Interactive Self-Standing Training Bag

University of Central Florida

College of Engineering and Computer Science

Department of Electrical and Computer Engineering

Dr. Samuel Richie

EEL4914: Senior Design I

100 Page Draft Checkpoint Report

Group 22

Hannah Clarke: Electrical Engineering

Joseph De La Pascua: Electrical Engineering

Nicole Karam Pannaci: Electrical Engineering

Natesha Ramdhani: Electrical Engineering

Table of Contents

Table of Figures

Table of Tables

Introduction: Project Statement/Narrative

For our Senior Design project, we are creating an interactive boxing/martial arts stand up workout bag, where the athlete can choose different types of workout patterns, and receive results to view progress. Our motivation behind this project stems from the idea of creating an easy yet engaging way for a trainee of any level, to exercise, train, and track their progress in various perspectives of fight training without the need for a training partner. The Coronavirus pandemic has made this even more relevant since most training locations closed for an extended time, leaving many people without a method or physical partner to train at home with.

Another aspect of our project is that our design will be at a lower cost to those on the market, which can do some, but not all, of the actions as our interactive design. Currently on the market, interactive standup punching bags usually sell for about $25,000. Because our design can provide feedback/results after a session as well as provide interesting/fun workout patterns, the user can practice their skills in an engaging manner without paying the $40 to $70 average hourly cost of a personal trainer. Additionally, our design can be used at home which saves the user the monthly cost of a gym membership necessary for the correct equipment: a cost that ranges from $10 to $100 per month.

In order to capture as many different aspects of training as possible, the bag will have a dual-sided design that accommodates various modes that the user can train with. “Side A” will have various zones with binary touch sensors that will detect whether a certain area was hit. These zones will be loosely based on different practical locations that a fighter would be making contact with in a boxing, kickboxing, or martial arts environment against a real opponent, so that the user can have the most immersive and accurate training possible. On “Side A”, the device should be able to generate random sequences of techniques landing in various areas; this can be extended to using multi-color indicators to signal which side or technique type should be used to attack the opponent. “Side B” will contain a target zone with a sensor/cluster of multiple sensors arranged within this target so that more focused training can also take place and create a complete, rounded training session. These sensors will gather data on the training session and report it back to the user so that the user can quantify their performance in different types of workouts and track their progress. Some modes that can be implemented by “Side B” include a cardio mode where users try to make as many hits as possible within a time limit, a reaction mode where users aim to land their hits based on a stimulus by the indicator, and an accuracy mode where the user practices hitting a specific area as precisely as possible.

A regular standup punching bag alone does not have the capability to improve a user’s performance, since it does not provide feedback or results without a boxing/martial arts trainer. With an interactive design, the user would be able to improve performance in precision, reaction time and speed. This design should be easy to use for first time athletes as well as experienced athletes because the user is able to select different mode levels that are most suitable for them, and increment their levels as needed to continue to receive better training.

Project Goals and Objectives

General Specifications

This device should:

- Be able to implement downtime modes
    - Idle Mode
    - Display Mode
    - Off Mode
- Be able to implement exercise modes
    - Combination Generator (Side A)
        - Uses set zones on opponent
        - Binary sensors count correct hits and save data for user
        - Practice training against an "opponent" to improve the user's skill
    - Cardio Mode (Side B)
        - Uses sensor target to count hits
        - User goal: make as many hits as possible in a certain time period
        - more hits = higher score
    - Reaction Mode (Side B)
        - Uses sensor to detect time for user to hit target after illuminating indicator
        - User goal: hit target as soon as possible after it lights up
    - Accuracy Mode (Side B)
        - Uses sensor to detect how close to center mark target is hit
        - User goal: hit center as accurately as possible to increase score

Marketing Requirements

- **Durability:** Device will be able to withstand impact from regular use. (+)
- **Cost:** Device must be cost efficient for a typical consumer; priced reasonably compared to other multimode home-gym fitness devices. (-)
- **Portability:** Device must be able to be moved if necessary without large risk of damage, and must be able to be used in a variety of different environments. (+)
- **Intuitive:** The user interface will be easy to understand for and welcoming to experts and enthusiasts alike. (+)
- **Versatile:** The device will have multiple modes of operation utilizing both sides and orientation of sensors for a well-rounded user experience. (+)
- **Device life:** The device should last as long as possible; at least 5 years for the user. (+)

Technical Requirements

- **Durability:** Components will be able to stand up to impact at a rate of over 120 hits/minute. (+)
- **Power:** Components will be power efficient; no more than 1.1kW. (-)
- **Compatibility:** Components will integrate well with other components used in the device. (+)

- **Component Costs:** Total cost of type of components of good quality will be within our budget, but as low as possible. (-)
- **Implementation Time:** The time taken for the design and implementation of modes using the sensors should not be excessive. Our goal is 12 weeks or less to build and develop. (-)
- **Longevity:** Components will not die out quickly, providing approximately 5 of years of use. (+)

House of Quality

Double Symbol = **Strong** Correlation | Single Symbol = **Weak** Correlation

Technical Requirements vs Technical Requirements

	Durability (+)	Power (-)	Compatibility (+)	Component Cost (-)	Implementation Time (-)	Longevity (+)
Implementation Time (-)						
Component Cost (-)						↓
Compatibility (+)				↑	↓	
Power (-)						↑↑
Durability (+)				↓		↑↑

Technical Requirements (Green) vs Marketing Requirements (Blue)

	Durability (+)	Power (-)	Compatibility (+)	Component Cost (-)	Implementation Time (-)	Longevity (+)
Durability (+)	↑↑			↓		↑
Cost (-)	↓		↓	↑↑		↓
Portability (+)	↑			↓		↑
Easy to Use (+)			↑		↓	
Versatile (+)		↓	↓	↓	↓	↑
Device Life (+)	↑↑			↑		↑↑

Table 1: House of Quality Specification Matrix

Realistic Design Constraints

Design constraints are involved in every engineering project. The following content provides certain constraints that will apply to the manufacturing and operation of the punching bag. These constraints include: Economic, Time, Environmental, Health, Safety, Social, Ethical, Political, and Manufacturability and Sustainability constraints. These constraints provide limits to engineering designs based on standard scarcity that exists in the material world. Some constraints, such as the health and safety, are less focused on scarcity and more focused on protecting both the consumer and the commons that the consumer is free to operate in. Each constraint will be considered both individually and with regard to the full scope of the project.

Economic and Time Constraints

The budget for our project is $500. To cover the costs of our project, our group has decided to self-fund the budget. The primary costs of the project are the punching bag itself and the electronics that will be used. The punching bag itself is the largest direct cost for this project, to lower this cost a used punching bag will be acquired. A used bag is preferred as it will be heavily modified to accommodate the electronics used. Although our selection of MCU has lowered the costs associated with the bag, multiple sensors will be acquired and used throughout the bag, adding more cost. Although sensors are generally inexpensive, multiple sensors are needed to cover all sections of the bag required to bring full functionality to the bag. Some of the components will be bought twice to cover any component failure that could happen along development of the project. Economic constraints are necessary to consider as most items need to be purchased to finish and manufacture the product. This will be finalized until the end of the project.

Another constraint to consider for the running cost of this project is the cost of electricity. This constraint should be considered not in the manufacturing, but in the final product operating cost. This product will require standard US AC power input, and therefore will require standard energy costs for operation. Power supply design and operations will factor in to how much power is needed to run this machine and therefore how expensive the operating costs will be. This constraint will differ depending on which region and power supplier are used to power the machine. Standard Duke Energy operating costs in Florida is an average of $0.12/kWh. This does not account for monthly fixed costs for power.

One large constraint for this project is the overall time limit of the project. The project will be done in April 2021. The research and testing of the project will be complete during the first semester of Senior Design, by the beginning of December 2020. After research and testing is complete, the manufacturing of the punching bag will begin in the second semester of Senior Design in Spring 2021. Product testing will be done during and after the manufacturing. Testing will be required to incorporate the design of both the code and the electronics in the punching bag. After extensive testing, the completed product will be presented to the judges panel at the end of the Spring 2021 semester. Time constraints are vital to consider as they dictate the schedule this project must follow.

Environmental, Health, and Safety Constraints

To protect the commons for the consumers and producers of different items, certain environmental, health, and safety constraints need to be considered. The main environmental constraint restricting the product is the cost and availability of electricity. Electricity in the surrounding area is supplied by Duke Energy, who owns a coal plant that provides energy and electricity to much of the Orlando/UCF area. The direct cost of the electricity, in terms of dollars, was given in the previous economic constraint section, but environmental considerations must be taken in the context of 21st century environmental degradation. Due to power being supplied by a coal power plant, running machinery in the Orlando area contributes to further carbon emissions in the atmosphere, exacerbating climate change.

To limit the effects of runaway climate change, measures should be taken in the design steps to use as little power as possible for the operation of the machine. This constraint is part of a collective effort to limit personal contributions to greenhouse emissions and should be evaluated in all future engineering projects and designs. This personal responsibility cannot replace certain standards set for power efficiency and regulations on limits of power and emissions usage.

Due to consumer use, health and safety constraints are required for the design of the machine. The most pressing issue is the electrical wiring of the machine. The sensors on the punching bag will need to be wired to the MCU to perform the calculations necessary to the project. These wires should be out of the way of the punching surfaces for the safety of the user and the components.

Another safety consideration is the overall weight of the machine. With the added components, the weight of the machine should not go over a threshold that would make it unsuitable to be carried without the water or sand stabilizer. This is important both for the usability and for the safety of the consumer. Due to the relatively low weight of the MCU, wires, and sensors when compared to the bag itself, this constraint should not be too limiting.

One final constraint is the heat of the circuitry and the MCU. The MCU should be housed in heat resistant plastic in order to shield it from the rest of the machine, which could be damaged when exposed to any heat from the MCU.

Social, Ethical, and Political Constraints

Surrounding the manufacturing of this project there are several social, ethical, and political constraints for our design. The social and ethical responsibility to lower power use and overall emissions has been previously discussed. Another ethical constraint applying to this project is the responsibility to report the correct calculations for each workout mode. This product will be used by athletes who must complete standard workout regimens many times a week. To aid their schedules, our machine needs to work every time it is powered on to be used and to report accurate information at the end of each section for the athlete to assess. For athletes to rely on our product, their trust should be gained by effective and habitual use. Sticking to this constraint will also fulfill political constraints including ethical manufacturing of the product; if the product does not report information correctly, customers have grounds to stop using the product and even grounds to sue depending on the accuracy of our product's ending information and assessment. Avoiding false advertising of our product in the future would work to ease this concern, especially around the assessment of the workout our product performs.

Manufacturability and Sustainability Constraints

Sustainability is a constraint with a lot of focus for this project. Not only should sustainability of power be taken into account for but also sustainability of building materials. Electronics especially do not fare well once thrown out, in fact many electronics need to be specifically recycled to be disposed of safely. Our component disposal will follow all electronic recycling guidelines outlined by the Environmental Protection Agency (EPA). Only the regulations from the EPA have to be followed as the State of Florida has no additional E-waste laws or regulations. An E-waste recycler with a third-party certification is recommended for use by the EPA and the Florida Department of Environmental Protection. Another constraint for manufacturing sustainability is the punching bag itself. To meet sustainability (and cost) constraints, a used bag will be acquired to use for the prototype. This assures that new material and products are not wasted in the construction of the machine. Buying used components is one major consideration we made to meet both the economic and sustainability constraints. Two constraints were effectively applied during our buying habits.

Standards

There are not many standards that apply to the interactive workout bag product. The few that do apply include power supply standards, standards for fitness equipment, communication standards and PCB standards. Each type of applicable standard is examined, as well as its correlation to the design.

Power Supply Standards

IEC 60906-2:2011

Also known as the IEC system of plugs and socket-outlets for household and similar purposes, Part 2: Plugs and socket-outlets 15 A 125 V a.c. and 20 A 125 V a.c., is a standard that applies any to the NEMA 5-15-P , also known as the "3-prong grounded plug" connection. This standard explains that this connection will provide protective earthing to equipment that is connected to the conductive parts of the socket. It also is electronically separated from the protective earthing circuit to provide electrical noise inmunity.

This standard applies to this project since we will need this type of plug to be compatible with the standard american wall outlet. Our design will also need the three prong connector (positive, neutral and ground) for our circuit design. It is also critical that our system has no electrical noise from the power supply, which the IEC 60906-2:2011 standard defines.

Regulation Workout Bag Standards (weight, size, etc)

The American Society for Testing and Materials (ASTM) is an international organization that standardizes the safety and testing of materials. The standard ASTM F2276-10(2015), also known as the Standard Specification for Fitness Equipment, is a standard that all designers and manufacturers for a fitness equipment must follow. The standard lays out specific criteria which is applicable to the workout bag design. In the standard, it is defined that only individuals age 12 or older can use this equipment, that the product must include all the documentation on how to assemble/build the product, include a list of parts and instructions, and provide adequate warnings. This standard also defines which exercise equipment is required to be used in an indoor setting or environment.

This standard applies to our project design because the workout bag falls under the classification of fitness equipment. Therefore, our project design will need to include a parts list, instructions and applicable warnings that include an age restriction.

PCB Standards

IPC-221B

The workout bag system implements two printed circuit boards, one in the base system and one in the UI system (remote). Association Connecting Electronics Industries, previously known as the Institute of Printed Circuits (IPC) sets the standards for all types of printed circuit boards (PCBs). In conjunction with the IPC standards.

The standards that apply to our PCB designs from the IPC are in IPC-2220 series. The IPC-221B standard is the standard that defines the fundamental design requirements (also called the generic design requirements) for designing the printed circuit boards, component mounting, and interconnecting structures. Because our PCBs will include a design and mounted components, it is important to follow the guidelines set by these standards.

Communication Protocol Standards

IEEE standard 802.15.4

The UI system and the base system communicate low rate wireless personal area network (LR-WPAN) which is defined by the IEEE 802.15.4 standard. The IEEE 802.15.4 standard is designed to create a WPAN that is low power. The IEEE 802.15.4 standard defines the physical layer (the bottom layer of the protocol layers) and affects the MAC layer, a sub layer of the network layer.

Zigbee, which is used to control the communication between the UI system and the base, builds on the IEEE 802.15.4 standard. The physical layer controls the interface of transmission, which include the transceiver and the transmission frequencies. The MAC (medium access control) layer controls the MAC frames that are then sent using the physical layer. The IEEE 802.15.4 standard also has three types of nodes on the network, a coordinator node, router node and device node. These nodes are able to interact as peer to peer networks or in spoke-hub networks, pictured below.

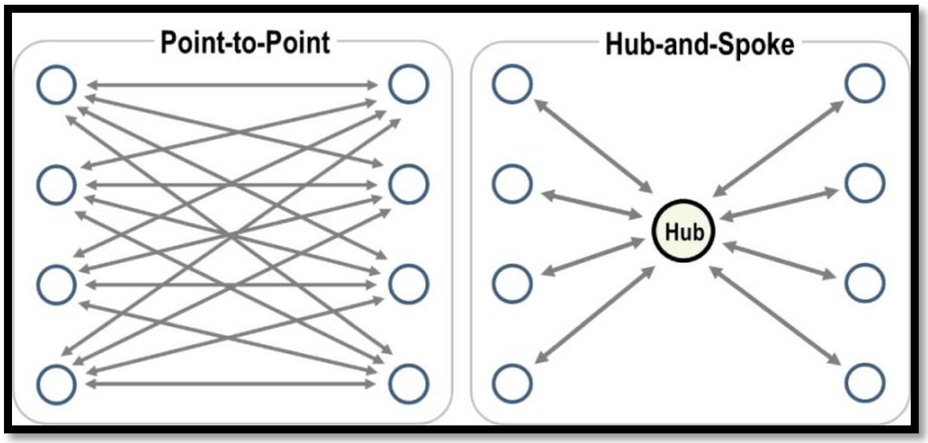
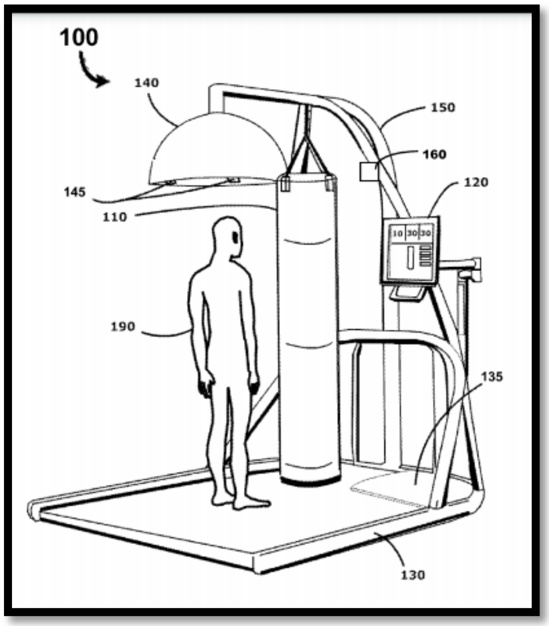


Figure 1: Point to Point (Peer to Peer) vs Hub and Spoke networks

The IEEE 802.15.4 standard also defines the RF parameters for the communication. In North America, the IEEE 802.15.4 standard uses 902 to 928 MHz frequency bands. There are a maximum of 10 supported channels with a bandwidth of 2MHz and a 0.6MHz guard band. The data rate is also defined at 40kbps. Although the data rate is low, this communication standard is perfect for our design, since only a few bytes need to be sent across the network, and the communication system needs to be low power.

Market Research

The idea of a dynamic interactive punching bag has existed for a while. The first time it was patented, a device like the one we are designing was back in 2010 [1] in which a patent was submitted for a martial arts bag with one or more sensors for measuring a workout performance. In which, they also show many pictures of many portrayals of this type of design in which our design is like a combination of the Figure 1 and figure 3A. In figure 1, it will have a coach that will show on the screen to help users to hit the bag. In the 3A case, it would impact areas around for the user to hit but it still does not mention how the results are recorded and shown to the user.



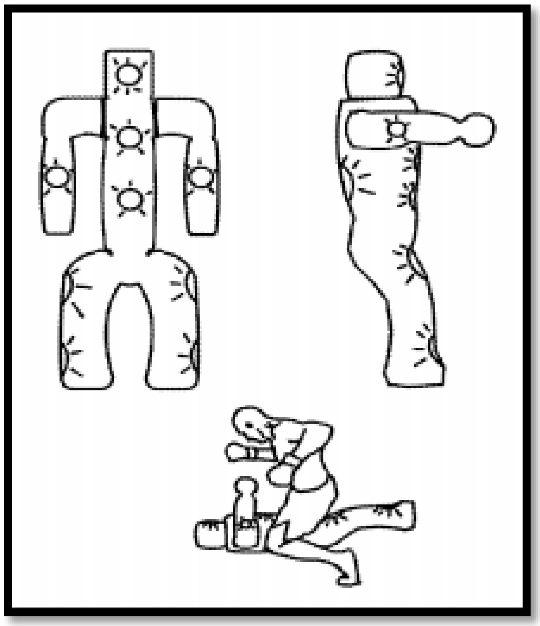
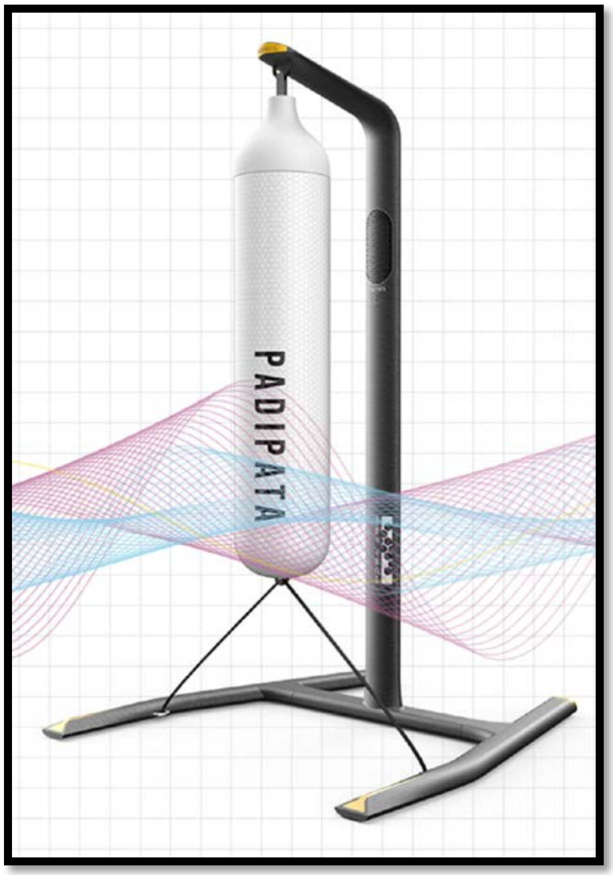


Figure 2 (Left): The main idea for the US Patent US20110172060A1
Figure 3 (Right): A variation of the US Patent US20110172060A1

We also found two research articles from the Czech Republic in which they used [7]and [8] In which, both measure a direct punch applied to a martial arts bag using strain gauges inside of the martial arts bag. In regards to the type of strain gauge used, both [7] and [8] use a strain gauge type SRK-3/V , and in [8] also uses a strain gauge L6E-C3-300Kg in their measurements. It was found interesting since in [8] came to the conclusion that L6E-C3-300Kg strain gauge has a better precision, speed, and simpler to use than the strain gauge type SRK-3/V, but in [8] that was made with the same people used the latter instead to further intensify the research about measurement direct punch force.

PADIPATA™ created an interactive punching bag, with all its sensors in the middle of the punching bag and it only leaves the user to use punching instead of using the user's legs as well. It is quite like the Figure 2 in the [1] reference. This device comes in 3 sizes, but is only available in the form of a hanging bag as opposed to a self-standing bag with a base on the ground, the design which our group has chosen. It also has surround sound with an interactive screen inside of the band. This product was created in Changsha City, People's Republic of China and retails for a price of $25,000 (USD); our intent is to create a device that is much lower in cost than that, so that it is accessible to a wider audience than the Padipata bag. The device our team will create should also employ different interactive modes to make it more efficient for the user.



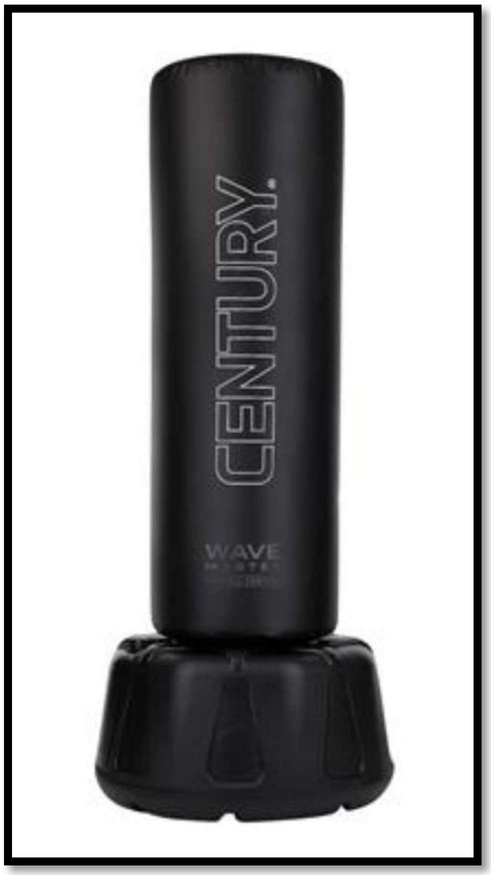


Figure 4 (Left): Padipata Bag Structural Design

Figure 5 (Right): Century Martial Arts Wavemaster Structural Design [32]

Technical Component Research and Standards Comparison

Sensors (Nicole)

Technical Standards

Sensors used in this device should:

1.1	Be durable enough to withstand the force of a punch/kick.
1.2a	Be able to detect distinct, rapid, successive hits in different locations (Side A).
1.2b	Be able to detect distinct, rapid, successive hits in the same location (Side B).
1.3	Be able to detect the location of a hit relative to a target point (Side B).
1.4	Consume a low amount of power.
1.5	Cover a zone with no more than four individual sensor units.

Table 2: Technical Standards for Sensors

Candidates for Use in Device

We will need sensors for our project, and these are the research done to find values of them. The original concept for the sensors in this device were inspired by the Dance Dance Revolution(DDR) home mats since these mats can support the weight of people stepping to it with extra force due to the excitement of the people when they play this game. These DDR mats use buttons to signal that an arrow was pressed, so no sensors are used. We could incorporate the following alongside/ instead of the buttons to enhance the how much force someone should apply to the self-standing kicking bag and/or the precision of their pressing to the area they have to punch. These components will be under a tarp to protect the electrical components.

Piezoresistance Sensors

The piezoresistance is the change of electrical resistance when a force is applied to it. These use a Wheatstone bridge to measure the change of resistance. It is very sensitive to changes, low cost, and resistant to many diverse changes. The drawback to this type of sensor is that it requires a lot of power for it to work.

Piezoelectric Sensors

Piezoelectrics use materials, such as quartz crystals or specially formulated ceramics, which generate a charge across the faces when pressure is applied. A charge amplifier converts this to an output voltage proportional to the pressure. A given force results in a corresponding charge across the sensing element. However, this charge will leak away over time meaning that the sensor cannot be used to measure static pressure. Low power and robustness. The drawback is that they are very complex, and they can only be used for dynamic pressure measurement.

Strain Gauge

A strain gauge is a mechanical way to measure the way the material changes when force is applied to an area. The possibility is using four long linear strain gauges to cover vertical, horizontal, and each diagonal sides, sharing the middle as the accuracy measurement of the hit, of an area to help the user to improve their precision and accuracy in each hit, as well when the user is hitting the area, this could tell the user to use more force to keep the same amount of force in each hit.

Strain Gauges are a possibility since these types of sensors are robust and their whole purpose is to feel the change of movement of an area. This sensor will be attached to the tarp and due to the force and change of strain this component will be able to sense it and send this change to the software to count it as a hit. If used, it would be ideal to have a 45 degree stack plus a horizontal in each area or a Membrane Rosette Strain Gauge since this one covers a more circular area than having four linear straight gauges.

Inertial Measurement Unit (IMU) Sensors

IMUs will allow us to know the direction and force of every hit on the target on side B of the design. An Inertial Measurement Unit (IMU) is a type of device that includes gyroscopes and accelerometers to measure angular rate, force, and force. An IMU works by collecting the raw data of what surrounds the device, and tells its user about the X,Y,Z surroundings of the device. Some uses of IMUs include robots, drones, video game remotes, aircrafts, antennas, GPS, sports applications, and virtual and augmented reality. IMUs are beneficial for our project since it will help the user tell the force and direction in each hit. The IMU will sense to move in consequence of the hit from the user and it will be recorded each time and then it will be processed by the MCU to make each hit clear and different from each other. The type of sensor that would be beneficial to our project would need to have the capabilities to sense sports movements and/or motion tracking and gesture location.

Accelerometers

In case that the IMUs on side B do not work as desired, accelerometers shall be used instead. Accelerometers measures the acceleration forces applied. In this case, it shall tell us the force that is applied to the hit area. Uses of accelerometers include cars, machines, buildings, process control systems, safety installations, and airbags. They will be satisfactory backup options for the devices if IMUs do not work; however, they only report force magnitude of the impact, not the direction. This information will be able to be manipulated to calculate desired results, regardless.

Binary Sensors

For Side A, we shall need five binary sensors. Binary sensor is a sensor that returns a 0 when it is not sensed and returns a 1 when it is sensed. Every switch and button can be considered a binary sensor since it gives a 1 when is pressed and 0 when it is not pressed. Some applications of binary sensors include buildings, cars, electronics, and robots. Binary sensors are the best option for this part of the device since it is basic but sufficient for the desired use and user input.

Design Matrix and Analysis

Standard:	1.1	1.2a	1.2b	1.3	1.4	1.5	Total
Accelerometers	2	5	4	1	5	5	22
IMU	1	4	5	5	4	4	23
MIMU	3	3	3	4	1	1	15
Strain Gauge	5	1	2	3	2	2	15
Rosette Strain Gauge	4	2	1	2	3	3	15

Table 3: Design Matrix for Sensors on Side B

We will be discussing accelerometers and IMU since these had the highest ranking among all the sensors. In **1.1**, these have low ranking since without any protective layer, these sensors will not survive any damages done by any hits/kicks by the user. In **1.2b**, IMU is ranked higher since it can differentiate each force of each hit/kick with the direction of the impact, and accelerometers will only measure the force of the hit/kick of the impact. In **1.2a** if it is just hitting in one area, accelerometers will do the job fine since it only measures the force applied to the hit/kick, while the IMU will collect the direction applied to the location as well. In **1.3**, IMU will measure the direction of the hit, so it will say how close it was to the target point. The accelerometer is the least desirable here since it only measures the force than according to the hit from the target area. In **1.4,** Accelerometers only measures the force and we can find one that can consume a lower amount of power for it to work. The IMU will not consume as much power as the accelerometer but since it measures more than one sensor, it could consume more power than the accelerometer. In **1.5**, The Accelerometer contains just one sensor and that one sole sensor measures the force of the hit and IMU possess different types of sensors, so it might just be one device that measures everything, but, in this ranking, it was considered as well all the sensors that are inside of the device.

Indicators (Natesha)

Technical Standards

The indicators used on this device should:

2.1	produce light with high enough intensity to be visible behind a translucent cover
2.2a	(each individual unit should) be durable enough to withstand repeated hits
2.2b	(connections between units should) be durable enough to withstand repeated hits
2.3	be able to easily shape as needed without excessive strain
2.4	sufficiently indicate zone in multiple colors with as few physical units as possible
2.5	have a lifetime comparable to that of the device itself.

Table 4: Technical Standards for Indicators

Candidates for Use in Device

There are three main options for the indicators used: addressable LED strip lights, individual LEDs, and electroluminescent (EL wire). More detail about each will be provided in the next sections, before being compared directly to each other in a decision matrix to pick the best candidate for indicators in this device.

Addressable LED Strip

One indicator option is to use strips of addressable LED lights. These have become commonplace in recent years due to their popularity for indoor and outdoor decorative applications. Each segment of the LED strip requires an input voltage for power and three connectors containing the needed value of each primary color (red, green, blue) to create the desired color. The segment itself contains RGB LED light units and a controller to determine the values for each primary LED color. The strip can be cut in between segments and attached together by connecting the input of one segment to the output of another; each segment can work individually as long as it is a part of a complete closed circuit.

Since each LED is individually addressable, this approach would allow for a large range of visual effects. Additionally, the popularity of these lights means we will have a large set of features to choose from, such as mounting color, input voltage, and differing color schemes. Some of these features come at the cost of others; a strip with a waterproof sheath will be more durable, a feature we will likely need. However, these sheathes may make our such strips significantly more difficult to shape and cut to our design. Other features will challenge us to balance their worth with their cost; strips with higher pixel density can display higher resolution effects, but at increased costs. Pricing is heavily dependent on the features offered as well as the length of the strip, so we would need a preliminary design and lighting patterns ready in order to be confident in our choice.

