OpenArcade, Group 3
Anish Paramsothy, paramsa
Chris Palermo, palerc1
Mitchel Cox, coxm12
Shivom Sharma, shars119
Jacqueline Leung, leungw18

McMaster University

Project Goals and Development Process

MECHTRON 4TB6, McMaster University

Date Submitted: September 27th, 2025

Due Date: September 28th, 2025

Contents

1 Glossary

Below is a list of definitions to ensure there is no confusion when reading the document:

- Module: A unit that contains electronics (boards and wiring), that will communicate inputs to another device.
- Parent Module: Describes the central hub that children modules will communicate to. This parent module will connect to the computer/console to communicate to the game.
- Child Module: Describes a unit that may contain joysticks, buttons, d-pads, etc. The children modules can be attached to each other to form a controller, or remain separate. These will communicate the inputs to the parent module and then to the game.
- Parent Board: Specified board used in the parent module.
- Child Board: Specified board used in the children modules.

2 Project Goals

2.1 Project Description

Design and development of an arcade/box style controller series of children modules (with a children modules containing: buttons, joysticks, d-pad, etc), that can be mechanically connected to each other to allow for gamers to develop their own combination and style of controller. Along with the idea that gamers can combine children modules together, we want to give them the option to play using the children modules separately. The children modules will be connected to a parent module (using wires or bluetooth) which will communicate the inputs of the children modules to the game.

2.2 Rationale

Currently, there are several game controllers that follow the ideology of "one size fits all", preventing gamers that may be injured, disabled, or lacking fine motor skills to play the games they want. This could be the case due to a variety of reasons:

- The controller itself is too small and is required to be held.
- Several controllers require a grip that becomes uncomfortable to the player after hours of gaming.
- Buttons and joysticks are close together and require the user to place their hands in positions that can potentially be uncomfortable.
- There is little room for customization.

2.3 Goals

We have several goals that we want to outline for the controller. We want to accomplish these to some degree of effectiveness.

2.3.1 Level 1 Goals

These goals are the most important for the functionality of the controller.

- Modularized: The controller should include multiple children modules, and for these to work together to play games.
- Customizable: The controller should be able to be attached to each other (if A and B are children modules, then they are to be attached in either A-B or B-A formats), or be able to be utilized

as separate and detached children modules. The joysticks should also be swappable and include additional customization options.

- Input Delay: Producing the lowest amount of input delay as possible, to prevent any scenarios where games become unplayable because the inputs are not responsive enough.
- Comfort: The controller should be attempting to be an ergonomic substitute to other controllers, so providing better options for buttons and joysticks and their placement should be a heavy consideration. Along with this, the location of where the controller will be (either in lap or on table) will also provide an improved gaming experience for the user.
- Functional: The children modules should be able to fit correctly together without any mechanical issues. Along with this, the controller should function correctly in terms of gameplay, where specific inputs (button presses or joystick movement) relate to the correct outputs.
- Robust: The controller should meet the basic needs of staying together and not falling apart when subject to stress. This could include the force created when pressing a button, or potentially the user resting their arm on the device while playing.
- Intuitive: The controller should not be confusing to use, the children modules should attach in a simple manner to prevent confusion.
- Connectivity: The controller should be able to connect to different types of devices, with the main priority being computer.

2.3.2 Level 2 Goals

These goals are added optional goals that we want to strive towards to optimize the product.

- Aesthetically pleasing: The controller should look nice in order to draw in consumers.

3 Development Process

3.1 Meetings

Meetings are planned to be held weekly during 3 core time slots:

- Tuesdays from 11:00AM to 12:00PM
- Thursdays from 4:00PM to 6:00PM
- Fridays from 12:00PM to 1:00PM

These time slots are meant for planning individual tasks and goals for the following meetings. The time slots are to be considered an open time to meet, but not a requirement. Group discussion outside of these meetings will be conducted to determine the next appropriate meeting date.

The group will also strive to meet at different times as well, such as the weekend, or whenever everyone is available.

