

# Tetra Board

Aleck Chen, EE, James Gnall, EE, Varak Mouradian, EE, and Vincent Nguyen, CSE

**Abstract**—Since its emergence in the 1930s, the household game Battleship has evolved from a pencil and paper game into a fully-fledged board game. In its current iteration, the board game requires the use of physical pieces of ships and pegs to play. This requirement may prove to be quite demanding for those with physical disabilities. Our focus is to create an authentic Battleship experience without the use of physical pieces. We will replace the traditional board game pieces with LEDs and implement a voice recognition system to play the game. Our team will seek to deliver a single player version of Battleship where a human player will compete against a computer. In keeping with the traditional aspects of Battleship, we will also implement a speaker to output the results of the player’s movements.

## I. INTRODUCTION

### A. Significance

The tabletop gaming industry has been known to lag behind other gaming industries when it comes to making their products more accessible to users. Game designers are prone to make assumptions about who is going to be playing their game and that in turn alienates demographics who struggle to play by the developer’s rules. As explained in an interview with Dr. Michael Heron of Meeple Like Us, an organization dedicated to promoting game accessibility, the requirement of tight physical placements and the inability to verbalize your instructions would turn many away from playing these types of board games [1]. Adapting a classic board game like Battleship for people with physical impairments is achieved by simplifying each of the core elements of the game as much as possible while keeping with the game’s aesthetic. Physical pieces used to represent ships and pegs to represent attacks can be relegated to simpler components that would operate accordingly without requiring players to move these pieces themselves. These pieces need to be able to operate based on the input of the player and since we would like to make the experience as hands free as possible, we will look to rely upon the player’s verbalization of their instructions to operate. By replacing physical pieces with LEDs and implementing a voice recognition system, players will only need to speak into a microphone to be able to play a game of Battleship.

### A. Context and Existing Products

The issue of accessibility in board games has been an issue for a long time and people have already tried to solve this

problem. For example, BrickSimple has created a version of the Battleship game that can be played using the Google Glass eyewear. This accomplishes a hands-free approach that our team is looking to solve as well by implementing a voice recognition system and the ability to play against other Google Glass users. The game is programmed to allow users to play the game at their leisure when they are completing other everyday tasks. The layout of this iteration is the traditional two boards per player layout for Battleship, all of which is displayed on the corner of the eyewear’s screen. The commands are inputted through a voice recognition system that accepts two corresponding numbers between 1-7 to represent a section on the gridded board. Once a command is received, the game will process it accordingly and deliver appropriate feedback on whether it was deemed a hit or miss. This version of Battleship promotes portability by being able to operate anywhere at any time. But as innovative as this design may be, most people do not want to be distracted by a game of Battleship when they are conducting their daily activities. The game is meant to be completed at the convenience of both players and nowhere is more convenient than in your own home. This makes the Google Glass component unnecessary when most people wish to conduct this game in a less distracting environment, unlike outside during the day.

Another design alternative that promoted more accessible play is shown on our mobile devices. With an abundance of applications found online nowadays, it comes as no surprise that classic board games like Battleship have made their way onto the online marketplace. One such mobile version of Battleship is known as Fleet Battle found on Android devices. This follows the traditional gameplay of Battleship but also includes many animations to enhance the experience while playing on your mobile devices. This is a welcome addition to a classic game and may prove that the future of board games is to end up on the mobile applications market. But in terms of addressing how accessible this application may be for those with physical disabilities, this game does not seem to address those concerns. With no voice recognition system to input in commands, the application prompts players to use the touch screen features on their smart devices to manually pick the point they wish to attack. This may prove to be quite an uncomfortable task for those with physical disabilities as when they would try to play this game with physical pieces. These movements needed by players may turn this demographic away from this title and that is something our group is looking to avoid.

### B. Societal Impacts

To create a welcoming experience for players with physical disabilities, our group is looking to remove any feature that may prove too difficult to perform while also maintaining the traditional gameplay of the board game. We first begin by

looking into replacing physical pieces with electrical devices that will represent the game’s pieces and adapt to any situation found throughout the experience. We found that light emitting diodes are perfect for representing our physical pieces as they can be easily programmed to match the situation we would like to present. Now, looking at how we would like the game to be operated by players that is achievable by all, implementing a voice recognition system has proven to be the perfect solution to our problem. Using vocal inputs to designate where a player wants to attack is the least physically involved way of playing Battleship, so we looked to implement this feature as effectively as possible. These main features work in tandem to simulate the experience of playing Battleship for those who have had trouble in the past when playing these types of board games. People who struggled with placing pieces into small holes on the boards will now have the ability to play this board game unobstructed by the physically demanding features found in previous iterations.