LEDs

Individual LEDs are another option for indicators: using individual LED bulbs with some kind of material within which the LEDs are scattered will allow the light to diffuse through it so that the entire panel is illuminated. This would likely be the cheapest option since each individual component has the lowest cost and the group has the largest control of exactly how they would be used. Although LEDs themselves are fairly durable due to their plastic, rounded shell that is unlikely to break when faced with blunt force, their connections to each other may end up compromised if hit incorrectly. They can be controlled directly by a microcontroller and a DC power source, so they will be fairly straightforward to program for different effects and they require little power to function (meaning that they will not heat up easily). LEDs can burn out if left on for too long, though, making the life of the indicator unpredictable.

Electroluminescent Wire

Electroluminescent wire (EL Wire) is made of a stiff, phosphor coated core, wrapped by a corona wire acting as a capacitive plate around this core and an outer PVC covering protecting the wire. EL wire, as well as derivatives like EL tape and panels, are frequently used in clothing applications due to the fact that they produce no heat, consume less power than LEDs, and can retain their shape while still being flexible enough to curl around a finger. This makes them a good fit for our project, able to withstand accidental hits by the user better than an LED based approach might be.

Since EL wire is capacitive, it cannot be controlled with a typical PWM signal. Instead, they are powered by an AC inverter, with the frequency and voltage of the AC signal dictating the wire's brightness. The wire's brightness is therefore dependent on the load it places on the AC source; using more or longer wires will dim them. Additionally, the wire dims gradually over time as the core's phosphor degrades, with a typical lifespan of 1-2 years if outdoors or exposed to direct UV light, or about 3 years if in normal indoor conditions and maybe even longer in lower light conditions. Their lifespans are also dictated by the voltage being applied to the wire, so by regulating the voltage applied, we may be able to make the wire last longer. Ideally, making the EL wire segments modular would be the best option for both profit and allowing the user to keep the device functioning as long as they would like to use it (with the bonus benefit of allowing the user to buy whichever different colors they would like to use).

To use EL wire in our design, we will need to account for either a DC to AC inverter with sufficient voltage/frequency. Alternatively, we can also connect the wire directly to an AC source and eliminate a potentially noisy inverter, designing separate circuitry to control these AC components.

Design Matrix and Analysis

Standard:	2.1	2.2a	2.2b	2.3	2.4	2.5	Total
Addressable LED Strip	3	3	2	1	3	3	**15**
Electroluminescent Wire	2	1	3	3	2	2	**13**
Individual LEDs	1	2	1	2	1	1	**8**

Table 5: Design Matrix for Indicators

The best scoring option for our project's indicators are the addressable LED strips. These are strips of many LEDs, which a controller is able to individually address to control brightness and color. They offer more than enough brightness **(2.1)** and lighting control **(2.4)** for our project, and are relatively more durable **(2.2)** and long-lasting **(2.5)** than our other options. One potential shortcoming of these strips may be their stiffness; the thickness of some LED strips can make them difficult to bend into tight shapes **(2.3)**, but the benefits of using this indicator far outstrip this inconvenience.

We can easily rule out individual LEDs, our lowest scoring option. While this option can offer the same potential as LED strips, we would need to hand-solder and wire each individual indicator, making this choice significantly more fragile and difficult to shape **(2.2, 2.3)** than other indicators. Their brightness **(2.1)** and ability to indicate in different colors **(2.4)** is limited by this as well, since we would like to avoid using too many physical units, and their individual brightness cannot be pushed too high without risk of burnout **(2.5)**.

Electroluminescent wires came close in score to addressable LED strips. These devices are made of a phosphor-covered core wire and two thin outer wires, and produce light (the color is chosen by its plastic sheath) when AC voltage is applied. Ultimately, they are limited in their brightness **(2.1)**, especially over long periods of time, as they lose 50% of their brightness in 3000 hours **(2.5)**. They are quite durable **(2.2)** and easy to shape to our needs **(2.3)**, but we would need to pack multiple runs of EL wire in an area to use multiple colors **(2.4)**, while LED strips are able to show multiple colors and effects with one unit.

Processor & Memory (Joseph)

Technical Standards

The processor used by the device should:

3.1	Have sufficient input lines for all sensors and communication necessary.
3.2	Have sufficient output lines for all indicators and communication necessary.
3.3	Communicate with the UI system to provide raw data.
3.4	Require internal ROM and RAM.
3.5	Work with a volatile memory large enough to store data from multiple sensors.

Table 6: Technical Standards for Processor and Memory

Considerations for Components

Four main contenders for our main processor were considered. The processor is required to take in inputs from the sensors and run calculations according to the data to assess the workout after it is completed. Our processor will also be the component that starts and controls all aspects of the workout modes. The main considerations taken to choose what type of processor was most appropriate for this design are: sufficient input lines, sufficient output lines, able to communicate with the sensors, and sufficient ROM and RAM. These four considerations were evaluated individually for each processor design and points were assigned to each selection considering how well it applied to that consideration. The product with the most points at the end of this analysis was chosen. Ultimately the processor chosen was the standard MCU.

FPGA

A field-programmable gate array (FPGA) is an integrated circuit that can be configured to have certain integrated circuitry configured by the designer after manufacturing. The main difference between an FPGA and a standard processor is the hardware of the FPGA can be configured as well as the software. To encode a FPGA, a hardware description language (HDL), like Verilog, must be used. The HDL is used to change the internal circuitry of the FPGA by assigning different components to the circuitry of the board. Once configured, SW can be loaded onto the FPGA and it can be used as a flexible MCU to perform calculations.

When evaluating the design considerations with the FPGA, the FPGA can be standardized by its compatibility with our final project. The first consideration to be evaluated is the number of input lines for the processor. The FPGA has User I/O pins that can be programmed to be input or output pins. Given the flexible nature of the FPGA, the pins themselves are designed to operate multi directionally. Although the pins have this flexibility, there are not as many total I/O pins on the standard FPGAs, certainly not as much as other options considered. The next design consideration is the ability to connect to and communicate with the sensors. FPGAs are compatible with connections such as UART, I2C, and SPI, making it compatible with the sensors used in the machine. The last design consideration evaluated is the availability of sufficient RAM and ROM to store the data from the sensors. FPGA boards have RAM and ROM of sufficient size, at around 256MB of RAM. Although many of the considerations are met with the FPGA, the overall cost and lack of I/O make this a lower contender to be included in the final design.

DSP

A digital signal processor (DSP) is a specialized microcontroller chip that is optimized for the needs of digital signal processing (DSPg). These processors are especially and widely used in fields such as audio signal processing, telecommunications, and in consumer electronics such as mobile phones. The main use for these processors is to measure, filter, and compress analog and digital signals. It does this by first converting the analog signal to digital, then applying processing filters, then converting the output digital signal to analog signal. Although DSPg algorithms can be run on any microprocessor, the DSP offers advantages in areas such as data compression that make DSP favored to use for this operation when it is the main operation for a system.

The first consideration to be evaluated is the number of input and output lines. There has to be sufficient I/O to collect the data from all of the sensors simultaneously. DSP processors have sufficient GPIO pins and 3 full UART interfaces, four SPI connections, and three I2C interfaces. The DSP has sufficient I/O to incorporate all of the sensors into the microcontroller. Given that the sensors communicate using UART, I2C, and SPI, there is enough I/O to healthily support the data collection.

The last consideration is the amount of RAM and ROM that the processor is able to support. The DSP has up to 4GB of expansion, with an onboard amount of about 1024KB. This is enough expansion to store the data from the sensors. The DSP meets all the requirements for the project, however because it is specialized for signal processing, the DSP does all of these functions while being more expensive than other, non-specialized processors.

Raspberry Pi

The Raspberry Pi is the most all-in-one processor that was selected for consideration. The Raspberry Pi is essentially a small computer included with a chip, I/O, and a specialized power input. The Pi acts as a desktop computer on a chip, even being able to load an operating system on the chip. The Pi is usually used for consumer grade projects where an operating system can be beneficial to add more complexity and control to an electrical project. The raspberry pi is the second most expensive processor at $50.

The PI has 40 standard GPIO pins used for I/O. There are also standard USB 2.0 and 3.0 ports for I/O. Even though there is an adequate amount of I/O. The key functionality of SPI, UART, and I2C is missing from this processor. This is enough to disqualify the raspberry pi from consideration, as the sensors communicate using the UART, SPI, or I2C protocols. The Raspberry Pi is not the component to be used.

MCU

A microcontroller is an integrated circuit that is designed to perform specific operations for a closed system. The standard components included with a microcontroller are the processor, memory, and general I/O. These are typically all included together on a single chip. Microcontrollers are primarily designed for embedded tasks that are automatically controlled. These tasks include anything from automobile engine control systems to implantable medical devices. Any time an automatic aspect of control is needed a microcontroller is generally an acceptable processor to use.

To evaluate the MCU, the first consideration of I/O lines needs to be addressed. There is a large amount of variability of I/O pins for different microcontrollers. The amount of GPIO pins that a standard MCU has is around 50 pins. This is a sufficient amount of I/O pins for the assigned tasks. The next consideration is the ability to communicate with the sensors. With 2 I2C ports, 4 UART ports, and 2 SPI interfaces, there is plenty of room to include all of the sensors for the machine. The final consideration is the amount and availability of RAM and ROM. With a ROM of 128KB and RAM of 8KB there should be sufficient memory to store the data from the sensors. Expansion of RAM and ROM is also available with the microcontroller. Due to the relatively low cost and the satisfaction of all of the considerations, the MCU will be chosen as the processor for the project.

Design Matrix and Analysis

Standard:	3.1	3.2	3.3	3.4	3.5	Total
FPGA	1	4	1	3	9	**9**
DSP	4	1	4	1	10	**10**
MCU (MSP)	3	2	4	4	13	**13**
Raspberry Pi	2	3	2	4	11	**11**

Table 7: Design Matrix for Processor and Memory

3.1 is the power analysis. For the point selection, the highest level was given to the processor with the lowest input power necessary to run. This is the DSP MCU. The MSP MCU was the second lowest input voltage requirement. This makes the MCU and DSP more attractive for total consumption of power and therefore cheaper operating costs. The Raspberry Pi runs at 5V: a contender for second place with the MSP. The FPGA board requires the most input voltage at 7V-15V, making it the most expensive operating cost.

3.2 is the compatibility analysis. The main components to compatibility for this project is speed of the processors and the I/O options to connect to the sensors and screens. FPGA has several features that make it very compatible with this project. It has the second fastest speed, it has ample RAM (256MB) and flash memory (16MB), and has SPI and UART interface for connection. Another unique feature of the FPGA is the ability to change both the software circuitry and the hardware circuitry on the same board. This will allow us to code a different hardware implementation for each mode of operation, allowing for much greater flexibility of operation. The Raspberry pi is another contender for this mode due to its fastest speed and the amount of I/O that is available both on the baseboard and with extensions to the controller. The I/O is USB and standard GIPO headers so it has greater compatibility with both USB and GPIO sensors. It has 4GB of SDRAM. The MSP MCU has many options in terms of I/O, particularly UART, I2C, and SPI are the three main connections. But the total processing speed is only up to 25MHz, making it the slowest processor, but it edges out the DSP due to many different options of I/O. It also comes with 128KB of flash memory and 8KB of RAM. The DSP controller has middle road speed, but only works with I2C connection, making the connection type limited. The DSP has 2MB of flash memory.

3.3 is the cost analysis. At $120-$200 the FPGA is the most expensive board, followed closely by the raspberry pi at $100. The MSP and DSP MCUs are each relatively the same price at around $15.

3.4 is the implementation time analysis. The implementation time for the MCU and Rasp. Pi are very similar due to both being already compatible microcontrollers with set hardware circuitry. This allows us for a more "plug and play" method for these processors. The FPGA board has programmable hardware circuits, which increases the implementation time because we have to code both hardware and software through the board. The FPGA does have I/O already part of the controller however which decreases the implementation time. The DSP MCU does not come with the same I/O board as the other three options, soldering would be necessary to implement the DSP MCU, making it have the worst implementation time. After considering all these options in the table above, the MSP MCU was determined to have the highest total score after the analysis.

UI System (Hannah)

Technical Standards

The device's user interface should:

4.1	Provide completed results to the user including number of total targets, number of total successful hits, total time per mode run, and average time per hit.
4.2	Receive user inputs such as turn on/off, select mode, and select level of difficulty.
4.3	Be capable of communication between itself and the base.

Table 8: Technical Standards for UI System

Considerations and Candidates for Components

Displaying Results and Messages to User

One of the main concepts that makes the workout bag an interactive design is the ability to show results of the workout session to the user in a simple and effective way. To do this, several types of ways to display results were examined and compared.

A simple option to provide the user with results would be to incorporate a USB port that the user can download the results from. This option would require the user to acquire a flash drive and a machine to view the results. Because we want our design to be self contained, meaning the user shouldn't need any other devices to use all the functions of the workout bag, this option is less favorable as the only method to provide the user with results. If there is extra time, our team can quickly implement this method as an additional function.

Another approach to provide results to the user, would be to incorporate a screen that displays results. There are several different types of screens that are comparable. The two types considered were a LED monitor and an LCD display. The LED monitor supports inputs of VGA, DVD or HDMI. This would require a processor that is fast enough along with a framer chip to support HDMI output. The LED monitor would be viable, if the processor in the handheld remote is a beaglebone (or similar), however, the cost of the LED monitors is far above our budget for displays, roughly $200 to $300. The LCD display (2004) is a compatible display with the microcontroller (by using I2C or SPI) and is within our budget.

The LCD displays also come with a display driver, therefore no additional driver chips are necessary. Although the display can only support 4 rows with 20 columns, it is sufficient for the data that is being displayed.

Receiving User Input

In order for the workout bag to be an interactive design, there must be a way for the UI system to receive user inputs. User inputs include selecting the workout mode for the session, selecting a difficulty level, and a way to begin, cancel, or exit the workout session. To reduce the complexity of the design, the types of user input options considered are limited to switches, buttons and LED lights, instead of considering any touchscreen inputs.

The best option to receive the user input would be a push button for each type of user input, instead of toggle switches. If toggle switches were used, each type of mode and difficulty setting, resulting in almost twice as many toggle switches as push buttons. In order to reduce the amount of push buttons, the options will need to be cycled through each time the button is pushed.

The user input also requires a way to indicate to the user which selections they have made, since the push buttons have to be pressed to cycle through the options. Two methods to complete this would be to have an LED light that is enabled by each option or have the option displayed on the LCD screen. Having an LED enabled for each option would require several extra components that use up several of the GPIO pins, therefore displaying the option on the LCD screen is the best choice.

Communication between the Base System and the User Interface

The UI system of the workout bag requires a way to communicate with the base system of the design. One way for the base to communicate to the UI system would be a wired connection, however, due to the amount of physical movement around/near the base system, there is a high possibility of the user tripping over the wires and causing injury to the user or damage to the device. Several different approaches were examined and compared.

There are several types of wired connections that are comparable. The two compared for this project were SPI and UART. UART stands for universal asynchronous reception and transmission. The advantages of using UART are the simplicity of implementation and operation, the lack of need for a clock, and the use of error detection (parity bit). However, UART’s frame sizes are limited to 9 bits and transmissions speeds are relatively low. Another disadvantage is that each chip's baud rate must be within a certain percentage of each other to avoid data loss.

SPI stands for serial peripheral interface. The advantages to using SPI is that the implementation and operation are also very simple. SPI is also able to send data continuously due to not requiring a start and stop bit for data transmission. SPI also has a faster data transmission than UART. However, there are downsides to using SPI as well. Such as the lack of flow control, receiving acknowledgments, and the use of more pins limiting the number of devices the chip can support.

There are also several types of wireless connections that are comparable as well. Three types of wireless communication types that were compared are WIFI, Bluetooth and Zigbee. The advantages to using WiFi allow each MCU to connect to the home's router, which allows the UI system to communicate with the base without line of sight. However, WIFI modules consume more power than the other options, which is critical when the UI system is battery operated and they are usually more expensive. The advantage to using Bluetooth is its relatively low power and inexpensive, unlike WiFi. However, the range of the communication is limited to relatively close range. The advantage to using Zigbee is that there is no need for a router and can operate in larger distances than Bluetooth. Zigbee also consumes a low amount of power, which is suitable for battery operated devices. The disadvantage to using Zigbee is the communication requires line of sight to the other device, and has a relatively low data rate.

The best fit for our design would be using Zigbee for the communication between the UI system and the base system. The UI system is battery powered, therefore, Zigbee's low power consumption has a high impact on the design decision. Although the data transmission rate is relatively low, our design only needs to transmit a couple bytes of data. Because this will be our teams first exposure to using Zigbee, an alternate choice of using SPI for a wired connection, as our team is familiar with this interface.

Design Matrix and Analysis

Category/Standard	Options	Ranking		
4.1: Display Results	LCD Screen (LCD2004)	3	■	■
	USB Port	2	■	■
	LED Monitor	1	■	■
4.2: Receive Input	Buttons with LCD	■	3	■
	Buttons with LED	■	2	■
	Toggle Switch	■	1	■
4.3: Communication Base ↔ UI	Zigbee	■	■	5
	WiFi	■	■	4
	Bluetooth	■	■	3
	SPI	■	■	2
	UART	■	■	1

Table 9: Design Matrix for UI System

Standard **4.1** (provide results to user)'s highest rank was given to the LCD screen, as it is compatible with many processors, and can immediately display results to the user. However, this is not a finalized parameter of the design, as the display will be the last component that is configured as its dependent on the processor selection and type of communication used.

Standard **4.2** (receive user inputs) was won by the push buttons w/ LCD Screen to show selection option. Push buttons are relatively easy to implement and to program a debounce method. As push buttons and LCD display units are relatively inexpensive, it is an easy and cheap way to be able to cycle through modes, while showing which sections have been made. Since push buttons can be easily programmed to cycle through modes with repetitive pushes, few buttons are necessary for proper execution.

Standard **4.3** (communication between base and UI system) is best matched in our design by using Zigbee for the communication between the UI system and the base system. The UI system is battery powered, therefore, Zigbee's low power consumption has a high impact on the design decision. Although the data transmission rate is relatively low, our design only needs to transmit a couple bytes of data. Because this will be our teams first exposure to using Zigbee, an alternate choice of using SPI for a wired connection, as our team is familiar with this interface.

Power Supply (Joseph)

Technical Standards

The device's power supply should:

5.1	Connect to a standard US AC 110-120V input.
5.2	Use an AC-DC voltage converter (such as a 20V 60Hz AC to 12V DC converter) to provide power to the system's DC components.
5.3	Provide AC voltage of necessary magnitude to any components that require an AC input.

Table 10: Technical Standards for Power Supply

Considerations for Components

The components that require power are the MCU, the sensors, and the LED strips. These can be powered in combination with a standard AC/DC power supply along with the output power pin with the MCU. The AC/DC power supply will need to take in a 120V/60Hz signal and convert it to a 12V DC signal. This signal will be used to power the LED strips, which require 12V of DC voltage. To power the MCU, an input voltage of 1.8 to 3.6V DC is required. The 12V output of the AC/DC converter will be lowered to this level with a DC/DC 12V-3.3V converter. After the conversion, the voltage will be the sufficient level to power the MCU. The power to the sensors will come from the MCU output power.

AC/DC and DC/DC Conversion

The design for the AC/DC conversion will be the flyback AC/DC converter. The schematic of a flyback converter is shown below.

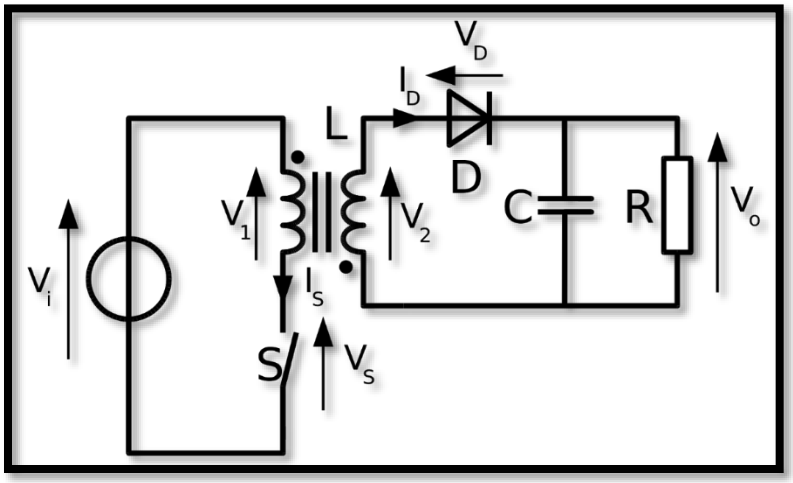


Figure 6: Flyback Converter schematic

This converter uses an inductor as an auto-transformer to "step down" the input voltage. Once the voltage signal is transferred across the transformer the energy in the inductor is slowly built up. Once the switch opens the input signal is interrupted and the energy is transferred to the secondary side of the converter. The diode on the secondary side acts as a rectifier and the capacitor smooths the rectified voltage, allowing for the initial AC voltage to become DC. The resistor in the diagram acts as the output load. Generally, a feedback loop is used to monitor the output voltage to make sure it stays within an acceptable range.

The standard DC/DC converter that will be used in the device is based on the buck-boost converter topology. A model of a buck-boost converter is shown below:

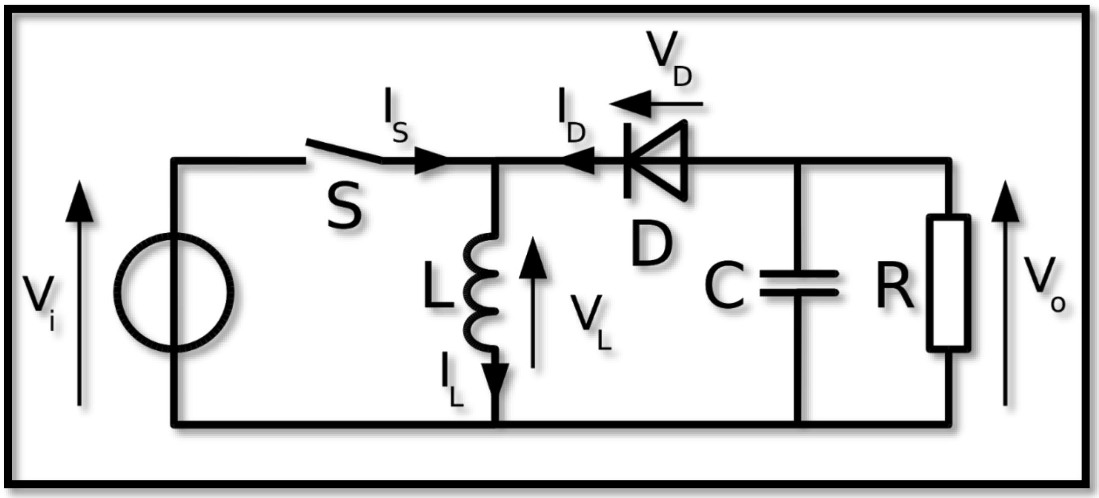


Figure 7: Buck-Boost Converter schematic

There are two operating states with this converter. The first takes place when the switch is closed. While closed the input voltage is allowed to supply the current to the inductor, filling it with energy, during this stage the capacitor supplies current to the output load as well. When the switch is closed, the inductor releases the energy to the output through the diode. In practice, the output current starts low then gradually increases until it reaches a steady state value.

Design Matrix and Analysis

The power supply selection is highly dependent on which processor is ultimately chosen. If the raspberry pi is chosen, the included power supply is what will be used. The Multi output wall adapter has a wide selection of different output voltages and can be used with the FPGA and MCU. The 3.3V wall adapter will be used if the DSP or MCU is chosen to be the processor. The total is the same for all three power supplies as they all have very similar implementations. They all are wall outlet adapters that will plug directly into the processor unit. The Multi output PS has a micro USB outlet. Due to several of the processors use of USB-A input, no matter which PS is chosen (apart from the raspberry pi PS) a micro USB to USB-A adapter will be needed.

Software (Hannah)

Technical Standards and Analysis

The software implemented by this device should:

6.1	Work with both the base system and the UI system.
6.2	Receive user inputs, communications, and computational mathematics.
6.3	Include low power modes while the system waits for input.
6.4	Utilize multiple timers to control the indicators and the sequence.
6.5	Detect when a sensor registers a hit and continue the sequence.

Table 11: Technical Standards for Software

Considerations for Components

The code for the entire project would be split up into two groups, the base and the UI system. For the UI system, the code will need to incorporate user inputs, communication, and simple math. It will also need to be able output the results to a downloadable file (USB connection) or to a LED screen. The UI code will require reading inputs from the buttons, and sending the proper data to the base processor. Next, the code will require to wait in a low power mode until it has received data back from the base, and process the data to provide results such as, number targets, number of hits, hit ratio, selected mode and total time. Lastly, the UI code will send the processed data to either be downloaded by the USB port, or printed to an LED screen.

The base system’s code must incorporate ways to handle sensor input, communications to the UI system, and timers. The base code will use a low power mode until data is received from the UI system, holding the mode of the run followed by the start signal. The code then processes which LED lights go on, and use a timer to keep the LED lit for a certain period of time. Next the code will need to read the sensors that correspond to the segment of where the LED target is. Next, the code will need to determine, from the sensor's data, if the target was hit during the allotted time, and change a register to hold the number total hits, as well as total of lit up targets. Lastly, the code will need to send the data to the UI system receiver, and return to low power mode.

As shown by the set of software standards, the programming language selection is highly dependent on which processor is ultimately chosen. The language in the FPGA is Verilog. The IDE for the MCU can compile embedded C. DSP programming uses C/C++. The language used for a Raspberry Pi is Java/Python/C/C++/C#.

Device Design Concept

This design is largely based on the experience that Natesha has had throughout years of martial arts training and teaching to students of all ages and experience levels. In order to build the necessary skills to be a well-rounded fighter, a martial artist must be able to execute their moves well, and be able to apply it in practice during live sparring matches. This concept is reflected in the dual-side design of the bag: Side A aids application in practice while Side B is geared toward refinement of the move being used.

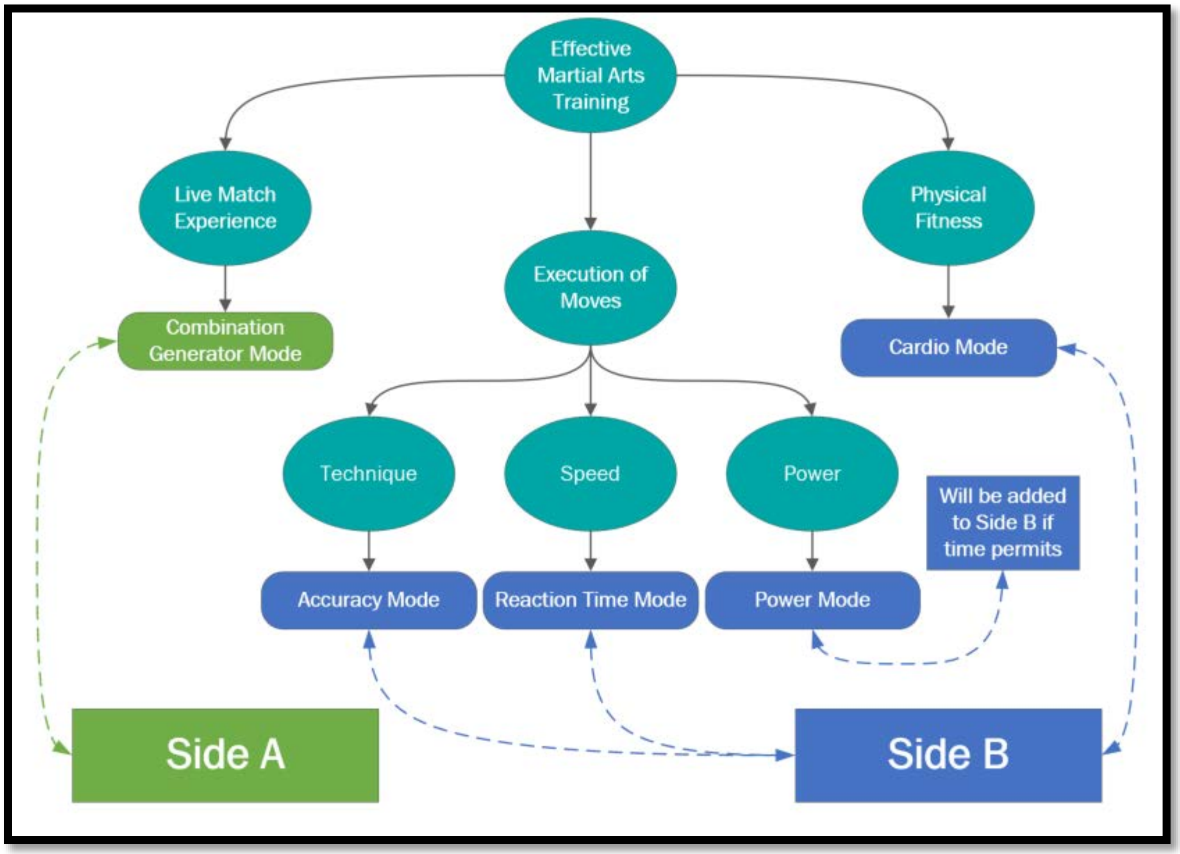


Figure 8: Association map of device active modes

In order to refine a move, such as a punch, a martial artist must first refine the technique (including how the move is executed and where the move is supposed to land on an opponent), then work on speed or timing. Power mode will be added to the device only if there is time to work on it without compromising the other determined aspects of the project. The hardware is available without adding extra budget, but it will be somewhat of a time drain, taking time that could be used to further improve existing features. This method builds on a move from the base up, so that more essential parts of execution in technique are not lost when aspects such as power and speed are added.

Functional Modes of Device's Side B

Side B was designed with these principles in mind. In the center of this side of the bag, there is a target that will be fitted with sensors and indicators to implement training modes inspired by the process of refining a move. Figure X shows the original concept of the target on Side B. The physical target's shape will be largely dependent on the orientation and size of the sensors, and the ability to shape the indicators around the target to sufficiently prompt the user to attack. Addressable LED strips are less conducive to round shapes, so a shape with relatively straight lines, such as a parallelogram, square, or rectangle, would likely be the most suitable for these indicators. Since this zone will include modes where the aim is to hit as close to the center of a target as possible, the best option will likely be a square, where all 4 sides are at an equal radius from the center, even if all points on each side are equidistant from the center.

