Handheld Emulation Station

Design Document

Group: sdmay19-25

Client/Faculty Advisor: Dr. Julie Rursch

Team Members:

Nick Lang

Sean Hinchee

Matthew Kirpes

Nic Losby

Jacob Nachman

Team Email: sdmay19-25@iastate.edu

Team Website: https://sdmay19-25.sd.ece.iastate.edu

Version 2.0

Updated: 12/2/2018

Table of Contents

1. Introduction

1.1 Acknowledgement

We would like to thank the GreatScott! YouTube channel for releasing very beginner friendly content that allows people to take the leap into the world of hardware design.

1.2 Problem and Project Statement

The problem to be addressed is that people do not have a dedicated platform from where they can play old school games in their pockets. We ourselves love to play old school games, but do not like wasting our phones battery life to do it. So we want to design a handheld emulation station that can fit in our pocket, with ideal battery life.

Our plan is to design a small scale embedded system that will run retro games for all users to enjoy. The hardware of the system will include a custom PCB which will connect a Raspberry Pi Zero, Battery, and Screen together, along with a circuit [2] that allows for charging the device.

Our software will consist of a custom kernel module as we will need it for USB bussing purposes. We also plan to design an Atari 5200 emulator to run on our software as well as make it compatible with RetroPi.

1.3 Project Scope

The hardware aspects for our emulation station will include our PCB board which we will use to connect a 3000 mAH, a 3.5 inch LCD display, and a Raspberry Pi 0. It will include a circuit [2] for charging the battery, and outputs for external digital displays. We will also design a case for our system as well as a dock the user can use to hook it up to an external device, such as TV or monitor.

The software aspects for our emulation station will include our custom kernel module which will be used so we can use USB in our desired fashion, as well as use it to support bluetooth connectivity with various controllers (Xbox, PS4, etc). We will also design our own Atari 5200 emulator in the Go programming language. We consider classic games to be any system older and including the Nintendo 64. In order to add this support, we will also be designing our system to work with the RetroPi project, which is a project that contains various emulators for different systems.

1.4 Operational Environment

Overall, we need to make sure that our emulation station will be able handle the cold and the heat. Users will have it in their pocket so we want to make sure it can handle all basic weather conditions. Water resistance is ideal so the user does not have to worry about their device getting ruined if they are out with it in the rain as well.

1.5 Intended Users and Uses

The intended users are people who want a handheld and portable platform in order to play classic games.

Uses

- Handheld and portable gaming
- Wireless when not charging
- Docking port for charging and to share display to another screen via HDMI

1.6 Assumptions and Limitations

Assumptions

- The games being emulated must be able to run on a raspberry pi zero
- One user at a time
- The games emulated must be owned due to copyright laws

Limitations

- The end product will not weigh more than 12 ounces
- The cost of the end product will not be more than one hundred fifty dollars
- An end product must be created before May 2019
- Battery Life(~2 hours)

1.7 Expected End Product and Deliverables

The end product will be a handheld and and portable gaming platform for classic games that can be run on a Raspberry Pi Zero. This will have a screen, battery, and tactile buttons. There will also be a docking port for charging and the ability to share display to another screen via HDMI. There will also be a user's manual along with the emulation system to describe its uses. This will all be delivered before May 2019.

2. Specifications and Analysis

2.1 Proposed Design

2.1.1 Hardware Design

Most of our Hardware Design revolves around our PCB. The PCB we are using will be the bridge for all of the components which include buttons, the Raspberry Pi Zero W, and the battery. The PCB will also contain the circuit [2] we use to make sure that the device can charge properly. You can see our designs by referring to image 1.0 and 1.1 in the appendix.

The next part of our hardware design will consist of the case. Our case design is simple, with a big base which is measured out to fit our expected system size inside comfortably. We can then simply put the case on top of the base, boxing in our system. We will then use screws to hold everything into place. For our current case rough draft, refer to image 1.2 in the appendix.

The final part of the hardware design is the dock. We would like the dock to act as a charging station, as well as a means to hook up our system to something like your TV. We would like to have the ability for users to use wireless controllers via bluetooth in order to play this way, however we can also add the ability to use wired controllers via USB if needed. You can see our rough draft design by referring Images 1.3 and 1.4 in the appendix.