### C. Requirements Analysis and Specifications

Requirements	Specification	Value
Portability	Double-Sided Briefcase	12” x 6” x 9”
Game Board	4 Boards	8” x 8” per board
Display	DotStar LED	5V supply
Sound	Speaker	“Hit” “Miss” “Sunken Ship”
Voice Control	1 Microphone	Alphanumeric Commands i.e. Alpha 3
Power	External 5V Wall Mount	5V

Table I: Requirements and Specifications

## II. DESIGN



Figure 1: Final Product

### A. Overview

Board games require physical movement, and those with poor motor skills struggle or will not have a chance to play. To approach this problem, we decided to use voice recognition. We want to eliminate as much of the physical movement required to play as possible. Our design specifications are shown in Table I.

The heart of our system is a Raspberry Pi 4, which is responsible for processing real-time voice recognition and the gameplay for our board games. The Pi will take in audio input from the microphone and process user commands. Then, it will update the LED boards and output correct sounds for the user.

The specification of requiring a double sided briefcase mainly affected the way that the four boards of LEDs were able to be implemented and connected. The necessity of portability ensured that the LEDs would need to be sturdy enough to be folded up for transportation. This resulted in the LEDs being soldered together with unequal spacing from vertical to horizontal. Originally a perfectly symmetrical 8x8 square of LEDs was planned for but the more solders required to connect the LEDs the less sturdy they were found to be. To combat this, wider spaced strips were bought and then connected together at the end of each row instead of each one soldered to the next.

For our game boards, we decided to use the dotstar LEDs to light up the board for either a “miss”, a “hit”, a “sunken ship”, and the different types of ships that you have available on your board. With the dotstar LEDs, we had very reasonable power usage for lighting up all 4 boards which contain 256 LEDs in total. With our current power consumption for the boards and raspberry pi, we are able to scale down the amount of voltage needed from our pcb. The only problem with the use of these LEDs was the fact that we could not physically replace each LED to its corresponding grid. In order to replace a single LED, we would need to replace the entire row of LEDs with our current configuration. This would require us to resolder on a new strip of LEDs to replace the row which would be inconvenient. Although the replacement of an individual LED would be a hassle, it is worth the tradeoff in terms of power for us to use them as it would be very unlikely that an LED would burn out due to the very minimal power consumption.

For our display, we chose to use the dotstar LEDs over many different types of displays. For Tetra Board, we wanted to keep as much of the old aesthetic of battleship with a slightly more modern approach to the game. With that being said, there were many options for different displays we could have used like neopixels, 4 pin LEDs, and a touch screen display. We didn’t use the neopixels as they weren’t synchronized with the raspberry pi clock even though they consumed around the same power as the dotstar LEDs. For the 4 pin diode LEDs, these were too power hungry and if we wanted to power all 4 boards with them, it would consume an

excessive amount of power that we couldn't continuously power up unless it was directly supplied from the wall. To add to this, with a constant high power consumption, the risk of burning out an LED would be high and this would cause many problems when it comes to replacing each individual LED on the grid. The 4 pin LEDs would also be much harder to solder together, risking the structural integrity of our boards. Lastly, we did not choose a touch screen display because we wanted to keep our product as hands-free as possible. We could have had a screen display and still used voice recognition, but we wanted to keep the old aesthetic and truly make a unique product that isn't just another screen display out there on the market. Although the screen display would have a much more modern look and structural integrity inside the briefcase, the cost of getting 4 screens for the entire board would be too high to be justifiable for this product. All in all, the dotstar LEDs have a good combination of compatibility, aesthetic, power consumption, and cost.

To enhance the experience of tabletop games like Battleship, we sought to add an external speaker that would play the appropriate sounds associated with the corresponding moves made during the game. Each of these distinct sounds would notify players that they have either damaged, missed or sunk the opposing player's ship. We used a wired speaker that was connected to the Raspberry Pi to play sounds after every move. We would eventually look to integrate a speaker that was compact enough to house somewhere around the briefcase so that it can be easily transported with the rest of the device.

For our voice recognition design, we chose to use one central microphone for both players in the game. The alternative to the single microphone would be using multiple headsets for each player. Having two headsets would be much easier for the voice recognition as the microphone would be picking up sounds directly from the respective players. The problem with this is the portability aspect of our product. We want to keep this product as hands-free as possible and using the headsets would take away from that. Although the voice recognition would be slightly more accurate with the individual headsets, this is a tradeoff we are willing to take as we have taken many measures to improve the voice recognition system itself.

To meet our power requirements so that our game board can maintain portability, we sought to design a voltage regulator PCB. The electronic hardware would receive a DC voltage input from a wall outlet and the PCB would then distribute 5V and 3A to all necessary hardware components. The components to be supplied were Raspberry Pi and the 4 LED boards. The PCB would be housed within the briefcase so that the board can be taken and used anywhere near a wall outlet.

The block diagram of our system is shown below in Figure II.