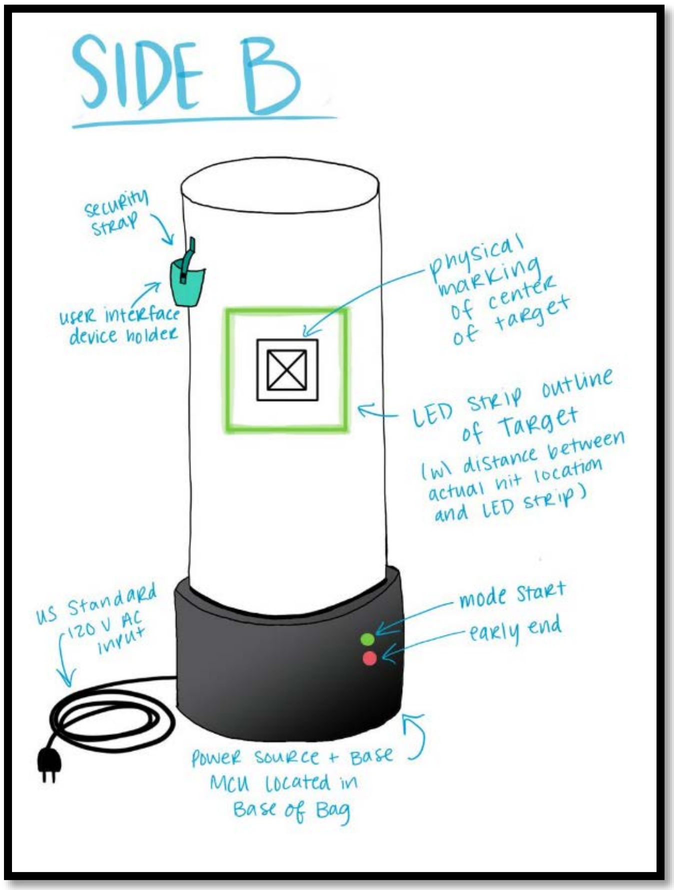

SIDE B
security strap
user interface device holder
physical marking of center of target
LED strip outline of Target (w\ distance between actual hit location and LED strip)
mode start
early end
US Standard 120 V AC input
Power source + Base MCU located in Base of Bag

Figure 9: "Side B" of the device, Annotated

Accuracy Mode

One mode will test the accuracy of the move, so that the user can work on the technique aspect of landing a move where they intend to on an opponent. Using the gyroscope and accelerometer of the IMU sensor, the device will be able to detect the location of a technique that lands on a target marked on the outside of the bag. Using the output registers from the high performance mode, a numerical score will be calculated for each impact. After a specific amount of time, the session ends and the numerical scores will be averaged and displayed to the user so they can save this value and may use it to track their progress as they train.

Reaction Time Mode

Another mode will test the user's reaction time using the sensors and indicators to allow the user to refine the speed of an attack on their opponent while reacting to a live stimulus as they would be in a real sparring match. The target's indicators will light as a timer is set to test the user's reaction time executing a technique. Once the target has sensed a hit, the timer will stop and register the reaction time of the user's technique. In one session, the user will have several reaction times stored that will be averaged and displayed to them. The user can save this value and may use it to track their progress as they continue their training over time.

Cardio Mode

As stated previously, general physical fitness is vital to the user's performance as a martial artist and athlete. With this in mind, a third mode on Side B was designed to augment the user's cardio-style warmup or workout. When using cardio mode, the user will be tasked to rapidly punch the bag for a given time period, with the goal of increasing their heart rate. The time will be determined by the user's difficulty selection: higher difficulty will provide a longer time duration. Their performance will not be measured in terms of accuracy or reaction time as in the other workouts, but instead by measuring how many hits were detected in the given time. Results will be reported to the user in the form of total hits counted and hit rate per second. The user can save their results, if desired, and may use it to track their progress.

Functional Mode of Device's Side A

While Side B focuses on refinement of the user's move execution, it lacks in getting the user in the mindset of using their skills directly against an opponent. Side A was designed to supplement Side B's refinement with the ability to target these techniques at practical locations as they align with the physical figure of an opponent. Key locations that a martial artist aims for on an opponent are highlighted by the indicators, and fitted with sensors that will tell whether or not the zone was hit during an allotted time. As with Side B, the shape of the zones will be dependent on the equipment needed for their proper function. The only requirement of the sensors on this side will be to detect when a hit has been successfully landed; therefore only a binary touch function of a sensor will be required, as long as the sensor is able to detect contact at any location within the zone. Indicators in Side A have the same constraints on the design of the zones as they did on Side B: zones made of straight lines, such as parallelograms, rectangles, and squares will be optimal.

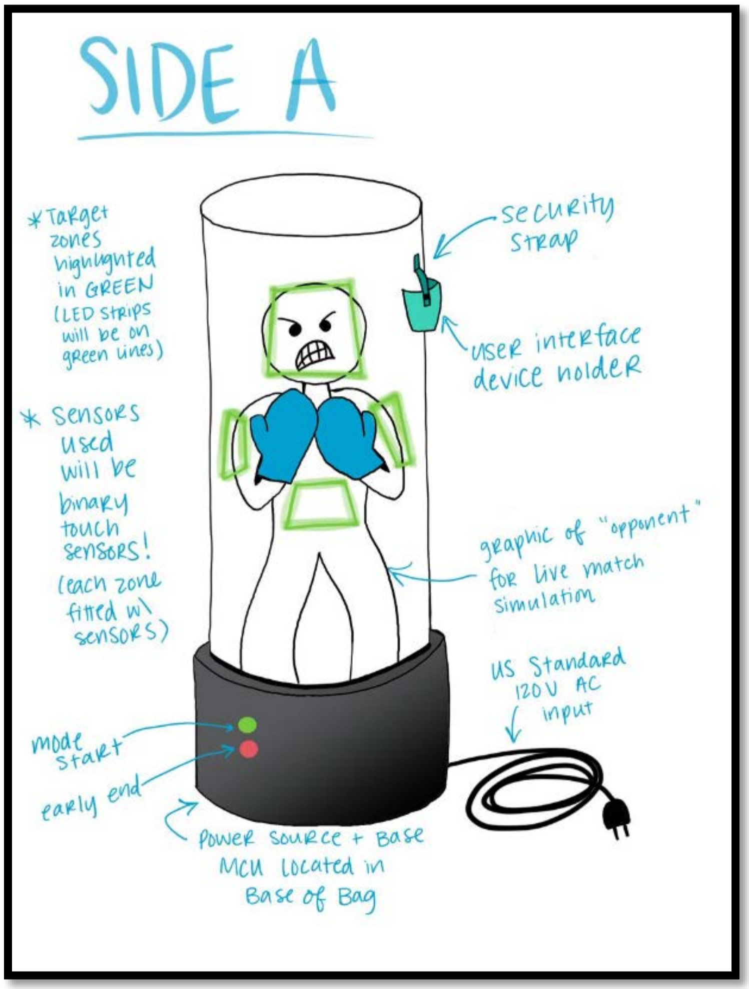

SIDE A
*Target zones highlighted in GREEN (LED strips will be on green lines)
* Sensors used will be binary touch sensors! (each zone fitted w\ sensors)
security strap
user interface device holder
graphic of "opponent" for live match simulation
US Standard 120V AC input
mode start
early end
Power Source + Base MCU located in Base of Bag

Figure 10: "Side A" of the device, Annotated

Combination Generator Mode

This setup on Side A will be used for a “combinations” mode which will prompt the user to hit them while the indicator is on so as to simulate target locations that open for contact on an opponent. Each successful hit will gain the user one point towards their total. After the zone is hit or timer runs out, the indicator will turn off and the next zone will soon be illuminated. Although it will operate similarly to the reaction time mode on Side B, more zones for contact provide an opportunity to expand on the skills used and further allow for growth on the skills practiced on a stationary target. The zones will light up in a random pattern, simulating inconsistently opening and closing target options while in a live match with an opponent. After a specific amount of time, the session ends and the user’s score will be recorded. The user will have the option to save the score so that they can track their progress over time.

General Performance

As mentioned in the descriptions, all modes will gather data to “score” the user on their performance quantitatively. The user will have the option to save this data and add it to the collection of their performance data. All data calculations will be performed on the base unit, and the UI system will display the mode specific results on the screen. In the event that the USB port is implemented, the user will have the option of downloading their session results to the flash drive. Otherwise, the results are cleared, and the user can select a new mode.

In addition to the active modes, the device should also employ passive modes for the time in between the active modes. One such mode will be used when the user is starting up the device and selecting a mode to use. The UI system will enter a low power mode, while the user is actively training with the base, and will exit the low power mode as once the base transmits the results of the session to the UI.

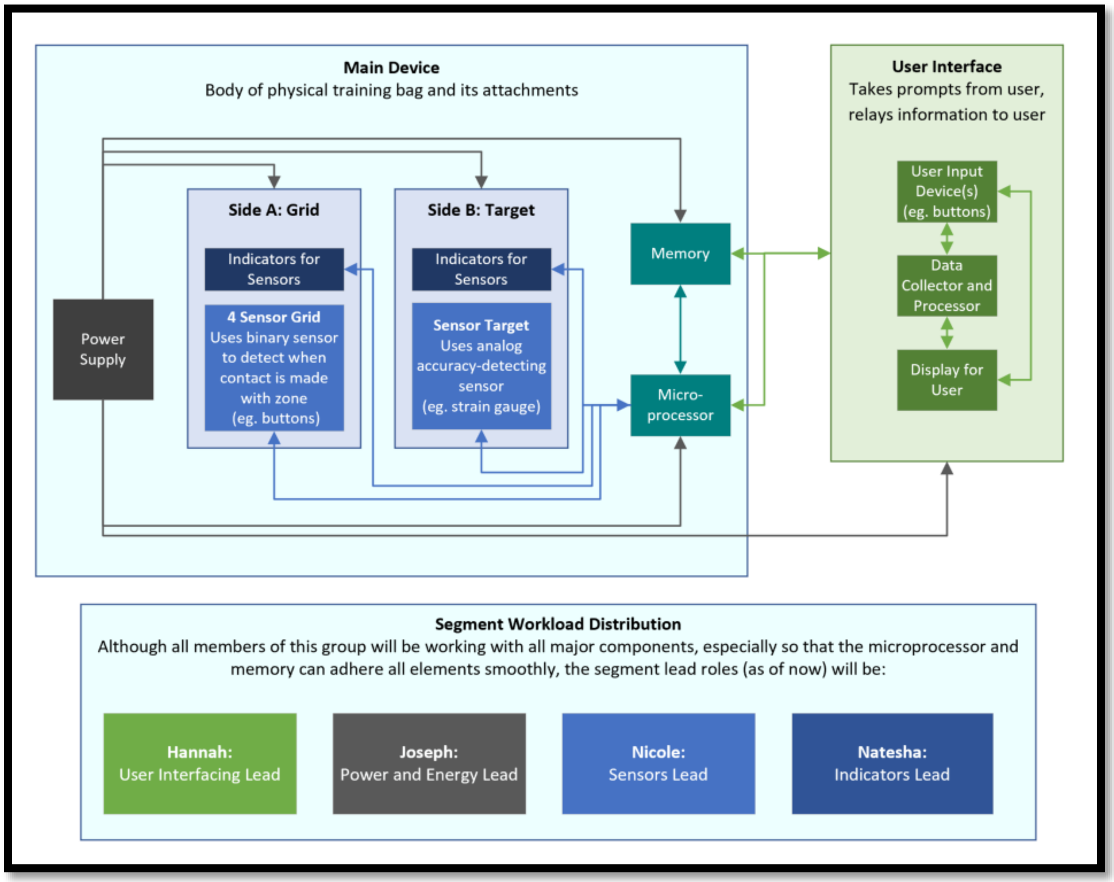
Alternatively, the base system will implement its low power modes while the user is using the remote to select the mode and difficulty. Once the base receives the mode and difficulty setting, it will exit the low power mode. The base returns to the low power mode once the results have been sent back to the UI.

Display/Ambient Mode

The option to add a display mode is also open, and would add a fun, homely touch to this device intended for home workouts. Within the typical home, a large standalone punching bag may be considered an eyesore in all places except the garage or home gym area. To make this device more attractive to consumers, we can program the lights on the device to create patterns with the LED lights that are fun for the user to look at or have running while the bag is not in use. This option is not necessary, but it may help the marketability of the product to consumers who may be on the fence regarding whether or not such an item belongs in their place of living.

Base System Hardware Design

This device will have two components that communicate with each other to make the user experience interactive and immersive. There will be a main device/Base System and a User Interface device. The base system will be based on the standing bag, fitted with indicators and sensors that will gather data on the user's performance. The UI device will be smaller and handheld with a space for safekeeping while the bag is in use. This will be where the user can give input (such as mode of exercise) and receive outputs (such as their exercise statistics).


Main Device
Body of physical training bag and its attachments
Side A: Grid
Indicators for Sensors
4 Sensor Grid
Uses binary sensor to detect when contact is made with zone (eg. buttons)
Side B: Target
Indicators for Sensors
Sensor Target
Uses analog accuracy-detecting sensor (eg. strain gauge)
Power Supply
Memory
Micro-processor
User Interface
Takes prompts from user, relays information to user
User Input Device(s) (eg. buttons)
Data Collector and Processor
Display for User
Segment Workload Distribution
Hannah: User Interfacing Lead
Joseph: Power and Energy Lead
Nicole: Sensors Lead
Natesha: Indicators Lead

Although all members of this group will be working with all major components, especially so that the microprocessor and memory can adhere all elements smoothly, the segment lead roles (as of now) will be:

Figure 11: Hardware connection block diagram; current distribution of group roles

Device Body Build and Design Plan

The body of this device will be a used Wavemaster branded bag by Century Martial Arts. These bags come in various sizes, but the most likely candidates for use in this project are the far left and far right positioned bags in the image below.

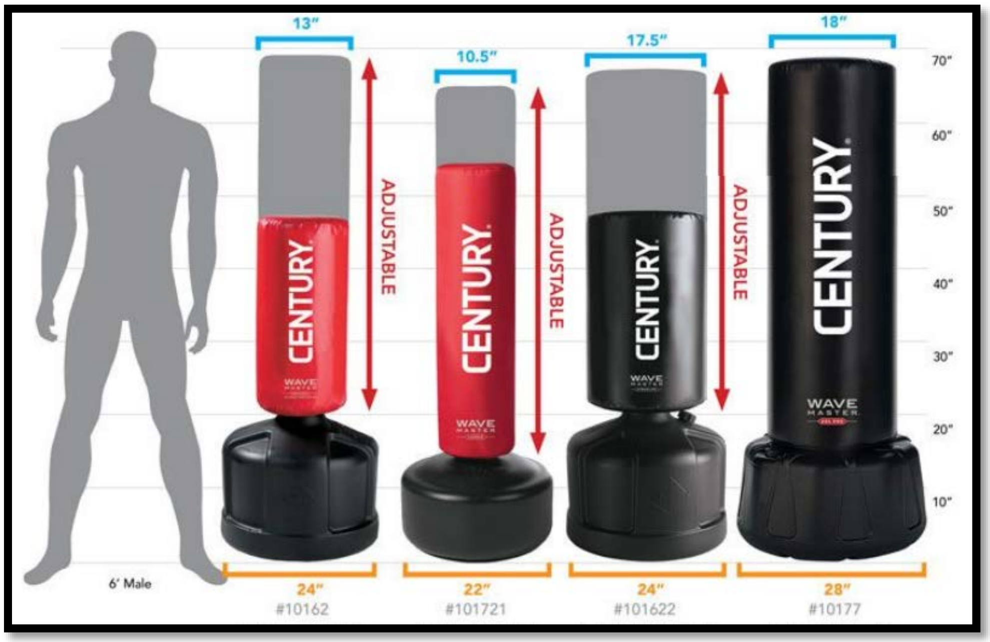
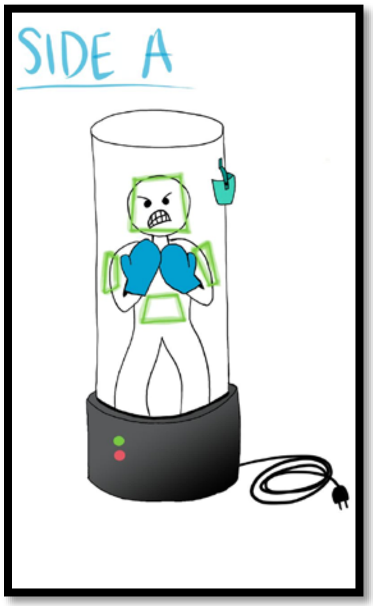


Figure 12: Century Martial Arts Wavemaster options for body base [32]

On this bag, there will be LED indicators designating the zones to hit as well as physical markings to guide the user. Side A's grid of sensors will be laid out to simulate hitting those areas on a physical opponent. Side B has a target showing the center that the user will be aiming for when using modes on this side. For each of these zones, it is required of the markers used to endure large amounts of physical contact without much deterioration of the design. These markings can be seen on the bag designs below.



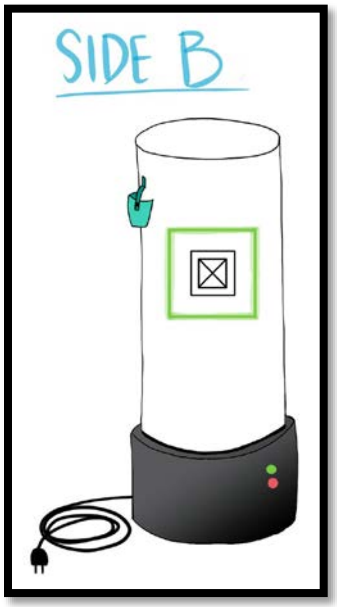


Figure 13 (Left): "Side A" of the device
Figure 14 (Right): "Side B" of the device

To create this, some mediums we could use include permanent marker, paint, tape, and vinyl decals. Permanent marker is cost effective, easy to use, and takes little dry time. Similarly, paint is inexpensive, has an intense pigment even over a black background, and simple to implement. However, extended periods of physical contact will cause the markings to erode and possibly stain the hands of the user. Tape (particularly electrical tape) can cover large zones and make many different shapes. However, the edges may start to raise after lots of contact, and bumps in the tape where the pattern curves may be uncomfortable to hit. Vinyl decal stickers may stand up to prolonged contact the best out of these options, but do not easily cover large areas. This would be ideal to use for small detail, especially in areas that will be frequently touched.

For the side A, the size of the hit areas will be determined on the side of the martial art that we get, so that the hit areas can be sized properly. For side B, the area should be broad enough to have the IMU move and record the direction and the force it moved. Since the bag we will be using for the project will be a used one since we will be modifying the martial arts from the inside as well , we will need to mention the modifications that shall be done on the inside as well . There shall be cabling for power connecting everything and there shall be a button to have an on/off for the whole system. There also should be a sturdy cushion between the point of contact of the user and the sensor to avoid damages done to the sensor and damages done to the user. LED cutouts will surround each target and the light will indicate the user which target to hit, for the program to count and to do the data science for the user interaction data.

For protection of the handheld unit while the bag is in use, a secure pocket will be securely attached to the bag.

Sensor Design: Side B

Based on the technical research presented in the sensors section of this report, the design of this device will make use of binary sensors as well as either IMUs or accelerometers.

Primary Sensor Choice: IMU

The following is a comparison of IMUs we found fitting our requirements: the FIS1000, the LSM6DSRTR, and the MPU-9250. We want an IMU which supports SPI, with at least a 6-axis sensor, and a high resolution. Minimizing cost and shipping time is desirable as well.

Out of these three, the best ones are the LSM6DSRTR and MPU-9250. The major distinguishing characteristic is mounting. The LSM6DSRTR will require careful soldering as a surface mount unit without pins, whereas MPU-9250 already comes mounted with pins for us to use.

	FIS1100	LSM6DSRTR	MPU-9250
Sports Application?	Yes	Yes	Yes
Obsolete?	Yes	No	Yes
LGA Package	3.3x3.3x1mm	2.5x3x0.86mm	3x3x1mm
Current	2.75mA	1.2 mA	3.4mA
Voltage(Min/Mx)	2.4V/3V	1.71V/3.6V	2.4V/3.6V
Output	Digital	Digital	Digital
Interface	I2C/SPI	I2C/SPI & MIPI I3C	I2C or SPI
Sensor type	6-Axis	6-Axis	9-Axis
Temp(Min/Max)	-40C/85C	-45C/85C	-45C/85C
Resolution	16 bit	16 bit	16 bit
Lowest Price per Unit	$4.02 (Digikey)	$4.77 (Digikey)	$8.99 (Amazon)
Stock at Above Source	3000	293	Undisclosed
Time to ship	5 days	Immediately	2 days

Table 12: IMU Candidate Comparison

Backup Sensor Choice: Accelerometer

The following is a comparison of accelerometers we found fitting our requirements. We want an accelerometer which supports SPI, with at least a 3-axis sensor, and a high resolution. Minimizing cost and shipping time is desirable as well.

All of the accelerometers shown are sufficient backup options, but have the same limitations as the surface mount LSM6DSRTR IMU that requires precise and careful soldering on a small scale.

	MC3672	MC3630	BMA490L
LGA Package	1.29x1.09x0.74 mm 8-ball	2.0x2.0x0.94 mm 12-pin	2.0x2.0x0.95 mm 12-pin
Current	-10uA/10uA	-10uA/10uA	13.5uA/150uA
Voltage(Min/Mx)	-0.3V/3.6V	-0.3V/3.6V	1.62V / 3.6V
Output	Digital	digital	Digital
Interface	I2C (1MHz) SPI (4MHz)	I2C (1MHz) SPI (4MHz)	SPI,I2C
Sensor type	3-Axis	3-Axis	3-Axis
Temp(Min/Max)	-40C/85C	-40C/85C	-40C/85C
Resolution	8,10 or 12 bits with FIFO	8,10 or 12 bits with FIFO	16 bits
Price (Mouser)	$3.30	$1.21	$3.80
Stock (Mouser)	321	7448	5333
Time to ship	Immediately	Immediately	Immediately

Table 13: Accelerometer Candidate Comparison

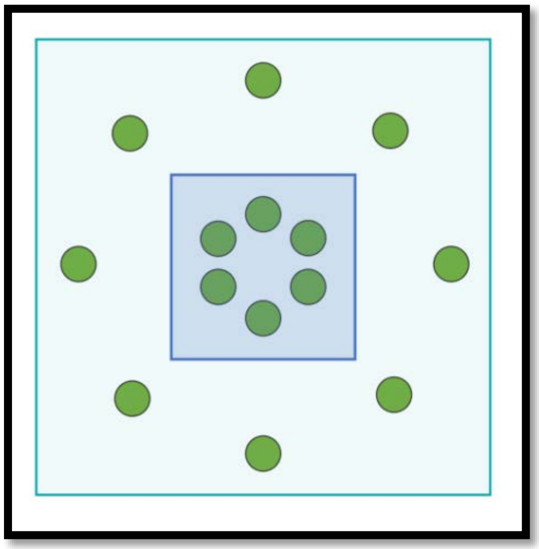
Side B Sensor Layout

The main thing we will do when we get the IMU is to test every single one of them to make sure that they accurately display all the desirable values before we put them in a grid type of system. There are three grid methods that we are planning to set up the building for the IMU sensors for side B.

Method 1: Independent Dual Rings

This method will have two areas, one inner and outer circles. All the sensors will have foam in front of the sensor and the point of contact as well as surrounding the sensors. The sensors will have a little space as well as having fabric elastic bands on the other side to help the IMU to move and give us the accurate values of the impact. The inner sensors will be marking the hit as an accurate hit, and the ones in the outer ones will mention that they are not an accurate hit and how far are these hits from the inner sensors.

Each sensor will have its own quadrant, for example for the inner square: the top left would have the quadrant one, the top middle would have quadrant two, the top right would quadrant three , the bottom left would have quadrant four, and so on for the inner square. We would follow the same dynamic for the outer square but we, top left would be quadrant seven, the top middle sensor would be eight, and so on.



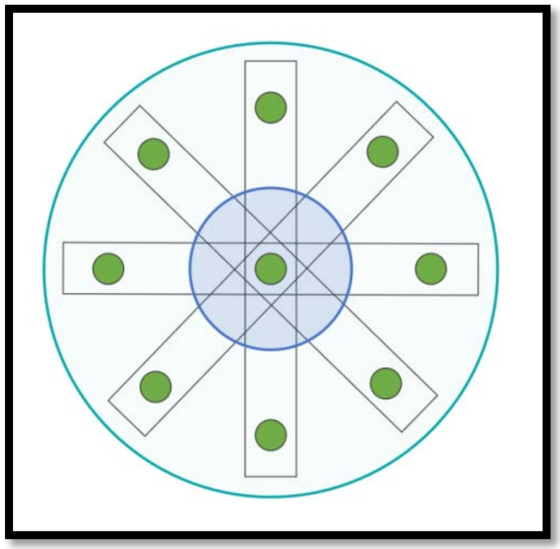


Figure 15 (Left): Visual Representation of Method 1 for side B
Figure 16 (Right): Visual Representation of Method 2 for side B

Method 2: Connected Dual Rings

In the IMUs connection, it will be having a cushion between the point of contact and the sensor and a small empty space to have space for the IMUs to move back and forth, as well as it will be suspended with four fabric elastic bands to help to make the mobility of the IMUs more sensible to measure the 3D plane for the gyroscope aspect of the sensor. There will be a middle sensor that will help if a hit is accurate, and sensors in each band to measure the hit. Nevertheless, the hits made in the outer circle will be also measured respectively from the middle sensor to measure how far were the hits from the center of the target. Each sensor will have its own quadrant, for example : the top left would have the quadrant one, the top middle would have quadrant two, the top right would quadrant three , middle left would have quadrant four, and so on.

Method 3: Square Grid Layout

This method will have 25 sensors spread out evenly by 11 inches by 11 inches square. They will be hanging by elastic bands to help them move accurately after each hit to precise the value of each hit. This method leaves no potential hit area empty, and it gives the program a clearer way to take all the hits. The middle for the center of the target would be six inches by six inches to give enough space for the user to interact with. It also gives us the ability to change the difficulty of each session for the user to interact. Wherever the user hits, the closest sensors will be able to have the highest value of the user impact. If the hit is recorded by two sensors for the accuracy mode, the processor will take the average of both data sets and say the average values of both hits. We will choose method three since this method applies to our desirable requirements for the recording of data from the user interaction.

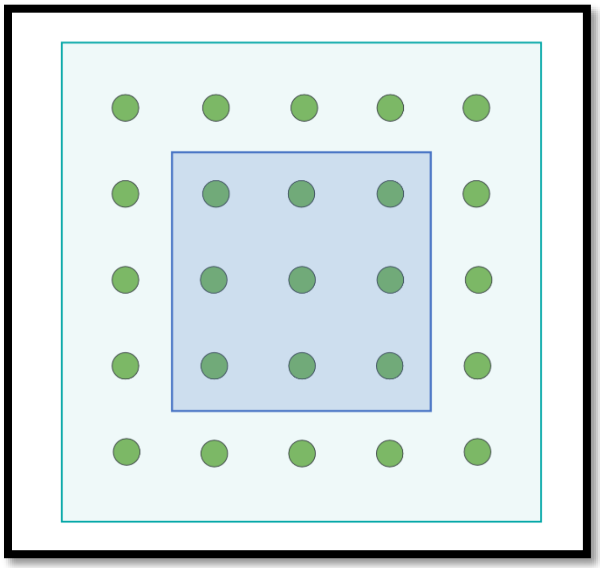


Figure 17: Visual Representation of Method 3 for side B

Sensor Design: Side A

Binary sensors collect information about the state of a device, returning a value of either 1 or 0. They can be made with a similar method as a mechanical switch. To make a binary sensor, we can put two pieces of metal attached by their backs to a piece of wire. One side will be connected to ground and the other one to power. There will be a third metal piece that is connected to the other end of a wire. In between the pieces of metal, there will be a small square foam space in the middle to help them avoid connecting). This method is based on the design of the buttons on a Dance Dance Revolution dance pad. We will be looking for either pieces of conductive metal to be able to cut to shape the switches needed, or for some sort of soft conductive material that will let current flow when contact is made. These materials can be found at stores such as Lowe's, Home Depot, or any local electronics store.

Binary Sensor Build Design

Binary sensors should be made of metal. Many metals are acceptable for the construction of the sensor given that certain conditions are met: the metal should not bend over time and it should be easy to cut into at least 3 pieces per sheet of metal. The image below will act as a guide to create the design of each of the five sensors.

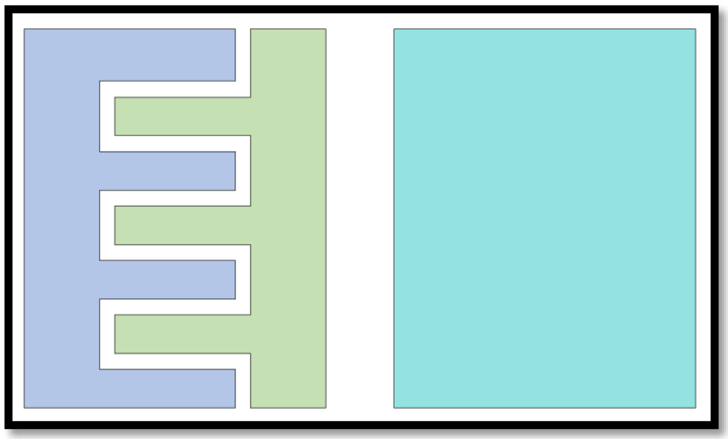


Figure 18: Visual representation of the two layers of the binary sensors for side A

The surrounding area shall have a cushion that will separate the rectangle from the other layer that contains two pieces, and it will only make a connection when it is pressed. One layer of the system, the rectangle on the right side of the image, will be connected to the voltage, and the other layer of the system, the left blue shape on the left side of the bottom image, will be connected in parallel to the processor to count that as a hit and the remaining right green shape will be connected to a resistor to ground. It will be behaving like a button configuration, but it will be on a bigger scale.

Indicator Design

The indicator that was decided upon after technical research is the addressable LED strip.

Choosing an LED Strip

When deciding on an LED light strip for the indicators in this device, the most important requirements are addressable LED cells, RGB LED capability, and the ability to cut and connect these strips in any way that we must manipulate (including short lengths). The strip should be durable, so descriptors such as waterproof and weatherproof are positives as well. We will likely need at least 12 feet of the LED strip to split between the five zones on the device, but we must be mindful of the cost so that it suits our budget.

In the product comparison research completed so far, the most viable option for use in this device is we have found for our needs so far is a BTF-LIGHTING 5 m (16.4 ft) RGB WS2811 Addressable LED Strip reel from Amazon. This product provides more than enough length for our needs. Its LEDs are organized in groups of 3, with each group able to be fully individually controlled. The strip can be cut as needed and resoldered together to make smaller sections of LED lighting, as our project requires. It is also IP67 rated waterproof, so it comes in a sturdy protective sheath.

The price for this LED strip is $20.79 (about $1.26/Ft) which is an ideal cost compared to many other strips we have seen, some of which have exceeded $90. This strip requires a 12 V input power supply and a ground connection. Information is sent to each of the addressable LED cell clusters on the data line of the strip. It should be able to work with the processor that has been selected for this device.

