ensure that each level or stage stayed within this range. If we exceeded these limits, we would have to create a new PICO-8 file for the level, limiting our ability to keep the entire game in a single PICO-8 file. Being unable to keep the entire game in a single PICO-8 file made it difficult for us to keep track of the students' reading comprehension scores throughout each level. We worked around this constraint by making each level and stage its own PICO-8 file, and we did not give the students' any final reading comprehension scores, but encouraged them with hints and presented with them the correct answer if they were unable to figure out the correct answer on the second try.

Another constraint was the font in PICO-8 because PICO-8 only allows a 128x128 pixel display. As a result, we had to use a pixel font, making it difficult to present longer or smaller texts in more conventional fonts like Times New Romans or Arial. This constrained us to shorten dialogues, questions, and answers to fit the space and to only display text in all caps. However, by shortening our dialogue, questions, and answers, we were able to keep the gameplay fun and engaging, helping maintain students' attention span while still preserving its educational value. At a production level, we would aim to make the font more readable, such as in Times New Romans or Arial font to make it easier for the users to read and understand.

Additionally, the 128x128 pixel screen size made it challenging to resize the characters, so we had to keep all the characters the same size throughout the game. This also meant that we had

to manually create every background and character, limiting our design choices for both characters and maps. Moreover, PICO-8's color palette is restricted to just 16 colors, which makes it difficult to create dimensional landscapes and characters with shadows and gradients.

Societal Impact

As the team that designed EMO it was critical to understand that our team JSJS were the client and that students and affiliates of the school were our stakeholders. We successfully designed a gaming system that promotes collaboration and team work while tackling issues within the classroom. Since we defined the problem, it was equally important to consider the stakeholders which include the students who use the system, the teachers who integrate it into the classroom, and the school district administrators who have to approve the device. Understanding and prioritizing our stakeholder needs were essential to the success of EMO and its ability to contribute to the classroom.

For the elementary school students, safety was paramount. Healey et al. (2019) emphasized the importance of considering charactics of toys, its potential misuse, and the obligations of overseeing play or usage, "the characteristics of the toy should be considered as well as how the toy might be used or abused and the amount of supervision or help needed for safe play.". Applying this to EMO, its physical body prioritized the safety of the child and eliminated sharp edges, hazardous materials, or any potential choking hazards. Furthermore, the screen and joycons were appropriate for the students and designed for prolonged use. The human computer interaction of the students was made important; the HCI community emphasized, "the design of educational technologies (ed-tech) provides a second reason for why the community has a pivotal role in the realization of this agenda"(Mechelen et al., 2023). This was accomplished by ensuring that the students only interact with the game software and designing the microcontroller to restrict access to all other external media from the internet. Furthermore, the physical safety of the students was prioritized but so was the psychological safety of the students. As researchers state, "the relationship between video games and culture … as a powerful element for the dissemination and transmission of knowledge, beliefs, ideas, and values" (Cerezo-Pizzaro et al). All games and UX/UI features were made to be inclusive, non-discriminatory, and promote the emotional welfare of the student. This was accomplished by using appropriate language in the dialogue, neutral characters that were animals, a conflict in the story but no violence was promoted, and a completed build with vibrant colors and no offensive or exclusive markings or features. EMO prioritized a teacher's ability to promote student collaboration and enhance student learning by integrating a standard metric for questions, the Virginia SOL; as well as successfully building a multiplayer experience that encourages collaboration throughout levels. EMO's games are engaging and provide team based skills, critical thinking, and educational value. This was accomplished through playable characters with unique abilities for tasks, exploring levels, understanding game dialogue, and creating an immersive learning experience with an emphasis on plot diagrams. The system was made to be user-friendly to both students and teachers for easy usage in classrooms through clear affordances and signifiers for both users such as the power button, the controller plug-ins, the wall plug in, and the screws in the back.

Lastly, for school district administrators, the economic sustainability of EMO was prioritized. The design utilized cost effective material within our project budget and was designed to be easily maintained. Using energy-efficient hardware and sustainable materials minimized its environmental impact and provided an eco-friendly solution which in turn reduced its cost. For example using wall power removed its need for a battery which would degrade and implemented software such that the device can be dormant reducing its power consumption. Furthermore, the parts selected were from trusted companies like Raspberry Pi so that the system is durable and can withstand a long product life cycle. Eco-friendly decisions is worth our design process since the usage of digital devices has increased and will increase as noted by Delgade et al ., "The number of investments made towards purchasing digital devices for students has increased dramatically over the past 25 years, causing rapid growth of the use of technology in K-12 classrooms" (405).

External Standards

For EMO, ASTM F963 is a critical requirement to follow. This shows a list of sections applicable to the toy we are making for children and whether the toy requires a third-party test to ensure its safety. This applies to us because if we want to make our product EMO to be put up to the market, we have to make sure that we follow the Consumer Product Safety Commission (CPSC) regulations. We were unable to ensure this as we would need to send our product to a third-party tester in order to get a license and be approved.

According to the NFPA, the "NFPA 70, National Electrical Code (NEC) is the benchmark for safe electrical design, installation, and inspection to protect people and property from electrical hazards." In order for our product to be passed, we have to make sure that there will be no electrical hazards. We made sure to use heatsinks, electrical tape, and fans to ensure no overheating and no accidental burning with our console. However, we used wood for our casing in order to have our product be within budget. This will likely not pass with the NFPA standards. However, this was done for our capstone project and for future usage, we would experiment with using plastic or any non-flammable material.

The Restriction of Hazardous Substances in Electrical and Electronic Equipment (RoHS) EU rules restrict the use of hazardous substances in electrical and electronic equipment. Although this is not required in the US, it has been a regulation in some states. Namely, California's Electronic Waste Recycling Act. This act restricts the usage of lead, mercury, and some other materials in electronic equipment. During the process of creating our console, we ensured that no materials including the wood, parts, and paint do not include hazardous materials. We ensured that electronic parts are not hanging loose and are closed within the console with a screwed on the backboard.

The IPC standard is used to ensure that the PCB that is used in our system is made reliably and safely inside our consoles. Our screen has screws on the back to mount on the Raspberry Pi as the screen is made to be compatible with a Raspberry Pi. By creating our PCB to be the same size as the Raspberry Pi screw holes, we purchased standoffs and ensured that the PCB was properly inserted and mounted reliably.

If we decide to go with obtaining data from the children, we have to make sure we follow

regulations with the Children's Online Privacy Protection Rule (COPPA). This says that for children under 13, we have to make sure that they follow compliance with data collection. We did not end up doing any data collection for our project due to the lack of time to implement a database, we did not need to find more information about COPPA.

Intellectual Property Issues

Our project will likely be patentable because it could pass through with the "novelty" requirement with regards to a custom PICO-8 game, Raspberry Pi optimizations, and unique controller functionalities. If there was an issue with the "novelty" requirement, our issue would be in correlation with prior art arrangements. An example of correlation with other project is the usage of some connectable controllers and screen based setups were well-documented by Nintendo. We also found that there were already many existing controller and screen based setups, so this could cause issues with the "novelty" requirement. However, after extensive research, there was no patent in correlation specifically with our game console: a wooden box frame, a stationary console that comes with an already inputted screen, the use of a Raspberry Pi microcontroller with a preloaded PICO-8 game installed inside.

