Sensor Suit

Authors: Amelia Lopez, Bethel Yohannes, Sophia Li
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— The Sensor Suit system aims to design and implement a wearable haptic system that provides physical feedback catered to players of the *The Last Spartan* desktop game. Our solution seeks compete with industry standard Haptic suits on the basis of its sizeability, comfort and cost effectiveness.**

***Index Terms*—Arduino, Gaming, Haptics, NodeJS, vibrotacticle, The Last Spartan, vest, coin motor, RGB lights**

1 INTRODUCTION

Virtual games are a story telling medium. They use audio, visuals, and user interaction to provide a form of entertainment that can be enjoyed by virtually everyone. Recently, the gaming industry has moved towards increasing immersion in games by providing virtual reality headsets, haptic feedback devices, controller-free experiences, better resolution in graphics, and more. Haptic feedback devices are a form tech that allows a user to feel a physical stimuli from their game actions in real time.

In this design report, we present a haptic vest specifically designed for the open source desktop game, *The Last Spartan*[1]. The vest is fitted with a range of vibration motors and RGB lights that will allow users to feel more immersed in the game by adding physical stimulation cues to this typically desktop-confined game experience. We also discuss the design, implementation, and testing of the vest system along with its impacts on game immersion.

2 USE-CASE REQUIREMENTS

The use-case requirements for our haptic vest implementation were driven by our objective to provide users with the most comfortable, immersive gaming experience while considering economic and social impacts to the user. As such, they can be divided into 4 categories: run-time and latency, feedback points, overall experience, and beyond engineering considerations.

2.1 Run-time and Latency

For the user to feel fully immersed, the initialization of the haptic response (i.e. vibration of a motor and/or RGB lights turning on) needs to feel indistinguishable from real-time in-game actions on screen. We also want to the system to have a long battery life, limiting how often our users will have to stop playing to replace the batteries. This minimizes the negative environmental impacts associated with the disposal and over-consumption of batteries.

2.2 Feedback Points

The user should feel haptic feedback throughout their entire torso and differentiate between which feedback points are active during game play. This means it's essential to place feedback points on the front, back, and sides of the torso. This allows for sufficient stimuli coverage while also providing enough space for the user to distinguish between the activated and deactivated feedback points.

2.3 Overall Experience

The goal of the wearable system is to enhance the user's gaming experience without causing any distractions or interruptions during their real-time game play. By providing an improved experience, the user can become fully immersed in the game's story and their actions within the game.

2.4 Beyond Engineering Considerations

The system has to be built with socioeconomic and cultural impacts in mind. Thus, it is imperative that the wearable system is able to be customizable to ensure a range of body types and genders are able to physically fit into the solution comfortably. Currently, existing gaming wearable solutions are only fitted for male body types, but our solution should be inclusive to all genders and body sizes. Additionally, the entire product should cost less than $200. This is to compete the existing haptic solutions for gaming applications, which can range anywhere between $300 to $8000.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Our system includes two main components: a software system which hosts the game and interprets the in-game actions and a hardware component in the form of a vest which is embedded with motors and RGB lights. The overall system can be seen in Figure 7.

3.1 Software Architecture

Our Software architecture primarily consists of JavaScript code that helps acquire the real time game ac-

tions of the player and a Node JS backend that relays this information to the Arduino IDE.

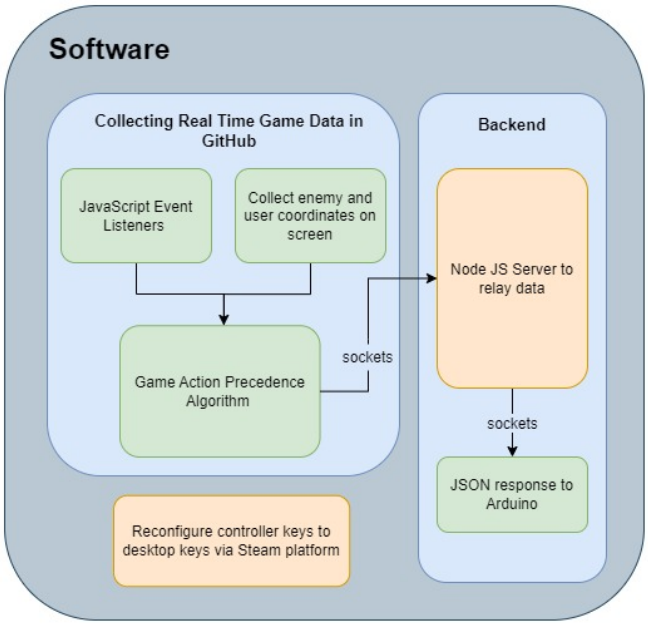

Figure 1: Shows our Software Architecture Diagram. Yellow boxes represent the components we are assembling while the green boxes represent the components we are designing.

