Defuse It or Lose It!

Ethan LaFleur, EE, Edward “Matt” Buiser, CSE, Krishna Vijayakumar, CSE

***Abstract* — Modern applications of virtual reality are mainly used to simulate experiences by emerging the user into an interactive virtual environment. Not many things originate in virtual reality that can be brought to the physical world. Rather, these systems are mostly used as simulation tools for real world experiences. However, one industry that makes use of creating completely new and interactive experiences that may be feasible to bring out of virtual reality, is video games. Some of these games are centered around interactive, virtual, handheld objects to progress in the game. Our project creates a physical prototype system of one of these games, “Keep Talking and Nobody Explodes.” The prototype will have limited features of the actual game but will demonstrate key aspects.**

I. Introduction

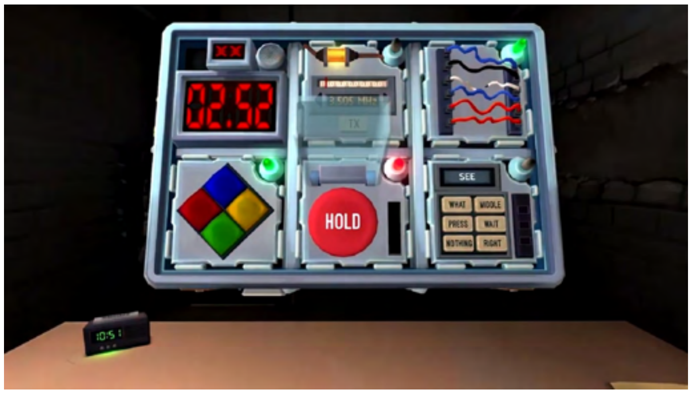


Figure 1: Sample “bomb” from online video game

The virtual reality game, “Keep Talking and Nobody Explodes”, has a fairly simple objective for the two players: complete all the puzzles in less than three tries and before time runs out. The game is played with two players, one who has the VR system and one who has a manual with the instructions on how to defuse the bomb. Since each player can only see their respective item, player one sees the virtual object and player two sees the manual, player two must communicate the instructions for each puzzle to player one. However, since player two does not know the configuration of each puzzle, s(he) must rely on player one to explain the puzzle’s configuration. The rest of the paper explains more on our prototype and how it compares to the virtual reality game.

A. Significance

Virtual reality provides many innovative and creative ideas for the engineering world to explore. We can simulate designs for systems with visual results to further our understanding of how a system may behave in the physical world. This is no exception with the video game industry. With the new advancements in technology, board games of previous generations feel outdated by the creativity and innovation of modern day simulated games on a device. This project seeks to revive the board game industry with much more immersive and elaborate games that can compete with the oversaturated video game industry.

B. Context and Competing Solutions in the Marketplace

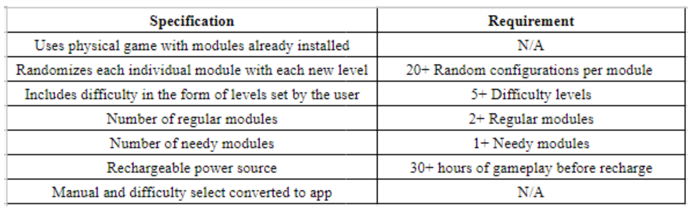
Currently, there are no physical implementations of “Keep Talking and Nobody Explodes” available on the market. However, there have been a few personal hobbyist projects created and shared on the internet since the game’s inception in 2015. These existing solutions have implemented much of the same functionality as the original game except for hot-swappable modules [9]. A project by Reddit user Benoit2600 uses Arduino Mega breakout boards in serial communication with a Raspberry Pi, a graphical user interface as the central hub, and 3D printing for the enclosure [10].

These projects are thoroughly designed solutions to the stated problem. However, they have limited scope for commercial use as they are bulky and expensive. Our design goal is to make our own slightly modified version of the game that is portable and inexpensive.

C. Societal Impacts

As of right now, virtual reality is a ground-breaking technology that has been pushing the boundaries of what we can do. Virtual reality is a great way for people to implement things that we cannot have readily available to us in the physical world. However, since this technology is so new, it is currently not affordable for everyone. By bringing a game like "Keep Talking and Nobody Explodes" to the physical world, we aimed to make this game more accessible to people who cannot afford VR technology. We feel that this game suits the physical world in a family board game type of setting. This game can help build communication and teamwork skills in a fun and entertaining way. We believe that bringing this game to more people will have a positive impact on the people who would play this game.

D. System Requirements and Specifications



Specification	Requirement
Uses physical game with modules already installed	N/A
Randomizes each individual module with each new level	20+ Random configurations per module
Includes difficulty in the form of levels set by the user	5+ Difficulty levels
Number of regular modules	2+ Regular modules
Number of needy modules	1+ Needy modules
Rechargeable power source	30+ hours of gameplay before recharge
Manual and difficulty select converted to app	N/A

Table 1: Requirements and Specifications

The first specification we came up with is the basis of our project. We wanted to have a physical version of the game with the modules already installed. Unlike the online game, the specific games that you play each time is not a randomized set , rather we have the games already installed and you play the same specific mini-games every time. This is due to the physical implementation and also the time constraints.

Next, each of the mini-games must be randomized each time the game is played. The solution to each game should not be repeated often enough for the player to notice. 20+ configurations for each level is suitable to achieve this specification.

The online version of the game includes difficulty levels, however it does allow the player to select what level they would like to play on. Our version must incorporate these difficulty levels while allowing the player to choose which one before the game is played. We decided on 5+ difficulty levels to give the game sufficient replayability.