Looking at some specific patents that are similar in the aspect of a video game console, we will look at the Nintendo Wii (Akio, 2013). The patent is filed as a "video game system with wireless modular handheld controller". This patent is specifically looking at the remote controller of the Nintendo Wii console. This is similar to our design with 4 directions, an A and B button, and a home button. The Nintendo Wii's controllers sense motion through illumination emitters which are not similar to our controllers that only use button press inputs. Additionally, our console integrates Raspberry Pi hardware with a gaming ecosystem designed for PICO-8, which is a creative and low-resource coding platform. This combination is not addressed in the patented system.

Another patent we looked at was the "Compact hand-held video game system"—otherwise known as the "GameBoy"—with a patent number of US5184830A (Satoru, 1993). This handheld video game takes in a physical cartridge, and has a large emphasis on a portable hand-held video game console. This device is similar to ours in terms of functionality, as it powers on and plays a single game upon bootup. However, our console runs a single game, which can be changed by replacing the Pico-8 game loaded at startup. This requires reprogramming the Raspberry Pi itself, and the console is not designed for handheld gaming. Our device is meant for stationary playing and requires two players in order to complete the video game. Therefore, our project exhibits several differentiating features compared to the "compact hand-held video game system" patent.

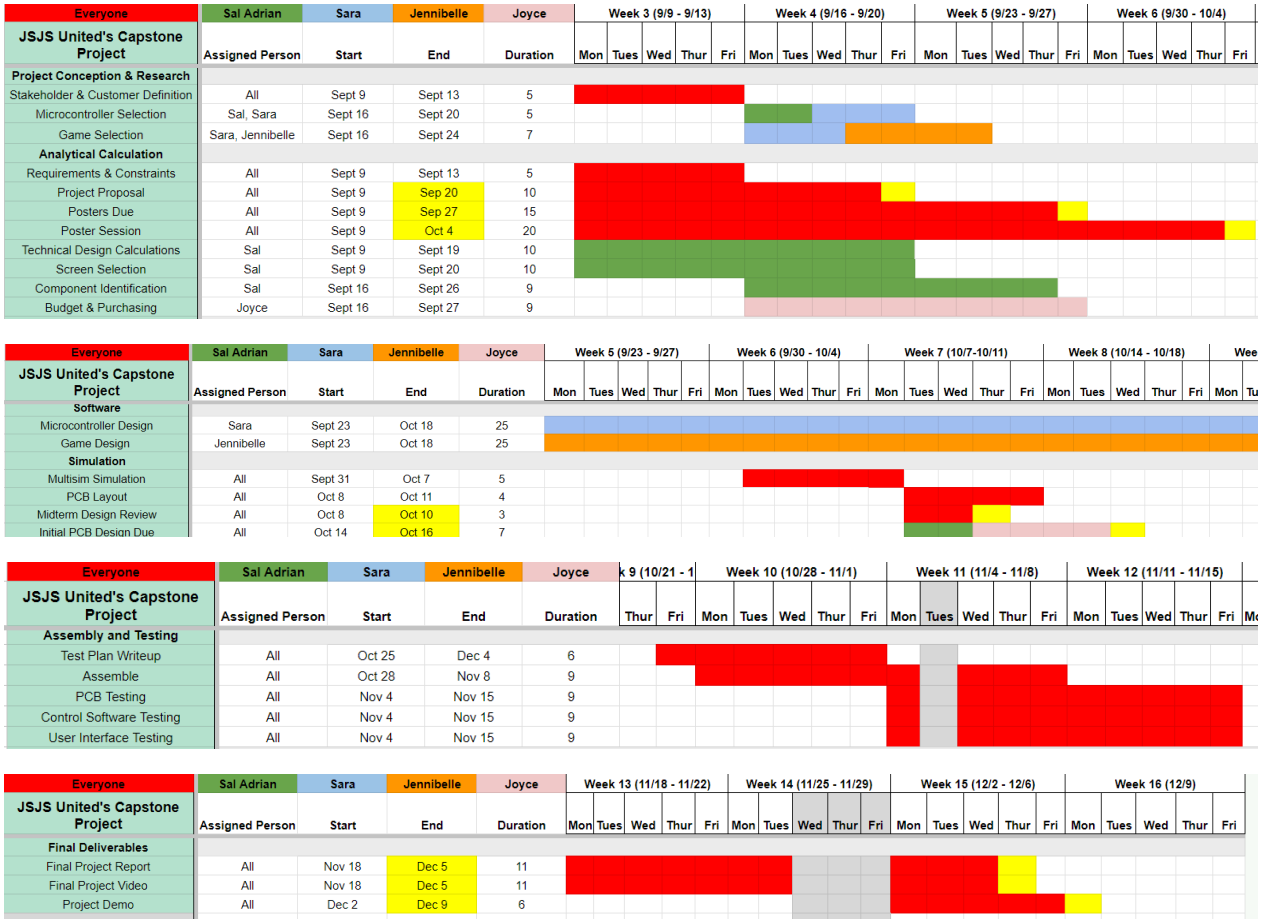
Finally, for a patent related to the software, we looked into Nintendo's video game process patent (Iwao, 2024). This patent talks about the functionality of throwing a Pokéball in Pokemon. This patent was made in 2024 and was referenced in the lawsuit regarding the action of throwing a Pokéball with a separate game called "Palworld". This patent is not in correlation with our video game. However, this highlighted the importance of researching the patents and

copyrights associated with a video game, even if you're drawing inspiration from it. Looking into this incident, we researched Pico-8 licensing. For users who purchase Pico-8, they are allowed to freely distribute their creative games for free or for a price. This helps us reinforce the idea that our video game will not experience patent issues since the video game is a simple story based narrative to support students to have better reading comprehension skills and stronger teamwork skills.

Timeline

This section should include the Gantt chart from your proposal as well as a final chart (showing the differences). You should explain the following and how your timelines changed throughout the course of the semester. Make sure your Gantt chart is legible! You may need to alter the format or break the Gantt chart up into sections so that when it is presented in the report, the text will still be large enough to be read.