2.1.2 Software Design

The kernel module is a key part of the project. We are designing our own kernel module in order to try and reduce potential input latency when a user presses a button or inputs a motion on the joystick. We will do this by utilizing interrupts upon peripheral action. The Raspberry Pi Zero also has 40 GPIO pins which we will be using to connect the 3.5 Inch LCD as well as all of our buttons and joysticks. These the kernel module will be interfacing with these inputs directly, which is why we chose to make our own for this project.

We also plan to make an Atari 5200 emulator do go along with our hardware. We chose to do an Atari 5200 emulator because there are very few good quality solutions out there and we feel like we could do it very well. On top of that, our software is going to be compatible with RetroPie so users can have access to various good quality emulators, since we can’t design them all ourselves.

2.2 Design Analysis

Our current PCB design consists of three total boards, a mainboard and two daughter boards currently attacked with a perforated edge. The mainboard contains the circuit for charge protection and the placement of the Raspberry Pi Zero W as well as the inputs and outputs. The daughter board contains a perforated edge with the right side containing the inputs we want on the right side of our device, and the left board containing the inputs we want on the left side of our device. This design is cost effective because a batch of ten boards only cost $2 if we keep it under 100mm x 100mm. The drawbacks to this are we have to figure out a way to connect the boards together which should be as easy as a ribbon cable soldered to the the provided pads.

Our case and dock design are both simple. We designed them using CAD in order to have the ability to 3D print them for cheaper costs. A simple design also makes it very user friendly so that they can just open it up and start playing. The dock has a seat for the device to sit nicely and has HDMI and USB-C output for video and charging purposes. The case has a simple base where we can screw the device in, and then a top outline which we can just place on top and screw into place. This leads to easy manufacturing and ease of use for the user, as it is a simple design to understand.

The kernel module will be designed with the hardware in mind. We want to use it so that the device can pick up the information from the GPIO inputs instantly in order to reduce latency lag. This will prove beneficial from a user standpoint because input lag is a very real issue in game design that needs to be figured out so the user does not get frustrated. Our custom module will allow the user to not get frustrated and just enjoy the device they are playing on.

3. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library.

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project.

The necessary types of tests are:

- Hardware testing for the power circuit [2] to ensure reliability and prevention of short circuits and well as overdischarge of the battery
- Hardware testing of the joint between the RPi and PCB to ensure it is electrically connected
- Hardware testing for the screen to ensure the displayed image is accurate
- Software testing with the buttons to ensure the input is registered quickly and accurately
- Hardware testing of the entire system to ensure power draw is at its minimum
- Software testing for the kernel module to ensure the kernel can be upgraded in the future without breaking the module
- Software testing for the emulator to ensure accurate and efficient emulation.

The procedure for testing the power circuit will be using a bench power supply to slowly lower input voltage (from the battery side) to check if the boost converter can handle the out of spec voltage in order to ensure system stability if the battery degrades over time. A challenge will be the simulating the power circuit since it has been a few years since anyone on the team has touched circuit simulators.

The procedure for testing the joint between the RPi and PCB as well as the screen will consist of an oscilloscope to check the signals being sent and if they match the expected waveform. The screen will also go through a visual check for the displayed colors and the LCD profile will be adjusted in order to ensure color accuracy.

The procedure for checking power efficiency will be using a multimeter in the amperage setting to check each components power draw and any software configuration will be tweaked to lower power draw.

The procedure for checking the kernel modules' extendability will be installing an updated kernel and adding the proper initrd or initramfs hooks to ensure our kernel module is rebuilt and reloaded with every new kernel.

The procedure for checking emulation accuracy will be based of existing emulator tests and well as checking the leaked reference manuals for the proper system with reverse engineering tools such as IDA and Binary Ninja to ensure the correct opcodes are being read and executed from the ROM.

3.1 Interface Specifications

We will be using the GPIO pins on the Raspberry Pi for button input and potential for driving the display as well. The PCB will give access to all of the 40 pins and the display pins are so small it would be very difficult to solder a ribbon cable across all of them to run to the display. If the GPIO pins are unable to be used for display output we could use an external FPGA to run the display it would increase our power consumption but reduce engineering difficulty.