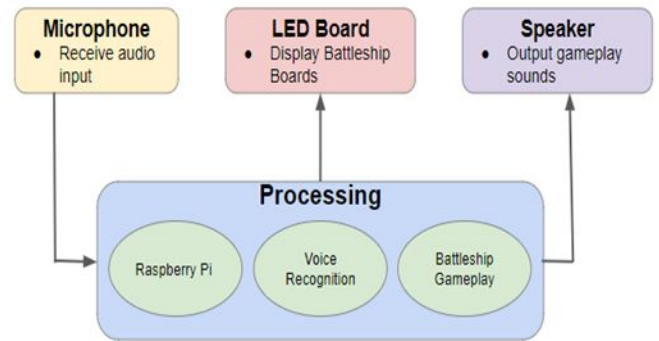


Figure II: Block Diagram

### B. Microphone

The microphone that is interfaced with the Raspberry Pi is a standard 3.5mm omnidirectional microphone. The microphone can pick up sound with equal gain from all directions of the microphone. The Raspberry Pi does not have an onboard microphone jack. For this reason, we used a USB adapter to interface the microphone. To use this microphone, we needed to install the PyAudio and SpeechRecognition Python libraries. PyAudio allows us to use Python to play and record audio streams.

### C. Speaker

We do not have a dedicated speaker for our system yet, but we used a standard speaker with a 3.5mm port to connect to the onboard 3.5mm jack on the Raspberry Pi. To correctly output sounds through the speaker, we adjusted the ALSA configuration file to set the universal output for the Pi. Additionally, we used the SimpleAudio Python package which provides cross-platform, dependency-free audio playback capability for Windows and Linux. We also created our own library of existing sounds (8 bit .wav files) to indicate "Hit", "Miss", and "Sunk".

#### D. LEDs

	1	2	3	4	5	6	7	8
A	●	●	●	●	●			
B								
C	●					●	●	
D	●							
E	●			●				
F	●			●				
G								
H						●	●	●

Figure III: Battleship LED Layout

For our version of the Battleship game, we are implementing LED boards in order to make the game more customizable and visually appealing. Skills from ECE 211 and 212 were used for this portion of the project like soldering and power analysis of the LED boards. There will be a total of four boards for our final project, one board for your ship placement and another board for tracking your opponent's board for each player in the two-player mode of this game. The two boards for each player will be perpendicular to each other with the ship placement board parallel to the floor as the base and the attack tracking board will be placed vertically. The two sets of boards will be divided by a double-sided suitcase.

The LEDs that we are using for the boards are Adafruit DotStar LEDs. [2] These were the LEDs we chose to use as it was individually addressable by the Raspberry Pi and included internal multiplexing. The DotStar has four different traces which are the power, clock, data, and ground. The timing between the microcontroller commands to the LEDs are important for the fluidity of the game and this makes the DotStar the best choice for the board as its clock timing is synced to the microcontrollers processing. This enables us to properly light up the corresponding LEDs without having any delays that could possibly interrupt the clock cycle. The board layouts themselves are arranged as an 8x8 LED grid shown in Figure III, where an individual LED represents a tile on your board that can be used to place parts of your ship.

Another reason for our choice of the DotStar is its power efficiency. A strip of 30 LEDs consumes a total of 9 watts of power with a 5 volt power supply which will drive 1.8 amps of current. At face value, this seems like a high amount of power being consumed if the LEDs are constantly on over the duration of a game that can take up to 20 minutes, but these are the specifications of the LEDs at maximum brightness. For

our implementation of the LEDs, we will be using a brightness of just 10% that can be controlled by the microcontroller. At maximum brightness, the LEDs were far too bright to look at without being blinded, so we chose to use 10% of its maximum brightness which is still very visible. At 10% brightness, each LED will consume approximately 30 mW of power. The colors for the DotStar that consume this much power are only the primary colors which are red, blue, and green. Colors that are mixed like purple or yellow will consume less power than 30 mW per LED. With all four boards completely lit up in the primary colors, the power consumed will be 7.68 W. This will be the power consumption of all 256 LEDs on the boards. This scenario will only be for a very short amount of time as all four boards would only be completely lit up at the very end of a Battleship game. Using our current brightness for the LEDs, it is very power efficient compared to other potential LEDs.

Our future boards for the project will have these LEDs evenly spaced out which will be approximately 8 by 8 inches to simulate the real-life Battleship game boards. The LEDs are currently soldered together, connecting each LED horizontally, then at the end of the row it is soldered to the next row below it. The Raspberry Pi sends data one way down the LED rows for them to function properly. We plan to have one PCB that can power all four boards. One of the boards is going to contain the PCB which will be the master board and that will be connected to the other 3 boards in parallel. The PCB is necessary for two player mode as our current design can only handle two LED boards. The Raspberry Pi only has two output pins to power two of the boards, so it is not feasible to just have the Raspberry Pi alone. With the PCB powering the four LED boards, the Raspberry Pi will solely send data to the boards to process commands taken in from the voice recognition software. This makes sure our Raspberry Pi does not overheat or burnout from constantly providing power to the boards. We plan to use a 5 volt wall mount to power the entire board setup.

