Game Frame

Group 16:
Allen Chion - Computer Engineering
Levi Masters - Computer Engineering
Israel Soria - Computer Engineering
Frank Weeks - Electrical Engineering and CS

Customers, Sponsors and Contributors

Potential customers

- UCF
    - Demonstration of Senior Design for youth STEM programs
        - Give kids something they can interact with instead of just looking at
    - Entertainment in game rooms such as the one in the 3rd floor of the student union
- Parents
    - Give kids something to do that doesn't need internet connectivity
        - Useful for car-rides or travel
        - Reduce need for parental oversight
    - More interactive than watching youtube videos
        - Physical buttons have better haptic feedback than a touch screen
        - Playing games is better for your brain than watching videos
- People looking for entertainment outside of just a computer or tablet
    - Buttons that are physically pressed is more satisfying than tapping or clicking
    - Something easy to keep in a car
    - Doesn’t have the distraction of social media

Potential Contributors

- Members of Group 16
    - Allen Chion - Hardware and software design and implementation
    - Levi Masters - Hardware and software design and implementation
    - Israel Soria - Hardware and software design and implementation
    - Frank Weeks - Hardware and software design and implementation
- Jon “Box” Klages - Consultant with knowledge and experience of arcade machine functionality and operation, including jubeat.

Project and Motivation

Our group consists of three computer engineers, and one electrical engineer and computer science dual-major, so we were looking for a somewhat software intensive project still involving electronics. After brainstorming and working through a few ideas, we arrived at a gaming robot that plays games like chess. This gaming robot would allow for enough design in the electrical engineering space and design that it makes sense for a senior design class as opposed to a software development class. As well, there is plenty of development in software and potential AI to give our three computer engineers space to code. Being an 8x8 grid essentially, we also have options to code in a few different games aside from just chess, including - checkers, connect 4, tic-tac-toe, or even a timing game like jubeat. Being this open ended allows us flexibility in the complexity of the project. We may run ahead or behind schedule and we can simply choose to add more or less features if we are near the end and should we want to implement more games. This project could also be something fun to show off at events like STEM day, which UCF often hosts for K-12 students.

Our very first ideas included a trading card sorter organizer, a poker playing robot, a smart blind system, and an eco-friendly liquid product dispenser (like soap, detergent, cleaning chemicals) for in-store distribution. After discussing our goals for this project, we leaned into the poker playing robot as it had the most amount of software complexity. However, various issues came to light such as the multitude of mechanical components involved in handling cards and poker chips. Eventually the gambling robot evolved into a gaming robot. We all look forward to working on this project, as the end product will be something fun to use, and each of us have something to work on in an area we like.

Market Analysis of Competitive products

Competitive products to ours include a palm-sized digital chess device which uses an unsophisticated LCD and typically requires using a stylus. Alternatively, there are portable chess sets which use a variety of physical chess pieces. A consideration for comparison is also a device like an iPad or tablet PC as these can allow versatility to add and play more games with the same device.

The proposed solution alleviates some of the downsides of these competitive products. Using a rhythm game cabinet known as jubeat for inspiration we wish to design a portable "Game Frame" with 8 by 8 (or 64) tangible buttons. The LCD will be of a better quality and not require the hassle of a stylus. Additionally, with portability in mind, supplementary pieces are undesirable and not required by our device. Even with chess as our focus, the device will have software modularity that will allow other games to be designed for it. Though it will not be able to have the full game library capabilities of a tablet, the tactile nature of the device should prove to be a more enjoyable experience.

With the aforementioned comparisons in mind there are several goals that exist. The device should be:

- Lightweight - No heavier than the approximate weight of a laptop
- Portable - Easy to use with no external pieces, and sustainable on a battery for period of time
- Reasonably priced - No more than the price of a tablet all together

An example of jubeat: Top-down view of gameplay

Requirement Specs

Power

The battery should be 12V and be chargeable off of the USB standard. As it is portable, the design will require a sufficient power supply unit, denoted by the subsequent time requirement. The device should last a minimum of 2 hours so as to be sufficient for portable play. Hopefully, we will be able to reach closer to 4 hours.

Size and Weight

The dimensions of the device should not exceed 12 inches in either length or width. The depth is not mandatory to be below a specific required length. A goal of keeping the device within 2 inches deep should help for the weight. Of which, the weight should not exceed 4 lbs so as to not weigh more than a laptop with similar dimensions.