3.1.1 User Input & Re-configuring Controller Keys

The traditional version of the game *The Last Spartan* was designed for desktop/laptop use, utilizing keyboard keys such as W-A-S-D, J, K, P, SPACE, and ENTER for avatar control. However, our vest solution aims to enhance immersion, and we realize that tethering users to the keyboard may not be ideal. To address this, we have opted to use a wireless Xbox controller as an input device that connects via Bluetooth to provide a keyboard-free experience, with a wireless range of up to 6+ ft. This particular controller was chosen for its widespread use and convenience.

To reconfigure the Xbox controller keys, we used the STEAM platform. STEAM allows you to reconfigure the controller keys to desktop keys via the controller settings. The mappings are listed below:

- 'w-a-s-d' keys → left-most joystick
- 'k' key → left trigger/left bumper
- 'j' key → right trigger/right bumper
- 'p' key → 'options' button
- 'SPACE' key → button A
- 'ENTER' key → button B

3.1.2 Game Data Acquisition

We are collecting the real-time game actions of the player by incorporating JavaScript event listeners into the game code, which listen for four specific events, namely getting hit by a small enemy, getting hit by a large enemy, experiencing low health, and jumping forcefully. We similarly track enemy and user coordinates on screen to deliver a differentiated haptic response to a large sized enemy as opposed to a small sized enemy.

We designed a Game Action Precedence Algorithm to avoid overwhelming our player with a variety of haptic responses when multiple game actions are taking place simultaneously. Table 1 helps to identify the event that takes precedence given multiple game actions are being played at the same time. In the event where the player dies, death will take precedence over all other game actions. Whereas, low health will have the least precedence and a forceful jump event will have the highest priority over the remaining game actions. We utilize the socket.io library to pass on this information to our Node JS server.

Table 1: Indicates the game action that will take precedence given two or more events have taken place at the simultaneously. In the event where the player dies, then death will take precedence over all other actions. Redundant action comparisons between the same events have purposefully been omitted

	Jump	Small Hit	Large Hit	Low health
Jump		Jump	Large Hit	Jump
Small Hit	Jump		Large Hit	Small Hit
Large Hit	Jump	Large Hit		Large Hit
Low health	Jump	Small Hit	Large Hit	

3.1.3 Backend & Server Architecture

The HTTP and socket.io libraries further aid us in creating a JSON response to be sent to the Arduino IDE. The haptic algorithm cases on the parsed JSON response and activates specific motors in the fashion described within Table 2 according to the prioritized game action.

3.2 Hardware Architecture

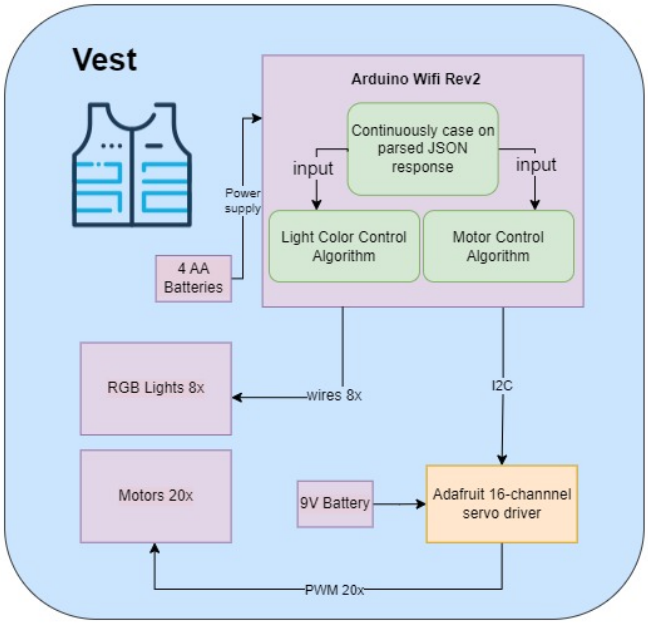

Figure 2: The hardware subsystem fully contained within the wearable vest

The hardware subsystem contains the physical Arduino WiFi module as well as the RGB lights and haptic feedback systems for the vest. All of hardware is built into the wearable vest.

(a) Arduino WiFi Rev2

The Arduino IDE is run from the computer while the physical microcontroller is wirelessly connected to the computer through Bluetooth.

Arduino code: The Arduino continuously looks for a JSON response from the NodeJS server. Once it receives a response, it cases on the action it should replicate. Each action has it's own unique pattern initialized in the startup. The unique responses will each have Class Response (a custom class for this project).

Once an haptic pattern is activated, it begins the algorithm to control the motors such that the physical feedback follows the designated patterns. For example, a forceful jump may cause the six lower motor systems to vibrate for 0.5 seconds, followed by a wave of the middle size motor systems, followed by a wave of the uppermost motor systems. These "waves" of sensation will travel from the bottom of the torso through to the top of the torso, simulating the sensation one feels when they jump from a elevated place.

Arduino hardware: The Arduino board will receive the signals from the Arduino IDE. The microcontroller will drive an SparkFun 16 output I/O expander (used as a motor driver) which will in turn drive 16 of the vibration motor systems located on the front and back of the torso. The Arduino uses I2C to communicate with the motor driver.
In addition to the motor driver and remaining motor systems, the Arduino board will also control the RGB lights on the front of the chest.