The kernel module will interface with the USB bus in order to enable interrupt-like performance for input lag reduction.

3.2 Hardware and software

We will be using a multimeter, bench power supply, oscilloscope, and a logic analyzer. The multimeter will be great for testing the battery voltage and current draw. The power supply will be used to test if circuits work first before introducing the varying voltage from the battery discharging. The oscilloscope and logic analyzer will help check the input from the buttons for accuracy as well as checking and debugging output to the display. We will be using EasyAda for circuit simulation as it is online, free to use, and can also convert the circuit schematics to PCB schematics which we can then edit and send in for ordering.

We will also use Eagle and numerous circuit simulation software in order to test our circuits before building them physically as well as to check the correctness of circuits built by the PCB supplier's robots.

3.3 Functional Testing

Unit tests can be easily composed for the kernel module as well as the potential emulator. The upsides to these methods are that we can incrementally develop tests as well as the software itself. In the case of the kernel module, these would take the form of tests to quantitatively test whether we are getting rapid enough input for USB. In the case of the potential emulator, the tests would be in regards to accuracy of emulation. Ideally, speed would come naturally to a properly implemented and accurate emulator. Potentially we may be able to test for speed in the emulator, but it is possible that this will not be practical in a unit test.

Emulator functionality will be tested via the Go programming language testing library. Our Atari 5200 emulator will be built in Go and targeting the kernel module for input and SDL2 for graphics. As such, SDL2 can be tested at build time as per the source distribution and the emulator proper can be tested using the test tool provided with the Go distribution. Testing using the test package within Go can create unit tests for internals and externals at once as the tests run within the package being tested. State can be accessed easily as a result which, when combined with copious amounts of assertion statements, leans the suite exceptionally well to testing an emulator.

System testing will most likely have to be done manually with quantitative results being calculated where possible. Ideally, the system's unit tests should take care of accuracy, so system tests should be focused on attempting to measure performance and minimize latency within the system itself.

3.4 Non-Functional Testing

Testing for performance will in major take place in the system testing phase as it is an important part of our system's functionality. That is, real time performance is a goal of the project and is a functional rather than non-functional goal.

Security will be tested by attempting to breach or penetrate the protections of the emulation station and derive from that point what potential strengths or weaknesses are within the architecture of the project. Compatibility with existing systems is tested by the process of loading existing roms and comparing the performance and accuracy experience with existing systems to our system.

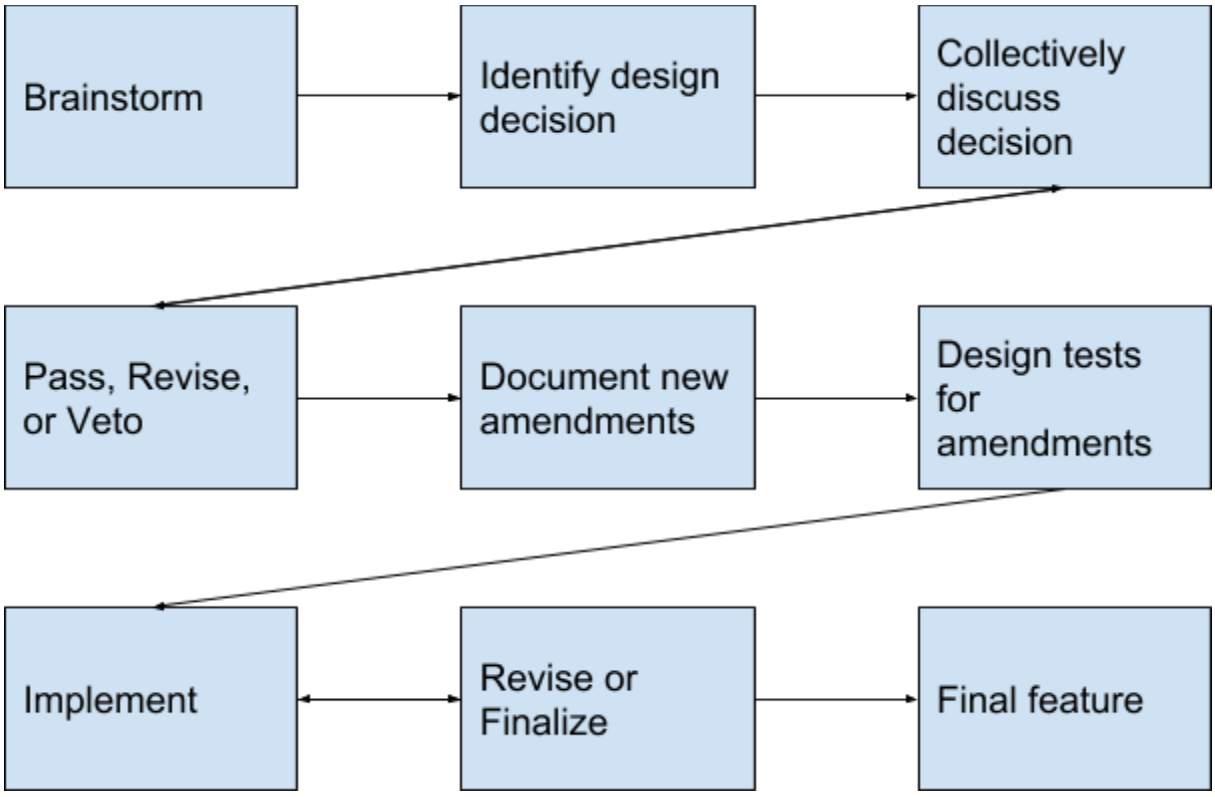
Usability, similarly, takes root in comparisons to existing systems and the usability present there and comparing our system thereto. We will be comparing them in multiple categories such as

