GameFrame

Group 16
Allen Chion
Frank Weeks
Israel Soria
Levi Masters

Motivation

- Each of our members wanted to implement certain skills and gain experience to put down on our résumé, such as AI
- Most projects are about something being useful, so we wanted to build something fun
- Help inspire future generations about STEM
- Provide a mobile arcade style experience

Goals and Objectives

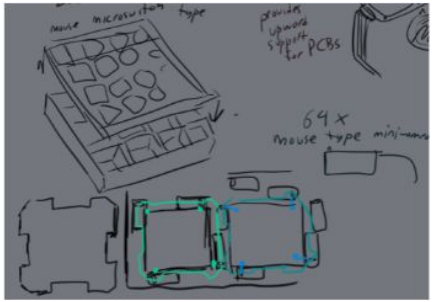
- Create a small and portable board game device
- Lightweight
- Relatively easy to use
- Long battery life

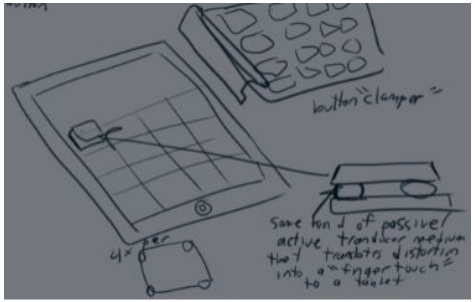
Specifications

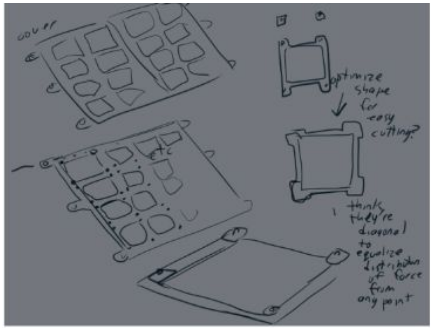
Weight	Less than 2lb
Dimensions	12in by 12in length and width for the base. The height we would like to be no taller than 2 inches
Battery life	2-4hrs
Speed	Less than 5ms response time
Monetary	Should not exceed $400
Software	Should be able to at least play chess

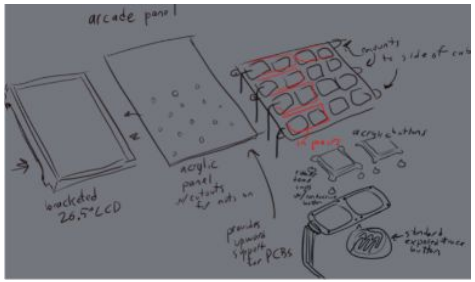
Design Diagram

- Cut costs
- Sustainability
- Button satisfaction
- Functional controller systems



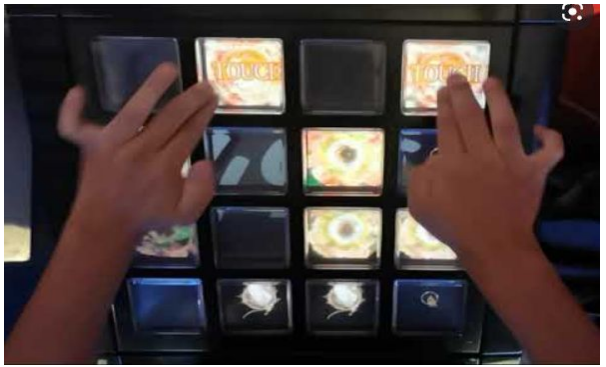






Approach

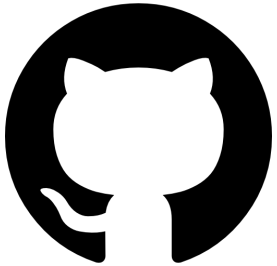
- We wanted this to be portable so the device should be as convenient to bring around as any other board game
- Use a 8x8 button layout for the user to interact with it
- Similar to how players interact with Jubeat, but all of the UI elements will also be done through these buttons
- For example, the chess pieces will appear on each block and pressing the button will select that piece
- The screen will be under the button layout



