Handheld Color Sliding Game - Prismatic Paths

Joshua Bell, Eric Espinosa, Linus Fountain

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — A one-of-a-kind handheld game device dubbed Prismatic Paths, a handheld game device merging retro gaming with modern technology. Inspired by a shared background in gaming, the compact, user-friendly design aims to evoke nostalgia while offering an enjoyable experience. By integrating modern-day microcontrollers, displays, and haptic feedback, it provides a fresh take on traditional gaming devices. Prismatic Paths serves as both a creative pursuit and a platform for skill development in the ever-evolving gaming industry. This project seeks to contribute to consumer joy and engage diverse audiences, reflecting our commitment to innovation and excellence.**

I. INTRODUCTION

Our group's project embodies a shared commitment to bringing joy to consumers while igniting our passion rooted in our tech-driven gaming upbringing. Inspired by our childhood experiences, we aim to craft a unique and compact game tailored for modern console enthusiasts, prioritizing ease of use and entertainment. Our endeavor extends beyond creativity; it serves as a platform for skill refinement and acquisition within a constantly evolving field, driven by our desire to innovate and contribute meaningfully to an industry that has shaped our lives.

As for the game rules, players are presented with 20 random arrows and directions of different colors on an LCD, with two main colors on either side of the display indicating the current target. To progress, users must manipulate the toggle past sections matching the target color and direction indicated by arrows. Completion of a section prompts the central color to change, and as stages advance, time constraints increase. The display shows remaining time and scores, with an option for players to view high scores from the main menu. Additionally, the game allows users to choose between single or dual toggle gameplay.

II. HARDWARE COMPONENTS

The system is most effectively illustrated by breaking it down into its individual components. These components were either bought or designed by the team, and they are interconnected to form the end product. This section offers a semi-technical overview of each of these components.

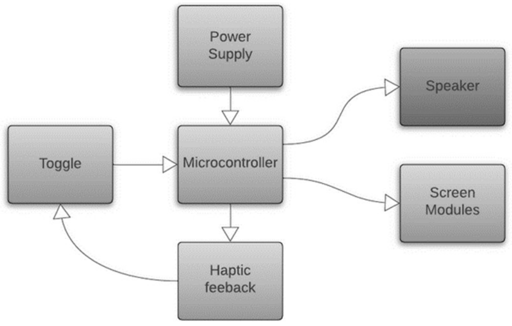


Fig.1. Hardware Block Diagram.

A. Microcontroller

The ATMega328P microcontroller is the engine of our device. This microcontroller was chosen initially due to its low costs and easy access since each member of our group owns an Arduino Uno, which houses the ATMega328P. Looking more into the device, we discovered features that aligned with our project in mind seamlessly. It operates at 16MHz, which is necessary for the frame rate, response time of inputs, and game logic. It has 32 KB of flash memory for storing program code, 2 KB of SRAM for data storage, and 1 KB of EEPROM so it can retain data when power is removed, which is essential to our project so the high score information is not erased. Since the Atmega328P is a part of the Arduino family, it has amazing support in the Arduino IDE.

B. Display

In order to bring our project to life, it becomes imperative for us to select an appropriate display that will serve as the canvas for showcasing the main gameplay of our game. The display for our project is the Adafruit 4.3 inch TFT display. This LCD display employs thin-film transistors to elevate the image quality and responsiveness of the screen. It has a pixel density of 480 x 272, an LED backlight, a resistive touchscreen overlay, and is refreshed at a constant 60 Hz. The group decided that this display offered the best combination of quality, compatibility, function and the best price point compared to the other types of displays that were in consideration. The TFT display was far above the other displays in consideration in terms of quality of display.

C. Display Driver Board

Since it is extremely rare for our microcontroller to be able to support the display, we needed to pair it with the Adafruit RA8875 driver board. This is a chip that can handle the video RAM and timing requirements all in the background. The Arduino does not consist of enough pins to connect the screen, so the board allows direct connection onto the development board as there is no way to connect our display. Our display will require a 60Hz refresh and 4MHz pixel clock that is not supported by the Arduino, so this driver board is a necessity for the functionality of our device in connection to the display. The way that this board works is it allows the screen's ribbon cable to be directly connected to the input port. Then on the RA8875 board there is a driver chip and also 768 KB of RAM that will allow the board to buffer the display.