Price

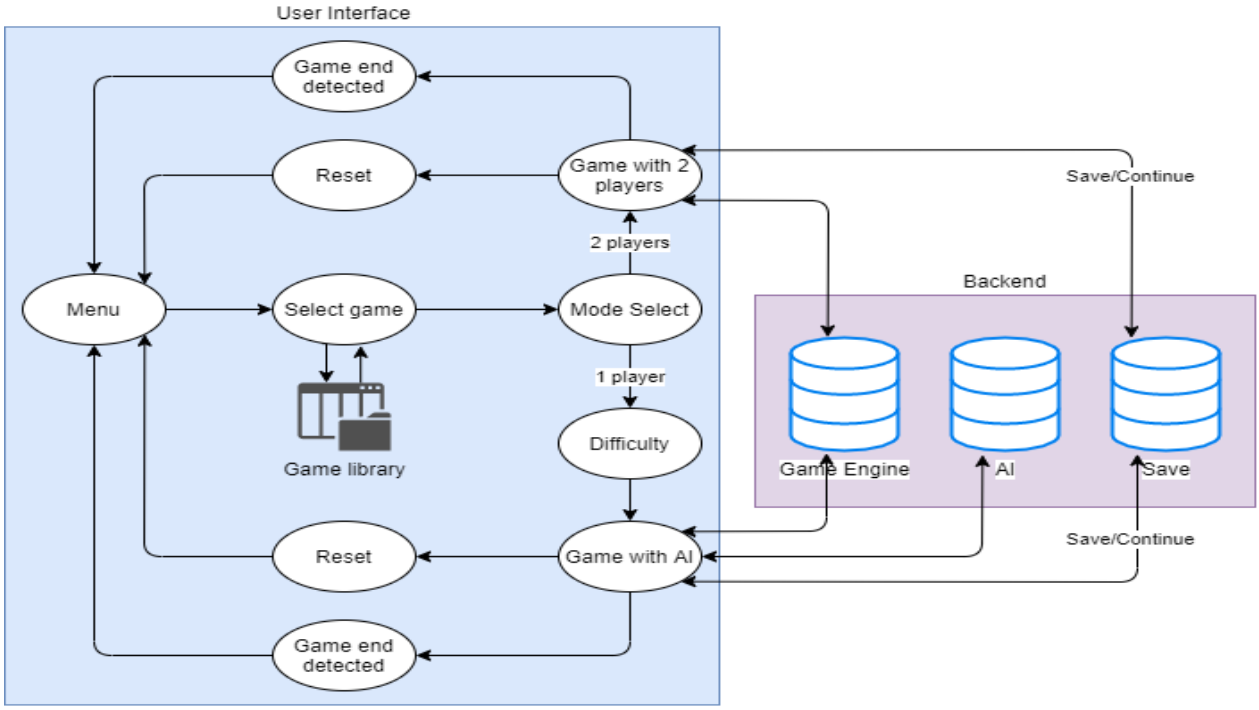
The device should not exceed that of a $400 tablet. Being priced above the $40-60 of comparable cheap portable board games is justifiable, however, with our intent of a higher quality experience. A soft requirement is to make it within the cost of more expensive portable chess games which can go upwards of **$300**.

Software and Microcontroller

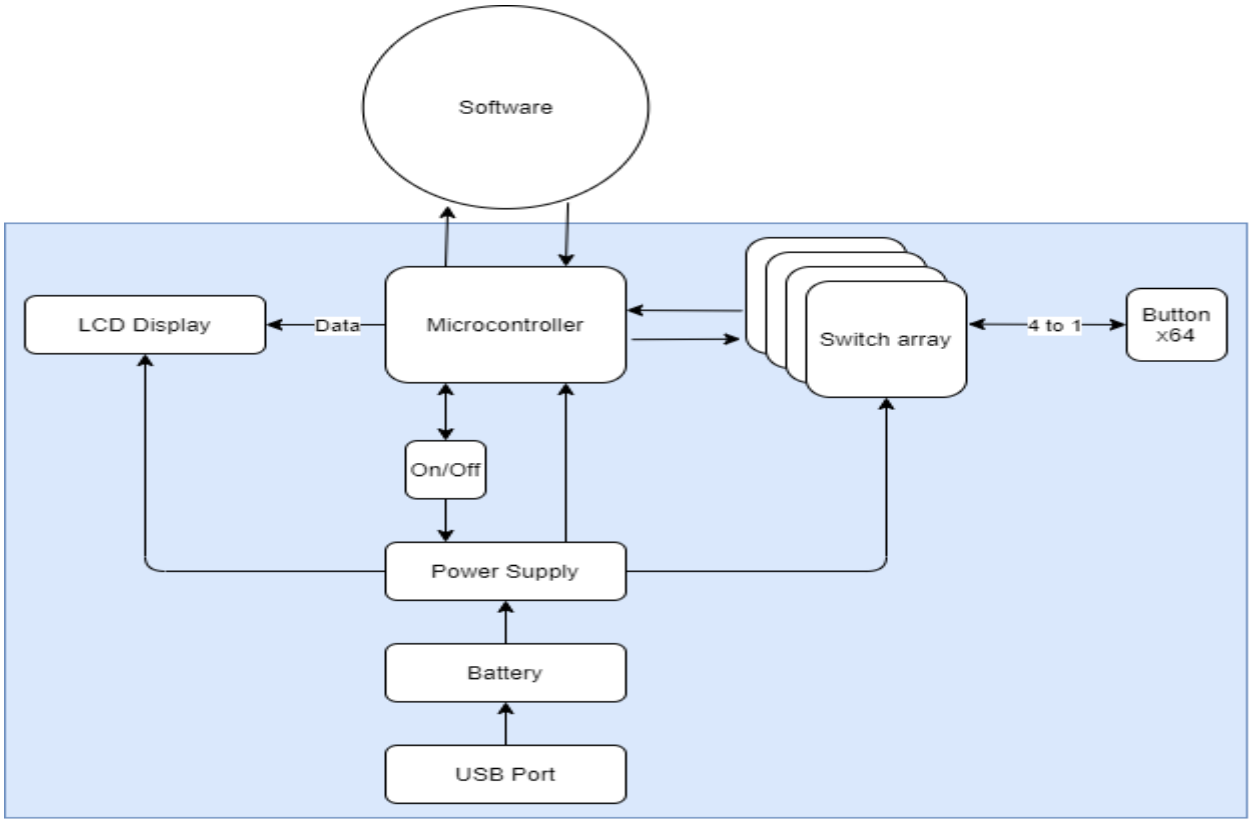
The initial decision for the software language is Python 3. As it natively supports Python 2 and Python 3, the Raspberry Pi 4 is our targeted microcontroller. This might be lowered to a more lightweight Raspberry Pi or Pyboard to help facilitate our power requirements.