Figure 14: Gantt Chart From Proposal



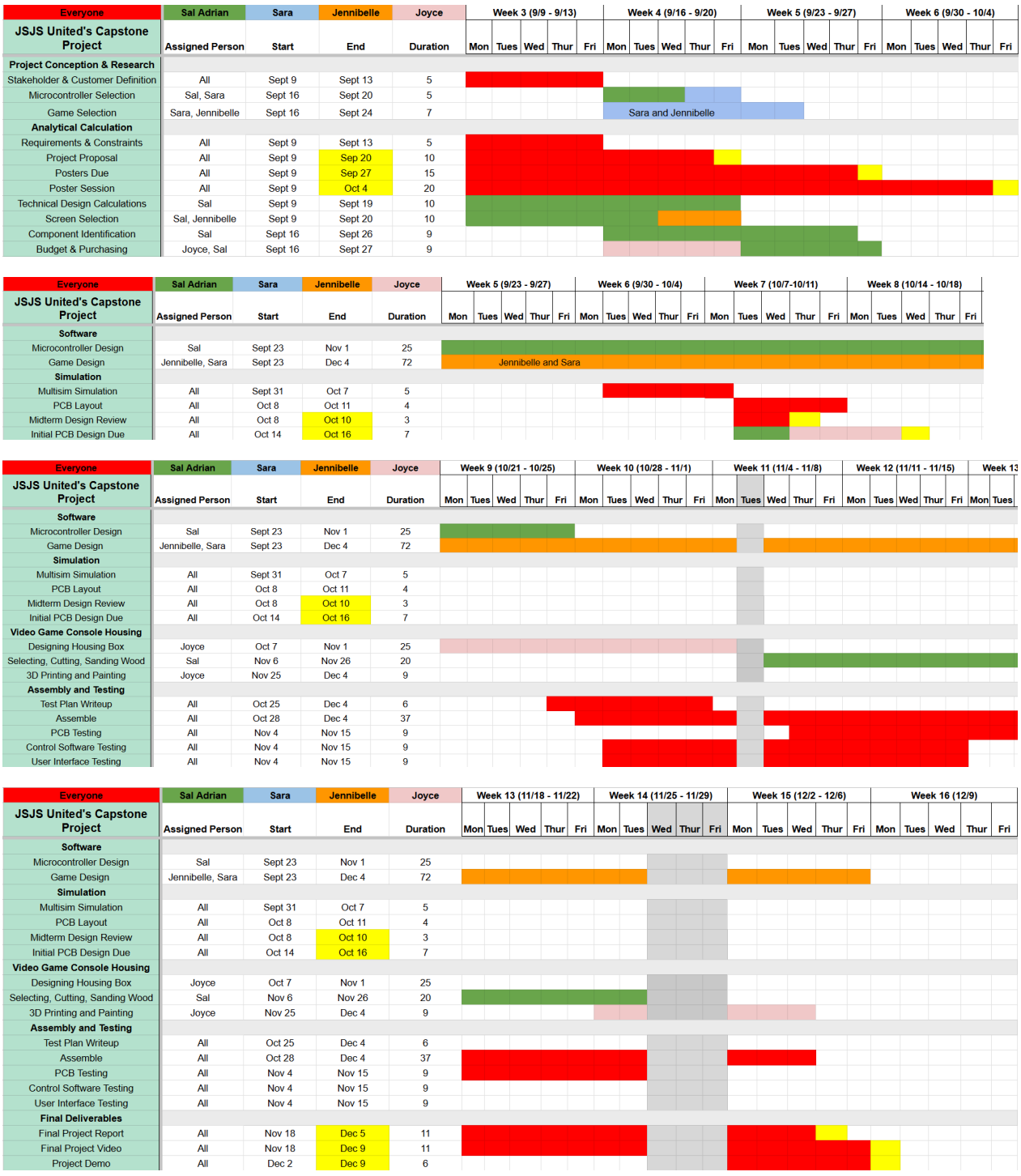
Everyone	Sal Adrian	Sara	Jennibelle	Joyce
JSJS United's Capstone Project	Assigned Person	Start	End	Duration
Project Conception & Research				
Stakeholder & Customer Definition	All	Sept 9	Sept 13	5
Microcontroller Selection	Sal, Sara	Sept 16	Sept 20	5
Game Selection	Sara, Jennibelle	Sept 16	Sept 24	7
Analytical Calculation				
Requirements & Constraints	All	Sept 9	Sept 13	5
Project Proposal	All	Sept 9	Sep 20	10
Posters Due	All	Sept 9	Sep 27	15
Poster Session	All	Sept 9	Oct 4	20
Technical Design Calculations	Sal	Sept 9	Sept 19	10
Screen Selection	Sal	Sept 9	Sept 20	10
Component Identification	Sal	Sept 16	Sept 26	9
Budget & Purchasing	Joyce	Sept 16	Sept 27	9

Everyone	Sal Adrian	Sara	Jennibelle	Joyce
JSJS United's Capstone Project	Assigned Person	Start	End	Duration
Software				
Microcontroller Design	Sara	Sept 23	Oct 18	25
Game Design	Jennibelle	Sept 23	Oct 18	25
Simulation				
Multisim Simulation	All	Sept 31	Oct 7	5
PCB Layout	All	Oct 8	Oct 11	4
Midterm Design Review	All	Oct 8	Oct 10	3
Initial PCB Design Due	All	Oct 14	Oct 16	7

Everyone	Sal Adrian	Sara	Jennibelle	Joyce
JSJS United's Capstone Project	Assigned Person	Start	End	Duration
Assembly and Testing				
Test Plan Writeup	All	Oct 25	Dec 4	6
Assemble	All	Oct 28	Nov 8	9
PCB Testing	All	Nov 4	Nov 15	9
Control Software Testing	All	Nov 4	Nov 15	9
User Interface Testing	All	Nov 4	Nov 15	9

Everyone	Sal Adrian	Sara	Jennibelle	Joyce
JSJS United's Capstone Project	Assigned Person	Start	End	Duration
Final Deliverables				
Final Project Report	All	Nov 18	Dec 5	11
Final Project Video	All	Nov 18	Dec 5	11
Project Demo	All	Dec 2	Dec 9	6

Figure 15: Final Gantt Chart



In our gantt chart we color-coordinated the cells to represent each group member's contributions and most progress occurred from Week 3 to Week 15. The yellow-highlighted cells mark key deadlines, such as the project proposal and final report, which we planned ahead to meet. The chart is divided into main sections, which are the Project Conception & Research, Analytical Calculations, Software, Simulation, Video Game Console Housing Development, Assembly & Testing, and Final Deliverables sections. The team was split into two groups where Sal and Joyce handled hardware, designing the microcontroller, configuring it, and developing the housing box, while Sara and Jennibelle focused on software, designing and coding the game.

We worked together to define the stakeholders and customer requirements, select the microcontroller, and choose the game. Sal managed hardware design calculations, selected the screen, and ensured proper power supply. Joyce and Sal handled budget and purchasing and Joyce also designed the PCB layout. Throughout the semester, we tested the integration of the joy con controllers, screen, and microcontroller with the game, ensuring correct functionality and user-friendliness.

Tasks that were done in parallel were the coding of the game, the raspberry pi configurations, and PCB designing. Tasks that were done sequentially were the selection of the game, then the selection of the controllers and screen, then analytical calculations to know how much power supply is needed to power the system, the PCB design, and then the assembly of the PCB, raspberry pi, controllers, and screen.

Throughout the semester, the timeline for the game design portion changed significantly, extending from an initial goal of completing it in 25 days by October 18 to working on it for a total of 72 days, with the final deadline on December 4. This extension was necessary due to the significant changes we made to the game's goals in the beginning of the semester. In the beginning, we knew we wanted to create a game for elementary students, but we were uncertain about which subject to focus on. Initially, we aimed to design a game that could cover all subjects for all elementary students. We were especially interested in creating a math game that would help students with basic operations such as addition, subtraction, multiplication, and division. However, after consulting with Eric Bredder, a UVA postdoctoral researcher who researches how technology can be effectively used to help teachers and students learn, we realized that our approach needed to be more focused on teamwork and learning, as opposed to being similar to preexisting educational games, such as *Kahoot!*, where the goal is simply getting questions correct at the fastest rate. Our goal was to create a game where students wouldn't feel discouraged or embarrassed by getting answers incorrect. We wanted to create a collaborative learning environment where students could work together, teach each other, and feel the desire to learn rather than focusing on getting the right answer quickly. We decided to create a reading comprehension game where students would answer questions together, with low stakes for getting answers wrong, but this change in direction led us to develop a storyline-based game, with an emphasis on reading comprehension, teamwork, and collaboration.

This new design concept required more brainstorming to create new levels, storylines, reading comprehension questions, and a hint system. This included doing research on existing third-grade reading SOL's to ensure our game met the educational standards for schools in Virginia. However, since we chose to use the PICO-8 game engine for its retro pixel style, offline functionality, and ability to run easily on platforms like the Raspberry Pi, we encountered challenges with memory constraints and displaying text on the screen. The 128x128 pixel constraint of PICO-8 made it difficult to display long question and answer text, which further delayed our progress. Therefore, the process of refining our reading comprehension concept, designing new levels, and PICO-8 memory and pixel display constraints required us to extend our game design timeline.