#### E. Processing

##### Microcontroller

The microcontroller we opted to use was the Raspberry Pi 4. The Pi 4 offers the processing power that our system requires for real time speech recognition. It also provides more than adequate storage space for the voice recognition library and the data for the games we will implement. Moreover, it has on-board Wi-Fi for our online speech recognition and many GPIO pins to interface the LED boards, microphone, and speaker.

To get started, we downloaded NOOBS, an installer for Raspbian, onto a formatted SD card. The Raspberry Pi automatically installs Raspbian with the installer on the SD card. Raspbian is a modified version of Debian Linux, which

we are familiar with. Raspbian comes with Python 2 by default. Before this project, we were not yet proficient in Python. Python 3 is generally easier to learn and it is future proof. For this reason, we installed and used Python 3. Many of today’s developers use and create libraries strictly for Python 3 and these are not backwards compatible with Python 2. In our case, the SpeechRecognition library, SimpleAudio package, and Adafruit\_Blinka package are only made available for Python 3. Additionally, Python 3 has faster runtimes and the community support for Python 3 is significantly better.

Getting started with the Adafruit DotStar LEDs was a simple process. We needed to install the Adafruit\_Blinka package that provides CircuitPython support in Python. This package allows for full control over the individual RGB LEDs for each board. The DotStar LEDs came in LED strips which are addressable as a 1D array; LEDs are indexed starting at 0 to X-1, where X-1 is the last LED at the end of the strip. We needed to account for this as we need each board to be represented as a 2D array. The first row of LEDs are addressed from 0-7, the second row is 15-8, and so on.

### Voice Recognition

For our MDR prototype, we chose to utilize a speech recognition package. A handful of packages for speech recognition exist on PyPI. The SpeechRecognition library acts as a wrapper for several speech APIs. SpeechRecognition requires PyAudio to interact with the microphone interfaced on the Raspberry Pi. We have tested the offline package, pocketsphinx, and google-cloud-speech.

Pocketsphinx was the only library that works offline, and thus most suitable for our project in terms of portability. However, we encountered many issues such as inaccurate speech recognition and its specific language model for English. At first, we used the default library and language model. It could pick up sentences accurately in a quiet environment. However, we did not want the entire English library. In an attempt to solve this issue, we created our own library for specific words to pick up. However, it continued to pick up words in the library that were not said, even in quiet environments.

In our testing, Google Speech Recognition was more accurate than pocketsphinx. Google-cloud-speech is an online voice recognition API that enables us to convert audio to text. This API applies powerful neural network models to the audio file that the Pi will send to Google. It returns a string of text. This method is very accurate, but in our testing, it takes up to 10 seconds to process a command.

In the second half of the year, we were able to implement a Butterworth filter for preprocessing speech before sending the audio file to Google Speech to Text. Moreover, we also implemented our own database for post-processing the output from Google Speech to Text. This significantly improved our

voice recognition software, especially in noisy environments.

### Battleship

The code for the battleship gameplay is written in python to program the Raspberry Pi. Programming techniques learned in both ECE 122 and ECE 242 have been utilized to complete this portion of the project. While these classes were taught in java and we are programming in Python, the ability to manipulate multidimensional arrays of objects as well as input and output data are invaluable skills that bridge languages. The gameplay currently only includes a single player mode in which the user can play against a Computer Programmed User.

To begin, both the CPU and the player have their board layouts randomly generated. The ships can be anywhere on the board, but they will always be continuous in either a horizontal or vertical direction, never diagonal. Eventually the ability to manually set ship locations will be implemented, although the rules of how ships can be set will still apply. It has one difficulty setting in which the computer guesses random locations on the board until it gets a hit, and then guesses spots around that hit until a boat is sunk. Once the boat is sunk, the CPU goes back to randomly attacking spaces.

Going forward, two additional difficulty settings of the CPU will be added to accommodate players looking for a challenging experience. The difficulty level will have the most impact towards the end of the game, when there will be locations that could not fit a ship that a more difficult CPU can choose not to target in its random guesses. If a boat is dunk for the player or computer, then the computer will recognize that and notify the player. It will also notify the player if either the CPU or the player wins the game and ends it. A second player option will also be added next semester, allowing for two users to play against each other.

### III. PROJECT MANAGEMENT

Goal	Status
The ability to play against CPU	Accomplished
Quick Play mode	Accomplished
Board lights up correctly on hit or miss	Accomplished
Speaker projects sounds for gameplay	Accomplished

Table II: Proposed MDR Deliverables

Our team has consistently met both with our advisor and amongst ourselves throughout the semester, allowing us to

progress effectively through the challenges this project has provided. Through frequent communication between group members over text and in person, we were able to collaborate to accomplish almost all our MDR goals, as shown above in Table II. We were able to successfully generate the code required to set up the boats for both the CPU and a player at the start of each game, the gameplay itself with one level of difficulty for the CPU, and the ability to light up the board to properly display the game as well as emit sounds out of the speaker corresponding to the game events. The two LED boards promised were created successfully. They were soldered together and attached to a piece of cardboard so they stayed in place. We currently have two options for inputting moves, one on the keyboard and one through voice command.

