Game Frame

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

Contents

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
        - Embedded programming
        - Hardware/software interfacing
        - System integration
    - Levi Masters
        - Back-end programming
        - Gaming engine development
        - Logic systems development
    - Israel Soria
        - Front end programming
        - Graphical development
        - UI/UX design
        - Additional back-end programming
    - Frank Weeks
        - Hardware implementation
        - Electronics design
        - System integration
- Jon “Box” Klages - Consultant with knowledge and experience of arcade machine functionality and operation, including jubeat.

Project 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. While engineering is often used to solve practical problems, we wanted to create something less practical and more fun. 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. One of the biggest issues was the dexterity of the mechanical arm. After some more consideration, we figured it would be better to change the game we want to play entirely than to try and solve the problem of playing a card game through different means. Eventually, the gambling robot evolved into a gaming robot.

Some of the motivation for our extra considerations, while still balancing key things like budget, is to build a project and skills worthy of our résumés. Experience with certain devices and languages, interfacing hardware components while following standards, and integrating AI into our game engine are all achievements and skills we can display at the end of a project like this.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.

Project Description

The “Game Frame” is a portable gaming device designed for those who want something more than tapping a screen or keyboard in their gaming experience. This device will be a stand alone gaming board capable of chess that can be played with two players against each-other, or one player against an AI. The Game Frame is intended to be enjoyable on-the-go. This machine utilizes an 8x8 grid of buttons that displays the current status of the game, and also allows the user to make moves by pressing them. The “Game Frame” will have the versatility for additional games to enhance the user experience. It will only need to be implemented via software configuration.

Project Challenges

For this project, each of the group members will take on some new roles we have never performed before. Each time we hope to implement a new feature we will have to educate ourselves and do research first before we can start making or tweaking it. This will make development much slower than a typical project where we would be more likely working with prior knowledge and experience that just compounds with some learning and research. We all have personal lives, school, and jobs to be involved with as well, and do not have an uninterrupted eight hours a day to devote solely to this project as we would with one in a career. Unique schedules that do not always line up also creates a need for some kind of organizational scheme. The scheme has to allow us to keep working on a schedule that works for each of us with the short time that aligns being allotted for important meetings and briefings.

This project began towards the end, or arguably the middle, of a pandemic. Working in such a way creates a list of additional challenges to overcome. Scarcity of resources has hit most sectors of global trade, and this will likely last the duration of the development of this project. Although distribution of all sorts is of concern, a very relevant-to-mention specificity would be the known semiconductor shortage. Higher prices are something we will have to cope with, as global infrastructure has taken a massive hit in the past year and a half. In the beginning phases of this project, we are confined to remote contact only, however this is the challenge that is most likely to be resolved as the world slowly reopens. Most of the discussion of the future and implementation of this project have had to take place over VoIP, leading to a lack of interpersonal communication.

Requirement Specifications

Hard requirements

Requirement Type	*Name*	*Description*	*Value*
Power	Battery Life	As it is portable, the design will require a sufficient power supply unit. 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.	2-4 hours minimum
Physical	Dimensions	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.	1-foot width or length maximum
	Weight	The weight should not exceed 4 lbs so as to not weigh more than a laptop with similar dimensions	4-lbs maximum
Monetary	Unit Cost	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.	$400 per unit maximum
Software	Chess	The device should display the ability to play through a game of standard chess with 1 or 2 players.	Playable chess Both 1 and 2-player functionality

Table 1 (Hard Requirements): This lists our mandatory requirements influenced by customer and group member desires.