(b) Vibration Motor Systems

The beginning of the vibration motor systems starts with the Arduino output from the motor control algorithm. The algorithm signals the first wave of vibrations, driving the SparkFun 16 Output I/O Expander to power on the specific motor systems. The Arduino to servo driver connection is made through I2C. The motors can run up to 12000 RPM when fully powered.

(c) RGB Light System

The RGB light system is controlled thorough the Arduino's digital I/O pins. The system we choose allows us to control how bright the lights will be and what colors the system will have.

4 DESIGN REQUIREMENTS

The design requirements for our haptic vest implementation are built off our use-case requirements, diving deeper into the quantitative specifications as well as integration of the subsystems into the overarching system. As such, they can be divided into 5 categories: in-game responses, run time and latency, motor and light responses, wireless range, and sizeability.

4.1 In-Game Responses

The system should respond uniquely to at least 4 different in-game actions while playing. The response should be in the form of a motor turning off/on at a specific frequency, as well as the RGB lights turning off/on at varying colors. The combination of the motor and light response throughout the front, side, and back of the vest should allow the player to become more immersed in the game by believing a specific in-game actions can be simulated in real life in the form of stimuli.

The 4 in-game actions we want our system to respond to are listed below:

- Low health
- Getting hit by a small enemy
- Getting by a large enemy
- Forceful Jump

4.2 Run Time and Latency

The system should last at least 4 hours of gaming time. This is in line with the battery life of existing haptic solutions. In order to ensure this, the battery capacity to both systems (the Arduino and the servo driver) must handle the current drawn from the motors, which average at about 60mA each (since they will not be on all the time) and light strips which average at about 120mA.

Moreover, in order for the response of the system to feel indistinguishable from the in-game actions, the the latency between an in-game action and a haptic response should fall anywhere between 16ms and 100 ms.

The minimum bound of this range was determined based on the monitor refresh rate, which is approximately 16ms and influences the smoothness of on-screen motion by varying the frequency at which the image is refreshed on the screen. The maximum bound was determined by the time it takes for the human brain to perceive something on-screen as instantaneous (100ms) [4].

4.3 Motor and Light Responses

The motors in the system must operate at vibration frequency range above 3000 RPM. This is the minimum RPM where vibrations become noticeable on the surface of the skin. We found this information through testing motors in our palms at different RPMs over 10 times.

For the light system, it's imperative that they are RGB lights that shine at a range between 300-500 lux. We chose this range in order to ensure that the lights are bright enough to shine through the lining of the vest in a typical room lighting setting. We also didn't want the brightness of the lights to be too intense as to cause a glare for the user while playing with the system.

4.4 Wireless Range

For the user to have a fully immersive experience, the entire system must be completely wireless and operate within a range of 6 ft to the laptop which the game is being played on. This wireless experience will allow the user a full range of motion, 360°, without being interrupted. In order to ensure this, the Arduino controlling the motors and lights will be secured on the vest, receiving information via Bluetooth from the laptop. The input controller device will also be fully wireless and connected via Bluetooth, functional from a distance of up 6ft as well.

4.5 Sizeability

The adjustable wearable system must extend and contract up to 4 inches around the side torso and shoulders in order to accommodate multiple body types. We chose 4 inches for this metric since our vest size will be a Medium, and the next size up/down for both men and women corresponds to about 4 inches.

In order for a user to extend/contract the vest, they can simply detach and attach the Velcro strips located at corresponding lengths, which secure the front and back pieces of the vest.

5 DESIGN TRADE STUDIES

Several of our decisions for the game, the software implementation, and the hardware implementations depended heavily on the design requirements. We outline our decisions below.

5.1 Why *The Last Spartan*?

We decided early on that our project would be hardware-focused. Once we settled on a haptic feedback system to improve game immersion, we had to choose a game. We created a decision matrix looking at various games, at first focusing on racing games. In the end, *The Last Spartan* provided an open-source GitHub[1] with readable code, an interactive game, and enough in-games actions to allow for unique haptic responses.

Figure 3: The intro screen with directions to start the game.

5.2 Software Decisions

Node JS provides a stable and free network to host our webapp and extract the game data. NodeJS is able to host web applications with a backend server architecture. This is a better alternative to many popular platforms like Heroku, Google Cloud, AWS, or similar platforms that allow users to host backend servers.

5.3 Hardware Decision

Finding the correct hardware was integral to our project. We settled on the Arduino Uno WiFi Rev2, an SparkFun I2C 16 Output I/O Expander, vibration motors, and Adafruit Digital LED Strips.

5.3.1 Arduino Uno WiFi Rev2

Our project applications revolve around controlling and organizing a large system of motors and lights. This encourages us to choose a microcontroller with large amounts of output. We do not require a large processing power nor complex interfaces into our hardware. This means we were not required to choose a RaspberryPi or Jetson which projects with complex computation and processing require. To accomplish the wireless requirement, we selected the Arduino WiFi.