The one aspect of our project that we were not able to fulfill completely for MDR was having the speaker work simultaneously with the voice command. When the voice command input is active, there is a feedback loop that results in a very irritating sound being emitted from the speaker. When the keyboard input is active however, the speaker works perfectly fine, emitting the programmed hit, sink, and miss sounds at the proper times. The voice recognition works decently well, however in its current state it is somewhat slow, as well as very inaccurate if there is more than a small amount of background noise.



Figure IV: Gantt Chart

While we have progressed quite well and a large portion of the project has been completed, there is still a good deal of work to be done. Figure IV, shown above, outlines our plans to continue steady progress next semester. Vincent will begin the semester by working on solving the issue with the speaker. He will then implement the code to allow a second option of setup with manual setup of ships. His main task for next semester will be helping to implement a more functional voice recognition software as that is what most of our project centers around. James will spend the beginning of the semester adding features to the gameplay such as difficulty levels to the CPU, and the ability to play a player versus player game. He will also assist Vincent in developing the enhanced voice recognition functionality. Varak's main task will be to design and set up the PCB, Aleck may assist him as well when needed. Once that is completed, he and Aleck are going to move on to setting up the housing of Tetra Board, a suitcase, ensuring that it looks good while also maintaining all of the functionality we desire. Aleck's main task of next semester

will be putting together the third and fourth arrays of LEDs, as well as setting up the original two arrays in a more aesthetically appealing and sturdy fashion. If time permits, he will then move on to adding the additional games of tic tac toe and Connect 4 to Tetra Board. As CDR comes closer we will all work on that presentation as well as delegate further tasks to those who have the least left to do with their given duties.

To complete all these challenging tasks on time, we will have to continue to work diligently and rigorously. As there are many more potential challenges, we will keep up with regularly meeting as a group to problem-solve new issues as they arise and make changes to who is working on what and when if a critical component is stalling. In order to come as far as we have, we have all had to contribute a great deal, and we plan to come into next semester with that same tenacity to complete Tetra Board as a well-designed, fully functional finished product by FDR.

## IV. The Product

### A. Product Overview

At the time of CDR Tetra Board had all of its components working but they were not all housed in the locations where they would have been for the final demo. Shown below in Figure V is the plan for the final demo, detailing where each of the core components of the block diagram would be located.

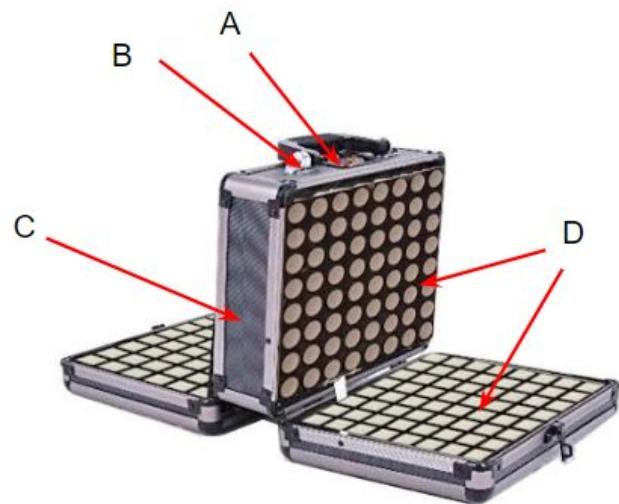


Figure V: Product Sketch

Label A is the microphone, which we have clipped to the top of the case where it can pick up the audio input from each player. Label B points to the speaker which would be placed at the top of the middle section of the case near the microphone.

## SDP20 – Team 7

Label C points to the center of the case, inside of which the Raspberry Pi which handles all of the processing would be located and connected to the boards, speaker, and microphone. Lastly, label D points to two of the LED boards, the other two boards are on the other side of the case. The PCB would also be located inside the middle section connecting to the boards and the Pi to power the system. This would also have a cord connecting to the side coming from a wall mount to provide power to the PCB. Shown below in Figure VI is a picture of Tetra Board set up to present for CDR.