Block Diagrams

Software Block Diagram:



Hardware Block Diagram:



Prototyping

CAD prototyping is a goal around the corner as we develop more specificity, but our current early-stage visual prototype for the project is based around a video of DIY jubeat assembly and on a rough basic design diagram of the physical components provided to us by our contact.

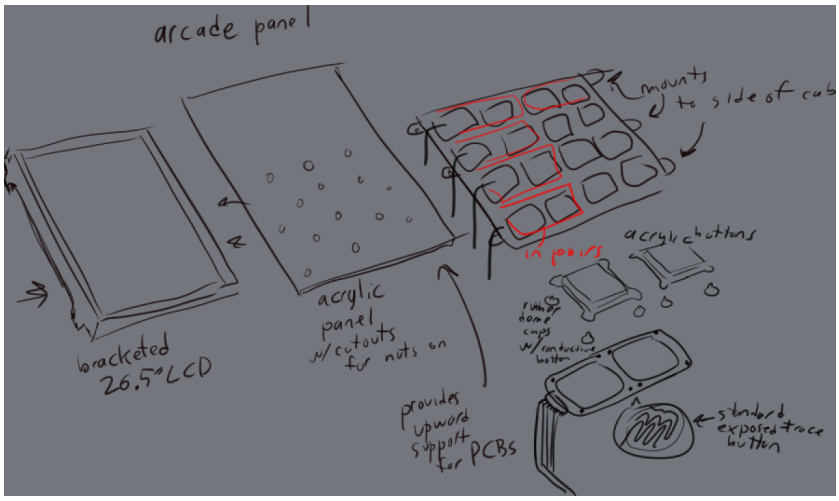


Figure: The rough diagram we were given

Budget

Item	*Cost*	*Source*
3D-printed Housing	$20 (about one roll of ABS or PETG)	ABS Filament 1kg
10-in LCD Display	$30-$60	LCD - Adafruit 7"
Acrylic Squares/Buttons	$3 - $ 28 (depending on thickness)	Acrylic Sheets - US Plastic
Acrylic mounting panel	$3 - $28 (depending on thickness)	See above
PCB	$0.50 - $300	PCB Manufacturer
Snap Action Switches	$64 - $192	UP01DTANLA04
Raspberry Pi 4	$35	Raspberry Pi 4 Mobel B 2GB
Battery	$13	12 V - Mad max
Total	**$168.50 - $676**	