D. Toggles

This device uses two analog COM-0932 toggles on either side of the display screen. These toggles are the key component that the user will directly interact with by using them to navigate throughout the menu screens and select different options by pressing in on the right hand side toggle. They are also the core device that controls the movement during the game. These specific toggles were chosen because of their inexpensive price point, comfortable and familiar design, as well as their ability to track smaller movements compared to digital toggles which only have a one or zero output.

E. Speaker

In order to provide an audio aspect to the device there is one Soberton Inc SP-3605 speaker inside of the case for another form of feedback that the user will experience. The purpose of this speaker is to give a countdown into the game, a countdown during the game so the user will know when the round is ending and then finally to have a tune play when the user achieves a high score. With this being such a small task for the speaker there was no need to have an overly powerful or large speaker for this game. Therefore, the decision to choose this speaker was due to its lightweight, low power consumption and low price point.

F. Haptic Feedback

The haptic feedback we used for our project is the Eccentric Rotating Mass(ERM) Tatoko vibration motor. It is a 3V DC motor, with an unbalanced mass connected to the shaft. Different vibration frequency intensities are created by spinning the offset mass at varying speeds. As the weight spins it pulls the device in every direction hundreds of times a second, causing the device to vibrate. The motor utilizes overdriving, it minimizes the startup time of the ERM. It utilizes 3V to startup the device until the weight is spinning at the desired amplitude and frequency, then the voltage is dropped to the minimum voltage of 1.5V, in order to sustain the vibration. For the ERM to stop, it has to be reverse driven, which switches the voltage polarity in order to change the direction the weight is spinning to slow down the device. The haptic we chose rotates at 12,000 RPM.

G. Battery

The battery is the second most important component for our project. Without a good battery, our device will not be able to last a very long time. For our project, we are using the Wavypo 9V lithium-ion batteries. These rechargeable batteries have longer battery life than the more common disposable alkaline batteries. They come with a pre-installed USB-C port to recharge the batteries, allowing us to recharge, reuse, and retest our project without waste. They are rated at 1300 mAh, at this rating our device lasts approximately 2 hours.

H. Case

The final version of the case consists of six 3D printed pieces, 20 threaded inserts, 14 nylon lock nuts and 34 screws. The purpose of the top piece is to hold the LCD in place against the cover. The cover itself has a spot that holds the LCD and allows for the ribbon cable to run down into the case, as well as grills for audio from the speaker to come out and two holes for the toggles. The next piece down is the one that holds most of the case together; the two covers both attach to it, as well as the piece that holds the LCD driver board and the battery. In addition to holding the case together, this piece is where the main PCB mounts, as well as where the haptic feedback modules are mounted. The fourth piece from the top is where the LCD driver board is attached, as well as where the battery and access to programming pins are aligned with the hole on the back cover. The second to last piece from the top is the back cover, we decided it best to have the battery protruding in its own compartment rather than increase the entire inside area of the case. The piece at the bottom is what the user unscrews to access the battery and programming pins.

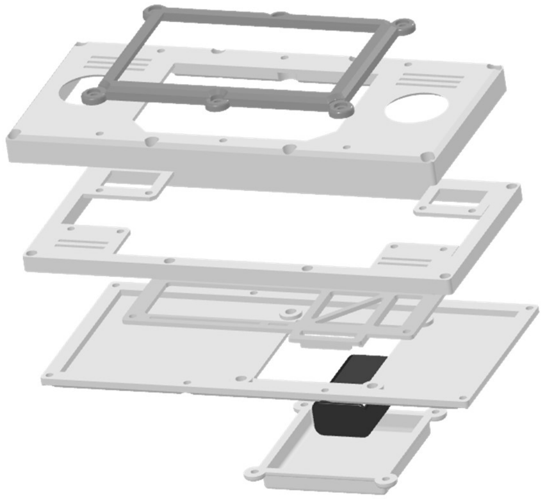


Fig.2. 3D Rendering of final case design.