Soft requirements

Requirement Type	*Name*	*Description*	*Value*
Physical	Dimension	A goal of keeping the device within 2 inches deep should help for our goal weight.	2-inches
Cost	Unit Cost	A soft requirement is to make it within the cost of more expensive portable chess games which can go upwards of $300.	$300 per unit maximum

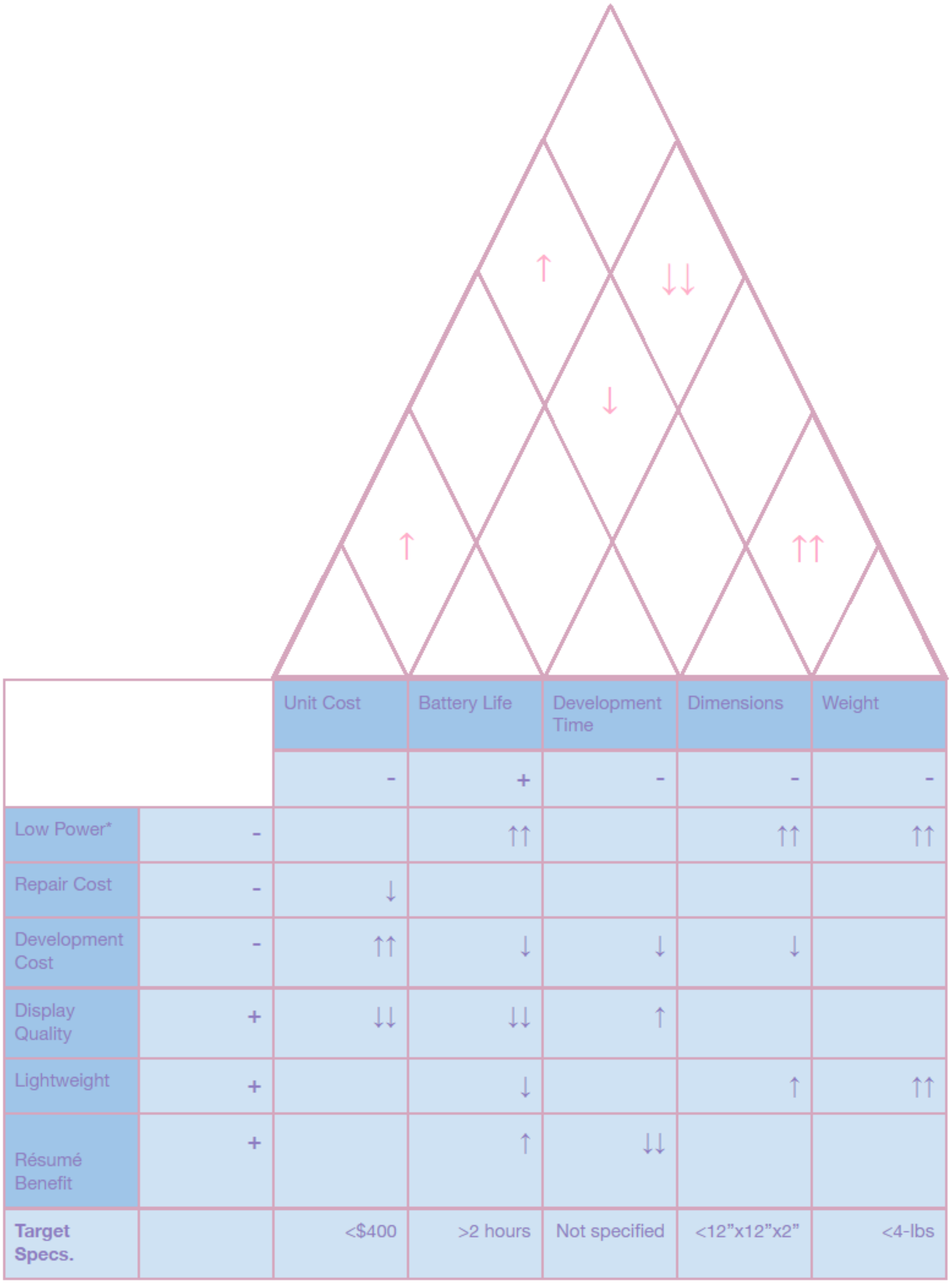
Table 2 (Soft Requirements): A table of our desired requirements that could represent stretch goals.