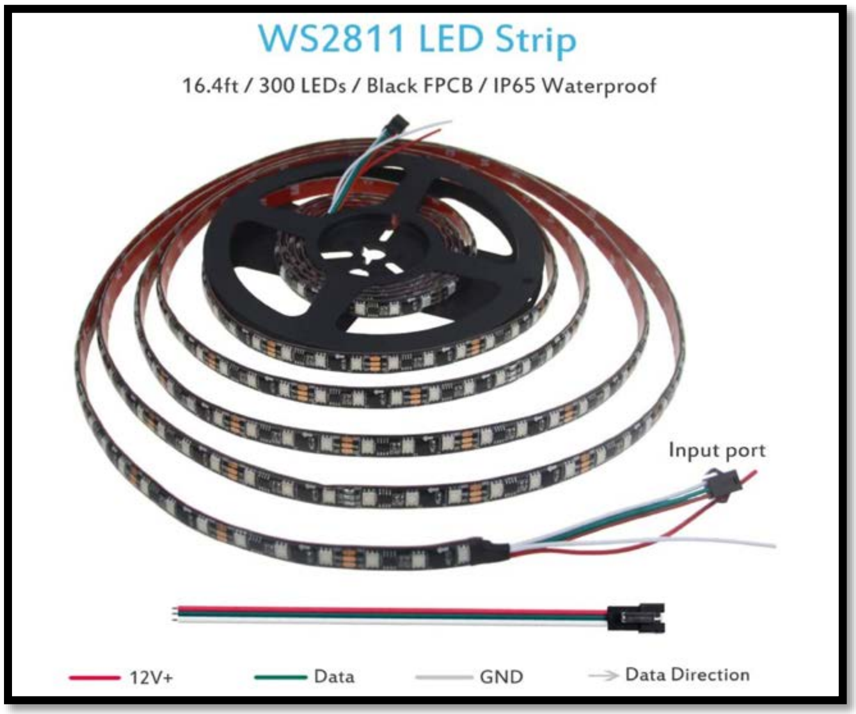


Figure 19: WS2811 LED Strip with wire ribbon configuration [2]

Attaching the LED Strip

Most LED strips come with an adhesive backing, but in experience working with these strips, this adhesive is not always as strong as the companies claim it to be. For this reason, we will likely need to find a different way to physically fit the indicators onto the bag, and make sure that they are secure enough that they will not move when the bag is being repeatedly hit with high speed and/or force.

Power Supply Design

The power supply will originate from the AC outlet and will be converted to a 12V DC signal with an AC/DC converter. This 12V DC signal will be used to power the LEDs directly. This is necessary as the LEDs require a 12V signal to power on. The sensors will be powered using the output power pin from the MCU. This will allow us to both power and read from the sensors using the same device. This will simplify the power design and allow us to use less wires. The MCU will be powered using both the AC/DC converter and the DC/DC converter. First the AC/DC converter is used to convert to a DC input, then the DC/DC converter is used to convert the 12V DC signal to a 3.3V DC signal suitable for the MCU.

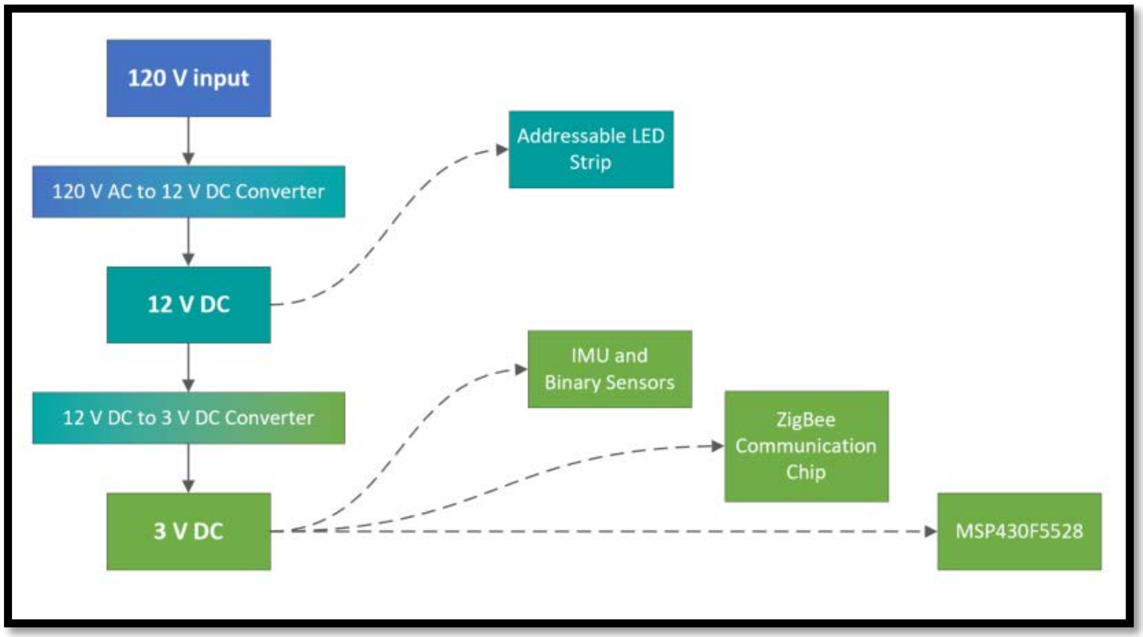

120 V input
120 V AC to 12 V DC Converter
12 V DC
12 V DC to 3 V DC Converter
3 V DC
Addressable LED Strip
IMU and Binary Sensors
ZigBee Communication Chip
MSP430F5528

Figure 20: Flow chart for Power Supply in Base System

Processor Design

TI MSP430F5528

This processor was chosen because it meets all of the technical requirements that were outlined in the research section. This processor, of the original contenders, has the best price to features ratio and meets the technical requirements without going overboard on features. The relevant specifications for this component are listed in the table below. The pinout diagram for this microcontroller is also pictured in the appendix of this document.

RAM	8KB	**SPI Interfaces**	4
ROM	128KB	**UART Interfaces**	2
Clock Speed	25MHz	**I2C Interfaces**	2
GPIO pins	47	**Timers**	16
Input Power	1.8V - 3.6V	**Extended Memory**	Available

Table 14: MSP430F5528 Specifications

The processor will be used to collect data from the sensors and to use that data to calculate results according to the data and power mode. The sensors will communicate with the processor using UART, SPI, and I2C. Most of these interfaces will be used to connect to all the sensors on the punching bag. To power the MCU, the DC/DC converter will be used to convert the 12V DC signal to an adequate 3.3V DC signal.

UI Hardware Design

The UI system that will be used to control the punching bag will be controlled by a handheld, wireless remote. Since the UI component of this overall device is essentially its own individual device with independent hardware requirements from the base, the details of its hardware design will be discussed in this section. The remote will have all of the component sections detailed in the Base hardware section. Each part should be evaluated and picked to best meet the requirement specifications. A sketch of the first draft of the UI Remote can be seen below.

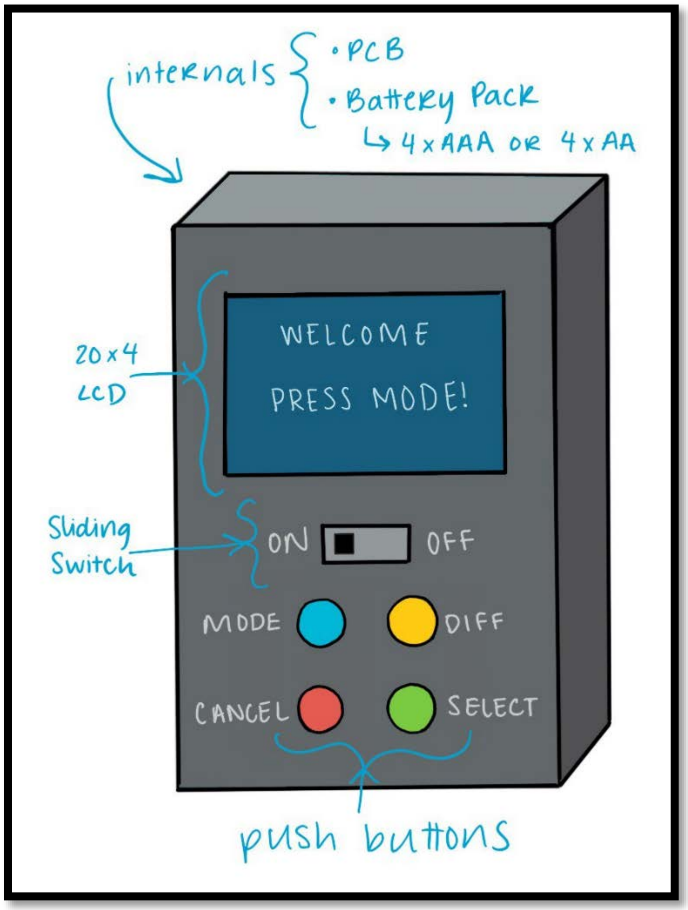


Figure 21: Draft sketch of the UI system for the device

Processor

To control the remote, a processor is required to perform the main calculations and operations based off the inputs for the UI hardware system. The processor needs standard GPIO pins to connect I/O to the MCU. The processor should also be capable of I2C communication. The I2C communication is required to talk to the LCD display on the remote. With these specifications a processor can be chosen to put in the remote. Due to the low expense and the meeting of the technical specifications, the same processor (MSP430F5528) can be used for the machine and the wireless remote.

Display

The LCD display is a 20 column by 4 row screen. The MSP chip communicates with the display controller (driver) to configure the input and output pins. The LCD driver uses I2C and requires two GPIO ports.

Buttons & Switches

Push buttons will be used to control different functions for the remote. The push buttons require GPIO pins to connect to the MCU. One pin is needed for each push button, for a total of four GPIO pins necessary for the four buttons. To read a high value from the push buttons, the push buttons should be pushed in. To set this up in the correct way, the resistors for these configurations should be set to pull down mode.

Another input method needed for the remote is a sliding switch. This switch will act as the power on/off for the remote. To connect the switch to the MCU one GPIO pin is required. The resistor can be set to pull down or pull up as each would have the same functionality on a switch.

Power Supply

The processor requires an input power from 1.8V to 3.6V. The LCD display requires 5V for full functionality and the Zigbee chip is 1.8V to 3.8V. This means that a 5V DC signal is the lowest possible voltage required to power on all of the electronics inside the remote. To generate this amount of voltage using wireless methods four AA batteries will be included in the remote. The four batteries together will supply 6V, which will be enough voltage to power all the essential electronics inside of the remote.

Protective Case

The housing of the remote is required to hold all of the components inside the remote. The only components that will be accessible from the outside are the switch, buttons, and the battery pack on the back of the housing. To design and manufacture a shell that will fit these requirements a 3D printer will be used.

The main advantages of using a 3D printer to fabricate the housing is the greater design flexibility by starting from scratch. This flexibility will allow the design to conform to any constraints that could come up during the manufacturing and designing phases. 3D printing will also greatly decrease the manufacturing time due to general ease of use. Holes will need to be cut out and sanded in order to put the final touches on the remote. The sanding is advised as the 3D print could leave over leftover scaffolding that could interfere with the fittings and the design.

The largest downside of 3D printing is attaining the resources necessary. The cost of materials can be steep if there is a need to re-print due to any errors. There is also the issue of finding the facility to print the case during a time when space is limited in UCF creator spaces. This particular downside, however, can be circumnavigated if we begin prototyping with sufficient time to find an alternative method of case construction.

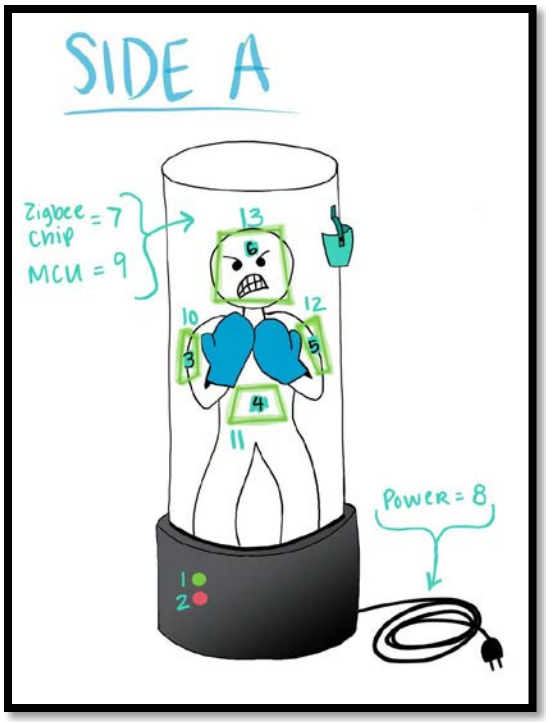
Printed Circuit Board Design

Since there are two main components of this device that each require their own printed circuit boards (PCB). These PCBs will each hold one microcontroller, as well as sufficient connection pins for all power, sensor, and indicator inputs and outputs. The base system will house one of the PCBs, as this is the location on the bag least likely to get damaged by a hit. The UI system handheld component will house the second PCB.

Since we are still early in our timeline, we are still finalizing our design concept for the printed circuit boards needed. At this point, we are in the stage of determining the necessary connections between all components in the device's base system and UI handheld system; the current connections will be detailed throughout this section.

Base System Design

A summary of the connections in the base system device are seen below. They will be detailed throughout this section. There are 15 points of interests between sides A and B: (1) Mode Start Button, (2) Early End Button, (3-6) Binary Sensor 1-4, (7) Zigbee Module, (8) Power Supply and Conversion, (9) MCU, (10-13) LED strips 1A-4A, (14) IMU 5x5 sensor grid, and (15) LED strip B.



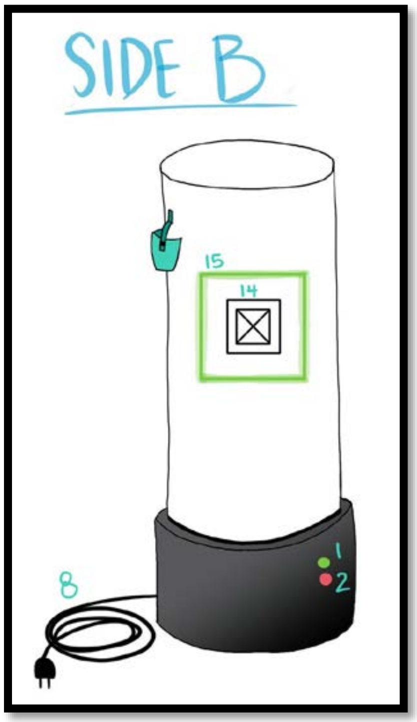


Figure 22 (Left): Side A with Component Reference Numbers

Figure 23 (Right): Side B with Component Reference Numbers

1 – Mode Start Button	P1.1, *(35)
2 – Early End Button	P1.2, *(36)
3 – Binary Sensor 1	P1.3, *(37)
4 – Binary Sensor 2	P1.4, *(38)
5 – Binary Sensor 3	P1.5, *(39)
6 – Binary Sensor 4	P1.6, *(40)
7 – Zigbee Chip	Power from step down 12V/3.3V DC-DC into VCC ^(24, 22, 20, 16, 14, 11, 8) Reset ^(25) -> *(57) Reg: ^(26) -> *(42) UART Send: ^(4) -> *(71) UART Receive: *(70) -> ^(5)
8 – Power	120V AC -> 12V DC from outlet 12V -> 3.3V DC to power MCU
9 – MCU	3.3V DC -> *(11, 25, 64, 89)
10 – LED Strip 1	Power input: 12V DC -> VDD GND -> *(26)
11 – LED Strip 2	Power input: 12V DC -> VDD GND -> *(26)
12 – LED Strip 3	Power input: 12V DC -> VDD GND -> *(26)
13 – LED Strip 4	Power input: 12V DC -> VDD GND -> *(26)
14 - IMU	Power: 3.3V High Rail (-0.5 <-> +6V) 3.3V DC -> *(13)1-25 – %(18)1-25 -> Ground SDA: %(24)^ SCL: %(23)^
15 – LED Strip 5	Power input: 12V DC -> VDD GND -> *(26)
LED Data Transmission	*(54) -> $(6)_4$ -> $(5)_4$ -> $(6)_3$ -> $(5)_3$ -> $(6)_2$ -> $(5)_2$ -> $(6)_1$ – chain NZR Comm. (I2C)

KEY:	
* = MCU Pin	^ = ZigBee Pin

Table 15 : Base System Pin Connection Summary

Mode Start and Early End (1, 2)

The mode start and early end buttons are two master control buttons that are located on the base of the punching bag. The mode start button controls the start of the workouts, when pushed the mode selected on the remote will start. This acts as the control for all workouts. The early end button is used to cancel the active workout. It will end any workout that is currently active. To achieve the proper functionality for these buttons, interrupts will be used. The buttons used are binary buttons with pull down resistors. The mode start button will be connected to P1.1 (MCU pin 35). The early end button will be connected to P1.2 (MCU pin 36). These pins are GPIO w/port interrupts, which is essential for proper functionality.

Binary Sensors (3-6)

Four binary sensors will be used to track the hits to the four hit sections on side A of the punching bag. Binary sensors will be used for side A because the only measure from side A is whether a hit is recorded in the area or not. These sensors require access to one GPIO pin each w/port interrupts. The port interrupts are necessary for optimal control. Binary sensor 1 will be connected to P1.3 (MCU pin 37); Binary sensor 2 will be connected to P1.4 (MCU pin 38); Binary sensor 3 will be connected to P1.5 (MCU pin 39); Binary sensor 1 will be connected to P1.6 (MCU pin 40). These binary sensors require pull down resistors to function properly.

Zigbee Module (7)

The Zigbee Module is a device that creates a personal area network that allows multiple different devices with a zigbee module to communicate with each other. The module is well suited for applications with low-power and low-bandwidth needs. There will be both a Zigbee coordinator (ZC) and a Zigbee end divide (ZED). The Zigbee transmits its signal over the 2.4GHz frequency band. The Zigbee device in the device is the ZED, to allow for the control of the system with the remote. The Zigbee chip needs an input voltage 1.8-3.8V, so the same rail powering the MCU will be used to power the Zigbee. To connect the power, connect the high 3.3V rail to pins 8, 11, 14, 16, 20, 22, and 24 on the Zigbee. To communicate with the MCU, the Zigbee needs to be connected via several pins. First the reset and VREG pins need to be connected to the MCU. To connect the RESET pin, connect pin 25 on the Zigbee to pin 57 on the MCU. To connect the VREG, connect pin 26 on the Zigbee to pin 42 on the MCU. The Zigbee can communicate using UART and SPI. The UART connection will be used to connect the Zigbee to the MCU in the device, as only communication between the Zigbee and MCU is required. To send data to Zigbee via UART, connect pin 70 on the MCU to pin 5 on the Zigbee. To send data via UART the opposite way, connect pin 4 on the Zigbee to pin 71 on the MCU.

Power Supply and Conversion (8)

There are three main components to consider in the power analysis of the device: (1) The 120V 60Hz AC / 12V DC wall outlet converter, (2) a 12V/3.3V DC/DC converter, and (3) a 12V/5V DC/DC converter. The first stage in the power supply chain is the main power from a wall outlet. A wall outlet converter will be used to transfer the energy from the AC wall outlet into a 12V DC signal. A DC signal will be used to power all of the electronics in the device as no functions of the device require a time dependent signal for analysis or power. After conversion, the 12V DC signal will be split into two separate signals, both at 12V DC. One of the signals will be run through a 12V/3.3V DC/DC converter. After conversion, the 3.3V DC signal will be used to power both the MCU and the IMUs on side B as both require the same input power voltage signal of 3.3V DC. The second 12V DC signal will go through a 12V/5V DC/DC converter. After conversion, the 5V DC signal will be used to directly power the four LED light strips.

MCU: MSP430F6459 (9)

The MCU to be used in both the device and the UI Handheld is the TI MSP430f6429 launchpad microcontroller. The input power for this MCU is 1.8-3.6V. The input voltage for the MCU in the remote will be 3.3V DC. To power the MCU, the high line from a DC/DC buck converter (3.3V) will be applied to pins 11, 25, 64, and 89. This has sufficient communication lines and GPIO ports to accommodate all of the modules that will be added to the remote control.

Addressable LED Strips (10-13, 15)

Four LED strips will be mounted next to the sensors on side A. These LED strips will indicate the target for several modes of operation for side A. The LEDs need to be powered and will have to be connected to the MCU to communicate with the rest of the device. To power the LED strips, a 5V DC signal is needed. This signal will come from the 12V/5V converter covered in the power section above. The ground for all of the LED strips will be connected to MCU pin 26.

IMU Considerations and Design (14)

The target and sensors for side B differ in functionality from the targets and binary sensors used on side A. Binary sensors only were needed for side A, but because side B measures accuracy from the center of the target, a different solution was needed to get the data needed to calculate distance to center. This solution required multiple different sensors for one target to get data from multiple different sections inside of the target. The data from one section relative to another gives us an approximate area of contact if all of the sensors are in a fixed, known, position on the target. Once it was decided multiple sensors were needed, IMUs were picked to be the sensor of choice because of their accurate force and acceleration data. Using the relative maximum acceleration data from each sensor, the position of the hit can be found (sensor with highest acceleration will be closest to the hit). Multiple different configurations for the IMU plate were considered.

The main contenders for application were a series of concentric circles of IMUs and a 5x5 grid of IMUs. The grid was eventually chosen as it evenly covers the whole area square target more efficiently than the concentric circle design. The last consideration for the IMU array is the method of powering the array. Each IMU requires an input of 3.3V DC to power on. This can be supplied by the same 3.3V signal powering the MCU. These devices will be connected to ground from pin 26 on the MCU.

I2C Communication

IMU

The IMU sensors communicate with the MCU using I2C. The basics of I2C have been covered in the previous sections. Multiple different "slave" devices can connect to one "master" device using a SDA and SCLK bus. Each successive slave device is connected to the bus in a serial fashion. To connect all of the IMU sensors together, connect all 25 IMUs to the SDA and SCLK bus. The start of the SDA bus is pin 65 on the MCU. The start of the SCLK bus is pin 66 on the MCU. The pin number for SDA on the IMU is pin 24. The pin number for SCLK on the MCU is pin 23.

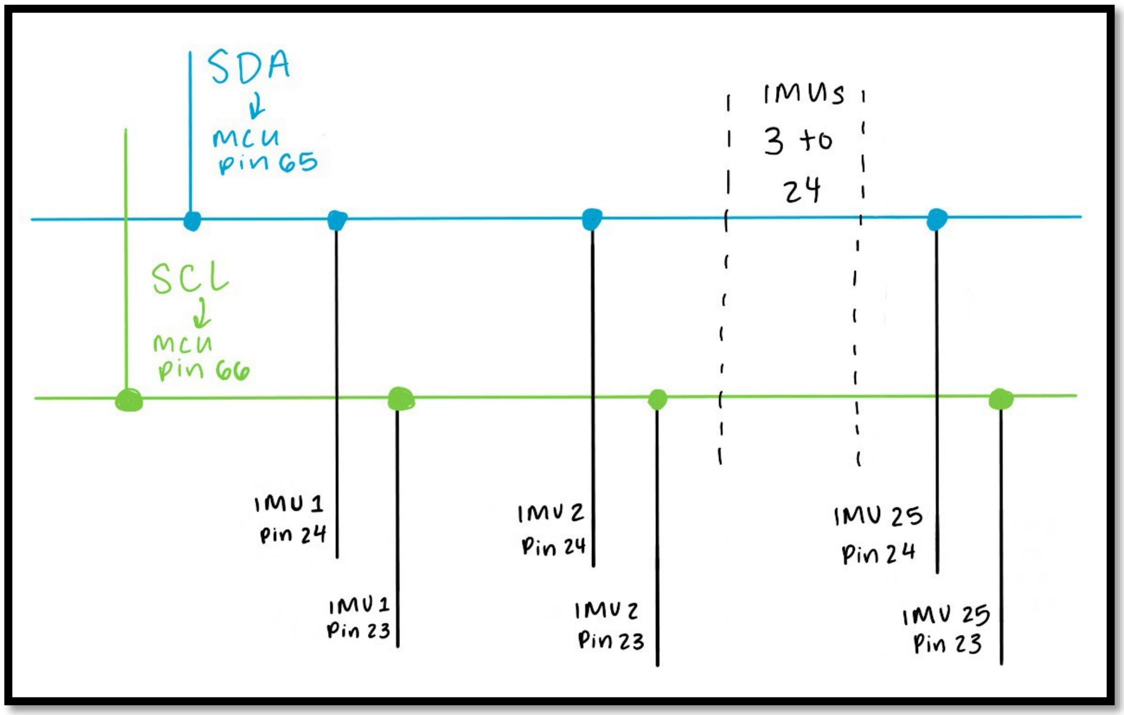


Figure 24 : IMU Communication with I2C setup

Addressable LED Strips

The LED strips have I2C communication available to connect to a controller. I2C allows us to connect multiple “slave” devices to one “master” device. For the light strips, the master will be the MCU and the slaves will be the LED strips. There will be four slaves connected to the SDA and SCL bus. These slaves are connected serially. Only SDA and SCL lines are needed to connect all of the strips together. To start the bus, connect SDA to pin 73 on the MCU and connect SDL to pin 74 on the MCU. Then connect each SDA port on the LEDs to the high line from the MCU. Connect each SCL port on the LEDs to the high line for SCL on the MCU. Once all devices are connected to the bus, communication should be ready.

UI Handheld System Design

There are several points of interest and function on the remote used to power on and control the device. There are a total of 9 points of interest labeled on the diagram. They include: (1) MCU, (2) Zigbee Module, (3-6) Buttons 1-4, (7) LCD Module, (8) ON/OFF switch, and (9) Power Supply. The following sections review the design considerations for the electrical and mechanical design of the UI Handheld.

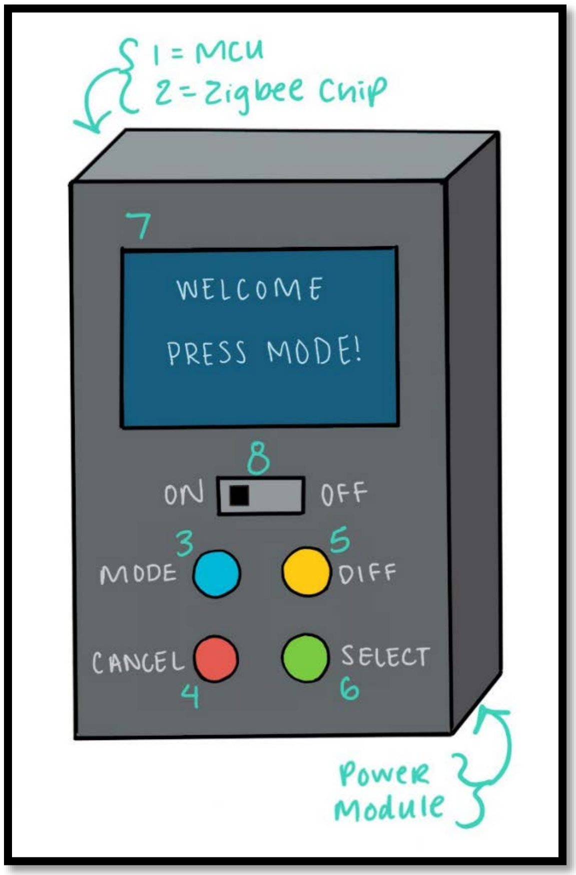


Figure 25: UI Handheld with Component Reference Numbers

1 – MCU	*(25) – DVCC1 (AAAx4 from ON/OFF (8)) *(26) – DVSS1 (Ground of AAA)
2 – Zigbee Chip	Power from *(25) into ^(24, 22, 20, 16, 14, 11, 8) Clock: *(59) -> ^(28) MOSI: *(60) -> ^(2) MISO: ^(1) -> *(61) CS: *(62) -> ^(3) Reset: ^(25) -> *(57) ^(26) -> (42) Optional: GPIO Connection
3 – Button 1	P1.1, *(35)
4 – Button 2	P1.2, *(36)
5 – Button 3	P1.3, *(37)
6 – Button 4	P1.4, *(38)
7 – LED I2C Module	5V -> Vcc (I2C) (AAA) GND -> Battery Ground SDA -> P2.1 *(18) SCL -> P2.2 *(19)
8 - Power Module and Switch	4 AA or 4 AAA Battery Pack
KEY:	
* = MCU Pin	^ = ZigBee Pin

Table 16 : UI Handheld Pin Connection Summary

MCU: MSP430F6459 (1)

The MCU to be used in both the device and the UI Handheld is the TI MSP430f6429 launchpad microcontroller. The input power for this MCU is 1.8-3.6V. The input voltage for the MCU in the remote will be 3.3V DC. To power the MCU, the high line from a DC/DC buck converter (3.3V) will be applied to pins 11, 25, 64, and 89. This has sufficient communication lines and GPIO ports to accommodate all of the modules that will be added to the remote control.

Zigbee Chip (2)

The Zigbee Module is a device that creates a personal area network that allows multiple different devices with a zigbee module to communicate with each other. The module is well suited for applications with low-power and low-bandwidth needs. There will be both a Zigbee coordinator (ZC) and a Zigbee end divide (ZED). The Zigbee transmits its signal over the 2.4GHz frequency band. The Zigbee device in the UI Handheld is the ZC, to allow for the control of the system with the remote. The Zigbee chip needs an input voltage 1.8-3.8V, so the same rail powering the MCU will be used to power the Zigbee.

To connect the power, connect the high 3.3V rail to pins 8, 11, 14, 16, 20, 22, and 24 on the Zigbee. To communicate with the MCU, the Zigbee needs to be connected via several pins. The Zigbee can communicate using UART and SPI. The SPI connection will be used for the remote MCU. To connect the SPI comm, connect the A1 SPI sCLK pin on the MCU (MCU pin 59) to pin 28 on the Zigbee. To connect the SPI MOSI data, connect the A1 MOSI pin on the MCU (MCU pin 60) to pin 2 on the Zigbee with a one way connection. To connect the SPI MISO data, connect pin 1 on the Zigbee with the A1 MOSI data pin on the MCU (MCU pin 61) with a one way connection. The last pin needed for SPI connection is the chip select, connect pin 62 on the MCU to pin 3 on the Zigbee module. Finally the reset and VREG pins are to be connected to the MCU. Connect the reset pin on the Zigbee (Zigbee pin 25) to pin 57 on the MCU. Connect the VREG pin on the Zigbee (Zigbee pin 26) to pin 42 on the MCU. Once fully connected, the Zigbee will talk with both the second Zigbee module and the MCU on the remote with SPI. The Zigbee also has optional GPIO connections to use UART or to free up data from SPI. These won't be used in this Zigbee.

Buttons (3-6)

4 buttons will be used by the user to select different modes for different workouts on the remote. These buttons will need to be set with a pull down resistor so it will read a high input, triggering an interrupt, when the buttons are pressed. The buttons have relatively simple connections to the MCU. Each button only needs one GPIO port w/interrupt to operate effectively. Button 1 will be connected to P1.1 (MCU pin 35);Button 2 will be connected to P1.2 (MCU pin 36); Button 3 will be connected to P1.3 (MCU pin 37); Button 4 will be connected to P1.4 (MCU pin 38). Simple interrupts are the only function of these buttons so these are the only connections needed.

LCD I2C Module (7)

The 20x4 LCD display comes connected to an external I2C module. This module greatly simplifies the connections needed to take full control of the LCD display. It reduces the number of inputs from 4 to 2 through I2C. The I2C has 4 pins that need to be connected. The input power required is 5V DC, this will be applied to the VCC pin on the I2C module. The ground pin on the I2C module will be connected to the ground of the battery pack. To communicate with the MCU through I2C a serial data and clock signal needs to be connected to both the LCD and MCU. 2 wires and 2 sets of pins are needed. Connect the SDA pin on the I2C module to P2.1 on the MCU (MCU pin 18). Connect the SCL pin on the I2C module to P2.2 on the MCU (MCU pin 19). Once these connections are made I2C communication will be available.