5.3.2 SparkFun I2C 16 Output I/O Expander

The SparkFun output expander allows us to drive our systems of motors. With its I2C capabilites, we are able to drive 16 systems of motors from 2 of the Arduino's outputs (SLC / SDA). Our original conception for an I2C interface for the rest of the system was an Adafruit 16-Channel PWM Servo Driver. However, the servo driver is designed for PWM specific servos while we needed a PWM option that worked with our 2 wire vibration motors.

5.3.3 Vibration Motors

We chose the "Mini Vibration Motors DC 3V 12000rpm Flat Coin Button-Type" as the motor type which will be installed around the entire vest. This decision came about because of the motor's small size (⌀10mm), its compact design which doesn't stick out as much as the other popular motors for haptic feedback (namely the Eccentric Rotating Mass vibration motor), and the max RPM value of 12000 which was impressive for its dimensions. A higher RPM will allow us to create an immediate, strong stimuli for our system that can be useful for short-burst actions such as a forceful jump.

5.3.4 Adafruit Digital LED Strip

The Adfruit Digital LED strip allows us to control the RGB channel of up to 30 LED's per strip using digital pins on the Arduino. We chose this specific design because of its flexible material that we can fit and secure to the vest, it's scalability, as well as it's compatibility with the "FastLED" library on the Arduino IDE.

5.4 Sensor Placements

In our use case, the placement of motors on the torso is a key factor towards creating a haptic feedback system that can provide users with a realistic and immersive experience. Thus, when it came time to determine which points on the torso we should place motors over, we relied heavily on prior research into haptic systems and spatial acuity. Spacial acuity is the ability to discern between two stimuli close in space. For instance, if two pins were placed on a single point in a person's hand, the person would perceive it as though one giant pin was placed on their hand. But if the pins were moved apart, the person would be able to distinguish which pin is where on the hand with a higher accuracy.

For the purposes of our project, we were looking for points on the body with high spatial acuity. By stimulating the high spatial acuity areas on our body that our suit will be in contact with, we could convince our brain to experience touch sensations that coincide with the actions taking place within the game, thus making for a reliably immersive experience.

From Mancini's article [3], we located feedback points on the back of the torso where spatial acuity was high, and therefore would be ideal placements for our motor systems. From Kang's article [2], we were able to do the same for the front of the torso. Due to the physical restrictions of our vest, the side torso placements were already predetermined since there is limited space due to the adjustable straps that are fitted on the side torso of the vest.

Our final motor placements are shown below. The majority of feedback points have 2 motors, with the exception of the feedback points over the heart and right lung which have 4 motors to a single system. The reason behind this is that based on Kang's article, whereby these regions were shown to have exceptionally high spatial acuity. We are going to take full advantage of this fact by localizing our low health event algorithm over this area, in order to stimulate an increased adrenaline rush such as a "thumping" effect over the heart.

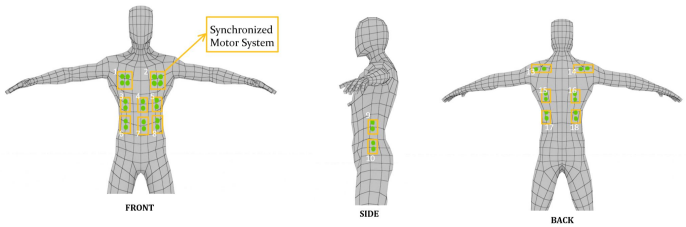

Figure 4: A total of 18 motors systems are shown (8 in the front, 2 on each side, and 6 on the back)

6 SYSTEM IMPLEMENTATION

As seen in our Architecture Section 3 we have many hardware components that have to be wired together within the suit. The components with the most wiring considerations are the Arduino WiFi module and the Sparkfun I/O Expansion seen in Fig 2.

6.1 Subsystem 1: Arduino Uno Wifi Rev2

The Arduino will control the light systems, 2 of the motor systems, as well as communication with the Sparkfun I/O expander. The light systems will be sorted into the

lights on the shoulders, the on the front-left of the torso, and the lights on the front-right of the torso. Each of these lights are connected to either the digital I/O pins or the analog I/O pins that the vibration motor systems utilize. The vibration motor systems are connected to PWM pins 6 and 9. The vibration motors require PWM frequency to function at a specified RPM. Lastly, to help drive the remaining 16 motor systems, the Arduino has it's SCL / SDA pins connected to the I/O expansion board for establishing an I2C connection.