The number of regular modules (these are the standard mini games like Simon Says and the Password game) that are present in our game should be at least two. Due to the time that it takes to develop these games, two is a sufficient amount of games to be present. As for needy modules (a module that must be done continuously while doing the other main modules such as holding a button for a certain amount of time), one of these is sufficient.

To keep the feeling of a board game, we wanted to have the system be powered by a rechargeable power source, rather than having to have it plugged into the wall while playing. 30+ hours of gameplay should be sufficient as this is roughly the amount of time a controller for a video game console lasts. Plus, this should be a fairly low power system.

And finally, we wanted to convert the manual to an app. We feel that a paper manual would be clunky and not in the spirit of the project, thus the app should contain both difficulty select and the manual.

II. Design

A. Overview

Our solution design takes a very modular approach to setting up the system consisting of a parent board, child boards, and an IOS application. The single parent board keeps track of time, the overall game state, when to start , and when a game is finished correctly or incorrectly.It also handles communication with the child boards and the application. While our project does not feature a large number of child boards, the number of child boards that can be used in our system is scalable due to the modularity of our system's design. Our project has two puzzle boards; each of these boards is tasked with keeping only the state on the board and communication with the parent board. The IOS application acts as a central hub for the game itself. A user is able to start the game and look at the instructions for solving puzzles through the use of this application.

B. Parent Board

The parent board consists of two Atmega328P MCUs in tandem. One Atmega is tasked with maintaining important information about the overall game. This includes the game's level of difficulty, a list of uncompleted and completed puzzles, and time elapsed. As mentioned, the parent board must wait for the level of difficulty to come from the IOS application via BLE technology in order to set up the rest of the game and start. Once the difficulty level is received by the parent board, the parent can then use this information to set up the rest of the game such as how much time is allotted to this attempt, and information about the serial number. This Atmega sends all the appropriate information to each of the child boards and to the other Atmega328P on the main board, so that each can start their individual tasks. The protocol used for sending information between the parent board and child boards is the I2C protocol, allowing for more simplicity in hardware.

The second Atmega328P on the main board has a single functionality, to keep track of time. This microcontroller includes a 4 digit 7-segment display that will count down the minutes and seconds left in the playthrough once the game has started. It determines the amount of time allotted based on the level of difficulty sent through I2C communication and then sends a message through I2C back to the parent if time has run out.

C. Child Boards

Password Game. The password game program runs on an Atmega328P MCU and works by randomly selecting one word from a list of 35, five letter words defined in the manual for “Keep Talking and Nobody Explodes.” For each of the five characters, an additional number of letters n is generated based on the difficulty level passed from the parent module such that each character position has an array of size $n+1$ letters. All of the additional letters are different from the one belonging to the word chosen. In the array of letters for each character position, a random one is chosen and displayed on a 16x2 liquid crystal display [5]. Each character has a corresponding push button which utilizes an internal input pull-up resistor such that a switch press is read on a falling edge. For every press, the character displayed changes, displaying the next element in the array. It is the job of one player to communicate the different letters and the player with the manual who has the list of possible passwords determines the correct word. The player “defusing the bomb” then presses the submit button and the program checks if the letters match the randomly chosen word. If letters match, then a green LED is lit signifying the module is solved and that gets communicated to the parent chip via I2C. If the letters do not match, then a red LED is lit signifying that a strike has happened which gets communicated to the parent via I2C.

Simon Says Game. The Simon Says program also runs on an Atmega328P MCU. The game is set up after the random seed, difficulty level, and a boolean indicating if there is a vowel in the serial number, are passed from the parent via I2C. A random sequence is flashed on four different colored LED arcade buttons using the standard C PRNG and seed. The length of this sequence l is based on the difficulty level. After the sequence has flashed, the player who can look at the bomb must remember the sequence and communicate it to the player with the manual. This player can reflash the sequence with a push button press. The player with the manual then determines the solution sequence based on the system’s current configuration which the other player must also communicate.