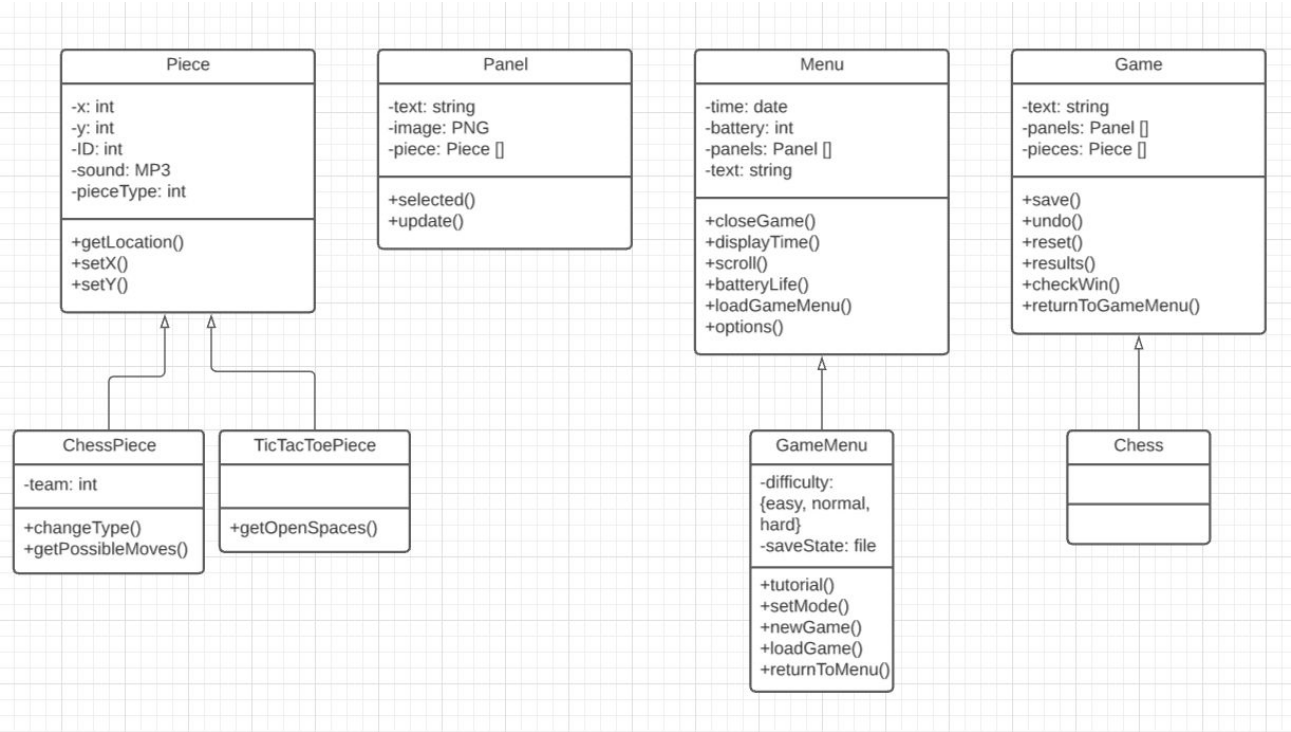
Software

Tools

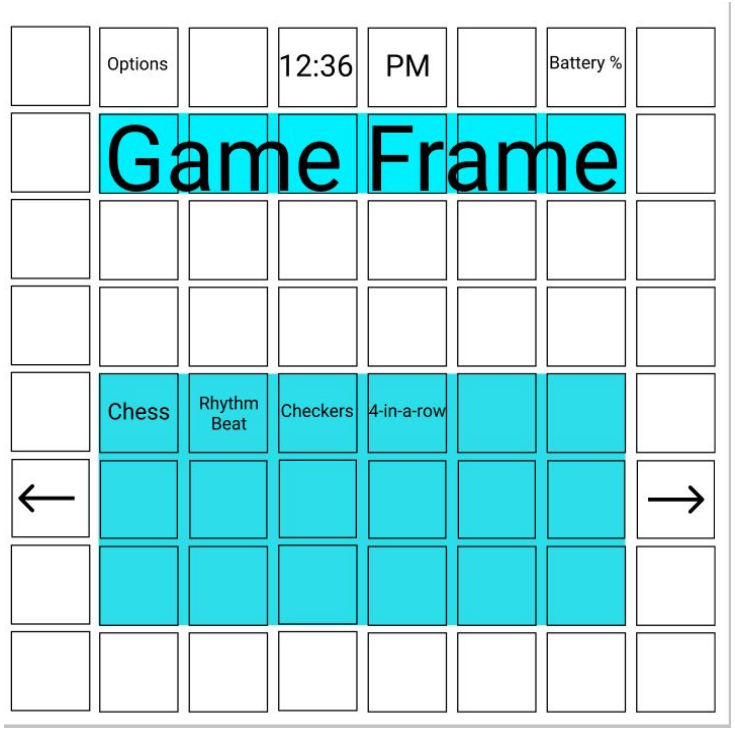
- Python
  - Keep everything the same
  - Assortment of libraries
- Github
  - Keep track of our code
- Discord
  - Communication