Additionally, a new subsection, "Video Game Console Housing," was added to the Gantt

chart to account for the steps involved in designing, selecting wood, cutting, sanding, 3D printing, and painting the console housing. These tasks extended the assembly timeline for the housing box from November 8th to December 4th.

Costs

The appendix shows a detailed spreadsheet of our cost. The total project came to $369.15. However, if we consider mass producing our project and purchasing our parts at a quantity of 10,000 the cost of the unit price varies. For the Raspberry Pi 4 Model B 8 GB of RAM the unit price stays the same so our bulk order order would be $750,000 on the Raspberry Pi ordering website. The cost of the USB-C bulk order would be $4,159.84 and a unit price of $0.88 using the digikey part number USB4085-GF-A. The cost of the barrel jack connector for 10,000 parts would be $3781.60 and a unit of ~$0.38 using the digikey part number PJ-102AH . For the screen bulk order on Amazon it would cost $999,990. Using the digikey part number WSU050-3000 for the wall plug-in comes to $87,929 and a unit price ~$8.79. Lastly using the parts number MFR-25FTF52-5K1for the resistors comes to a total of $81.60 and a unit price of $0.00816.

For all of the other reimbursements in our budget it is difficult to determine the cost of a bulk order since they were either bought in retail stores or from an Amazon order. However, if we assume that the prices stay the same it would be the cost of the purchase times 10,000. Not all of our components come from a distribution center like digikey so continuing to use the same outlets we used during our capstone would be an unsustainable business model. But if we were to find a way to find distributors and minimize our manufacturing costs by a large margin we could implement automation in our manufacturing. Since most of our assembly requires soldering, screwing, and cutting wood we can implement a series of manufacturing tactics.

For starters the soldering could be completed using automated soldering robots or waving soldering machines. The screwing of the box and the stacked PCB and microcontroller could potentially be accomplished by using screw tightening robots. Lastly for the wood processing and this could be accomplished by CNC router machines. All of these processes could have a tool to assist the automation; however, we would still need to consider the cost of conveyor belts and pick and place those robots. Overall, the automation for our design would be an incredibly high initial investment that would not be able to sustain itself economically. In order for our project to be able to be manufactured at a scale of this magnitude it would require a lot of redesigning and reduction of material costs.

Final Results

In this section you should explain the functionality of your final prototype in detail. You should honestly assess and explain which of the success criteria defined in your proposal you met and which you did not.

Hardware

Figure 16: Front of EMO with screen and power button

Figure 16 shows the front of the body with the screen, power button, usb plug-ins under the d-pad, and other cosmetic buttons. The screen shown is slipped into the laser cut fitting and then mounted with 4 screws for extra support. The laser cut is made such that none of the screen gets covered or removed. The USB-A plug-ins located in the left corner are for the controllers to be plugged into. Lastly, the power button can be located on the right side of the body and is for the Raspberry Pi to enter either a reset, shutdown, or on mode. For reset the user can click the button twice, for shutdown holding the button longer than 3 seconds and letting go will shutdown the Raspberry Pi, and clicking the button once while on will do nothing but when the Raspberry Pi is shutdown it will turn it back on.

Figure 17: Back of EMO with wiring

Figure 18: A user playing the PICO-8 game with the Joycon

Figure 17 shows the wiring within the body of EMO and figure 18 shows a user using the console to play the PICO-8 game stored in the Raspberry Pi. Lastly, the hardware was able

to successfully power the system using the wall plug in and provide a durable stable and safe console with no sharp corners. Upon assessing hardware specs and requirements the hardware team concluded that all functional requirements were met and other features were implemented furthermore to improve the user experience like the power button. The console launches the game on startup and leaves no way for students to access the internet. As well the hardware inside is protected and properly insulated with a fan to ensure prolonged usage and to withstand the young user demographic.

Video Game Console Housing

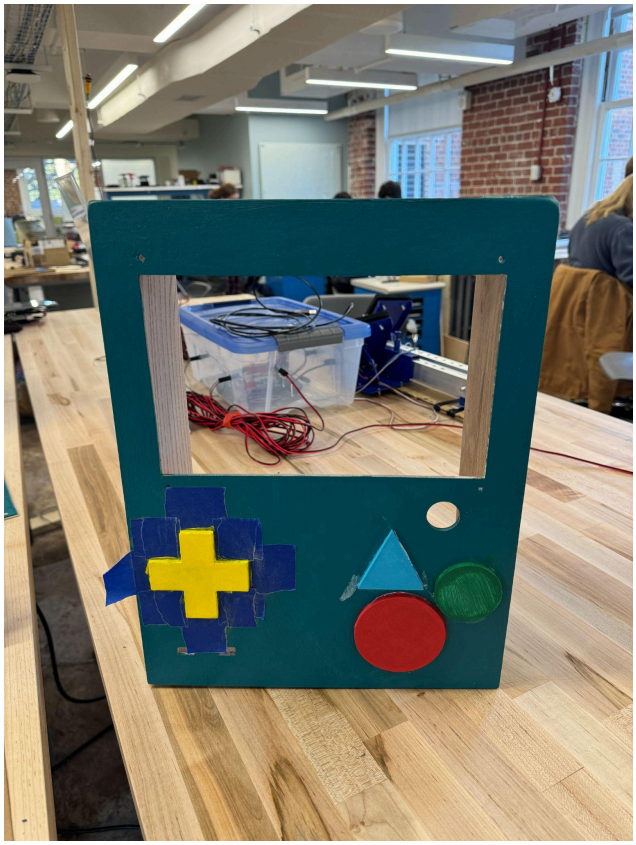


Figure 19: Front of Box

The entire box was painted teal to look like BMO, and the front of the box showcases multiple openings for the screen, power button, and USB ports. The colorful buttons at bottom are for decorative purposes only, and cannot be removed.

Figure 20: Back of Box

The back of the box includes ventilators for proper air flow as the Raspberry Pi does produce some level of heat. There is also a small hole at the bottom for the wall plugin to go through. The back of the box was attached to the sides with screws, whereas the front of the box was attached with wood glue. This was to ensure easy access to the circuitry inside.

Figure 21: 3D Printed Arm

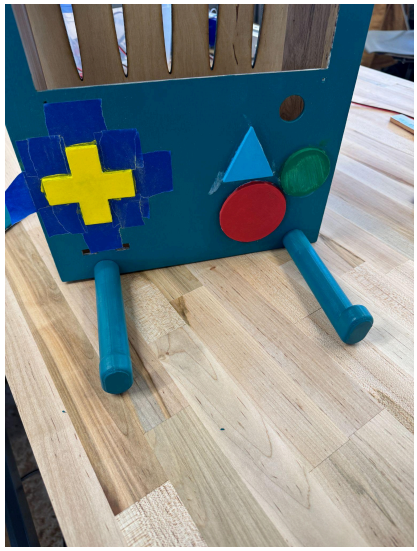


Figure 22: 3D Printed Leg

As shown above, the 3D printed arms and legs are quite tubular. The legs will be attached to the bottom of the box and the arms will be attached to the sides of the box with super glue where it will write out “EMO”.