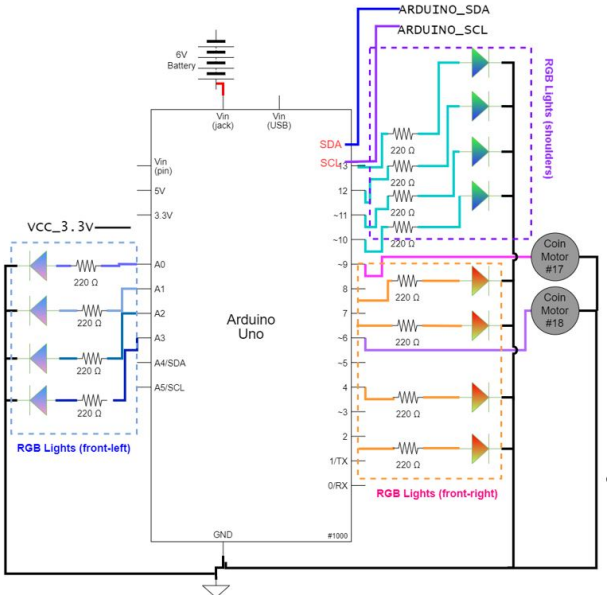

Figure 5:

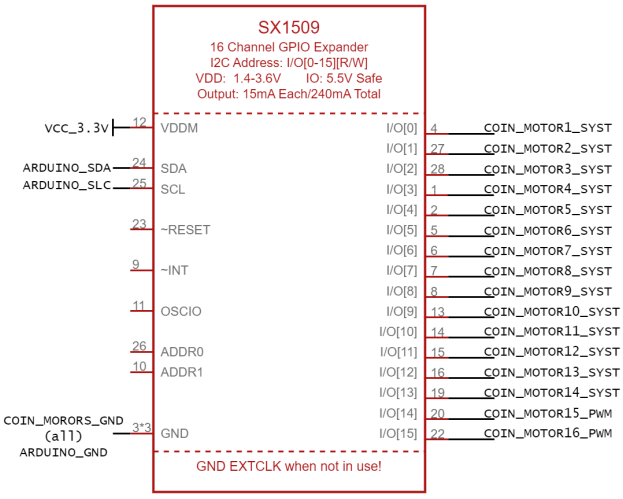

Figure 6:

6.2 Subsystem 2: Sparkfun Output I/O Expander Breakout

The second main subsystem is the I/O expansion breakout. This breakout has its SCL / SDA pins connected to those of the Arduino so it can receive the signals on which motor subsystems to activate. It then uses it's 16 output pins to send signals the motor systems. These signals control how fast the motors should spin. These considerations are important considering how we want different sensations to feel as seen in Table 2.

6.3 Additional Subsystem

Once we extract the game action and send it to the hardware to interpret and control the motors, it is important to know *why* we want specific motors to be activated. One of the use-case requirements includes creating at least four unique haptic responses. To do this, we plan to create hapic feedback for following in-game actions: low health, getting hit (small enemy), getting hit (large enemy), and forceful jump.
For each response, we can choose which motor systems should respond and in which order. These motor systems are labeled in Figure 4. Table 2 goes in depth into how the vibration motor feedback system will respond to each in-game action.

Table 2: The sensations the vibration motors will simulate

	Wave1	Wave2	Wave3	Wave4
Low Health	2	2	2	2
Small Hit	4	3,5,7	1,2,6,8	
Large Hit	4,1,2	3,5,7	3,5,6,8	9-12
Forceful Jump	6-8, 17-18 10,12	3-5,15-16 9,11	1-2,13-14	

- For low health, we want to focus on areas around the chest. When the Spartan in the game drops below 25% health, the motors around the chest should begin to beat. The RGB lights will also begin to flash red. As the character drops below 10% health, the frequency of the motors and lights will speed up to simulate a quicker heartbeat due to adrenaline of being in a low health situation.
- Getting hit by a small enemy will cause a small propagation of force from the center of the chest outwards. This expanding force can simulate being hit and the percussion expanding across the torso.

- Getting hit by a large enemy will cause a larger propagation of force from the center of the chest outwards. This expanding force can simulate a large impact and the percussion expanding across the torso. The hit is also from higher on the torso due to the larger enemies being taller.
- A forceful jump will provide a wave of feedback emanating from the bottom of the torso and reverberating to the top of the torso. This simulates the impact someone will feel when they jump from some distance off the ground. When they land, the impact will travel from their feet through the rest of their body.

7 TEST & VALIDATION

To ensure we are meeting and keeping our design requirements, we will implement a variety of tests on our software, hardware, and fully integrated systems.

7.1 Testing, Validation and Metrics for Hardware

For our hardware components, we aim to test the battery life and motor sensation.

7.1.1 Battery Life

To measure that the battery life (t in hours) of the system meets our goal of at least 4 hours, we are going to divide the battery capacity (C in mA * hours) by the sum of the current drawn (A_i in mA) from our N hardware components attached to the Arduino. Since we have two systems powered by a battery, we will repeat this process for the Adafruit servo driver. The equation can be seen below.

$$t = \frac{C}{A} = \frac{C}{\sum_{i=1}^{N} A_i}$$

7.1.2 Evaluation of Motor Sensation

Feedback from different members of the gaming community will be essential in evaluating how effectively our motor configurations simulate different sensations. We therefore plan to recruit around 20 gamers to test the immersive experience of our haptic suit. We will have our participants play *The Last Spartan* game without wearing our haptic suit initially. After this first round of gaming, the participants will be asked to wear our haptic suit and play the game a second time. Once both rounds of gaming are complete, we will have our recruits fill out a 10 question survey about the differences in their experiences of playing the game under the two separate conditions. Each question within the survey will contain a 5 point likert scale. A score of 1 on this scale indicating that the user cannot feel any sensation while a score of 5 suggests that the user finds the sensation they are feeling to be uncomfortable or painful. We hope to achieve a failure rate of less than 15% where we anticipate at least 3 out of our total of 20 participants will not find our system immersive or better than the standard desktop gaming experience.