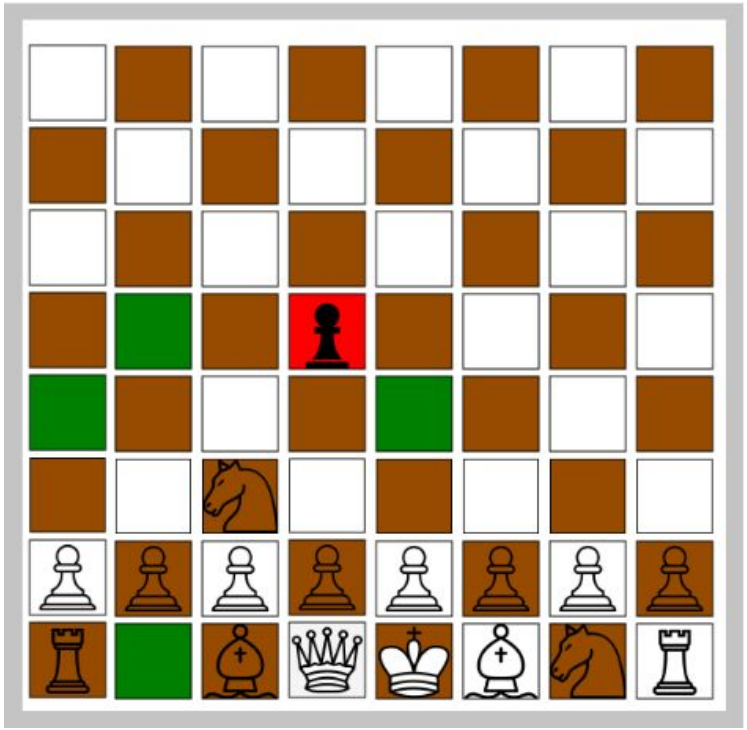
Class Diagram



User Interface Prototype



Menu Example



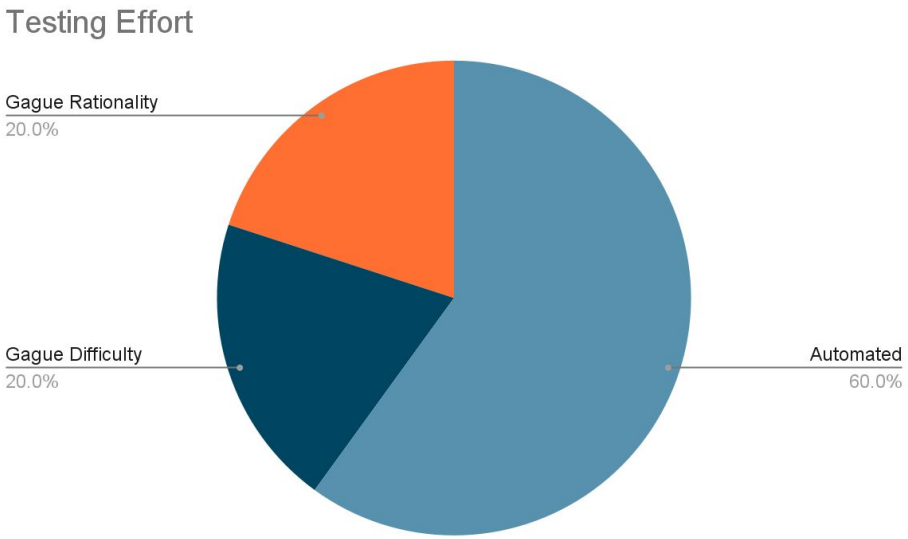
Game Example

Game Engine Testing

- Automated testing to quickly ensure all pieces are movable at all locations
- Human testing to validate automated testing
- Human testing on physical board to ensure proper usage of input and output
- Automated testing on physical board to ensure output is working correctly

