Midterm Documentation

Electronic Product Design for Music and Audio

Silas Wang

Product Overview:

My Product Design Product is a handheld game-controller device. It is designed to simultaneously control music software / instruments and a video game, or some software that interfaces with HID input. A demo of the project can be found here.

Features:

Here is a nonexhaustive list of everything that's been accomplished:

- Breadboard prototype with 12 push-buttons (4 DPAD buttons, 4 XYAB buttons, LB, RB, and two miscellaneous buttons), 2 potentiometers, 2 joysticks (each with an X and Y potentiometer and a select button), a gyroscope (temperature, X/Y/Z readings, and rotational information), and a DAC.
- A USB-Composite device system through modifying `boards.txt` and `usb_desc.h` within the internal Teensy files of the Arduino IDE.
  - The ability to simulatneously send MIDI-CC and keyboard input through a single parameter.
- Signal smoothing code for `analogRead()`
- Parameter reading to the display and Serial communication
- Object-Oriented parameter design
- A frame-buffered display for CPU-efficient display rendering.

Roadmap / Feature Wishlist:

A nonexhaustive list of all the planned future features (highlighted features are prioritized)

- Object Oriented parameter design improvements
  - callback functions for reading values / handling events asynchronously
- Menu / GUI infrastructure to be able to design systems that allow the user to visually edit the different parameters on a given program or to provide visual feedback to the user for the software.
- Asynchronous `analogRead()` using the `enableInterrupts()` and `disableInterrupts()` methods from the Teensy 4.1 `ADC.h` class. This must be done as libraries like `analogReadAsync()` does not work with the hardware architecture of the Teensy.
- Behaviors
  - Allows the user to set the ways in which a parameter behaves. One per parameter, per output must be allowed.
  - For instance, a button can be a simple toggle switch (on/off), trigger a random value upon button press, cycle through an array of values, probablistically trigger a set of values, etc.
  - Similarly, potentiometers can also simply display the X and Y values, a polar coordinate, or a logical computation between the two (print X if X>Y, Y otherwise), etc.
  - Far-fetched behaviors:
    - physics simulations (each bounce of a marble in a container is an output of some kind)
    - time-stretching (ie the incoming values for a parameter will be slower by a given factor)
    - logic gates (ie a button will only output a value v if conditions A and B are met. If we assume condition A is when the button is pressed and condition B when the gyroscope detects very little movement, we can easily create very dynamic conditionals for data to be sent.)
- Modifiers
  - Allows the user to fine-tune the parameter's pre-behavior / post-behavior values to their liking. Aggregates in a tree data structure.

  - for instance, a button can obtain the behavior to send values $a, b,$ or c as an output value. The user can set what each value is, and whether or not he wishes to bias an output (ie introduce a coefficient that will be summed or multiplied with the output), slew the output (if a was sent out first and b is the next value, we interpolate between the two discrete values), or invert the output.
  - similarly, a potentiometer can obtain the behavior to send the polar coordinates of its respective x and y potentiometers. It can decide the smoothing value, bias the outputs using some mathematical function, or feed the outputs into a separate function of its own (ie if x and y are phase or frequency parameters for a Lissajous figure, and the output of the function is the phase or frequency relationships of the two)
- Modulators
  - additional functions that control different modifiers / behaviors (in the scenario of the button above, if we had a way to deterministically change which values a, b, c were being outputted in addition to the button trigger, or a way to modify a, b, c by some amount)
  - audio thru modulator that lets the user control parameter behaviors with audio (a button press gets frequency modulated with a incoming audio signal)
- Bluetooth support for wireless data transmission (if latency is bearable)
- Lithium-ion battery power supply
- tinyUSB stack implementation for thorough USB-Compositing
  - XInput option within the USB stack. (the current duct-taped method does not allow this, but I'm not sure if that is due to Teensy hardware limitations or because of the jerry-rigged composite device)

Weeks 8-13 Work Schedule

- Week 8: Finish implementing Object Oriented organization, add callback functions, aggregate fabrication materials / design process
- Week 9: Finish implementing Object Oriented organization, Menu Design aggregate fabrication materials / design process
- Week 10: Menu Design, Behaviors, PCBs
- Week 11: Menu Design, Behaviors, PCBs, Enclosure
- Week 12: Menu Design, Behaviors, Enclosure
- Week 13: Menu Design, Behaviors, Enclosure, final product.