Constraints

Constraint Type	*Name*	*Description*	*Cause*
Power	Battery	The device must be able to operate off of sustained battery life.	The device is portable
	Power Port	There needs to be a port that allows charging via a USB power source.	The device needs to be rechargeable
Physical	Buttons	There should be 64 physical buttons or partitions.	The device must play chess
Software	Game Library	Games solutions that require a GPU cannot be implemented.	Computational power of hardware
Cost	Budget	The project has a budget limited by the expenditure capacity of our group members.	We are not sponsored

Table 3 (Constraints): A collection of constraints imposed upon is by outside factors or design decisions.

House of Quality



		Unit Cost	Battery Life	Development Time	Dimensions	Weight
		-	+	-	-	-
Low Power*	-		↑↑		↑↑	↑↑
Repair Cost	-	↓				
Development Cost	-	↑↑	↓	↓	↓	
Display Quality	+	↓↓	↓↓	↑		
Lightweight	+		↓		↑	↑↑
Résumé Benefit	+		↑	↓↓		
Target Specs.		<$400	>2 hours	Not specified	<12”x12”x2”	<4-lbs

<u>Figure 1 (House of Quality)</u>: A QFD of our initial design
**It should be noted that Low Power is using “-” to denote that* less *power consumption is desirable. That is to say, the device would be* more *low power.*

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 a jubeat cabinet and example:
Top-down view of gameplay

Image credited: Apetc, used under Creative Commons Attribution 3.0 Unported

Block Diagrams

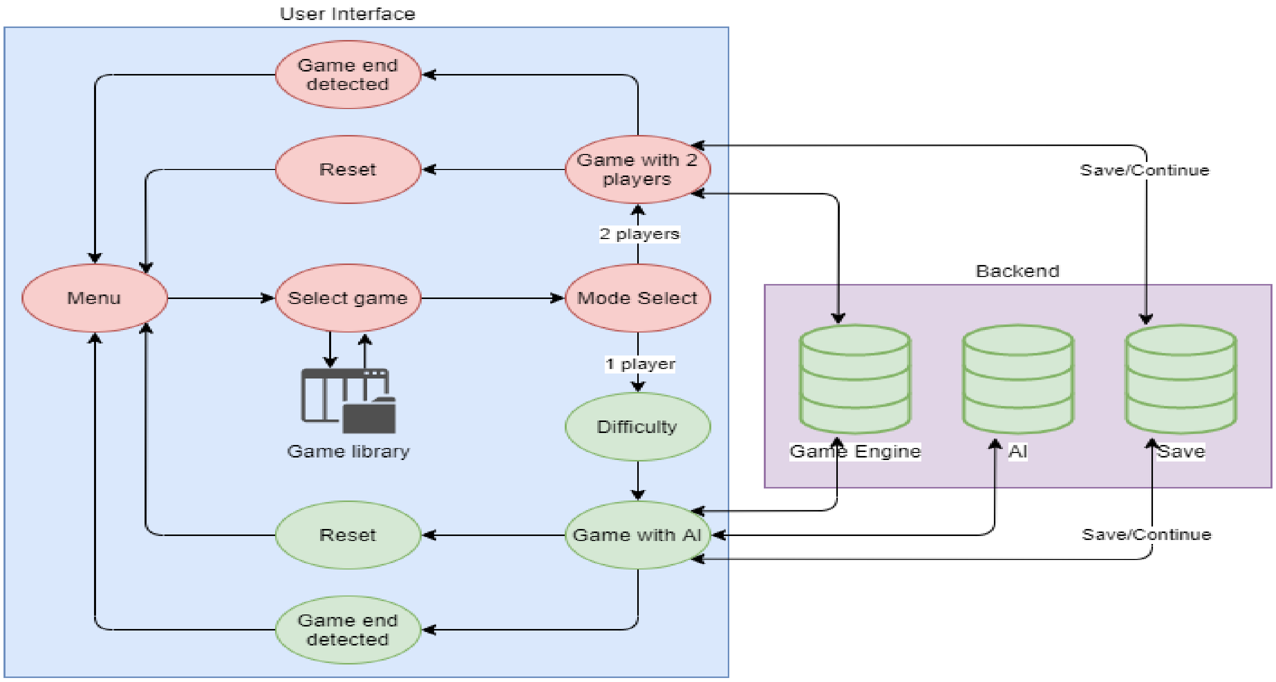


Figure 2 (Software Block Diagram): The organizational structure of our software design