AI Testing

- Automated testing to ensure AI can complete games of different states
- Human interaction to ensure AI behaves rationally
- Human interaction to gauge difficulty level



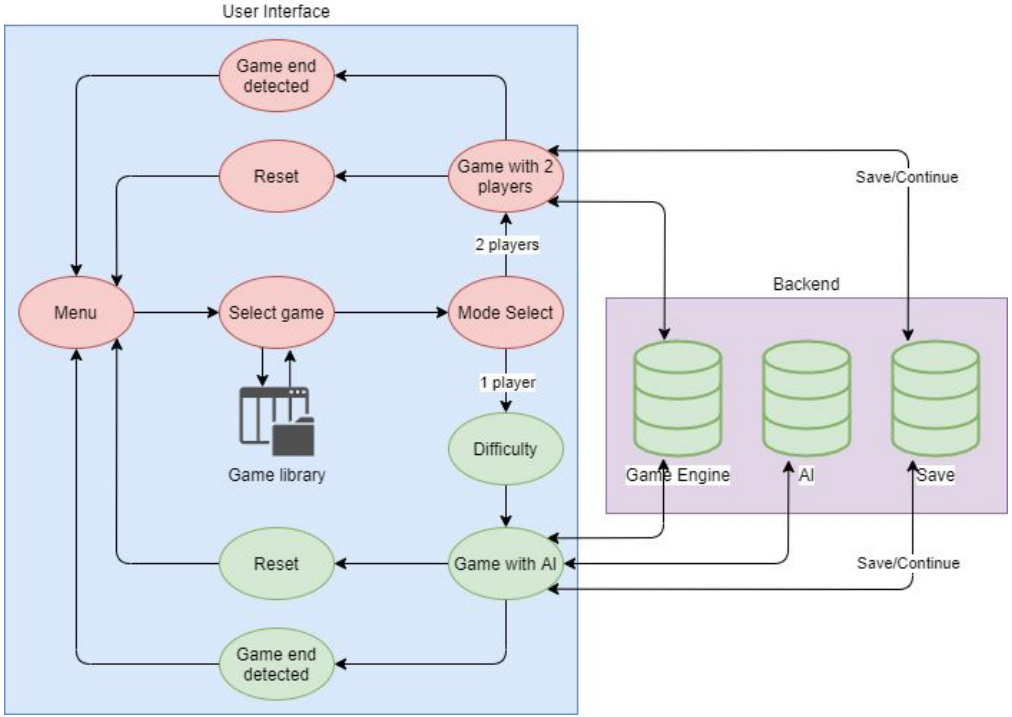
Automated Testing

- Automation for AI produces logs of each move and win status
- Game engine testing moves pieces in many scenarios and keeps a log of resulting board state
- Pieces moved to new spots displayed on physical board
- Flash pieces on screen to test output

E2 -> E4 | D7 -> D5 | E4 -> D5 [CAPTURE] | C8 -> F5 | ... | C6 -> C3 [WIN WHITE]

E2- > E4 [P(A2, B2, C2, D2, E4), N(C1, F1), Q(D1), K(E1)] | D1 -> G4 [P(A2, B2, C2, D2, E4), N(C1, F1), Q(G4), K(E1)] | E1 -> E2 [P(A2, B2, C2, D2, E4), N(C1, F1), Q(G4), K(E2)]