Software

Figure 23: Start Screen Sequence

The start screen displays a sequence featuring an endearing smiley face as the game console boots up. The design of the face aims to give the console a personality, making it feel like its own character. This entertaining introduction features visuals that were inspired by BMO’s face from “Adventure Time.” The face sequence lasts about 3.5 seconds, with each frame displayed for approximately 1 second, before transitioning to the menu screen, as shown in the figure below.

Figure 24: Main Menu Page

There is also a working menu page where players can choose from three options, which are starting the game, viewing the instruction screen, or selecting a level.

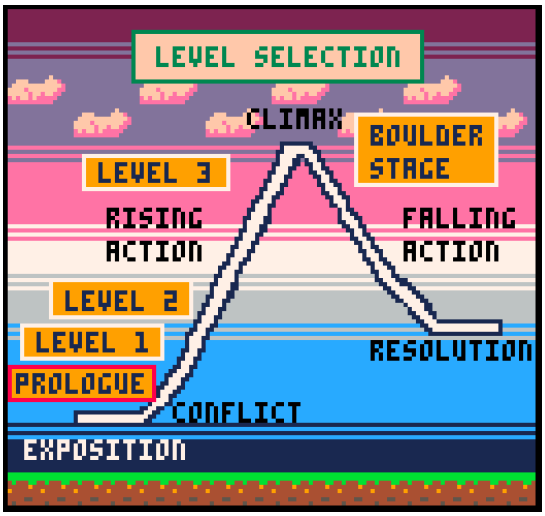


Figure 25: Level Selection Page

The level selection screen features a plot diagram that allows players to navigate through the levels and view the overall story in a plot diagram format, as shown in the figure above. This was designed to help students understand the structure of the plot diagram and how the game's story fits within it. Additionally, it enables players to pick up where they left off or replay their favorite levels or scenes without restarting the game. A red border indicates the selected level, and players can use the arrow buttons to move up and down, then press A to select their desired level using their controllers.

After clicking on starting the game, the prologue scene will be loaded in. The prologue scene shows why the two players have to climb a mountain together to escape a cheetah.

Figure 26: Prologue Stage (Pip learns of the Cheetah and tells Pete)

First, Pip, the yellow monkey, leaves the house and interacts with Miguel, the blue monkey. Miguel, one of the villagers in the town, informs Pip about the approaching cheetah that will attack their village. After learning, Pip walks over to Pete, the red monkey, and tells Pete about the approaching cheetah. Each character's house is color coded with the main character design color. After learning about the cheetah, Pete says they have to go talk to Percy about what to do.

Figure 27: Prologue Stage (Pip and Pete go to Percy's House)

When arriving in front of Percy's house, Pete knocks on Percy's door. However, there was no response and Pete exclaimed "Bro! Open the door! We have something important to tell you!!!". After no response, Pete knocks harder and finds out the door was unlocked. Pip questions this by saying "Weird… Percy never leaves his door unlocked." Then, they enter the house and find a singular note on the dining table with no Percy in the house. After reading the note, Pete exclaims "We have to go!" This concludes the prologue section of the story.

Figure 28: First Question Correct Scenario

At the beginning of level 1, we will ask 4 questions about the story of the prologue. When answering a question, the player can scroll up and down to select one of the answers. After clicking A, this confirms the selection of their answer to the question. If the answer choice is correct, the screen will display the words “Correct!” and will move on to the next question. Once all four questions are answered, the game will move onto the teamwork based levels featuring obstacle courses.

Figure 29: First Question Incorrect Scenario

In the situation that the question was answered incorrectly, it first displays the words “That was incorrect. You can answer it one more time! ^-^ Lets look at a hint.” This will be the exact same for all questions. However, afterwards there will be a hint that will be displayed that is unique to each question. If the question is answered incorrectly twice, the correct answer will be displayed for the user to have an understanding of the story. After this, the story will continue and move on.

Next, The game features three working levels with questions asked at the beginning of each level. Each level has two-player interactions designed to help one another. The two player characters, Pip and Pete each have their unique strengths and weaknesses. Pip, being smaller, is

able to navigate through small cracks to flip switches, while Pete, the larger and stronger monkey, can move boxes to overcome obstacles such as using the boxes to hold up a seesaw or jump over ditches.

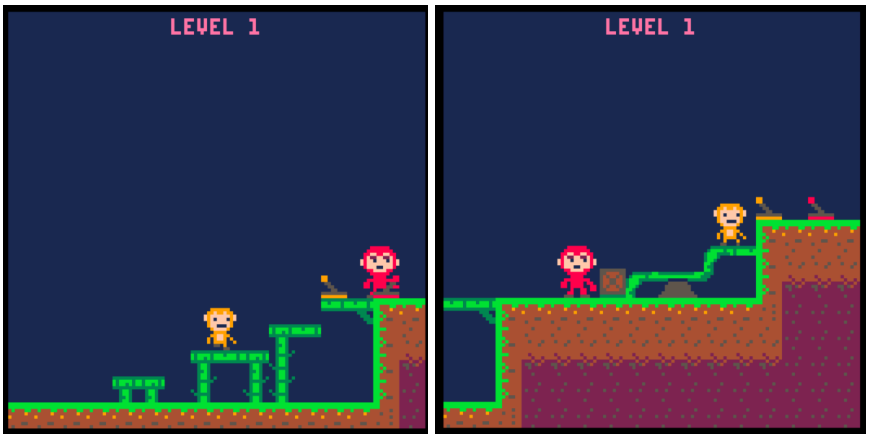


Figure 30: Level 1 (Parts 1 and 2)

Level 1 is the easiest of the three. The first part requires the players to jump on top of the bamboo platforms and flip their designated switches. These switches are color-coordinated to each player, and once both switches are flipped, the camera will pan to Part 2 of Level 1. Part 2 introduces a seesaw mechanic where if a player stands on the higher end of the seesaw, it will flip, sending the player down and raising the opposite end. To cross to the other side, Player 2 can either stand on the left end of the seesaw to let Player 1 pass, or push the box onto the left end of the seesaw. The second method will allow both players to stand on the right side of the seesaw without being pushed down, and reach their designated switches. This seesaw mechanic encourages communication and teamwork to coordinate their movements. Additionally, only Player 2 can push the box, emphasizing his strengths within the duo. This creates a teamwork dynamic within the players where Player 1 must rely on Player 2 to overcome obstacles and complete the level.

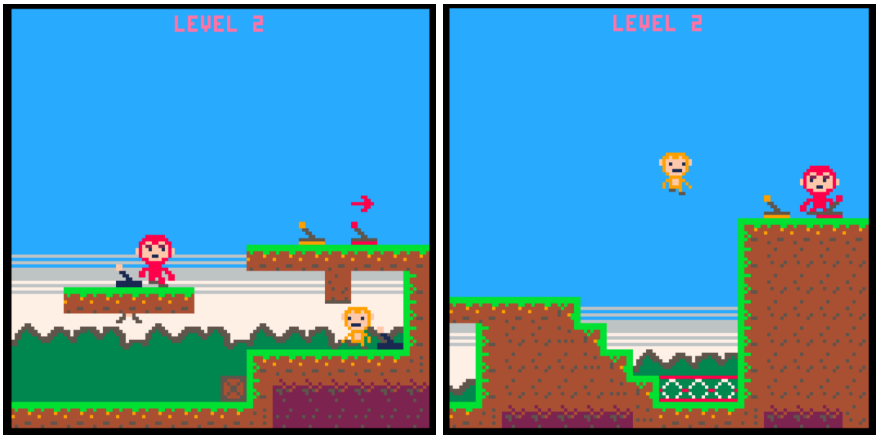


Figure 31: Level 2 (Parts 1 and 2)

Level 2 will highlight both Player 1 and Player 2's unique strengths. In Part 1 of Level 2, only Player 2 can move the box so they can jump on top of the cliff, and only Player 1 will be small enough to squeeze within the small enclosed area where the switch is located to bring down

the platform. The switch on the right that is next to Player 1 will bring the floating platform down and the switch on the left that is next to Player 2 will bring the floating platform up.