Power Switch, Supply, and Conversion (8)

To power the UI Handheld a portable battery pack will be connected to the back of the housing. It should be accessible in order to replace dead batteries. Four AAA (or AA) will be needed to generate a 6V DC input signal. The high signal will be connected to the switch in order to establish an On/Off switch. After coming through the switch the high will be split into two separate signals. One will power the LCD display and the other will power the MCU. To power the LCD one of the high lines from the switch will be connected to a voltage divider resistor circuit to convert the 6V signal into a 5V signal. The 5V signal will then become the input power for the LCD display. To power the MCU, the other high line from the switch should be run through a 6V/3.3V buck boost DC/DC converter. This converted 3.3V signal will be used to power the MCU by connecting it to pins 11, 25, 64, and 89 on the MCU.

UI and Mode Software Design

The software for the interactive workout bag is implemented using agile software development. This method emphasizes the use of adaptive design and development. The technologies integrated in our designs thus far are not set in stone, per say. During development, the technologies, and design choices may change, due to our team taking advantage of adapting our designs and development as new information is uncovered while the product is being built.

With this method of design, it allows us to implement specific components, test their performance, and make adjustments to each component along the way, it also allows our design to be flexible, and to take other design restrictions into account.

Design Environment / Tools

Design Method

The software for the interactive workout bag is implemented using agile software development. This method emphasizes the use of adaptive design and development. The technologies integrated in our designs thus far are not set in stone, per say. During development, the technologies, and design choices may change, due to our team taking advantage of adapting our designs and development as new information is uncovered while the product is being built.

With this method of design, it allows us to implement specific components, test their performance, and make adjustments along the way, it also allows our design to be flexible, and to take other design restrictions into account.

Design Environment / Tools

The software design environment depends greatly on the processor that is used. Because our design implements the use of MSP microcontrollers, Code Composer Studio (CCS) will be used as the IDE to write, push, and debug the program onto the microcontrollers. This choice is beneficial for the team because everyone has experience with programming microcontrollers to perform various tasks, which include the low power modes, interruptions, display drivers, and more. CCS uses embedded C as the program language, which all of our team members have experience in as well. CCS also already comes with the header files and interrupt vectors for the MSP chips, and multiple useful libraries used for complex math and communications, which makes it easier to use as well.

General Overview

The software for the workout bag will be split into two different areas: UI system and Base system. The UI system is a separate device from the base and handles the interface between the user and the system. The base system is the electronic components tasked with reading sensors, providing light sequence patterns, and communicating with the UI system (also called the remote).

In general, the user turns on the device with the UI system. In the UI system, the user selects the mode and difficulty on the remote. The UI system then transmits the users' selections to the base. Once the base receives the mode and difficulty, the workout bag's targets blink to indicate to the user that they can press the start button. The base system then executes the desired mode and difficulty by enabling various timers, sensors and target area LEDs. The base system will then complete the sequence, and compute the results related to the mode, and send them back to the UI system for the user to see. The UI system then returns to the beginning, allowing the user to select another mode and difficulty or to shut down.

UI System

The first action to happen, is when the user turns on the system by using the on/off switch. The system then turns on the screen, and displays its welcome message, and prompts the user for their selection. There are two branches based on the user selection, system off or mode. If the OFF button is pushed, the system calls an interrupt to terminate the program and shut down. If the Mode button is pushed, the user is able to select a mode. Once the Select button is pressed, the user then can choose a difficulty setting, based on the mode. The user then presses select a second time. The mode type and difficulty type are sent to the base system MCU and the UI system enters a low power mode, while the rest of the sequence happens at the base. In order for the user to be in place when the session begins, the workout bag's target areas will blink to indicate to the user that they can begin the session by pressing the start button on the base.

Low Power Modes

The UI system will not be used the majority of the time, since the user's main interaction is with the workout bag. Since the UI system is battery powered, the low power modes on the MCU will be used while the user is completing a session. Once the UI system transmits the necessary data to the base system, the UI system will enter a low power mode to conserve power.

As the UI system receives data, the MCU is interrupted, and begins processing the display sequence for the data that is given based on the mode selected.

LCD Display

Welcome Screen

The LCD display will have several different types of messages printed to the screen. The first screen displayed is the Welcome Screen. On the first row, the display shows "<name of product>". On the third row, the display shows "Press Mode". Because the buttons do not have individual LEDs, the display will show the mode and difficulty selection on the screen, as the user presses the buttons to cycle through the options.

User Input

The user inputs their workout selection by using the buttons under the display. Each button is labeled with their function. The functions for the buttons are Mode, Difficulty, Select and Cancel. The buttons used for the user inputs are susceptible to the bouncing effect, where the user may press the button only once, but the MCU receives several phantom pushes. To correct this, a de-bouncing method is used, that incorporates a small delay that disables the button to avoid the processor receiving multiple button pushes.

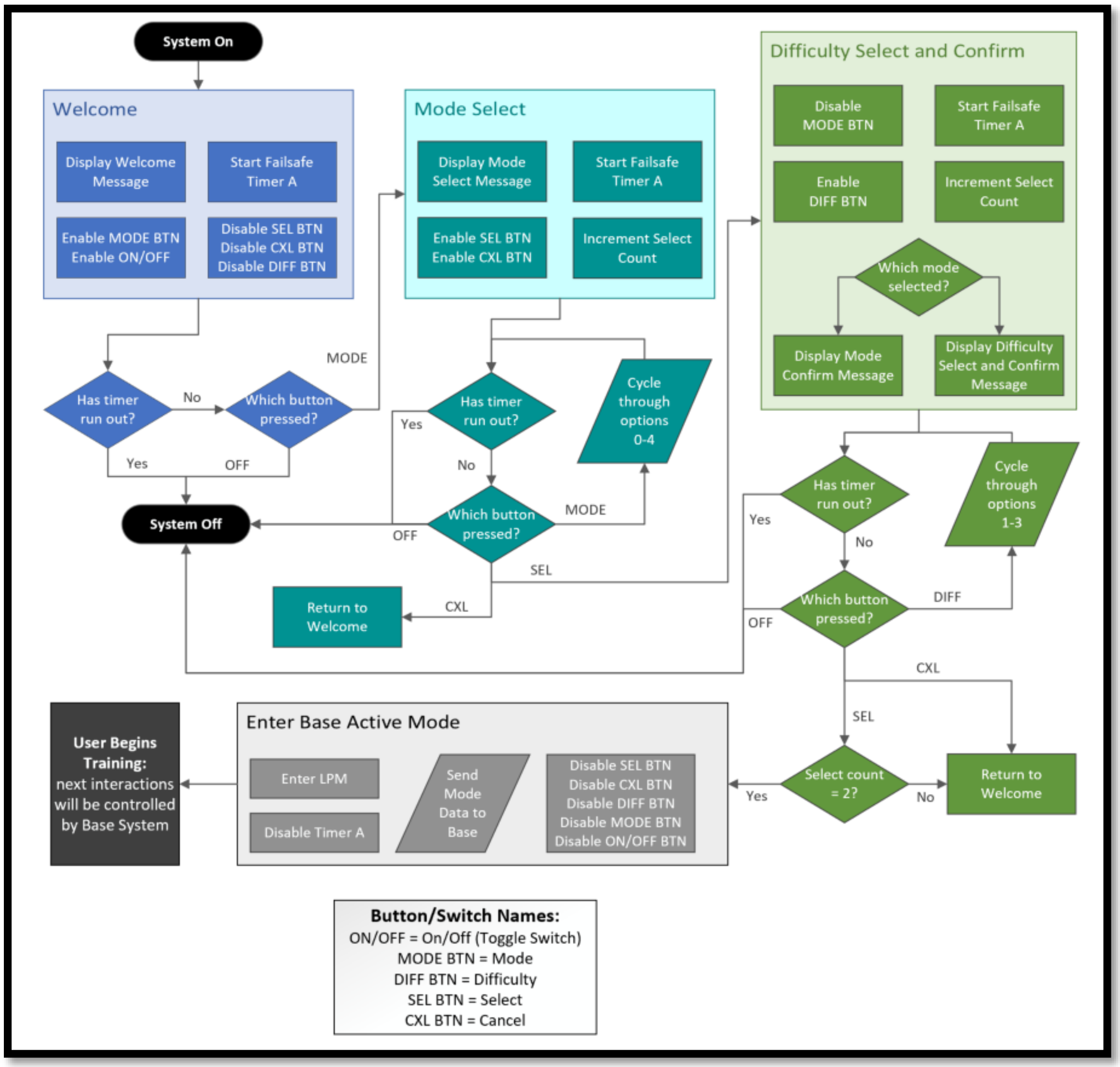


Figure 26: Software flow chart for UI and Initialization

MODE (MODE BTN)

The user is to press the MODE button to cycle through the available modes (5). On the display in the first row, the first six columns display “MODE: “. Columns 7 through 20 display the mode options, where each press of the mode button, changes the mode option that is displayed. Internally, the first time the mode button is pressed, the mode register is set to 0. Each time the button is pressed, it increments the register, until it holds number 4 (last mode), and then returns to 0 and keeps going. To select a mode and continue, the user presses the select button. The mode option and the register values are shown in Table X, below.

Reg Val	Mode Option
0	Combination Generator Mode
1	Reaction Time Mode
2	Accuracy Mode
3	Cardio Mode
4	Ambient Mode

Table 17: Mode options and register values

SELECT (SEL BTN)

When the user has the mode they wish to use on the screen, the select button is used to store this mode, and move on to the next function. When the select button is pushed 1 time, the mode button is disabled and the difficulty button is enabled based on the mode type. Then on the display, the second row reads “DIFF: “. The second time the select button is pressed, the mode and difficulty buttons are disabled and the UI system sends the mode and difficulty setting to the base.

DIFFICULTY (DIFF BTN)

Based on the mode selection, different options of difficulty will be available. The options in total are: Easy, Medium and Hard. Modes 0,1,2,and 3 are able to have different difficulty settings (Easy, Med, Hard), whereas mode 4 does not have a difficulty setting. Only in modes 0, 1, 2 and 3 can cycle through the modes, meaning the difficulty button is enabled for those modes only.

When mode 0 is selected, the difficulty setting is adjusting the amount of time the allotted time to hit a target area, where the allotted time gets shorter as the difficulty setting is increased. When mode 1 is selected, the delay between targets is randomized with higher variances. When mode 2 is selected, the difficulty setting is adjusting the threshold distance for accurate hits. As the difficulty setting increases, the threshold distance reduces, effectively making it harder to complete accurate hits. When mode 3 is selected, the difficulty setting is controlling how long the session lasts. As the difficulty setting increases, the length of the session increases.

When mode is set to 4, the difficulty setting is not applicable. In mode 4, the workout bag is in the ambient mode and is not used for training or measuring the user’s ability.

Reg Val	Difficulty Option
0	Easy
1	Medium
2	Hard
3	X

Table 18: Difficulty options and register values

CANCEL (CXL BTN)

The cancel button is used to start over the selections. When pressed, it resets the mode, difficulty, and select registers, and returns to the welcome screen. This can be pressed at any time until the user presses the select button for the second time. To cancel a workout after the select button has been pressed twice, the user will need to press the exit early button on the base.

The cancel button is also used to return to the welcome screen after results have been displayed to the user. When pressed after the results are displayed, the function remains the same, which returns to the welcome screen to make selections.

UI Failsafe Timer 0

The UI system is highly dependent on the user performing an action (making selections and pressing buttons), which leaves the UI system to be potentially stuck in an infinite loop of waiting for input response. In the event that the user walks away from the UI system, the remote requires a way to shut off to avoid remaining on and wasting battery. To do this, a timer is used that resets every time an active button is pressed. If the timer is able to reach its end, an ISR is called to shut down the UI system.

Results Transmission

After the base system has transmitted the user's results to the UI system, the UI system is interrupted to exit low power mode. The display is turned on, and the result categories and their values are displayed to the user. The results are displayed until the user presses cancel or until the failsafe timer goes off. The display then returns to the welcome screen, allowing the user to make another selection, or shuts off, depending on if the user presses mode, uses the OFF button or the failsafe timer goes off.

Because the LCD display has 4 lines that can hold up to 20 characters, the results category names and values must be able to be displayed within those limits. This means if the category abbreviations are within 5 characters and the results values are 3 characters (0-999), then a maximum of 6 results can be displayed (two columns of 3 rows). This is because the first row of the display is needed to display the users' selections, in the manner of: MODE / DIFFICULTY.

Communication with Base

The base and the UI system will need to communicate in order to send the user inputs to the base and send the results back to the user. To do this, the MCU in the base and the UI system will require a transceiver. The transceiver is connected to the MCU with SPI (4 wire) connections, and uses the IEEE standard 802.15.4 for creating the wireless personal area network (WPAN). To integrate this with the MSP chips, a CC2XXX transceiver chip can be used to support the frame handling, buffering, and transmission. The chip contains the zstack software developers kit to configure the communications between the two devices on the WPAN. This works with IEEE Standard 802.15.4 as shown in Figure 18.

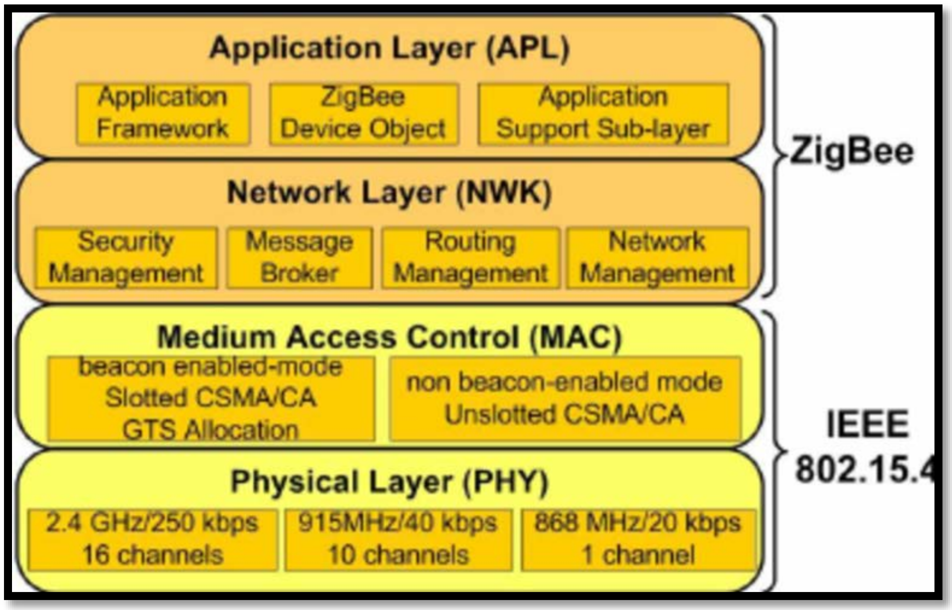

Application Layer (APL)
Application Framework
ZigBee Device Object
Application Support Sub-layer
Network Layer (NWK)
Security Management
Message Broker
Routing Management
Network Management
Medium Access Control (MAC)
beacon enabled-mode Slotted CSMA/CA GTS Allocation
non beacon-enabled mode Unslotted CSMA/CA
Physical Layer (PHY)
2.4 GHz/250 kbps 16 channels
915MHz/40 kbps 10 channels
868 MHz/20 kbps 1 channel
ZigBee
IEEE 802.15.4

Figure 27: Relation of ZigBee to IEEE Standard 802.15.4 [18]

Our team has not had the opportunity to work with Zigbee to create wireless communications between two devices. Therefore, the program's design is not solely dependent on the use of wireless communication. In the event our team is unable to successfully transmit the required data between the two MCU’s, a wired connection is able to be used instead (SPI, I2C, UART). The remote’s pouch is on the side of the bag, reducing the probability of the user tripping on any exposed wires.

Base System

When the base system receives power, the MCU initializes its devices (communication, LEDs and Sensors) and then enters low power mode. Once the base system receives the user's mode selection and difficulty selection from the UI system, the base system will enable its LEDs to cycle through a color sequence. This is to indicate to the user that the base system is ready for the user to press the Start Button. Once the user presses Start, the base MCU will then load the premade sequences based on the mode selected. The base system cannot enter low power modes here, as the processor is actively turning on/off target indicators and capturing sensor data from the target areas that are light up. Once the main sequence timer ends, the processor can begin to compile the data into the results which are provided to the user, and then sends the results to the UI system and returns to low power mode.

Low Power Mode

While the base system is not in use, the MCU will enter a low power mode, to reduce the power consumed during inactive moments. When the UI system transmits data to the base system with Zigbee, the base system's receive flag is raised and an interrupt is called to "wake" the MCU and to begin processing. When the base system has completed its tasks and has sent the results to the UI system, the MCU will return to the specific low power mode. The MCU is able to enter/leave low power modes in the event of transmission or reception through Zigbee.

Target Areas (LEDs)

The workout bag has five different modes: Combination Generator mode, Reaction Time mode, Accuracy mode, Cardio mode and Ambient mode. Each mode will have unique target areas that the base MCU will need to enable to indicate to the user where the hits should take place.

On Side A, the four target areas will incorporate binary sensors that detect whether a certain zone on the bag was hit.

On Side B, the center target area will use IMUs to determine not only if a hit is registered, but also determine the distance of the hit from a center location marked on the bag.

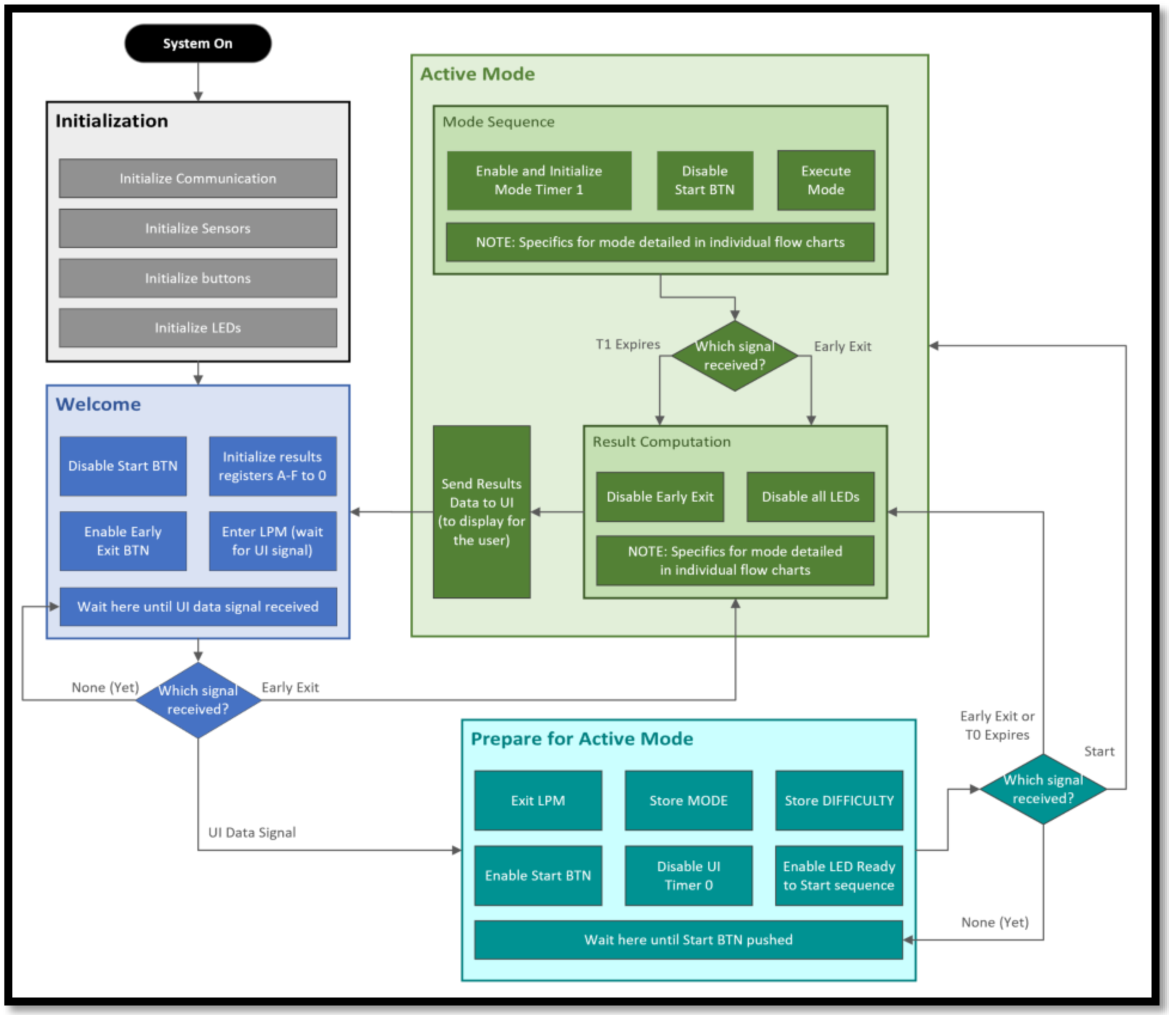


Figure 28: Software flow chart for Base Active Mode Initialization

Combination Generator Mode

The Combination Generator mode is on side A of the workout bag. The addressable LED strips outline four target areas that light up to indicate where the user should hit. Each target area is identified with a numerical value, 0 through 3. In this mode, the MCU will begin its first timer, which limits the session to a specific number of seconds.

Next, the MCU will generate a random number between 0 and 3, and turn on the corresponding target area. Simultaneously, the MCU will start its second timer that controls how long the target area is on. When the second timer ends, an interrupt is called, and the program returns to the random number generator, and the second timer is started. The length of the second timer is variable, as the user is able to select different difficulty settings, where the harder settings result in shorter secondary timers, meaning the user has less time to hit the target area.

The user shouldn't have to wait for the secondary timer to end if they have successfully hit the target area. Therefore, if the user hits the target area before the secondary timer ends, the MCU calls an interrupt to return to the random number generator to continue with the session. Lastly, once the first timer ends another interruption is called, and it ends the session by turning off all LEDs and sensors.

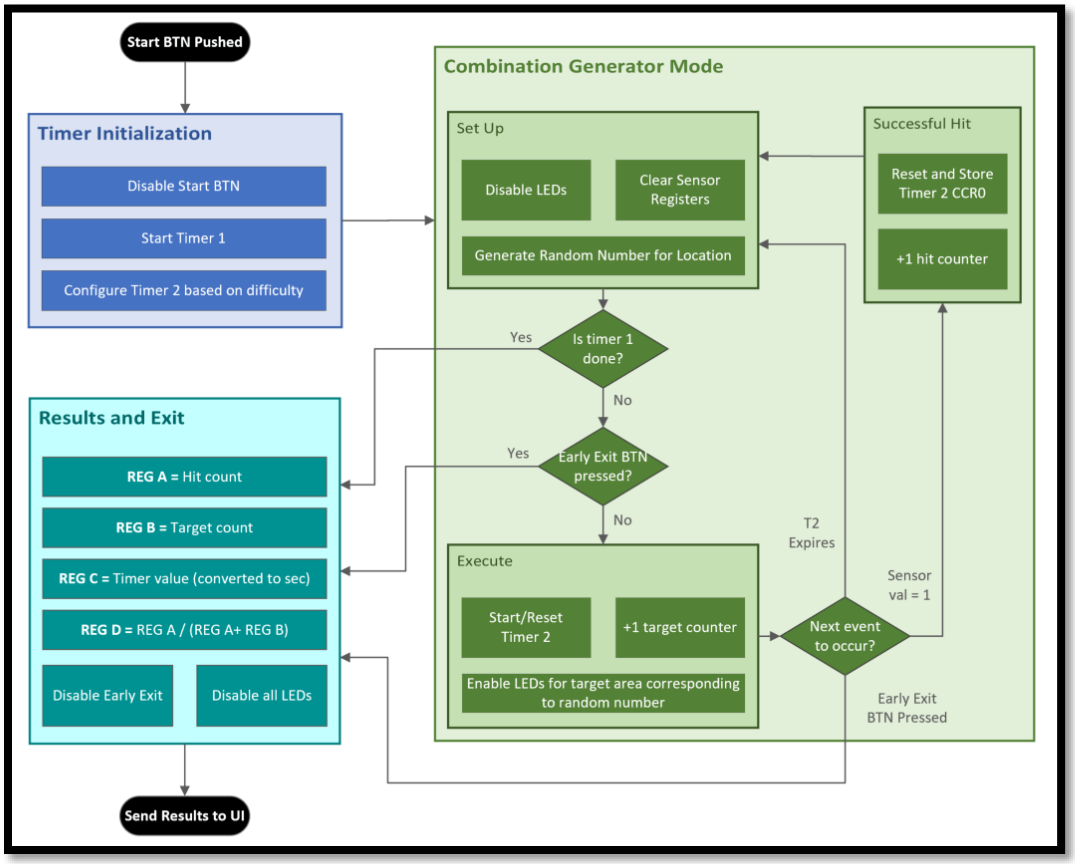


Figure 29: Software flow chart for the Combination Generator Mode on Side A

Reaction Time Mode

The Reaction Time mode is on side B of the workout bag. Side B has one target area and one IMU sensor. For the reaction mode, the MCU starts its first timer, which controls how long the session lasts. This time is set by the developers, and not adjustable. Next, a second timer begins as the target area is enabled. Once the sensor detects a hit, the second timer ends and the target area is turned off. The MCU waits in a delay loop for a short period of time and the process repeats. This delay is adjustable, with the difficulty setting. The time from when the target area is enabled (second timer starts) and when the second timer ends is the amount of time the user reacts. Each hit's reaction time is stored, and averaged for the final result, until the first timer ends.

Since this mode is calculating reaction time, the program is dependent on the user's hit to move forward with the sequence. However, the entire sequence is based on the first timer. Therefore, when the first timer expires, the program is able to move on to the results computation, without getting stuck in an infinite loop, however, this will affect the end results.

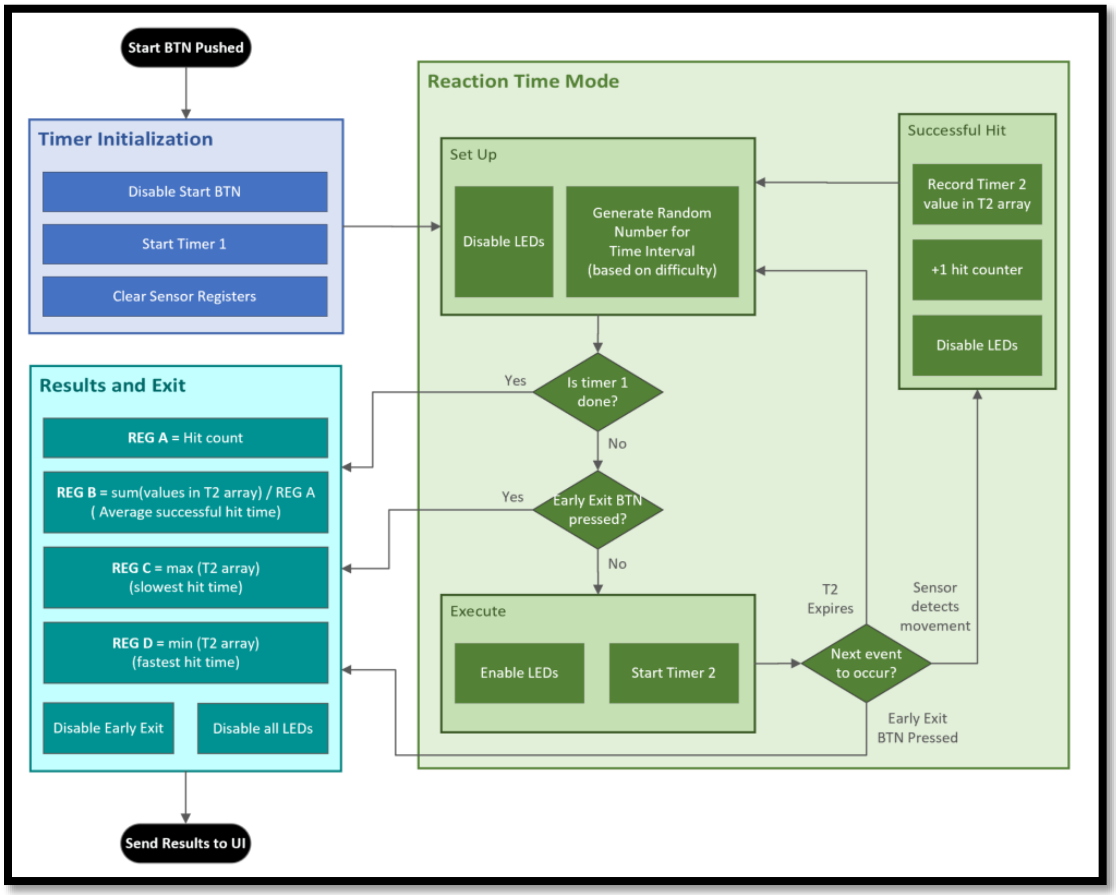


Figure 30: Software flow chart for the Reaction Time Mode on Side B

Accuracy Mode

The Accuracy mode is on side B of the workout bag. Similar to the reaction time mode, the accuracy mode has one target area that lights up. The MCU starts its first timer, which controls the length of the session. Next, the second timer starts when the target area is lit. Different to the reaction time mode, the Inertial Measurement Unit (IMU) sensor is used to locate the hit's distance from the center of the target area, which is marked on the outside of the workout bag. Once a hit is detected, the process repeats itself in the same way as the reaction time mode, however the LEDs do not turn off. The difficulty setting changes the threshold distance for an accurate hit, instead of the delay, as in the Reaction Time mode. When the first timer reaches its limit, an ISR is called to exit the session and display the results.

The MCU will have predetermined thresholds of distance to measure the level of accuracy of each hit. The user can choose the difficulty setting, whereas the difficulty settings get harder, the threshold distance becomes smaller. Hits with a distance greater than the threshold are considered a "hit", while hits with a distance smaller than the threshold are considered an "accurate hit".