III. HARDWARE DETAIL

This section will review how all of our hardware components from Section II connect to the circuit and device as a whole. It will also go over any circuit components that were not mentioned in the hardware components section.

A. Buck Converter

Our device is using the TPS563252DRL step down buck converter to convert our 9V input to a 5V input to power on our ATmega328P and the rest of the device. The TPS563252DRL operates from 3V to 17V input voltage and 0.6 to 10V output voltage, so this is within our usage range. Using the Texas Instruments Software Webench, it was able to provide us a circuit to use for our buck converter. Referencing the data sheet and double checking some of the math, we ensured that the values provided to us were correct and within the proper range of operation for the converter. Our power supply connects to a diode that protects the power from current flowing into it, which then connects to the buck converter circuit.

The circuit then starts off with two capacitors parallel to one another, Cin and Cinx. For these capacitor values, the datasheet recommended one decoupling capacitor, one bulk capacitor, and one frequency filtering capacitor. The decoupling value is recommended to be over 10uF and the frequency filtering to be 0.1uF or 100nF. So for the bulk capacitor C1, we chose 10uF. For the decoupling capacitor, it was also 10uF, and for our filtering capacitor Cinx, we chose 100nF.

Following the input capacitors, are the voltage divider resistors. These resistors are used to lower the input voltage to the appropriate voltage operating range for the converter. This ensures that we don't supply a higher or lower voltage outside the operating range and can power the converter on.

Coming out the output of the converter, starting with the Power Good(PG) channel, is a 10Kohm pullup resistor that is used to monitor the output.

Following PG is the switch(SW) channel. It connects to a 2.2 uH inductor L1, which then connects to a 33pF capacitor Cff. The datasheet did not provide any information regarding these components. They did provide the recommended values for them depending on the target output voltage. After the Cff is another voltage divider with the resistors Rfbt and Rfbb. This connects to the Feedback(FB) channel on the converter. Using Eq. 1, we can double check our resistor values to the output to ensure they are correct.

$$V_{OUT} = 0.6 \times \left(1 + \frac{R_4}{R_5}\right). \qquad (1)$$

Plugging in the values of R4=220k Ohm and R5=30k Ohm, we get our target output voltage of 5V.

Finally, the last components of the converter are the parallel output capacitors Cout and Coutx. For these capacitors, we followed the recommended datasheet values for Cout when voltage output is at 5V. They are connected in parallel to the output load current.

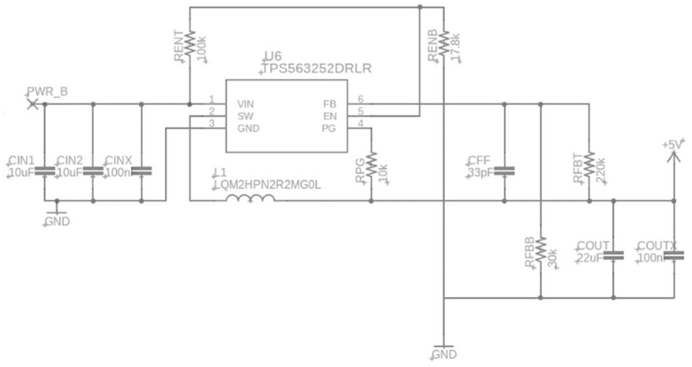


Fig.3. Buck Converter Schematic.

B. Haptic and Speaker

The haptic feedback modules each use one digital pin. One of the schematic requirements of the haptic feedback modules is that in order to control them with the digital I/O pins an NPN transistor had to be used. This is because the haptic feedback modules are motors, and require a higher current than is recommended to be sent through the ATmega328P's I/O pins. So rather than connecting straight to the motor, the pin that controls each module is connected to a 10kOhm resistor and the base of the transistor.

Another requirement of the haptic feedback modules is the diodes D3 and D4. Motors are inductive loads, and so a diode is used to safely let the electromagnetic field of the motor collapse. The speakers are simple, one wire connects to a digital I/O pin and the other wire connects to ground.