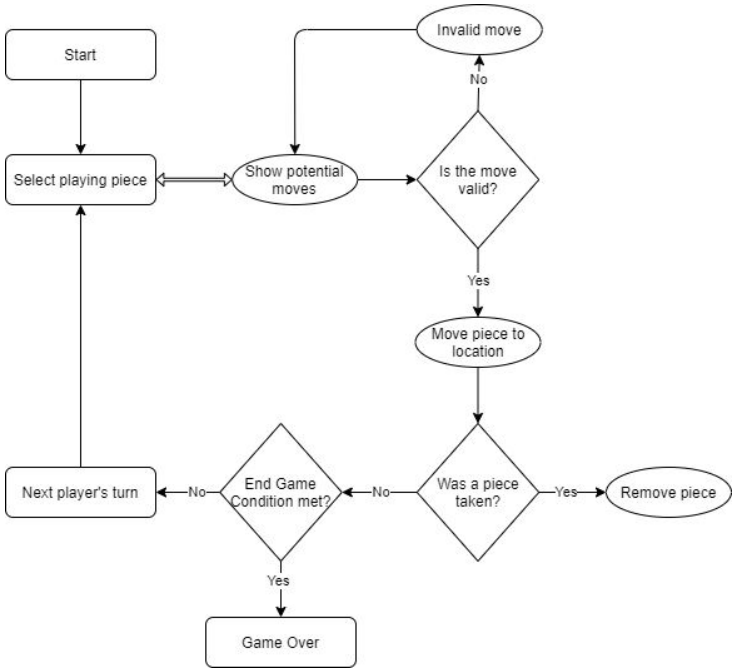
Game Engine Design



- Games are created in “select game” function
- AI is activated for solo play
- Save game is available for one game at a time
- Game engine gives “check_win()” function call after each move

Game Engine Design

- Each tile is given a state, including what piece is present
- Each piece tells the tile valid moves
- The game engine presents the player with possible moves
- Invalid moves are ignored (no action taken, remains player's turn)
- State of pieces stored in array with attributes



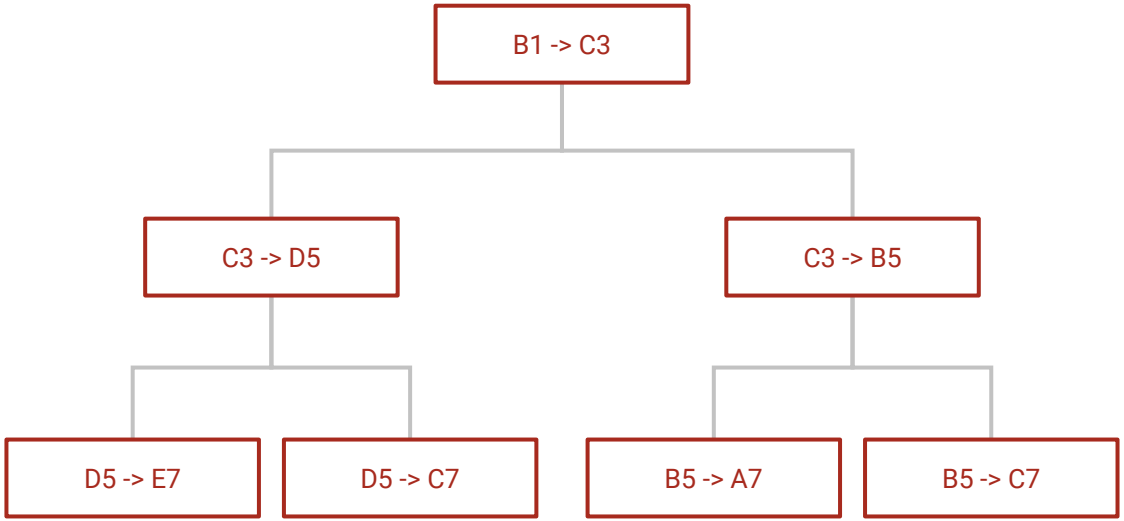
AI (Player vs. Computer)

- Python/PyCuda is the main language used to program the Jetson nano, which is useful for AI/ML programming
- Neural network is used for most AI computer gaming applications
- Different levels of training are given at each stage of difficulty
  - This emulates human playing experience fairly similarly
- Monitored throughout training to gage difficulty levels

AI Training

- Reinforcement learning
- Multiple renditions of the AI for each difficulty level, which are trained differently
- AI plays against itself many times in order to become more difficult
- Add human training as much as possible
- AI plays against its own different renditions of difficulty
- Specific renditions of the AI created only for training

AI Alternate Considerations



- Backtracking
- Try a bunch of simulated games
- Choose best outcome probability
- Simulates AI and Player moves