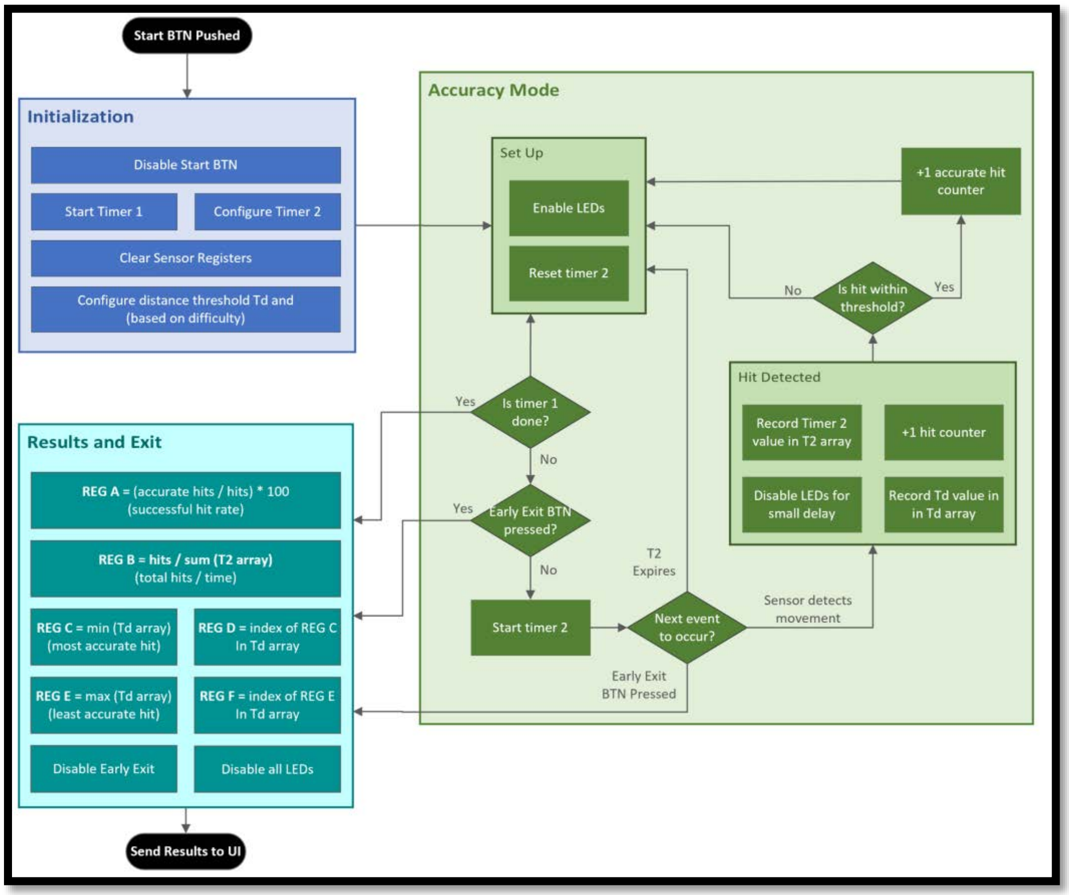


Figure 31: Software flow chart for the Accuracy Mode on Side B

Cardio Mode

The cardio mode is also on side B of the workout bag. In the cardio mode, one target area is used and remains lit for the entire session. The goal is to have as many hits as possible during the allotted time. The MCU will start its timer and the target area will be lit. Because the cardio mode is only counting the number of hits during the session, only one timer is used. Each time the user hits the target area successfully, a register holding the number of hits increments by one. Once the timer ends, the session is over, and the program moves on to calculating and displaying results.

The cardio mode is not dependent on receiving the user's hits to move forward, therefore, if the user decides to not hit the workout bag during the time frame, the score will be affected at the end. If the user would like to end the session without affecting their score at the time (by a significant amount), they will need to use the exit early button.

In the cardio mode, the length of the session is variable with the difficulty setting. Therefore, to make the workout more challenging, the length of the session increases for each difficulty setting.

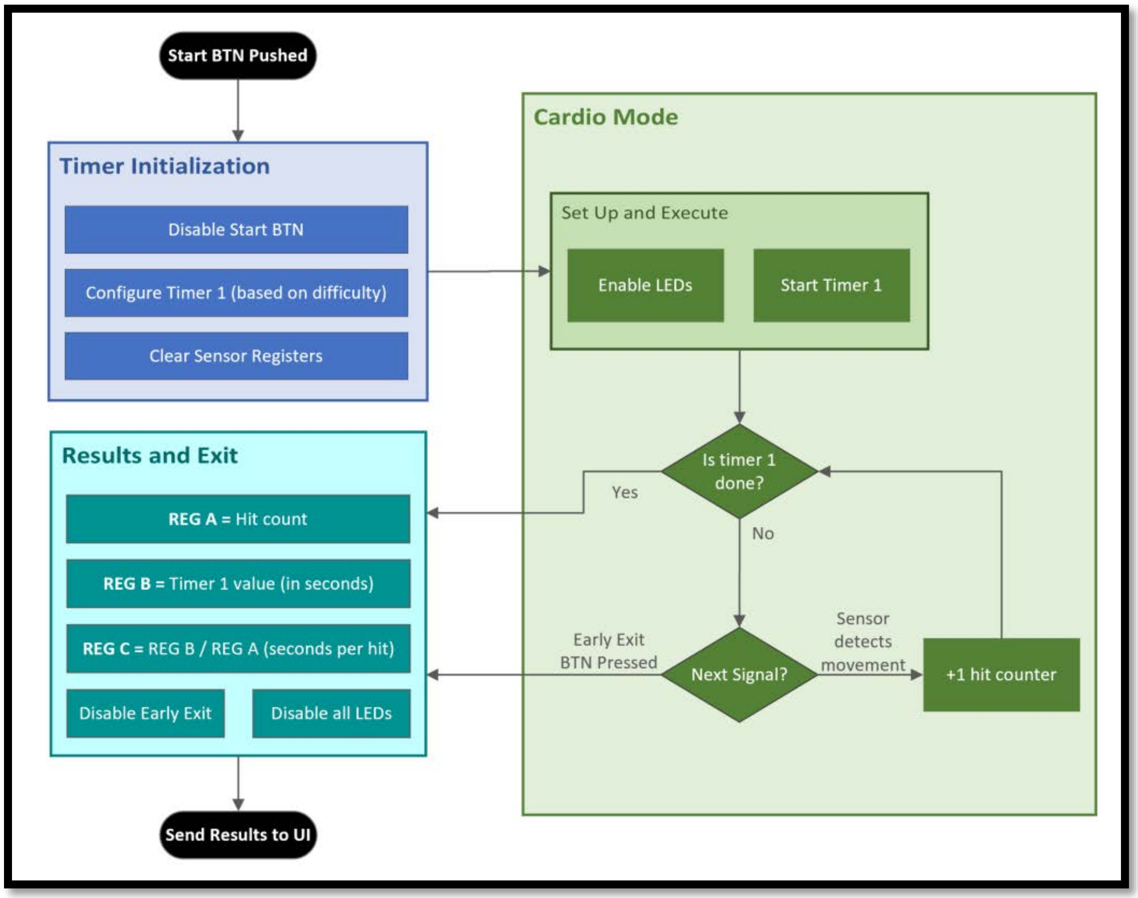


Figure 32: Software flow chart for the Cardio Mode on Side B

Ambient Mode

The last mode that the workout bag includes is the Ambient mode. This mode uses side A and B. In the ambient mode, all the LEDs are enabled, and cycled through the display colors. Because this mode does not require any user interaction, the mode will require an automatic way to shut off after a given amount of time. This prevents the workout from being in an infinite loop with all the LEDs on. This requires the use of one timer, that repeats itself for a given number of times, to allow the Ambient mode to remain on for a decent length of time before shutting off.

The ambient mode can also be disabled by using the exit early button to return to the welcome screen/mode selection.

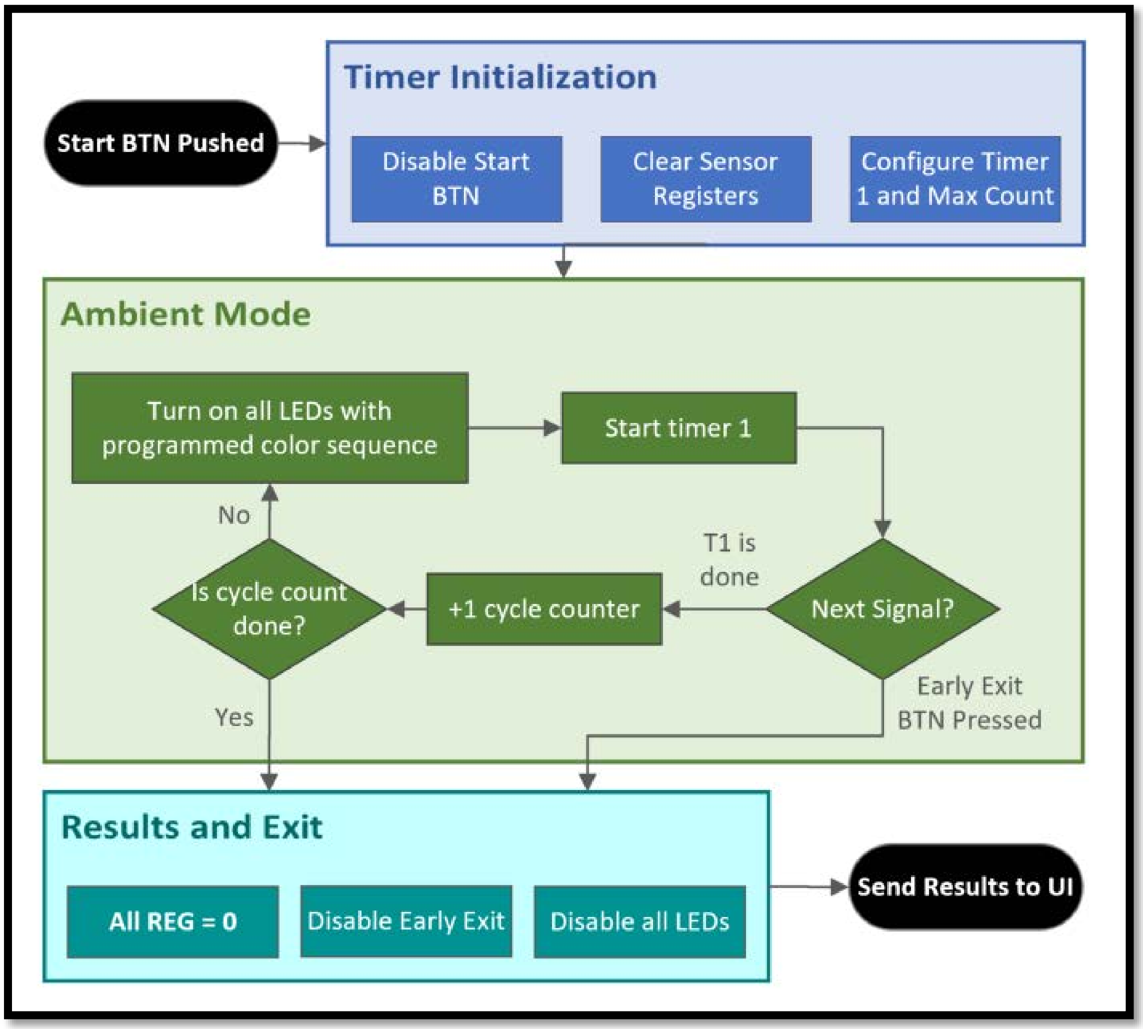


Figure 33: Software flow chart for the Ambient Mode using both sides A and B

Results Computation

The workout bag will have several different methods of computing results based on the mode of the session. In general, the MCU will collect the relevant data according to the mode in easily accessible memory. Once a session is complete, the MCU will perform basic arithmetic to find the desired output values. These are the values that the base system transmits to the UI system.

Combination Generator Mode

In this mode, the user is tasked with hitting different target areas in a specific amount of time. The overall goal of this mode, is to measure the number of successful hits within the given time allotment. Therefore, the results sent to the UI system include the total time of the session, the number of successful target hits, the number of total target areas, and the average time for a hit.

The total time of the session comes directly from the first timer. Because the first timer can be interrupted with the exit button, the program uses the value of the timers compare register to calculate the time by knowing the clock type's frequency. This is done by dividing the total number of cycles (this also needs to be calculated, as the total number of cycles can be higher than the maximum value a register can hold) by the clock's frequency. For example, if the total number of cycles is 32,768 (0 to 32,767) and the clock frequency is 32kHz, then the elapsed time is 1 second.

The total number of successful hits is calculated in real time. As a sensor detects a hit, the MCU increments the register that holds this value. Similarly, the total number of targets is also calculated in real time. Because this mode is timer based, the total number of targets depends on how fast the user is able to hit them. Therefore, this value is also calculated in real time, by incrementing another register every time the target area is updated, meaning when the secondary timer's interrupt is called.

Lastly, the average time for a hit is calculated in two stages. First, each time a hit is registered, the secondary timer value is added into a specific register. This creates a sum of the total amount of time of every hit. In the second stage, the total time of every hit is divided by the number of successful hits.

Reaction Time Mode

In this mode, the user is training their reaction speed, therefore the results that are most useful are the average time for a hit, the number of hits, the longest time for a hit, and the shortest time for hit. These are the four results that will be sent to the UI system.

Similar to the combination generator, the average time for a hit is calculated the same exact way. In the event the user discontinues the session without pressing the exit early button, the time it takes for the system to cycle through will not be counted, since the sensors did not detect a hit, allowing the results to not be affected. The total number of hits are also calculated the same way as the combination generator's successful hits.

The longest time for a hit and shortest time for a hit are found by storing each hit's time in an array. At the end of the session, the maximum and minimum value of the array are found, and stored in separate registers. The array holding all the times is then cleared to avoid keeping unnecessary data. If the sensors do not detect a hit, meaning the user has allowed the secondary timer to expire and the exit ISR is called, the time is discarded so that it will not be stored into the array, allowing the results to be unaffected.

Accuracy Mode

In the accuracy mode, the user is training their ability to produce accurate hits to a given target. The results that are most useful are: most accurate hit (best) distance, most accurate hit (best) reaction time, least accurate hit (worst) distance, least accurate hit (worst) reaction time, accuracy rate, and hit rate. These are the 6 results sent to the UI system.

The best and worst hit distances are found by storing each hit distance into a distance array. At the end of the session, the maximum and minimum value of the array are found, and stored into separate registers. The best and worst hit times are found the same way. Similar to the reaction mode, when the second timer's ISR is called, it means the bag did not receive a hit, and the time is not stored. This keeps the index of both distance and time arrays to be consistent, where the index relates the time of the hit to the distance of the hit from the center.

The accuracy rate is found by using the distance array, where each index stores that hits distance. Each hits' distance is then compared to the threshold value, which is set depending on the difficulty setting. If the hits' distance is less than the threshold distance, the hit is considered accurate and a register that holds the number of accurate hits is incremented by one. At the end of the array, the total number of accurate hits is divided by the total number of hits (length of distance array) to give the accuracy rate (a decimal ratio).

The hit rate is found by dividing the total time of the session by the total number of hits. This allows the user to identify their progress of their accuracy rate with the speed at which they are training.

Cardio Mode

In the cardio mode, the user is performing as many hits as possible in a given amount of time. The most useful results for this mode are total number of hits, total time, average hit time and hit rate.

The total number of hits is found in real time, by incrementing a register each time a sensor detects a hit in the target area. The total time is directly from the timer. Because the timer can be interrupted with the exit button, the program uses the value of the timers compare register to calculate the time by knowing the clock type's frequency. This is done the exact same way as in the combination mode generator.

The average time per hit is calculated by dividing the total time value by the total number of hits. This allows the user to see the average time it takes them to complete a hit. Lastly, the hit rate is found by dividing the total number of hits by the total time value, or inverting the average time per hit value. This allows the user to see the average amount of hits per second they are able to complete.

Early Exit

The system has another way to end the sequence early, if desired by the user. A button located on the side, near the bottom of the base, is an exit button, that allows the user to terminate the current sequence with an interrupt and skip to computing the results and sending them to the UI system. Basically, this button allows the user to call the first timers (session timer) interruption and end the program early. Since the user may want the data from their progress so far, the current results are still processed and sent to the UI system.

LCD Function Analysis and Development

LCD Configuration

The type of LCD used will be the HD44780 that has 4 display rows. Each line is able to display up to 20 characters. Before data can be displayed to the screen, the LCD chip must be configured. The LCD is able to be configured using the register select pin (Rs). When Rs is low (logic 0), the LCD chip interprets the data lines (D0 - D7) as control commands, whereas when Rs is high (logic 1), the LCD chip interprets the data lines as display data. In order to simplify our design, a function will be called that sets Rs to 0, and then transmits a series of control commands to initialize the LCD before any display data is transmitted. Unlike a 1602 display, the display addressing must be configured before each display data is sent. This function will be a return type of void and the argument will be an unsigned integer. By changing the argument integer, the function will transmit a series of preset commands to initialize the LCD in the correct way based on what is desired. For example, to display the welcome screen message, the cursor location and display data address must be configured before any character generator addresses are sent. By using the data sheet of the HD44780 display, the desired control commands can be created. The next several sections summarize the types of commands that will be needed.

Clear Display

One of the most used types of commands will be to clear the display and to set the display data address (DDRAM) to 0 in the address counter, which is the first cell of the first line. Since clearing the display is a control command, the register select (Rs) will be set to 0. When the register select is 0, the read/write register (R/W) is set to low (logic 0). The only bit set to logic 1 is D0. This command essentially sends a space character (code 20H) to the character generator RAM (CGRAM) address/DDRAM address, and then sets the DDRAM address counter (AC) to 0.

The clear display command will be used each time the LCD display is initialized, either when it first powers on or when returning to the welcome screen. The return to the welcome screen function is called when the cancel button is pressed. The clear display command is also sent after waiting a specific amount of time after the select button is pushed for the second time. This allows the user to return the remote to the holder, and press the Start button on the base, before clearing. The clear command will also be sent when the failsafe timer ISR is called, to clear the display.

Return Home

In order to return the cursor to the home position (DDRAM AC set to 0), the return home command can be used. This command leaves the contents of the CGRAM/DDRAM unchanged, meaning the display will not clear. This command also returns the display shift to the original position. To execute this command, the register select is still set to logic 0, and D1 is set to logic 1. D0 is not read for this command, so its value can be logic 0 or 1.

Entry Mode

The entry mode is used to configure the address counter increment/decrement as well as the setting the display shift. I/D (increment/decrement) is set to 1 to increment AC as the DDRAM is written to. Therefore, once a character code is written to DDRAM, the DDRAM AC is incremented so that the next symbol is in the next address over. It is important to note that at the end of a row, the next address does not go to the following row. Instead, it continues on the same line. Line 1 is split across rows 1 and 3. Similarly, S (display shift) will shift the entire display when the DDRAM is written to in the opposite direction that is set by I/D. The effect of the display shift makes it look as if the cursor is standing still. For our purposes, the display shift will always be 0 and the increment will always be to the right (logic 1), meaning the display shift is disabled and text will go from left to right.

Display Control

Display control is used to turn the display on or off, turn the cursor on or off and turn the cursor blink on or off, where on is logic 1 and off is logic 0. For our purposes, the cursor and cursor blinking will always be set to 0 (off), as the UI system is only used to display the user selections and results. The display will be set to on during initialization and set to off when the failsafe timers ISR is called. Turning the display off does not change the contents in the CGRAM/DDRAM. If the power is removed from the LCD, the next time the LCD is turned on, the chip automatically is configured to its original settings. The display control command will need to be sent to configure the settings desired for our projects use.

Cursor / Display Shift

The cursor or display shift allows the display to shift the cursor position (left or right) or the display (left or right). By setting S/C to 1, the display is enabled to shift and setting S/C to 0 enables the cursor to shift. The left or right shifting is controlled by setting R/L. R/L set to 1 shifts right, and R/L set to 0 shifts left. Since our program does not require a display shift for any messages, the S/C bit will always be set to 0.

Function Set

The function set controls several functions of the device. First is the data length (DL), which is how the display chip knows how to read the data lines. For our purposes, we will always use 8-bit data (1 byte). To do this, DL is set to 1. If set to 0, the display chip reads data 4 bits at a time. Next, N controls the number of display lines, where if set to 0 it is 1 line and set to 1 it is 2 lines. Although the user sees four rows on the display, there are really two lines (rows 1 and 3 are considered 1 line). One line can hold up to 40 characters (40 addresses), which would be using rows 1 and 3 for line 1 and rows 2 and 4 for line 2. The table below shows the DDRAM addressing of each row and column, where the addresses are in hexadecimal. Lastly, F is controlling the font size of the characters. Because we will be implementing two lines, F must be 5x8 dot count, which is logic 0.

Set DDRAM Address

To display characters in specific position/row on the display, the DDRAM address can be specified, by setting the ADD bits to the correct hex address listed in the table below. For our purpose, we will always be using a 2-line display, where 1 line is 40 addresses long. Rows 1 and 3 are line 1 and rows 2 and 4 are line 2. This function will be used repeatedly, as several of the screen functions have data displayed in specific areas. For example, the welcome screen is centered across row 1. Therefore, to begin displaying text, the starting address would need to be set using this command.

The UI device displays several strings of characters on the four rows of the display. The set DDRAM address command is used repeatedly to set the cursor to a specific position and row to begin displaying data. This command only needs to be used when the incremented DDRAM address is not the desired position for the next characters sent.

Position	1	2	3	4	5	6	7	8	9	10
Row 1	00	01	02	03	04	05	06	07	08	09
Row 2	40	41	42	43	44	45	46	47	48	49
Row 3	14	15	16	17	18	19	1A	1B	1C	1D
Row 4	54	55	56	57	58	59	5A	5B	5C	5D

Position	11	12	13	14	15	16	17	17	19	20
Row 1	0A	0B	0C	0D	0E	0F	10	11	12	13
Row 2	4A	4B	4C	4D	4E	4F	50	51	52	53
Row 3	1E	1F	20	21	22	23	24	25	26	27
Row 4	5E	5F	60	61	62	63	64	65	66	67

Table 19: Address indices corresponding to row and position of the 20x4 LCD

Write Data to Display

To write data to the display, Rs is set to 1. Because of this, D0 – D7 are read into the CGRAM/DDRAM. Therefore, to send a character to the screen, D0 – D7 are configured to the specific character generator codes. For example, to send the character "H", the following bits are configured, which are shown in the table below. When this happens, assuming the DDRAM address is set at 0x00, the display will print the letter "H" at Row 1 Position 1, and then increment the DDRAM address to 0x01 (assuming in entry mode I/D is set to 1). Therefore, the next time that DDRAM is written to, it displays the character the next position in row 1, which is at address 0x01.

Rs	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	0	0	0

Table 20: CGRAM codes to display 'H' on one cell of the LCD screen

The list of CGRAM addresses and their corresponding character are found in the HD44780 datasheet and is also shown below. Each character's data byte (D0-D7) will need to be defined in the UI system's code. D7-D4 are the upper bits, and D3-D0 are the lower bits, therefore the hexadecimal code for the character "H" is 0x48. This command is used repeatedly throughout the LCD message functions, as it is used to print one character to the display at a time. Therefore, to create a string in a row, the Write Data to Display command is used for each character.

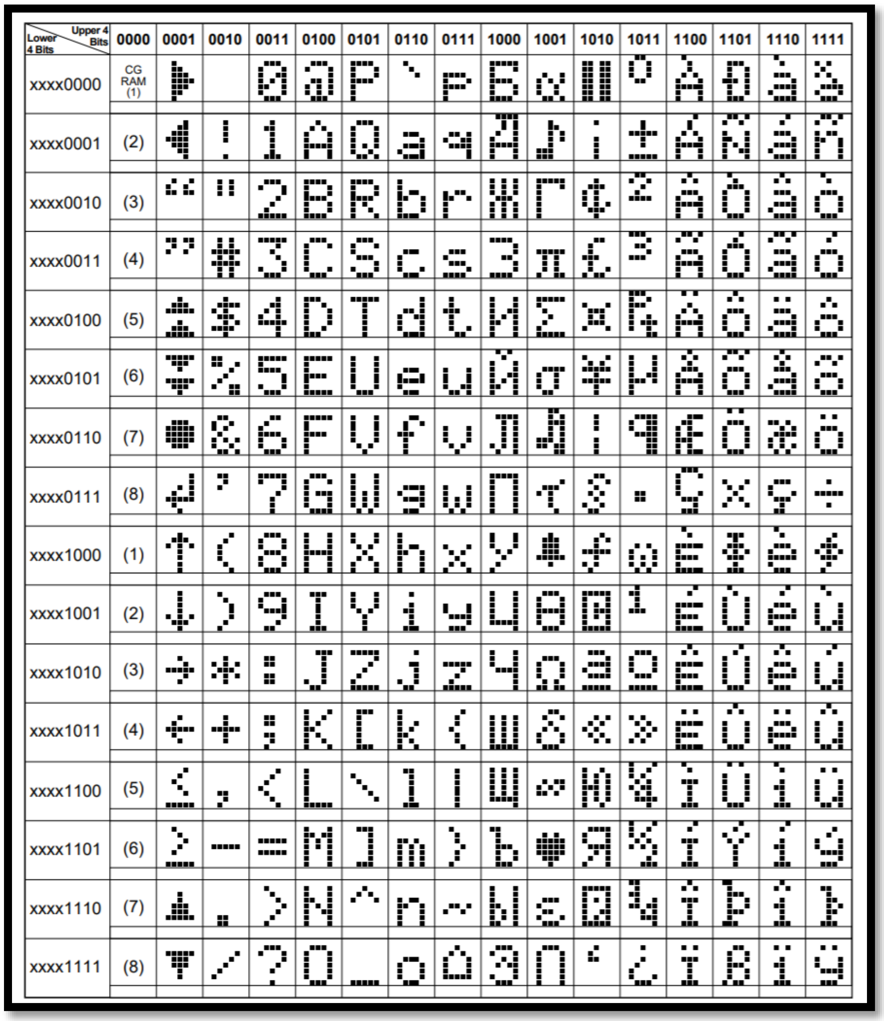

Upper 4 Bits
Lower 4 Bits
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
CG RAM (1)
xxxx0000
xxxx0001
xxxx0010
xxxx0011
xxxx0100
xxxx0101
xxxx0110
xxxx0111
xxxx1000
xxxx1001
xxxx1010
xxxx1011
xxxx1100
xxxx1101
xxxx1110
xxxx1111
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

Figure 34: CGRAM address table

The complete function table discussed in this section can be seen below.

Rs	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Function
0	0	0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	0	0	1	—	Return Home
0	0	0	0	0	0	0	1	I/D	S	Entry Mode
0	0	0	0	0	0	1	D	C	B	Display Control
0	0	0	0	0	1	S/C	R/L	—	—	C/D Shift
0	0	0	0	1	DL	N	F	—	—	Function Set
0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Set DDRAM ad
1	0	Upper bits				Lower bits				Write to CG/DD RAM

Table 21: LCD Basic function table

LCD Message Functions

There are several premade messages that the MCU will send to the LCD display when a specific event occurs. The list of messages are: Welcome Screen, Mode Select, Difficulty Select, Start, and various results messages, depending on the mode. The message functions are called by the various interrupt service routines from either button interrupt flags, timer interrupt flags or receive data interrupt flags.

The table below summarizes the functions needed for the LCD display. This list is not a finalized list, as more functions may be required to make the system perform better. Each category will be discussed in more detail throughout this section.

Function	Commands
Initialize LCD	Clear, Function Set, Entry Mode, Display Control
Welcome	Clear, Set DDRAM, Write to DDRAM
Mode Select	Clear, Set DDRAM, Write to DDRAM
Difficulty Select	Clear, Set DDRAM, Write to DDRAM
Start Screen	Clear, Set DDRAM, Write to DDRAM
Results	Clear, Set DDRAM, Write to DDRAM

Table 22: LCD Message function summary table

Pre-Mode Execution Messages

Power On and Initialization

When the device is powered on, the LCD must be first initialized before any data can be displayed to the screen. One of the first functions called in the main will be to initialize the LCD, by sending the following commands. Once this step is complete, the LCD will continue to increment the DDRAM address as data is written to the DDRAM. This step also configures the display to have 2 lines (4 rows), with a font of 5x8 dots (required for 2-line display).

1. Clear Display
2. Function Set:
    a. DL = 1 (8-bit interface)
    b. N = 1 (2-Line Display)
    c. F = 0 (5x8 Dot Font)
3. Display Control
    a. Display ON
    b. Cursor OFF
    c. Blinking OFF
4. Entry Mode
    a. I/D =1 (Increment)
    b. S = 0 (No Display Shift)

Welcome Message

The welcome message function is called after the LCD initialization function. The welcome function will consist of several commands and data writes. The welcome message consists of "WELCOME" centered in the first row and "PRESS MODE! in the third row, also centered. To do this, the next LCD function to be called will first set the DDRAM address to 0x06, which is where the first letter (W) will print. For the next six characters, the DDRAM address will increment for each write. Next, the DDRAM address is updated to 0x18, where "P" will be printed. For the next 10 characters, the DDRAM address will be incremented for each write. An example is shown below.

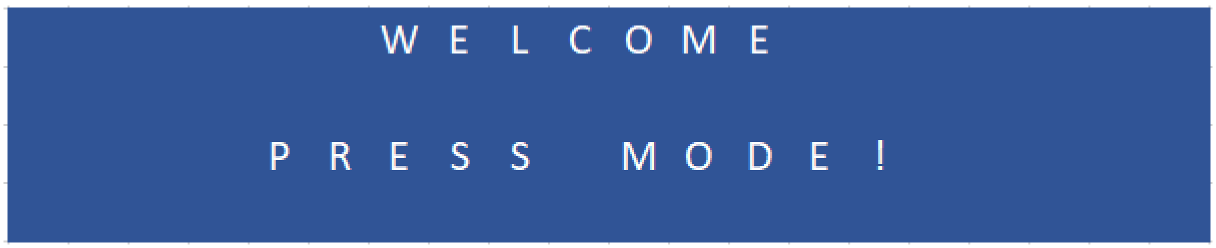


Figure 35: Welcome screen display message sample

Mode Select

Once the mode button is pushed, and the correct ISR completes, the next LCD message function to be sent is the mode select. In the mode select, the first command sent is to clear the display, and the DDRAM is set to 0x00 automatically by using the clear display command. Next, each letter of "MODE :" is displayed in row 1. The first mode displayed is the combination generator mode, which is abbreviated to " COMBO GEN", where the first character is a space at address 0x06. As the Mode button is pushed to cycle through the options, the DDRAM address is set to 0x06 and prints each type of mode corresponding to the mode select register that is updated with the button's ISR. Each mode selection option is shown below, where the space is counted as a character that is sent.

Mode Select Register Value	Mode Chosen (SIDE)	Mode Choice Display
0	Combination Generator (A)	MODE : COMBO GEN
1	Reaction Time (B)	MODE : REACT TIME
2	Accuracy (B)	MODE : ACCURACY
3	Cardio (B)	MODE : CARDIO
4	Ambient (BOTH)	MODE : AMBIENT

Table 23: Mode register values with corresponding modes

Difficulty Select

After the select button is pushed once, the DDRAM address is updated to 0x40, which is the address of the first position of row 2. The letters "DIFF :" are printed in order, since the DDRAM address increments after every write. When the mode select register is 0-3, the first difficulty selection displayed is " EASY", where the first character (space) is at address 0x46. As the difficulty select button is pushed to cycle through the options, the DDRAM address is set to 0x46 and prints the three types of difficulty that correspond to the difficulty select register which is updated with the button's ISR. When the mode select register is 4, the LCD will fill the rest of the 2nd row with spaces, starting at 0x40, as the difficulty select button will be disabled. Each option for row 2 is shown below:

If Mode Select Register Value = 0-3		
Difficulty Select Register Value	Difficulty Chosen	Difficulty Choice Display
1	Easy	DIFF : EASY
2	Medium	DIFF : MED
3	Hard	DIFF : HARD
If Mode Select Register Value = 4		
N/A; no difficulty choice for ambient mode		

Table 24: Difficulty register values with corresponding difficulties

Start

Once the select button has been pushed twice within the process of choosing mode and difficulty, the UI system is sending the mode and difficulty selections to the base. The UI system waits for all data to be transmitted before displaying the next screen. For the next screen, the DDRAM address is updated to 0x58 and displays the words "PRESS START" in the center of the row 4. After a timer's ISR is called, the clear display command is sent, and the display control command is used to turn the display off so the UI systems MCU can enter a low power mode as it waits for results to be sent back.