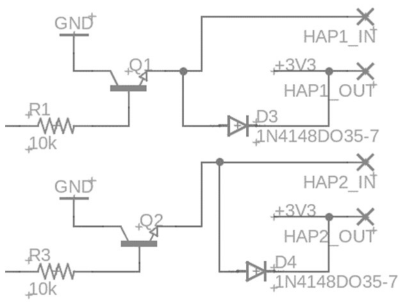


Fig.4. Haptic and Speaker Schematic.

C. Toggle

The toggles are the main element on our board that requires analog pins. Each toggle requires two analog inputs, one for the x axis and one for the y axis. The reason each axis needs an analog pin is because the axis measurements are made using mechanical potentiometers. In addition to the two analog pins, each toggle requires one digital pin for the button.

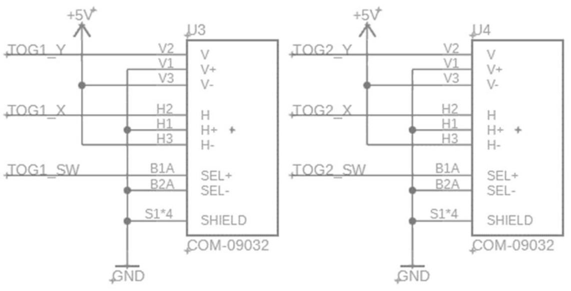

+5V
U3
TOG1_Y
TOG1_X
TOG1_SW
V2 V1 V3 H2 H1 H3 B1A B2A S1*4
V V+ V- H H+ H- SEL+ SEL- SHIELD
COM-09032
GND
+5V
U4
TOG2_Y
TOG2_X
TOG2_SW
V2 V1 V3 H2 H1 H3 B1A B2A S1*4
V V+ V- H H+ H- SEL+ SEL- SHIELD
COM-09032
GND

Fig.5. Toggles Schematic.

D. Crystal Clock

The crystal clock we used for our design was the standard crystal clock that the Arduino came with.

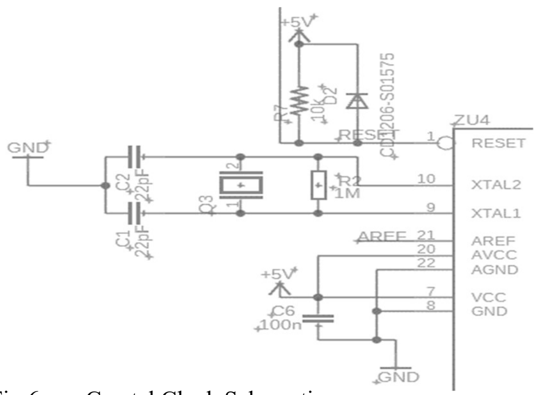


Fig.6. Crystal Clock Schematic.

E. PCB Design

The biggest consideration in the fabrication of the PCB stems from the Senior Design PCB requirements. In order to meet these requirements our toggles must be attached to the PCB that we design, as opposed to being on a breakout board and wired. This has a significant effect on the size requirement of our PCB, as we have one toggle on each side of the display. This means that our PCB must be about six or more inches long to reach. Another choice we've made is to use the ATmega328P-PU variant of the microcontroller. This allows us to take the chip out of its socket and program it by plugging it into an Arduino Uno. This simplifies the schematic and PCB by allowing us to not worry about any sort of USB to program through. Over the course of the project we determined we wanted to program over the programming pins due to concerns with harming the microcontroller by repeatedly removing it from the slot, so access points were added that allowed the programming pins to be soldered to and accessed from the back of the device. Another consideration that has to do with the toggles is ruggedness. For this we have included four mounting holes for each toggle, so that there is no give when the user is interacting with the toggles.