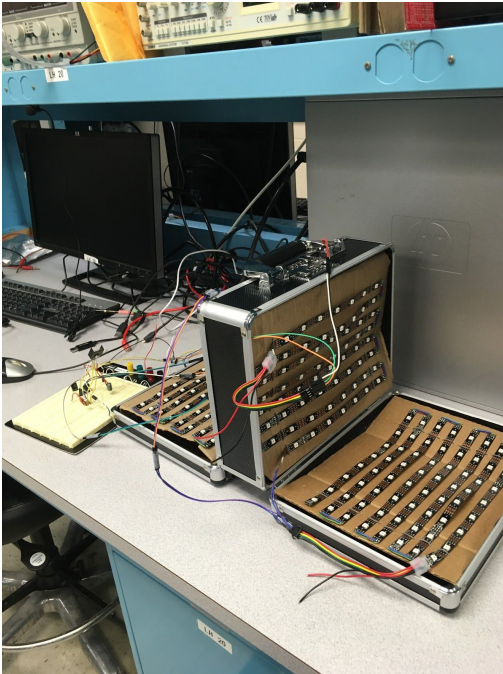


Figure VI: State of Tetra Board at CDR

As can be seen in the picture, the four boards are inside of the briefcase holding all 256 LEDs. At this time our PCB had not been shipped so we have the power supply setup in the breadboard seen off to the left. The microphone is attached to the desk shown hanging down in the top left of the image. It was set up for easier use since it was to be used while sitting at the computer at that time. We were still using a separate speaker at that time; it is not shown in this image.

### Electronic Hardware Component

Production of the PCB began at the end of the Fall semester. The goal for our electronic hardware component was that our design needed to be supplied enough power to function while also maintaining the portability specification we had set for our team. Both the Raspberry Pi and all 4 of the LED boards needed to be supplied with ample power. The Raspberry Pi 4 Model B our team had been using required a 5V 3A input to function according to its datasheet. After researching about the power requirements and running our own set of power consumption tests, we determined that all 4 LED boards

would also need a 5V 3A input to function. Since we wanted our game board to be portable, we decided to have our PCB receive input from an AC-DC converter that is connected to a wall outlet. The goal was to populate the PCB inside the briefcase, allowing a small opening in which the hardware component would receive the wall outlet's input. The most appropriate voltage regulator our team found to distribute the ample power requirements was the LM2596T which can output 5V 3A while receiving a 12V DC input. Our team deemed that both the LED boards and the Raspberry Pi needed to be supplied by separate voltage regulator circuits to avoid one hardware component from siphoning the current from the other component. The schematic below represents two LM2596T circuits from their respective datasheets sharing the same input but having separate outputs.

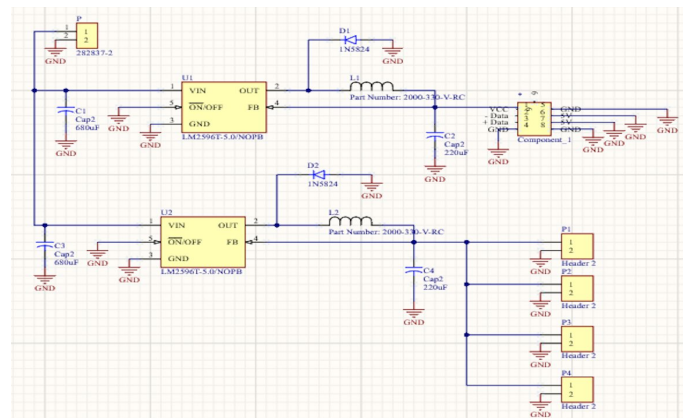


Figure VII: PCB circuit schematic

Before our CDR presentation, our team was able to test one LM2596T circuit to see how well our game board functioned with that input. The picture below depicts the successful test we ran with our circuit. One LM2596T circuit was able to supply enough power to all 4 LED boards. Unfortunately, the fabricated PCB did not arrive before CDR and we were not able to populate our PCB onto our game board.

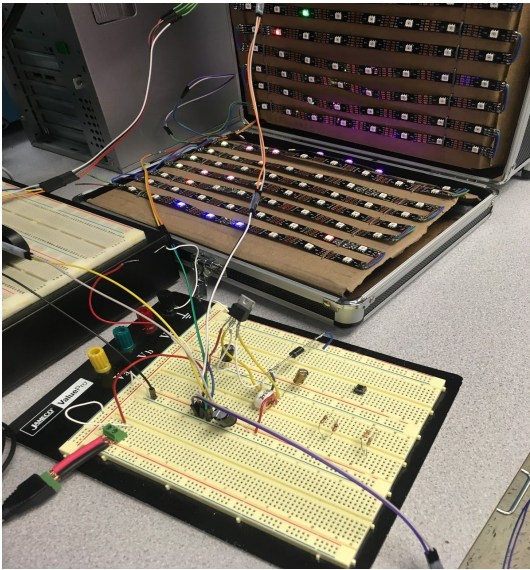


Figure VIII: Electronic Hardware Component implemented with design

### C. Product Functionality

At the time of CDR, the Tetra Board was fully functioning and the product was demonstrated to our evaluators in person. Our speech recognition required a lot of troubleshooting and improvement throughout both semesters. We tried multiple voice-recognition libraries, including AWS Transcribe and Google Speech to Text. Our solution to reducing noise was to preprocess audio using a Butterworth filter and post-process using a database of common mistakes received from Google Speech to Text. Moreover, at the beginning, the LEDs did not work well and we thought our Pi was not working properly. However, we found that using SPI and I2C pins instead of I/O pins on the Pi provided a much better, reliable connection to the LEDs. We did not run into any issues implementing software to the Pi or software to display the LEDs. In the end, all of our components in the block diagram were working properly.