Fabrication:

Design Goals:

My product will functionally look similar to handheld gaming consoles, namely the Nintendo Switch, the Wii-U, and the Steam Deck, shown below. I plan on laser cutting an enclosure (smaller than my current breadboard layout) to ensure that the device maintains a small formfactor and is handheld. I might need to include additional 3D printed components (for the DPAD, or a custom joystick footprint).

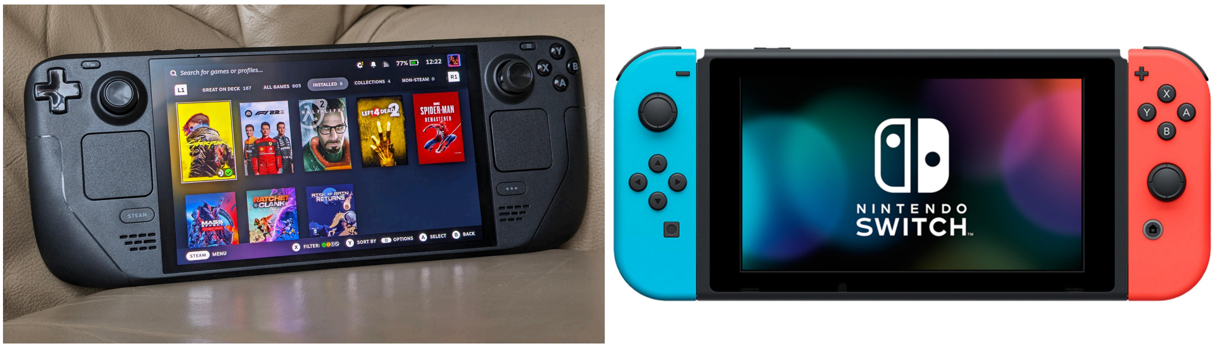

Figure 1: The Valve Steam Deck (left) and Nintendo Switch (right)

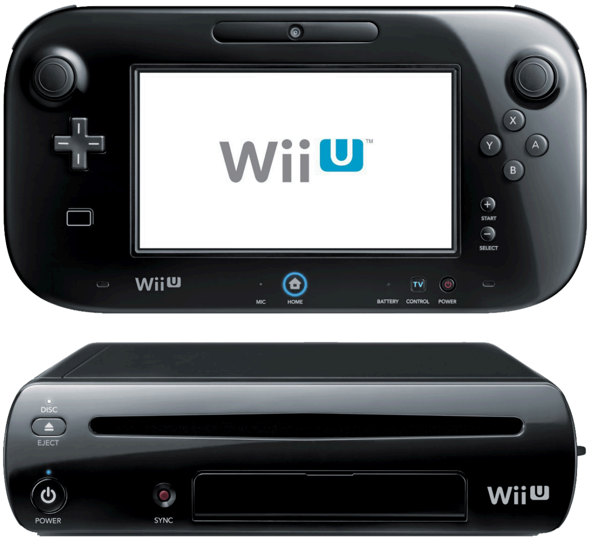

Figure 2: The Nintendo Wii-U

Inspirations:

My product takes inspiration from the Y2K “translucent tech” design, and the transparent yet minimal asthetic from tech companies like Nothing and Teenage Engineering.