Part 2 of Level 2 will have a trampoline where if the players are standing on top of the trampoline and jump by pressing the “A” button on their controller, they will be able to jump higher so they can flip their designated switches. Part 2 offers a fun break from the rest of the more challenging levels to keep the game lighthearted and entertaining for the students.

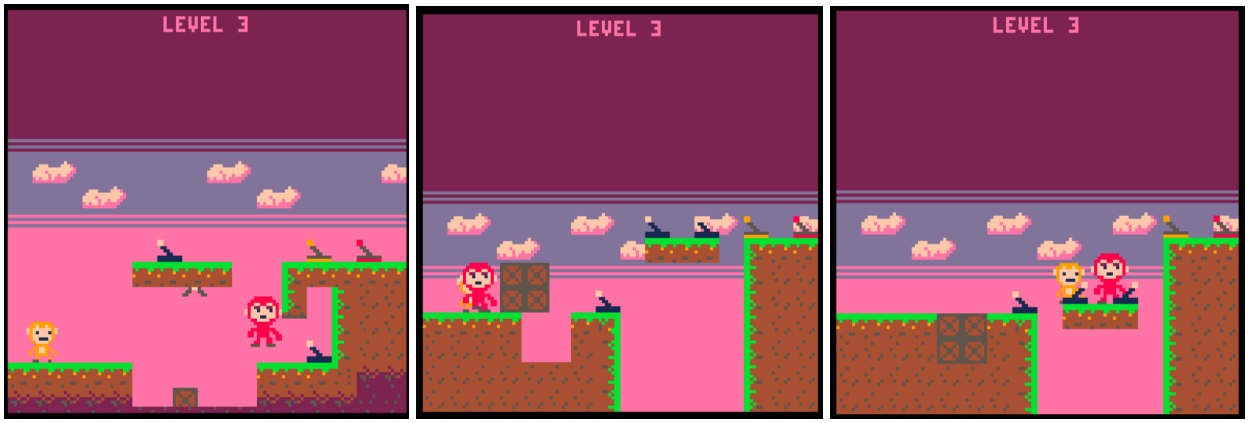


Figure 32: Level 3 (Parts 1 and 2)

Part 1 of Level 3 is very similar to Part 1 of Level 2, but in Level 3, Player 2 must throw the box into the ditch and jump on top of the box in order to make it to the other side. Part 2 of Level 3 focuses more on teamwork, where Player 1 and Player 2 must work together to push a large box at the same time to clear the path. Afterward, each of the players need to flip one of two switches simultaneously to raise the platform.

The game has a climax, falling action, and resolution stage with a storyline that guides players through the narrative.

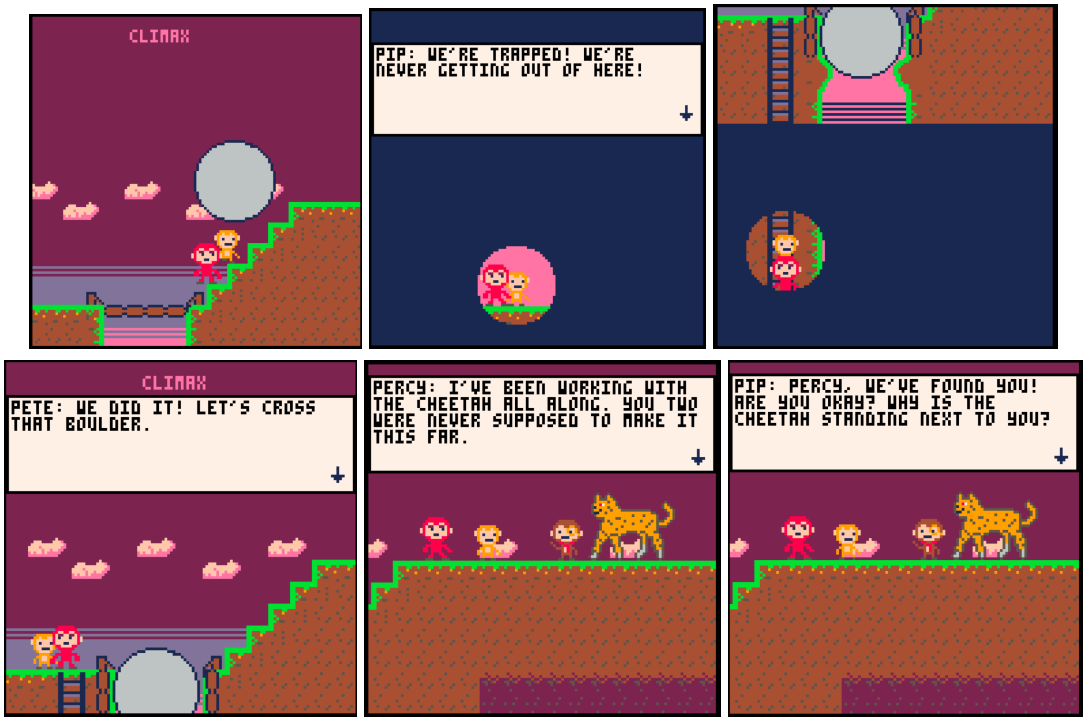


Figure 33: Climax Stage

The climax starts off with the two players crossing a bridge, but soon after they cross the bridge, a very large boulder falls down the mountain and breaks the bridge, pushing them down a very deep ditch. The players will only be able to see a circle surrounding Player 1. The two players will need to work together staying inside the circle to find a way out of the ditch and back up the mountain. They can do this by moving together left and right. To the left, there will be a ladder they can climb to reach the top of the mountain. After climbing, they'll be able to cross where the broken bridge used to be by stepping on top of the boulder. Once they reach the top of the mountain, they will finally find their missing brother, Percy, and the cheetah. This is when they begin to question why Percy and the cheetah are working together.

Figure 34: Falling Action and Resolution Stages

These images show three screenshots from the falling action and resolution of the story. In these scenes, Percy will explain why he chose to work with the cheetah, and the two brothers, Pip and Pete, will apologize to Percy for making him feel isolated, and Percy will apologize to them for not communicating to them how he has been feeling earlier. They learn the importance of effective communication when dealing with feelings of isolation and grudges in relationships. The three brothers then explain to the cheetah that they want to stick together from now on, and Percy expresses that he no longer needs assistance from the cheetah to feel wanted and to have revenge. The cheetah, frustrated and hesitant at first, eventually understands and lets them go. The cheetah realizes that the brothers' bond is too strong and there's nothing left he can do to manipulate them, so the cheetah leaves.

After the resolution stage, they are given 4 questions to ensure that they two players have followed the storyline narrative in a very similar format to the question and answer format shown in Figure 28 and 29. With the questions, they are given a question with 4 answer options of A through D. After selecting one of the answers, the players are given instant feedback if they are correct or not. If they are incorrect, they are displayed a hint to give them context of the specific scene of the question we were asking. After answering the question, if the answer is wrong once again, we will not add a punishment but give them the answer to the question. Because we ask questions regarding the timeline and information of the storyline, we ensure that the players relearn the scene they missed if they get the answer incorrect twice.