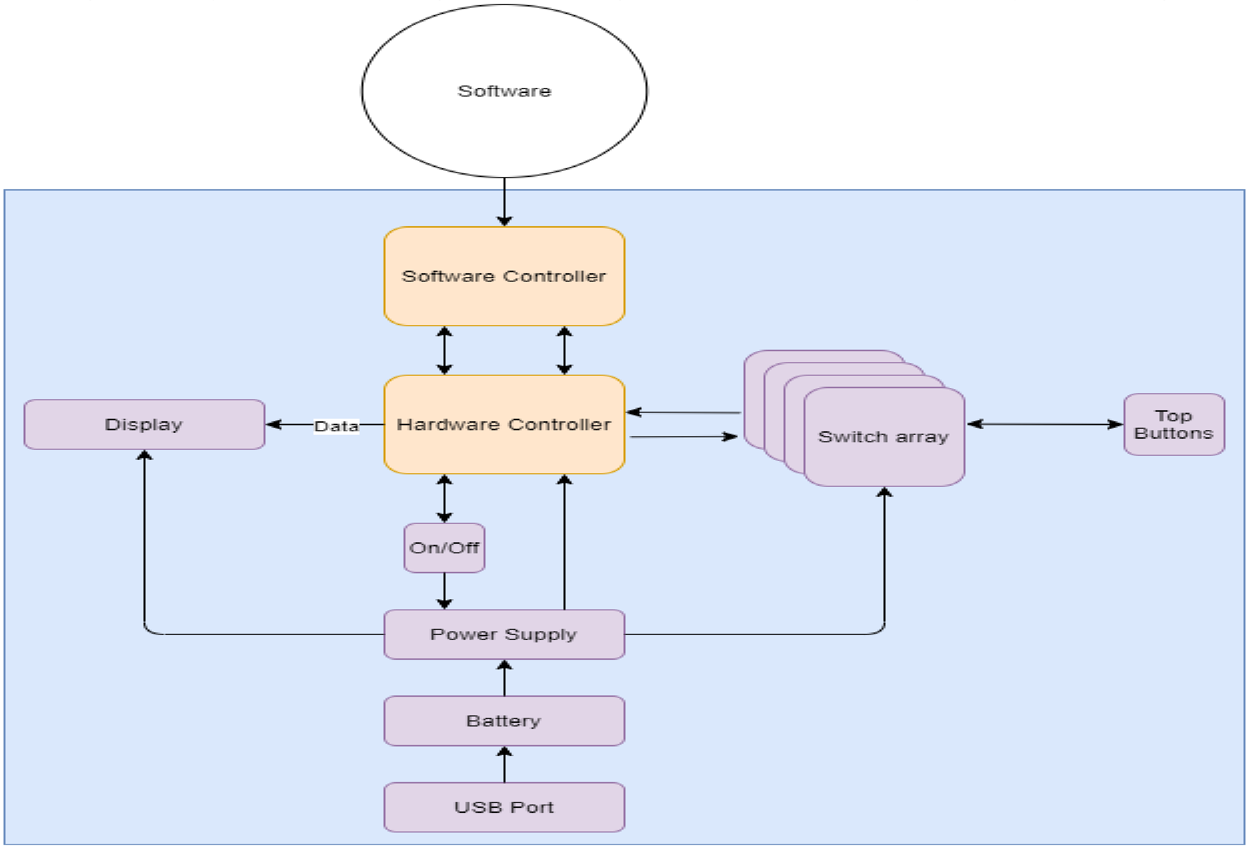


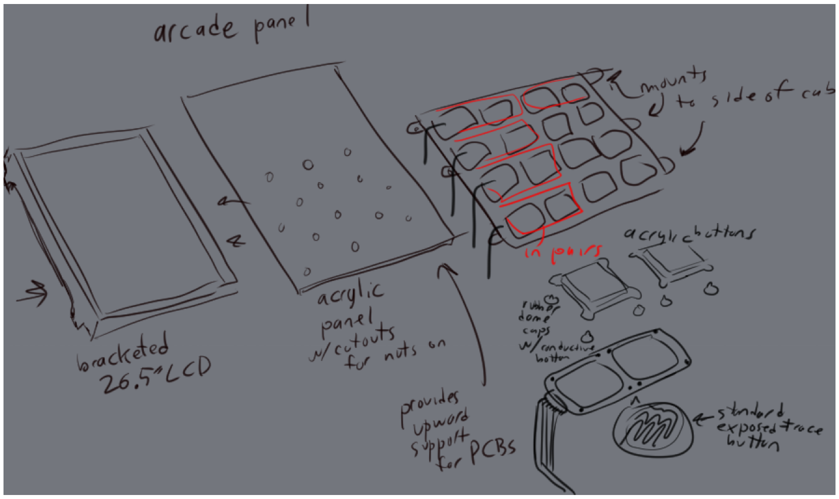
Figure 3 (Software Block Diagram): The organizational structure of our hardware design

Contribution Legend

Red	*Israel*
Green	*Levi*
Purple	*Frank*
Orange	*Allen*

**It is likely that everyone will have some hand in each part of the project, but this denotes who will be primarily focusing on each task.*

Prototyping


arcade panel
mounts to side of cab
bracketed 26.5" LCD
acrylic panel w/ cutouts for nuts on
in pairs
acrylic buttons
rubber dome caps w/ conductive bottom
provides upward support for PCBs
standard exposed trace button

<u>Figure 4 (Prototype Sketch)</u>: A rough diagram we were given

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 (Prototype Figure 1).

In order to add more clarity to parts of the diagram a description of the components might help. Of particular note is the PCB and the acrylic panel. The bottom right shows an example of jubeat's PCB. The machine uses eight different 2-button PCBs. Due to the large holes and thin PCB, a flimsiness exists. The acrylic back panel helps to keep the LCD screen clean of debris of course, but it also functions to add support to the PCB. One other thing that is not clear just from the diagram is that the buttons have a specific diagonal bevel in the corners. This causes a better slide and feel when the button comes back up. As of the moment, we do not have a cost-efficient way to replicate proprietary acrylic, but this might also not be needed for chess to feel comfortable nor might it be needed for our smaller button size.