Hardware

Hardware - Overview

Block Diagram

Microcontroller

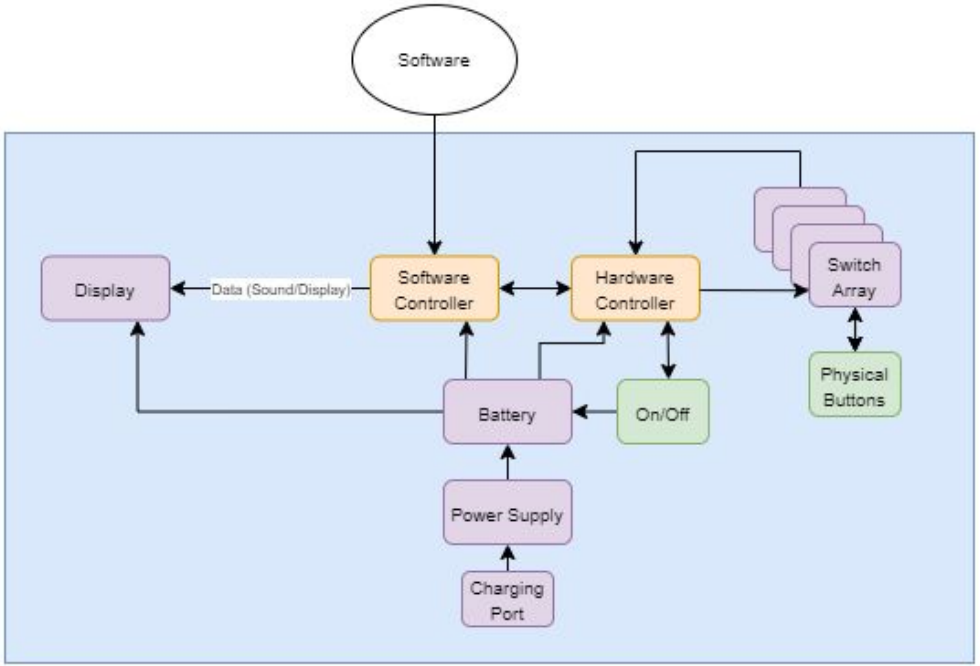
Switches

LCD

Power

PCB

Hardware - Block Diagram



Hardware - Microcontroller

MSP 430

Functions:

Communicating with Jetson Nano

Controlling switches

Hardware - Microcontroller

MSP 430 - G2ET

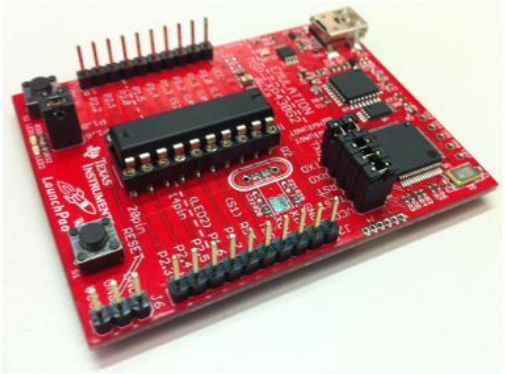
Reasons:

Simple and familiar to use

Cheap ($10-15)

Low power (1.3-400μA)

DIP-Socket compatible chip is available

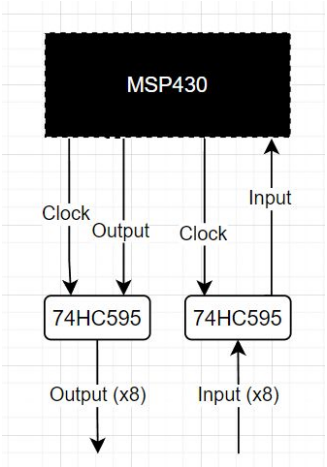


Hardware - Shift Registers

64 buttons means 64 wires

Solution:

- 74HC595 (8-bit shift register IC) x 2
  - One input line, one output line
- Shift registers require clocks
  - Use two output lines from MSP 430