Group members are meant to come to meetings with some individual work done, and all work completed is to be documented in meetings notes on Google Drive.

3.2 Overall Process Workflow

Below is a list of steps that outline the workflow in which the OpenArcade controller will be designed and fabricated.

We have several requirements/acceptance criteria in each step of the workflow, and the task will be defined as completed once the criteria for that step is completed.

> *Note that these steps in the workflow are subject to change depending on a variety of factors. This can include:*
>
> - *The step is missing any minor details related to the completion of it.*
> - *The step is to moved and completed at another point in the workflow that may make more sense.*
> - *The step is unneeded because it was completed passively in earlier steps.*

1. Discussion of Hazard Analysis: Meet in a few group meetings to identify hazards related to all aspects of design, including hazards during development, and hazards during use. This will all be noted in a hazard analysis document. To complete this task, all possible hazards should be identified.
2. Development of System Requirements: Outlining the hardware/software/mechanical required to complete the various milestones of the project. This document will be modified during the entire timeline of the project, as the group will define what is needed for the *proof of concept*, *rev 0* and *rev 1* milestones. The document will include a list of software and hardware used, along with the material that will be used for the housing. This will aid in drafting a bill of materials (BOM).
3. Concept Design 1: Drafting an initial concept design to show as the proof of concept. This should include showing the functionality of buttons on a simple scale. The goal is to develop a written plan of what the proof of concept should be, and finalizing what hardware will be required for that.
4. Proof of Concept: The proof of concept will be used to outline basic functionality of the design. This will include:
    - Ability to communicate between parent board and computer.

- Functionality of children boards (basic electrical connection to buttons, joysticks, etc) to a breadboard with LEDs
- BOM including hardware choices

The proof of concept will be successful if the button and joystick inputs correspond to the correct outputs on the breadboard. Along with this, another successful test will be if the parent board correctly communicates to the computer.

5. Concept Design 2: Begin planning the finalized design. The group will work in determining how the children modules (with buttons, joysticks, etc) will communicate with the parent module, so that direct inputs can result in outputs seen on the computer. A rough mechanical design will be made with cardboard or other simple materials so that the placement of the buttons and joysticks can be determined. Essentially a primarily housing for the controller. The children modules will not be attachable yet.
6. Rev 0: Showcase the buttons and joysticks communicating to the game itself. The demo will show how the buttons can be configured to specific outputs, and that they communicate the correct outputs for the corresponding inputs. Children modules will communicate to the parent module, which will be communicated to the game. Begin development of button configuration app.
7. Concept Design 3: Discussion and planning of the finalized design. This will include the mechanical housing for all modules, along with how they will be attached to each other to form the full controller. The housing should contain the electrical and hardware components, while also preventing the movement/shifting of the internal pieces. Children modules should be attachable to each other. Emitted heat from the controller should also be considered when developing the housing.
8. Rev 1: The finalized design of the controller. The demo will include two children modules that can be attached to each other, and can successfully communicate with the game through the parent module with the correct outputs. The controller will be compared to other controllers to showcase why it is an effective substitute for the controllers in the market. Finalize development of button configuration app.

3.3 Coding Standards

3.3.1 General Practices

We will aim to write modular code with unit and integration tests, following a functional coding approach. We will avoid writing redundant code and avoid regressions where possible. Our workflow will be Agile workflow, delineating tasks flexibly and pivoting when necessary. All code will exist on a single repository, allowing us to maintain simplicity for writing, testing, building and deploying any new code. We will aim to document our code with comments where necessary, using readable function names while producing larger pieces of documentation for interactions between multiple pieces of software or hardware interactions.

3.3.2 Version Control

We will be using GitHub as our main version control platform. Changes will be made as "features", we will constrain changes to be single additions where possible, keeping different components decoupled. Each feature will be a branch and changes will be reviewed in pull requests. When the relevant reviewers for the code have approved, we will squash and rebase, then merge the changes into the main branch as the source of truth.