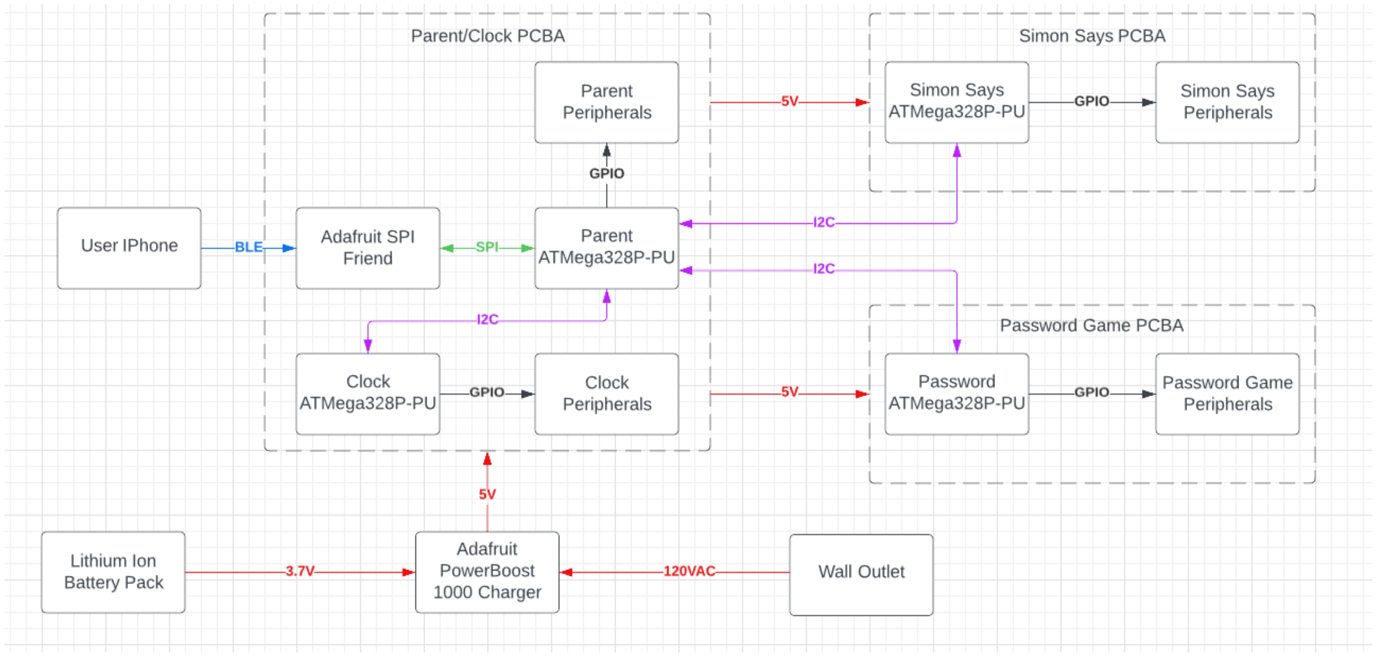


Figure 2: High level look of hardware design

The solution changes based on the existence of a vowel in the serial number and the number of strikes made so far in the attempt. The input sequence gets pressed on the arcade buttons and checked to see if it matches the solution sequence. If it does, a green light briefly flashes and the sequence length lengthens by 1 and the process repeats until three rounds are done. Then the green light is turned on and the module communicates to the parent that it is solved via I2C. If a wrong answer is inputted at any moment, a strike is sent to the parent via I2C.

III. Prototype

A. Overview

Our system incorporates all the components listed in the block diagram in Figure 2. It consists of 3 unique PCBA designs, one for each of the separate mini-games/modules. The parent PCBA controls all the communication between the boards and keeps track of all the necessary information such as total strikes, and the status of each game. The child boards contain their own hardware components and ruleset for their own respective game. The BLE application allows the user to input the difficulty setting and contains the manual.

B. Hardware Components

Moving onto the hardware of our system, there are a variety of components that will be broken down by module. Firstly, we have the parent module. At the heart of this we have an ATmega328P [1] which is running at 8 MHz and being powered with ~5V. This MCU is what controls everything for this module. It communicates with the Adafruit SPI Friend [2] using SPI, and with all the other modules using I2C. It also controls an LCD display [5] that is used for the serial number, and 3 red LEDs indicating strikes. In terms of our power supply, we are using the PowerBoost 1000 Charger from Adafruit [3]. That allows us to use a LiPo battery that can be recharged and also have the output of the battery be boosted up to 5V which we need.

Next we have the clock module. Again, this is controlled by another Atmega328P. This MCU only worries about implementing an accurate timer and controlling a 4 digit 7 segment display.

The first of two mini-games is our Simon Says game. Like all the other modules it is controlled by an ATmega328P MCU running at 5V and 8 MHz. The first piece of hardware for this game includes four LED 30mm arcade buttons in red, green, blue and yellow from Adafruit. These act both as buttons and LEDs similar to the online game version. This module also includes two regular LEDs, one in red and one in green. These indicate the status of this game as complete or incomplete. And finally, it includes one normal push button that allows the player to replay the sequence if they need to see it again.

Our last module is the Password game module. The Atmega328P controls another LCD display that displays the word for the player to cycle through. Under this we have six normal push buttons, five corresponding to each of the five letters in the word, and one to submit the password.