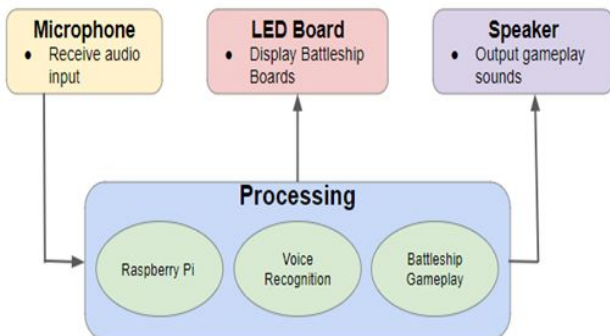


Figure II: Block Diagram Shown Again

### D. Product Performance

At the time of CDR, we had met most of our products specifications. Looking at table I from earlier we can see the different specifications that our product was going to meet. We successfully implemented alphanumeric commands from Alpha 1 through 8 to hotel 1 through 8. We had fully implemented our speaker that was able to sound off on a “hit”, “miss”, and a “sunken ship” with their own respective sounds. For our lights on the board, we implemented dotstar LEDs which only require a 5V source for all 256 LEDs in order to function properly with all different colors at our desired brightness. The entire product is housed inside a double-sided briefcase with all four boards implemented inside of it. A picture of the boards housed inside the briefcase can be seen above from figure VI. The one specification that we did not have at the time of CDR was the external 5V wall mount. We had planned to implement this for FPR, as it would have been extraneous to implement the wall mount when the PCB for our product was implemented inside our product. We had also planned to use an external wall mount of 12V instead of 5V as our product had changed with the use of our PCB now being a step down voltage regulator at 5V, so it wouldn’t have made sense for us to continue using the 5V wall mount. All in all, our product had met most of the specifications from table I with just a slight change in the Power specification. In terms of using the product itself, We had very fast response times from when commands were input into the microphone to have the right position on the 8x8 grid light up. We were able to implement different colors and also had a robust speech recognition working for our set of commands that were needed to play battleship on our product.

### V. CONCLUSION

Looking forward, we plan to integrate the PCB together with the entire product to ensure that it works as a whole. We will also continue adding components such as the other two boards and more options within the battleship game. We will look to play around with adding more games to Tetra Board, such as Connect 4 and Tic Tac Toe and possibly a game that we come up with ourselves. We may also add more manually operated components that could be used in the case that someone with less severe physical impairments were in a noisy location and still wanted to play.

### ACKNOWLEDGMENT

Our group would like to thank our advisor, Professor Polizzi, for giving us phenomenal insight and advice throughout our project.

### REFERENCES

- [1] Rollins, Brandon. “How to Develop Visually and Physically Accessible Board Games.” *Brandon the Game Dev*, 13 Nov. 2019



- [2] Burgess, Phillip. "Adafruit DotStar LEDs." Adafruit Learning System, 26 Nov. 2019
- [3] IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7," in *IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)*, vol., no., pp.1-3951, 31 Jan. 2018 doi: 10.1109/IEEESTD.2018.8277153
- [4] IEEE Std 802.15.1-2005 – Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)". IEEE Standards Association. doi:10.1109/IEEESTD.2005.96290. ISBN 0-7381-4707-9. Retrieved June 30, 2011.

## APPENDIX

### A. Design Alternatives

#### 4-Pin Diode LED

The main design alternatives for our project were the choice of LEDs for the Battleship boards. Our initial implementation of the LEDs was to use four pin LEDs. These LEDs were not addressable by the Raspberry Pi, which would cause many problems with running the Battleship gameplay software. We could not send commands to these LEDs to light them different colors. The LEDs were also limited in color selection. The main problem with these LEDs was its power consumption. They did not have an option to adjust the brightness from the Raspberry Pi and more importantly they did not include multiplexing. These LEDs would consume much more power than we would want over the period of a 10 to 20-minute game of Battleship which is not reasonable. This would make the LEDs very prone to burning out after extended amounts of time of being lit up.

#### Neopixels

Another design alternative that we tried to use were the Neopixels. The Neopixels were a better option than the 4-pin diode LEDs as they were individually addressable by the Raspberry Pi and offered multiplexing. This solved the two issues with the 4-pin diodes as the Raspberry Pi was not able to individually light up each LED and copious amounts of power were consumed constantly. The problem we faced with the Neopixels was its internal clock when being addressed by the Raspberry Pi. The clock signals were not synced, and this would cause inaccuracies in the timing of commands to be processed by the Raspberry Pi until an LED would light up. The complexity of the clock timing made the Neopixels a bad choice for our project.