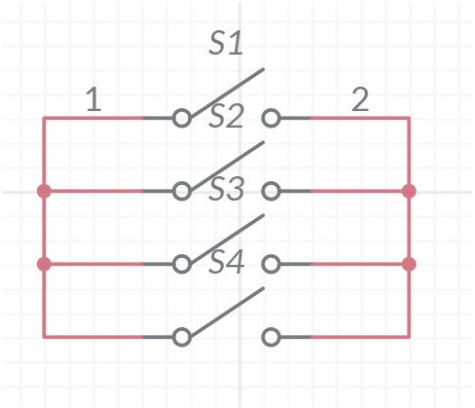


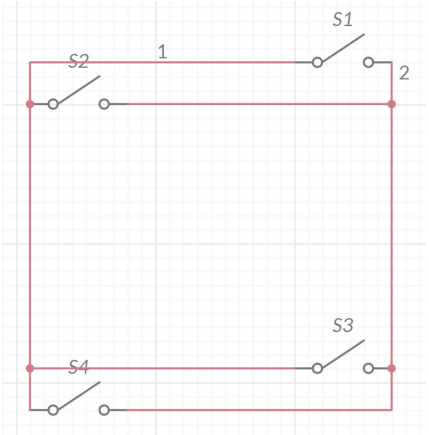
Hardware - Switches

Main method of human interfacing and control

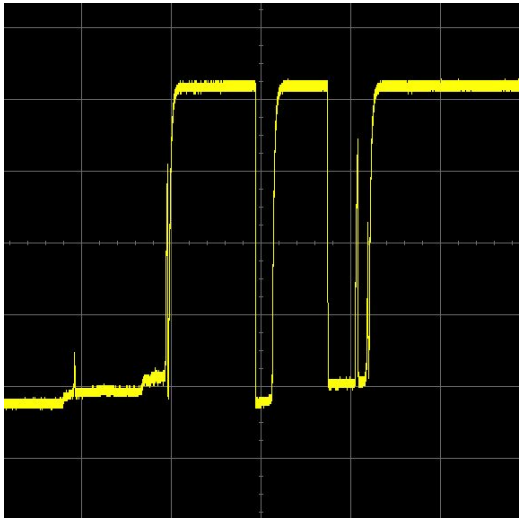
Array of 64 buttons (8x8)

4 Switches/Contacts per button





Hardware - Switches - Debounce



Switching states can create bounce

Solutions:

- Software - While iterating
- Hardware - Filters

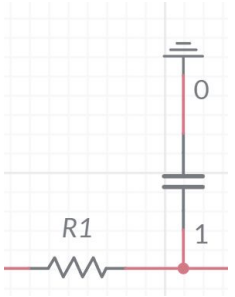
Hardware - Switches - Debounce

While Iterating:

- Track how long a switch is in switched state
- Useable outside of debounce

Contact switches might have worse debounce

- Use hardware solution
- RC spot factored into first PCB




```
while(programLoop)
        addToCounterVariables //Add to each that reads high
        resetCounterVariables //Reset each that reads low
        for each counterVariable{
            if(threshold)
                SendAsPressed
        }
```

Hardware - LCD

Features:

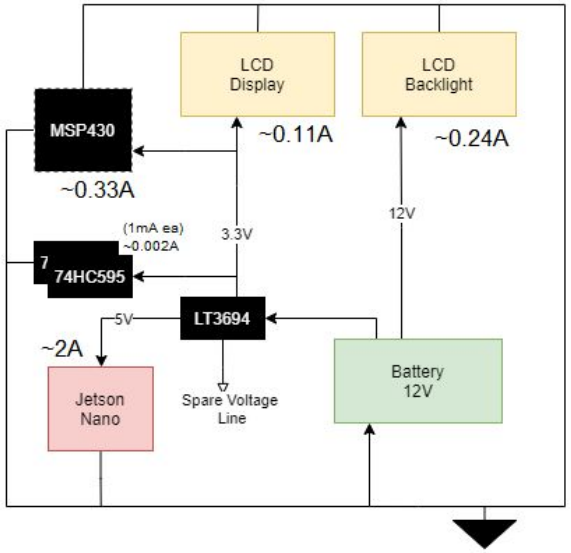
- HDMI compatible
- Built in sound

Hardware - Power

Battery

Voltage regulation

- LT3694 (Switching regulator)
  - 2.6A max output
  - 3 output lines



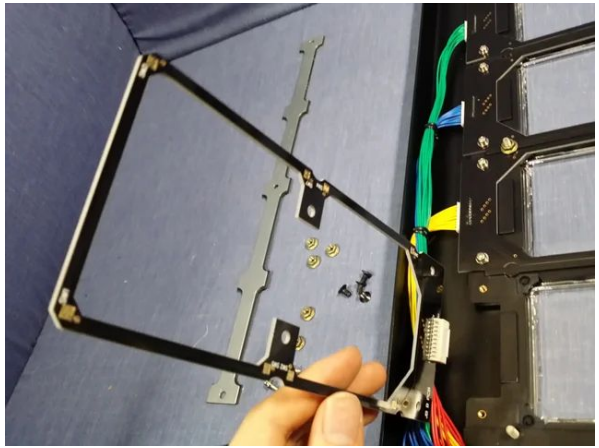
Hardware - PCB

Two PCBs:

- Main
  - MSP430, Regulators, Shift registers, Hardware Debounce
- Switches
  - Similar idea to jubeat

Jetson Nano separate

- LCD connected to Nano



Hardware - PCB

OUR HARDWARE GUY IS BEHIND

Hardware - PCB

OUR HARDWARE GUY IS BEHIND

(but is using Eagle)

Hardware - Testing

Switch debounce

Amperage

- Truly under 2.6A?