IV. SOFTWARE

In order to develop a full understanding of the intricate details about the software side of this project the following sections will be diving into our project's architecture. The following software diagrams have served as a framework for this project as well as a guide that we have been able to depend on. These diagrams encompass every aspect of the software that has been used, both from the user's standpoint as well as from the device standpoint. Due to the complexity of the programming for this project not everything will be labeled in the following charts as placing every item into the charts may be too tedious and excessive and pull away from the original intent of these charts which is to allow the reader to have a deeper understanding of our goals.

A. Flowchart Diagram

Upon launching our game, the initial software processes follow a systematic sequence, as depicted in the accompanying diagram. The program initializes by loading the designated class responsible for the main menu, incorporating both visual styling elements and navigational logic. User inputs, facilitated through toggles, are tracked to enable seamless interaction. From the main menu, users are presented with several choices: proceeding to gameplay, accessing high scores, or adjusting settings.

Opting to view the high scores entails a series of operations. The device retrieves and organizes high score data for display, potentially sorting it for clarity. Stylistic features are loaded, maximizing efficiency by reusing design elements where applicable. Conversely, selecting gameplay triggers dynamic level generation, ensuring each playthrough offers a fresh experience. Randomizing arrow sequences maintains novelty, while real-time toggle tracking facilitates smooth transitions between screens, enhancing user immersion. Additionally, level-specific timers adjust dynamically, optimizing challenge levels.

For those adjusting game mode, a dedicated screen offers options spanning normal, left hand only or right hand only. Once chosen, users seamlessly return to the main menu, retaining access to all functionalities. Post-game analysis determines progression or prompts a reset, culminating in either a celebratory acknowledgment of high scores or a return to the main menu for subsequent decisions. This systematic approach ensures a fluid user experience, balancing engagement with ease of navigation, thus enhancing overall gameplay satisfaction.

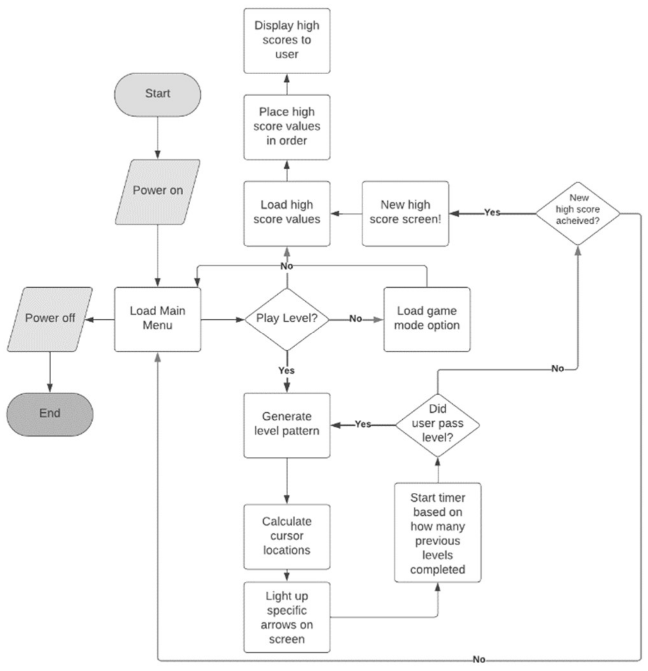


Fig.7. Flow Chart Diagram.

B. Use Case Diagram

The use case diagram provides a visual roadmap illustrating the interaction between the user and the game system, emphasizing user-driven actions while excluding autonomous device movements. User actions encompass a range of activities from toggling controls to navigating menus and gameplay, fostering an immersive experience. Meanwhile, the device handles software tasks like level generation, scorekeeping, menu display, and toggle monitoring, ensuring a smooth operation. While the diagram simplifies complexities for clarity, it maintains comprehensiveness, offering insight into fundamental user-device dynamics without overwhelming detail.