Initial milestones

Milestone	*Milestone Type*	*Details*
Senior Design I		
Functional input/output	Hardware	The microcontroller should be able to read user input from buttons on a 64-panel grid. Input should be sent into the microcontroller for usage in the game engine.
Application can run a game	Software	The software application should have the basic structure of a game (chess) set up, and logic/rules to the game implemented and interactable. Ex: bishops move diagonally, pieces are captured on being attacked, captured king wins the game
Application can play against user	Software	Given a working application, the AI should be able to play a game of chess against the user. It is responsible to both make a decision as to what piece to move, as well as send the game engine a signal to move the piece. It should stop processing moves on capture of a king, draw, or reset.
Senior Design II		
Physical board layout and encasing	Hardware	The playing board on which the users will interact with the game and AI should be laid out in such a fashion that the corresponding game pieces are in the position they correspond to in the game engine. Power switches, reset buttons, and menu buttons should also be attached in their permanent location.
Software-hardware integration	Hardware-Software	The hardware input from the completed player board should fully interact with all components of the application. No computer aid should be required to change game-modes, reset, power on/off, or enable/disable AI or two-player modes.
Fully battery powered	Hardware	The entire game board should fully run off of the battery pack. No external power input should be required, and battery life must last at least 2 hours (as per the requirements).
Fully standalone	Hardware-Software	The game board should not need any aid from wired or wireless computers, internet, or power cords, and it should be able to be transported freely without issues in stability. Ex: If the board is tilted while mid-game, this should not affect performance.
Working Product	Hardware-Software	Game board on startup should boot to an interface which allows the user to start a game, choose game modes, and play a game. Menu should be accessible at the end of a game, and when the menu button is pushed. Board should recover to a usable state after power drain.

Project Matrix

Project Name	*Project Complexity*	*Familiarity with Technology*	*AI Requirements*	*Personal Interest*	*Mechanically Challenging*
Smart Blind	Neutral	Neutral	Low	Low	Low
Poker Hand	High	Low	High	High	High
Refillable Station	High	Neutral	Neutral	Neutral	Moderate
Game Frame	Moderate	High (consultant)	Moderate	High	Low

The interest for the poker playing robot was the highest, and it was the project with the most prominent AI development and computer vision. The portable game device ended up having good interest (though admittedly slightly less). The benefits of the game device ended up being that it was complex enough without being overly complex, we had an outside source to consult with, it was not very mechanically challenging, and it had some component of AI with which we could implement. This allowed those who wanted to develop those skills to do so while also maintaining a solid level of project interest. Thus, it is what we decided.

Alternate considerations and tradeoffs

Language and microcontroller

Python is an expensive language requiring more processing resources than others. It also lacks the resource management and speed provided by an alternative like C. A Raspberry Pi is monetarily more expensive than other microcontrollers that might have less powerful specs in them. The Raspberry Pi, however, has specs powerful enough to implement Python as a default language. Python was chosen despite its shortcomings because of its power and ease of use. Python has access to a wide array of open source AI libraries, so if we augment our game engine with AI or think of additional features that utilize computer vision, Python will already have access to the tools needed.

Another highly favored language might be MicroPython - a Python based programming language designed for the limited resources of microcontrollers. Using less resources might allow saving on unit production costs. Since we are not yet sure of the extent of MicroPython library capabilities, the group decision was to go with Python. If MicroPython proves compatible and it is feasible within reason to port our work, then we might do so as the progress unfolds.

An honorable mention is the Teensy++ 2.0. We stumbled upon this through a DIY jubeat video that our consultant shared with us of another individual. In that video, the person utilized this microcontroller for the panels, but we decided that it wasn't for us. The largest benefit is how compact it is. The person in this video utilized a separate computer entirely to run his game. We decided that we wanted a controller we would be sure could handle both the buttons and the game instead.

LCD Panel

Rather than a single LCD panel with 64 different buttons we considered using 64 partitioned displays for each button. Due to the cost of the individual panel being considerably lower than buying 64 units of the various segmented options (LCD-Keys, small LCD display by Suzohapp, etc). Additionally, it might prove an easier feat to implement the partitioning of the segments on the software side.

Alternate project ideas

As shown in our Project Matrix, located above, we were considering three other projects at first. We thought of a smart blind that would integrate a bunch of features including some features from a previous group. We thought the design was simple and reliable, but we didn't know if this project would be something that we would be proud to create and tell future employers. So, we decided to keep it as a fail-safe in case we couldn't think of a better design. Also, one of our team members wanted to gain more experience with artificial intelligence, so we decided this project would not suffice. A different project we considered was a refillable station that would be used by grocery stores for refilling plastic containers to lower the waste of plastic. This project lacked something that really held our interests similar to the smart blind, so we decided to not choose this project as well. It also seemed like it would be something that really could not be built upon with additional features.

We then considered a kind of robotic arm that plays poker with another person. We realized this project could be a bit closer to the field of artificial intelligence since we would make the robot learn when it is wise to bet and when to fold. This project also grasped our attention more in comparison to the other two projects, because it seemed more entertaining than the others. However, we realized that it would be very difficult to pull this off since the arm would need to be very dexterous since it would be very demanding for the arm to be able to pick up playing cards. So we tried to stray away from card games to avoid the challenges that arose from them. We then decided on chess using the grid-based buttons featured in jubeat. With this system, we can add more features, such as playing jubeat or other possible games in addition to chess, depending on how successfully we proceed with the project.