We met most of our goals because the game offers functioning, challenging, and engaging multiplayer obstacles, with unique roles for the two characters, requiring them to work together and collaborate to succeed and overcome each level. The game begins at the start screen, which leads to the menu page. Selecting the 'Start Game' button will show and start the

prologue stage, followed by progressions through levels 1, 2, and 3, leading into the climax stage. Additionally, there is a popup reading comprehension question at the beginning of levels 1 and 2. However, we were unable to track teacher output, such as how long each player played, who they played with, and how well they performed in the reading comprehension tests.

Engineering Insights

Hardware

We developed many embedded skills working on a Raspberry Pi microcontroller. We already had experience working with STM32 microcontrollers, but the language and UI in the STM32 microcontrollers weren't the same as the Raspberry Pi microcontroller. We got the chance to improve our skills in referring datasheets and finalizing decisions with references to documentation, logic, and numerical values. Furthermore, since we had to make custom wiring, our test plan provided a great opportunity to apply circuit analysis and critical thinking on measuring values from our custom builds and manufactured parts with no datasheets. For manufactured parts with no datasheet, such as the ROADOM screen, planning was delayed as we had to wait to physically measure values in order to continue design. Thus we learned that having a datasheet can ensure confident decision making and promote parallel work among tasks.

What worked really well with designing the power system was thoroughly reviewing datasheets and confirming assumptions with analytical values. Before manufacturing the PCB, we assumed that all cables would draw max current and ensured there was enough power in the supply to support all devices. Once the board was manufactured, the connection to the other hardware was straightforward since we had already properly accounted for all device needs.

The important lesson that was learned from hardware is that even if a well thought action plan was made, there should always be space for errors. Many times throughout the semester, setbacks were caused by either delays in woodworking or faulty connections and wiring. Our plan worked from its conception to the end; however, the steps to implement were far from easy. Debugging and making time to research solutions and parts can cause a lot of stress if there is no room in the schedule for it. Our advice is to remember that you're a student, and while you may identify issues and plan solutions, the time required for implementation is always an unknown variable, so be patient and gracious. Failing to do so can cause conflicts in other classes and strain yourself and the team for time.

Video Game Console Housing

Using tools such as bandsaws and planners, we learned how to cut and sand with wood. We also learned how to laser cut wood using a software called Rhino. This required us to upload files of our laser cut design and execute them onto the laser cutter. Working with glue and paint, we also learned how much time the application process took. Additionally, we learned how to use SolidWorks and operate a 3D printer to get our console's legs and arms as seen in Figure 21 and Figure 22.

In terms of advice to give to a future Capstone student, we would say to expect things to take twice as long as you would initially hope for, and to plan accordingly. For example, we were expecting to finish before Thanksgiving break, however, we did not end up finishing until the Saturday or Sunday before the day of demoing our project. We would also encourage to have good relationships with your group mates, as that does really make or break a project as you constantly have to debate on which design choices to go with.

Software

We developed several new technical skills, including improving our ability to refactor code to make it more concise and organized. This was done by grouping large sections of code that serve a similar purpose into their own functions, such as the seesaw or platform mechanics were split into their own functions. This helps ensure that others who view our code can easily understand it, especially given the memory limitations we faced. We also learned the importance of commenting code so that we can revisit and understand what we worked on in the past, which makes debugging easier in the future. As two people were actively working on the software, having comments helped us understand one another's code. Additionally, we gained a deeper understanding of how video games function, particularly the game loop. The game loop consists of two main parts which are update and display. The update phase processes the player's input, calculates new velocities and positions, and handles collisions. The display phase draws the updated frame to the screen. We also learned valuable debugging techniques, such as ensuring the proper order of layers when displaying items, calculating positions of game objects when offsetting maps across different levels, and using timers to display images in a sequence, for example, our start screen with different faces. We also committed our code to the remote repository frequently to document our progress, worked in our own branches, and followed best practices for merging changes to the main branch to avoid conflicts. This included always making sure to pull requests before adding and committing changes.

In terms of teamwork, we found that the weekly meetings were really helpful in keeping us accountable for our deliverables. They allowed us to share our progress, so we had a good understanding of what everyone was working on each week. This also gave team members the opportunity to offer suggestions or help with each other's tasks. We also realized that our team communicated very well because we felt comfortable offering constructive criticism without attacking each other personally. Instead of criticizing a team member's effort, we focused on suggesting improvements in a helpful and positive tone, which helped avoid putting anyone in a defensive state. Additionally, we made an effort to get to know each other personally, not just academically. This made us more comfortable asking for help and working together for extended periods. Another important lesson we learned was the value of seeking help when needed. When we felt overwhelmed or stuck on a problem, getting input from others with different perspectives was incredibly helpful in overcoming challenges.

An advice we would give is when researching what game engine to use, consider factors like memory capacity, sprite rendering, screen size, font input, and sprite import options. For example, PICO-8's 64 KiB memory limit, 128x128 pixel display, 16 color palette, and sprite

constraints made it difficult to design stages and manage complex objects since each moving object needed to be manually coded for position, speed, gravity, and collision detection. We also faced limitations on sprite usage, which slowed our progress in creating each scene because we had to manually make every single sprite through a very limited 8 by 8, 16 by 16, or 32 by 32 pixel art format. We suggest comparing the experiences from other people who have used PICO-8 and other engines to better understand their constraints and decide whether it would be the right fit for your project or not. While we managed with PICO-8, using preexisting sprites, we would have preferred making sprites on tools like Figma, a design tool that lets you test graphic design ideas, for more color options and pixels. PICO-8's limited audio capabilities also made it clear that an engine with better support for audio file imports would have been beneficial. Looking back, we realized that understanding an engine's RAM usage, token limits, and external sprite import capabilities earlier would have saved time. We also found that engines with easier screen resizing, better font options, and the ability to import audio source files would be more efficient. Having real-time visual feedback on sprite placement would save time compared to manual calculations. Additionally, we would prioritize a game engine with support for larger maps and features like level fading. Next time, we would prefer using a language like C for better memory management and more efficient code. Unlike Lua, C supports more streamlined game mechanics, and game engines like Unity that provides pre built functions for tasks like collision detection, reducing the need for extensive manual coding.

Future Work

Hardware

The first area that could be improved is the microcontroller RAM selection. For the Raspberry Pi Model B we chose 8GB of RAM but our system could have worked with 1 GB of RAM which would have brought the microcontroller selection from $75.00 to $35.00. Next would be accounting for the wire connections from the packaging better. The cutout for the space and the length between that cutout and the sides prevented the micro-usb cable from connecting to the side of the screen to the PCB as seen in Figure 17 (Back of EMO with wiring). An improvement would have been to measure the true width with all of the cables as opposed to just the screen.

Designing a way to have the fan be over the CPU would be a lot more effective than being on the side. Designing a 3D print hold that could be screwed in the through holes of the Raspberry Pi such that there is a way to hang the fan over the CPU would provide better cooling. Lastly, designing hooks on the inside so that the cables can be neatly hanged instead of loosely hanging with zip ties. The hanging of the cables can present strain on their jacks and connectors after prolonged use in a real world production scenario.

Video Game Console Housing

Initially, we wanted to contact a plastic manufacturer to potentially make our box from start to finish, however, we decided it would be best for experience sake to make it ourselves.