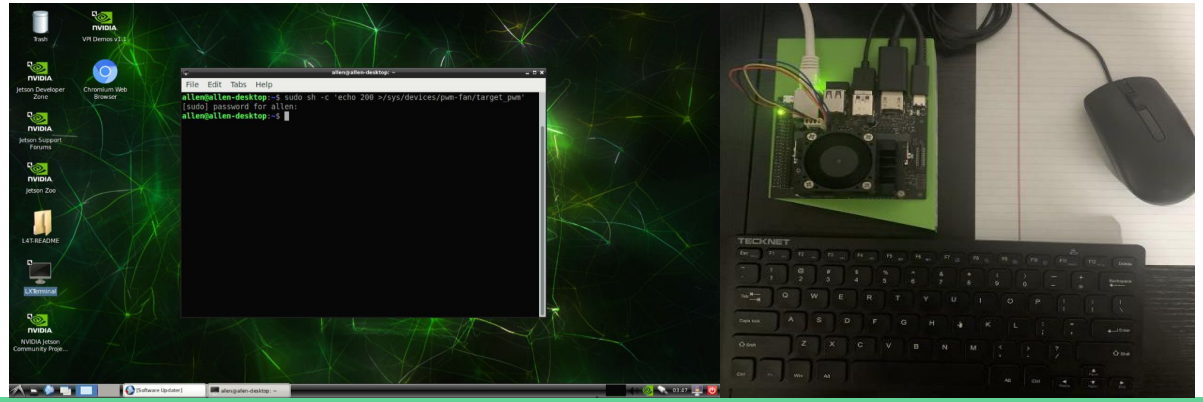
Ability to run in low power mode

- To extend battery life

Temperature Testing

Jetson Nano

- We decided to go with the 2GB version because we reasoned that we would not need more than that to run simple board-style games since the 4GB version costs a lot more
- Built for AI, so this will be running our software
- Hardware can be controlled directly using the Linux terminal or running a script
- Python wrapper built in makes coding it a lot easier.



Design Constraints

- Economic
  - Since this project is not financed, all the funding is coming from ourselves.
  - Ideally spend the least amount of money possible, but we do not plan on cutting too many corners.
- Political/Ethical
  - We have to limit the games that are public domain and do not breach any copyright.
  - No rhythm games that use music since we do not want to deal with licensing.
  - We don’t want to just copy someone else’s game.
- Health and Safety
  - Ensuring the device does not heat up too much.
  - No exposed electronics.
  - Making sure the device is not too heavy.

Budget and Financing

Item	Cost
NVIDIA Jetson Nano 2GB Developer Kit DC 5V cooling fans (2 pack)	$77.79
MSP-EXP430G2ET	$21.28
Switch Regulator	$10
Shift registers	$1
LCD Display (have not chosen yet)	~$60-100

Work Distribution:

- Levi Masters:
  - Game engine and AI
  - Backend
- Israel Soria:
  - UI elements
  - Front end
- Frank Weeks
  - Hardware
- Allen Chion
  - Hardware and software integration
  - embedded programming

Issues and Possible fixes

- We haven’t picked an LCD panel yet and depending on what we end up choosing, the housing will have to be adjusted to fit it
  - Due to size variations, some of the LCD screen space will be “wasted” since we plan on having the interface be a square
  - The pixels on the edges will be wasted space hidden under the buttons and housing
- Cooling
  - The Jetson nano so far has been the hottest component
  - Probably will mount the Jetson nano in a way that the CPU fan exhausting out of the housing

Things we need to work on

- Wait for our PCB to arrive
- Picking an LCD screen
- Design the housing
- Divide up different sections of the big LCD screen into their own 8x8 sections
- Figure out how to circulate the air to cool the jetson nano
- Building our first prototype

Percent Completed