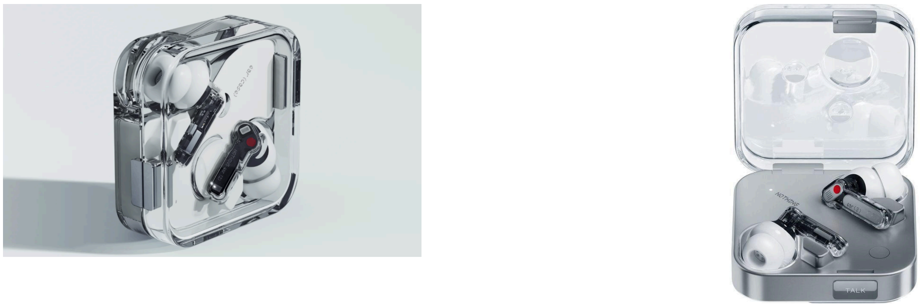

Figure 3: The Nothing Ear 2 (left) and Nothing Ear 3 (right) wireless earbuds

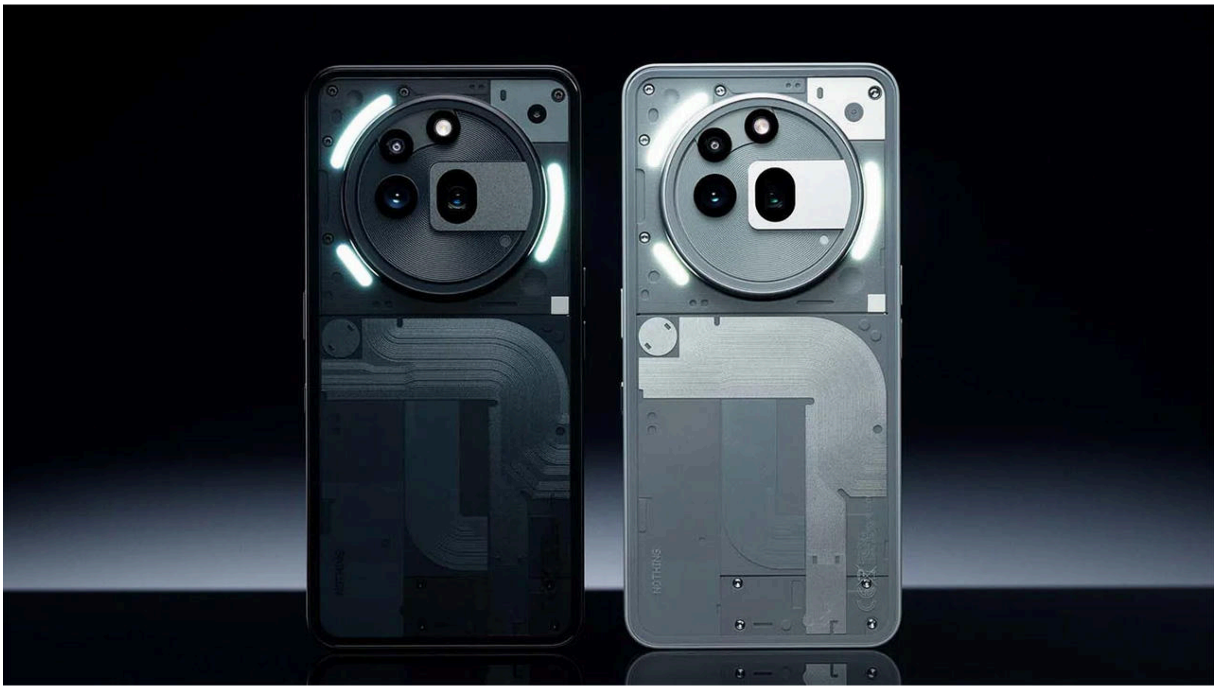

Figure 4: The Nothing 3a Pro phone lineup

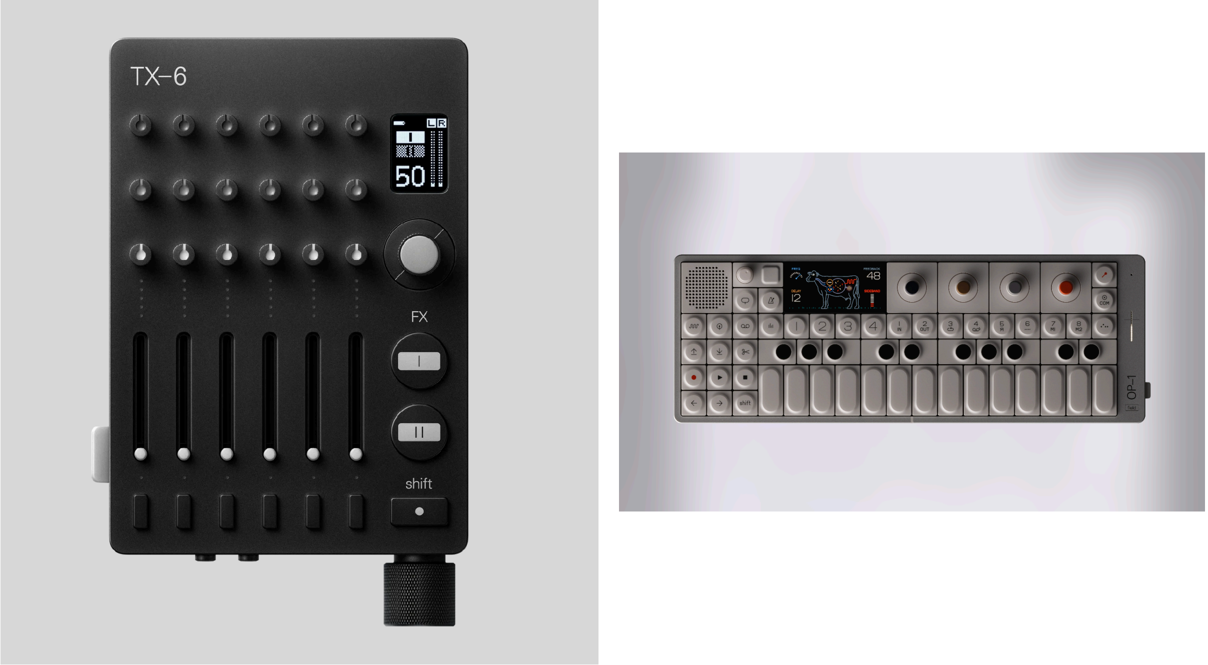

Figure 5: The Teenage Engineering TX-6 and OP-1

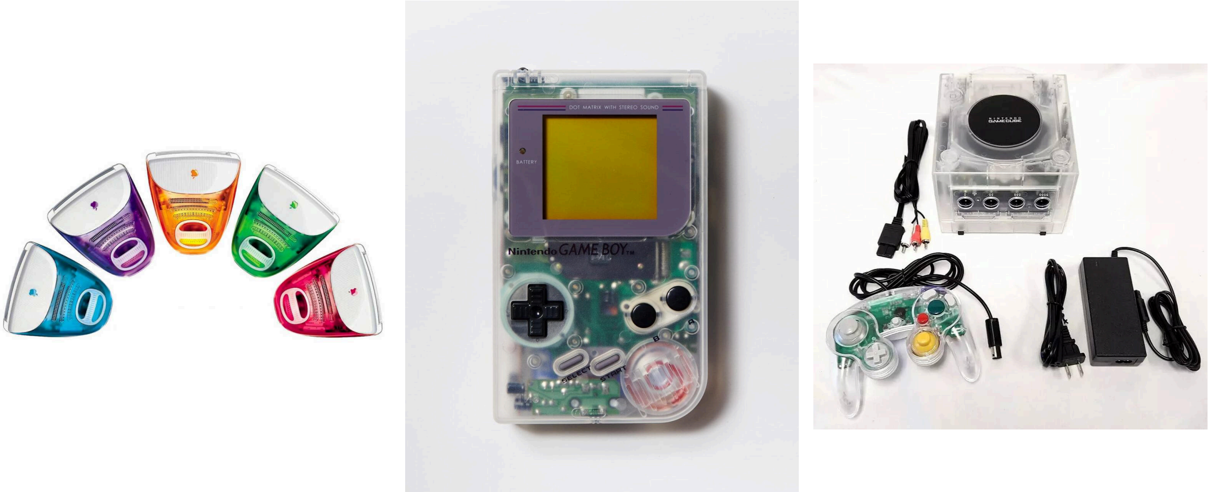