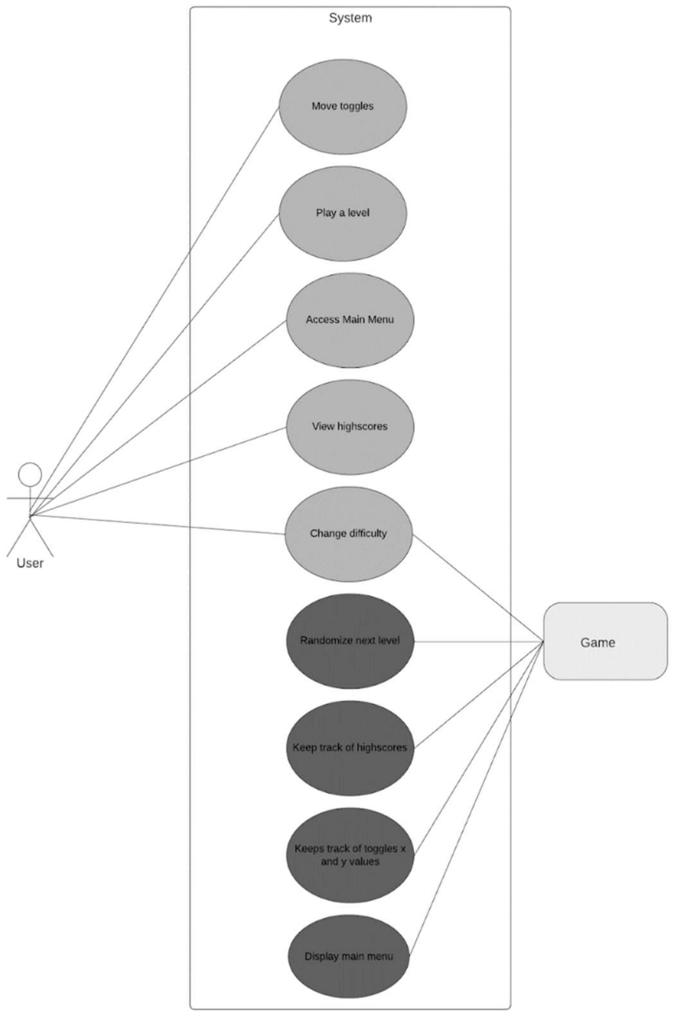


Fig.8. Use Case Diagram.

C. State Diagram

The state diagram below can be used as a guide to visually see the various key states that the device can assume, offering a detailed insight into the user experience. Upon powering on the device, users are welcomed by the home screen, where they are presented with two distinct options: the opportunity to view other previously achieved high scores on the device, change the difficulty of the gameplay or to go straight into gameplay. If the user selects to change the game mode, they will then be presented with a new change game mode state, and the same will occur if the user opts to view previous high scores. After the user is finished with both of these states they will be redirected to the main menu screen. The journey through the levels is intricately tied to the user's performance, creating a dynamic and responsive gaming environment. As users navigate through successfully completed levels, they are rewarded with a seamlessly transitioned experience, encountering new randomly generated levels to maintain the flawless gameplay. This will continue until the user fails to complete a level within the designated time frame. The inevitable conclusion of a failed level leads the user to a game over screen.

Post-game over, the device dynamically evaluates the user's performance by comparing the newly achieved score with the high score stored on the device. If the user has surpassed their previous high score, they will be presented with a high score screen, acknowledging their accomplishment. Conversely, if the score falls short, the user seamlessly returns to the main menu, providing them with an opportunity to regroup and commence a fresh gaming experience. This thoughtful progression ensures a captivating and user-centric gaming journey, where each state serves a purpose in enhancing the overall engagement and satisfaction of the gaming experience.

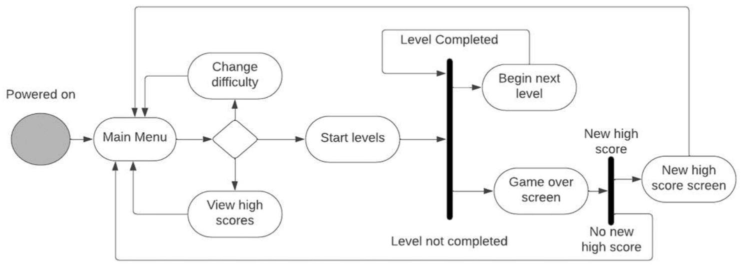


Fig.9. State Diagram.