7.2 Testing, Validation and Metrics for Software

For our software components, we aim to test the JSON data transmission as well as our ability to parse the JSON data.

7.2.1 Arduino IDE Receives JSON Response

Our team plans to utilize a variety of tests to validate the proper operation of our JavaScript event listeners and our Node JS backend. We want to assure that 100% of our game action data is being transferred as a JSON response from our Node JS backend to our Arduino IDE. To validate this, we will be printing the JSON response being received by our Arduino on the Arduino's serial monitor and compare it to the JSON response we originally sent out. This comparison will allow us to detect and avoid discrepancies between the two responses and therefore prevent data loss.

7.2.2 Correctly Parse Game Data to Motor Algorithm

We also will be testing that the active motors are correctly configured against their corresponding game events. This will be configured according to the specifications highlighted in Table 2 For instance, the game action of a forceful jump will have a motor configuration where a wave of vibrations that spread from the bottom of the torso to the top of the torso. Whereas the motor response for low health will cause the motors around the chest to beat and the RGB lights will also begin to flash red.

In order to accomplish this, we will be testing whether we are correctly parsing the game data from the Node JS backend to our Arduino IDE. Ideally, all four of our unique haptic responses should be configured to their corresponding motor configurations by our minimum viable product deadline.

7.3 Testing, Validation and Metrics for the entire Haptic Feedback System

The testing, validation and metrics of our entire feedback system focuses on assessing the cost effectiveness of our solution while achieving an appropriate wireless range and latency. We have successfully met our metrics for cost effectively designing a haptic suit as the total cost of our parts sum to $199.

7.3.1 Wireless Range

We plan to achieve a wireless range of up to 6 ft for the player. This metric is restricted to be within 6ft because the user will not be able to see the game screen properly when moving beyond this distance. We plan to measure the wireless range of our system by recording the distance at which the user is can still play the game.

Our initial tests with the Xbox controller revealed that the game responded to the controller's action from father than 30 feet away. This distance exceeds our required 6 ft as the tester was no longer able to see the screen at 30 ft.

7.3.2 Latency

We aim for a latency < 100 ms. To achieve this, we will print out when the server sends an in-game action to the Arduino. We will have our phones recording in slow motion so we are then able to capture when the printing of the response occurs and the delay until the motors start vibrating. As per our TA's recommendations, we also plan to use a slow motion camera to test latency.

7.3.3 Impact on Game Performance

Ideally, our haptic vest solution will have a positive impact on the game performance and enjoyment of the user while playing *The Last Spartan*. Therefore, we plan to recruit more players to test different game metrics such as the total number of enemies defeated, how long each gamer ended up playing for and how much they enjoyed the immersive experience. The methodology for our investigation of game performance and player satisfaction will largely be the same as that of our evaluation of motor sensation. The only difference lies in the performance variables we are measuring for. The questions on the survey will therefore reflect these new measures.

8 PROJECT MANAGEMENT

8.1 Schedule

In order to segment our project into manageable parts that we can get done by the end of April, we created a schedule broken down by tasks which is shown in Fig. 8.

8.2 Team Member Responsibilities

Each team member's unique responsibilities are listed below:

- Amelia is responsible for the human integration component and hardware communication. This includes research areas on body for motors, research PCB for motor attachment, testing and configuring RGB lights into vest, and ensuring the Arduino is working wirelessly.
- Bethel is responsible for the software components. This includes integration with the game, extracting game data, and transferring that data to the Arduino IDE.
- Sophia is responsible for the hardware components. This includes working with the Arduino breakout board, vibration motor system testing, designing the hapic algorithms, and receiving data from the NodeJS framework.

We designed the task assignments based on interest, expertise, and ability to parallelize the tasks. This allows us to work on separate components of the project at the same time such that a minimal product is produced by each of our milestones.

8.3 Bill of Materials and Budget

The bill of materials and budget per robot can be found in Table 3. The primary expense of this project is the vest and Arduino WiFi device. The aditional main expences are the RGB lights and the motors.

8.4 Risk Mitigation Plans

Given the current state of our project, we foresee possible challenges in keeping wiring of the system compact due to the volume of connections in our vest. Our vest will be also be moving around as users move and take on / off the vest. This constant movement could disconnect parts so we need to come up with a way to quickly and easily access specific motor systems and elements to fix any issues. To mitigate this, we have to balance wire length, making it long enough to expand and contract with the different sizes of users while not getting tangled. We will also need to use color-coordinated ribbon wires that correspond with specific systems in order to locate them with ease.