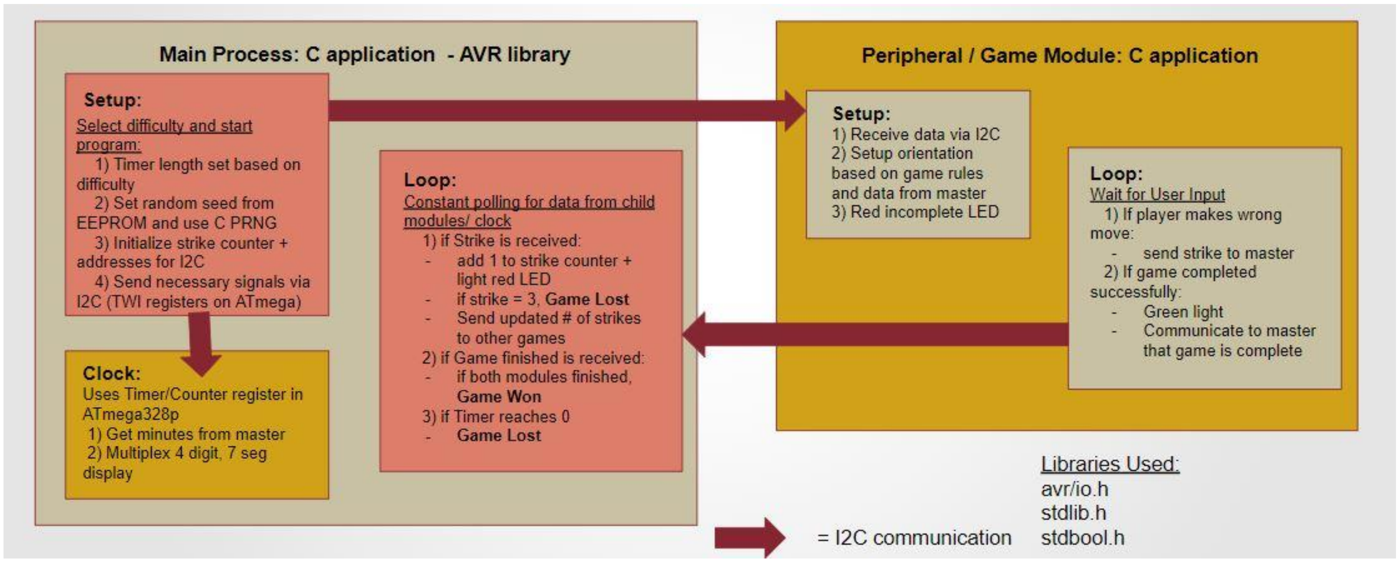

Main Process: C application - AVR library
Setup:
Select difficulty and start program:
1) Timer length set based on difficulty
2) Set random seed from EEPROM and use C PRNG
3) Initialize strike counter + addresses for I2C
4) Send necessary signals via I2C (TWI registers on ATmega)
Clock:
Uses Timer/Counter register in ATmega328p
1) Get minutes from master
2) Multiplex 4 digit, 7 seg display
Loop:
Constant polling for data from child modules/ clock
1) if Strike is received:
- add 1 to strike counter + light red LED
- if strike = 3, Game Lost
- Send updated # of strikes to other games
2) if Game finished is received:
- if both modules finished, Game Won
3) if Timer reaches 0
- Game Lost
Peripheral / Game Module: C application
Setup:
1) Receive data via I2C
2) Setup orientation based on game rules and data from master
3) Red incomplete LED
Loop:
Wait for User Input
1) If player makes wrong move:
- send strike to master
2) If game completed successfully:
- Green light
- Communicate to master that game is complete
Libraries Used:
avr/io.h
stdlib.h
stdbool.h
= I2C communication

Figure 3: Game state diagram / software component diagram

C. Software Components

The software in the system consists of four programs written in C++ using Microchip Studio. All programs include the C standard library, the C standard boolean library, and the AVR input/ output library. The parent module utilizes the Arduino BLE library for communication between the iPhone application. The program also makes use of EEPROM to store, retrieve, and update the random seed through configurations to the EEPROM registers [1]. The pseudo-random number generator used to vary the game's configuration is provided by the *rand()* function available in the standard C library. For serial communication of necessary information, all four programs utilize the I2C library provided by Arduino which greatly simplifies the complex signals involved in I2C communication. The chip on the parent board which passes the necessary variables to start the game and acts as the event handler is set up in master mode while the peripheral chips are set up in slave mode.

D. The Application

The application is a fairly simple IOS application that implements the Core Bluetooth stack for BLE app development. The application GUI is displayed in Figure 4, which shows the basic home screen and another screen for the manual PDF file. The home screen features a connect button that will begin setting up the BLE communication between the app and the system, a slider to select the level of difficulty, and the play button which sends the level to the device and transfers the user to the PDF of the manual.