Budget and Financing

Item	*Cost*	*Source*	*Description*
3D-printed Housing	$20 (about one roll of ABS or PETG)	Amazon	ABS Filament
7-in LCD Display	$30-$60	Digikey	
Acrylic Squares/Buttons	$3 - $ 28 (depending on thickness)	US Plastic	
Acrylic mounting panel	$3 - $28 (depending on thickness)	See above	
PCB	$0.50 - $300	Gerber Labs	
Snap Action Switches	$64 - $192	TE Connectivity	Estimate is for many switches
Arcade-style Buttons	$2.30	Northeast Coast Customs	Only need 1-2
Software Controller	$35	Raspberry Pi	Raspi 4 Model B 2 GB
Hardware Controller	$24	pjrc	
Battery	$13	Amazon	12V (1 AH)
Screws	$14	Ace Hardware	100 count
Total	**$208.8 - $716.3**		

Table 4 (Estimated Unit Cost): A tabulation and estimate total of the various parts to make up one unit

Item	*Cost*	*Source*	*Description*
No Housing	$0		
LCD Display	$26+$12 = $38	eBay Amazon	
Loose Wire	$24	Bulk Wire	
Tactile Switches	$0.18	Digikey	64 minimum needed, 256 preferred
Software Controller	$35	Raspberry Pi 4 Mobel B 2GB	
Hardware Controller	$24	pjrc	pjrc
Power source	$13	Amazon	12V (1 AH)
Total	**$145.52 - $180.08**		

Table 5 (Estimated Prototyping Cost): This is like the prior estimated unit cost but with the goal of using cheaper materials or excluding others to prototype.