3.3.3 Testing Plan

Our testing methodology will go hand in hand with our feature creation. Each change should be accompanied by unit tests for the smallest unit of code that is testable in our case a function. We will try to avoid redundant tests, and each of these tests will be run in a GitHub Actions Workflow. The workflow will automate testing before changes can be merged, this will occur by spinning up a Docker container that runs the tests in a virtual environment to check that there are no regressions from added changes based on our expected behaviour. If our expectations were wrong we will aim to correct the tests. We will be using the following libraries for testing:

Language	Testing/Build Tools
Python	pytest, strawberry, codeQL
TypeScript/JavaScript	Jest, Bun, npm
C/C++	gtest, Make/Bazel

Table 1: Testing Tools

3.3.4 Code Formatting

To ensure that changes are strictly functional we will run consistent code formatters and linters, as we establish the standards the settings of each may change. A code formatter will align the written code based on the configured settings, this eliminates redundant changes from showing in any Git Diffs e.g. whitespace additions.

Language	Formatter
Python	Ruff
TypeScript/JavaScript	Prettier/ESLint
C/C++	clang-format
LaTeX	latexindent

Table 2: Formatting Tools

3.4 Group Roles

Below is a table outlining the rough group responsibilities. These are subject to change over product development.

Some clarifying points:

- Software design refers to the code required for the children and parent boards to communicate, while also outputting the correct meaningful output to the specified input (otherwise defined as firmware). Software design will also include development of the configurator app.
- Electrical design refers to the hardware set up. This will include how the components of the children modules (buttons, joysticks, etc.) physically connect and transfer information to the children boards, how the children boards physically transfer the information of these inputs to the parent board, and how the parent board transfers this information to the computer.
- Mechanical design refers to the housing of the modules. This will include how the modules will be physically attachable to each other, how the electronics will securely be fit and fastened into the housing, and how the wiring will be managed.

Group Members	Roles
Anish	• Documentation • Mechanical design/implementation • Electrical design/implementation
Chris	• Electrical design/implementation • Software design/implementation
Jacqueline	• Software design/implementation • Electrical design/implementation
Mitchel	• Mechanical design/implementation • Electrical design/implementation
Shivom	• Software Design/implementation • Electrical design/implementation

Table 3: Group roles

3.5 Technology

Below is a list of all technology that is planned to be used during the project. We want to note that these are subject to change.

Software	Uses
GitHub	For any code and version control.
Google Drive	Any collaborative documents and shared files. **Used for CAD/mechanical version control.**
Autodesk Inventor	CAD for the housing of the parent and children modules, and for any custom components.
STM32CubeIDE, VSCode, Vim	IDE for code and microcontrollers.
C/C++	Language for firmware.
TypeScript/JavaScript	UI language for configurator application.
Python	Miscellaneous scripting language.
KiCad, EasyEDA, Altium	Software for PCB design.
PrusaSlicer, Bambu Studio	Software for 3D printing.
Docker	Containerizing code to run on multiple machines (MacOS, Linux, Windows).
Heroku/Render	Website hosting platform for configurator application.
Figma, Mermaid, MSPaint, Procreate	Softwares used for systems diagramming.
LaTeX	Tool used for documentation.

Table 4: Software and their uses

Hardware	Uses
3D Printers (Bambu Lab A1, Thode Makerspace: Prusa MK4S)	To print the module housings (both parent and child) and any other custom components.
STM32F Series Boards	Controller for the parent module.
ESP32s Series Boards, Raspberry Pi Pico	Child module control board (Bluetooth connection or wired to central unit).
Arcade Buttons/Joysticks	Hardware for inputs from children modules.
Miscellaneous Electrical Components (Capacitors, Resistors, Power Supply, etc.)	Any electrical components required to build the children modules.
Breadboards	Used in concept design and design verification.
Breakout Boards	Used for making custom cables for USB, micro-USB .
Bluetooth Card (Wi-Fi 5 enabled)	For communication between children and parent boards.
Oscilloscope/Multimeter	Troubleshooting and testing.
Cables (USB, Power cable)	Used for power and data transfer.

Table 5: Hardware and their uses