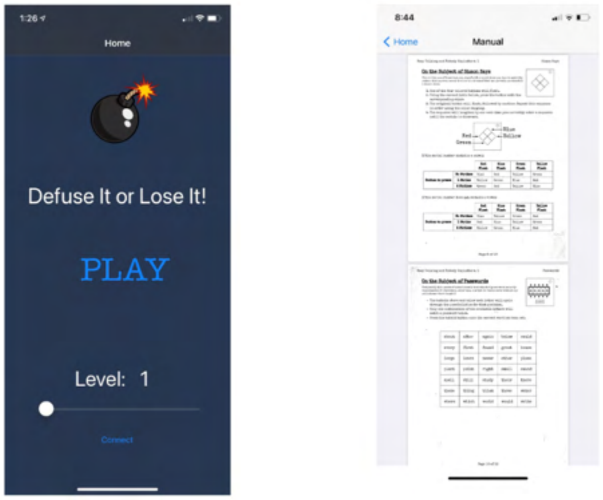


Figure 4: App home screen & manual

E. Custom Hardware

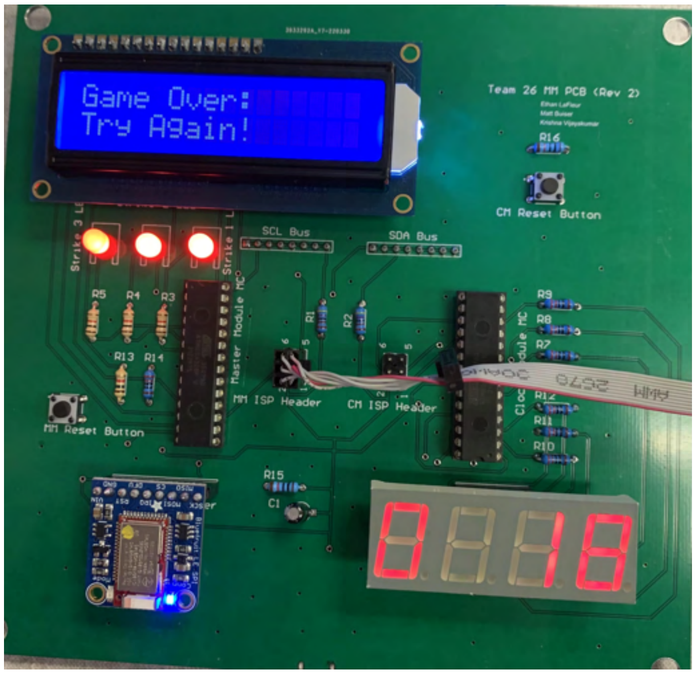


Figure 5: Parent/Clock Module PCBA

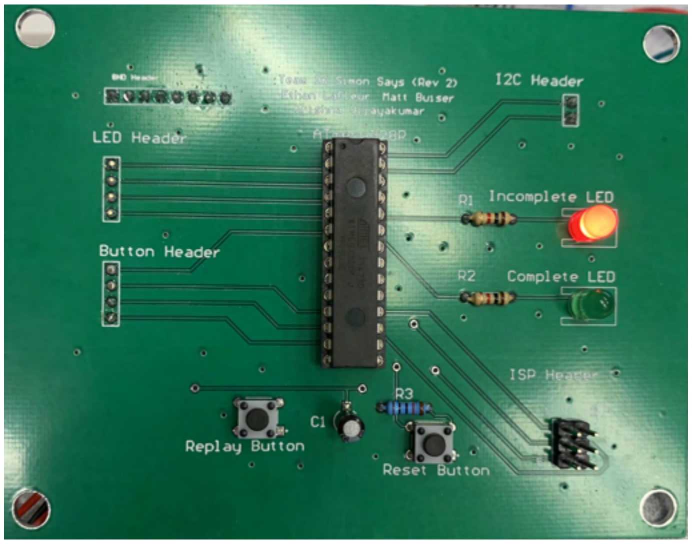


Figure 6: Simon Says Module PCBA

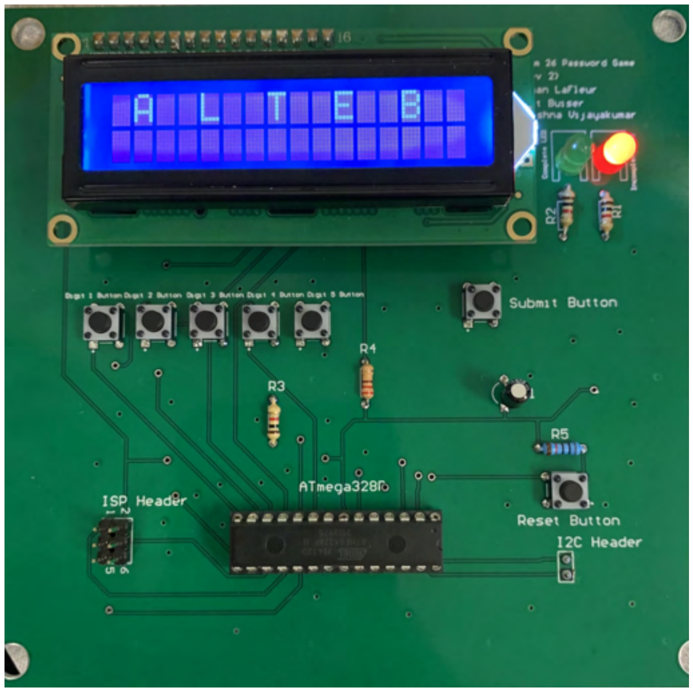


Figure 7: Password Game Module PCBA

As mentioned in the overview, we have three unique PCBAs. One containing the Parent and Clock modules, another for the Simon Says module, and one for the Password module. The design for these PCBAs was done exclusively in the Altium Designer software. The fabrication was done by JLCPCB [4]. Each PCB has an ISP header that allows it to be programmed via the Atmel ICE. The power input to the system is on the Parent PCB in the form of a MicroUSB that receives 5V. These PCBs were designed to distribute power from the Parent PCB to the other Child PCBs using JST connectors. There are I2C headers on all PCBAs that allow for communication between the boards. And of course, each PCBA contains all of its own specific hardware directly on the PCB. The population of the PCBAs was done through a variety of soldering techniques including through hole, hot air surface mount soldering, and manual hand surface mount soldering. The testing of these were mainly done by transferring what we had on solderless breadboards to the custom hardware. We were looking to see if we had the same functionality on the PCBA as we had on the breadboard.

F. Prototype Functionality

Our prototype should feature similar functionality to the virtual reality game it is based on; however, some of this functionality may be fairly limited. One of these functionalities that must be demonstrated is replayability. This means that the game should feel unique every time it is played or having different combinations of puzzle solutions. Our solution is to add elements of randomness to different parts of the system such that the user can play enough times before recognizing they are getting repeated solutions.

Another aspect that must be realized is a portable power source. Since virtual reality systems are portable, our design should be as well. This means that our system should be cordless and have a power supply that allows for multiple playthroughs on a single charge.

G. Prototype Performance

Randomness. The source of randomness in both modules comes from a pseudo-random number generator whose seed is determined by a linear congruential generator programmed in the Parent module and passed via I2C. Since the random seed is a byte, there are 256 total configurations for the game (we can exponentially increase this number in a future endeavor by having each module calculate a seed separately). The requirements for randomness in our system is that each password/button flash happens equally as often and that repeated configurations are unlikely to happen within 5-10 rounds of gameplay. These properties can be tested by looking at the serial number generated by the parent module. The serial number consists of four letters and two numbers meaning there are 1020 letters and 560 numbers for the total number of distinct sequences in the game. Since the Password and Simon Says games use the same PRNG, this means that if the serial number sufficiently satisfies the requirements for randomness, so do these games. We calculated the frequency of each letter in the string of 1020 letters and found that no letter appeared more than 46 times and no letter appeared less than 36 times. Also, the configurations of the device do not repeat until 256 rounds because they are seeded with a linear congruential generator with modulus 256. This level of randomness is sufficient for the loose requirements for our design. Refer to Appendix C for more information regarding the testing of randomness properties.

Power. Since our project is based on a virtual reality game, the prototype's battery life should be comparable to, if not better, than a virtual reality head set's battery life. As stated by Oculus, the typical VR headset will last about two to three hours on a single charge [8]. This is a fairly short battery life as other handheld gaming products such as controllers and other battery powered devices may last up to thirty hours on a single charge so for our design, we would like the system to be more comparable to this thirty hour battery life. Our prototype runs on a 6600 mAH lithium ion battery. As described in the Appendix under testing methods, we find the average current draw of our system is about 27 mA. This gives us a total of 244 hours of play time which is well over the battery life of the average video game controller and virtual reality system.