Figure 6: Y2K translucent style tech products. the Apple iMac G3 (left), Nintendo GameBoy (middle), and Nintendo GameCube (right)

Rough Sketch:

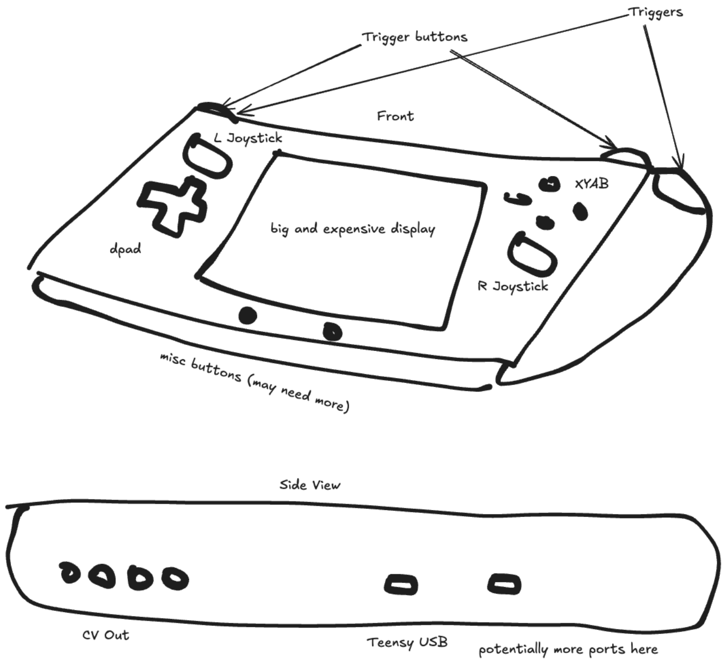

Figure 7: rough sketch of my game controller

For more information or source code, consult the GitHub repository.