their portability using size and weight, as well as using other factors such as durability such as resistance to fall damage, button wear, and case scratching.

3.5 Process

Methods described in section 3.2 are explained in line with their definitions.

Flow diagram for testing and design development:



This is the overarching process we have taken for creating and revising this project. We brainstorm new ideas or the exact implementation or

3.6 Issues and Challenges

3.6.1 Issues

The main issues we have run into so far are with the PCB design. As beginners of PCB and circuit design we had to learn and retain enormous amounts of information which can best be done with trying it out for ourselves, failing, and learning from the failures.

The mainboard V1 did not account for the required trace width for the amount of current that will travel over the traces. Based on our design specification of two amps to power the Raspberry Pi Zero W comfortably, a minimum width of 0.781mm is required as based on IPC-2221 [3] for all external layers of the PCB. V2 has most of the correct trace width set however the ground traces should still be enlarged just in case. We also need a ground plane instead of all the grounds being wired together locally. A universal ground plane is better circuit reliability.

The daughterboard need to easily be separated once delivered but must not separate while being manufactured. The current solution was to add pre drilled holes to the PCB. This is fine for now but will be revised in V4.

3.6.2 Challenges

The main challenge for PCB design is still tool experience. These are very sophisticated pieces of software that allow you to do many many things and thus have options scattered everywhere. Learning which options we need and where they are hidden has proven to be a real challenge. In fact, the main reason we do not currently have a ground plane is because we were unable to find the option to do so before we needed to send the gerber files over and order V2.

Another unexpected challenge is since we proposed this project we have to outline and think about every aspect of this project. We do not have the luxury of a company making the creative decisions the pointing us in the preferred direction. This does give us the ability to add aspects we are passionate about such as the dock.

3.7 Results

We are just beginning to breach into the assembly and testing phases. Based on our test, we have concluded our hardware is currently nonfunctional. This is due to the missing transistor that was out of stock but is en route. It is unlikely to be delivered before this document is due. Once we have all parts in hand and soldered onto the board, this section will contain the the images of oscilloscope testing of the PCB as well as the unit tests for the kernel module.

The testing phase will be very important to this project and therefore this section will contain lots of information about the struggles and successes of the testing phase but since we are unable to seriously test the hardware yet, the information is not available at this time.

4. Closing Material

4.1 Conclusion

Our current design will allow us all of our desired deliverables to the customer. A 3000 mAh battery will allow us huge amounts of battery life based on the specifications of our software and the software we will be developing the hardware for. Our solution is cost-effective, as it has been designed to be cheap to produce, while also thinking about the users needs. We want our users to have minimal latency issues when using our device, and we achieve that with our development of a custom kernel module. We want this device to be something that is by gamers, for gamers. We have used that as our philosophy throughout the project and our current project matches those needs.