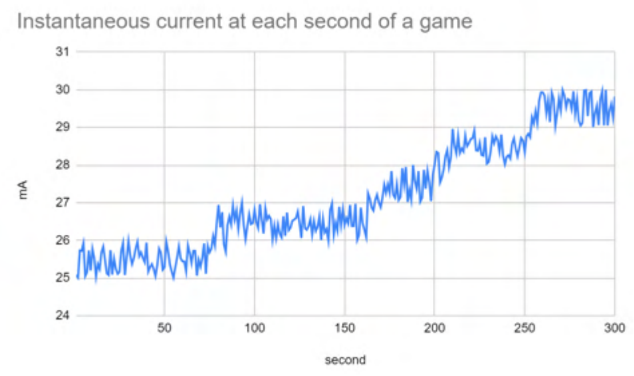


Figure 8: Graph of instantaneous velocity sampled at each second

IV. Conclusion

Altogether, this project turned out very well. We originally aimed to bring the fun online game “Keep Talking and Nobody Explodes” to the physical world. Although our game is not nearly as developed as the online game, we have provided a prototype that includes two of these mini-games and has lots of the same functionality of the original game. We are able to communicate with the systems via bluetooth, and communication between the subsystems works flawlessly. We believe that our take on the game is one that stands out among others, but also can be built upon in the future.

V. Acknowledgements

Team 26 would like to thank all the people who have helped us along the way, as there have been many. First off, we would like to thank our evaluators Professor Hollot, Professor Pourgahily, and now Professor Soules. They all gave invaluable advice and feedback throughout the duration of this project. Next we would like to thank Wouter Schievink who helped us with all of our orders and helped answer any questions we had for him. Next, we would like to thank Charles Malloch who helped us learn some of the protocols used in our project. We would like to give a big thank you to Chris Caron who answered our endless questions about nearly every facet of our project. We would also like to thank Shira Epstein for suggesting this project idea as we did not have a definitive one coming into SDP. And finally, we would like to thank Professor Gong for giving us the direction we needed to progress this project to where it is now.

VI. Appendix

A. Design Alternatives

An alternate design we considered was having multiple games on a single Atmega328P MCU to avoid dealing with potentially complex serial communication protocols and reducing the number of chips required. We decided against this design mainly because it makes the scalability of modules much more difficult. In addition to the messy circuitry, adding more games might require adding another chip which must communicate serially with another chip and one or more programs must keep track of games running on their chip as well as games running on any other chips. Our current modular design which uses one chip to setup and manage the game state allows for seamless additions and removals of peripheral modules with the two wire I2C protocol and minimal changes to the code. This is important because we wanted to keep our options open on how many games to add as well as make any future enhancements easy.

Another design we considered was in respect to the difficulty selection and starting the game. An option we considered was to do this locally on the embedded system itself. However, we did not like the idea of having a paper manual in the modern computing era, so we decided to create an IOS application for it. We decided that adding bluetooth capabilities here would enhance the user experience and make the game simpler to understand. We chose bluetooth for wireless communication because it is able to be used locally without any connection to the internet and can still satisfy any distance requirements.

B. Technical Standards

Our project included several technical standards. The main ones were I2C, SPI, and BLE. These were used in various locations in our project to implement seamless communication between the various devices incorporated in our project. I2C was the most prominent communication protocol that was used.

This was chosen because it allowed us to use one device to control all the other devices and how they communicate. Making use of the Arduino "Wire.h" library we were able to implement polling from our parent microcontroller to all the child microcontrollers. This protocol also only uses two wires which was useful as we had many peripheral devices that used a lot of GPIOs. The two wire interface also provides scalability for serial communication between devices as all the clock lines and all the data lines for each device can be arranged in parallel on two rails. This can allow for more games to be added in a possible future endeavor.

For the communication between the parent microcontroller and the Adafruit SPI Friend [2], we used SPI of course. SPI was chosen in this case because in our original prototype using the Adafruit Feather 32u4 Bluefruit LE, we implemented this communication using SPI as it was already present on the breakout board. Once we moved to the standalone component, implementation of SPI was easy as we already knew the method.

And finally, our application was implemented via Bluetooth Low Energy. The leading reason we chose Bluetooth LE is because it is low energy. Implementing BLE allowed us to get rid of a paper manual, and made use of an IPhone which holds a majority of the market share of mobile devices. BLE was also useful as it does not require anything other than a cell phone which can be used anyplace, and does not need to be connected to a larger network.

C. Testing Methods

Randomness. The PRNG used in our system is provided by the *rand()* function available in the C standard library. The seed to the PRNG is assigned by a linear congruential generator which has requisite coefficients to make its cycle length equal to the modulus which is 256. This is because the current seed is stored in EEPROM which uses 8 bits per address. The current seed is used as the input to the following seed. We use this LCG to seed the PRNG because noticeable patterns in the serial number and words chosen for the password game occurred with simple incrementing of the seed. The number of possible configurations is thus, 256. This number can be exponentially increased by having each module generate its own seed for its random number generator. The properties of randomness our system must satisfy are that each word chosen for the password game, LED flashed for a sequence in the Simon Says game, and character in the serial number is around equally likely to happen i.e. there is no human identifiable deviation from a uniform distribution. This property can be tested by checking the number of letters in the serial number. A serial number contains six characters, four letters and two numbers. Since there are 256 possible configurations before repetition, we recorded 256 serial numbers containing 1020 letters.