D. Class Diagram

In fig. 10. it is easy to see that the gameController is acting as a central hub. While in the actual code this is broken down into many different functions such as startGame() and handlMainMenu() this gives a good idea of how the game is operating internally. The system first boots up and it sets up all of the core values such as loading the EEPROM memory which in this device only has the high score data saved there. Conveniently enough this EEPROM memory does not interfere with the memory location that the program is stored on the atmega328p so there are no issues with the high score data being lost.

From the setup the main menu loads and is the central part of the program. At the main menu the user can select to change some settings, view the high scores or play the game. During any of these sections an input register is running behind the scenes to detect if the user attempts to move the toggle in a certain Y direction. When the user selects to begin the game by pushing in on the right toggle many things begin to happen in the software. There is a countdown function that is called that will display a three, two, one countdown as well as play an audio cue and give physical feedback using the haptic. At the time the countdown is complete randomly colored arrows will be displayed. The arrow colors are randomized for every new level using “randomSeed” in Arduino IDE which is first called in the setup function.

During the gameplay the software is keeping track of time, user input, and points scored using separate functions and a looping system for all of them. Once roughly 30 seconds have passed the game will compare the users final score with the high scores that are saved in the EEPROM memory to determine which place they achieved and will prompt them with an appropriate screen before returning to the main menu.

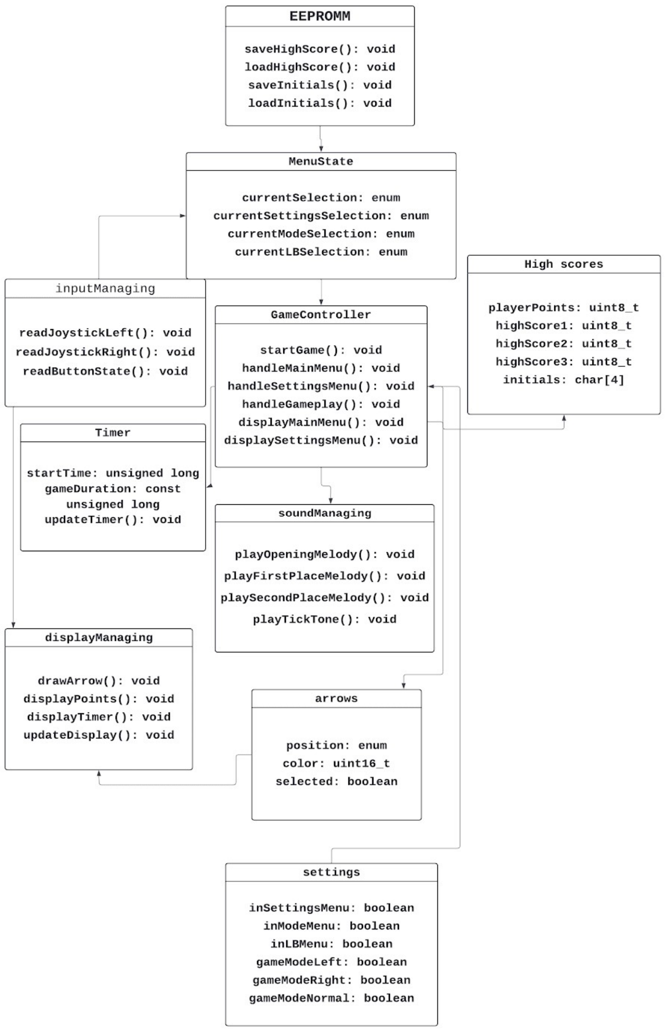


Fig.10. Class Diagram.

E. IDE and Language

This project utilizes the Arduino Integrated Development Environment (IDE) to accomplish its software objectives. The Arduino IDE provides a platform for writing code using a pseudo-C++ language specifically tailored for Arduino boards. Although Arduino employs its own variant of C++, the syntax and structure closely resemble traditional C++, making it familiar to programmers. While other IDEs could have been used for this project, the decision to opt for Arduino IDE was primarily driven by its simplicity and ease of use. The plug-and-play nature of Arduino IDE greatly facilitated the initial stages of the project, allowing for rapid prototyping and development.