4.2 References

[1] “From Idea to Schematic to PCB - How to do it easily!” https://www.youtube.com/watch?v=35YuILUlfGs

[2] “DIY LiPo Charge/Protect/5V Boost Circuit” https://www.youtube.com/watch?v=Fj0XuYiE7HU

[3] “Generic Standard on Printed Board Design” http://www.ipc.org/TOC/IPC-2221.pdf

4.3 Appendices

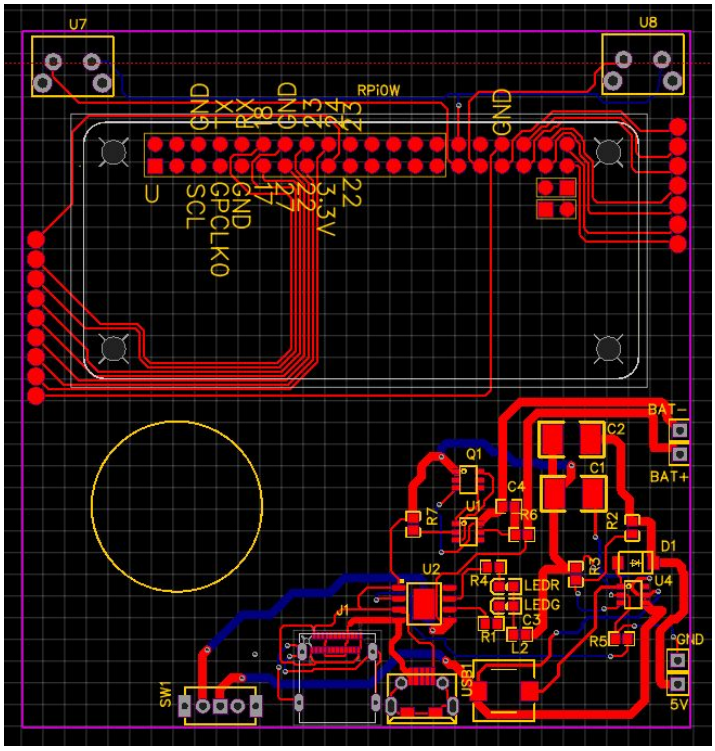


Image 1.0: The PCB v2 for our main board.

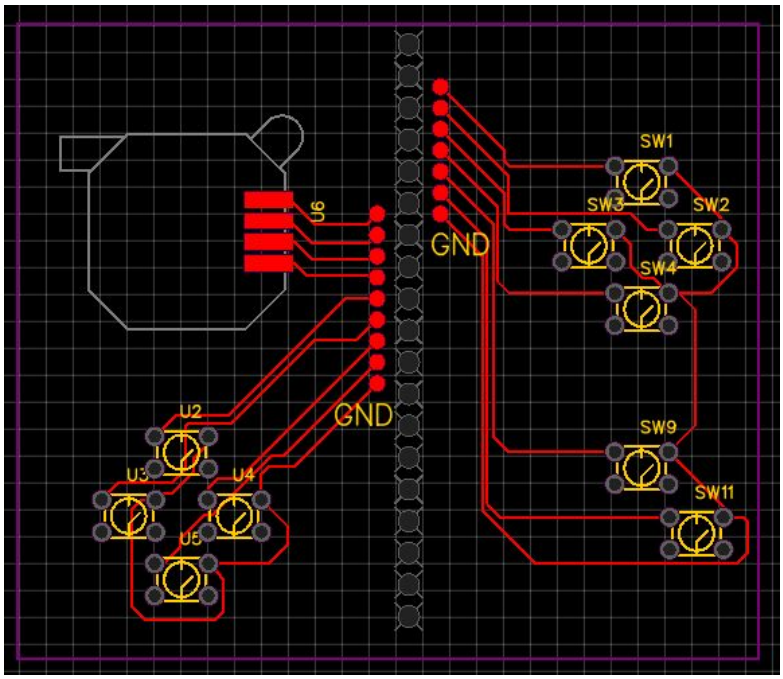


Image 1.1: The PCB v2 for our system peripherals.