Item	*Cost + Shipping*	*Arrival Status*	*Purchaser*
Total			

Table 6 (Billing Materials): We have not yet started to spend on our project so this table is currently a placeholder. In the future it will denote the actual expenditures of our project.

Switches

Name	*Cost*	*Source*
Tactile Switch	$0.18	Digikey
Snap Action Switch	$2.91	Digikey
Silicon Push Buttons	$0.001 - $0.19	Alibaba eBay
Arcade-style Buttons	$2.30	Northeast Coast Customs

Controllers

Name	*Cost*	*Source*
Raspi 4	$35	Raspberry Pi
Raspi Pico	$4	Raspberry Pi
RP2040	$1	CanaKit
Teensy	$24	pjrc
Pyboard	£28 ($33.5)	MicroPython
Nvidia Jetson Nano	$99	Nvidia

Displays

Name	*Cost*	*Source*
Single LCD	$53	Digikey
Microcontroller/ Display Separate	$26+$12 = $38	eBay Amazon
64 Individual Displays	Not Yet Available	Suzohapp

Housing

Name	*Cost*	*Source*
None	$0	
Cardboard	$0.24	Packaging Prices
3d-Printed Housing	$20	Amazon
Crafted Housing	Unknown	
Plastic Enclosure	$15 (+$4 to include screws)	polycase

Batteries (Placeholder table)

Name	*Cost*	*Source*

Table 7-11 (Research Cataloging): Costs of researched parts catalogued in the budget section for ease of use. Descriptions are excluded and explained in research section of this document.

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

Table 13 (Project Decision): Project ideas taken into consideration

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.

Initial milestones

Milestone	*Milestone Type*	*Due Date*	*Details*
Senior Design I			
Functional input/output	Hardware	07/18/21	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	07/18/21	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	08/06/21	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	09/19/21	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 to hardware integration	Hardware -Software	10/10/21	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	10/24/21	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	11/07/21	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	11/30/21	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.

Table 12 (Milestones): A general outline of our project development milestones throughout the two coming semesters

Alternate considerations

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.

Parts research, options, and tradeoffs

Switches

In order for the buttons to even operate we need switches. There are several options we have looked into thus far. At the moment, the plan is to utilize four switches for each of the 64 buttons - one for each corner. We might be able to implement the device with fewer as the project and its design unfolds.

Tactile Switches:

Cost* Datasheet

These small switches are around 18-25¢ and are cheap in cost compared to other options making them good for prototyping. Because they are the cheapest, they might also lead to a device with the least responsive feedback and being the cheapest feel amongst the choices. These have a lower voltage rating and minimum current than the next option that we'll go over, making them better for a portable device. The datasheet graciously provides an expected durability of 100,000 uses so that we know its lifespan and can further estimate potential repair costs down the road. Though repair costs are not an important decision-making factor for our project, when referring to the cheap cost of these switches, it would also have a positive benefit to lowering repair costs.

Snap Action Switches:

Cost* Datasheet

Snap action switches are opposite of the previous option in many ways. It is by far more expensive, but has potential for the best tactile response and feel when used concerning purchasable switch components. The DIY video of someone making a jubeat machine used these switches. A huge problem is that it requires much more voltage for operation. The amperage is still manageable, however it is higher. Higher amperage demand would mean that the battery could not last as long. The datasheet has information pertaining to running the device on lower loads. The tradeoff might help the battery life positively while negatively impacting the durability of the switches.