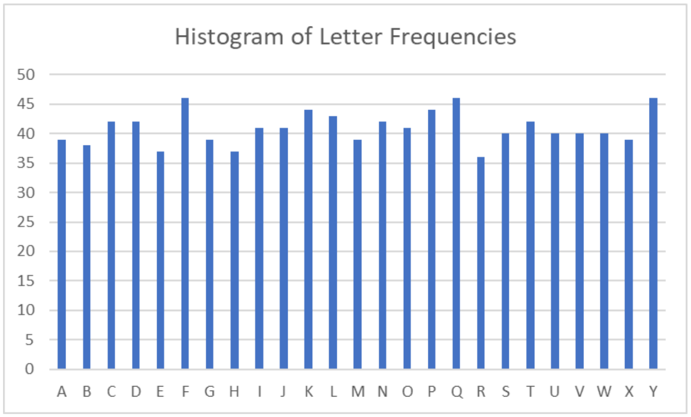


Figure 9: Histogram of letter frequencies

Looking at the histogram of letter frequencies as shown in Figure 5, no letter appears less than 36 times and no letter appears more than 46 times. For a casual player, this is close enough to a uniform distribution such that deviations are not noticeable. Applying a more quantitative metric such as the Kolmogorov-Smirnov goodness-of-fit test for uniform distributions which quantifies distance between the distribution of samples from the reference cumulative distribution function, results in sufficient evidence against uniformity [6]. Another property regarding the randomness of our system is that the configurations should not repeat in 10 or less consecutive rounds. This property can be tested simply through analysis of the random number generator. The *rand()* PRNG is deterministic and is seeded with a deterministic linear congruential generator, so the configurations cycle through the length of the modulus on the LCG which is 256. This means that a user would have to play the game more than 256 times to see repeated configurations. One downside to the deterministic nature of our system is that its configuration can be determined with knowledge of the seed or the coefficients of the LCG. However, this information is not easily accessible to a player as it is in firmware and EEPROM memory. An attacker could connect a supported AVR debugger to the on-chip 6-pin ISP header or perform other circuit level attacks, but we do not consider this as an issue because the system is meant for casual fun. Thus, we believe that security through obscurity is sufficient and the properties of randomness sufficient and are satisfied.

Power. We wanted to test how long our system should last such that the user does not have to recharge it after very few uses. This means that our system should be able to last on a single charge for an extended period of time on our 6600 mAH battery. We measure current drawn from the battery by connecting an ammeter in series with our system as shown in Figure 9. We then estimate how many games can be played on a single charge of the battery. We do this by periodically sampling the instantaneous current draw using the ammeter over a small sample of five minute games to find the average current draw of each playthrough. From these samples, we average these playthroughs to determine an overall average current draw for a single playthrough. We then determine how many games with this average current draw to determine approximately how long the battery should last.

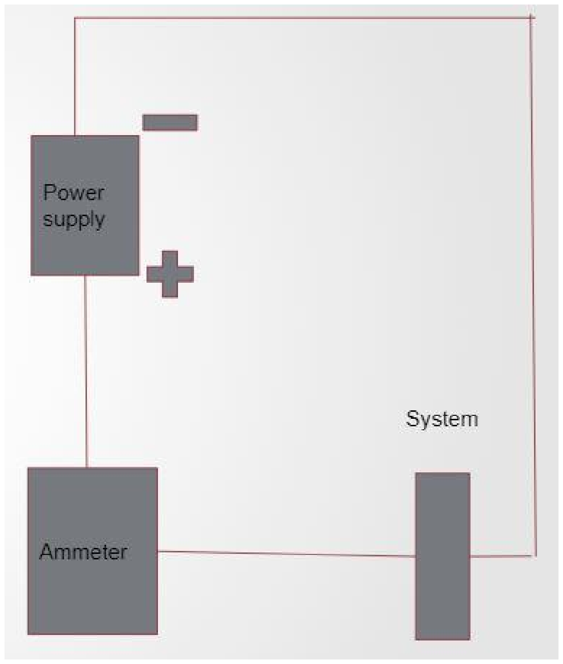
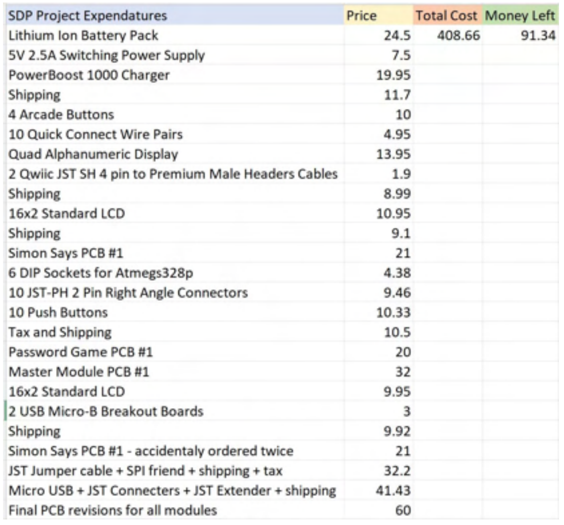


Figure 10: Circuit to analyze power

D. Project Expenditures



SDP Project Expendatures	Price	Total Cost	Money Left
Lithium Ion Battery Pack	24.5	408.66	91.34
5V 2.5A Switching Power Supply	7.5		
PowerBoost 1000 Charger	19.95		
Shipping	11.7		
4 Arcade Buttons	10		
10 Quick Connect Wire Pairs	4.95		
Quad Alphanumeric Display	13.95		
2 Qwiic JST SH 4 pin to Premium Male Headers Cables	1.9		
Shipping	8.99		
16x2 Standard LCD	10.95		
Shipping	9.1		
Simon Says PCB #1	21		
6 DIP Sockets for Atmegs328p	4.38		
10 JST-PH 2 Pin Right Angle Connectors	9.46		
10 Push Buttons	10.33		
Tax and Shipping	10.5		
Password Game PCB #1	20		
Master Module PCB #1	32		
16x2 Standard LCD	9.95		
2 USB Micro-B Breakout Boards	3		
Shipping	9.92		
Simon Says PCB #1 - accidentaly ordered twice	21		
JST Jumper cable + SPI friend + shipping + tax	32.2		
Micro USB + JST Connecters + JST Extender + shipping	41.43		
Final PCB revisions for all modules	60		

Table 2: SDP Expenditures

Our project stayed comfortably under the $500 budget.