Because of this, we had to learn how to work with wood, glue, and paint, making the overall product look a little too “homemade”. For future iterations, we think it would be best to have some kind of plastic manufacturer make the box with plastic just so that it looks seamless and ready to be used by any child without cautions for safety. In addition, we think it would be best to use screws for the arms, legs, and buttons, instead of super glue, so that it cannot be ripped apart by a child.

Software:

We initially wanted a data output for the teachers to see exactly which students were playing with each other, exactly how long they were playing with each other, and their performance on the reading comprehension tests. We were unable to implement this feature due to the constraint of not being able to use a single PICO-8 file throughout the project and having to break the code into different files for each level and stage, so we couldn't track user data consistently throughout the entire game. This feature would have been useful for teachers to monitor student progress. The memory limitation challenge in PICO-8 was unexpected, so we recommend using a game engine like Unity that can handle more memory or RAM, allowing the entire game to be coded in one file. Unity would also provide more options for expanding the game with additional colors, music, and larger screen sizes. Additionally, Unity supports multiple fonts and resizing, which would make the texts in the reading comprehension portion easier to read. While PICO-8 gave our game a charming retro style and helped us learn about concising coding and loading multiple cartridges into one main game file, for real-world expansion and implementation, we would suggest using Unity since it can accommodate the memory requirements for larger games since most Unity games typically use 1-4GB of RAM.

References

References

Akio I. (2013). *Video game system with wireless modular handheld controller* (U.S. Patent No. 8430753B2). U.S. Patent and Trademark Office. https://patents.google.com/patent/US8430753B2/en

Cerezo-Pizarro, Mario, Francisco-Ignacio R., Jorge G., and Jairo M. (2023). *"The Cultural Impact of Video Games: A Systematic Review of the Literature." Education Sciences, vol. 13, no. 11, 2023, p. 1116.* https://doi.org/10.3390/educsci13111116.

Chiu, M. (2023, November 13). *Knock the rust off your social skills after pandemic setbacks.* Baylor College of Medicine. https://www.bcm.edu/news/knock-the-rust-off-your-social-skills-after-pandemic-setbacks

Delgado, A. J., Wardlow, L., McKnight, K., & O'Malley, K. (2015). *Educational technology: A review of the integration, resources, and effectiveness of technology in K-12 classrooms.* http://www.jite.org/documents/Vol14/JITEv14ResearchP397-416Delgado1829.pdf

Hayre J. (2023). *COVID-19, education and child health. BMJ paediatrics open, 7(1), e001863*. https://doi.org/10.1136/bmjpo-2023-001863

Healey, Aleeya, et al. (2019). *"Selecting Appropriate Toys for Young Children in the Digital Era." Pediatrics, vol. 143, no. 1*,https://publications.aap.org/pediatrics/article/143/1/e20183348/37330/Selecting-Appropriate-Toys-for-Young-Children-in

IPC. (n.d.). *IPC standards: Association connecting electronics industries*. IPC. https://www.ipc.org/ipc-standards

Iwao K. (2024). *Game Program, Game System, Game Device, and Game Processing Method* (Japanese

Patent No. 7545191B1). Japanese Patent and Trademark Office. https://patents.google.com/patent/JP7545191B1/en?oq=7545191

Mechelen, Maarten Van, et al. *"Emerging Technologies in K–12 Education: A Future HCI Research Agenda."* ACM Transactions on Computer-Human Interaction (TOCHI), vol. 30, no. 3, June 2023, pp. 1–40. EBSCOhost, https://doi.org/10.1145/3569897.

National Fire Protection Association. (n.d.). *NFPA 70: Standard for electrical safety in the workplace*. National Fire Protection Association. Retrieved September 19, 2024, from https://www.nfpa.org/codes-and-standards/nfpa-70-standard-development/70

Newegg. (n.d.). *2 pack USB wired controller for NES games, PC USB controller retro gamepad joystick Raspberry Pi gamepad controller for Windows PC Mac Linux RetroPie NES emulators.* https://www.newegg.com/p/2S7-09K5-00222?Item=9SIBUGFK9K7171&srsltid=AfmBOoo9zGdBXeiIUeSx16vIVMEf42AEj4IW9yD8XNXVkWihlzZ_ge8Q

Raspberry Pi Foundation. (2024). *Raspberry Pi documentation: Computers.* https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-5157

Satoru O., Shin K. (1993). *Compact hand-held video game system* (U.S. Patent No. 5,184,830A). U.S. Patent and Trademark Office. https://patents.google.com/patent/US5184830A/en

U.S. Consumer Product Safety Commission. (n.d.). *ASTM F 963 chart: Standard consumer safety specification for toy safety.* U.S. Consumer Product Safety Commission. Retrieved September 19, 2024, from https://www.cpsc.gov/Business--Manufacturing/Business-Education/Toy-Safety/ASTM-F-963-Chart

Virginia Department of Education (2022). *Instruction in Virginia's public schools is guided by the* Standards of Learning (SOL). Virginia Department of Education.https://www.doe.virginia.gov/teaching-learning-assessment/instruction

Appendix

In this section you should include helpful information that does not fit into the above categories but will be helpful in understanding and assessing your work. Your budget outline should also go here.

		Electronics & electrical components			
Index	**Manufacturer Part #**	**Digikey Part #**	**Mouser Part #**	**Newark Part #**	**RS Part #**
1	SC0195(9)	2648-SC0195(9)-ND			
2					
3	USB4085-GF-A	2073-USB4085-GF-ADKR-ND			
4	PJ-102AH	CP-102AH-ND			
5	WSU050-3000	237-1387-ND			
6	MFR-25FTF52-5K1	13-MFR-25FTF52-5K1CT-ND			
7					
8					
9					
10					

		Hardware and material		Last resort supplier
Index	**Manufacturer Part #**	**McMaster Part #**	**Grainger Part #**	**Amazon Part #**
1	SC0195(9)			
2				B09XDK2FRR
3	USB4085-GF-A			
4	PJ-102AH			
5	WSU050-3000			
6	MFR-25FTF52-5K1			
7				
8				
9				
10				

Index	**Manufacturer Part #**	**Qty in Stock**	**Qty Req'd**	**Per Unit Price**	**Cost**	**Team Name**	**Comments**
1	SC0195(9)	4750	1	$75.00	$75.00	JSJS United	Raspberry Pi 4 Model B 8GB of Ram
2		30	1	$99.99	$99.99	JSJS United	ROADOM Raspberry Pi Screen, 10.1'' Touchscreen Monitor; currently 20% off with Prime Deal, but list price is $99.99; the item is sold in limited quantities and the maximum quantity possible is 30
3	USB4085-GF-A	36185	2	$1.18	$2.36	JSJS United	USB-C Receptacle
4	PJ-102AH	63750	2	$1.02	$2.04	JSJS United	CONN PWR Jack
5	WSU050-3000	289	2	$10.59	$21.18	JSJS United	AC/DC Wall Mount
6	MFR-25FTF52-5K1	590	5	$0.10	$0.50	JSJS United	Resistors
7					0		
8					0		
9					0		
10					0		

Start Balance	Cost of Parts Order		Current Balance	Reimbursement Item
$500.00		$188.28	$311.72	
			$6.31	Fan
			$10.52	Power Buttons
			$8.41	USB-C Open End
			$18.86	Stand Offs and USB Extend Cable
			$11.56	USB-C Cable and USB-C to MicroUSB
			$30.31	Plywood
			$10.59	PCB
			$16.84	Joycons
			$17.34	Art Supplies
			$12.09	Art Supplies
			$26.27	Art Supplies
	Remaining after Reimbursemnts ->		$130.85	