Since the clear display command is not used during the mode and difficulty selection screens, the user's chosen options remain on rows 1 and 2. An example of the display is shown below:

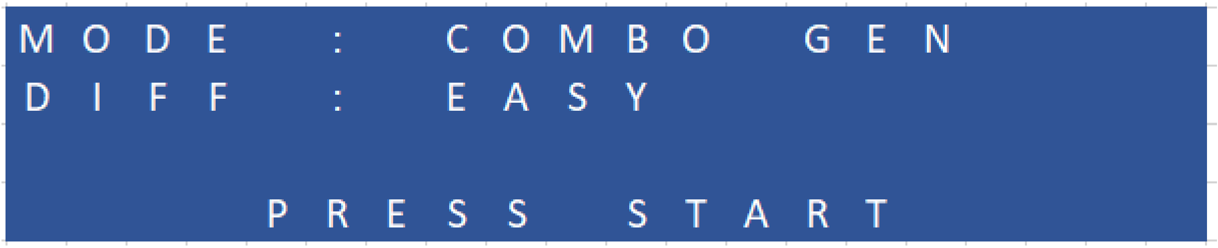


Figure 36: Start screen display message sample

Results Display Message Functions

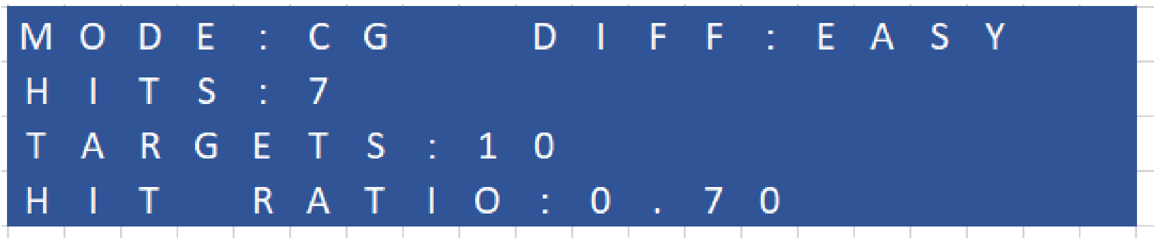
As the MCU wakes up when it receives the results from the base, the LCD initialize function is called first. Next, based on the mode select and difficulty select registers, specific result messages are displayed using row 1. Because the DDRAM address is already at 0x00, the next function sends words "MODE:XX DIFF:YYY/Y", where XX indicates the which mode is used (combination generator = CG; Reaction Time = RT; Accuracy = AC; Cardio = CD) and YYY/Y indicates which difficulty is used (EASY, MED, HARD) when mode select register is equal to 0-3. When the mode select register is set to 4 (ambient mode), the clear display and welcome message are called immediately, since there are no results. The next rows will hold the results, where the message displayed depends on the mode selected. Each modes' results are examined below. In general, each mode's results use a combination of setting the DDRAM address and writing to the DDRAM. In addition, the results numerical value must be parsed into single digits, left to right. This allows each digit to be defined as the character generator code that is sent when using the write to DDRAM command.

Ambient Mode and Early Exit

In the event that the user presses the cancel button or the failsafe timer's ISR is called, the clear display command is sent, and then the welcome screen message function is called again to repeat the process. Note, that if the mode was Ambient, the clear display command and welcome message function are called immediately, as there are no results to display.

Combination Generator Mode

The results that the UI system receives when in the combination generator mode are the number of hits, number of targets and the hit ratio (hits/targets). The DDRAM address is set to 0x40 (position 1 of row 2). Next, the MCU sends data to write to the screen “HITS:##”, where ## is the character that represents the numerical value of the number of hits. The DDRAM address is then set to 0x14 (position 1 of row 3) and sends the data “TARGETS:##”, where # is the character that represents the numerical value of the number of total targets. Next, the DDRAM address is set to 0x54 (position 1 of row 4) and prints “HIT RATIO:X.XX”, where X.XX is a character representation of a float. Each digit of the integer / float value will need to be extracted to send the correct character code. An example of the results screen is shown below.

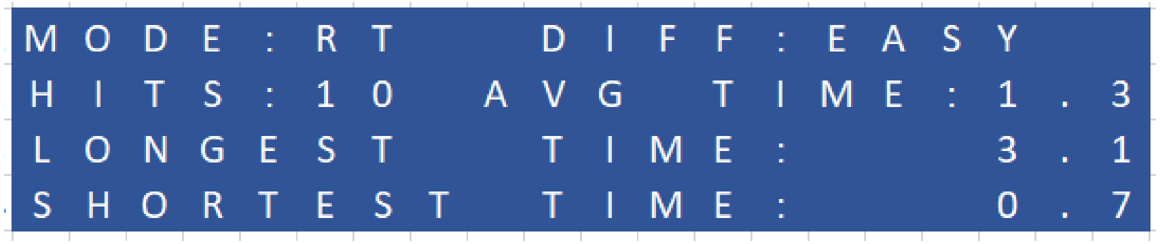


```
MODE:CG  DIFF:EASY
HITS:7
TARGETS:10
HIT RATIO:0.70
```

Figure 37: Combination Generator results display message sample

Reaction Time Mode

In the reaction time mode, the results received are the number of hits, the average time per hit (in seconds), the longest hit time (in seconds) and the shortest hit time (in seconds). The DDRAM address is set to 0x40, and the data sent is “HITS:## AVG TIME:X.X”. Next the DDRAM address is updated to 0x14. Note that this must happen, as row 3 position 1’s address is not next once the entire row 2 is used. The data sent is now “LONGEST TIME: X.X”. The DDRAM is then set to 0x54, and the data sent is “SHORTEST TIME: X.X”. Each numerical value must be parsed to extract each digit which corresponds to the character code. An example is shown below.

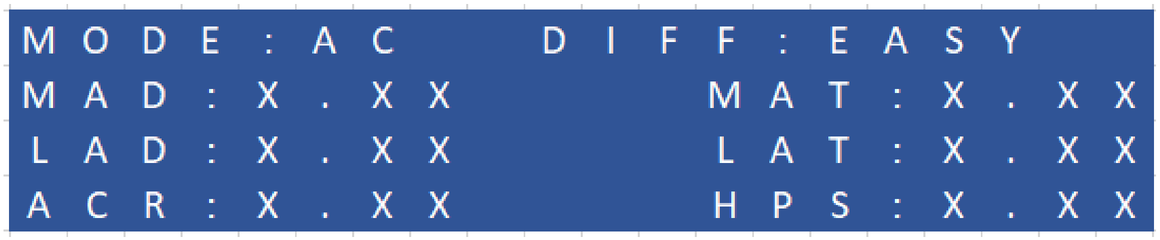


```
MODE:RT  DIFF:EASY
HITS:10 AVG TIME:1.3
LONGEST  TIME:   3.1
SHORTEST TIME:   0.7
```

Figure 38: Reaction Time Mode results display message sample

Accuracy Mode

In the accuracy mode, the received results include most/least accurate distance, most/least accurate time, accuracy rate, and hits per second. Because there are six total results and a limited number of characters on the display, the categories are abbreviated, where MAD/MAT is the most accurate distance and time, respectively. LAD/LAT is the least accurate distance and time. ACR is the accuracy rate, and HPS is the hits per second. First the DDRAM address is set to 0x40 and the following data is displayed: "MAD:X.XX   MAT:X.XX". Next, the DDRAM address is set to 0x14 and the data sent to be displayed is "LAD:X.XX   LAT:X.XX". Next, the DDRAM address is set to 0x54, and the data sent to be displayed is "ACR:X.XX   HPS:X.XX". Each place holder (X) is the character representation of each digit of the desired result. An example is shown below.

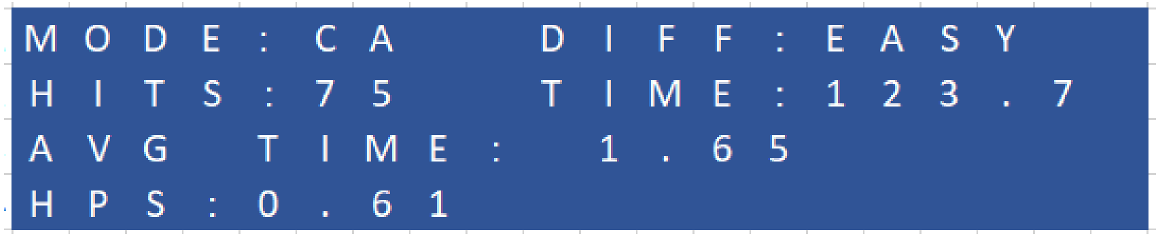


```
MODE:AC   DIFF:EASY
MAD:X.XX    MAT:X.XX
LAD:X.XX    LAT:X.XX
ACR:X.XX    HPS:X.XX
```

Figure 39: Accuracy Mode results display message sample

Cardio Mode:

In the cardio mode, the received results include number of hits, total time, average time per hit and hits per second. First the second row's data is sent by first setting DDRAM to 0x40 and then using the write command and sending the following sequence "HITS:XXX TIME: XXX.X". Because the values can be from 1 digit to 3 digits, each character code will depend on the length of the data. Next, the DDRAM is updated to 0x14, and the write command sends "AVG TIME:XX.XX". Lastly, the DDRAM is updated again to 0x54 and the data sent is "HPS:X.XX".



```
MODE:CA   DIFF:EASY
HITS:75   TIME:123.7
AVG TIME: 1.65
HPS:0.61
```

Figure 40: Cardio Mode results display message sample

LCD Communication

The desired method for the MCU to communicate with the LCD display is to implement the I2C protocol. By using I2C, the MCU only requires the use of two pins, the serial data line and the serial clock line (SDA and SCL). To begin, a function is called to initialize the I2C pins. First, the two pins must be diverted to use the I2C functionality by configuring the pin select lines. Next, the two control registers for I2C must be configured to allow the MCU to act as the master, set the baud rate, select the clock frequency, select the clock divider, set the slave device address, enable the transmit buffer interrupts, and more.

I2C

Within the eUSCI_Bx Control Word Register 0, specific bits are enabled to change the functionality of the eUSCI. First the control register must have its least significant bit equal to 1 (UCSWRST). This allows the other bits of the control register to be modified. Once done, the UCSWRST must be set back to 0.

Next, the following masks are used to enable certain bits of the word control register. The mask UCMST added with UCMODE_2 added with UCSSEL_2 added with UCTR sets the MCU as the master device that transmits with I2C functionality and selects its clock as the submaster clock. When the master device is ready to send the start signal, the word control register is masked with the start signal (UCTXSTT). When the master is ready to stop sending data, the control register is masked with the stop signal (UCTXSTP).

Next, another function is created to transmit data from the master to the slave. The master sends the start sequence, UCTXSTT and sends the slave address with the least significant bit set to 0. This tells the display that the master is writing to the data lines. Upon acknowledged, the master begins to transmit the desired data: either control commands or data write commands. Before the MCU places more data into the transmit buffer, it must wait for the transmit buffer interrupt flag to go down. This means that the transmit buffer is emptied after the data in the buffer has been sent out. Once the master is finished sending data, it transmits a stop signal, UCTXSTP, and the function returns. This transmit function must be called for every sequence of data that is requested to be displayed.

GPIO Pin Connection

In the event that the I2C protocol is unable to be successfully be used to display data on the LCD, 6 additional GPIO pins can be connected directly to the LCDs pins to control Rs, R/W and D0 – D7. However, by controlling these registers directly, a delay must be incorporated for the LCD chip to read these registers. The HD44780 datasheet lists the minimum required delay for each type of command. Each command requires at least 37s except for the return home command, which requires 1.52ms. A simple delay loop function can be called each time a delay is needed.

The GPIO pins used to control Rs, R/W and D0 – D7 must be configured as output pins, by masking them with their corresponding bit (P1DIR |= BIT0). Next, to set a bit to high, the P1OUT control register is masked with the bit (P1OUT |= BIT0). To set the bit to low, the control register is masked with the inverse of the bit (P1OUT &= ~BIT0).

Zigbee Communication Configuration and Functions

Zigbee Configuration

The base system and UI system communicate using the IEEE 802.15.4 protocol. Both MCUs (in the base and the UI system) will be connected to a Zigbee device (XBee chip). The two XBee chips communicate over a wireless personal area network (WPAN). The WPAN must be configured in each XBee before they can be used in the system. Within the network, there are three roles of an XBee chip, a router, an end device, and a coordinator. Each network must have one and only one coordinator. The coordinator XBee can communicate with all end devices within the network (acting as a router) as well as establishing and maintaining the network (allowing specific devices to join). The end device XBee can only communicate with a router (or coordinator, as the coordinator has the functions of a router). Because our design only incorporates two devices that communicate, it does not matter which device is the coordinator.

XBee device is technically composed of two devices, the antenna and the explorer. The explorer is connected to the MCU using serial data lines (UART), and then the explorer is placed on top of the antenna (aligning the pins). When the MCU sends data through the antenna, it uses the transmission (TXD) pin which is connected to the IN pin on the explorer. The receive data, the MCU reads the received pin (RXD), which is connected to the OUT pin on the explorer. Lastly, the explorer is powered using the 3.3V pin from the MCU. The ground pin is connected to one of the ground pins of the MCU.

In order to configure the two XBee devices, a program called XCTU. XCTU is a free program that allows developers to interact with general digital RF modules (like Zigbee) in a simple, easy to use, graphical interface. XCTU also helps test the communication between devices within the network. The XBee explorer comes with a micro USB port, to let the developer connect the XBee directly to a computer to configure the WPAN.

Coordinator & End Devices Configuration

To set up one XBee as the coordinator, the explorer must be attached to the antenna. On the computer, open the XCTU program and plug the explorer into the USB port. In XCTU, search the USB port by clicking “Discover Devices” and selecting which port to search. Once the device shows up, select “Add Device”. Then open the configuration settings to configure the role of the XBee. First select a Channel ID and PAN ID. These ID’s must be the same for both XBee devices. Next, select a destination high and a destination low. This allows the coordinator to know which device (if more than one end device) that the data should be routed to. 64-bit, 16-bit and strings are acceptable as inputs. Lastly, enable the device as a coordinator. Once finished, the settings are written to the XBee. It is best practice to physically label which XBee is the coordinator.

Next, plug the next XBee device into the USB port and follow the same instructions, making sure that the channel ID and PAN ID are matching. Instead of enabling this device as the coordinator, the device is enabled as an end device. The settings are then written to the XBee chip.

To test the communication between the devices, plug both XBee's into two different USB ports. Once both devices are added, the console can be opened. When selected on each device, data can be typed into the console. To make sure they are communicating, both devices should echo the same data in both the consoles.

MCU and XBee Communication

Because the MCU and the XBee explorer use UART to communicate, the MCU will need to divert two pins that support the eUSCI communication. Therefore, a function for diverting those pins and configuring the two word control registers will need to be called before any communication can happen between the explorer and the MCU. In addition, the receive and transmit interrupt vectors will also need to be configured, which allows the MCU to transmit/receive the data stored in the transmit/receive buffers.

UI System

Because the XBee explorer is directly connected to the TXD and RXD pins on the MCU, the transmit and receive interrupts will be called when their respective flags are raised. Therefore, there is not a direct function called when data is ready to be sent from the UI system to the base system. Instead, once the select button has been pushed twice, the correct button ISR is called which stores the mode select and difficulty select into the TXD buffers. This action calls the UART ISR that transmits the mode and difficulty selection to the XBee.

Similarly, when data is being received from the XBee chip, meaning the base has sent results to the UI, the MCU exits low power mode from the ISR. The receive buffer is emptied into a register on the MCU, and the flag is lowered. The results are then stored one by one in separate registers, as data is received from the XBee. These are the registers that LCD write function uses to send to the DDRAM data registers for the LCD.

Base System

For the base system, the process is nearly identical. When the UI system has sent data through the XBee network, the receive interrupt is called when the RXD interrupt flag is raised. The data is moved over to an internal register until all data is received. Based on the data received, the base system performs the specific mode/difficulty settings and computes results. Once the results are ready, the data is moved over, one by one, into the TXD buffer, which sends the data to the XBee, which sends the information UI system.

It is important to note that the several status registers must be polled before transmitting data. First, within the UART function, the MCU must check and wait until the transmit lines are available, meaning there is not a current interrupt pending/happening. Next, the MCU must also check and make sure the XBee is available and not currently receiving or transmitting data. If both of those conditions are met, then the system may send data to the buffers.

Button Configuration and Functions

Button Configuration

The buttons on the UI device and at the bottom of the base are used as the user inputs. When each button is pressed, an ISR is called to handle the specific action that must be completed. The button ISRs are using the majority of the created functions to perform the necessary actions, such as updating a specific register, writing data to the LCD, or initiating serial communication to XBee. Depending on which pins are used as button inputs, the corresponding port vector will be used to handle the ISR. Several pins shar a port vector, therefore each time the ISR is called, the program must check which pin had raised an interrupt flag.

As stated before, the pins assigned to the buttons will need to be configured as inputs, by masking the pin with the specific bit. For example, to have P1.0 be used as an input, the direction must be set to logic 0 (P1DIR &= ~BIT0). This will have to be done for each pin that is acting as an input to the system (in the UI and the base). Next, the pins will have to also be configured for how they are used as inputs, for example, as a pull down or pull up. For our case, each button is a pull up, meaning when the button is pushed, the circuit is pulled up to high, closing the circuit. The resistor must also be enabled for this pin to work. Next, the interrupt must be enabled, so that when the flag is raised, the port's ISR is called.

Lastly, the buttons must incorporate a debouncing method. When the button is pushed, the MCU may interpret several pushes, due to the instability at the edges of the push. Therefore, the buttons interrupt must be disabled for a short duration, so that it is not called during these phantom pushes. The duration should be made to be around 20ms. To implement this, a basic for loop can be used to wait out the duration.

Instead of the program polling to check if the button has been pressed each clock cycle (by checking the interrupt flag), the program will use interrupt service routines (ISRs) to perform specific actions. This allows the MCU to sit idle while the user is not pressing any buttons and keeps the pipeline free of instructions. Another benefit is that the code size can be reduced slightly, as each button would require a function to poll the interrupt flags.

Button Functions (ISRs)

Mode Select Button ISR

When the UI system first turns on, the first button that is enabled in the main is the Mode Select Button. When this button is pushed, the mode select register is incremented, until it reaches the last mode, and then loops back to the beginning value. The mode select register must be set at mode 4 within the main; when the mode button is pushed and the ISR is called, the next increment will force the mode select register to go to 0, so that the first mode is displayed (combination generator). This is also the register that is sent to XBee (and to the base) to determine which mode sequence is selected.

Within the ISR, the failsafe timer is reset, and the clear display function is called, to clear the welcome message. This also sets the cursor to position 1 or the first row. Then based on the mode select register, the write to DDRAM function is called a specific amount of times to print the first mode. The ISR exits and returns to the main and waits for another user input. Every time the mode select button is pressed, this ISR is called and these actions are repeated.

A counter will be used to determine if it is the first time the mode button is pushed, so that the clear display function is not called every time. This will also determine the DDRAM address that is sent to move the cursor to the correct location to print only the mode to the screen. Also note, that the failsafe timer is reset for every press, not just the first press.

Within the mode select ISR, the select button is enabled, so that the user is able to select a mode and move forward with the next selection (difficulty setting). The cancel button is also enabled so that the user can return to the welcome screen at any time.

Difficulty Select Button ISR

After the user has selected a mode, they move on to the difficulty selection (meaning the difficulty select button is enabled from the select button ISR). The program will call the difficulty select button's ISR each time the user presses the difficulty button. The ISR is similar to the mode button ISR, as each time it is pressed, the difficulty select register is incremented (and looped) based on the value of the mode select register. The failsafe timer is also reset for every press. The ISR will also call the set DDRAM address and write to DDRAM functions to print the specific difficulty message, depending on the mode selected and the value of the difficulty select register. Then, every time the difficulty button is pushed, the ISR repeats itself, and performs the correct function based on the new updated values.

Because this ISR is never calling the clear display function, a counter does not need to be used. However, similar to the mode button ISR, the beginning value of the difficulty select register will also need to be at the last option, so when the button is pushed, the first difficulty selection that appears is "Easy".

Select Button ISR

The select button is used for two different applications. When the mode button is enabled and pressed for the first time, its ISR enables the select button interrupt. The select button performs different tasks based on the number of times it has been pressed. Therefore, a counter is used to determine which tasks must be done. This is basically keeping track of what stage the user is at when using the UI system.

Once the user has made their mode selection and presses the select button for the first time, the counter is increased by 1. Next, the mode select button enable interrupt is turned off, so that when the user accidently presses the Mode Button, the mode button ISR is not called. Next, the difficulty select button interrupt is enabled based on the mode select register. This allows the user to use the difficulty select button to cycle through the options, if applicable. The failsafe timer is also reset each time the difficulty button is pressed.

However, if the mode is in the ambient mode, meaning the difficulty button is disabled, the timer will not reset. Next, the set DDRAM address function is called to move the cursor to position 1 of row 2, and prints the category title, only if the mode select register is set to 0 through 3. This is where the difficulty category is printed, by calling the write to DDRAM function. This allows the difficulty button ISR to only need to set the DDRAM address at the position where the difficulty type will be printed.

Once the user has pressed the select button twice, meaning they have made their difficulty setting selection, the ISR is called again and increments the counter. When this counter is set to 2, the interrupts for the mode, difficulty and cancel buttons are disabled, and the failsafe timer interrupt is also disabled. The ISR then moves the required data (mode select register value and difficulty select register value) into the UART transmit buffers, one at a time. Before the buffer is written to, the ISR checks to see if XBee is available and the transmit buffer flags are down. Then the ISR calls the set DDRAM address and writes to DDRAM to print the Start message on the fourth row.

After a suitable amount of time, the clear display function is called. Lastly, the ISR exits and enters a suitable low power mode that waits until the receive buffer interrupt flag is raised which calls the receive buffer interrupt. The mode and difficulty select registers are not reset to their initial values, as they are needed when the results are displayed.

Cancel Button ISR

As stated before, the cancel button is used to return to the welcome screen. This allows the user to “start over” with the mode and difficulty selections. The cancel button interrupt is enabled when the mode button is pressed for the first time and is disabled when the select button is pushed for the second time. Because the cancel button ISR performs the same action each time it is pressed (not mode dependent or status dependent), no counters will be necessary. The first action performed is checking if the mode button interrupt is enabled. If not, it is enabled. Next the difficulty button and select button interrupts are disabled. Next, the mode and difficulty select registers are reset to the initialized value, as well as the mode and select button counters. Next, the clear display function is called, and the welcome message is called. This clears the display and prints the welcome message for the user, which instructs them to press the mode button. Lastly, the mode and difficulty select registers are set to their initial values, as well as the mode and select button counters.

The cancel button interrupt is enabled again once the MCU has received data from XBee. The failsafe timer is also started, and the interrupt for the timer is also enabled. This is because the user will require a way to return to the welcome screen once they have reviewed their results.

Start Button ISR

The start button is located at the bottom of the base system. While the user is interacting with the UI system, the base system is in a low power mode. The base system exits the low power mode when the receive buffer interrupt is called and moves the data from the buffer into the appropriate registers to hold the mode selection and difficulty selection. Once all data has been received, the start button, early exit button and failsafe timer interrupts are enabled.

When the user pushes the start button, the ISR is called. The mode select and difficulty select registers are read to determine which mode function that is called. The easiest way to implement this, is to have a switch case statement to determine the mode function, where the mode function takes the difficulty select register value as the argument. Depending on the mode function, various LEDs and sensors are enabled/read and specific timers are used. Note, that the timers used inside the mode functions also have interrupts that are enabled/used. Before the mode functions are called, the ISR disables the start button interrupt, so avoid restarting the mode.

The start button also disables the failsafe timer of the base. The failsafe timer is implemented to avoid the system remaining stuck in a portion of the program because it is waiting for the user's input. Therefore, once the start button is pressed, the modes timers are used instead, so the interrupt for the failsafe timer is disabled.

Early Exit Button ISR

The early exit button is used to quit the workout session and skip to calculation the results and send the data to the UI. When the user presses the early exit button, its ISR is called. Within the ISR, the early exit button interrupt is disabled, so that it cannot be called multiple times. Next, the results function is called, based on the mode select register value. Once the result functions are finished, the results registers are moved into the transmit buffers, to send XBee (and the UI) the requested data. Upon exiting the ISR, the MCU returns to the low power mode and waits for the receive buffer interrupt to be called.

Timer Configuration and ISR

The UI and base system implement timers. In the UI system, a failsafe timer is used and in the base system a failsafe timer and various mode function timers are used. Because our design is dependent on receiving inputs from the user to proceed in the next step in the program, failsafe timers can be implemented to either move on to the next sequence (in the base system) or turn the system off (in the UI system).

When choosing which pin is diverted to be used as the failsafe timers, the ISR priority list must be referenced. In the event that multiple ISRs are called at the same time (unlikely, but possible), the MCU has a list of which ISRs have higher priority. The failsafe timers should have a lower priority than the button ISRs. This is because the design should move forward if the button is pressed as the timer goes off. This is partly due to the fact that when the buttons are pressed, the failsafe timers are renewed, meaning the interrupt is temporarily disabled and the interrupt flags are lowered. Then the timer's interrupt is re-enabled, and the timer is started. The other part is due to the fact that the design is waiting on the user to press a button. Therefore, when the user does press it at the same time as the timer ISR is called, it would not make logical sense to shut down (sleep).

Timer ISR

For the UI system, the failsafe timer's ISR is called if the timer is able to reach its end. Because each button ISR renews the timer, the timer ISR is only called if the user chooses to no longer select choices to begin a workout session. Therefore, when the failsafe timers ISR is called, the first function called is the clear display function. Then the display control command is called to turn the display off. Next, all button interrupts are disabled. The only way to turn the system on again, is to use the on/off sliding switch. First slide the switch to the off position to disconnect the power. Then slide the switch to the on position to power the device back on. This allows the MCU to begin at the top of the main, allowing all systems to re-initialize its systems.

For the base system, the failsafe timer's ISR is used to prevent the program from waiting for the user to press the start button. In the event that the failsafe timer's ISR is called, all button interrupts are disabled, the results functions are executed and the results registers are moved into the transmit buffers, to send XBee (and the UI) the requested data. Upon exiting the ISR, the MCU returns to the low power mode and waits for the receive buffer interrupt to be called. This allows the program to pick up at the UI system to print results (if any). If the user still has not interacted with the device, the UI system's failsafe timer will activate and send the UI system into a sleep mode.

Mode Functions and Sensor/LED Configuration

General Mode Functions 0-4

When the base has detected that the user has pressed the start button, the mode function (dependent on the mode select register) is called. The mode functions each begin by starting the first timer, which controls the total length of the session. Depending on the mode, the first timer can be fixed, or variable (dependent on the difficulty select register). Next, the sensor and LED configuration functions are called. the second timer is configured and started. The rest of the mode function is controlled by interacting several ISR's (timer and sensor) to move through the session. The session ends when the first timer's ISR is called, which turns the LEDs and sensors off and continues to the results sections.

LEDs

The LEDs used to outline the target areas are controlled by the WS2811 driver. The driver is connected to the MCU over SPI to control the LED group, the LED brightness and the LED color. The color is controlled by sending 24 bits of RGB values (255,255,255). The brightness is controlled with 8 bits (255) and the LEDs are grouped into 3 (3 LEDs per group). During the mode function, these are the values that will be controlled to either turn on a target area or turn off a target area.

LED Functions

For each mode function, the communication between the MCU and the LEDs will need to be configured, by calling another function that sets up the SPI interface. This includes controlling which module is the master and slave and the interrupts required. Next, each mode function will turn the LEDS on/off from a timer's ISR or the sensor's ISR (binary or IMU). To turn the LEDs on, the brightness will be turned up, and the color will be set with a RGB value. To turn them off, the brightness will be turned all the way down, and the color will be set to white (0,0,0).

Sensors

The sensors will be used within the mode functions, to retrieve data as the user interacts with the workout bag. The combination generator mode function reads from the binary sensors located at the center of each of the target areas on Side A. The accuracy, reaction time and cardio modes functions read from multiple IMU sensors that are located in various locations inside the target area. Because the ambient mode only uses the LEDs, the ambient mode does not read from any sensors. Inside each mode function, specific sensor configuration functions are called to configure the sensors in a particular way that is best for the mode. At the end of each mode function, another sensor configuration function is called that is used to turn off the sensors and clear any saved data.

Binary Sensor Functions: Combination Generator

The binary sensor reads two types of values, 0 and 1. The sensor will be configured as a pull up resistor, similar to a button. When the user hits the bag, the sensor pulls up to high voltage, reading logic 1. Because the pins connected to the binary sensor are configured as an input with a pull up resistor, when the pin goes to logic 1, the interrupt flag is raised and the sensor's ISR is called. Within this ISR, the hit counter is incremented by 1, clears the interrupt flag and continues within the mode function. When the target is not hit, the pin remains at logic 0, indicating no hit is happening.

IMU Functions: Side B

The IMU sensor used is the LSM6DSR, which communicates the MCU to send data bytes and read configuration bytes. Therefore, within the accuracy mode, creation time mode, and cardio mode functions, the I2C configuration function will be called first. LSM6DSR also supports communication using SPI as well. In the event that I2C is not the type of communication used, a SPI configuration function will be called instead. Each type of configuration and data return types are explained for each mode. In general, when the IMU has data to be sent (meaning FIFO size is greater than a set threshold), the device, an interrupt is called internally and transmits the contents of FIFO to the MCU over I2C. The contents in FIFO are then dumped and the process repeats.

Reaction Mode & Cardio Mode

Within the reaction mode and cardio mode functions, the sensors module is configured to read from the accelerometer and the gyroscope. In the reaction mode, a hit is determined if the IMU sensors have detected movement. The threshold of what amount of movement is considered a hit will be determined after several rounds of testing. To turn the accelerometer and gyroscope modes on, the control registers CTRTL1_XL and CTRL2_G are configured to active by writing ODR_XL[3:0] into CTRL1_XL and ODR_G[3:0] into CTRL2_G. Next, the mode type of the accelerometer and gyroscope are configured to the high performance mode. To do this, the XL_HM_MODE and G_HM_MODE bits are set to 1 within the CTRL6_C and CTRL7_G control registers.

In the reaction time mode and cardio mode, the sensors detect movement (linear and angular acceleration) and the values are in the appropriate output register. As data is entered, an interrupt can be enabled to detect when new data is available. This new data interrupt will be used to detect if a hit occurred, since data will be entered into these registers. For the reaction time mode, the new data interrupt will turn off the target LEDs, stop the secondary timer, record its time, and reset the timer. A delay will occur, based on the difficulty setting, and then turn on the target area LEDs, and start the timer. Within the ISR, the hit counter will also increment. In the cardio mode, the new data interrupt will only increment the hit counter. The target LEDs stay on, and the one timer continues. Once the session timer (first timer) ISR is called (meaning the round is over), the sensors are configured back to power-down mode, so that the sensors are not consuming power. Because both modes collect the same data, the same function can be called.