PCB Trace Switches:

Cost*

One of the best options is probably implementing electrical contact buttons with PCB trace membranes for the switch. Jubeat uses this method of implementation. PCB prototyping is the most expensive prototyping option leading to an obvious downside. For unit cost this would probably be the cheapest to produce since it is part of the PCB cost.

Arcade buttons:

Cost

It has been discussed that auxiliary buttons on the side might be useful in our design. They could operate not only as UI or power control, but could also be used by various games down the road. They might not be necessary and might be an additional feature. It details an expected 10,000,000 uses on the kind we looked at.

**It should be noted that the examples in the links do not contain bulk costs. Since we would have a potential 256 switches, bulk costs are actually a viable consideration. Bulk Cost Example for Conductive Buttons*

Controllers

There are a large variety of controllers to choose from. The plan for now uses two. A microcontroller will work for managing the hardware and the array of switches while also interfacing with another controller or computer which manages the software. Another viability is to use a microcontroller or other controller with two processors on it. The benefit of the separate controllers is the ability to facilitate simultaneous progress and work on the hardware and software side of things.

Raspberry Pi:

Raspberry Pi of course is known for their variety of products and options to use as controllers. The obvious downside of one of these is that they are not as impressive for inclusion on a resume compared to other controllers.

Raspi 4

A Raspberry Pi 4 is monetarily more expensive than other controllers that might have less powerful specs in them. The Raspberry Pi 4, however, has specs powerful enough to implement Python as a default language. The power is not free however and might affect our desired battery life. For the time being, it seems the safest choice for handling the software until we know more.

Raspi Pico

The Pico is an appealing incredibly cost efficient option at only $4. The size and lower specs make it desirable as a microcontroller, but if it is sufficient to handle our software, it seems the superior choice.

RP 2040

It should be noted that the microcontroller chip can be bought by itself and designed around. This is an even cheaper option and allows our computer engineers the potential to develop their architectural skills.

Teensy:

Feature Table

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 originally decided that we wanted a controller we would be sure could handle both the buttons and the game instead. Now that we're considering two devices this has changed, however. The size, lower power requirements, and cost make this a viable and desirable option as a microcontroller for managing the hardware.

Pyboard:

Feature Table

This is the official board of MicroPython. The board is not far in price from a Raspi at about $28, but it has considerably less specs. There are cheaper microcontroller options for handling our hardware, but this could be a consideration for managing software. MicroPython might be a good high level language to work with and the power consumption might be more manageable than the Raspberry Pi.

Nvidia Jetson Nano:

Feature Table

Pardon the informal language, but this thing is an absolute beast. This microcontroller touts the highest costs and specs we would even consider for our project. This could do and handle everything, including putting a strain on our goal unit cost. A positive aspect is the specific skill of knowing how to code for this looks fantastic to employers. The other most notable factor is how this would raise the ceiling on our game library having a GPU bound constraint. This product might also allow us to forgo a microcontroller for a screen panel and handle that itself. This really would be an all-in-one, which would make it a more manageable cost, but less divisible project.

Programming Languages

The language we use will mostly be learned and determined based on the controllers that we find to be most appropriate. There were some considerations we have regarding a couple languages though in terms of hopes and preference. Preference of which obviously will influence our decision for controllers alongside the tradeoffs of the controllers.. Whatever the high level language a controller has, the architecture specific assembly language it has is bound to be necessary to learn as well.

Python:

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. Python is desirable for us 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 to facilitate their actualization.

Micropython:

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, Python has more appeal for our group. If MicroPython proves compatible and it is feasible within reason to port our work to a more desirable controller, then we might do so as the progress unfolds.

C:

This language is one that everyone on the project is familiarized with and it has unparalleled convenience with access to the low level through memory manipulation. Transferring to and from the low level language also becomes easier with assembly directives.