F. Memory Limitations

One significant challenge encountered during the development process was memory limitations. As the project involved creating a fairly complex game with multiple game modes and user interface screens, managing memory resources became crucial. The Arduino IDE provided valuable insights by displaying the memory usage for each compiled sketch, enabling the programmer to optimize code efficiency effectively. Despite facing memory constraints, the project team successfully achieved their objectives through innovative approaches such as judicious use of variables and efficient management of screen states.

G. Graphical User interface

The Graphical User Interface or GUI of the game was quite a tricky thing to handle. The RA8875 driver board has many limitations on how many different shapes are possible to make display as well as how many colors. The group was only able to figure out how to get about 8 different colors as well as only basic shapes such as squares, triangles, circles, and ellipses. With that being said the group was pleased with the GUI design considering the limitations. Below are displayed the Main Menu of Prismatic Paths and also a still shot of the gameplay.

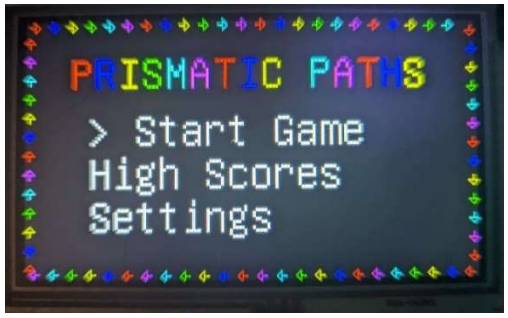


Fig.11. Final Main Menu of Prismatic Paths.

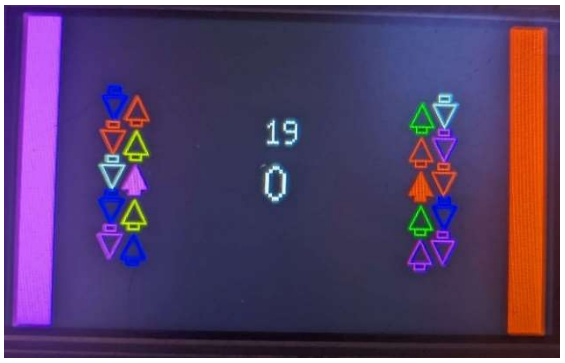


Fig.12. Still shot of gameplay.

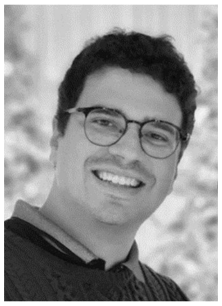
V. CONCLUSION

The goal of this project has been to design, construct, and demonstrate a handheld game system that has hardware tailored specifically to the rules and feel of the game. The design of the game is such that the user will use a toggle to quickly move a cursor through colored slices of a circle in the semi random sequence generated by the game. The game features the option to play with both toggles at the same time, or to do one toggle at a time. There is also a leaderboard that will be kept for each toggle option.

In conclusion, this project required us to research and carefully consider every component in the design and how they will affect the user's experience as well as fit into our vision. We had to determine what could present the user with a visually and tactilely enjoyable experience, to this end we chose a display with a high enough refresh rate and included haptic feedback in the design.

VI. ABOUT US

Joshua Bell is a 23-year-old graduating Computer Engineering student. Josh hopes to work for an established company such as Google, EA, Intel, Apple, etc. He has no preference on the exact specifics of his career as long as he is able to continue to learn.



Eric Espinosa is a 26-year-old electrical engineering student who is currently working at TRC as a power distribution intern. He plans to work there for a few more years in hopes of being able to transfer to a branch in Hawaii or Washington state. His dream is to one day work for a company such as AMD, Nvidia, or Intel.

Linus Fountain is a 25-year-old electrical engineering student. Linus is an aspiring artist that hopes to put his experience and education in engineering towards something that also satisfies his creative drive.

ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Dr.s ChungYong Chan, Arthur Weeks, Lei Wei, Matthew Gerber and Wei Sun; University of Central Florida