5 Testing

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

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

- Temperature
  - Will have a temperature sensor to measure temperature within the case to ensure the device does not overheat
- Cabinet
  - Measuring and verifying it fits within a 2'x2'x2' area
- Audio
  - Manually
- Power
  - Tested by multimeter
- Circuits
  - Tested by multimeter
  - Simulated through TinkerCad before finalizing
- Software
  - Manual testing when needed
  - JUnit for edge case testing
  - each screen will be tested

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

- User Interface

    - Controls interact with the UI and games
- Cabinet Frame
    - components fit and are secure within frame
- Raspberry PI
    - Receives inputs from controls
    - Displays on monitor
    - Audio output

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

Hardware:

- the Case is 2x2x2
- Simulating the needed circuits using TinkerCad or another software
- Creating each PCB
- Verify each module (audio, temperature, lighting, controls, display, power) is working correctly and separately
- Combining each one by one Raspberry Pi testing each component as we integrate them
- Power circuit will supply 12 volts, 5 volts, and 3.3 volts and have screw terminals for attachments.

Software:

- testing each page connects well together
- testing that the file system connects to the frontend pages user interface pages
- testing that our controls will integrate to the game button presses
- integrating all of the software onto the raspberry pi and ensure it can handle it

Most of, all testing that the hardware works properly with the software

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

The most important tests on the system level are related to interaction of components of the machine. Ideally when the components are hooked together they are already properly tested and confirmed to be working properly, So it would only be important to test the interfaces that connect them this could include:

- Input from arcade joystick to software
- Output from software to speakers and monitor
- Input from on/off switch to pi
- Output from software to LED strips on side of machine
- Input from game filled USB

This is not a comprehensive list, but just to get a general idea of components that may be broken because of how different from each other these problems are it will probably be necessary to test each manually as thoroughly as possible.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure do not break? Is it driven by requirements? Tools?

Hardware: As we add more ideas to create our arcade machine, we try to implement new or improve prior components. Our machine is pretty modular on the hardware side, regression testing isn't an important factor if we add something new but it will still be done. We will check the power, our circuits, the audio and display, and other components frequently to ensure everything runs as it should. As we add more features we will test the core components and make sure they continue to function as expected.

Software: The addition of new games to the cabinet need to be addressed in regards to testing. We plan on implementing a CI/CD system to push new additions to the system without it breaking the current version. We plan to have low coupling and high cohesion so that if we wanted to change one portion of our software it would not break or cause major changes to other units.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

Our acceptance testing will include actually playing the games on the machine. We will look for things like consistent frame rate and reliable controls. Our client is in our group so as long as he determines that the games are playable at a reasonable level, then we will meet our acceptance testing criteria.

5.7 Security Testing (if applicable)

N/a

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

- Audio
  - an ability to control the output volume of the speakers from 0% to 100%
- Power
  - To adapt 120 volts wall power to the needed powers or our systems
- Circuits
  - (on/off) to control the power reliability (99% of the time) on and off with a button press.
  - (temp sensor and fan) to have a temperature sensor as an input to the Raspberry Pi. When too hot, the Pi will turn on the fan. When the temperature returns to a good operating temp, the fan will turn off.
- Software
  - Ability to boot up into the UI after being turned on
  - Ability to use UI to load games and display them on monitor

- ○ Ability to play game with no interruption or unintended effects