We also foresee integration being a significant time investment. We want to ensure that once we get the software and hardware components communicating, the latency is not slower than what we specified in our design requirements and our power consumption doesn't go over the battery capacity. We plan to mitigate this by having multiple small checkpoints in which we continuously test these components, and alter our code to reduce the rate of data transmission in order to meet our latency requirements.

9 SUMMARY

We aim to build a wireless haptic vest that connects to the desktop game *The Last Spartan* and improves game immersion through a vibrotactile feedback system. We will do this by building essentially two products: a desktop game that has a Node JS backend collecting data, and haptic vest with Arduino WiFi controlling vibrating motors and

RGB lights. In turn we hope this solution provides a cost-effective alternative which increases user immersion in *The Last Spartan* and allows a user to distinguish unique haptic-response for each in-game action.

Glossary of Acronyms

- MQTT – Message Queuing Telemetry Transport
- OBD – On-Board Diagnostics
- RPi – Raspberry Pi
- PWM - Pulse Width Modulation
- RPM - Revolutions per Minute
- RGB - Red, Green, Blue
- LED - Light Emitting Diode
- IDE - Integrated Development Environment
- PCB - Printed Circuit Board

References

[1] Michael Ferron. *js13k-TheLastSpartan*. Sept. 2020. URL: https://github.com/ferronsays/js13k-TheLastSpartan.

[2] Kang D. Kwon O Lee CG. "Pneumatic and acoustic suit: multimodal haptic suit for enhanced virtual reality simulation". In: *Virtual Reality* (2023). DOI: https://doi.org/10.1007/s10055-023-00756-5.

[3] Flavia Mancini et al. "Whole-body mapping of spatial acuity for pain and touch". In: *Annals of Neurology* 75.6 (2014), pp. 917–924. DOI: https://doi.org/10.1002/ana.24179. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ana.24179.

[4] Robert B. Miller. "Response time in man-computer conversational transactions". In: New York, NY, USA: Association for Computing Machinery, 1968. ISBN: 9781450378994. DOI: https://doi.org/10.1145/1476589.1476628.

Table 3: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Parts Incorporated in Device					
Arduino Uno Wifi Rev2	ABX00021	Arduino	1	$53.00	$53.00
SparkFun I/O Expander	SX1509	Sparkfun	1	$6.95	$6.95
Vibrating Motor (12000 RPM)	NA	Amazon	2	$13.99	$27.98
Vest with Adjustable Straps (men's medium)	PWV-BPC-BS	PWVests	1	$36.99	$36.99
Adafruit RGB LED strip	2578	Adafruit	2	$19.99	$39.98
4 AA battery holder (2 pack)	NA	Amazon	1	$5.99	$5.99
AA battery pack (24 count)	NA	Amazon	1	$18.27	$18.27
set of wires 200 pieces	NA	Amazon	1	$9.99	$9.99
					$199
Parts for Testing / Idealization					
Xbox wireless controller	XS series	Target	1	$49.97	$49.97
TOTAL					$250.12