Accuracy Mode

Unlike the reaction time and cardio modes, the accuracy mode uses the sensors to find the location of the hit on the target area. The accuracy mode also only uses the accelerometer, configured in the active and higher performance mode. There are several IMU sensors located within the target area, where some are in the outer ring, and some are in the inner ring. When a hit occurs, the sensor's closet will have a higher acceleration than sensors further away. By reading the direction and magnitude of the sensor's acceleration and knowing where the sensors are located within the target area, an approximation can be made of where the hit landed, relative to the center of the target area.

The data required from the sensors will include reading from the x, y and z outputs of the accelerometer of each sensor. The sensors module will send all three registers for each sensor to the MCU. The MCU will use these during the results computation function to derive the sensor that was closest and find the approximate radius of the hit from the center. A hit is detected when these registers are written to, which calls an interrupt to send the data back to the MCU, stop, store and reset the timer, increment the hit counter and disable the target area LEDs. Once the first timer’s ISR is called, the sensors are powered down, using another configuration function.

Results Calculation Functions

Once the mode function is completed and the program returns to the main, the next function called is the results function, based on the mode select register. The results function uses pointers to access the stored data that was collected throughout the mode function. Each modes' results function is explained below. After each result is computed and sent, the registers are then cleared.

Note that Ambient mode does not have action to be taken, so no results need to be calculated upon mode completion.

Combination Generator Results

During the combination generator mode function, the hit counter and target counter registers are accessed and calculated. The hit counter and target counter values are copied into the byte 1 and byte 2 registers. Byte 3 register value is found by dividing the hit counter value by the target counter value. When the program is ready to send the results data back to the UI system, byte 1, 2 and 3 registers are loaded into the transmit buffer, one at a time. It is important to note that the base system must send them in the same order that the UI expects to receive them.

Reaction Time Results

During the reaction time mode function, the hit counter is calculated and an array is filled with each hit's timer value, where each index indicates the hit number and the value at that index is that hit's time. The hit counter value is copied into byte 1 register. Byte 2 register's value is found by finding the maximum value within the timer array, and byte 3 register value is found by finding the minimum value within the array. Lastly, byte 4 register is found by summing the time array, and dividing by hit counter. Then, byte 1-4 registers are moved into the transmit buffer, to send the data back to the UI system.

Cardio Mode Results

The cardio mode function of the hit counter is calculated during the cardio mode function, and the timer value is stored at the end of the mode. In byte 1, the hit counter value is stored. In byte 2, the one timer's value is stored. In byte 3, the time is divided by the hit counter, to give the average hit time (seconds per hit). In byte 4, the hit counter is divided by the time to give the hits per second value. All byte 1-4 are then stored into the transmit buffer, and then cleared.

Accuracy Mode Results

In the accuracy mode function, the hit counter is calculated and the time array is filled, where each index stores each hit’s time. The accuracy mode function also returns data from each sensor. The data returned fills each sensor's x, y and z acceleration values, which get stored into each sensor's x,y and z acceleration array, where each index stores each hits x,y and z acceleration value. For every index, the values of each sensor's x,y and z array is compared to find the maximum value. The sensor that had the highest value is considered the closest to where the hit occurred. Because each sensor's location is known, with a predetermined radius to the center, the distance array will be filled with the sensor’s distance from the center. Next, each index is examined if the distance is greater or less than the threshold (set by difficulty level). If it is, an accurate hit counter is incremented. Next, byte 1 is filled with the maximum of the time array. Byte 2 is filled with the minimum of the time array. Byte 3 and 4 are filled with the maximum and minimum of the distance array. Byte 5 is filled with the accurate hit counter divided by the hit counter. Lastly, byte 6 is filled with the hit counter divided by the sum of the time array. Bytes 1 through 6 are moved into the transmit buffer and then cleared.

Device Test Plan

Side A: Binary Sensors

To test the sensors built for this device, they will be pressed repeatedly and at random times. The MCU will be coded based on the results of the procedure. The MCU should turn on the internal light for testing purposes, and this testing will be replaced by the actual code when the testing aspect is done for all the binary sensors. The internal light will be a confirmation that the sensor is working.

Side B: IMU Target

For testing, we shall set up a pendulum type of system, with a ball of known weight suspended above the device by a string of length L and tester. The ball will be pulled back a known distance D from the side of the bag, then released so that it will hit the IMU target area with a known force and direction. This will provide consistency in the test setup to make sure the system is being receptive and giving us the same values for every hit.

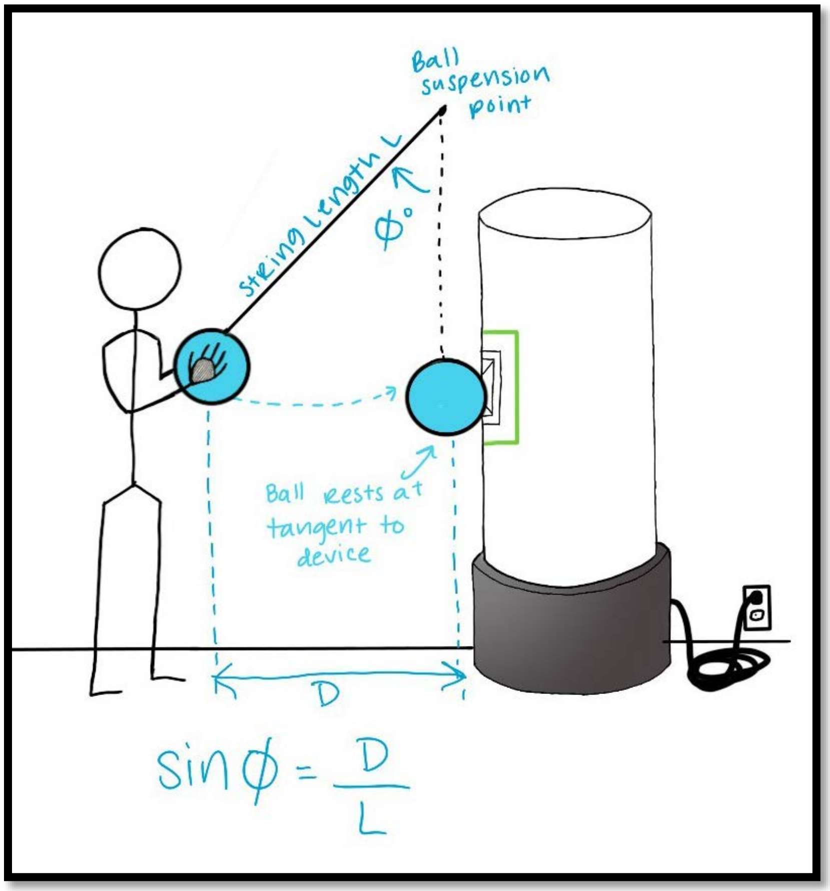


Figure 41: Visual Representation of the testing for side B (front view)

It is also possible to use this setup to test hits made to a point other than the direct center so that calibration of the sensors to a non-ideal hit is possible. To test hits above or below the center of the target, the length of the string, L, can be altered so that it is longer or shorter than the necessary length to reach the center of the device. To test hits to the left or right, the tester can position themselves at an offset distance to the left or right of the target as seen in figure X.

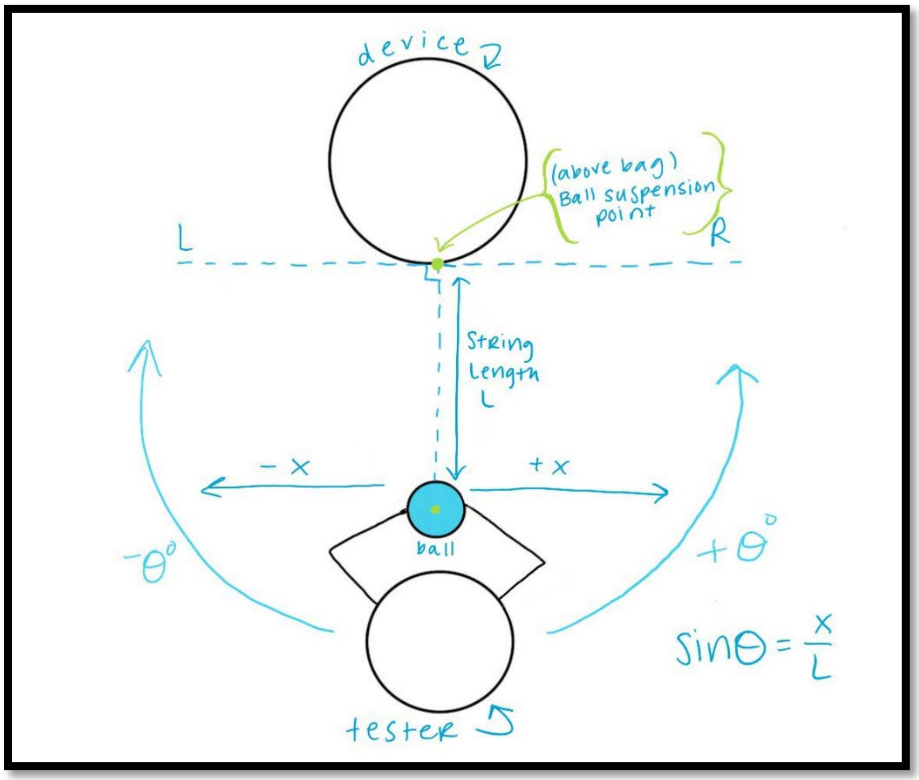


Figure 42: Visual Representation of the testing for side B (top view)

For the final stage of testing, the users will be able to make real contact with the bag so that we are able to see how the sensors will react to real stimuli that is not as predictable as the test setup.

User Manual

This section will include information that is vital to the User Manual that will be provided with the device, including user considerations for safety and directions for use.

General Safety Considerations

This equipment can be used by individuals of 12 of age or older(ASTM F2276-10(2015)). This device should be used with enough space for the user to move around the equipment, to interact freely with it, and to avoid hurting others around them. If the user has medical conditions, they should ask their doctor before performing any type of interactions on the equipment. This device can be interacted by the user if they are wearing gloves or by their own bare hands. If the user becomes disoriented while interacting with this equipment, they must stop immediately and ask for help.

This device is required to be used in an indoor setting or environment (ASTM F2276-10(2015)). The user must interact with the screen to make any selections on the desirable mode and difficulty the user wants to interact with.

Please read and follow the directions for proper use of the device.

Directions for Use

1. Turn on the device's UI Handheld. The Welcome screen will display, asking you to select your mode.
2. Use the mode button to cycle through the mode options until you find the desired mode. Press select to continue to difficulty.
3. Use the difficulty button to cycle through the difficulty options until you find the desired option. Changing difficulty may affect the length or speed of the mode, depending on the activity required. Note that if Ambient mode is selected, there will be no difficulty options to select from. Press select to continue to confirmation.
4. Place the UI Handheld in a safe and secure place, such as the pouch included on the side of the device. It will not be needed until completion of the mode.
5. Position yourself on the correct side of the bag for your chosen mode. Combination Generator Mode will be on side A while the Accuracy, Reaction Time, and Cardio Modes will be on side B.
6. Press the start button on your side of the base of the device to indicate that you are ready to start the mode. The light sequence to prepare to begin will play. The mode activity should be started once the light sequence is complete. More information about each mode can be found below.
7. If at any time during the session you wish to end before completion of the mode, press the early exit button to end your workout and receive your results. Otherwise, complete the mode to receive your results.
8. Once the mode has been completed, retrieve the UI Handheld to receive results.
9. Press cancel to return to the welcome screen and choose a new mode, or power off device UI handheld.

Combination Generator Mode

This mode requires use of side A of the bag. Once the mode has started, press the areas that light up before the timer for the area is up. Your ultimate goal is to hit as many correct areas as they can before time runs out. The sensors in each area will count how many times a correct hit was landed on the bag. Once the timer is complete, the handheld device will provide you with the number of correct hits landed, number of total targets, and your hit ratio of correct hits to total targets.

Reaction Time Mode

This mode requires use of side B of the bag. Once the mode has started, the target's indicators will light up as a signal to hit the target. Your ultimate goal is to have the shortest possible reaction time by hitting the target as quickly as possible once the indicator has lit up. The sensors and timers will determine your reaction time for each hit. Once the timer is complete, the handheld device will provide you with the total number of hits you landed, your average reaction time, your shortest reaction time of the session, and your longest reaction time of the session.

Accuracy Mode

This mode requires use of side B of the bag. Once the mode has started, the target's indicators will light up as a signal to hit the target. Your ultimate goal is to hit the target as close to the center as possible for as many times as you can before time runs out. Once the timer is complete, the handheld device will provide you with the distance from the center (MAD) and time taken (MAT) to land your most accurate hit, the distance from the center (LAD) and time taken (LAT) to land your least accurate hit, your average accuracy over the session (ACR), and your hit rate (HPS) in hits per second.

Cardio Mode

This mode requires use of side B of the bag. Once the mode has started, the target's indicators will light up as a signal to hit the target. Your ultimate goal is increase your heart rate by hitting the target as many times as you can before time runs out. Once the timer is complete, the handheld device will provide you with your total number of hits, the overall time of the session, your average time per hit, and your hit rate (HPS) in hits per second.

Ambient Mode

This mode is not an exercise tool, but instead will illuminate the LEDs equipped in a multicolor pattern for the duration of the internal timer. Once the timer is complete, the bag's lights will turn off and the UI handheld will return you to the Welcome screen. If you wish to turn off the lights before the end of the timer, simply press the Early Exit button on the base of the device.

Administrative Content: Design Considerations and Decisions

Initial Milestones and Timing for Senior Design 1 and 2

Components: Decide Processor, Sensors, Indicators, UI System

The prototype should be done by the end of December, and the prototype UI System should be finished by the end of January, to give us enough time to debug the project, and move our components over to the final design. Temperature sensors are optional to the very end to measure the temperature of the sensor grid and to make sure it is not overheating.

Construction: Complete Sensor Grid (Side A) and Target (Side B)

This project will need multiple areas with sensors, so these will need to be completed as soon as possible to be able to incorporate the electrical components, indicators, and the software which depend on the sensors. The deadline for the first sensor grid prototype is beginning of January, and the other three are no later than the end of January.

Prototype Testing: Processors, Sensors, UI design, RF Control

The prototype hardware will be assembled in the month of January, along with the UI and software design completion. Once both the prototype and the UI/Software are complete, the testing phase will commence. During testing, each sensor on side A and side B will need to be configured and monitored individually to make sure all sensors are working properly. A testing apparatus will be constructed that allows for the same impact force levels for every hit, before finally testing the prototype on our own.

Final Build: PCB and Interface Completion

The PCB components should be completed no later than the beginning of March. This will give us time to finalize the project and be sure that all modes that are included are functioning as intended. Once this has been verified, we would like to begin working on additions that will make this device something that will be worth the cost for a customer. One possibility of this is making an extra display on the body of the bag to interface with the user and add another element of interactivity. This will be challenging due to the frequent impacts the bag will face that could damage the screen, but it would allow more elements to be added to diversify the training possibilities from this device alone.

Group Member Familiarity with Project Elements

- **Hannah** is familiar with software since she works in a company with software engineers. She also likes writing code and programming in her free time.
- **Joseph** is familiar with processors and microcontrollers. He has knowledge of the process of ensuring a system is powered sufficiently.
- **Nicole** took an in-depth sensors class, so she is familiar with various types of sensors that we could use for this project.
- In addition to knowledge of typical electrical engineering curriculum, **Natesha** has many years of experience with and regularly uses a self-standing kicking bag since she is a fourth-degree black belt in karate and works as a sensei in her dojo.

With the group's combination of knowledge of electrical engineering principles and real-world experience with all aspects of this project, we should be able to successfully create and develop this device.

Project Challenges and Unfamiliarity

Most challenges that our group will be facing are a result of the CoVID-19 pandemic that we are currently living through. Due to social distancing, we will need to work on our sections separately; as of now we will not be able to get together unless it is to test the device. As of now we are doing most of our meetings and communication via Discord, but once sensor/indicator testing begins, we will likely face difficulty in being sure everyone has what they need to individually test components.

In general, being able to optimize the parts used to cater to all specifications as well as being low cost, accessible, and realistic for the scope of this design prototype will be a challenge. The pandemic has also slowed electrical component production around the world, meaning that we will have to be sure about our designs before ordering many items and begin attaining and assembling components as soon as possible in attempts to avoid roadblocks.

One non-pandemic related obstacle is the size and scale of this project. A self-standing training bag is a large and heavy piece of boxing equipment, so this component should be in a location accessible to all members with sufficient space to work on device assembly and construction.

Budget and Financing

Our group will self-fund this project. We are budgeting to contribute a maximum of $500 USD, divided equally among our teammates. As of now, no equipment aside from standard circuit components (eg. wires, resistors) has been acquired. All parts of this project must be attained at some later point. Table X below depicts estimated costs of each component.

The individual component that would cost the most money is a self-standing kicking bag since it is a big and technical instrument aimed at people who are experienced in this area. However, finding a used bag will lower the cost of this item, and will not make a difference in the quality of the bag as it will need to be changed to incorporate our components regardless of its condition.

Although sensors are generally inexpensive, we would need enough components to detect contact in the large target area so that the device can test accuracy in addition multiple sensors to detect contact in a binary "hit or no hit" manner in all areas of the sensor grid.

Component	Price (USD)	Quantity	Total (USD)
Binary Sensor Materials	$5.00 to $10.00	8	$40.00 to $80.00
IMUs	$5.00	40	$200.00
Processor: MSP430f6459	$7.50	2	$15.00
XBee Antenna Chip	$20.00	2	$40.00
XBee Explorers	$25.00	2	$50.00
Buttons	$15.00	12 in pack	$15.00
Standup Punching Bag	$130.00	1	$130.00
Addressable LED Strips	$21.00	2	$42.00
20x4 LCD Screen	$12.00	1	$12.00
120V AC to 12V DC Conv.	$14.00	1	$14.00
eBoot Mini DC/DC Conv.	$9.00	6 in pack	$9.00
Miscellaneous + Shipping	$20.00 to $30.00	n/a	$20.00 to $30.00
Total			$587.00 to $637.00

Table 25: Expected costs for this project

References

[1] A. D. Morales, M. C. Morales and L. C. Eldridge, Jr., "Interactive systems and methods for reactive martial arts fitness training". US Patent US20110172060A1, 18 11 2010.

[2] ALITOVE Store, "ALITOVE RGB Addressable LED Strip WS2811 12V LED Strip Lights," Amazon, [Online]. Available: amazon.com.

[3] American Heart Association, "Warm Up, Cool Down," American Heart Association, 1 09 2014. [Online]. Available: https://www.heart.org.

[4] Association Connecting Electronics Industries, "IPC 2221B," Techstreet, 01 11 2012. [Online]. Available: https://www.techstreet.com.

[5] ASTM International, "ASTM F2276 - 10(2015)," ASMT International, 2015. [Online]. Available: https://www.astm.org.

[6] D. Hunt, "MSP430F5529LP example code: LCD Counting Example Using a 4-bit Interface," davesmotleyprojects, 2019. [Online]. Available: http://www.davesmotleyprojects.com.

[7] D. Lapkova and M. Adamek, "Using Strain Gauge for Measuring of Direct Punch Force," *XXI IMEKO World Congress "Measurement in Research and Industry",* 2015.

[8] D. Lapkova, L. Kralik and M. Adamek, "Possibilities Of Force Measuring In Professional Defense," *XXI IMEKO World Congress "Measurement in Research and Industry",* 2015.

[9] Digi International Inc, "XBee 1mW Trace Antenna - Series 1 (802.15.4)," Sparkfun, [Online]. Available: https://www.sparkfun.com/products/retired/11215.

[10] Duke Energy Indiana, "Average Duke Energy Electric Bill," Duke Energy, 08 05 2020. [Online]. Available: https://www.citact.org.

[11] Enviorment and Protection Agency, "Regulations, Initiatives and Research on Electronics Stewardship," Enviorment and Protection Agency, [Online]. Available: epa.gov.

[12] F. D. o. E. Protection, "Electronic Waste," Florida Deparment of Enviormental Protection, [Online]. Available: https://floridadep.gov/waste/permitting-compliance-assistance/content/electronics-waste.

[13] Hitachi, "HD44780U(LCD-II)," Hitachi, [Online]. Available: http://site.gravitech.us/MicroResearch/Others/LCD-20x4B/HD44780.pdf.

[14] International Electrotechnical Commission, "IEC 60906-2:2011," Webstore, 12 05 2011. [Online]. Available: https://webstore.iec.ch.

[15] IvenSense, "MPU-9250 Product Specification," InvenSense, 20 June 2016. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf.

[16] InvenSense, "MPU-9250 Register Map and Descriptions," IvenSense, 7 January 2015. [Online]. Available: https://invensense.tdk.com/download-pdf/mpu-9250-register-map/.

[17] J. Lightfoot, "Get Started with XBee – A Beginner's Tutorial," Atomic Object, 18 July 2016. [Online]. Available: https://spin.atomicobject.com/2016/07/18/xbee-tutorial/.

[18] J. T. Adams, "An Introduction to IEEE STD 802.15.4," IEEE, 24 07 2006. [Online]. Available: https://ieeexplore.ieee.org.

[19] J.-P. Rodrigue, "Point-to-Point versus Hub-and-Spoke Networks," Hofstra University, 2020. [Online]. Available: https://transportgeography.org.

[20] Lady Ada, "Overview |EL Wire," Adafruit, 29 07 2012. [Online]. Available: https://learn.adafruit.com.

[21] Longrunner, "for ArduinoIDE, Longruner 20x4 LCD Display Module IIC/I2C/TWI Serial 2004 with Screen Panel Expansion Board White on Blue, 4 pin Jump Cables Wire Included," Amazon, [Online]. Available: https://www.amazon.com.

[22] M. Alves, "The IEEE 802.15.4/ZigBee protocol stack architecture," ResearchGate, [Online]. Available: https://www.researchgate.net.

[23] Raspberry Pi Foundation, "Raspberry Pi 4 Tech Specs," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org.

[24] "SparkFun XBee Explorer USB," Sparkfun, [Online]. Available: https://www.sparkfun.com/products/11812.

[25] ST Microelectronics NV, "Data Sheet iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope," [Online]. Available: https://www.st.com/resource/en/datasheet/lsm6dsr.pdf.

[26] Systronix, "Systronix 20x4 LCD Brief Technical Data," Systronix, 31 July 2000. [Online]. Available: http://www.systronix.com/access/Systronix_20x4_lcd_brief_data.pdf.

[27] T. K. Hareendran, "EL Wire Experiments & A Minor Hack," Electro Schemantics, 02 05 2019. [Online]. Available: https://www.electroschematics.com.

[28] Texas Instruments Incorporated, "66AK2G1x Multicore DSP+Arm KeyStone II System-on-Chip (SoC) (Rev. F)," Texas Instruments Incorporated, [Online]. Available: https://www.ti.com.

[29] Texas Instruments Incorporated, "MSP430F5528," Texas Instruments Incorporated, 11 9 2020. [Online]. Available: https://www.ti.com.

[30] Texas Instruments Incorporated, "ZigBee RF Transceiver Datasheet," Texas Instruments Incorporated, 12 2007. [Online]. Available: ti.com.

[31] The Hook Up, "The COMPLETE guide to selecting individually addressable LED strips," The Hook Up, 5 08 2019. [Online]. Available: http://www.thesmarthomehookup.com.
[32] "Wavemaster," Century, [Online]. Available: https://www.centurymartialarts.com.
[33] yida, "UART vs I2C vs SPI – Communication Protocols and Uses," Seed Studio, 25 09 2019. [Online]. Available: https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/.
[34] yida, "UART vs I2C vs SPI – Communication Protocols and Uses," Seed Studio, 25 09 2019. [Online]. Available: https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/.

Appendix

A) Pinout Diagram of MSP430F5528

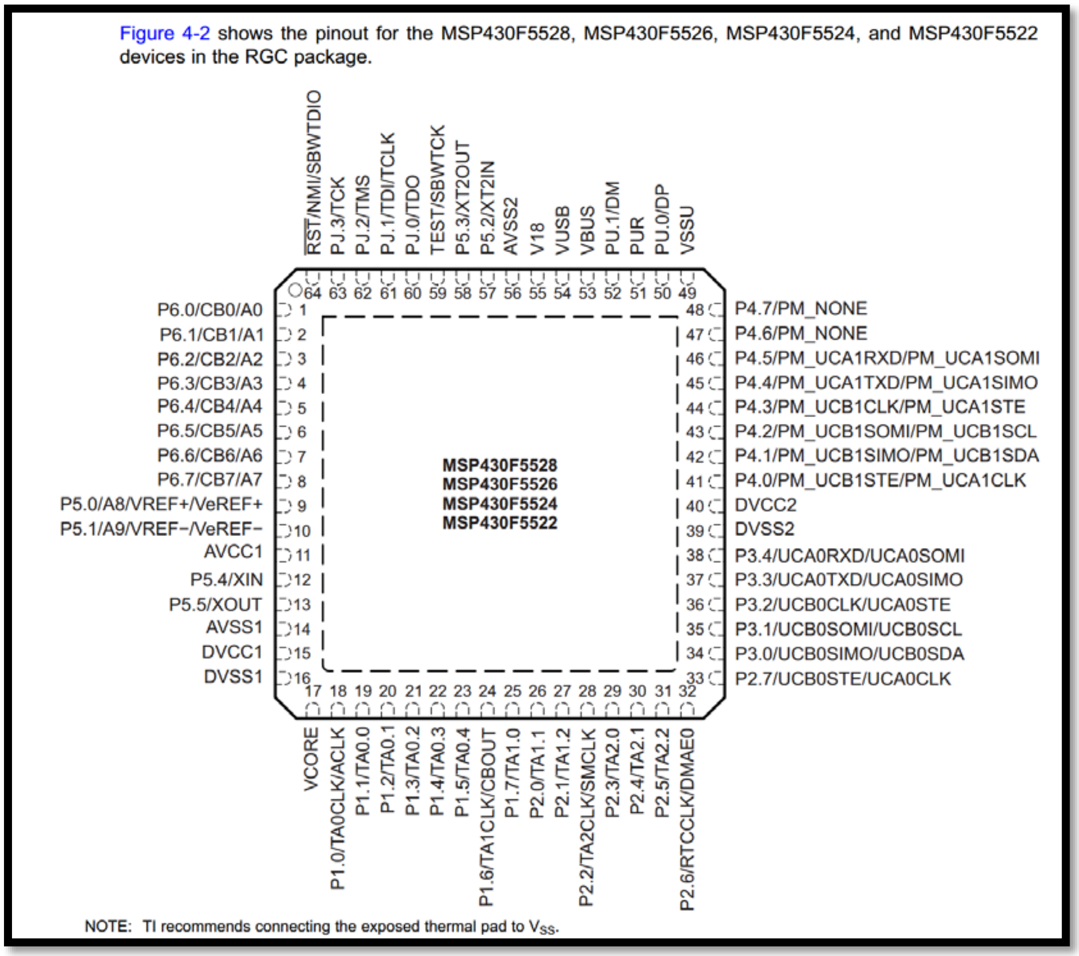


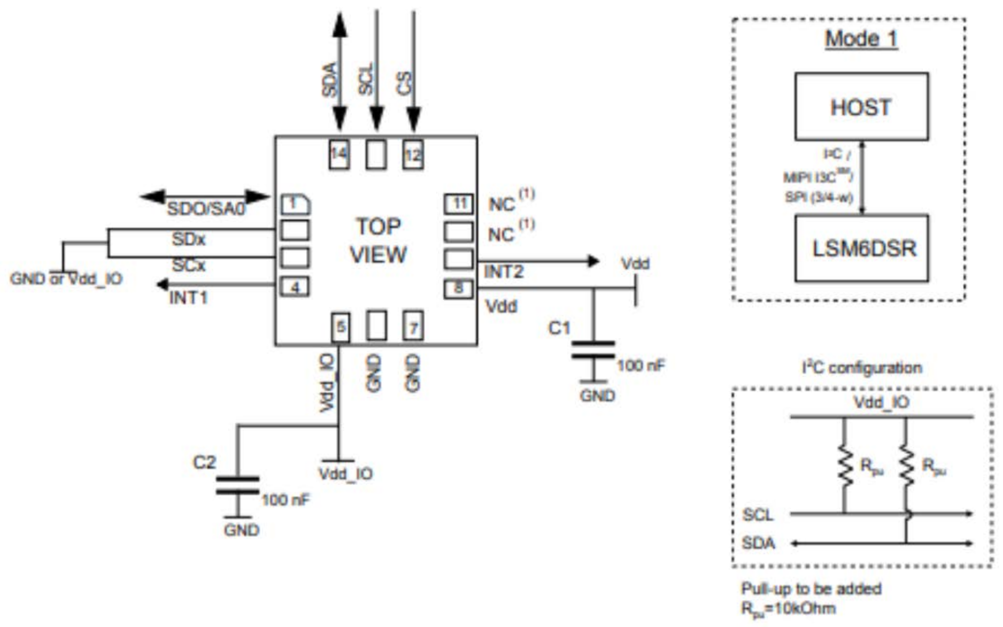
Figure 4-2 shows the pinout for the MSP430F5528, MSP430F5526, MSP430F5524, and MSP430F5522 devices in the RGC package.

NOTE: TI recommends connecting the exposed thermal pad to V_{SS}.

B) Pinout Diagram of LSM6DSR

LSM6DSR electrical connections in Mode 1

Figure 22. **LSM6DSR electrical connections in Mode 1**



1. Leave pin electrically unconnected and soldered to PCB.

The device core is supplied through the Vdd line. Power supply decoupling capacitors (C1, C2 = 100 nF ceramic) should be placed as near as possible to the supply pin of the device (common design practice).

The functionality of the device and the measured acceleration/angular rate data is selectable and accessible through the SPI/I²C/MIPI I3C^{SM} interface.

The functions, the threshold and the timing of the two interrupt pins for each sensor can be completely programmed by the user through the SPI/I²C/MIPI I3C^{SM} interface.

C) Pinout Diagram for MPU-9250

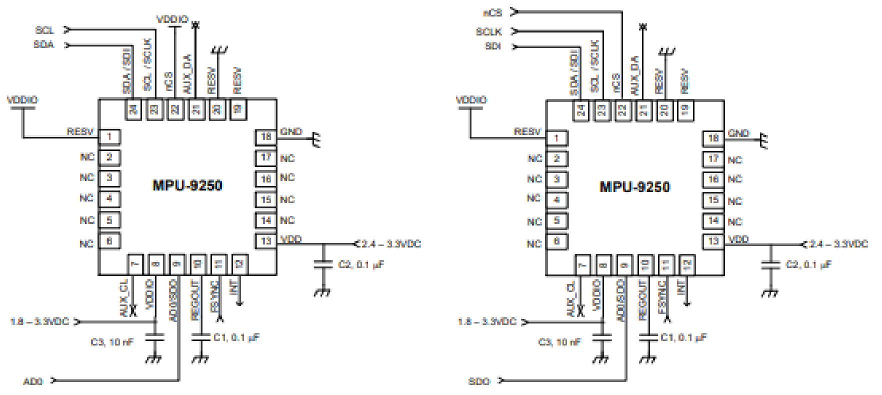


Figure 2 MPU-9250 QFN Application Schematic: (a) I2C operation, (b) SPI operation

Note that the INT pin should be connected to a GPIO pin on the system processor that is capable of waking the system processor from suspend mode.