E. Project Management

Being a three person team made sure that our whole team had a significant role in the development of the project. Ethan was mainly responsible for the hardware of the project. He was in charge of assembling the project on breadboards and PCBAs. He was also responsible for designing, populating, and testing the PCBAs. He was also in charge of designing the enclosure for our final version of the prototype. And finally, he was team coordinator. Krishna was mainly in charge of the development of the software for all the modules. He worked closely with the code in both Arduino and in C. He did research on how to implement some of the technical standards we used like I2C. Krishna was also the budget lead, being responsible for submitting orders for all the necessary parts. Edward was in charge of the bluetooth application development. He developed the application for an IOS device and implemented bluetooth communication into the project. When not working on the application, he assisted Krishna with the development of the C code. He was in charge of submitting and keeping track of the orders from JLCPCB [4].

F. Beyond the Classroom

1) Ethan: Before this project I did not have much hands-on experience. Firstly, this project was very helpful on learning how to be a part of a team. I had worked on group projects before, but nothing on this scale. Managing a project like this has definitely helped me grow as an engineer. In terms of actual engineering skills, I got to work with many things I had never seen before. I learned how to solder both through hole and SMD. And of course, got lots of experience with PCB design using Altium Designer. I worked with different communication protocols and standards for microcontrollers like I2C and SPI, and sharpened my C coding skills as well. All in all this project exposed me to a lot of different things which I believe will help me in the future as I move into a professional career.

2) Matt: Like Ethan, I also did not have much hands-on experience with developing an engineering project let alone carrying one out on a small team. This experience gave me a much better understanding of the engineering process and what it takes to develop a system from the original brainstorming of ideas to finalizing the design on a custom PCB. This process has also taught me how to work as a group in a whole new capacity. Many assignments throughout my college career have been completed on my own or in a pair. Something about working in a slightly larger group showed me how important communication can be on large scale projects like the Senior Design Project. We found that it may be difficult to understand a part code or design that someone else has written earlier and so it is important to communicate to your team, whether it is personally explaining what has been done or commenting code to explain what has been done. This project has also helped me develop my engineering skills. We have not used many of the parts and protocols used in our project so I got a better understanding of how to research, implement, and test these parts and protocols. I believe that SDP helped me better develop my engineering skills that I will eventually use in my career.

3) Krishna: Similarly to Matt and Ethan, I have not had any experience with a year-long, independent project, which forced me to adapt quickly, manage time effectively, and collaborate with the group. Working primarily with low-level embedded software, a major skill I needed to develop was troubleshooting. The more limited debugging capabilities on embedded systems and having such a wide range for the source of issues guided me to develop innovative debugging techniques that ranged from many different layers of a computing system. Some methods I utilized throughout the project were analyzing circuit components and their connections, reading circuit-level analog and digital signals, and modifying/ stepping through various critical parts of code. Some of these skills and knowledge will definitely carry over to my professional career.

VII. References

[1]Atmel Corporation, “ATmega328P Datasheet,” Datasheet, 2015.

[2] K. Townsend, “Introducing the Adafruit Bluefruit LE SPI Friend,” *Adafruit.com*, 07-Jul-2015. [Online]. Available: https://learn.adafruit.com/introducing-the-adafruit-bluefruit-spi-breakout. [Accessed: 08-Apr-2022].

[3] L. Ada, “Adafruit Powerboost 1000C,” *Adafruit.com*, 21-Apr-2015. [Online]. Available: https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost. [Accessed: 08-Apr-2022].

[4] “PCB prototype & PCB fabrication manufacturer,” *JLCPCB*. [Online]. Available: https://jlcpcb.com/. [Accessed: 08-Apr-2022].

[5] Hitachi, “HD44780U (LCD-II),” Datasheet.

[6] “Kolmogorov-Smirnov Test,” *Kolmogorov-Smirnov test - Encyclopedia of Mathematics*. [Online]. Available: https://encyclopediaofmath.org/wiki/Kolmogorov-Smirnov_test. [Accessed: 06-Apr-2022].

[7] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. Urbana, IL: Univ. of Illinois Press, 1949.

[8] “VR headsets, Games & Equipment,” *Oculus*, 11-Oct-2020. [Online]. Available: https://www.oculus.com/. [Accessed: 08-Apr-2022].

[9] “R/KTANE - I made a physical version of the video game keep talking and nobody explodes!,” *reddit*. [Online]. Available: https://www.reddit.com/r/ktane/comments/a3fgyt/i_made_a_physical_version_of_the_video_game_keep/. [Accessed: 08-Apr-2022].

[10] “R/3dprinting - I made a bomb (Keep Talking nobody explodes),” *reddit*. [Online]. Available: https://www.reddit.com/r/3Dprinting/comments/j0q9ls/i_made_a_bomb_keep_talking_nobody_explodes/. [Accessed: 08-Apr-2022].