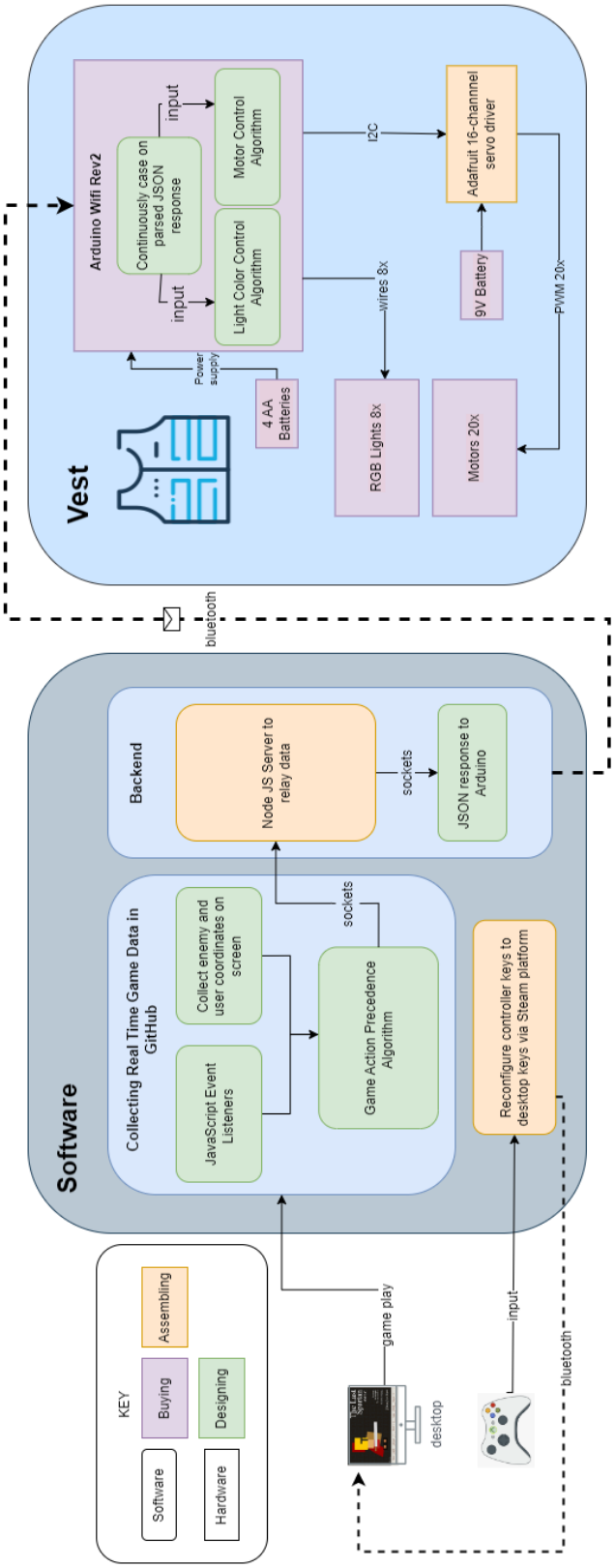

Figure 7: A full-page block diagram of system implementation

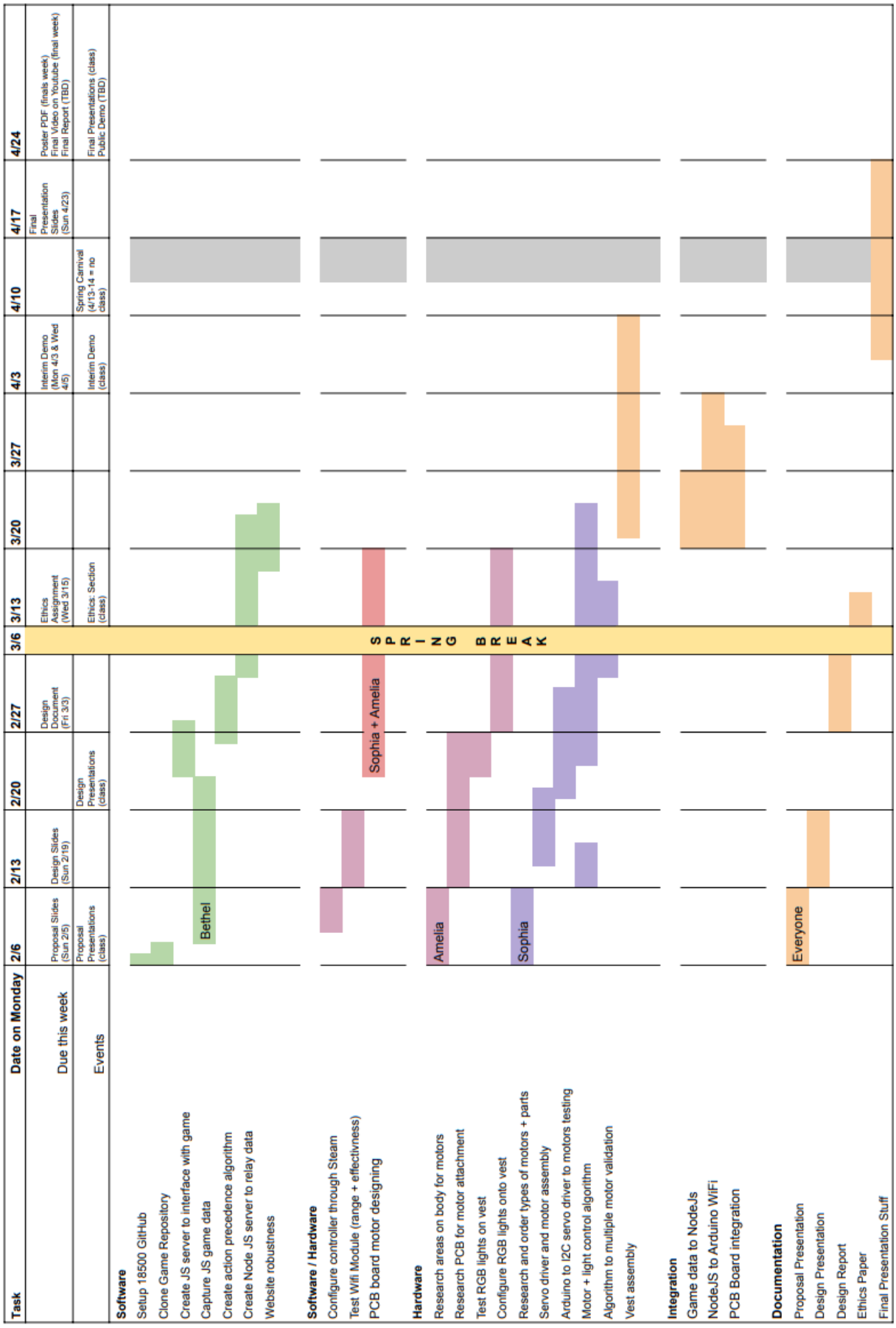

Figure 8: Gantt Chart