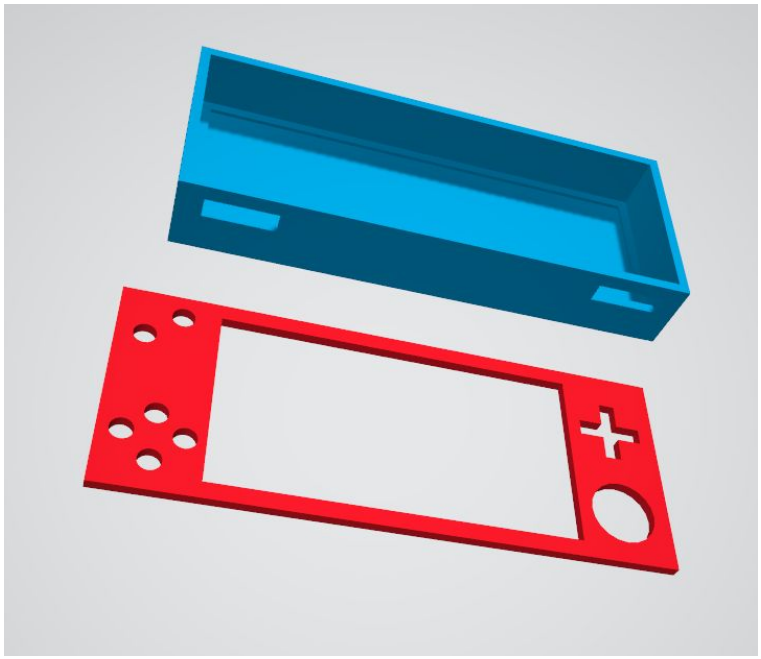


Image 1.2 Emulator Case Design v1 (Rough Draft)

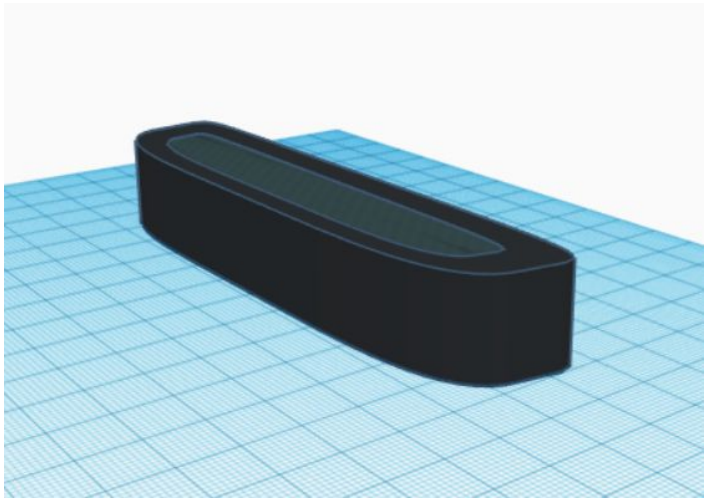


Image 1.3 Dock Design v1 (Front, Rough Draft)

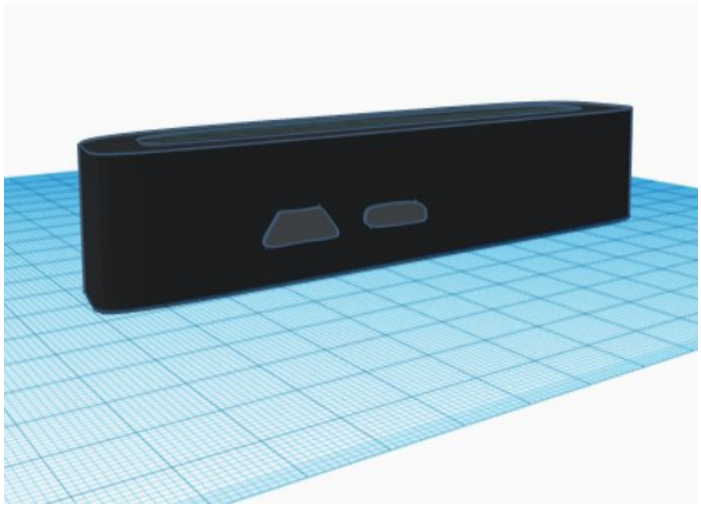


Image 1.4 Dock Design v1 (Back, Rough Draft)