### B. Technical Standards

The IEEE 802.11 protocol specifies media access control and physical layer protocols for communication via wireless local area network communication which our voice recognition software and text to speech software utilized.

#### 1074-2006 - IEEE Standard for Developing a Software Project Life Cycle Process (Game Software Development)

- Stage 1: Planning and Requirement Analysis
- Stage 2: Defining Requirements
- Stage 3: Designing the Product Architecture
- Stage 4: Building or Developing the Product
- Stage 5: Testing the Product
- Stage 6: Deployment in the Market and Maintenance

The 1074-2006 standard defines how a Software Project Life Cycle is created and can be used by anyone tasked with creating and maintaining a project either entirely composed of or merely containing software. As shown above, there are six stages of the process which end in the deployment and maintenance of the system. Tetra Board obviously did not get to that stage but when programming the strategy encouraged by this standard was loosely followed.

The IEEE 1003.1 standard defines operating system environment and interface guideline standardizing command interpretation as well as utility programs to allow for portability of source code. It is made up of four main parts, general terms, concepts, interfaces, and utility and C header conventions. [3] When using the Raspberry Pi linux was necessarily used to navigate and operate the software for Tetra Board.

The IEEE 802.15.1 standard specifies that it defines physical layer (PHY) and Media Access Control (MAC) specification for wireless connectivity with fixed, portable and moving devices within or entering personal operating space. [4] In the case of our product, we had built in bluetooth capabilities through our microcontroller that was able to connect with the speaker if we needed it to be wireless. Currently the onboard speaker of the product is a part of our product, but the bluetooth capabilities are available if a customer would want to freely move the speaker around, granting extra portability of the product and more personal customization.

### C. Testing Methods

For testing methods, we have implemented ways to test if we are taking in the correct inputs from the microphone. One of the ways to test this was to develop test cases in the voice recognition software to tell us what words and letters the microphone is picking up. The voice recognition software would print out whatever we spoke into the microphone to see if it would pick up the words we were saying. Sometimes the microphone would pick up words instead of numbers or unwanted spaces and to combat this, we implemented code that would parse these unwanted spaces and numbers to decipher whether it was a valid or invalid command.

In addition, we also had code in our Battleship software that showed the player's ship and attack board. The board would update based on the inputs from the microphone and be processed by the microcontroller. This helped us make sure that the LED board was taking in the correct data from the

## SDP20 – Team 7

Raspberry Pi so the correct LED would be lit up corresponding to the visual display from the software in the command prompt.

#### D. Team Organization

Varak has been taking charge of doing most of the administrative work to help keep us on pace for deadlines like bench side meetings, PDR, MDR, and weekly meetings. He has additionally been helping debug complications of our project. Vincent has been doing most of the ordering for the parts, facilitating the software for the voice recognition, and making sure the software and hardware are compatible. James has been facilitating the software for the Battleship gameplay in order to make sure it is compatible with the voice recognition. He and Vincent have been working together to make sure the software is working properly. Aleck has been working on the hardware side with the LED boards and making sure the hardware is compatible with the software.

So far, our current team organization is working well. The team organization was not assigned at the beginning of the semester, rather individual team members took the initiative, and these were the eventual roles that stemmed from that. With everyone working on their respective tasks, we can make good progress on our project. We are working well in our current roles and there have not been many problems as team members are willing to help others out when problems arrive. Overall, the current team structure is proving to be successful and we hope that we can keep this up for the final product as there will be some differences in tasks for each team member.

#### E. Beyond the Classroom

Aleck has developed skills with hardware debugging and figuring out problems with the LEDs when they are not functioning properly. James became proficient in python and continued to improve upon his software skills. Varak learned how to regulate power supply amongst all the LEDs and how to implement speaker control during gameplay. Vincent has developed skills in speech recognition and become proficient in Python while doing so. He also learned how to set up a Raspberry Pi for development and how to interface different components on the Pi. He has developed software and hardware debugging skills as a result.

When it comes to projects that incorporate both hardware and software, debugging is a necessary skill to have. Regardless of skill level for either, it is important that the hardware is compatible with the software and vice versa. As professionals, we want to remove all the bugs that we can and provide a good user experience for those testing our product.

#### F. Budget

For our project, we have used approximately 40% of our overall budget up to this point. This is not too bad as our projected cost for the other two boards and the PCB should not bring us over the total budget given to us by the ECE

department. So far, our spending can be seen in Table III, which is shown below.

Raspberry Pi 4	\$48.01
DotStar LEDs x2	\$49.90
HDMI	\$5.98
2-sided Briefcase	\$27.16
Microphone	\$12.99
Adaptor	\$7.99
Micro SD Card	\$9.40
Testing Components	\$23.87
<b>Total</b>	<b>\$185.30</b>

Table III: Costs