Assembly:

Although it has already been mentioned that we will probably have some amount of assembly for any of our choices, it should be noted that which assembly we work with can be a valuable asset for job applications. The other thing to bring up is that assembly is very close to the hardware. As such, we expect that using assembly will definitely be more prominent for our hardware controller, while higher level languages will probably remain most preferable for the software device.

Displays

Rather than a single LCD panel with the 64 different buttons we considered using 64 partitioned displays for each button instead. This leads to two primary options for the display - partitioned or unpartitioned.

Single LCD Panel:

Datasheet

It might prove an easier feat to implement the partitioning of the segments on the software side, so a single panel might be preferred. Also, due to the cost of the individual panel being considerably lower than buying 64 units of the various segmented options we prefer a single panel. Both of our inspiration sources utilize one LCD. The handheld and portable chess devices on the market also have an individual LCD screen albeit smaller. One screen also allows for more versatility on the software side, especially if creating additional games. The operating voltages and current for this seem reasonable for a portable device, but there is concern with the additional amount required for the backlight.

Controller Screen

Another less convenient but much cheaper solution to implementing one screen is buying a microcontroller designed for display control and buying the screen separately.

Separated Display Options:

Example LCD buttons Datasheet

Aside from complicating the software and hardware design, especially at the interfacing level, quality panels for segmentation boast a significant cost increase due to needing 64 of them. The prices of the particular LCD buttons provided are unavailable unless contacted, so an accurate price estimate of these are not yet known. Suzohapp as a manufacturer, while providing quality parts and giving good ideas for components that might fit in our device, tend to have costly products. Using 64 of these drawing .8A each is quite worrisome for portable device design too. A positive is that none of the housing or frame space between buttons will be wasted pixels on screen. The cost of repair and maintenance might also be better should one of these fail compared with an approximate 10” LCD display.

Housing

Not many options came to us in regards to housing our production. We are fortunate to be in an age of 3D-printing, so it seemed an obvious choice for us. Still, we want to leave room for other considerations should they arise.

None/Cardboard:

We obviously have no desire to create a device housed in cardboard or without housing. The reason this option is mentioned is simply for early prototyping. The cost is as minimal as it gets. At some point in the process we might even use this option as a way to verify dimensions.

3D-printed housing:

3D-printers are able to create good prototype housing while also being reasonably accessible. The monetary cost is reasonable, but prototyping and mistakes definitely cost more time. CAD knowledge is required to create something with this method, but this is seen as a boon for the project, since this kind of experience is nice for our résumés.

Crafted housing:

Similar to buying acrylic for mounting the PCB or for making buttons, we could fashion our own housing out of some plastic (pre-cut or cut ourselves) and screw them together ourselves. This could be cheaper, but also requires some handiwork and might end up looking less professional than a 3D-printed alternative, which should be comparable in price. Professionalism might not be as important in prototyping stages, but at that rate nothing or cardboard is cheaper.

Plastic Enclosures:

Some companies, like Polycase, sell premade plastic housing. These can be quite affordable, but they are less customizable. Instead of using these, it might be better to just 3D-print a more custom tailored one ourselves.

Batteries

One of the most important factors for making our device portable will be the battery. Because the device is portable we have leaned heavily towards recharging as an option almost to the point that it's our decision. The downside is that cost is on the development and fabrication sides, rather than offloading it to the consumer's wallet. Disposable batteries also have the benefit of allowing the customer more playtime on the go without the need to recharge. That said, despite disposable batteries being better for monetary design, it is worse for the consumer's pockets and for the environment. We would not be knights if we did not have this bare minimum consideration for the environment in our design. That leads us to the most common materials for rechargeable batteries as our first consideration. Of the common materials, there are four:

Lead-Acid, *Nickel-Cadmium*, *Nickel-Metal Hydride*, *Lithium-Ion*

Works Cited and Permissions

This is a placeholder section. All sources are currently in-line hyperlinks. No permission has been requested or needed yet.